High-Performance dsPIC33A Core with Floating-Point Unit, High-Speed ADCs and High-Resolution PWM

dsPIC33AK128MC106 Family

Operating Conditions

- 3.0V to 3.6V: -40°C to +85°C, DC to 200 MHz
- 3.0V to 3.6V: -40°C to +125°C, DC to 200 MHz
- 3.0V to 3.6V: -40°C to +150°C, DC to 200 MHz

High-Performance dsPIC33A DSP/CISC CPU

- 32-bit Comprehensive Instruction Set for Optimized Speed and Program Code Size:
  - 16-bit dsPIC33 core compatible
  - Non-paged linear Data/Flash 24-bit addressing space
  - 16-bit/32-bit instructions for optimized code size and performance
- 32-bit Wide Data Paths
- Single and Double Precision Floating-Point Unit (FPU) Coprocessor
- 2 Kbyte Instruction Cache
- Sixteen 32-bit Working Registers
- Dual 72-bit Accumulators Supporting 32-bit and 16-bit Fixed-Point DSP Operations
- Eight Level Deep Working Register Contexts
- Eight Level Deep Accumulator Register Contexts
- Eight Level Deep Floating Point Register Contexts

Memory Features

- Up to 128 Kbytes of Program Flash Memory:
  - 10,000 erase/write cycle endurance
  - 20 years minimum data retention
  - Self-programmable under software control
  - Programmable code protection
  - Flash Error Correcting Code (ECC)
  - Programmable OTP regions
  - Entire Flash OTP by ICSP™ write inhibit
  - 64 x128-bit OTP area
- Up to 16 Kbytes of RAM Memory:
  - 6-channel hardware Direct Memory Access (DMA) module
  - RAM Error Correcting Code (ECC)
  - RAM Memory Built-In Self-Test (MBIST)

Controller Features

- High-Current Sink/Source Capable I/Os
- Programmable Weak Pull-Up and Pull-Down Resistors
- Programmable Open-Drain Outputs
- Edge or Level Change Notification Interrupt on I/O pins
- Peripheral Pin Select (PPS) Remappable Pins to Reduce Board Layout Complexity
- Multiple Interrupt Vectors with Individual Programmable Priority
- Five External Interrupt Pins
- Selectable Oscillator Options Including:
  - 8 MHz, 1% at 0ºC-85ºC Internal Fast RC (FRC) oscillator
  - 8 MHz, 2% Internal Backup Fast RC (BFRC) oscillator with 32 kHz divided output
  - High-speed crystal resonator oscillator or external clock
- Two 1.6 GHz PLLs for Peripheral which can be clocked from the FRC or a Crystal Oscillator
- Reference Clock Output (REFO)
- Low-Power Modes (Sleep and Idle)
- Power-On Reset and Brown-Out Reset

High-Speed PWM

- Four PWM Generators (Four Pairs, Eight Outputs)
- Up to 2.5 ns PWM Resolution
- Dead Time for Rising and Falling Edges
- Dead-Time Compensation Supports Lower Speed Operation
- PWM Support for:
  - BLDC, PMSM, ACIM, SRM and Stepper Motors
- Fault and Current Limit Inputs
- Flexible Trigger Configuration for ADC Triggering

Two High-Speed Analog-to-Digital Converters

- 12-bit Resolution
- Up to 40 Msps Conversion Rate
- Up to 22 Analog Input Pins
- 20 Settings Channels. Each Channel:
  - Supports Discrete Configuration
  - Can be assigned to any analog input (I/O pin or internal signal)
  - Can be set to a different sampling time
  - Can be configured as single-ended or differential
  - Conversion result can be formatted as unsigned or signed
  - Conversion result can be left-aligned (fraction format)
  - Has a separate 32-bit conversion result register
- Supports Four Sampling modes:
  - Oversampling of multiple samples
  - Integration of multiple samples
  - Window (multiple samples accumulated when the gate signal is active)
  - Single Conversion
  - All channels have a digital comparator to detect when the conversion result is less than, greater than, in bounds or out of bounds for the configurable thresholds
  - Three channels support second result accumulator to implement second order filters
- Band Gap Reference and Temperature Sensor Diode Inputs

Other Analog Features

- Three 5 nS Analog Comparators with 12-bit Pulse Density Modulation DACs:
  - Input multiplexing
  - Slope compensation
  - One DAC output buffer
- Three Rail-to-Rail 100 MHz Operational Amplifiers with:
  - 100 V/µS slew rate
  - 1 mV offset (typical)
  - User calibration of input offset voltage
- Four 10 µA Constant Sources + Four Programmable Sources

Peripheral Features

- Three 4-Wire SPI Modules:
  - 4-byte FIFO
  - Variable data width
  - I^2S mode
- Two I^2C modules:
  - Independent Host and Client Logic
  - Supports 100 kHz, 400 kHz and 1 MHz Bus Specifications
  - 7-bit and 10-bit Device Addresses
  - Supports IPMI Standard, SMBus and PMBus
- Three Protocol UARTs with 8-Character RX/TX FIFOs and Automated Handling Support for:
  - LIN 2.2
  - Digital Multiplex 512 (DMX)
  - Smart Card (ISO 7816)
  - IrDA®
- Two Single-Edge Nibble Transmission (SENT) Modules
- One Dedicated 32-bit Timer/Counter
- Four Single Output Capture/Compare/PWM/Timer (SCCP) Modules:
  - Flexible configuration as PWM, input capture, output compare or timers
  - Two 16-bit timers or one 32-bit timer in each module
  - Single PWM output pin
- One Quadrature Encoder Interface (QEI):
  - Four inputs: Phase A, Phase B, Home, Index
- Four Configurable Logic Cells (CLC) with Internal Connections to Select Peripherals and PPS
- Bidirectional Serial Synchronous (BiSS) Encoder Interface with up to Four Client Encoders Support
- Peripheral Trigger Generator (PTG):
  - 10 input trigger sources from other peripheral modules
  - 5 output triggers to other peripheral modules
  - 4 individual interrupt request signals
  - CPU independent state machine-based instruction sequencer

Security Module

- Secure Boot
- Secure Debug
- Immutable Root of Trust (IRT)
- Code Protect
- ICSP Program/Erase Disable (Entire Flash OTP by ICSP Write Inhibit)
- Firmware IP Protection
- Flash Write Protection

Safety Features

- Windowed Watchdog Timer (WDT)
- Deadman Timer (DMT)
- Four I/O Integrity Monitors (IOIM)
- Fail-Safe Clock Monitor (FSCM) with Automatic Switchover to Backup Clock Source with:
  - Programmable over-frequency/under-frequency thresholds
- Flash Error Correcting Code (NVM ECC)
- RAM Error Correcting Code (RAM ECC)
- RAM Memory Built-In Self-Test (MBIST)
- 32-bit Cyclic Redundancy Check (CRC) Module
- Entire Flash OTP by ICSP™ Write Inhibit
- Capless Internal Voltage Regulator
- Virtual PPS Pins for Redundancy and Monitoring
- Temperature Sensor Diode

Functional Safety

Functional Safety Readiness – ISO 26262/IEC 61508/IEC 60730

To learn about the Functional Safety Readiness of this device family and various Functional Safety standards an application can target using this device family, visit www.microchip.com/dsPIC33-Functional-Safety

Qualification

AEC-Q100 REV H:

- Grade 1: -40°C to +125°C
- Grade 0: -40°C to +150°C Planned

Programming and Debug Interfaces

- Three Programming and Debugging Interfaces:
  - Two-wire ICSP™ interface with non-intrusive access and real-time data exchange with application
- Five Program Addresses and Five Full-Featured Breakpoints
- IEEE Standard 1149.2 Compatible (JTAG) Boundary Scan

Targeted Applications

- Power Factor Correction (PFC):
  - Interleaved PFC
  - Critical Conduction PFC
  - Bridgeless PFC
- DC/DC Converters:
  - Buck, Boost, Forward, Flyback, Push-Pull
  - Half/Full-Bridge
  - Phase-Shift Full-Bridge

  - Resonant Converters
- DC/AC:
  - Half/Full-Bridge Inverter
  - Resonant Inverter
- Motor Control:
  - BLDC
  - PMSM
  - SR
  - ACIM
- Advanced Sensor Interfacing
- High-Performance Embedded Control
- Safety-Critical Designs
- Digital Lighting

dsPIC33AK128MC106 Product Family

The dsPIC33A family names, pin counts, memory sizes and peripheral availability of each device are listed in Table 1, and their pinout diagrams are included as well.

Table 1. dsPIC33AK128MC106 Family

Product	Pins	Program Memory (Kbytes)	Data Memory (Kbytes)	General Purpose I/O/PPS	High-Resolution PWM (Generator Pairs)	Two 12-bit ADCs (External Analog Inputs)	Remappable Peripherals								Op Amplifiers	Comparators	12-bit DACs	I^2C	32-bit CRC	DMA (Channels)	Packages
							Dedicated 32-bit Timers	UART	BISS	SCCP(1)	CLC	SPI/I^2S	SENT	QEI							
dsPIC33AK32MC102	28	32	8	19/19	4*2	11	1	3	1	4	4	3	2	1	2	3	3	2	1	6	SSOP/VQFN
dsPIC33AK32MC103	36	32	8	27/27	4*2	15	1	3	1	4	4	3	2	1	3	3	3	2	1	6	VQFN
dsPIC33AK32MC105	48	32	8	35/35	4*2	18	1	3	1	4	4	3	2	1	3	3	3	2	1	6	VQFN/TQFP
dsPIC33AK32MC106	64	32	8	49/49	4*2	22	1	3	1	4	4	3	2	1	3	3	3	2	1	6	VQFN/TQFP
dsPIC33AK64MC102	28	64	16	19/19	4*2	11	1	3	1	4	4	3	2	1	2	3	3	2	1	6	SSOP/VQFN
dsPIC33AK64MC103	36	64	16	27/27	4*2	15	1	3	1	4	4	3	2	1	3	3	3	2	1	6	VQFN
dsPIC33AK64MC105	48	64	16	35/35	4*2	18	1	3	1	4	4	3	2	1	3	3	3	2	1	6	VQFN/TQFP
dsPIC33AK64MC106	64	64	16	49/49	4*2	22	1	3	1	4	4	3	2	1	3	3	3	2	1	6	VQFN/TQFP
dsPIC33AK128MC102	28	128	16	19/19	4*2	11	1	3	1	4	4	3	2	1	2	3	3	2	1	6	SSOP/VQFN
dsPIC33AK128MC103	36	128	16	27/27	4*2	15	1	3	1	4	4	3	2	1	3	3	3	2	1	6	VQFN
dsPIC33AK128MC105	48	128	16	35/35	4*2	18	1	3	1	4	4	3	2	1	3	3	3	2	1	6	VQFN/TQFP
dsPIC33AK128MC106	64	128	16	49/49	4*2	22	1	3	1	4	4	3	2	1	3	3	3	2	1	6	VQFN/TQFP

Note:

1. SCCP can be configured as a PWM with one output, input capture, output compare, 2 x 16-bit timers or 1 x 32-bit timer.

Pin Diagrams

Figure 1. 28-Pin SSOP

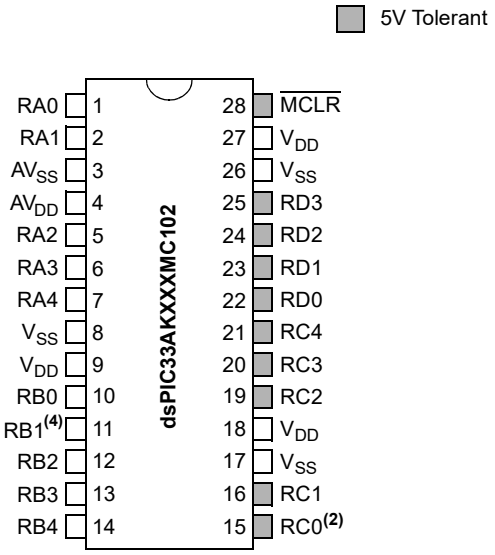


Table 2. 28-Pin SSOP Complete Pin Function Descriptions(1,3)

Pin	Function	Pin	Function
1	PGD2/AD2AN6/CMP3C/ISRC2/IBIAS2/**RP1**/SDA2/IOMF2/RA0	15	OSCO/CLKO/**RP33**/IOMF5/RC0(2)
2	PGC2/DACOUT1/AD1AN7/AD2AN3/CMP1D/CMP2D/ CMP3D/**RP2**/SCL2/RA1	16	OSCI/CLKI/**RP34**/IOMF6/RC1
3	AV_{SS}	17	V_{SS}
4	AV_{DD}	18	V_{DD}
5	OA1OUT/AD1AN0/CMP1A/**RP3**/RA2	19	PGC3/**RP35**/PWM4H/RC2
6	OA1IN-/AD1ANN1/AD2AN0/**RP4**/RA3	20	PGD3/**RP36**/PWM3H/IOMD0/RC3
7	OA1IN+/AD1AN1/CMP1B/**RP5**/RA4	21	**RP37**/PWM3L/IOMD1/RC4
8	V_{SS}	22	**RP49**/PWM2H/IOMD2/RD0
9	V_{DD}	23	TCK/**RP50**/PWM2L/IOMD3/RD1
10	OA2OUT/AD2AN1/CMP2A/**RP17**/INT0/RB0	24	TDO/**RP51**/PWM1H/IOMD4/RD2
11	TMS/OA2IN-/AD1AN4/AD2ANN1/**RP18**/RB1(4)	25	TDI/**RP52**/PWM1L/IOMD5/RD3
12	OA2IN+/AD2AN4/CMP2B/**RP19**/RB2	26	V_{SS}
13	PGD1/AD1AN5/CMP1C/ISRC0/IBIAS0/**RP20**/SDA1/RB3	27	V_{DD}
14	PGC1/AN2AN5/CMP2C/ISRC1/IBIAS1/**RP21**/SCL1/RB4	28	$\overline{MCLR}$

Note:

1. **RPn** represents remappable peripheral functions.
2. This pin has 8x drive strength.
3. Unless otherwise stated, pins are 4x drive strength. Refer to Electrical Specifications for current drive strength details.
4. A pull-up resistor is connected to this pin when device is erased (JTAG enabled) and during programming.

Pin Diagrams

Figure 2. 28-Pin VQFN

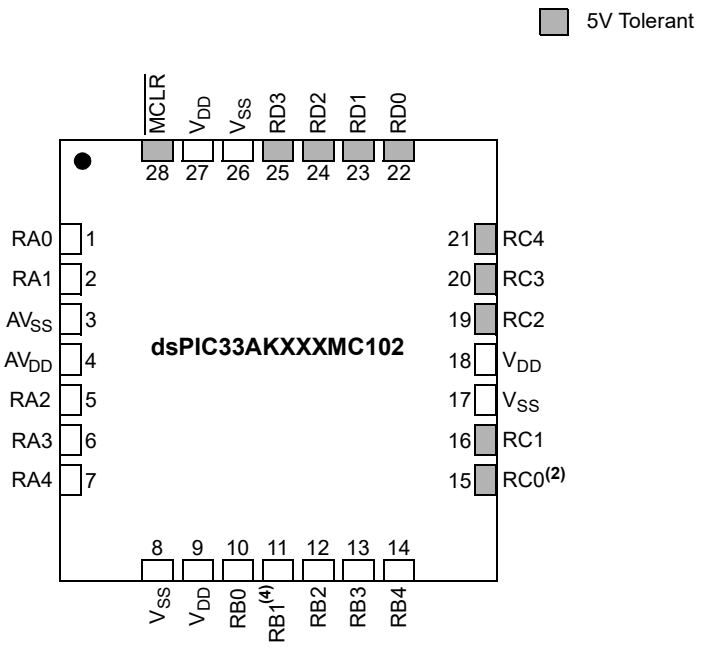


Table 3. 28-Pin VQFN Complete Pin Function Descriptions[1,3]

Pin	Function	Pin	Function
1	PGD2/AD2AN6/CMP3C/ISRC2/IBIAS2/**RP1**/SDA2/IOMF2/RA0	15	OSCO/CLKO/**RP33**/IOMF5/RC0[2]
2	PGC2/DACOUT1/AD1AN7/AD2AN3/CMP1D/CMP2D/ CMP3D/**RP2**/SCL2/RA1	16	OSCI/CLKI/**RP34**/IOMF6/RC1
3	AV_{SS}	17	V_{SS}
4	AV_{DD}	18	V_{DD}
5	OA1OUT/AD1AN0/CMP1A/**RP3**/RA2	19	PGC3/**RP35**/PWM4H/RC2
6	OA1IN-/AD1ANN1/AD2AN0/**RP4**/RA3	20	PGD3/**RP36**/PWM3H/IOMD0/RC3
7	OA1IN+/AD1AN1/CMP1B/**RP5**/RA4	21	**RP37**/PWM3L/IOMD1/RC4
8	V_{SS}	22	**RP49**/PWM2H/IOMD2/RD0
9	V_{DD}	23	TCK/**RP50**/PWM2L/IOMD3/RD1
10	OA2OUT/AD2AN1/CMP2A/**RP17**/INT0/RB0	24	TDO/**RP51**/PWM1H/IOMD4/RD2
11	TMS/OA2IN-/AD1AN4/AD2ANN1/**RP18**/RB1[4]	25	TDI/**RP52**/PWM1L/IOMD5/RD3
12	OA2IN+/AD2AN4/CMP2B/**RP19**/RB2	26	V_{SS}
13	PGD1/AD1AN5/CMP1C/ISRC0/IBIAS0/**RP20**/SDA1/RB3	27	V_{DD}
14	PGC1/AD2AN5/CMP2C/ISRC1/IBIAS1/**RP21**/SCL1/RB4	28	$\overline{MCLR}$

Note:

1. **RPn** represents remappable peripheral functions.
2. This pin has 8x drive strength.
3. Unless otherwise stated, pins are 4x drive strength. Refer to Electrical Specifications for current drive strength details.
4. A pull-up resistor is connected to this pin when device is erased (JTAG enabled) and during programming.

Pin Diagrams

Figure 3. 36-Pin VQFN

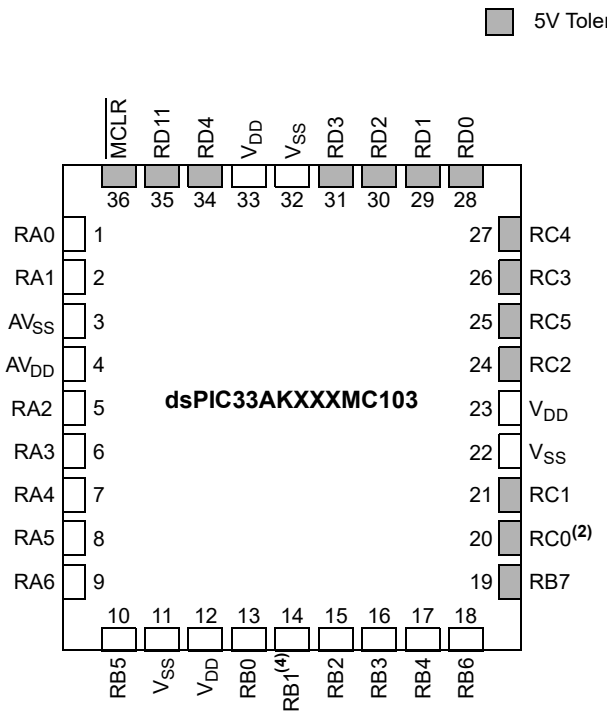


Table 4. 36-Pin VQFN Complete Pin Function Descriptions[1,3]

Pin	Function	Pin	Function
1	PGD2/AD2AN6/CMP3C/ISRC2/IBIAS2/**RP1**/SDA2/IOMF2/RA0	19	AD2ANN2/AD2AN8/**RP24**/IOMF0/RB7
2	PGC2/DACOUT1/AD1AN7/AD2AN3/CMP1D/CMP2D/ CMP3D/**RP2**/SCL2/RA1	20	OSCO/CLKO/**RP33**/IOMF5/RC0[2]
3	AV_{SS}	21	OSCI/CLKI/**RP34**/IOMF6/RC1
4	AV_{DD}	22	V_{SS}
5	OA1OUT/AD1AN0/CMP1A/**RP3**/RA2	23	V_{DD}
6	OA1IN-/AD1ANN1/AD2AN0/**RP4**/RA3	24	PGC3/**RP35**/PWM4H/RC2
7	OA1IN+/AD1AN1/CMP1B/**RP5**/RA4	25	**RP38**/PWM4L/RC5
8	OA3OUT/AD1AN3/CMP3A/**RP6**/RA5	26	PGD3/**RP36**/PWM3H/IOMD0/RC3
9	OA3IN-/AD1AN2/**RP7**/RA6	27	**RP37**/PWM3L/IOMD1/RC4
10	OA3IN+/AD2AN2/CMP3B/**RP22**/RB5	28	**RP49**/PWM2H/IOMD2/RD0
11	V_{SS}	29	TCK/**RP50**/PWM2L/IOMD3/RD1
12	V_{DD}	30	TDO/**RP51**/PWM1H/IOMD4/RD2
13	OA2OUT/AD2AN1/CMP2A/**RP17**/INT0/RB0	31	TDI/**RP52**/PWM1L/IOMD5/RD3
14	TMS/OA2IN-/AD1AN4/AD2ANN1/**RP18**/RB1[4]	32	V_{SS}
15	OA2IN+/AD2AN4/CMP2B/**RP19**/RB2	33	V_{DD}
16	PGD1/AD1AN5/CMP1C/ISRC0/IBIAS0/**RP20**/SDA1/RB3	34	**RP53**/PCI22/RD4
17	PGC1/AD2AN5/CMP2C/ISRC1/IBIAS1/**RP21**/SCL1/RB4	35	**RP60**/RD11
18	AD1ANN2/AD1AN8/**RP23**/RB6	36	$\overline{MCLR}$

Notes:

1. **RPn** represents remappable peripheral functions.
2. This pin has 8x drive strength.
3. Unless otherwise stated, pins are 4x drive strength. Refer to Electrical Specifications for current drive strength details.
4. A pull-up resistor is connected to this pin when device is erased (JTAG enabled) and during programming.

Pin Diagrams

Figure 4. 48-Pin VQFN, TQFP

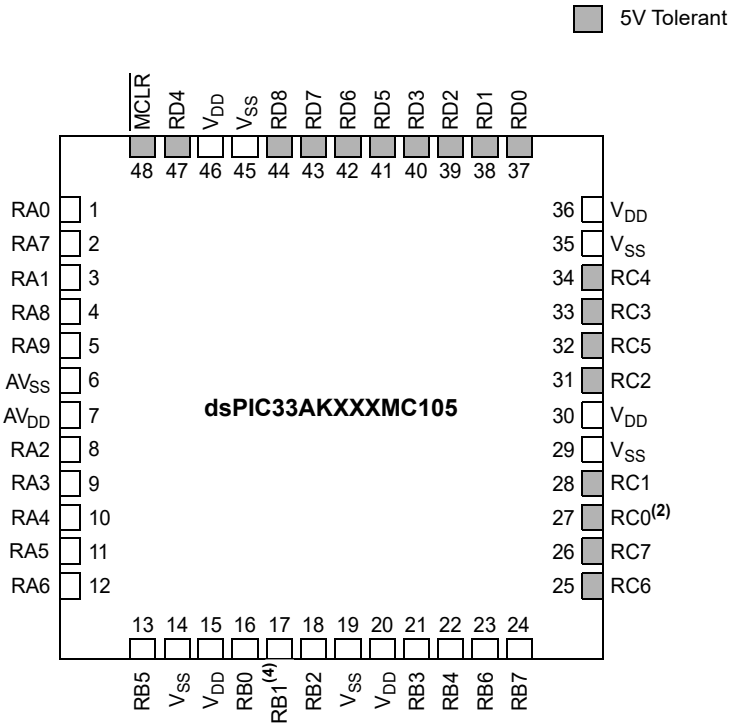


Table 5. 48-Pin VQFN, TQFP Complete Pin Function Descriptions(1,3)

Pin	Function	Pin	Function
1	PGD2/AD2AN6/CMP3C/ISRC2/IBIAS2/**RP1**/SDA2/IOMF2/RA0	25	**RP39**/RC6
2	AD1AN6/**RP8**/IOMF1/RA7	26	**RP40**/RC7
3	PGC2/DACOUT1/AD1AN7/AD2AN3/CMP1D/CMP2D/ CMP3D/**RP2**/SCL2/RA1	27	OSCO/CLKO/**RP33**/IOMF5/RC0(2)
4	AD2AN9/ISRC3/IBIAS3/**RP9**/RA8	28	OSCI/CLKI/**RP34**/IOMF6/RC1
5	AD1ANN3/AD1AN9/**RP10**/RA9	29	V_{SS}
6	AV_{SS}	30	V_{DD}
7	AV_{DD}	31	PGC3/**RP35**/PWM4H/RC2
8	OA1OUT/AD1AN0/CMP1A/**RP3**/RA2	32	**RP38**/PWM4L/RC5
9	OA1IN-/AD1ANN1/AD2AN0/**RP4**/RA3	33	PGD3/**RP36**/PWM3H/IOMD0/RC3
10	OA1IN+/AD1AN1/CMP1B/**RP5**/RA4	34	**RP37**/PWM3L/IOMD1/RC4
11	OA3OUT/AD1AN3/CMP3A/**RP6**/RA5	35	V_{SS}
12	OA3IN-/AD1AN2/**RP7**/RA6	36	V_{DD}
13	OA3IN+/AD2AN2/CMP3B/**RP22**/RB5	37	**RP49**/PWM2H/IOMD2/RD0
14	V_{SS}	38	TCK/**RP50**/PWM2L/IOMD3/RD1
15	V_{DD}	39	TDO/**RP51**/PWM1H/IOMD4/RD2
16	OA2OUT/AD2AN1/CMP2A/**RP17**/INT0/RB0	40	TDI/**RP52**/PWM1L/IOMD5/RD3
17	TMS/OA2IN-/AD1AN4/AD2ANN1/**RP18**/RB1(4)	41	**RP54**/ASCL1/RD5
18	OA2IN+/AD2AN4/CMP2B/**RP19**/RB2	42	**RP55**/ASDA1/RD6
19	V_{SS}	43	**RP56**/ASCL2/IOMD7/IOMF4/RD7
20	V_{DD}	44	**RP57**/ASDA2/IOMD6/IOMF3/RD8
21	PGD1/AN1P5/CMP1C/ISRC0/IBIAS0/**RP20**/SDA1/RB3	45	V_{SS}
22	PGC1/AD2AN5/CMP2C/ISRC1/IBIAS1/**RP21**/SCL1/RB4	46	V_{DD}
23	AD1ANN2/AD1AN8/**RP23**/RB6	47	**RP53**/PCI22/RD4
24	AD2ANN2/AD2AN8/**RP24**/IOMF0/RB7	48	$\overline{MCLR}$

.....continued

Pin	Function	Pin	Function

Note:

1. **RPn** represents remappable peripheral functions.
2. This pin has 8x drive strength.
3. Unless otherwise stated, pins are 4x drive strength. Refer to Electrical Specifications for current drive strength details.
4. A pull-up resistor is connected to this pin when device is erased (JTAG enabled) and during programming.

Pin Diagrams

Figure 5. 64-Pin VQFN, TQFP

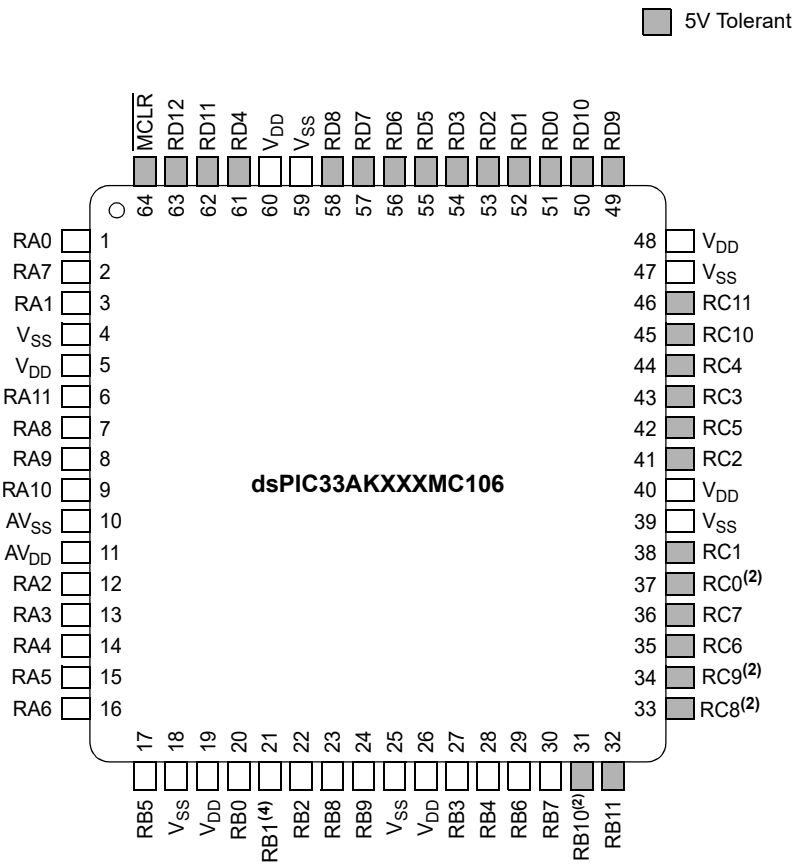


Table 6. 64-Pin VQFN, TQFP Complete Pin Function Descriptions[1,3]

Pin	Function	Pin	Function
1	PGD2/AD2AN6/CMP3C/ISRC2/IBIAS2/**RP1**/SDA2/IOMF2/RA0	33	**RP41**/IOMD11/IOMF11/PCI20/RC8[2]
2	AD1AN6/**RP8**/IOMF1/RA7	34	**RP42**/IOMD10/SDO2/IOMF10/PCI19/RC9[2]
3	PGC2/DACOUT1/AD1AN7/AD2AN3/CMP1D/CMP2D/ CMP3D/**RP2**/SCL2/RA1	35	**RP39**/RC6
4	V_{SS}	36	**RP40**/RC7
5	V_{DD}	37	OSCO/CLKO/**RP33**/IOMF5/RC0[2]
6	AD1AN10/**RP12**/RA11	38	OSCI/CLKI/**RP34**/IOMF6/RC1
7	AD2AN9/ISRC3/IBIAS3/**RP9**/RA8	39	V_{SS}
8	AD1ANN3/AD1AN9/**RP10**/RA9	40	V_{DD}
9	AD2ANN3/AD2AN7/**RP11**/RA10	41	PGC3/**RP35**/PWM4H/RC2
10	AV_{SS}	42	**RP38**/PWM4L/RC5
11	AV_{DD}	43	PGD3/**RP36**/PWM3H/IOMD0/RC3
12	OA1OUT/AD1AN0/CMP1A/**RP3**/RA2	44	**RP37**/PWM3L/IOMD1/RC4
13	OA1IN-/AD1ANN1/AD2AN0/**RP4**/RA3	45	**RP43**/IOMD9/IOMF9/RC10
14	OA1IN+/AD1AN1/CMP1B/**RP5**/RA4	46	**RP44**/IOMD8/IOMF8/RC11
15	OA3OUT/AD1AN3/CMP3A/**RP6**/RA5	47	V_{SS}
16	OA3IN-/AD1AN2/**RP7**/RA6	48	V_{DD}
17	OA3IN+/AD2AN2/CMP3B/**RP22**/RB5	49	**RP58**/IOMF7/RD9
18	V_{SS}	50	**RP59**/RD10
19	V_{DD}	51	**RP49**/PWM2H/IOMD2/RD0
20	OA2OUT/AD2AN1/CMP2A/**RP17**/INT0/RB0	52	TCK/**RP50**/PWM2L/IOMD3/RD1

.....continued

Pin	Function	Pin	Function
21	TMS/OA2IN-/AD1AN4/AD2ANN1/**RP18**/RB1(4)	53	TDO/**RP51**/PWM1H/IOMD4/RD2
22	OA2IN+/AD2AN4/CMP2B/**RP19**/RB2	54	TDI/**RP52**/PWM1L/IOMD5/RD3
23	AD1AN11/**RP25**/RB8	55	**RP54**/ASCL1/RD5
24	AD2AN10/**RP26**/RB9	56	**RP55**/ASDA1/RD6
25	V_{SS}	57	**RP56**/ASCL2/IOMD7/IOMF4/RD7
26	V_{DD}	58	**RP57**/ASDA2/IOMD6/IOMF3/RD8
27	PGD1/AD1AN5/CMP1C/ISRC0/IBIAS0/**RP20**/SDA1/RB3	59	V_{SS}
28	PGC1/AD2AN5/CMP2C/ISRC1/IBIAS1/**RP21**/SCL1/RB4	60	V_{DD}
29	AD1ANN2/AD1AN8/**RP23**/RB6	61	**RP53**/PCI22/RD4
30	AD2ANN2/AD2AN8/**RP24**/IOMF0/RB7	62	**RP60**/RD11
31	**RP27**/SCK2/RB10(2)	63	**RP61**/PCI21/RD12
32	**RP28**/SDI2/RB11	64	$\overline{MCLR}$

Note:

1. **RPn** represents remappable peripheral functions.
2. This pin has 8x drive strength.
3. Unless otherwise stated, pins are 4x drive strength. Refer to Electrical Specifications for current drive strength details.
4. A pull-up resistor is connected to this pin when device is erased (JTAG enabled) and during programming.

Pinout I/O Descriptions

Table 7. Pinout I/O Descriptions

Pin Name(1)	Pin Type	Buffer Type	PPS	Description
AN1P0 - AN1P11	I	Analog	No	ADC1 positive input channels
AN1N1 - AN1N3	I	Analog	No	ADC1 negative input channels
AN2P0 - AN2P10	I	Analog	No	ADC2 positive input channels
AN2N1 - AN2N3	I	Analog	No	ADC2 negative input channels
ADTRG31	I	ST	Yes	ADC Trigger Input 31
CLKI	I	ST/CMOS	No	External Clock (EC) source input. Always associated with OSCI pin function.
CLKO	O	—	No	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes. Always associated with OSCO pin function.
OSCI	I	ST/CMOS	No	Oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise.
OSCO	I/O	—	No	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes.
REFCLKI	I	ST	Yes	Reference clock input
REFCLKO	O	—	Yes	Reference clock output
INT0	I	ST	No	External Interrupt 0
INT1	I	ST	Yes	External Interrupt 1
INT2	I	ST	Yes	External Interrupt 2
INT3	I	ST	Yes	External Interrupt 3
INT4	I	ST	Yes	External Interrupt 4
IOCA[4:0]	I	ST	No	Interrupt-on-Change input for PORTA
IOCB[15:0]	I	ST	No	Interrupt-on-Change input for PORTB
IOCC[15:0]	I	ST	No	Interrupt-on-Change input for PORTC
IOCD[15:0]	I	ST	No	Interrupt-on-Change input for PORTD
IOCE[15:0]	I	ST	No	Interrupt-on-Change input for PORTE
IOCF[15:0]	I	ST	No	Interrupt-on-Change input for PORTF
IOMD[n:0]	O	ST	Yes	I/O Monitor Reference
IOMF[n:0]	I	ST	Yes	I/O Monitor Feedback
QEIA1	I	ST	Yes	QEI Input A1
QEIB1	I	ST	Yes	QEI Input B1
QEINDX1	I	ST	Yes	QEI Index 1 input
QEIHOM1	I	ST	Yes	QEI Home 1 input
QEICMP	O	—	Yes	QEI comparator output
RA0-RA4	I/O	ST	No	PORTA is a bidirectional I/O port

Legend: CMOS = CMOS compatible input or output; TTL = TTL input buffer; Analog = Analog input; P = Power; ST = Schmitt Trigger input with CMOS levels; O = Output; I = Input;
PPS = Peripheral Pin Select

Notes:

1. Not all pins are available in all package variants. See the Pin Diagrams section for pin availability.
2. These pins are remappable as well as dedicated.

.....continued

Pin Name(1)	Pin Type	Buffer Type	PPS	Description
RB0-RB15	I/O	ST	No	PORTB is a bidirectional I/O port
RC0-RC15	I/O	ST	No	PORTC is a bidirectional I/O port
RD0-RD15	I/O	ST	No	PORTD is a bidirectional I/O port
RE0-RE15	I/O	ST	No	PORTE is a bidirectional I/O port
RF0-RF15	I/O	ST	No	PORTF is a bidirectional I/O port
T1CK	I	ST	Yes	Timer1 external clock input
$\overline{U1CTS}$	I	ST	Yes	UART1 Clear-to-Send
$\overline{U1RTS}$	O	—	Yes	UART1 Request-to-Send
U1RX	I	ST	Yes	UART1 receive
U1TX	O	—	Yes	UART1 transmit
$\overline{U1DSR}$	I	ST	Yes	UART1 Data-Set-Ready
U1DTR	O	—	Yes	UART1 Data-Terminal-Ready
$\overline{U2CTS}$	I	ST	Yes	UART2 Clear-to-Send
$\overline{U2RTS}$	O	—	Yes	UART2 Request-to-Send
U2RX	I	ST	Yes	UART2 receive
U2TX	O	—	Yes	UART2 transmit
$\overline{U2DSR}$	I	ST	Yes	UART2 Data-Set-Ready
U2DTR	O	—	Yes	UART2 Data-Terminal-Ready
$\overline{U3CTS}$	I	ST	Yes	UART3 Clear-to-Send
$\overline{U3RTS}$	O	—	Yes	UART3 Request-to-Send
U3RX	I	ST	Yes	UART3 receive
U3TX	O	—	Yes	UART3 transmit
$\overline{U3DSR}$	I	ST	Yes	UART3 Data-Set-Ready
U3DTR	O	—	Yes	UART3 Data-Terminal-Ready
SENT1	I	ST	Yes	SENT1 input
SENT2	I	ST	Yes	SENT2 input
SENT1OUT	O	—	Yes	SENT1 output
SENT2OUT	O	—	Yes	SENT2 output
PTGTRG24	O	—	Yes	PTG Trigger Output 24
PTGTRG25	O	—	Yes	PTG Trigger Output 25
TCKI1-TCKI9	I	ST	Yes	SCCP Timer Inputs 1 through 9
ICM1-ICM9	I	ST	Yes	SCCP Capture Inputs 1 through 9
OCFA-OCFD	I	ST	Yes	SCCP Fault Inputs A through D
OCM1-OCM9	O	—	Yes	SCCP Compare Outputs 1 through 9
SCK1	I/O	ST	Yes	Synchronous serial clock I/O for SPI1
SDI1	I	ST	Yes	SPI1 data in
SDO1	O	—	Yes	SPI1 data out
$\overline{SS1}$	I/O	ST	Yes	SPI1 Client synchronization or frame pulse I/O

Legend: CMOS = CMOS compatible input or output; TTL = TTL input buffer; Analog = Analog input; P = Power; ST = Schmitt Trigger input with CMOS levels; O = Output; I = Input;
PPS = Peripheral Pin Select

Notes:

1. Not all pins are available in all package variants. See the Pin Diagrams section for pin availability.
2. These pins are remappable as well as dedicated.

.....continued

Pin Name(1)	Pin Type	Buffer Type	PPS	Description
SCK2	I/O	ST	Yes	Synchronous serial clock I/O for SPI2
SDI2	I	ST	Yes	SPI2 data in
SDO2	O	—	Yes	SPI2 data out
$\overline{SS2}$	I/O	ST	Yes	SPI2 Client synchronization or frame pulse I/O
SCK3	I/O	ST	Yes	Synchronous serial clock I/O for SPI3
SDI3	I	ST	Yes	SPI3 data in
SDO3	O	—	Yes	SPI3 data out
$\overline{SS3}$	I/O	ST	Yes	SPI3 Client synchronization or frame pulse I/O
SCL1	I/O	ST	No	Synchronous serial clock I/O for I2C1
SDA1	I/O	ST	No	Synchronous serial data I/O for I2C1
ASCL1	I/O	ST	No	Alternate synchronous serial clock I/O for I2C1
ASDA1	I/O	ST	No	Alternate synchronous serial data I/O for I2C1
SCL2	I/O	ST	No	Synchronous serial clock I/O for I2C2
SDA2	I/O	ST	No	Synchronous serial data I/O for I2C2
ASCL2	I/O	ST	No	Alternate synchronous serial clock I/O for I2C2
ASDA2	I/O	ST	No	Alternate synchronous serial data I/O for I2C2
BISS1SL	I	ST	Yes	BiSS1 Return Input
BISS1GS	I	ST	Yes	BiSS1 Get Sense
BISS1MO	O	ST	Yes	BiSS1 Output
BISS1MA	O	ST	Yes	BiSS1 Clock
TMS	I	ST	No	JTAG Test mode select pin
TCK	I	ST	No	JTAG test clock input pin
TDI	I	ST	No	JTAG test data input pin
TDO	O	—	No	JTAG test data output pin
PCI8-PCI18	I	ST	Yes	PWM Inputs 8 through 18
PCI19-PCI22	I	ST	No	PWM Inputs 19 through 22
PWMEA-PWMEF	O	—	Yes	PWM Event Outputs A through F
PWM1L-PWM4L(2)	O	—	Yes	PWM Low Outputs 1 through 4
PWM1H-PWM4H(2)	O	—	Yes	PWM High Outputs 1 through 4
CLCINA-CLCIND	I	ST	Yes	CLC Inputs A through D
CLC1OUT-CLC8OUT	O	—	Yes	CLC Outputs 1 through 8
CMP1	O	—	Yes	Comparator 1 output
CMP1A-CMP3A	I	Analog	No	Comparator Channels 1A through 3A inputs
CMP1B-CMP3B	I	Analog	No	Comparator Channels 1B through 3B inputs
CMP1C-CMP3C	I	Analog		Comparator Channels 1C through 3C inputs
CMP1D-CMP3D	I	Analog	No	Comparator Channels 1D through 3D inputs
			No	
DACOUT1	O	—	No	DAC1 output voltage

Legend: CMOS = CMOS compatible input or output; TTL = TTL input buffer; Analog = Analog input; P = Power; ST = Schmitt Trigger input with CMOS levels; O = Output; I = Input;
PPS = Peripheral Pin Select

Notes:

1. Not all pins are available in all package variants. See the Pin Diagrams section for pin availability.
2. These pins are remappable as well as dedicated.

.....continued

Pin Name(1)	Pin Type	Buffer Type	PPS	Description
IBIAS3, IBIAS2, IBIAS1, IBIAS0/ISRC3, ISRC2, ISRC1, ISRC0	O	Analog	No	Constant-Current Outputs 0 through 3
OA1IN+	I	—	No	Op Amp 1+ input
OA1IN-	I	—	No	Op Amp 1- input
OA1OUT	O	—	No	Op Amp 1 output
OA2IN+	I	—	No	Op Amp 2+ input
OA2IN-	I	—	No	Op Amp 2- input
OA2OUT	O	—	No	Op Amp 2 output
OA3IN+	I	—	No	Op Amp 3+ input
OA3IN-	I	—	No	Op Amp 3- input
OA3OUT	O	—	No	Op Amp 3 output
PGD1	I/O	ST	No	Data I/O pin for Programming/ Debugging Communication Channel 1
PGC1	I	ST	No	Clock input pin for Programming/ Debugging Communication Channel 1
PGD2	I/O	ST	No	Data I/O pin for Programming/ Debugging Communication Channel 2
PGC2	I	ST	No	Clock input pin for Programming/ Debugging Communication Channel 2
PGD3	I/O	ST	No	Data I/O pin for Programming/ Debugging Communication Channel 3
PGC3	I	ST	No	Clock input pin for Programming/ Debugging Communication Channel 3
$\overline{MCLR}$	I/P	ST	No	Master Clear (Reset) input. This pin is an active-low Reset to the device.
AV_{DD}	P	P	No	Positive supply for analog modules. This pin must be connected at all times.
AV_{SS}	P	P	No	Ground reference for analog modules. This pin must be connected at all times.
V_{DD}	P	—	No	Positive supply for peripheral logic and I/O pins
V_{SS}	P	—	No	Ground reference for logic and I/O pins

Legend: CMOS = CMOS compatible input or output; TTL = TTL input buffer; Analog = Analog input; P = Power; ST = Schmitt Trigger input with CMOS levels; O = Output; I = Input;
PPS = Peripheral Pin Select

Notes:

1. Not all pins are available in all package variants. See the Pin Diagrams section for pin availability.
2. These pins are remappable as well as dedicated.

To Our Valued Customers

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at docerrors@microchip.com. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Website at:

www.microchip.com/

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000000A is version A of document DS30000000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Website; www.microchip.com/
- Your local Microchip sales office (see last page)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our website at www.microchip.com/ to receive the most current information on all of our products.

Terminology Cross Reference

Table 8 provides updated terminology for deprecated naming conventions. Register and bit names remain unchanged, however, descriptions and usage guidance may have been updated.

Table 8. Terminology Cross References

Use Case	Deprecated Term	New Term
CPU	Master	Initiator
DMA	Master	Initiator
I^2C	Master	Host
	Slave	Client
SPI	Master	Host
	Slave	Client
UART, LIN Mode	Master	Commander
	Slave	Responder
PWM	Master	Host
	Slave	Client

Table of Contents

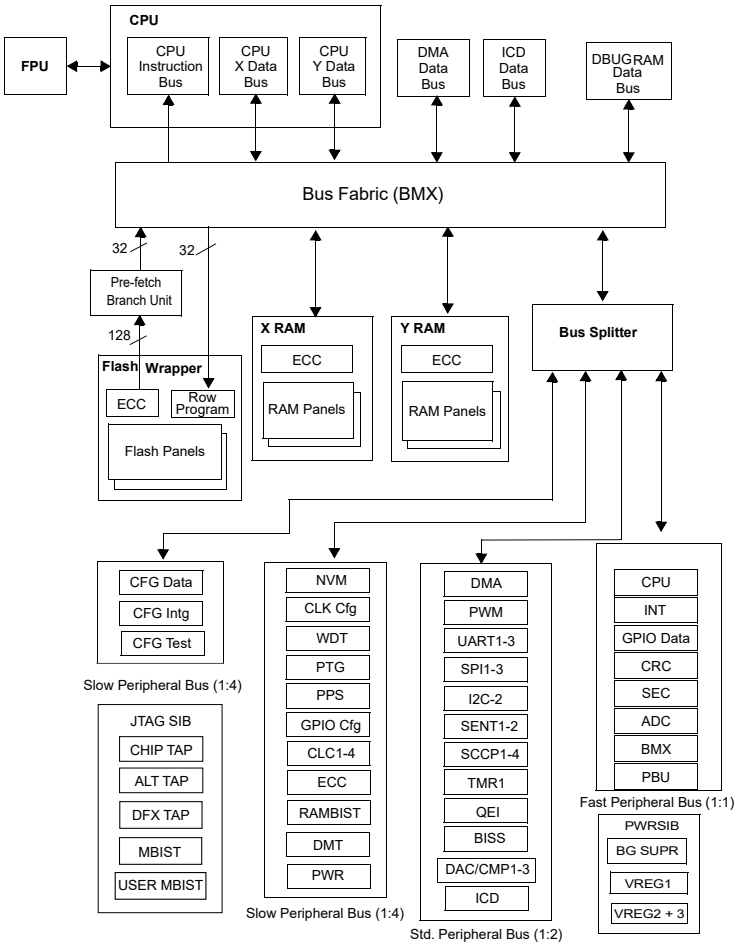
1. Device Overview

This document contains device-specific information for the dsPIC33AK128MC106 Digital Signal Controller (DSC) family of devices.

The dsPIC33AK128MC106 devices support a high-performance architecture with the Digital Signal Processor (DSP) and a Single and Double Precision Floating Point Unit (FPU). The dsPIC33AK128MC106 family of devices operate with an internal core supplied with a 1.1V using a low-voltage regulator.

Figure 1-1 shows a general block diagram of the core and peripheral modules of the dsPIC33AK128MC106 family.

Figure 1-1. dsPIC33AK128MC106 Family Block Diagram



Notes:

1. Not all I/O pins or features are implemented on all device pinout configurations. See Pinout I/O Descriptions for specific implementations by pin count.
2. Some peripheral I/Os are only accessible through Peripheral Pin Select (PPS).

2. Guidelines for Getting Started with Digital Signal Controllers

2.1 Basic Connection Requirements

Getting started with the dsPIC33AK128MC106 family devices requires attention to a minimal set of device pin connections before proceeding with development. The following pins must always be connected:

- All V_{DD} and V_{SS} power supply pins must be properly biased with required voltages (see 37. Electrical Characteristics)
- All AV_{DD} and AV_{SS} analog supply pins must be properly biased regardless of which analog modules or components of the dsPIC33A device are used (see 37. Electrical Characteristics)
- $\overline{MCLR}$ pin is connected with V_{DD} and V_{SS} based on circuit or application needs
- PGCx/PGDx pins used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes (see 2.4. ICSP Pins)
- OSCI and OSCO pins when an external oscillator source is used (see 2.5. External Oscillator Pins)

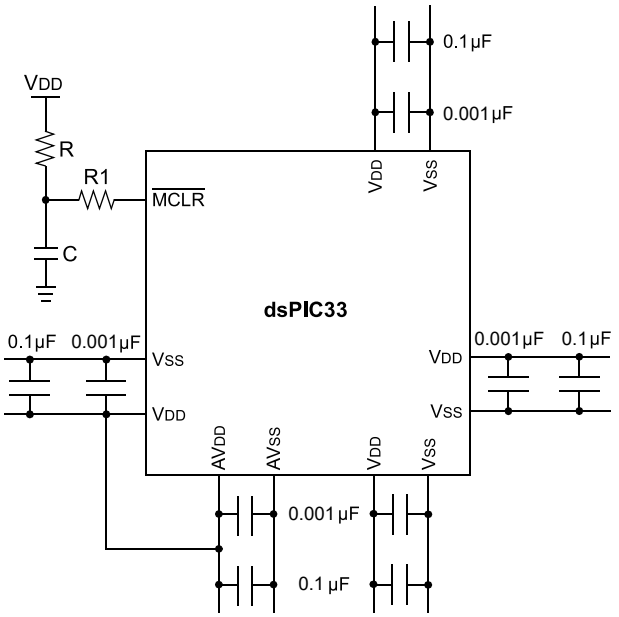
2.2 Decoupling Capacitors

The use of decoupling capacitors on every pair of power supply pins, such as V_{DD}, V_{SS}, AV_{DD} and AV_{SS}, is required.

Consider the following criteria when using decoupling capacitors:

- **Value and type of capacitor:** Recommendation of 0.1 µF (100 nF), in parallel with a 1000 pF (1 nF), 10-20V. These capacitors should be low-ESR and have a resonance frequency in the range of 20 MHz and higher. Ceramic capacitors are recommended.
- **Placement on the Printed Circuit Board:** The decoupling capacitors should be placed as close to the pins as possible. It is recommended to place the capacitors on the same side of the board as the device. If space is constricted, the capacitor can be placed on another layer on the PCB using a via; however, ensure that the trace length from the pin to the capacitor is within one-quarter inch (6 mm) in length.
- **Handling high-frequency noise:** If the board is experiencing high-frequency noise above tens of MHz, add an additional ceramic-type capacitor in parallel to the decoupling capacitors. The value can be in the range of 0.01 µF to 0.001 µF. Place this capacitor next to the primary decoupling capacitors. In high-speed circuit designs, consider implementing a decade set of capacitances as close to the power and ground pins as possible. For example, 0.1 µF in parallel with 0.01 µF and 0.001 µF.
- **Maximizing performance:** On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. Equally important is to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB track inductance.

Figure 2-1. Recommended Minimum Connection



2.3 Master Clear ($\overline{MCLR}$) Pin

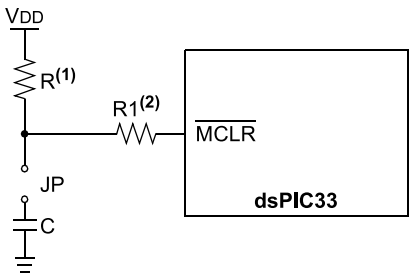
The $\overline{MCLR}$ pin provides two specific device functions:

- Device Reset
- Device Programming and Debugging

During device programming and debugging, the resistance and capacitance that can be added to the pin must be considered. Device programmers and debuggers drive the $\overline{MCLR}$ pin. Consequently, specific voltage levels (V_{IH} and V_{IL}) and fast signal transitions must not be adversely affected. Ensure that the $\overline{MCLR}$ pin V_{IH} and V_{IL} voltage specifications are met.

For example, Figure 2-2 shows the $\overline{MCLR}$ pin connections with general circuit components used, such as resistor R, series resistor R1 and capacitor C, and their placement. It is recommended to place these passive components with one-quarter inch (6mm) from the $\overline{MCLR}$ pin.

Figure 2-2. Example of $\overline{MCLR}$ Pin Connections



Notes:

1. R ≤ 10 kΩ is recommended. A suggested starting value is 10 kΩ. Ensure that the $\overline{MCLR}$ pin V_{IH} and V_{IL} specifications are met.
2. R1 ≤ 470Ω will limit any current flowing into $\overline{MCLR}$ from the external capacitor, C, in the event of $\overline{MCLR}$ pin breakdown due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS). Ensure that the $\overline{MCLR}$ pin V_{IH} and V_{IL} specifications are met.
3. C ≤ 1 µF may be recommended. However, values of C should be based on reset timings required for any application. Make sure to isolate C from the $\overline{MCLR}$ pin during programming and debugging operations.

2.4 ICSP Pins

The PGCx and PGDx pins are used for programming and debugging purposes. It is recommended to keep the trace length between the ICSP connector and the ICSP pins on the device as short as possible. If the ICSP connector is expected to experience an ESD event, a series resistor is recommended, with the value in the range of a few tens of Ohms, not to exceed 100 Ohms.

Pull-up resistors, series diodes and capacitors on the PGCx and PGDx pins are not recommended as they will interfere with the programmer/debugger communications to the device. If such discrete components are an application requirement, they should be removed from the circuit during programming and debugging. Alternatively, refer to the AC/DC characteristics and timing requirements information in the respective device Flash programming specification for information on capacitive loading limits and pin Voltage Input High (V_{IH}) and Voltage Input Low (V_{IL}) requirements.

2.5 External Oscillator Pins

When the Primary Oscillator (POSC) circuit is used to connect a crystal oscillator, special care and consideration is needed to ensure proper operation. The POSC circuit should be tested across the environmental conditions in which the end product is intended to be used. The load capacitors specified in the crystal oscillator data sheet can be used as a starting point, however, the parasitic capacitance from the PCB traces can affect the circuit, and the values may need to be altered to ensure proper start-up and operation. Excessive trace length and other physical interaction can lead to poor signal quality. Poorly tuned oscillator circuits can have reduced amplitude, incorrect frequency (runt pulses), distorted waveforms and long start-up times that may result in unpredictable application behavior, such as instruction misexecution, illegal opcode fetch, etc. Ensure that the crystal oscillator circuit is at full amplitude and the correct frequency before the system begins to execute code. In planning the application's routing and I/O assignments, ensure that adjacent port pins, and other signals in close proximity to the oscillator, do not have high frequencies, short rise and fall times, and other similar noise. For further information on the Primary Oscillator, see 12.4.3. Primary Oscillator (POSC).

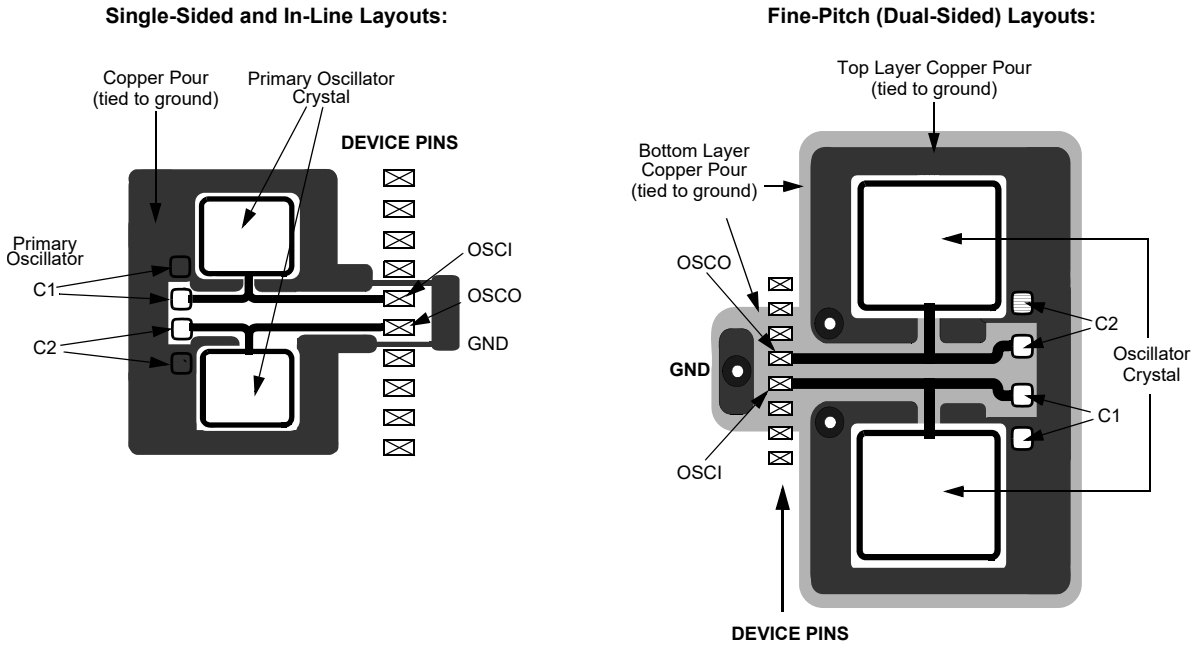
2.6 External Oscillator Layout Guidance

Use best practices during PCB layout to ensure robust start-up and operation. The oscillator circuit should be placed on the same side of the board as the device. Also, place the oscillator circuit close to the respective oscillator pins, not exceeding one-half inch (12 mm) distance between them. The load capacitors should be placed next to the oscillator itself, on the same side of the board. Use a grounded copper pour around the oscillator circuit to isolate them from surrounding circuits. The grounded copper pour should be routed directly to the MCU ground. Do not run any signal traces or power traces inside the ground pour. If using a two-sided board, avoid any traces on the other side of the board where the crystal is placed. Suggested layouts are shown in Figure 2-3. With fine-pitch packages, it is not always possible to completely surround the pins and components. A suitable solution is to tie the broken guard sections to a mirrored ground layer. In all cases, the guard trace(s) must be returned to ground.

For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the Microchip website (www.microchip.com):

- *AN943, "Practical PICmicro® Oscillator Analysis and Design"*
- *AN949, "Making Your Oscillator Work"*
- *AN1798, "Crystal Selection for Low-Power Secondary Oscillator"*

Figure 2-3. Suggested Placement of the Oscillator Circuit



2.7 Oscillator Value Conditions on Device Start-up

If the PLL of the target device is enabled and configured for the device start-up oscillator, the maximum oscillator source frequency must be limited to a certain frequency (see 12.4.2. Phase-Locked Loop (PLL)) to comply with device PLL Start-up conditions. This means that if the external oscillator frequency is outside this range, the application must start up in the FRC mode first. The default PLL settings after a POR with an oscillator frequency outside this range will violate the device operating speed.

Once the device powers up, the application firmware can initialize the PLL SFRs, CLKDIV and PLLFBD, to a suitable value, and then perform a clock switch to the Oscillator + PLL clock source. Note that clock switching must be enabled in the device Configuration Word.

2.8 Unused I/Os

Unused I/O pins should be configured as outputs and driven to a Logic Low state. Alternatively, connect a resistor (1k-10k ohm) between V_{SS} and unused pins, and drive the output to a logic low.

2.9 Bulk Capacitors

On boards with power traces running longer than six inches in length, it is suggested to use a bulk capacitor for integrated circuits, including DSCs, to supply a local power source. The value of the bulk capacitor should be determined based on the trace resistance that connects the power supply source to the device and the maximum current drawn by the device in the application. In other words, select the bulk capacitor so that it meets the acceptable voltage sag at the device. Typical values range from 4.7 µF to 47 µF.

3. CPU

The dsPIC33AK128MC106 family has a Fixed-Point fractional DSP engine supporting the Central Processing Unit (CPU). The CPU processes instructions out of program memory and utilizes system RAM to perform tasks and calculations. The CPU is interfaced to memory and peripherals through the bus matrix. The CPU supports coprocessors, including the Floating Point Unit (FPU) for mathematical computation.

CPU key features:

- 32-bit working registers
- Unified memory map
- 5-stage instruction pipeline
- Conditional branching with speculative execution
- Instruction pre-fetch cache
- Mathematical support
- Low overhead loop support

3.1 Architectural Overview

The dsPIC33A CPU has 32-bit (data) modified, Harvard architecture with a 5-stage instruction pipeline, single phase clock design, with 32-bit instructions.

The CPU has a 32-bit instruction word with a variable length opcode field. The CPU also supports some instructions that are only available in 16-bit format. The Program Counter (PC) is 24 bits wide to access a 16MB (24-bit address) unified linear address map.

The CPU supports up to eight addressing modes. A 5-stage fully interlocked instruction pipeline with reduced branch latency and hardware mitigated pipeline hazard stalls helps maintain throughput and provides predictable execution. Most instructions execute in a single-cycle effective execution rate, with the exception of instructions that change the program flow. A hardware program loop construct is supported by the overhead free `REPEAT` instruction, which is interruptible at any point. For loops greater than one instruction, the DBT (Decrement Test and Branch) instruction may be used to reduce loop overhead.

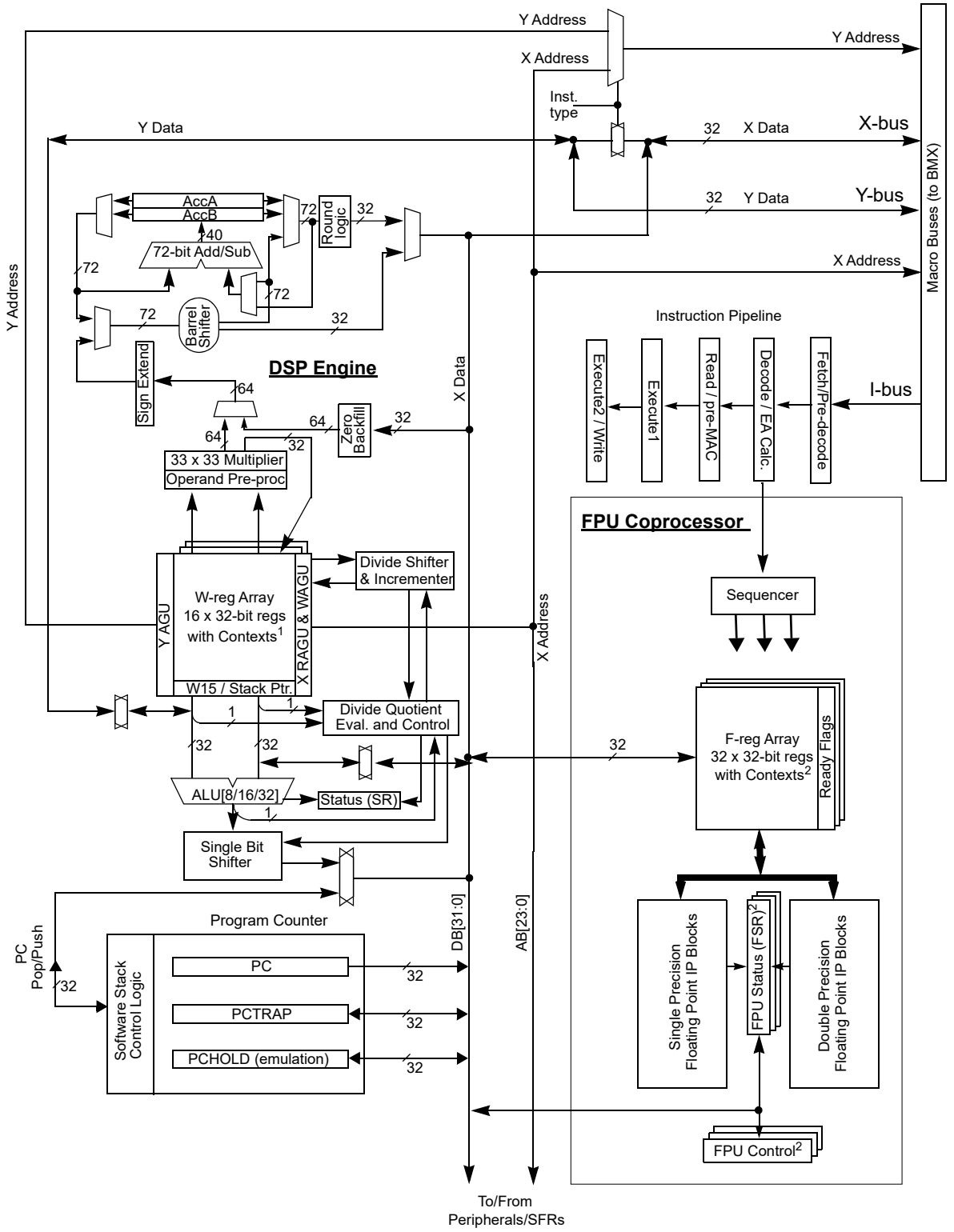
The CPU supports High Performance Math Support with a tightly coupled 16/32-bit Integer and a Fixed-Point fractional DSP engine with a 72-bit shifter, saturation and rounding support. There is an optional common issue Single and Double Precision Floating Point Unit (FPU) coprocessor with an independent load-store execution pipeline.

CPU Supports closely coupled coprocessor macros with the following features:

- Decode and issue from the CPU pipeline into independent coprocessor pipeline(s)
- Pipeline hazards detected and mitigated in both the CPU and coprocessor(s)
- Dedicated data move and conditional coprocessor status branch instructions
- Coprocessor interrupt support

Figure 3-1 illustrates the dsPIC33A CPU block diagram.

Figure 3-1. dsPIC33A Core Conceptual Block Diagram with FPU Coprocessor



3.2 CPU Register Descriptions

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	PC	31:24								
		23:16	PC[23:16]							
		15:8	PC[15:8]							
		7:0	PC[7:0]							
0x04	SPLIM	31:24								
		23:16	SPLIM[23:16]							
		15:8	SPLIM[15:8]							
		7:0	SPLIM[7:0]							
0x08	RCOUNT	31:24	RCOUNT[31:24]							
		23:16	RCOUNT[23:16]							
		15:8	RCOUNT[15:8]							
		7:0	RCOUNT[7:0]							
0x0C	DISIIPL	31:24								
		23:16								
		15:8								
		7:0						DISIIPL[2:0]		
0x10	CORCON	31:24								
		23:16								
		15:8				US				
		7:0	SATA	SATB	SATDW	ACCSAT			RND	IF
0x14	MODCON	31:24								
		23:16								
		15:8	XMODEN	YMODEN						
		7:0	YWM[3:0]				XWM[3:0]			
0x18	XMODSRT	31:24								
		23:16	XMODSRT[23:16]							
		15:8	XMODSRT[15:8]							
		7:0	XMODSRT[7:0]							
0x1C	XMODEND	31:24								
		23:16	XMODEND[23:16]							
		15:8	XMODEND[15:8]							
		7:0	XMODEND[7:0]							
0x20	YMODSRT	31:24								
		23:16	YMODSRT[23:16]							
		15:8	YMODSRT[15:8]							
		7:0	YMODSRT[7:0]							
0x24	YMODEND	31:24								
		23:16	YMODEND[23:16]							
		15:8	YMODEND[15:8]							
		7:0	YMODEND[7:0]							
0x28	XBREV	31:24								
		23:16								
		15:8		XBREV[14:8]						
		7:0	XBREV[7:0]							
0x2C	PCTRAP	31:24								
		23:16		PCTRAP[22:16]						
		15:8	PCTRAP[15:8]							
		7:0	PCTRAP[7:0]							
0x30	FEX	31:24	FEX[31:24]							
		23:16	FEX[23:16]							
		15:8	FEX[15:8]							
		7:0	FEX[7:0]							
0x34	FEX2	31:24	FEX2[31:24]							
		23:16	FEX2[23:16]							
		15:8	FEX2[15:8]							
		7:0	FEX2[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x38	PCHOLD	31:24								
		23:16	PCHOLD[23:16]							
		15:8	PCHOLD[15:8]							
		7:0	PCHOLD[7:0]							
0x3C	VFA	31:24								
		23:16	VFA[23:16]							
		15:8	VFA[15:8]							
		7:0	VFA[7:0]							
0x40 ... 0x1E0F	Reserved									
0x1E10	HPCCON	31:24								
		23:16								
		15:8	ON		CLR					
		7:0								
0x1E10	HPCSEL0	31:24						SELECT[3][4:0]		
		23:16						SELECT[2][4:0]		
		15:8						SELECT[1][4:0]		
		7:0						SELECT[0][4:0]		
0x1E14	HPCSEL1	31:24						SELECT[7][4:0]		
		23:16						SELECT[6][4:0]		
		15:8						SELECT[5][4:0]		
		7:0						SELECT[4][4:0]		
0x1E18 ... 0x1E1F	Reserved									
0x1E20	HPCCNTL0	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E24	HPCCNTH0	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E28	HPCCNTL1	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E2C	HPCCNTH1	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E30	HPCCNTL2	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E34	HPCCNTH2	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E38	HPCCNTL3	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E3C	HPCCNTH3	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1E40	HPCCNTL4	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E44	HPCCNTH4	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E48	HPCCNTL5	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E4C	HPCCNTH5	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E50	HPCCNTL6	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E54	HPCCNTH6	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E58	HPCCNTL7	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E5C	HPCCNTH7	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E60	CHECON	31:24								
		23:16								ISBBUF
		15:8	ON				CHEINV	CHECOH		
		7:0								FLTINJ
0x1E64	CHESTAT	31:24								
		23:16								
		15:8								
		7:0						TPE	RD	PAR
0x1E68	CHEFLTINJ	31:24								
		23:16								
		15:8								
		7:0	FLTPTR[7:0]							

3.2.1 CPU Program Counter Register

Name: PC
Offset: 0x000

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	PC[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PC[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – PC[23:0] Program Counter bits

3.2.2 Stack Pointer Limit Value Register

Name: SPLIM
Offset: 0x004

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	SPLIM[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SPLIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SPLIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – SPLIM[23:0] Stack Limit Address bits

3.2.3 REPEAT Loop Counter Register

Name: RCOUNT
Offset: 0x008

Bit	31	30	29	28	27	26	25	24
	RCOUNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RCOUNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RCOUNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RCOUNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – RCOUNT[31:0] Current Loop Counter Value for REPEAT Instruction

3.2.4 DISIIPL(W) Instruction Current IPL Threshold

Name: DISIIPL
Offset: 0x00C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						DISIIPL[2:0]		
Access						R	R	R
Reset						0	0	0

Bits 2:0 – DISIIPL[2:0] DISIIPL(W) current IPL threshold value

3.2.5 Core Mode Control Register(1)

Name: CORCON
Offset: 0x010

Note:

1. The Core Control register (CORCON) has bits that control the operation of the DSP multiplier hardware. The IPL3 bit is concatenated with the IPL[2:0] bits (SR[7:5]) to form the CPU Interrupt Priority Level.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				US				
Access				R/W				
Reset				0				

Bit	7	6	5	4	3	2	1	0
	SATA	SATB	SATDW	ACCSAT			RND	IF
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit 12 – US Unsigned or Signed Multiplier Mode Select bit

Value	Description
1	Unsigned mode enabled for DSP ops
0	Signed mode enabled for DSP ops

Bit 7 – SATA AccA Saturation Enable bit

Value	Description
1	Accumulator A saturation enabled
0	Accumulator A saturation disabled

Bit 6 – SATB AccB Saturation Enable bit

Value	Description
1	Accumulator B saturation enabled
0	Accumulator B saturation disabled

Bit 5 – SATDW Data Space Write from DSP Engine Saturation Enable bit

Value	Description
1	Data Space write saturation enabled
0	Data Space write saturation disabled

Bit 4 – ACCSAT Accumulator Saturation Mode Select bit

Value	Description
1	9.63 saturation (super saturation)
0	1.63 saturation (normal saturation)

Bit 1 – RND Rounding Mode Select bit

Value	Description
1	Biased (conventional) rounding enabled
0	Unbiased (convergent) rounding enabled

Bit 0 – IF Integer or Fractional Multiplier Mode Select bit

Value	Description
1	Integer mode is enabled for DSP multiply
0	Fractional mode is enabled for DSP multiply

3.2.6 Modulo Addressing Control Register(1)

Name: MODCON
Offset: 0x0014

Note:
1. The MODCON register enables and configures Modulo Addressing (circular buffers).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	XMODEN	YMODEN						
Access	R/W	R/W						
Reset	0	0						

Bit	7	6	5	4	3	2	1	0
	YWM[3:0]				XWM[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	1	0	0	0	1

Bit 15 – XMODEN X RAGU & X WAGU Modulus Addressing Enable bit

Value	Description
1	X AGU Modulus Addressing enabled
0	X AGU Modulus Addressing disabled

Bit 14 – YMODEN Y AGU Modulus Addressing Enable bit

Value	Description
1	Y AGU Modulus Addressing enabled
0	Y AGU Modulus Addressing disabled

Bits 7:4 – YWM[3:0] Y AGU W Register Select for Modulo Addressing bit

Value	Description
1111	Modulo Addressing disabled (W15 does not support Modulo Addressing)
1110	W14 selected for Modulo Addressing
...	
0000	W0 selected for Modulo Addressing

Bits 3:0 – XWM[3:0] X RAGU & X WAGU W Register Select for Modulo Addressing bit

Value	Description
1111	Modulo Addressing disabled (W15 does not support Modulo Addressing)
1110	W14 selected for Modulo Addressing
...	
0000	W0 selected for Modulo Addressing

3.2.7 X AGU Modulo Addressing Start Register

Name: XMODSRT
Offset: 0x0018

Note:

1. The XMODSRT and XMODEND registers hold the start and end addresses for modulo (circular) buffers implemented in the X data memory address space.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	XMODSRT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	XMODSRT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	XMODSRT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – XMODSRT[23:0] X RAGU & X WAGU Modulo Addressing Start Address bits(1)

3.2.8 X AGU Modulo Addressing End Register(1)

Name: XMODEND
Offset: 0x001C

Note:

1. The XMODSRT and XMODEND registers hold the start and end addresses for modulo (circular) buffers implemented in the X data memory address space.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	XMODEND[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	XMODEND[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	XMODEND[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – XMODEND[23:0] X RAGU & X WAGU Modulo Addressing End Address bits

3.2.9 Y AGU Modulo Addressing Start Address Register(1)

Name: YMODSRT
Offset: 0x0020

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	YMODSRT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	YMODSRT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	YMODSRT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – YMODSRT[23:0] Y RAGU Modulo Addressing Start Address bits

3.2.10 Y AGU Modulo Addressing End Register(1)

Name: YMODEND
Offset: 0x0024

Note:

1. The YMODSRT and YMODEND registers hold the start and end addresses for modulo (circular) buffers implemented in the Y data memory address space.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	YMODEND[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	YMODEND[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	YMODEND[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – YMODEND[23:0] Y RAGU Modulo Addressing End Address bits

3.2.11 X AGU Bit Reversal Addressing Control Register(1)

Name: XBREV
Offset: 0x0028

Note:
1. The XBREV register sets the buffer size used for Bit-Reversed Addressing.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
		XBREV[14:8]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	XBREV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 14:0 – XBREV[14:0] X AGU Bit Reversed Modifier bits

3.2.12 Captured PC Address at Time of Trap Register

Name: PCTRAP
Offset: 0x002C

Notes:

1. PCTRAP[0] always reads as 0.
2. If the current IPL is greater or equal to 8, the PC address will not be captured.
3. Hardware update is blocked after the first PCTRAP update occurs, preventing newer traps from overwriting the source address of older ones. Update can be re-enabled by user attempting to write 24'h000000 to PCTRAP (write will not occur, preserving PCTRAP contents).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		PCTRAP[22:16]						
Access		R	R	R	R	R	R	R
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PCTRAP[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PCTRAP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 22:0 – PCTRAP[22:0] Captured PC Address at time of trap exception(1,2,3)

3.2.13 Force Execution Instruction Register 1(1)

Name: FEX
Offset: 0x0030

Bit	31	30	29	28	27	26	25	24
	FEX[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FEX[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FEX[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FEX[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 31:0 – FEX[31:0] For 2 word operations, FEX contains the first instruction to be executed using the UFEX instruction.

FEX is only visible as a R/W register in Debug mode. In all other operating modes, it is read-only of all 0's.

3.2.14 Force Execution Instruction Register 2(1)

Name: FEX2
Offset: 0x0034

Bit	31	30	29	28	27	26	25	24
	FEX2[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FEX2[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FEX2[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FEX2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 31:0 – FEX2[31:0] For 2 word operations, FEX contains the second instruction to be executed using the UFEX instruction.

FEX is only visible as a R/W register in Debug mode. In all other operating modes, it is read-only of all 0's.

3.2.15 Debug Hold PC Register

Name: PCHOLD
Offset: 0x0038

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	PCHOLD[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PCHOLD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PCHOLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 23:0 – PCHOLD[23:0] Debug Hold PC register bits
PCHOLD is only visible as a R/W register in Debug mode. In all other operating modes, it is read-only of all 0's.

3.2.16 Vector Fail Address Register

Name: VFA
Offset: 0x003C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	VFA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	VFA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VFA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 23:0 – VFA[23:0] Vector Fail Address Register bits

3.2.17 CPU STATUS Register(1)

Name: SR

Note:

1. The CPU STATUS register is not memory mapped.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	VF					CTX[2:0]		
Access	R					R	R	R
Reset	0					0	0	0

Bit	15	14	13	12	11	10	9	8
	OA	OB	SA	SB	OAB	SAB		IPL3
Access	R/W	R/W	R/W	R/W	R	R/C		R/C
Reset	0	0	0	0	0	0		0

Bit	7	6	5	4	3	2	1	0
	IPL[2:0]			RA	N	OV	Z	C
Access	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 23 – VF Vector (Fetch) Fail Status bit

Value	Description
1	Indicates to the Bus Error handler that the source of the bus error is a vector fetch. The vector data read will be substituted with the contents of the Vector Fail Address (VFA) SFR.
0	Indicates to the Bus Error handler that the source of the bus error is not a vector fetch.

Bits 18:16 – CTX[2:0] Current (W register) Context Identifier bits

Value	Description
111	Context 7 is currently in use
110	Context 6 is currently in use
101	Context 5 is currently in use
100	Context 4 is currently in use
011	Context 3 is currently in use
010	Context 2 is currently in use
001	Context 1 is currently in use
000	Context 0 is currently in use

Bit 15 – OA Accumulator A Fractional Overflow Status bit

Value	Description
1	Accumulator A fractional overflow has occurred (its contents can no longer be represented as a 1.31 fractional value)
0	Accumulator A not overflowed

Bit 14 – OB Accumulator B Fractional Overflow Status bit

Value	Description
1	Accumulator B fractional overflow has occurred (its contents can no longer be represented as a 1.31 fractional value)
0	Accumulator B not overflowed

Bit 13 – SA Accumulator A Saturation/Sign Overflow 'Sticky' Status bit

Value	Description
1	Accumulator A is saturated, or has been saturated at some time, or has overflowed into bit 71 (if saturation is disabled)
0	Accumulator A is not saturated or has not overflowed into bit 71 (if saturation is disabled)

Bit 12 – SB Accumulator B Saturation/Sign Overflow 'Sticky' Status bit

Value	Description
1	Accumulator B is saturated, or has been saturated at some time, or has overflowed into bit 71 (if saturation is disabled)
0	Accumulator B is not saturated or has not overflowed into bit 71 (if saturation is disabled)

Bit 11 – OAB OA || OB Combined Accumulator Fractional Overflow Status bit

Value	Description
1	Accumulators A or B fractional overflow has occurred (one or both of their contents can no longer be represented as a 1.31 fractional value)
0	Neither Accumulators A nor B have overflowed

Bit 10 – SAB SA || SB Combined Accumulator 'Sticky' Status bit

Value	Description
1	Accumulators A or B are saturated, or have been saturated at some time, or have overflowed into bit 71 (if saturation is disabled)
0	Neither Accumulator A nor B are saturated or have overflowed into bit 71 (if saturation is disabled)

Bit 8 – IPL3 MS-bit of CPU Priority Level Nibble bit

Value	Description
1	CPU Priority ≥ 8 (trap exception underway)
0	CPU Priority < 8 (no trap exception underway)

Bits 7:5 – IPL[2:0] CPU Interrupt Priority Level status bits
User Mode: This bit is R/C-0 (read only if Supervisor Mode supported) and will reset to 1'b0.
Supervisor Mode: This bit is R/C-0 (CPU will reset into Supervisor Mode).

Value	Description
111	All interrupts disabled
110	Level 7 interrupts enabled
101	Level 6 and 7 interrupts enabled
100	Level 5 through 7 interrupts enabled
011	Level 4 through 7 interrupts enabled
010	Level 3 through 7 interrupts enabled
001	Level 2 through 7 interrupts enabled
000	Level 1 through 7 interrupts enabled

Bit 4 – RA REPEAT Loop Active bit

Value	Description
1	REPEAT loop in progress
0	REPEAT loop not in progress

Bit 3 – N ALU Negative bit

Bit 2 – OV ALU Overflow bit

Bit 1 – Z ALU 'Sticky' Zero bit

Value	Description
1	An operation which effects the Z bit has set it at some time in the past
0	The most recent operation which effects the Z bit has cleared it (i.e. a non-zero result)

Bit 0 – C ALU Carry/$\overline{\text{Borrow}}$ bit
SR[31:0] is stacked during exception processing, preserving context.

3.3 CPU Operation

3.3.1 Instruction Set

The dsPIC33A instruction set has two classes of instructions: MCU instructions and DSP instructions. These two classes are seamlessly integrated into the architecture and execute from a single execution unit. The instruction supports integer, fixed point and floating-point math operation.

3.3.2 Data Space Addressing

The Data Space is split into two blocks as X and Y data memory. Each memory block has its own independent Address Generation Unit (AGU). The MCU class of instructions operates solely through the X memory AGU, which accesses the entire memory map as one linear data space. Certain DSP instructions operate through the X and Y AGUs to support dual operand reads, which splits the data address space into two parts.

In dsPIC33A devices, overhead-free circular buffers (Modulo Addressing mode) are supported in both X and Y address spaces. The Modulo Addressing removes the software boundary checking overhead for DSP algorithms. The X AGU Circular Addressing can be used with any of the MCU class of instructions. The X AGU also supports the Bit-Reversed Addressing mode to greatly simplify input or output data reordering for radix-2 FFT algorithms.

3.3.3 Addressing Modes

The CPU supports up to eight addressing modes as shown in Table 3-1

Table 3-1. MCU Instruction Addressing Mode Definitions

Function (Source, ppp)	Function (Destination, qqq)	Description
EA = [Ws + Wb]	EA = [Wd + Wb]	Indirect with (signed) Register Offset
EA = SR	EA = SR	Status Register direct
EA = [Ws+=1]	EA = [Wd+=1]	Register indirect pre-incremented
EA = [Ws-=1]	EA = [Wd-=1]	Register indirect pre-decremented
EA = [Ws]+= 1	EA = [Wd]+= 1	Register indirect post-incremented
EA = [Ws]-= 1	EA = [Wd]-= 1	Register indirect post-decremented
EA = [Ws]	EA = [Wd]	Register indirect
EA = Ws	EA = Wd	Register direct

Each instruction is associated with a predefined addressing mode group, depending upon its functional requirements. For most instructions, the dsPIC33A CPU can execute all of the following functions in a single instruction cycle:

- Data memory read
- Working register (data) read
- Data memory write

- Program (instruction) memory read

As a result, three-operand instructions can be supported, allowing A + B = C operations to be executed in a single cycle.

3.3.4 Programmer's Model

The programmer's model for the dsPIC33A CPU is shown in Figure 3-2. All registers in the programmer's model are memory-mapped and can be manipulated directly by instructions. Table 3-2 provides a description of each register in the programmer's model.

In addition to the registers contained in the programmer's model, the dsPIC33A devices contain control registers for Modulo Addressing, Bit-Reversed Addressing and Interrupts. These registers are described in subsequent sections of this document.

All registers associated with the programmer's model are shown in Figure 3-2.

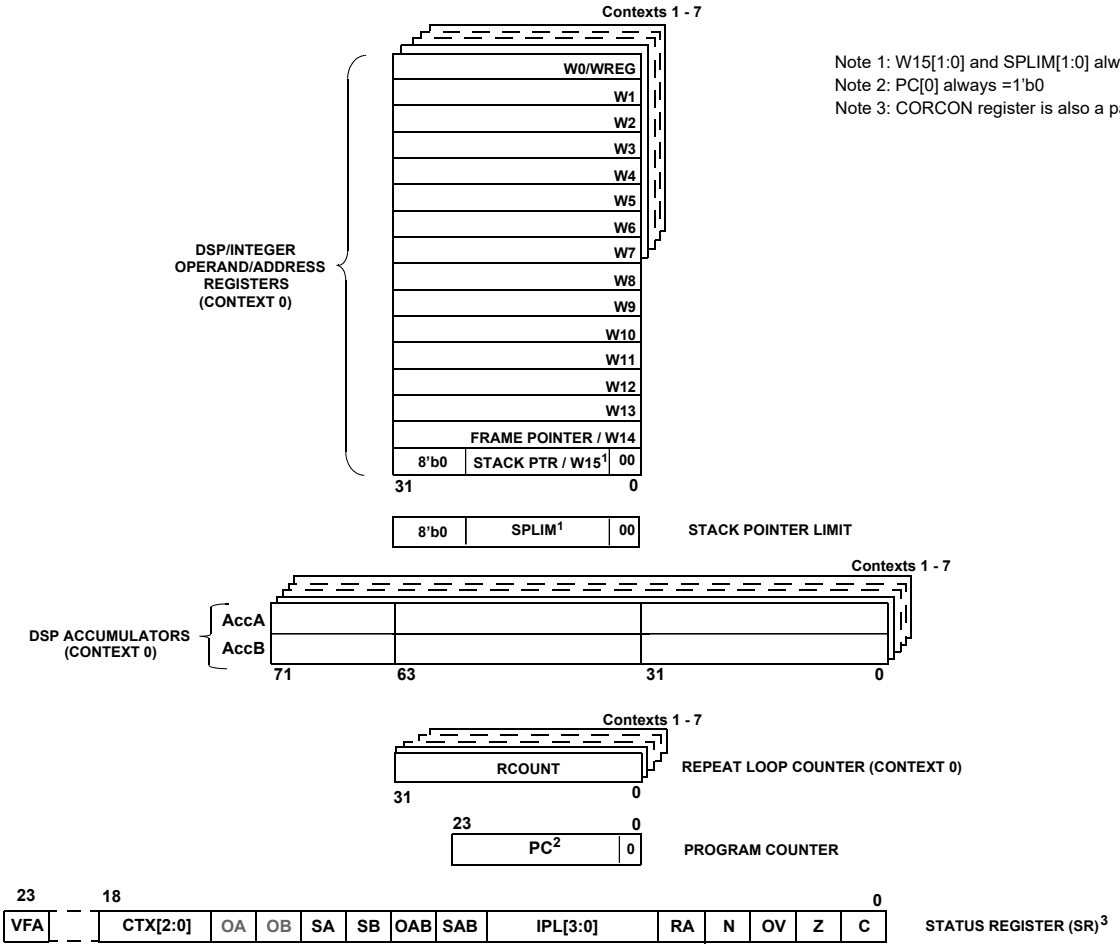
Table 3-2. Programmer's Model Register Descriptions

Register(s) Name	Description
W0 through W15(1)	Working Register Array (Default Context)
W0 through W7(1,2)	Working Register Array (Alternate Context 1-7)
ACCA,ACCB(1)	72-bit DSP Accumulators (Context 0-7)
PC	24-bit Program Counter
SR(1)	ALU and DSP Engine Status Register
SPLIM	Stack Pointer Limit Value Register
RCOUNT	`32-bit REPEAT` Loop Count Register (Context 0-7)
CORCON	DSP Engine Configuration

Notes:

1. W0 through W15, ACCx and SR are not mapped to memory.
2. W0 through W7 are part of Alternate W register sets.

Figure 3-2. dsPIC33A CPU Programmer's Model



3.3.5 DSP Engine and Instructions

The DSP engine features:

- A high-speed, 33-bit by 33-bit multiplier
- A 72-bit ALU
- Two 72-bit saturating accumulators
- A 72-bit bidirectional barrel shifter, capable of shifting a 40-bit value up to 32 bits right, or up to 32 bits left, in a single cycle

The DSP instructions operate seamlessly with all other instructions and are designed for optimal real-time performance. The `MAC` instruction, and other associated instructions, can concurrently fetch two data operands from memory while multiplying two W registers. This requires that the data space be split for these instructions and linear for all others.

3.3.6 Exception Processing

The dsPIC33A devices have a vectored exception scheme, with up to eight possible sources of non-maskable traps and up to 246 possible interrupt sources. Each interrupt source can be assigned to one of seven priority levels.

In addition, each of the Alternate W register contexts can be associated with its own Interrupt Priority Level (IPL) for exception handling. See 3.3.9. Alternate Working Register Arrays for more information.

3.3.7 CPU Register Descriptions

3.3.7.1 SR: CPU STATUS Register

The dsPIC33A CPU has a 32-bit STATUS Register (SR). A detailed description of the CPU SR is shown in 3.2.17. SR.

SR contains:

- All ALU Operation Status flags
- The CPU Interrupt Priority Level Status bits, IPL[3:0]
- The `REPEAT` Loop Active Status bit, RA (SR[4])
- The DSP Adder/Subtracter Status bits

The SR bits are readable/writable with the following exceptions:

- The RA bit (SR[4]) is read-only
- The OA, OB (SR[15:14]), OAB (SR[11]), SA, SB (SR[13:12]) and SAB (SR[10]) bits are readable and writable; however, once set, they remain set until cleared by the user application, regardless of the results from any subsequent DSP operations.
  Note: Clearing the SAB bit also clears both the SA and SB bits. Similarly, clearing the OAB bit also clears both the OA and OB bits. A description of the STATUS Register bits affected by each instruction is provided in the *"dsPIC33A Programmer's Reference Manual"*.
- The CTX bit (SR[18:16]) is read-only; it reflects which W register context is currently in use by the CPU
- The VF bit (SR[23]) is read-only

3.3.7.2 CORCON: Core Control Register

The Core Control register (CORCON) has bits that control the operation of the DSP multiplier.

3.3.8 Working Register Array

The Working (W) registers can function as data, address or address offset registers. The function of a W register is determined by the addressing mode of the instruction that accesses it.

The dsPIC33A instruction set can be divided into two instruction types: Register instructions and File register instructions.

3.3.8.1 Register Instructions

Register instructions can use each W register as a data value or an address offset value. Example 3-1 shows register instructions.

Example 3-1. Register Instructions

```
MOV.w    W0, W1              ; move contents of W0 to W1
MOV.w    W0, [W1]            ; move W0 to address contained in W1
ADD.w    W0, [W4], W5        ; add contents of W0 to contents pointed
                           ; to by W4. Place result in W5.
```

3.3.8.2 File Register Instructions

File register instructions operate on a specific memory address contained in the instruction opcode and register, W0. W0 is a special Working register used in File register instructions.

The File register address space is determined by the maximum address range of the file instructions, which is either 64 KB (if a W-reg operand is required) or 1 MB (if no W-reg operand is required), and encompasses the user RAM area and Special Function Registers (SFRs) within DS.

Example 3-2 shows File register instructions.

Example 3-2. File Register Instructions

```
ADD.w 0x4500, Wn                ; (0x4500)+w0 -> 0x4500
ADD.w 0x4500, w0, Wn                        ; (0x4500)+w0 -> 0x4500
ADD.w 0x4500, w4, Wn                        ; (0x4500)+w4 -> 0x4500
```

3.3.8.3 W Register Memory Mapping

The W registers are not memory-mapped, and thus, it is not possible to access a W register in a File register instruction. This helps in eliminating data hazards.

3.3.8.4 W Registers and Byte Mode Instructions

Byte instructions that target the W register array affect only the Least Significant Byte (LSB) of the target register. Since the Working registers are memory-mapped, the LSB and the Most Significant Byte (MSB) can be manipulated through byte-wide data memory space accesses.

3.3.9 Alternate Working Register Arrays

Alternate Working register arrays are a subset of the Working registers (W0 through W7). Depending on the specific device, up to seven Alternate Working register arrays may be implemented. Each set implements registers W0 through W7, AccA, AccB, RCOUNT, and DSP related CORCON control bits (US, SATA, SATB, SATDW, ACCSAT, RND, IF).

The Alternate W registers are not memory-mapped to data memory space just like the default W array.

All W register arrays are persistent; that is to say, the contents of the default and Alternate W registers do not change whenever the CPU switches to another set. This saves time by reducing the amount of saving and restoring of register contents, making this very useful for time-critical applications.

Each Alternate W array is inherently assigned to a respective IPL (e.g., IPL4 is assigned to Context 4) and Interrupt Service Routine (ISR) in the application code. The Current Context Identifier (CTX[2:0]) status field is located within the Status Register (SR). Each context is associated with a specific Interrupt Priority Level (IPLV). The context is exited during execution of RETFIE instruction of the interrupt ISR.

During an exception processing, the (CTX[2:0]) status field located within the Status Register (SR) is stacked. The stacked SR.CTX[2:0] represents the CPU register context in use at the time of the exception. The value is updated whenever the register context is changed, either through automatic interrupt-based hardware switching, or as the result of a context change brought about by the execution of a CTXTSWP{W} instruction.

Depending on the device, different context Working register behavior can be observed with nested interrupts.

Consider the example, as shown in Figure 3-3, where there are nested interrupts. In this case, the system is configured as follows:

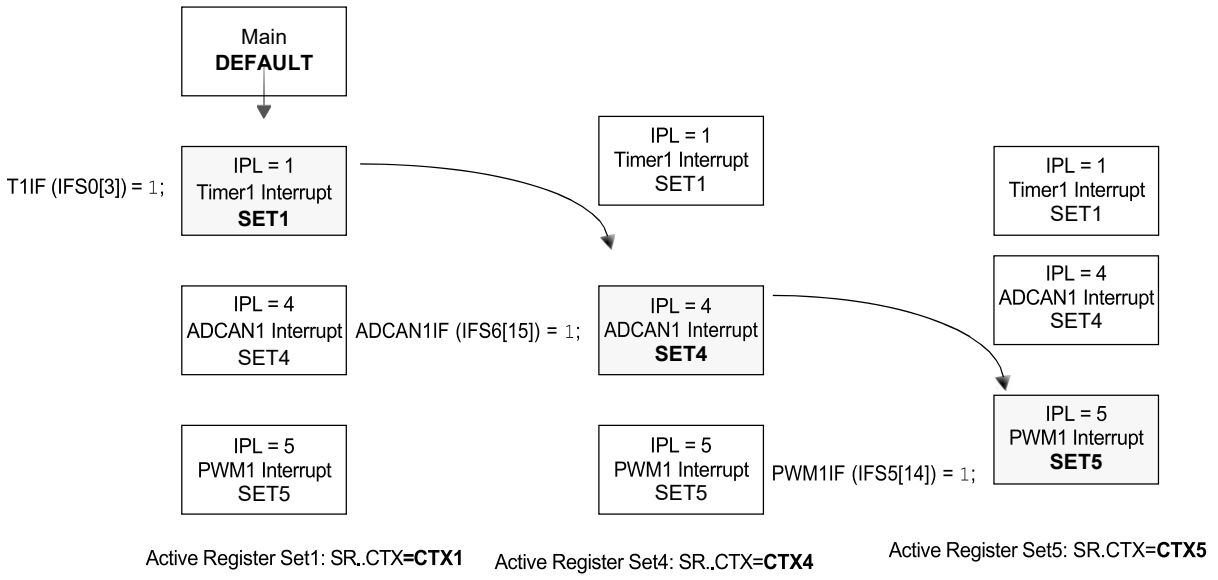
- Timer1 interrupt with an Interrupt Priority Level (IPL) of 1. The Alternate Working Register Set 1 (CTX1) has an IPL of 1.
- ADCAN1 interrupt with an IPL of 4. The Alternate Working Register Set 4 (CTX4) has an IPL of 4.
- PWM1 interrupt with an IPL of 5, The Alternate Working Register Set 5 (CTX5) has an IPL of 5.

The application begins in the main function. At some point in time, the Timer1 interrupt flag is set and the program jumps to the Timer1 ISR. The register set switches from the default Working register set 0 to the Alternate Working register set 1, CTX1. At some point during the Timer1 ISR, the ADCAN1 conversion completes, and its interrupt flag is set. Because it has a higher IPL, the program jumps to the ADCAN1 ISR. The register set switches from the set 1, CTX1 Alternate Working register set to the Alternate Working register set 4, CTX4. At some point during the ADCAN1 ISR, the PWM1

interrupt flag is set. Because the PWM1 IPL is higher than the ADCAN1 IPL, the program jumps to the PWM1 ISR and remains in the Alternate Working register set 5 CTX5.

Once the PWM ISR execution is completed, the program jumps back to the ADCAN1 ISR using CTX4. Similarly, after the execution of the ADCAN1 ISR, the program jumps back to the Timer1 ISR using CTX1. Exceptions above IPL7 (i.e., traps) will execute in whatever register context the CPU was in prior to the trap event.

Figure 3-3. Nested Interrupt Context Flow for dsPIC33A Devices



3.3.9.1 Alternate Working Register Set

Alternatively, before enabling interrupts associated with a particular context, the application may manually switch to it by executing the `CTXTSWP` instruction. `CTXTSWP` does not affect the CPU IPL; it is used to support software context switching for either context initialization, run-time usage of contexts within procedure calls or the like, thus operating independently from the interrupt system.

3.3.10 Software Stack Pointer

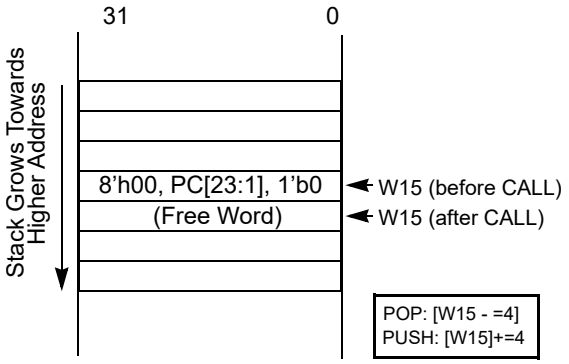
The W15 register serves as a dedicated Software Stack Pointer (SSP) and is automatically modified by exception processing, subroutine calls and returns; however, W15 can be referenced by any instruction in the same manner as all other W registers. This simplifies reading, writing and manipulating the Stack Pointer (for example, creating stack frames).
Note: To protect against misaligned stack accesses, W15[1:0] is fixed to '00' by the hardware.

W15 is initialized to 0x4000 during all Resets. This address ensures that the Software Stack Pointer points to valid RAM in all dsPIC33A devices and permits stack availability for non- maskable trap exceptions. These can occur before the SSP is initialized by the user software. Reprogramming the SSP to any location within data space is possible during initialization.

The Software Stack Pointer always points to the first available free word in the data space (RAM) and fills the software stack, working from lower toward higher addresses. Figure 3-4 illustrates how it pre-decrements for a stack pop (read) and post-increments for a stack push (writes).

When the PC is pushed onto the stack, PC[23:0] are pushed onto the first available stack word, as shown in Figure 3-4.

Figure 3-4. Stack Operation for a CALL Instruction



3.3.10.1 Software Stack Examples

The software stack is manipulated using the PUSH and POP instructions. The PUSH and POP instructions are the equivalent of a MOV instruction with W15 as the Destination Pointer. For example, the contents of W0 can be pushed onto the stack by:

```
PUSH W0
```

This syntax is equivalent to:

```
MOV.L W0,[W15++]
```

The contents of the Top-of-Stack (TOS) can be returned to W0 by:

```
POP W0
```

This syntax is equivalent to:

```
MOV.L [--W15],W0
```

Figure 3-5 through Figure 3-8 illustrate examples of how the software stack is used. Figure 3-5 illustrates the software stack at device initialization. W15 has been initialized to 0x00004000. This example assumes the values, 0xAAAAAAAA and 0xBBBBBBBB, have been written to W0 and W1, respectively. In Figure 3-6, the stack is pushed for the first time and the value contained in W0 is copied to the stack. W15 is automatically updated to point to the next available stack location (0x00004004). In Figure 3-7, the contents of W1 are pushed onto the stack. Figure 3-8 illustrates how the stack is popped and the Top-of-Stack value (previously pushed from W1) is written to W3.

Figure 3-5. Stack Pointer at Device Reset

Figure 3-6. Stack Pointer After the First PUSH Instruction

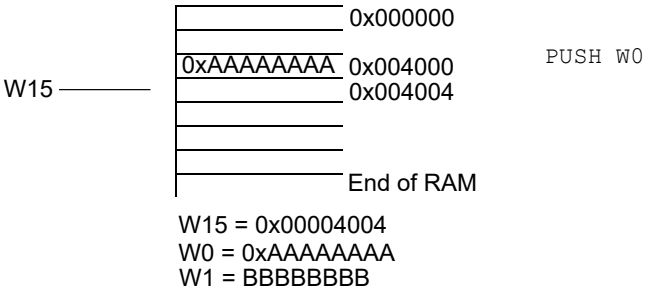


Figure 3-7. Stack Pointer After the Second PUSH Instruction

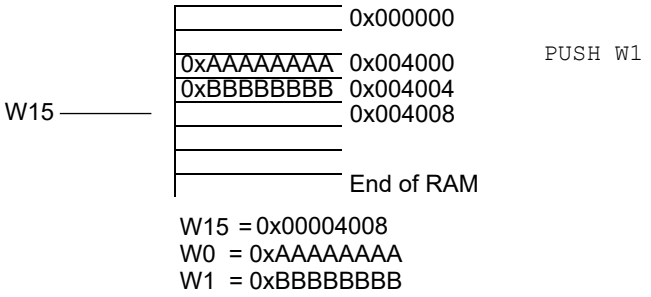
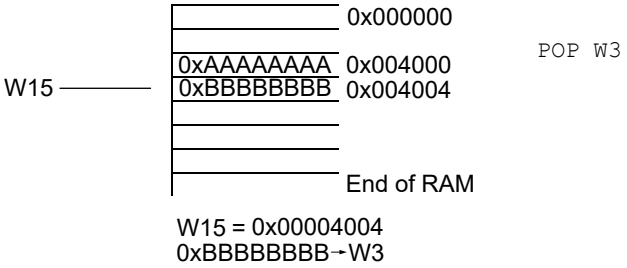


Figure 3-8. Stack Pointer After a POP Instruction



3.3.10.2 W14 Software Stack Frame Pointer

A frame is a user-defined section of memory in the stack that is used by a single function. The Working register, W14, can be used as a Stack Frame Pointer with the LNK (link) and ULNK (unlink) instructions. W14 can be used in a normal Working register by instructions when it is not used as a Frame Pointer.

3.3.10.3 Stack Pointer Overflow

The Stack Pointer Limit (SPLIM) register specifies the size of the stack buffer. SPLIM is a 32-bit register, but SPLIM[1:0] is fixed to '00' because all stack operations must be long word-aligned.

The stack overflow check is not enabled until a Long word write to SPLIM occurs. After this, it can only be disabled by a device Reset. All Effective Addresses (EAs), generated using W15 as a source or destination, are compared against the value in SPLIM. If Effective Addresses (EAs) exceed the contents of the SPLIM register, and a PUSH operation is performed, a stack error trap occurs on a subsequent PUSH operation. For example, if it is desirable to cause a stack error trap when the stack grows beyond address 0x5000 in RAM, initialize the SPLIM with the value 0x4FFC.

Note: A stack error trap can be caused by any instruction that uses the contents of the W15 register to generate an Effective Address (EA). Therefore, if the contents of W15 are greater than the contents of the SPLIM register by a value of four, and a `CALL` instruction is executed or if an interrupt occurs, a stack error trap is generated.

If stack overflow checking is enabled, a stack error trap also occurs if the W15 Effective Address calculation wraps over the end of data space.

A pre/post inc/dec operation is performed on W15 that results in EA[1:0] != 2'b00 (i.e., not long word aligned). This will detect byte and word pre/post inc/dec operations that are otherwise considered aligned but would result in a misaligned Stack Pointer.

Note: A write to the SPLIM should not be followed by an indirect read operation using W15.

3.3.10.4 Stack Pointer Underflow

The stack is initialized to 0x4000 during a Reset. A stack error trap is initiated if the Stack Pointer address is less than 0x4000.

Note: Locations in data space between 0x0000 and 0x3FFF are, in general, reserved for core and peripheral Special Function Registers (SFRs).

3.3.11 Arithmetic Logic Unit (ALU)

The dsPIC33A ALU is 32 bits wide and is capable of addition, subtraction, single bit shifts and logic operations. Unless otherwise mentioned, arithmetic operations are 2's complement in nature. Depending on the operation, the ALU can affect the values of the following bits in the STATUS Register:

- Carry (C)
- Zero (Z)
- Negative (N)
- Overflow (OV)

The ALU can perform 8/16-bit or 32-bit operations, depending on the mode of the instruction that is used. Data for the ALU operation can come from the W register array or data memory depending on the addressing mode of the instruction. Likewise, output data from the ALU can be written to the W register array or a data memory location.

Note:

1. Byte operations use the 16-bit ALU and can produce results in excess of eight bits. However, to maintain backward compatibility with PIC® MCU devices, the ALU result from all byte operations is written back as a byte (i.e., the MSB is not modified) and the STATUS Register is updated based only upon the state of the LSB of the result.

3.3.11.1 Byte to Word Conversion

The dsPIC33A CPU has two instructions that are helpful when mixing 8-bit and 16-bit ALU operations.

The Sign-Extend (`SE`) instruction takes a byte value in a W register or data memory and creates a sign-extended word value that is stored in a W register.

The Zero-Extend (`ZE`) instruction clears the 8 MSbs of a word value in a W register or data memory and places the result in a destination W register.

3.3.12 DSP Engine

The DSP engine is a block of hardware that is fed data from the W register array, but contains its own specialized result registers. The DSP engine is driven from the same instruction decoder that directs the MCU ALU. In addition, all operand Extended Addresses (EAs) are generated in the W register array. Concurrent operation with MCU instruction flow is not possible, though both the MCU ALU and DSP engine resources can be shared by all instructions in the instruction set.

The DSP engine consists of the following components:

- High-speed, 33-bit by 33-bit multiplier
- Barrel shifter
- 72-bit adder/subtractor
- Two target Accumulator registers
- Rounding logic with selectable modes
- Saturation logic with selectable modes

Data input to the DSP engine is derived from one of the following sources:

- Directly from the W array for dual source operand DSP instructions. Data values fetched via the X and Y memory data buses.
- From the X memory data bus for all other DSP instructions.

Data output from the DSP engine is written to one of the following destinations:

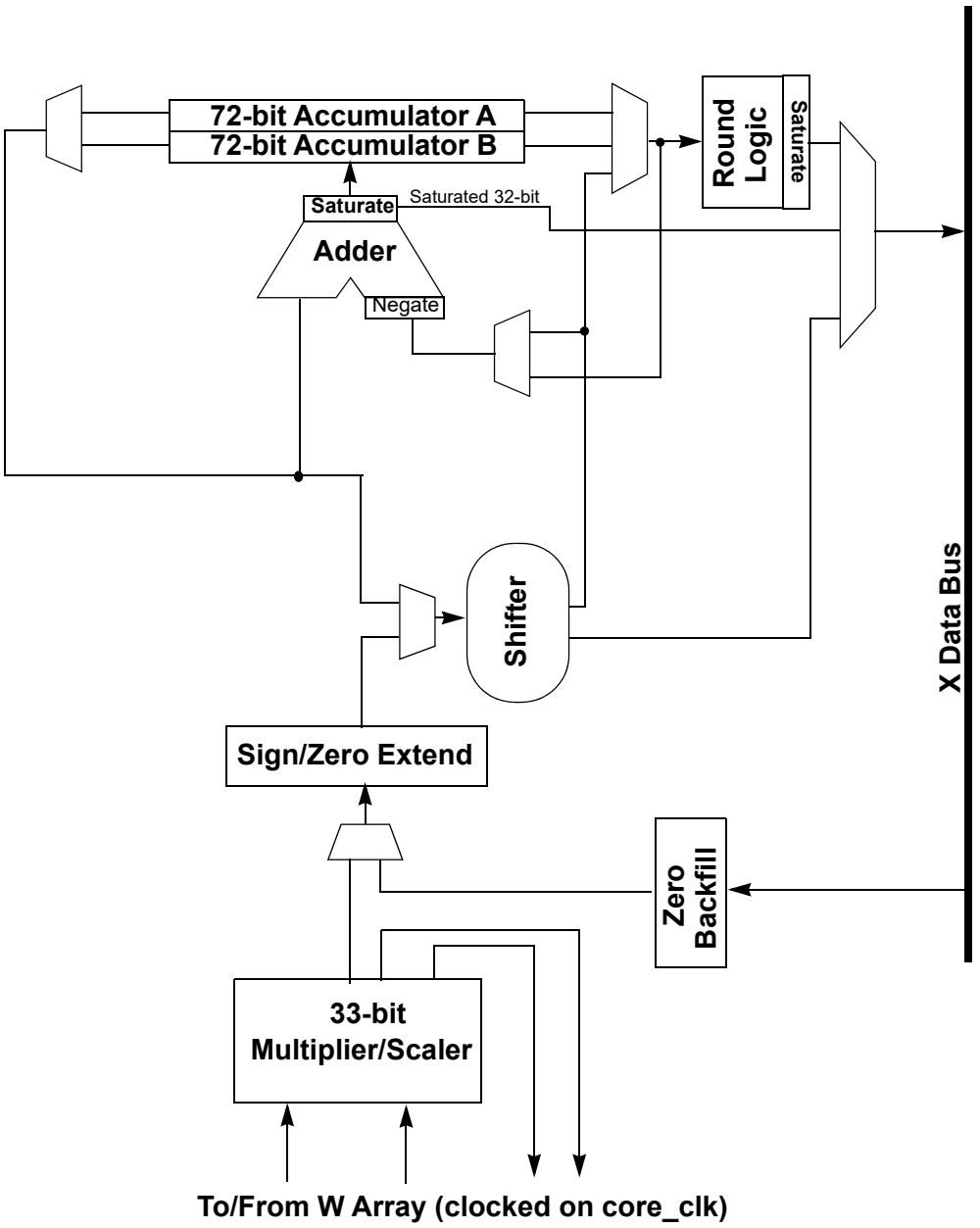
- The target accumulator, as defined by the DSP instruction being executed.
- The X memory data bus to any location in the data memory address space.

The DSP engine can perform inherent accumulator-to-accumulator operations that require no additional data.

The MCU shift and multiply instructions use the DSP engine hardware to obtain their results. The X memory data bus is used for data reads and writes in these operations.

Figure 3-9 illustrates a block diagram of the DSP engine.

Figure 3-9. DSP Engine Block Diagram



3.3.12.1 Data Accumulators

Two 72-bit data accumulators, ACCA and ACCB, are the Result registers for the DSP instructions listed in 3.3.12.2.1. DSP Multiply Instructions. Each accumulator is not memory-mapped and is referred as these three registers, where 'x' denotes the particular accumulator:

- ACCxL: ACCx[31:0]
- ACCxH: ACCx[63:32]
- ACCxU: ACCx[71:64]

For fractional operations that use the accumulators, the radix point is located to the right of bit 31. The range of fractional values that can be stored in each accumulator is -256 to +(256 - 2**-63).

For integer operations that use the accumulators, the radix point is located to the right of bit 0. The range of integer values that can be stored in each accumulator is -0x80_00000000_00000000 to 0x7F_FFFF_FFFF_FFFF_FFFF.

3.3.12.2 Multiplier

The dsPIC33A devices feature a 33-bit-by-33-bit multiplier shared by both the MCU ALU and the DSP engine. The multiplier is capable of signed, unsigned or mixed-sign operation and supports either 9.31 fractional (Q.31) or 64-bit integer results.

The multiplier takes in 32-bit input data and converts the data to 33 bits. Signed operands to the multiplier are sign-extended. Unsigned input operands are zero-extended. The internal 33-bit representation of data in the multiplier allows correct execution of mixed-sign and unsigned 32-bit by 32-bit multiplication operations.

The representation of data in hardware for Integer and Fractional Multiplier modes is as follows:

- Integer data is inherently represented as a signed two's complement value, where the Most Significant bit (MSb) is defined as a Sign bit. Generally speaking, the range of an N-bit two's complement integer is $-2^{(N-1)}$ to $2^{(N-1)}-1$.
- Fractional data is represented as a two's complement fraction, where the MSb is defined as a Sign bit and the radix point is implied to lie just after the Sign bit (Q.X format). The range of an N-bit two's complement fraction with this implied radix point is -1.0 to $(1 - 2^{(1-N)})$.

The range of data in both Integer and Fractional modes is listed in Table 3-3. Figure 3-10 and Figure 3-11 illustrate how the multiplier hardware interprets data in Integer and Fractional modes.

The Integer or Fractional Multiplier Mode Select (IF) bit (CORCON[0]) determines integer/ fractional operation for the instructions listed in Table 3-4. The IF bit does not affect MCU multiply instructions listed in Table 3-5, which are always integer operations. The multiplier scales the result one bit to the left for fractional operation. The LSb of the result is always cleared. The multiplier defaults to Fractional mode for DSP operations at a device Reset.

Table 3-3. dsPIC33A Data Ranges

Register Size	Integer Range	Fraction Range	Fraction Resolution
16-Bit	-32768 to 32767	-1.0 to $(1.0 - 2^{-15})$ (Q1.15 Format)	3.052×10^{-5}
32-Bit	-2,147,483,648to 2,147,483,647	-1.0 to $(1.0 - 2^{-31})$ (Q1.31 Format)	4.657×10^{-10}
64-Bit	-9.223372037e18 to 9.223372037e18	-1.0 to $(1.0 - 2^{-63})$ (Q.1.63 Format)	1.08420×10^{-19}
72-Bit	-2.361183241e21 to 2.361183241e21	-256.0 to $(256.0 - 2^{-63})$ (Q.9.63 Format with 8 Guard bits)	1.08420×10^{-19}

Figure 3-10. Integer and Fractional Representation of 0x40000001

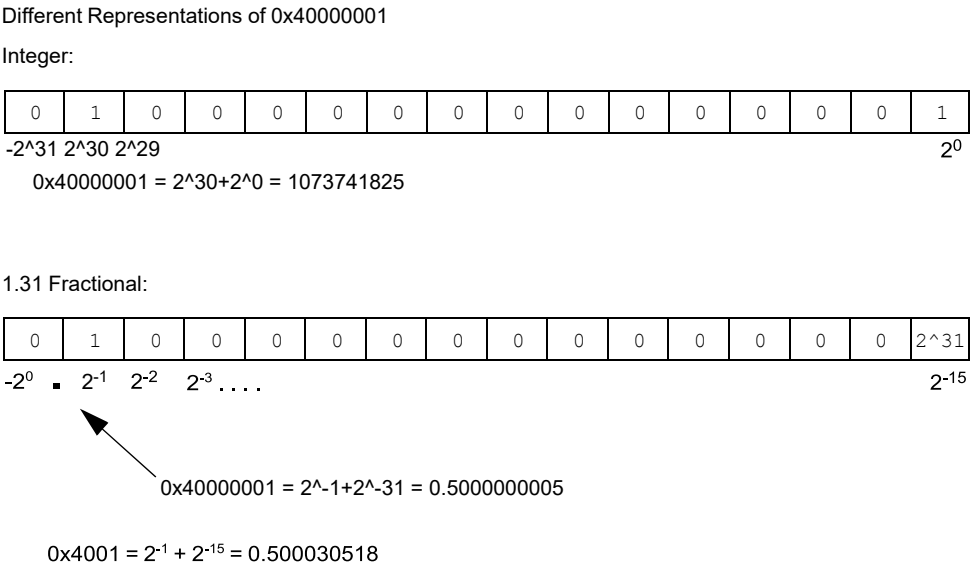
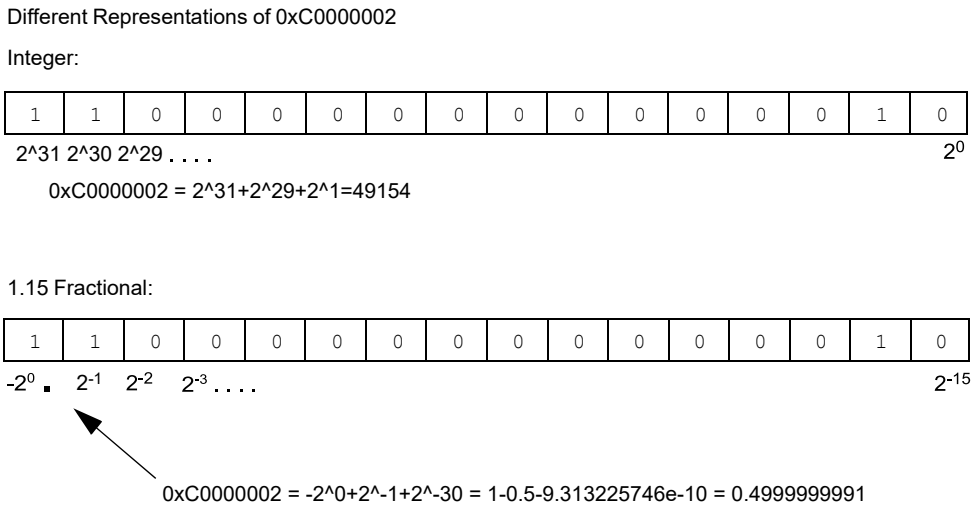


Figure 3-11. Integer and Fractional Representation of 0xC0000002



3.3.12.2.1 DSP Multiply Instructions

The DSP instructions that use the multiplier are summarized in Table 3-4.

Table 3-4. DSP Instructions that Use the Multiplier

DSP Instruction(1)	Description	Algebraic Equivalent
MAC	Multiply and Add to Accumulator or Square and Add to Accumulator	*a = a + b * c a = a + b2*
MSC	Multiply and Subtract from Accumulator	*a = a – b * c*
MPY	Multiply	*a = b * c*

Note:
1. DSP instructions using the multiplier can operate in signed or unsigned Fractional (1.15/1.31) or Integer modes.

.....continued

DSP Instruction(1)	Description	Algebraic Equivalent
MPYN	Multiply and Negate Result	*a = -b * c*
SQR	Square to Accumulator	*a = b ^ 2*
SQRAC	Square and Accumulate	*a = a + (b ^ 2)*
ED	Partial Euclidean Distance	$a = (b - c)^2$
EDAC	Add Partial Euclidean Distance to the Accumulator	$a = a + (b - c)^2$

Note:

1. DSP instructions using the multiplier can operate in signed or unsigned Fractional (1.15/1.31) or Integer modes.

The DSP Multiplier Unsigned/Signed Control (US) bits (CORCON[12]) determine whether the DSP multiply instructions are signed (default) or unsigned. The US bits do not influence the MCU multiply instructions, which have specific instructions for signed or unsigned operation. If the USx bits are set to '01', the input operands for instructions shown in Table 3-4 are considered as unsigned values, which are always zero-extended into the 33rd bit of the multiplier value. If the USx bits are set to '00', the operands are sign-extended.

If the USx bits (CORCON[13:12]) are set to '10', the operands for the instructions listed above are considered as unsigned values. The result is zero-extended prior to any operation with the accumulator (which will always effectively be signed).

3.3.12.2.2 MCU Multiply Instructions

The same multiplier supports the MCU multiply instructions, which include integer, 32-bit signed, unsigned and mixed-sign multiplies, as shown below. All multiplications performed by the MUL instruction produce integer results. The MUL instruction can be directed to use byte or word- sized operands. Byte input operands produce a 16-bit result and word input operands produce either a 16-bit result or a 32-bit result, either to the specified register(s) in the W array or to an accumulator. Word input operands produce a 32-bit result and word input operands produce either a 32-bit result or a 64-bit result, either to the specified register(s) in the W array or to an accumulator.

Table 3-5. MCU Instructions that Utilize the Multiplier

MCU Instruction(1)	Description
MUL/MUL.UU	Multiply two unsigned integers and generate 64-bit results.
MUL.SS	Multiply two signed integers and generate 64-bit results.
MUL.SU/MUL.US	Multiply a signed integer with an unsigned integer and generate 64-bit results.
MULD.UU	Multiply two unsigned integers and generate 72-bit results.
MULD.SS	Multiply two signed integers and generate 72-bit results.
MULD.SU/ MULD.US	Multiply a signed integer with an unsigned integer and generate a 72-bit result.
MULW.UU	Multiply two unsigned integers and generate 32-bit results.
MULW.SS	Multiply two signed integers and generate 32-bit results.
MULW.SU/MULW.US	Multiply a signed integer with an unsigned integer and generate a 32-bit result.
MULB	Multiply two unsigned 8-bit integers and generate 16-bit results.
MULW	Multiply two unsigned 16-bit integers and generate 32-bit results.
MULL	Multiply two unsigned 32-bit integers and generate 32-bit results.

Note:

1. MCU instructions using the multiplier operate only in Integer mode.

3.3.12.3 Data Accumulator Adder/Subtractor

The data accumulators have a 72-bit adder/subtractor with automatic sign extension or zero extension logic for the multiplier result. It can select one of two accumulators (A or B) as its pre-accumulation source and post-accumulation destination. For the `ADD` (accumulator) and `LAC` instructions, the data to be accumulated or loaded can optionally be scaled via the barrel shifter prior to accumulation.

The 72-bit adder/subtractor can optionally negate one of its operand inputs to change the sign of the result (without changing the operands). The negate is used during multiply and subtract (`MSC`) or multiply and negate (`MPYN`) operations.

The 72-bit adder/subtractor has an additional saturation block that controls accumulator data saturation, if enabled.

3.3.12.3.1 Accumulator Status Bits

Six STATUS Register bits that support saturation and overflow are located in the CPU STATUS Register (SR) and are listed in Table 3-6.

Table 3-6. Accumulator Overflow and Saturation Status Bits

Status Bit (SR Location)	Description
OA ([15])	Accumulator A overflowed into guard bits (ACCA[71:63])
OB ([14])	Accumulator B overflowed into guard bits (ACCB[71:63])
SA ([13])	ACCA saturated (bit 63 overflow and saturation) or ACCA overflowed into guard bits and saturated (bit 71 overflow and saturation)
SB ([12])	ACCB saturated (bit 63 overflow and saturation) or ACCB overflowed into guard bits and saturated (bit 71 overflow and saturation)
OAB ([11])	OA logically ORed with OB, clearing OAB clears both OA and OB
SAB ([10])	SA logically ORed with SB, clearing SAB clears both SA and SB

The OA and OB bits are modified each time data passes through the accumulator add/subtract logic. When set, they indicate that the most recent operation has overflowed into the accumulator guard bits (ACCx[71:64]). This type of overflow is not catastrophic; the guard bits preserve the accumulator data. The OAB Status bit is the logically OR value of OA and OB.

The OA and OB bits, when set, can optionally generate an arithmetic error trap. The trap is enabled by setting the corresponding Overflow Trap Flag Enable bit (OVATE or OVBTE) in Interrupt Control Register 4 (INTCON4[10:9]) in the interrupt controller. The trap event allows the user to take immediate corrective action, if desired.

The SA and SB bits can be set each time data passes through the accumulator saturation logic. Once set, these bits remain set until cleared by the user application. The SAB Status bit indicates the logical OR value of SA and SB. When set, these bits indicate that the accumulator has overflowed its maximum range (bit 63 for 64-bit saturation or bit 71 for 72-bit saturation) and are saturated (if saturation is enabled).

When saturation is not enabled, the SA and SB bits indicate that a catastrophic overflow has occurred (the sign of the accumulator has been destroyed). If the Catastrophic Overflow Trap Enable (COVTE) bit (INTCON4[8]) is set, SA and SB bits will generate an arithmetic error trap when saturation is disabled. The SA and SB bits can be set in software, enabling efficient context state switching.

3.3.12.3.2 Saturation And Overflow Modes

The dsPIC33A CPU supports three Saturation and Overflow modes.

- **Accumulator 71-Bit Saturation**
  In this mode, the saturation logic loads the maximally positive 9.63 value (0x7F_FFFF_FFFF_FFFF_FFFF) or maximally negative 9.63 value (0x80_0000_0000_0000_0000) into

the target accumulator. The SA or SB bit is set and remains set until cleared by the user application. This Saturation mode is useful for extending the dynamic range of the accumulator.

To configure for this mode of saturation, set the Accumulator Saturation Mode Select (ACCSAT) bit (CORCON[4]). Additionally, set the ACCA Saturation Enable (SATA) bit (CORCON[7] and/or the ACCB Saturation Enable (SATB) bit (CORCON[6]) to enable accumulator saturation.

- **Accumulator 63-Bit Saturation**
  In this mode, the saturation logic loads the maximally positive 1.63 value (0x00_7FFF_FFFF_FFFF_FFFF) or maximally negative 1.63 value (0xFF_8000_0000_0000_0000) into the target accumulator. The SA or SB bit is set and remains set until cleared by the user. When this Saturation mode is in effect, the guard bits, 64 through 71, are not used except for sign extension of the accumulator value. Consequently, the OA, OB or OAB bits in SR are never set.

  To configure for this mode of overflow and saturation, the ACCSAT (CORCON[4]) bit must be cleared. Additionally, the SATA (CORCON[7]) and/or SATB (CORCON[6]) bits must be set to enable accumulator saturation.

- **Accumulator Catastrophic Overflow**
  If the SATA (CORCON[7]) and/or SATB (CORCON[6]) bits are not set, then no saturation operation is performed on the accumulator, and the accumulator is allowed to overflow all the way up to bit 71 (destroying its sign). If the Catastrophic Overflow Trap Enable (COVTE) bit (INTCON4[8] in the interrupt controller) is set, a catastrophic overflow initiates an arithmetic error trap.

Accumulator saturation and overflow detection can only result from the execution of a DSP instruction that modifies one of the two accumulators via the 72-bit DSP ALU. Saturation and overflow detection do not take place when the accumulators are accessed via the MCU class of instructions. Furthermore, the Accumulator Status bits shown in Table 3-6 are not modified. However, the MCU Status bits (Z, N, C, OV, DC) will be modified, depending on the MCU instruction that accesses the accumulator.

3.3.12.3.3 Data Space Write Saturation

In addition to adder/subtractor saturation, writes to data space can be saturated without affecting the contents of the source accumulator. This feature allows data to be limited, while not sacrificing the dynamic range of the accumulator during intermediate calculation stages. Data space write saturation is enabled by setting the data space write from the DSP Engine Saturation Enable (SATDW) Control bit (CORCON[5]). Data space write saturation is enabled by default at a device Reset.

The data space write saturation feature works with the `SAC` and `SACR` instructions. The value held in the accumulator is never modified when these instructions are executed. The hardware takes the following steps to obtain the saturated write result:

1. The read data is scaled based upon the arithmetic shift value specified in the instruction.
2. The scaled data is rounded (`SACR` only).
3. For Word mode instruction, scaled/rounded value is saturated to a 16-bit result based on the value of the guard bits. For data values greater than 0x007FFF, the data written to memory is saturated to the maximum positive 1.15 value, 0x7FFF. For input data less than 0xFF8000, data written to memory is saturated to the maximum negative 1.15 value, 0x8000. Similarly, the data written to memory is saturated to maximum positive/negative 1.31 value for Long Word mode operation.

3.3.12.3.4 Accumulator Write Back

The `MAC` and `MSC` instructions can optionally write a rounded version of the accumulator that is not the target of the current operation into data space memory. The write is performed across the X-bus into the combined X and Y address space. This accumulator write-back feature is beneficial in certain algorithms, such as FFT and LMS filters.

Two addressing modes are supported by the accumulator write-back hardware:

- W0, W1, W2, W3 or W13, Register Direct: The rounded contents of the non-target accumulator are written into the destination register as a 1.15 (Word mode) or 1.31 (Long Word mode) fractional result.
- [W13++] or [W15++], Register Indirect with Post-Increment: The rounded contents of the non-target accumulator are written into the address pointed to by W13 or W15 as a 1.15 (Word mode) or 1.31 (Long Word mode) fraction. W13 or W15 is then incremented by 2/4 depending on selected Word/Long Word mode. [W15++] is equivalent to a push onto the system stack.

3.3.12.4 Round Logic

The round logic can perform a conventional (biased) or convergent (unbiased) round function during an accumulator write (store). The Round mode is determined by the state of the Rounding Mode Select (RND) bit (CORCON[1]). It generates a 16-bit 1.15 or 32-bit 1.31 data value, which is passed to the data space write saturation logic. If rounding is not indicated by the instruction, a truncated 1.15 or 1.31 data value is stored.

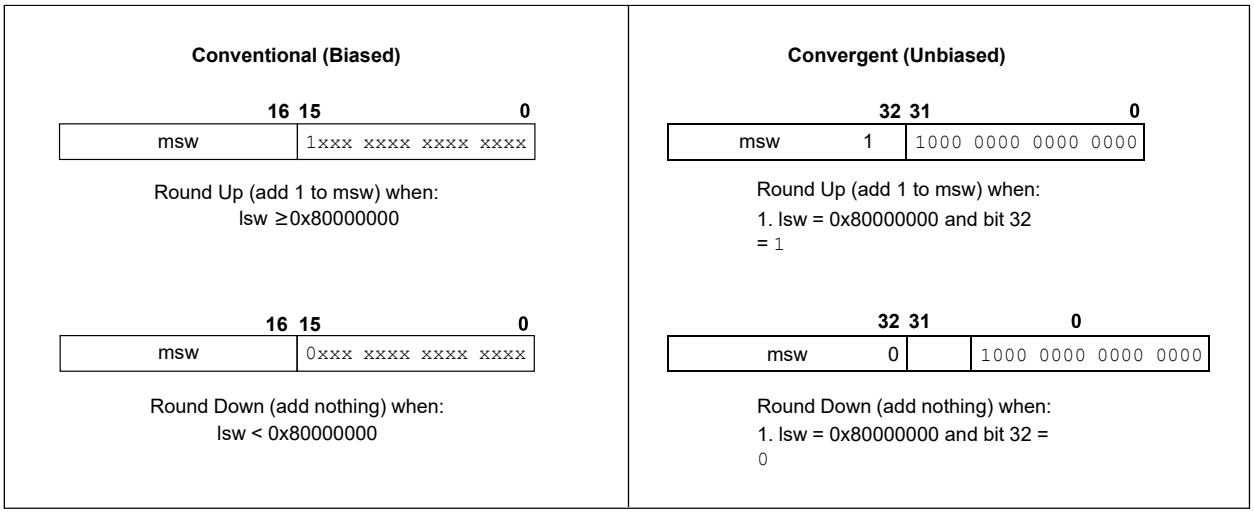
The two Rounding modes are shown in Figure 3-12. Conventional rounding takes bit 31 of the accumulator, zero-extends it and adds it to the most significant word (msw), excluding the guard or overflow bits (bits 32 through 63). If the least significant word (lsw) of the accumulator is between 0x80000000 and 0xFFFFFFFF (0x80000000 included), the msw is incremented. If the lsw of the accumulator is between 0x0000 and 0x7FFFFFFF, the msw remains unchanged. A consequence of this algorithm is that over a succession of random rounding operations, the value tends to be biased slightly positive.

Convergent (or unbiased) rounding operates in the same manner as conventional rounding except when the lsw equals 0x80000000. If this is the case, the LSb of the msw (bit 16 of the accumulator) is examined. If it is '1', the msw is incremented. If it is '0', the msw is not modified. Assuming that bit 16 is effectively random in nature, this scheme removes any rounding bias that may accumulate.

The SAC and SACR instructions store either a truncated (SAC) or rounded (SACR) version of the contents of the target accumulator to data memory via the X-bus (subject to data saturation).

For the MAC class of instructions, the accumulator write-back data path is always subject to rounding. An overflow that occurs as a consequence of a rounding operation will also be subject to saturation.

Figure 3-12. Conventional and Convergent Rounding Modes



3.3.12.5 Barrel Shifter

The barrel shifter can perform up to a 32-bit arithmetic right shift, or up to a 32-bit left shift, in a single cycle. DSP or MCU instructions can use the barrel shifter for multibit shifts.

The shifter requires a signed binary value to determine both the magnitude (number of bits) and direction of the shift operation:

- A positive value shifts the operand right
- A negative value shifts the operand left
- A value of '0' does not modify the operand

The barrel shifter is 72 bits wide to accommodate the width of the accumulators. A 72-bit output result is provided for DSP shift operations, and a 32-bit result is provided for MCU shift operations.

Table 3-7 provides a summary of instructions that use the barrel shifter.

Table 3-7. Instructions that Use the DSP Engine Barrel Shifter

Instruction	Description
ASR	Arithmetic multibit right shift of data memory location
LSR	Logical multibit right shift of data memory location
SL	Multibit shift left of data memory location
SAC	Store DSP accumulator with optional shift
SFTAC	Shift DSP accumulator

3.3.12.6 DSP Engine Mode Selection

These operational characteristics of the DSP engine, discussed in previous sections, can be selected through the CPU Core Configuration register (CORCON):

- Fractional or integer multiply operation
- Conventional or convergent rounding
- Automatic saturation on/off for ACCA
- Automatic saturation on/off for ACCB
- Automatic saturation on/off for writes to data memory
- Accumulator Saturation mode selection

3.3.12.7 DSP Engine Trap Events

Arithmetic error traps that can be generated for handling exceptions in the DSP engine are selected through the Interrupt Control Register 4 (INTCON4). These are:

- Trap on ACCA overflow enable using OVATE (INTCON4[21])
- Trap on ACCB overflow enable using OVBTE (INTCON4[20])
- Trap on catastrophic ACCA and/or ACCB overflow enable using COVTE (INTCON4[19]). Occurrence of the traps is indicated by these error status bits:
  - OVAERR (INTCON4[5])
  - OVBERR (INTCON4[4])
  - COVAERR (INTCON4[3])
  - COVBERR (INTCON4[2])

  An arithmetic error trap is also generated when the user application attempts to shift a value beyond the maximum allowable range (±32 bits) using the SFTAC instruction. This trap source cannot be disabled and is indicated by the Shift Accumulator Error Status (SFTACERR) bit (INTCON4[1] in the interrupt controller). The instruction will execute, but the results of the shift are not written to the target accumulator.

3.3.13 Divide Support

The dsPIC33A CPU supports the following types of division operations:

- DIVF: 16/16 signed fractional divide
- DIVF: 32/16 signed fractional divide
- DIVF.L: 32/32 signed fractional divide
- DIV.SL: 32/32 signed divide
- DIV.UL: 32/32 unsigned divide
- DIV.S: 32/16 signed divide
- DIV.U: 32/16 unsigned divide
- DIV.S: 16/16 signed divide
- DIV.U: 16/16 unsigned divide

The quotient for all divide instructions can placed in any Working register, Wm. The remainder is placed in W(m+1). The 32/16-bit divisor can be located in any W register. A 32/16-bit dividend can be located in any W register. The integer 16/16 divide instructions will either zero or sign extend the least significant dividend word into the most significant dividend word during the first iteration to create a 32-bit dividend.

All 16-bit/16-bit and 32-bit/16-bit divide instructions are iterative operations and must be executed six times within a REPEAT loop. All 32-bit/32-bit divide instructions are iterative operations and must be executed ten times within a REPEAT loop.

The developer is responsible for programming the REPEAT instruction. A complete divide operation takes seven or eleven instruction cycles to execute.

The divide flow is interruptible, just like any other REPEAT loop. All data is restored into the respective data registers after each iteration of the loop, so the user application is responsible for saving the appropriate W registers in the ISR. Although they are important to the divide hardware, the intermediate values in the W registers have no meaning to the user application. The divide instructions must be executed seven or eleven times in a REPEAT loop to produce a meaningful result.

A divide-by-zero error generates a math error trap. This condition is indicated by the Arithmetic Error Status (DIV0ERR) bit (INTCON4[0] in the interrupt controller).

3.3.14 Instruction Flow Types

Most instructions in the dsPIC33A architecture occupy a single word of program memory and execute in a single cycle. However, some instructions take two or more instruction cycles to execute. Consequently, there are seven different types of instruction flow in the dsPIC® DSC architecture.

3.3.15 Loop Constructs

The dsPIC33A CPU supports two REPEAT constructs to provide unconditional automatic program loop control. The REPEAT instruction implements a single instruction program loop. REPEAT instructions use control bits within the CPU STATUS Register (SR) to temporarily modify CPU operation.

3.3.15.1 REPEAT Loop Construct

The REPEAT instruction causes the instruction that follows it to be repeated a specified number of times. A literal value contained in the instruction, or a value in one of the W registers, can be used to specify the REPEAT count value. The W register option enables the loop count to be a software variable.

An instruction in a REPEAT loop is executed at least once. The number of iterations for a REPEAT loop is the 20-bit literal value + 1 or Wn + 1. The syntax for the two forms is shown in 3.3.15.1. REPEAT Loop Construct.

Example 3-3. REPEAT Loop Construct

```
;    Using a literal value as a counter
REPEAT #lit20    ; RCOUNT <-- lit20
(Valid target Instruction)
;
;    Using a W register as a counter
REPEAT Wn    ; RCOUNT <-- Wn
(Valid target Instruction)
```

3.3.15.1.1 REPEAT Operation

The loop count for REPEAT operations is held in the 32-bit Repeat Loop Counter register (RCOUNT), which is memory-mapped. RCOUNT is initialized by the REPEAT instruction. The REPEAT instruction sets the REPEAT Loop Active (RA) Status bit (SR[4]) to '1' if the RCOUNT is a non-zero value.

RA is a read-only bit and cannot be modified through software. For REPEAT loop count values greater than '0', the Program Counter is not incremented. Furthermore, Program Counter increments are inhibited until RCOUNT = 0.

For a loop count value equal to '0', REPEAT has the effect of a NOP and the RA (SR[4]) bit is not set. The REPEAT loop is essentially disabled before it begins, allowing the target instruction to execute only once while pre-fetching the subsequent instruction (i.e., normal execution flow).

Note: The instruction immediately following the REPEAT instruction (i.e., the target instruction) is always executed at least one time and it is always executed one time more than the value specified in the 20-bit literal or the W register operand.

3.3.15.1.2 Interrupting a REPEAT Loop

A REPEAT instruction loop can be interrupted at any time.

The state of the RA bit is preserved on the stack during exception processing to enable the user application to execute further REPEAT loops from within any number of nested interrupts. After SR is stacked, the RA Status bit is cleared to restore normal execution flow within the ISR.

Note: If a REPEAT loop has been interrupted, and an ISR is being processed, the user application must stack the Repeat Count register (RCOUNT) before it executes another REPEAT instruction within an ISR.
If a REPEAT instruction is used within an ISR, the user application must unstack the RCOUNT register before it executes the RETFIE instruction.

Returning into a REPEAT loop from an ISR using the RETFIE instruction requires no special handling. RETFIE pops the PC and that becomes the address of the next instruction to be fetched in its F-stage. The RETFIE instruction is "padded" with FNOPs (2) so the target instruction of the RETFIE PFC can execute as normal.

Early Termination of a REPEAT Loop

An interrupted REPEAT loop can be terminated earlier than normal in the ISR by clearing the RCOUNT register in software.

3.3.15.1.3 Restrictions on the REPEAT Instruction

Any instruction can immediately follow a REPEAT except for the following:

- Program Flow Control instructions (any branch, compare and skip, subroutine calls, returns, etc.)
- Another REPEAT or DTB instruction
- DISICTL, ULNK, LNK, PWRSAV or RESET instruction
  - MOV.D instruction
    Note: Some instructions and/or Instruction Addressing modes can be executed within a REPEAT loop, but it might not make sense to repeat all instructions.

3.3.16 Data Space Address Generation Units (AGUs)

dsPIC33AK128MC106 family devices contain three independent address generator units. The X RAGU and X WAGU support byte (.b), word (.w) and long (.l) word sized data space reads and writes, respectively, for MCU instructions, and word or long word reads and writes for DSP instructions. The Y AGU supports word and long word sized data reads for the DSP MAC-class of instructions only. The AGUs are each capable of supporting two types of data addressing:

- Linear Addressing
- Modulo (circular) Addressing

In addition, the X WAGU can support Bit-Reversed Addressing.

Linear and Modulo Data Addressing modes can be applied to any address within the unified address space. Although Bit- Reversed Addressing will work with any EA calculation, by definition it is only applicable to data space.

Data space memory is organized as 32-bit words; all Effective Addresses (EAs) point to bytes. Instructions can thus access any byte or aligned word (data words at an even byte address) or aligned long word (data words at an even 32-bit word address).

Misaligned accesses are not supported, and if attempted they will initiate an address error trap. The least significant 2 bits of the EA is used to determine the byte or upper/lower 16-bit word access. EA[0] will always be 1'b0 for word accesses, and EA[1:0] will always be 2'b00 for long word accesses.

SFRs and RAM support byte, word, and double word read or write operations.

When executing instructions that require just one source operand to be fetched from (and one result to be written back to) data space, the X RAGU and X WAGU are used to calculate the EAs of the source and destination, respectively. The AGUs can generate an address to point to anywhere in the 16 Mbyte address space. They support all MCU addressing modes and Modulo Addressing for low overhead circular buffers. The X WAGU also supports Bit-Reversed Addressing to facilitate FFT data reorganization.

When executing instructions which require two source operands to be concurrently fetched (i.e. the MAC class of DSP instructions), both the X RAGU and Y AGU are used simultaneously.

The dsPIC33AK128MC106 device family contains an X AGU and a Y AGU for generating data memory addresses. Both X and Y AGUs can generate any EA within the available data memory range. However, EAs that are outside of the physical memory provided return all zeros for data reads and writes to those locations and therefore have no effect. Furthermore, an address error trap will be generated. For more information on address error traps, refer to 10. Interrupt Controller.

3.3.16.1 Address Generation Units and DSP Class Instructions

The Y AGU and Y memory data path are used in concert with the X RAGU by the DSP class of instructions to provide two concurrent data read paths. For example, the MAC instruction can simultaneously fetch two operands to be used in the next multiplication.

DSP class of instructions may use any W-reg (except W15) for either X or Y address space accesses, unlike previous dsPIC devices. Any data write performed by a DSP class instruction takes place in the combined X and Y data space and the write occurs across the X-bus. Consequently, the write can be to any address regardless of where the EA is directed.

The Y AGU only supports Post-Modification Addressing modes associated with the DSP class of instructions. The Y AGU also supports Modulo Addressing for automated circular buffers. All other (MCU) class instructions can access the Y data address space through the X AGU when it is regarded as part of the composite linear space.

3.3.16.2 Data Alignment

The ISA supports long word (32-bit), word (16-bit) and byte (8-bit) sized operations. Data is aligned in data memory and registers as long words, but all data space EAs resolve to bytes. Data word and byte reads will read the complete 32-bit word that contains the word or byte, using the LSbs of any

EA to determine which word or byte to select within the CPU. The selected word or byte is placed onto the lsw or byte of the X data path (no byte accesses are possible from the Y data path as the MAC-class of instruction can only fetch words or long words). That is, data memory and registers are organized as four parallel byte-wide entities with a shared (long word) address decode but separate write lines. Data byte writes will only write to the corresponding side of the array or register which matches the byte address.

Note: Byte reads will always read the entire word, so mechanisms to clear or set peripheral status bits when read (e.g. quick flag clearing mechanisms) are not allowed.

As a consequence of this byte addressability, all EA calculations must be scaled to step through long word aligned memory. For example, the core must recognize that post modified register indirect addressing mode, [Ws]+=1, will result in a value of Ws+1 for byte operations, Ws+2 for word operations, and Ws+4 for long word operations.

Misaligned word or long word accesses are not supported. For word accesses, the LSb of the EA must be 1'b0. For long word accesses, the least significant 2 bits of the EA must be 2'b00. Therefore, care must be taken when mixing operations of different data widths or translating from 16-bit dsPIC code. Should a misaligned read or write be attempted, an address error trap will be forced. If the fault occurs during a read access, the read will be allowed to complete. If the fault occurs during a write access, the write will also be allowed to complete (inhibiting the write would have been possible but inconsistent with other situations where an errant write could not be inhibited). In both cases, the address error trap will be asserted. The next instruction (already pre-fetched and underway) will be executed while the exception is arbitrated and acknowledged. When this instruction completes, the trap will then be taken, allowing the system and/or user to examine the machine state subsequent to execution of the address fault.

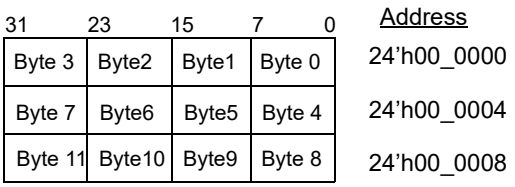
Note: Byte and word ALU operations can produce results in excess of a byte or a word. However, to maintain 16-bit dsPIC backwards code compatibility, the ALU result destination write from all operations maintains the same width as that of the source operands (i.e. MSbs of the destination are not modified) and the SR is updated based only upon the state of the result data.

A sign extend (SE) instruction is provided to allow users to translate 8-bit to16-bit, and 16-bit to 32-bit signed values. Alternatively, for unsigned data, users can clear the MS portion of any W register through executing a byte or word zero extend (ZE).

Note: Care must be taken when mixing byte and word size instructions/operands.

Although most instructions are capable of operating on long word, word or byte data sizes, it should be noted that the DSP and some other instructions operate on long word or word sized data only.

Figure 3-13. Data Alignment



3.3.17 MAC Instructions

The dual source operand DSP instructions (ED, EDAC, MAC, MPY, MPYN, SQR, SQRAC, MSC, SQRSC and SQRN), also referred to as MAC instructions, use a simplified set of addressing modes to allow the user application to effectively manipulate the Data Pointers through register indirect tables.

These instructions support various addressing modes for X and Y data bus, where W-registers accessing these data buses may be any W-reg (except W15) for either X or Y address space accesses. Pre or post modification values are scaled based upon instruction operand width. The MAC-class instruction also supports the ability to write the contents of the accumulator that is not being used

as the instruction result destination to a memory or W-register as defined by the instruction with a restricted set of addressing modes. This is referred to as the Accumulator Write Back (AWB).

Note:
AWB is only intended for use when the DSP engine is operating in fractional data mode. It can only write the MS portion of the target accumulator fractional value.

MAC-class instructions are no longer tied to operand reads of X and Y address space. Operands may both be sourced from X-space, resulting in reading the operand data sequentially rather than concurrently. This will add an additional RAM data fetch delay (typically one cycle) to all such instructions.

3.3.18 Modulo Addressing

Modulo Addressing mode is a method of providing an automated means to support circular data buffers using hardware. The objective is to remove the need for software to perform data address boundary checks when executing tightly looped code, as is typical in many DSP algorithms.

Modulo Addressing can operate in either Data or Program Space (since the Data Pointer mechanism is essentially the same for both). One circular buffer can be supported in each of the X (which also provides the pointers into Program Space) and Y Data Spaces. Modulo Addressing can operate on any W Register Pointer. However, it is not advisable to use W14 or W15 for Modulo Addressing since these two registers are used as the Stack Frame Pointer and Stack Pointer, respectively.

In general, any particular circular buffer can be configured to operate in only one direction, as there are certain restrictions on the buffer start address (for incrementing buffers) or end address (for decrementing buffers) based upon the direction of the buffer.

The only exception to the usage restrictions is for buffers that have a power-of-two length. As these buffers satisfy the start and end address criteria, they can operate in a Bidirectional mode (that is, address boundary checks are performed on both the lower and upper address boundaries).

3.3.18.1 Start and End Address

The Modulo Addressing scheme requires that a starting and ending address be specified and loaded into the 24-bit Modulo Buffer Address registers: XMODSRT, XMODEND, YMODSRT and YMODEND.

Note: Y space Modulo Addressing EA calculations assume word-sized data (LSb of every EA is always clear).

The length of a circular buffer is not directly specified. It is determined by the difference between the corresponding start and end addresses. The maximum possible length of the circular buffer is 32K words (64 Kbytes).

3.3.18.2 W Address Register Selection

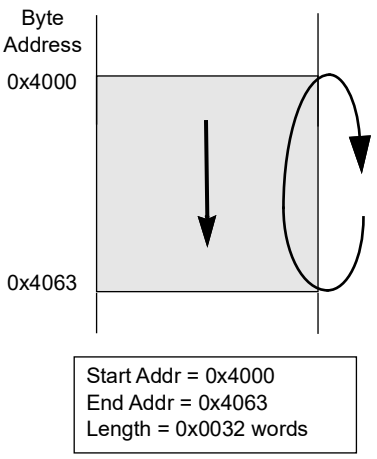
The Modulo and Bit-Reversed Addressing Control register, MODCON[15:0], contains enable flags, as well as a W register field to specify the W Address registers. The XWM and YWM fields select the registers that operate with Modulo Addressing:

- If XWM = `1111`, X RAGU and X WAGU Modulo Addressing is disabled
- If YWM = `1111`, Y AGU Modulo Addressing is disabled

The X Address Space Pointer W (XWM) register, to which Modulo Addressing is to be applied, is stored in MODCON[3:0]. Modulo Addressing is enabled for X Data Space when XWM is set to any value other than '`1111`' and the XMODEN bit is set (MODCON[15]).

The Y Address Space Pointer W (YWM) register, to which Modulo Addressing is to be applied, is stored in MODCON[7:4]. Modulo Addressing is enabled for Y Data Space when YWM is set to any value other than '`1111`' and the YMODEN bit is set (MODCON[14]).

Figure 3-14. Modulo Addressing Operation Example




```
; Set XMODEN bit and XWM = w7
; w7  selected for X-AGU modulo addressing
   MOV.l 0x8007, w6
   MOV.l w6, MODCON

; set modulo start address
   MOV.l #0x4000, w6
   MOV.l w6, XMODSRT
; set modulo end address
   MOV.l #0x4063, w7
   MOV.l w7, XMODSRT

   MPY.l [w7]+=4, w5, A; w7 = 0x4004
   REPEAT #10
   MAC.l [w7]+=4, w5, A
;Content of W7 rolled back to 0x4000
```

3.3.18.3 Bit-Reversed Addressing

Bit-Reversed Addressing mode is intended to simplify data reordering for radix-2 FFT algorithms. It is supported by the X AGU for data writes only.

The modifier, which can be a constant value or register contents, is regarded as having its bit order reversed. The address source and destination are kept in normal order. Thus, the only operand requiring reversal is the modifier.

3.3.18.3.1 Bit-Reversed Addressing Implementation

Bit-Reversed Addressing can only be enabled through the use of the movr.(w/l) instruction. This type of addressing is effective when used with pre-modified or post-modified destination addressing. The destination Bit-Reversed Addressing modifier is sourced from XBREV.XB[14:0].

If the length of a bit-reversed buffer is M = 2^N bytes, the last 'N' bits of the data buffer start address must be zeros.

The XB[14:0] bits are the Bit-Reversed Addressing modifier, or 'pivot point', which is typically a constant. In the case of an FFT computation, its value is equal to half of the FFT data buffer size.

Note: All bit-reversed EA calculations assume either word-size (where the least significant bit of every Effective Address is always clear) or long word-size (where the two least significant bits of the Effective Address are always clear), based on the operation data width selected. The XB value is scaled accordingly to generate compatible (byte) addresses.

Bit-Reversed Addressing is only possible when using the MOVR instruction, and it can target a 16-bit or 32-bit sized data. MOVR instruction supports Register Indirect with Pre-Increment or Post-Increment Addressing and 16/32 bit-sized data writes. When Bit-Reversed Addressing is active, the W Address Pointer is always added to the address modifier (XB) and the offset associated with the Register Indirect Addressing mode is ignored. In addition, the LSb of each 16-bit address and the LS 2-bits of each 32-bit address, will always be zero for both source and destination EAs. The MOVR instruction also supports "in-place" data re-ordering (where only one data buffer is used for both source and destination), source and destination indirect addressing may use the same register

Note: Modulo Addressing and Bit-Reversed Addressing can be enabled simultaneously using the same W register, but the Bit-Reversed Addressing operation will always take precedence for data writes when enabled.

If Bit-Reversed Addressing has already been enabled by setting the BREN (XBREV[15]) bit, a write to the XBREV register should not be immediately followed by an indirect read operation using the W register that has been designated as the Bit-Reversed Pointer.

Figure 3-15. Bit-Reversed Addressing Example

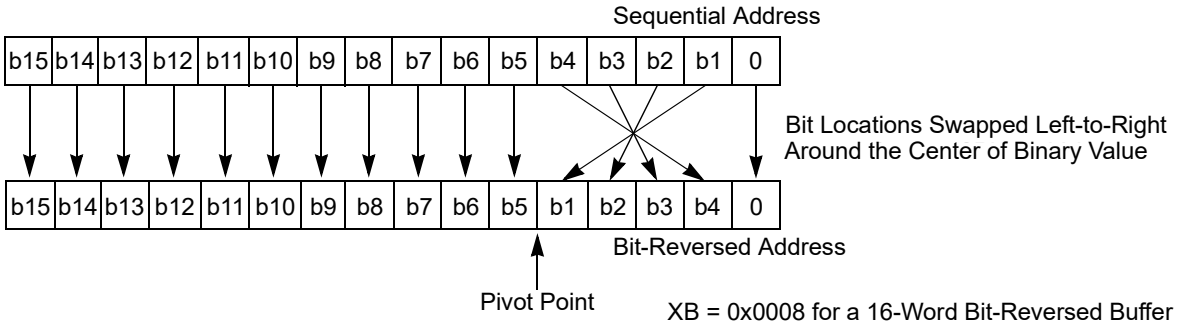


Table 3-8. Bit-Reversed Addressing Sequence (16-Entry)

Normal Address					Bit-Reversed Address				
A3	**A2**	**A1**	**A0**	**Decimal**	**A3**	**A2**	**A1**	**A0**	**Decimal**
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	8
0	0	1	0	2	0	1	0	0	4
0	0	1	1	3	1	1	0	0	12
0	1	0	0	4	0	0	1	0	2
0	1	0	1	5	1	0	1	0	10
0	1	1	0	6	0	1	1	0	6
0	1	1	1	7	1	1	1	0	14
1	0	0	0	8	0	0	0	1	1
1	0	0	1	9	1	0	0	1	9
1	0	1	0	10	0	1	0	1	5
1	0	1	1	11	1	1	0	1	13
1	1	0	0	12	0	0	1	1	3
1	1	0	1	13	1	0	1	1	11
1	1	1	0	14	0	1	1	1	7
1	1	1	1	15	1	1	1	1	15

Example 3-4. 32-Bit Data, Two Buffer Bit-Reversed Data Reordering Example

```
; Two buffer (input and output) bit reversed data re-order subroutine for 32-
bit (real)
; data values
;
; W0: Temp
; W1: Data table size N (long words)
; W8: Input data table pointer (natural order) initialized to start of table
; W9: Output data table pointer (bit reversed) initialized to start of table

       push.l w0
       mov.sl #_XBREV, w0
       lsr.l w1, #1, [w0] ; XBREV = N/2

       sub.l #1, w1
       repeat w1
       movr.l [w8++], [w9++] ; Move data from input to output buffer, then
                             ; bump natural order and bit reversed pointers
       pop.l w0
       return
```

3.3.19 Address Register Dependencies

The dsPIC33A architecture supports a data space read (source) and a data space write (destination) for most MCU class instructions. The EA calculation by the AGU, and subsequent data space read or write, each take one instruction cycle to complete. This timing causes the data space read and write operations for each instruction to overlap.

3.3.20 Multiplier

Using the high-speed, 33-bit x 33-bit multiplier, the ALU supports an unsigned, signed or mixed-sign operation in several MCU Multiplication modes:

- 32-bit x 32-bit signed
- 32-bit x 32-bit unsigned
- 32-bit signed x 5-bit (literal) unsigned
- 32-bit signed x 32-bit unsigned
- 32-bit unsigned x 5-bit (literal) unsigned
- 32-bit unsigned x 32-bit signed
- 16-bit unsigned x 16-bit unsigned

3.4 Prefetch Buffer Unit (PBU)

The Prefetch Buffer Unit (PBU) in the dsPIC33A core devices accelerates the interface between the dsPIC33A program Flash memory and the CPU instruction bus. The PBU can predictively prefetch the next sequential address and cache fetched program data that are the target of a CPU instruction fetch.

PBU in dsPIC33A core devices supports the following functions:

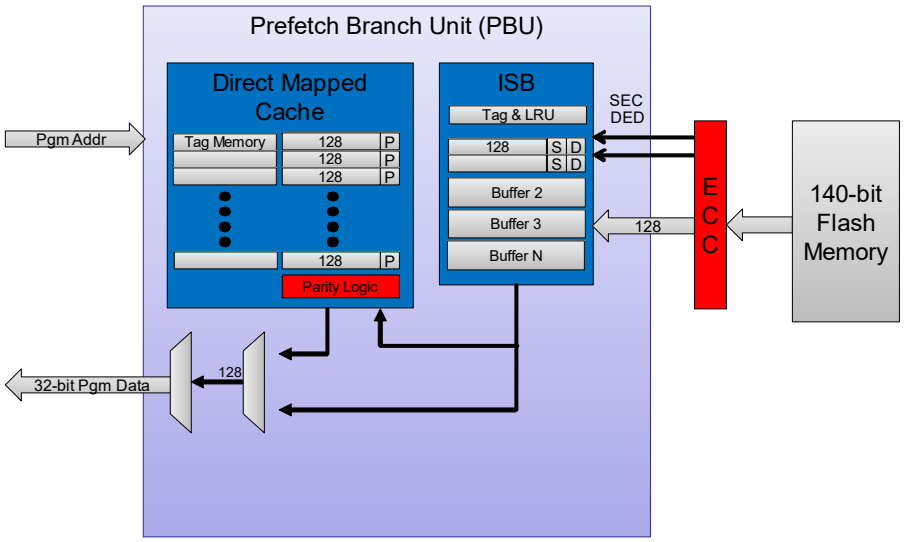
1. PBU accelerates the execution of linear program code flow.
2. As cache accelerates the execution of non-linear program flow changes (branches).

The PBU in the dsPIC33A core devices have the following features:

- Provides interface between Program Flash Memory (PFM) and CPU instruction bus
- Instruction Stream Buffers for prefetching and caching of linear PFM instruction flows
- Instruction Cache for caching of most frequently hit target instructions
- Provides parity checks on program data stored in the Instruction Cache to ensure data integrity

The PBU block diagram in Figure 3-16 shows data paths to and from the PBU in the dsPIC33A environment. The PBU provides data when the CPU fetches program data from Flash memory. It may provide program data from an internal buffer, or it may fetch program data from Flash if the requested program data is not available. Flash fetch operations are therefore accelerated when data are sourced from internal PBU buffers.

Figure 3-16. PBU Block Diagram



3.4.1 Architectural Overview

The PBU is a direct mapped 128-line cache that helps in providing faster program data fetches to the CPU from Flash memory. The PBU provides program data from an internal instruction buffer (ISB), but if it is not available in the internal buffer, the PBU may fetch program data from Flash. Flash fetch operations are therefore accelerated when data are sourced from internal PBU buffers.

The PBU provides an interface between the Program Flash Memory (PFM) and the CPU instruction bus and have the following components associated for operation:

- Instruction Stream Buffer (ISB) - Also termed as the Prefetch Unit (PFU), is available for prefetching and caching of linear PFM instruction flows. ISB is the component that buffers program data words from the program memory. The ISB consists of one or more buffers of a fixed depth. Each buffer holds one or more lines of data fetches from Flash memory. The data held in each buffer represents a linear code flow. These are termed as internal PBU buffers.
- Instruction Cache (IC) - Also known as Branch Target Instruction Cache (BTIC), is used for the caching of target instructions that are most frequently hit. The Instruction Cache refers to the cache memory and associated control logic that form the cache. A cache consists of N lines, directly mapped or through M-way associative. The PBU supports a direct mapped 128-line cache. The required width for the cache is 129-bits. The PBU Cache has two operating modes: IC mode and BTIC mode.
- Integrity Checking Logic - Provides parity checks on program data stored in the Instruction Cache to ensure data integrity. This logic provides parity checking and fault injection on the contents of RAM associated with the Instruction Cache.

The PBU assumes Flash data width and Flash access speed are sufficient to allow linear program execution at the desired speed using only the ISB. The ISB serves as the prefetch buffer and allows the next line of Flash to be fetched as instructions from the current line are executed.

The Instruction Cache becomes useful when there are frequent program flow changes in the source code. A program flow change will result in extra clock cycles because the current Flash fetch must be allowed to complete and then a new fetch must be initiated at the new location. If the desired program data is available in the Instruction Cache, the data may be sourced immediately without waiting for the ISB to complete a new fetch from Flash. However, PBU uses a larger, direct-mapped Instruction Cache and has little control and status interface available to the user as its operations are transparent.

The PBU does not provide data or caching for initiators other than the CPU instruction bus. Data access by the CPU data bus and other bus initiators is accomplished via a dedicated read buffer in the NVM wrapper.

The ISB has multiple buffers also called slices. The ISB Slices help increase performance with CALL/RETURN and other flow changes in the code that return back to the previous code stream.

The ISB is two levels deep in the dsPIC33A PBU. For the first generation of dsPIC33A devices, Flash access time is fast enough to support linear code execution with the given program data word width. Therefore, only one level of prefetch buffer is required. The CPU can execute from the first level, while the next fetch occurs into the second level.

In the case where the code to be executed has a linear flow, no further caching of data would be necessary. However, program flow changes insert latency into the code flow. A prior Flash fetch must be completed and discarded. Then, a new Flash fetch must be started in the new flow. This process can add a variable amount of clock cycles to the execution time, depending on when the flow change occurred relative to the prefetch that was in progress.

When Flash access time is fast enough to support continuous linear program flow, full instruction caching is not required. The cache could be configured as a BTIC, for which only the targets of program flow changes are cached. This mode of cache increases the effective cache size because all program data words do not have to be cached. However, the BTIC operating mode places more

burden on the internal data buses of the PBU. Program data must be transferred from the cache memory to the ISB when a flow change occurs so that a prefetch of the following data words can take place.

3.4.2 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1E60	CHECON	31:24								
		23:16								ISBBUF
		15:8	ON				CHEINV	CHECOH		
		7:0								FLTINJ
0x1E64	CHESTAT	31:24								
		23:16								
		15:8								
		7:0						TPE	RD	PAR
0x1E68	CHEFLTINJ	31:24								
		23:16								
		15:8								
		7:0	FLTPTR[7:0]							

3.4.2.1 Cache Control Register

Name: CHECON
Offset: 0x1E60

Notes:

1. After being set, this bit will be cleared by hardware after the cache and ISB invalidations are completed. Any automatic invalidation will also result in this bit being cleared.
2. This setting is useful when programming non-program data into Flash (emulated EEPROM).

Legend: R = Readable bit; S = Settable bit; Hardware Clearable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
								ISBBUF
Access								R/W
Reset								1

Bit	15	14	13	12	11	10	9	8
	ON				CHEINV	CHECOH		
Access	R/W				R/S/HC	R/W		
Reset	1				1	1		

Bit	7	6	5	4	3	2	1	0
								FLTINJ
Access								R/S/HC
Reset								1

Bit 16 – ISBBUF ISB Buffer Selection bit

Value	Description
1	When On = 0, ISB buffer 1 will be used for prefetch
0	When On = 0, ISB buffer 0 will be used for prefetch

Bit 15 – ON Cache ON bit

Value	Description
1	Cache and all ISB slices are enabled
0	All cache lines and ISB buffers except for the first buffer slice are invalidated. ISB operates with one buffer slice, two deep buffer (basic prefetch mode)

Bit 11 – CHEINV Manual Invalidate Control bit(1)

Value	Description
1	Force invalidation of all cache and ISB lines
0	Invalidation of Instruction Cache and ISBs occurs according to CHECOH bit

Bit 10 – CHECOH Cache Coherency Control bit(2)

Value	Description
1	Invalidate cache upon a Flash programming event
0	Do not invalidate cache on a Flash programming event

Bit 0 – FLTINJ Fault Inject Control bit

Value	Description
1	Parity fault injection enabled for one-time event; cache line will be invalidated and flushed when access occurs and upbs_event[1] will be asserted to indicate an integrity error to the system.
0	Parity fault injection disabled

3.4.2.2 Cache Status Register

Name: CHESTAT
Offset: 0x1E64

Legend: R = Readable bit; S = Settable bit; Hardware Clearable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						TPE	RD	PAR
Access						R/C/HAS	R/W	R/S/HC
Reset						0	1	1

Bit 2 – TPE Read Error Status bit

Value	Description
1	A read error event has occurred; the parity error is latched and a MISS is generated
0	No TAG memory read error event has occurred

Bit 1 – RD Read Error Status bit

Value	Description
1	A read error event has occurred; the CPU has fetched a word from the ISB with a security error
0	No read error event has occurred

Bit 0 – PAR Cache Parity Error Status bit

Value	Description
1	A parity error event has occurred; the CPU has fetched a word from the cache with a parity error
0	No parity error event has occurred

3.4.2.3 Cache Fault Injection Register

Name: CHEFLTINJ
Offset: 0x1E68

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	FLTPTR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 7:0 – FLTPTR[7:0] Fault Injection Pointer bits

Value	Description
255-129	No effect
128	Cache Data Line Parity bit
127	Bit 127 of cache data line
...	
1	Bit 1 of cache data line
0	Bit 0 of cache data line

3.4.3 Operation

The PBU registers only have control to enable or disable certain PBU functions. Parameters such as ISB depth, ISB number of buffers, cache associativity, etc., are all fixed.

The CHECON.ON bit is reset to '1' by default. This provides the best CPU performance for both linear code and program flow changes. The ON bit can be cleared in software to disable most caching functions and make the PBU behave as a basic 2-deep prefetch buffer. This results in lower code performance due to longer program flow changes but still gives deterministic execution behavior, and thus the program flow changes to longer execution time but takes a constant number of cycles.

3.4.3.1 Cache/ISB Manual Invalidation

Manual invalidation of the Instruction Cache and ISBs is used to force cache coherency when the user knows that the cache and Flash contents may not match. It occurs under the following conditions:

- CHECON.CHEINV Control bit: When set by software, this bit will invalidate both the Instruction Cache and all ISBs. This bit will clear automatically by hardware after the cache and ISB memory have been invalidated.
  Note: CHECON.CHEINV is also cleared should an automatic invalidation occur after the bit has been set.

- CHECON.ON control bit: The Instruction Cache memory and ISB buffers are invalidated when the CHECON.ON bit is cleared. This will be the case out of Reset. Execution continues using only a single default ISB slice. Setting CHECON.ON has no effect with respect to cache/ISB invalidation as it is already invalidated and the active ISB will contain valid data from the current instruction flow.

3.4.3.2 Cache/ISB Automatic Invalidation

Automatic invalidation of the Instruction Cache and ISBs is used to ensure cache coherency when the device knows that the cache and Flash contents may not match. It occurs under the following conditions:

- Flash write operation: Automatic invalidation only occurs if the Cache Coherency Control bit (CHECON.CHECOH) is set (default) and Flash is programmed/erased. This is only applicable for writing to the active panel in dual-panel devices.
- Parity error: Refer to 3.4.3.3. PBU Data Error Handling for further details. Only the accessed cache line of the ISB buffer is affected, and the remainder of the cache memory does not need to be invalidated.

3.4.3.2.1 Cache Invalidation when Writing to Flash

Whenever the Flash is written, the user has the option to automatically invalidate the instruction Cache and ISBs using the CHECON.CHEOH control bit. If instruction data are being written to Flash, invalidation ensures Flash memory contents remain synchronized with the Cache and ISB contents.

Note: To fully ensure correct operation, it is recommended that the final instruction that initiates Flash programming be followed by 4 NOP instructions to flush the instruction pipeline. This ensures coherency since the remaining instructions in the CPU pipeline will take no action.

3.4.3.3 PBU Data Error Handling

The PBU handles error correction in two ways. First, any data errors that originate from the program memory are tracked. Secondly, internal PBU data errors that may occur while program data are stored in the PBU Cache RAM are monitored.

3.4.3.3.1 Program Memory Data Errors

Error status is captured from the program memory and buffered along with the fetched program data word in the ISB. Consequently, each line in the ISB has 129 bits: 128 bits of data and 1 bit for error status. The error status bit indicates the data read from the program memory are unusable and incorrect. The program memory data can be bad for multiple reasons, including an uncorrectable ECC error and a security violation that would suppress the data. In any case, the data are not a valid CPU instruction and should not be executed by the CPU.

The PBU does not generate any kind of event, trap or interrupt when bad data are fetched from the program memory. This is because the ISB may speculatively fetch data that would never be executed by the CPU. Secondly, the CPU may speculatively fetch instructions from the PBU during conditional branches that may never get executed.

A bus error signal is passed with the program data to the CPU for instructions fetched from the PBU. If the program data are invalid with the bus error signal asserted, then the CPU can suspend execution in the pipeline and cause a trap event to occur.

3.4.3.3.2 Cached Data Error

The second method of PBU error handling occurs when the cache has detected a parity error on a cached line of program word data. When valid program data is cached for later consumption, then the error status bit is stripped, and the program data word is stored in the cache memory. A single even parity bit is calculated and stored along with the data. This parity bit is used to protect the system from data corruption that could occur in the cache RAM.

A maskable interrupt event is generated by the PBU when a parity error is detected on a cache line. In this case, the cache will invalidate the line with the parity error and the program data must be re-fetched from the program memory. Other than the interrupt event, the only other effect that can

be observed during a cache parity error is additional execution latency caused by Flash program fetch. No address associated with the parity error is captured.

3.4.3.3.3 Corrected Program Memory Data Errors

The program memory error correction logic may correct a data error in the program data supplied to the PBU. These corrections are not reported to the PBU which is usually done for the uncorrectable errors. Specifically, a single-bit corrected error (SEC event) is not reported to the PBU.

The program memory is responsible for tracking and reporting the corrected event. These actions serve as a warning to the system software that integrity of the NVM data may be failing.

3.4.3.4 Cache Fault Injection

A single bit error can be injected on any of the data bits of the cache line or the associated parity bit. The error injection is performed by XORing the data read from the cache line with a '1'. Since the PBU can cache program data from a variety of address locations depending on the program flow, it is impractical to perform error injection for a particular program memory fetch address.

The PBU error injection, when enabled with the FLTINJ bit, will cause a one-time error injection the next time the cache memory is accessed by the CPU. The CHESTAT.PAR bit will indicate when the error injection has been performed. At this time, the PBU will also signal that an integrity error has occurred by creating an interrupt event. The user will not be able to determine which line of the cache buffer caused the event and fetch address.

A write to the FLTPTR register while FLTINJ = 1 will have the effect of re-arming the fault injection. This will help facilitate a software test routine that cycles through an error injection on each bit.

3.4.3.5 Non-Cached Events

Certain fetches from the NVM are not cached. These include:

- Interrupt Vector fetches
- Fetches of debug executive code
- Fetches of invalid program memory data

All these types of fetches are not cached to avoid cache thrashing. Thrashing occurs when other useful data are evicted from the cache and replaced with less useful or invalid data. Inhibiting caching during the above fetches is expected to improve the overall efficacy of the cache, resulting in more cache hits at run-time.

Interrupt Vector fetches are a special type of non-cached event. Specifically, only one program word is fetched from the NVM when the CPU indicates a vector fetch and the ISB is bypassed. When an interrupt occurs, the interrupt vector address is fetched from the vector table, then the instruction at the interrupt vector address is fetched. There is no need for an ISB to perform a prefetch and fetch the program word after the one that contains the interrupt vector address. This would be wasteful and produce extra latency in the servicing of the interrupt event.

The PBU monitors whether the CPU is executing user code or debug executive code. Instructions fetched from the debug executive code are not cached. This avoids additional indeterministic behavior when code execution transitions from the debug executive code back to user mission-mode code.

In addition, the BMX supports execution from RAM, and a RAM based Interrupt Vector Table (IVT). Program or vector fetches from RAM are also non-cached events. However, this capability introduces the possibility of both a vector and its associated handler routine being in either NVM or RAM. Whenever the IVT and/or an exception handler (interrupt or trap) is located within RAM, this is treated as a special case by the PBU to maintain efficient operation.

3.4.3.6 PBU Performance Monitoring

Each word of data requested on the CPU instruction bus will be sourced either from the ISB or the Instruction Cache and not the external NVM. This ensures that each fetch of program data can be completed in minimal time, which maximizes application performance.

Program data that are not already present in the ISB or cache must be fetched from the NVM, which takes additional cycles and decreases overall application performance. Once program data in a particular NVM program word has been consumed by the CPU, it is stored in the Instruction Cache for later use. The exceptions to this are program data with uncorrectable errors, security violations, or debugger executive program data. Once stored, the program data will be available for later reuse in the Instruction Cache until those contents are erased and replaced with another program data word.

3.4.3.6.1 Cache Busy Cycles

The program word is cached on the cycle following the fetch from NVM. During this time, the IC will be busy because of the write to cache memory. If the CPU requests program data during this cycle, the PBU will check the contents of the ISB for an address tag match. In most cases, the data may be sourced from the ISB while the IC is busy. This results in an ISB hit and no extra cycle penalty is incurred. When the IC is busy and no ISB hit occurs, then an extra cycle of latency will be inserted while the PBU waits for the IC write cycle to complete. Then, the IC address tags are checked for an IC hit.

3.4.3.6.2 PBU Performance Event Outputs

The PBU has event outputs that can be connected to external performance counters at the device level for characterization of the PBU performance. These events can be counted over a period of application execution and compared with the total number of executed instructions and/or the total number of elapsed clock cycles to get a measurement of the PBU efficacy. The performance event signals available from the PBU include:

- Instruction Cache “hit” event
- Instruction Stream Buffer “hit” event
- PBU “hit” event
- Instruction Cache “busy” event

The IC hit event indicates when a particular instruction was fetched from the cache memory. The ISB hit event indicates when a particular instruction was fetched from the ISB. This generally happens on the second fetch from a program word while that word is written to the cache memory.

The PBU hit event is of most interest for PBU performance analysis. This event signal is the logical OR of the IC hit and the ISB hit events and indicates that the PBU was able to source the requested data without initiating a new NVM fetch.

The IC busy event is used to count the number of extra cycles that were inserted when the ISB could not source the requested data and a Wait state was necessary to determine if the data were available in the IC. An IC busy event is expected to be infrequent and would occur during program flow changes.

3.4.3.6.3 Factors Affecting PBU Efficacy

PBU efficacy is not a constant value. For a given code segment such as function call, the efficacy of the PBU will be very much dependent on these factors:

- The code that was executed prior to a given code segment
- The size of the code
- The specific location of this code in memory
- Flow changes that occur during the execution of a specific code segment

Different performance results are possible when a specific segment of code is executed in one context vs. another context. The prior code executed will determine what code data is present in the cache memory. The prior code may have evicted all program data associated with the segment of interest. However, if the segment of interest is repetitively executed, then there is a strong possibility that program data associated with this segment will remain in the cache memory without eviction.

In general, a small segment of code which is repetitively executed will produce the best PBU performance results. This is because the code size is small enough to fit within the cache memory and the repetitive nature of the code will maximize the reuse of the cache contents with a minimum of evictions. The absolute location of a code segment within memory will impact the PBU performance. This is closely related to how the code is compiled, optimized, and linked during the software development process.

Two different program data words in a segment of code could have the same address tags. If these program data words are executed often, then numerous cache evictions and NVM fetches will result during code execution. A larger cache memory and/or increased cache associativity can both help this issue. A larger cache memory increases the number of available address tags, while increased associativity increases the number of location options where a specific program data word could be stored. The more flow changes that occur in each segment of code, the higher the possibility that PBU performance will be reduced.

3.4.3.6.4 Implications of Variable NVM Wait States

The NVM Wait states are currently fixed at three, supporting a 4-cycle NVM read access time, and the nature of the PBU/NVM data access handshake is not sensitive to NVM access time. However, variable access times could be advantageous:

1. Devices are designed to target maximum frequency, and the NVM read access time is based upon this requirement. But not all applications will require full-speed operation and/or may be willing to trade-off speed for lower power consumption. Consequently, it may be desirable to allow the user to select fewer (or no) NVM read access Wait state when operating at lower frequencies. This will improve the IC/ISB miss latency and decrease the effective CPI (clocks per instruction) metric, improving overall device execution efficacy.
2. Slower Flash panels will consume less power so future devices may support different speed NVM. Zero Wait state linear code execution directly from Flash would, of course, no longer be possible but would rely on the ISB and IC implementations.

3.5 Performance Monitor Unit (PMU)

The performance monitor provides a method to analyze code efficiency, and allows software routines that incur processor stalls to be identified and optimized. In the dsPIC33A family of devices, the architecture does not have a fixed relationship between the CPU clock speed in MHz and the throughput of the CPU in MIPS (Million Instructions per Second). The throughput of the CPU is dependent on extra cycles incurred from the following:

- CPU pipeline data dependency
- Branches or program flow changes
- Cache misses
- Slow memory or SFR accesses
- Arbitration between bus masters
- A bus that is slower than the CPU

The performance monitor counts the events that cause extra cycles to be inserted into the program flow and the number of elapsed clock cycles. Using this information, the cycles-per-instruction (CPI) can be calculated and the reasons for poor code efficiency can be determined. The CPI value is the number of elapsed clock cycles divided by the number of opcodes that were executed. The stall cycle types listed above will increase the CPI.

The performance monitor uses a set of event signals from the CPU to determine stalls. The module features eight independent 64-bit counters to capture the number of events.

3.5.1 Device-Specific Information

Table 3-9. Performance Monitor Summary

Number of counters	Peripheral Bus Speed	Clock Source
8	Standard	Standard Speed Peripheral Clock

Table 3-10. Counter Event Source Selection

SELECT n [4:0]	Event source	Note
18	Fetch stage PBU miss	This event indicates that the requested program data could not be sourced from either the cache memory or the ISB. Therefore, a new fetch from program memory with additional execution cycles was required to obtain the data.
17	Fetch stage PBU hit	This event indicates that the requested program data was sourced from either the cache memory or the ISB. Therefore, no additional execution cycles were required to fetch the instruction.
16	Fetch stage cache busy	Indicates a cycle during which time the cache was busy transferring data from the instruction stream buffer (ISB) to the cache memory.
15	Fetch stage program memory vector fetch	Indicates that the CPU is fetching an interrupt vector and is aligned with a Program Flow Change event. This event can be used to count interrupt events.
14	Fetch stage program memory program flow change	Indicates that a change in program flow has occurred. This could be due to a CALL, RETURN, RETFIE, conditional or unconditional branch, or interrupt event.
13	Fetch stage read stall	Indicates an extra cycle is needed to fetch a program word from memory. This could be caused by a cache miss or an arbitration conflict when fetching program words and data from the same memory.
12	Fetch stage interrupt latency count enable	Indicates the number of cycles due to interrupt latency.
11	Address stage stall	Indicates that CPU pipeline was stalled in the Address stage for any reason, possibly because the instruction is being discarded.
10	Address stage read stall	Indicates that an instruction could not continue because of extra latency reading a RAM or SFR location.
9	Address stage FPU read stall	Indicates that CPU execution is presently stalled because the CPU cannot read from a FPU register. This has occurred because the FPU is currently busy updating the register data.
8	Address stage FPU instruction stall	Indicates that execution in the FPU coprocessor is currently stalled due to a register data dependency.
7	Address stage hazard	Indicates an extra execution cycle caused by a data dependency upon an earlier instruction in the CPU pipeline, which could not be forwarded.
6	Read stage branch mispredict	Indicates an extra execution cycle caused by mispredicted program flow changes.
5	Read stage conditional branch	Indicates the occurrence of a conditional branch instruction. The count of conditional branch instructions can be compared to the number of branch mispredictions in order to determine the effectiveness of the CPU branch prediction logic.
4	Write stage stall	Indicates that an instruction could not continue because of extra latency writing to RAM or SFRs.
3	Write stage FPU stall	Indicates that CPU execution is presently stalled because the CPU cannot write to the FPU registers. This has occurred because the FPU is currently busy working on the existing register data.
2	CPU instruction completed	Indicates that an instruction in the CPU pipeline has completed.
1	CPU cycle elapsed (reference)	This event count provides the total number of CPU clock cycles elapsed.
0	None	

3.5.2 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1E10	HPCCON	31:24								
		23:16								
		15:8	ON		CLR					
		7:0								
0x1E10	HPCSEL0	31:24				SELECT[3][4:0]				
		23:16				SELECT[2][4:0]				
		15:8				SELECT[1][4:0]				
		7:0				SELECT[0][4:0]				
0x1E14	HPCSEL1	31:24				SELECT[7][4:0]				
		23:16				SELECT[6][4:0]				
		15:8				SELECT[5][4:0]				
		7:0				SELECT[4][4:0]				
0x1E18 ... 0x1E1F	Reserved									
0x1E20	HPCCNTL0	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E24	HPCCNTH0	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E28	HPCCNTL1	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E2C	HPCCNTH1	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E30	HPCCNTL2	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E34	HPCCNTH2	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E38	HPCCNTL3	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E3C	HPCCNTH3	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E40	HPCCNTL4	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E44	HPCCNTH4	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E48	HPCCNTL5	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1E4C	HPCCNTH5	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E50	HPCCNTL6	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E54	HPCCNTH6	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E58	HPCCNTL7	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E5C	HPCCNTH7	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							

3.5.2.1 HPCCON Register

Name: HPCCON
Offset: 0x1E10

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON		CLR					
Access	R/W		R					
Reset	0		0					

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 15 – ON On Control bit

Value	Description
1	Module is enabled and counters increment on event signals
0	Module is disabled and counters do not event on event signals. Counter values may be read.

Bit 13 – CLR Clear Control bit

A write of a '1' to this location will cause the event counters to clear. This bit may be set at any time whether the PMU is in the Enabled state or the Disabled state. This bit location always reads as '0'.

3.5.2.2 HPCSEL0 Register

Name: HPCSEL0
Offset: 0x1E10

Bit	31	30	29	28	27	26	25	24
				SELECT[3][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
				SELECT[2][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				SELECT[1][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				SELECT[0][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 28:24 – SELECT[3][4:0] Counter #3 Event Source Selection bits
These control bits determine which event is counted by the associated counter.
See Table 3-10 for assignments.

Value	Description
11111-00001	Selects the event to be monitored
00000	No event selected (1'b0), counter is disabled

Bits 20:16 – SELECT[2][4:0] Counter #2 Event Source Selection bits
These control bits determine which event is counted by the associated counter.
See Table 3-10 for assignments.

Value	Description
11111-00001	Selects the event to be monitored
00000	No event selected (1'b0), counter is disabled

Bits 12:8 – SELECT[1][4:0] Counter #1 Event Source Selection bits
These control bits determine which event is counted by the associated counter.
See Table 3-10 for assignments.

Value	Description
11111-00001	Selects the event to be monitored
00000	No event selected (1'b0), counter is disabled

Bits 4:0 – SELECT[0][4:0] Counter #0 Event Source Selection bits
These control bits determine which event is counted by the associated counter.
See Table 3-10 for assignments.

Value	Description
11111-00001	Selects the event to be monitored
00000	No event selected (1'b0), counter is disabled

3.5.2.3 HPCSEL1 Register

Name: HPCSEL1
Offset: 0x1E14

Bit	31	30	29	28	27	26	25	24
				SELECT[7][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
				SELECT[6][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				SELECT[5][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				SELECT[4][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 28:24 – SELECT[7][4:0] Counter #7 Event Source Selection bits
These control bits determine which event is counted by the associated counter.

Value	Description
11111–00001	Selects the event to be monitored
00000	No event selected (1'b0), counter is disabled

Bits 20:16 – SELECT[6][4:0] Counter #6 Event Source Selection bits
These control bits determine which event is counted by the associated counter.

Value	Description
11111–00001	Selects the event to be monitored
00000	No event selected (1'b0), counter is disabled

Bits 12:8 – SELECT[5][4:0] Counter #5 Event Source Selection bits
These control bits determine which event is counted by the associated counter.

Value	Description
11111–00001	Selects the event to be monitored
00000	No event selected (1'b0), counter is disabled

Bits 4:0 – SELECT[4][4:0] Counter #4 Event Source Selection bits
These control bits determine which event is counted by the associated counter.

Value	Description
11111–00001	Selects the event to be monitored
00000	No event selected (1'b0), counter is disabled

3.5.2.4 HPCCNTLx Register

Name: HPCCNTLx
Offset: 0x1E20, 0x1E28, 0x1E30, 0x1E38, 0x1E40, 0x1E48, 0x1E50, 0x1E58

Bit	31	30	29	28	27	26	25	24
	HPCCNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HPCCNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HPCCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HPCCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HPCCNT[31:0] Event Counter bits

3.5.2.5 HPCCNTHx Register

Name: HPCCNTHx
Offset: 0x1E24, 0x1E2C, 0x1E34, 0x1E3C, 0x1E44, 0x1E4C, 0x1E54, 0x1E5C

Bit	31	30	29	28	27	26	25	24
	HPCCNT[63:56]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	HPCCNT[55:48]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	HPCCNT[47:40]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	HPCCNT[39:32]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HPCCNT[63:32] Event Counter bits

3.5.3 Operation

The performance monitor operates on the basis of comparing the counter values to the number of CPU cycles. To capture the number of CPU cycles as a reference, one of the available counters is used.

For example, counter 0 can be used for the reference count, and the remaining counters can be used to monitor the available events.

3.5.3.1 Event Selection

Each counter has an associated control to select one of the event sources. The SELECTn[4:0] bits in HPSEL0 and HPSEL1 select one of the signals that are listed in Table 3-10. The CPU cycle elapsed event is the reference and is incremented on each CPU cycle. The CPU instruction completed event indicates that the CPU pipeline has completed. Comparing instructions completed to cycles elapsed yields the CIP value. The ideal value is one. The remaining stall, branch or hazard events can be used to determine where stalls occur and what part of the code to optimize.

3.5.3.2 Counters

Each 64-bit counter is split across a pair of 32-bit registers, HPCCNTLx and HPCCNTHx. The registers are read- only and do not have provisions for saturation or roll over events. It is up to user software to halt the module before saturation occurs. The counters can be reset with the CLR bit (HPCCON[13]). The counters are started and stopped using the ON bit (HPCCON[15]). The count values should only be read when ON = ‘0’.

3.5.3.3 Debugging

Provisions have been made to support the performance monitor in Debug mode. By default, the module is halted in Debug mode to avoid counting cycles associated with the debug executive.

3.5.3.4 Operation in power saving modes

The Performance Monitor module does not operate in Sleep or Idle modes.

3.6 Floating-Point Unit (FPU) Coprocessor

The dsPIC33A FPU Coprocessor includes hardware implementations of the most common floating-point operations for both Single Precision (32-bit) and Double Precision (64-bit) data formats. It is intended to significantly accelerate C compiler floating point operations when compared to executing software library equivalents and is designed to be compliant with the IEEE 754-2008/2019 floating point standards. It also includes additional non-IEEE compliant features which may be enabled to handle subnormal values and improve performance.

3.6.1 Features

- Comprehensive IEEE 754-2008/2019 compliant instruction set
  - Supports both Single and Double Precision operations for most instructions
  - Supports all required rounding modes
- Closely coupled to dsPIC33A CPU core
  - Instructions issued from CPU core as part of application instruction stream
  - Independent instruction pipeline and hazard management
- 32 x 32-bit data registers (F-regs)
  - May be used to hold 32-bit Single Precision or 64-bit Double Precision values
  - Base plus 7 partial FPU register contexts
- Optional subnormal handling for improved performance
  - Subnormal result “Flush-To-Zero” (FTZ) mode
  - Subnormal operand “Subnormals-Are-Zero” (SAZ) mode
- Comprehensive exception implementation and reporting structure
  - IEEE 754-2019 compliant exception implementation
  - Additional exceptions supported for Huge Integer results and Subnormal operands
- Debug features supported:
  - Exception address capture register (FEAR)
  - Exception break signaling
  - NaN propagation

3.6.2 Architectural Overview

The FPU macro relies on the associated dsPIC33A CPU for all instruction fetches, most decoding, and for all operand movement to and from the system memory. The FPU contains no local memory other than its own register set. Being coupled to the CPU, data size nomenclature is common to both CPU and FPU wherein a word is 16-bits wide, a long word is 32-bits wide and a double word is 64-bits wide.

FPU instructions are part of the CPU Instruction Set Architecture and are executed as part of the CPU code image. FPU instructions are therefore executed as a part of the normal execution flow. There are no restrictions with regards to when FPU instructions may appear within the instruction flow.

The CPU can issue, and the FPU can accept, no more than one instruction per clock cycle. However, once issued, the CPU and FPU use independent pipelines to execute the instruction. Consequently, there can be multiple instructions in the process of being executed in both pipelines at any one time. The FPU pipeline will stall the CPU when it is unable to accept any more instructions. The FPU pipeline is also sensitive to speculative instruction control from the CPU (i.e., such that not all issued FPU instructions will be committed). This allows FPU instructions to be located within speculative execution slots that follow conditional branches.

After successful issue of an FPU instruction, the CPU continues as if executing a single cycle FNOP instruction, and the FPU instruction execution continues within the FPU. Therefore, as some FPU instructions require several cycles to complete, subsequent CPU (and/or FPU) instructions can be fetched, issued and executed (dependencies aside) while the FPU operation progresses. Only when the CPU encounters a hazard with the FPU will it be stalled until the hazard is resolved.

Data and structural hazards are detected and mitigated in both the CPU and FPU and can result in operational stalls which will extend the execution time and increase the effective Cycles Per Instruction of both CPU and FPU instructions.

Note: Refer to the dsPIC33A Programmer's Reference Manual for the syntax of all FAND, FIOR, FMUL, etc, instructions.

3.6.2.1 Instruction Pipeline Overview

The pipeline stages consist of Read (RD), Execute (X[n]) and Write-Back (WB), differentiated from the equivalent CPU pipeline stages through the use of different nomenclature. The RD-stage is a single cycle operation (unless stalled). The WB-stage is always a single cycle operation. However, the execute stage will consist of as many cycles as deemed necessary for the selected instruction functional block. Most basic functions are single cycle execute operations, though more complex functions (e.g., divide) can be many cycles.

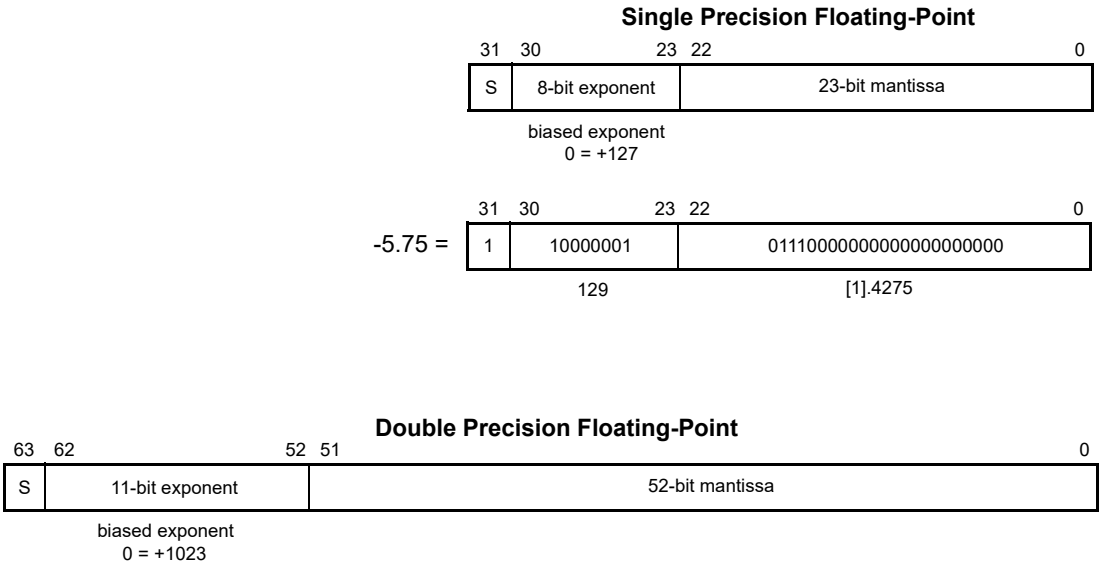
Each instruction that is issued to the FPU must be completed (or killed if speculative) in the order issued. That is, Out of Order (OoO) execution is not supported. However, as the execution time of the FPU instructions can vary considerably, in-order execution requires logic to tag each instruction as it is committed for execution, then track its progress as it flows through the instruction pipeline. Subsequent instructions will therefore be stalled until such time that earlier ones have progressed to allow for sequential, in-order execution.

3.6.2.2 Introduction to Floating Point

The IEEE standard for Floating-Point Arithmetic (IEEE 754-2008) specifies the floating-point data formats which are comprised of a Sign bit, an exponent value and a (fractional) mantissa value. The dsPIC33A Floating-Point Unit (FPU) supports both Single Precision (32-bit, SP) and Double Precision (64-bit, DP) operations for most (though not all) instructions. To avoid the need for another Sign bit in the exponent, the IEEE floating-point format exponent is biased by 127 (SP) or 1023 (DP). Consequently, for any datum, the required IEEE exponent value = datum exponent + bias.

In addition, the '1' to the left of the most significant bit of the mantissa is implied for all numbers except subnormal numbers and is consequently referred to as the leading bit convention "hidden bit." The mantissa is therefore a fractional value with an implied integer value of [1].

Figure 3-17. IEEE Floating-Point Data Formats and Single Precision Example



An IEEE floating-point number can therefore be represented as:

$(-1)^S \times [1].(m_{(base2)}) \times 2^{(e-bias)}$

where:

- '*S* indicates the sign of the number (same values as a signed integer value)
- 'e' represents the exponent value
- 'm' represents the fractional mantissa value
- 'bias' is 127 (SP) or 1023 (DP)

For example, -5.75 = -(1.4275 x 2^2). In IEEE SP format this would be represented as:

(-1)1 x [1].4275 x 2(129-127)

or (as shown in Figure 3-17):

S = 1, exponent = 129_{10}, mantissa = [1].427510 or:

0xC0B8 0000

3.6.2.2.1 IEEE 754-2008 Compliance

This module is compliant with the IEEE 754-2008 Standard for Floating-Point Arithmetic for data formats, supported signaling and quiet branch predicates, exception status flags, and exception status behavior.

Note that the IEEE 754-2008 minNum(x,y) and maxNum(x,y) definitions are supported only through the largely compatible IEEE 754-2019 minimumNum(x,y) and maximumNum(x,y) operations via the FMINNUM and FMAXNUM instructions. The functional differences related to how +0 and -0 are considered:

- IEEE 754-2008 minNum(x,y) / maxNum(x,y): Operand values +0 and -0 are regarded as equivalent. The (implementation dependent) result could therefore be either +0 or -0.
- IEEE 754-2019 minimumNum(x,y) / maximumNum(x,y): Operand values +0 and -0 are not regarded as equivalent such that -0 compares to less than +0. The result will therefore be the correct sign of 0 based on the selected operation.

Features Beyond IEEE 754 Requirements

Exception Address Capture Register

The Floating-Point Exception origination address capture register (FEAR) captures the address of the instruction that generates a floating-point macro exception, provided the associated exception mask bit is clear. If the exception is masked, nothing is captured.

This register is intended for use during system debug, though the FEAR register is read/write in both Mission and Debug modes.

Huge Integer Exception

This exception is signaled whenever a Float-to-Integer conversion operation (FF2DI and FF2LI) results in an integer value that is larger than the destination register can represent. It is not defined within any IEEE 754 specification apart from a reference to setting the Invalid exception should an integer value exceed the destination size unless this “cannot otherwise be indicated.”

3.6.2.2.2 IEEE 754-2019 Compliance

Minimum and Maximum Functions

The FPU module supports all minimum and maximum operations defined in the IEEE 754-2019 standard. The IEEE 754-2008 minNum(x,y) and maxNum(x,y) operations are not directly supported.

- FMINNUM, FMAXNUM: IEEE 754-2019 minimumNumber(x,y)/maximumNumber(x,y) functions. When one of the input operands is a NaN and the other input is a floating-point number (that is not a NaN), the instructions will return the floating-point number. If both input operands are a NaN, the instructions will return a qNaN.
- FMIN, FMAX: IEEE 754-2019 minimum(x,y)/maximum(x,y) functions. When one (or both) of the input operands is a NaN, the instructions will return a qNaN.

Refer to the truth table shown in Table 3-11 for a definition of how NaN operands are handled.

For all minimum and maximum operations, any finite operand value will compare as less than +infinity, or greater than -infinity. Operand value of -0 compares to less than +0.

Table 3-11. FMINNUM/FMAXNUM/FMIN/FMAX Operation

Op	Source Operands		Invalid Exception Mask	Result Fd	FSR.INVAL	Invalid Exception Taken?
	Fb	Fs				
FMINNUM FMAXNUM FMIN FMAX	FPN1	FPN2	Don’t care	FPN1 or FPN2(1,2,3,4)	0	No
	qNaN1	qNaN2	Don’t care	qNaN1 or qNaN2 (5)	0	No
	sNaN	qNaN	1	qNaN (Fs) (6)	1	No
			0			Yes
	qNaN	sNaN	1	qNaN (Fb) (6)	1	No
			0			Yes
	sNaN1	sNaN2	1	Quieted sNaN1 or sNaN2 (5)	1	No
			0			Yes
FMINNUM FMAXNUM	FPN1	qNaN	Don’t care	FPN1	0	No
	qNaN	FPN2	Don’t care	FPN2	0	No
	FPN1	sNaN	1	FPN1	1	No
			0			Yes
	sNaN	FPN2	1	FPN2	1	No
			0			Yes

.....continued						
Op	Source Operands		Invalid Exception Mask	Result Fd	FSR.INVAL	Invalid Exception Taken?
	Fb	Fs				
FMIN FMAX	FPN1	qNaN	Don't care	qNaN (Fs)	0	No
	qNaN	FPN2	Don't care	qNaN (Fb)	0	No
	FPN1	sNaN	1	Quieted sNaN (Fs)	1	No
			0			Yes
	sNaN	FPN2	1	Quieted sNaN (Fb)	1	No
			0			Yes

Notes:

1. FPN1 and FPN2 are floating-point numbers that are not a NaN (i.e., normal, zero, infinity or sub-normal).
2. Result determined by FMINNUM/FMIN or FMAXNUM/FMAX operation.
3. Operand value of -0 compares to less than +0.
4. If Fb = Fs (and of the same sign, including infinities), result (Fd) will be loaded with Fb.
5. NaN with largest significand will be passed to result (Fd), quieted if an sNaN.
6. qNaN values have priority over sNaN values (see Table 3-13).

Clamping (Limit) Functions

Although not specified in any IEEE 754 standard, the ISA supports a clamping (or limit) instruction (FFLIM) intended for use where an input operand needs to be constrained between an upper and lower limit. It serves a similar purpose to the integer equivalent FLIM instruction and is essentially a concurrent execution of FMIN and FMAX operations with a common operand. Refer to the truth table shown in Table 3-12 for a definition of how NaN operands are handled.

Any finite operand value will compare as less than +infinity or greater than -infinity. Operand value of -0 compares to less than +0.

For FFLIM operations, when both upper and lower limits are either both qNaN or both sNaN values, a NaN significand comparison (to select result NaN source) is not required, and the Fb NaN will be the default source for the result. This differs from how coincident NaN values are treated in general.

Furthermore, a NaN input value (Fd) will cause the limit values to be ignored and will become the source for the result. That is, checks between input NaNs and limit values (NaNs or otherwise) are also not required.

Table 3-12. FFLIM Operation

Fb[3] (Lower Limit)	Fs[3] (Upper Limit)	Fd (Input Value)	Invalid Exception Mask	Fd (Result)	FSR.INVAL	Exception Taken?
FPNL	FPNU	FPN	Don't care	FPNL or FPNU or FPN[2]or Distinguished qNaN[3]	0	No
FPNL	qNaN_U	FPN	Don't care	qNaN_U	0	No
FPNL	sNaN_U	FPN	1	Quieted sNaN_U	1	No
			0			Yes
qNaN_L	FPNU	FPN	Don't care	qNaN_L	0	No
sNaN_L	FPNU	FPN	1	Quieted sNaN_L	1	No
			0			Yes
sNaN_L	sNaN_U	FPN	1	Quieted sNaN_L[5]	1	No
			0			Yes
sNaN_L	qNaN_U	FPN	1	qNaN_U[6]	1	No
			0			Yes
qNaN_L	sNaN_U	FPN	1	qNaN_L[6]	1	No
			0			Yes
qNaN_L	qNaN_U	FPN	Don't care	qNaN_L[5]	0	No
Don't care	Don't care	sNaN	1	Quieted sNaN[7]	1	No
			0			Yes
Don't care	Don't care	qNaN	Don't care	qNaN	0	No

Notes:

1. FPNL and FPNU are floating-point numbers that are not a NaN.
2. Result determined by FFLIM operation.
3. If Fs is less than Fb (and neither Fs nor Fb are NaN values), the result will be the distinguished qNaN, and the Invalid exception will be signaled.
4. FFLIM operation based on IEEE 754-2019 minimum(x,y) and maximum(x,y) operation definitions.
5. Unlike FMIN/FMAX operations, no magnitude comparison of limit NaN values is required. Default result will always be sourced from Fb.
6. qNaN values have priority over sNaN values (see Table 3-13).
7. Unlike FMIN/FMAX operations, no comparison of limit and input (Fd) values is required. Default result will always be sourced from Fd.

NaN Propagation

The FPU macro supports NaN (payload) propagation to facilitate code debugging. After the CPU issues an instruction to the FPU, the source operands are examined and a NaN value detected, compared, and then propagated. Two operand instructions propagate NaN values as shown in Table 3-13.

The FMAC instruction is a special case with respect to NaN propagation as it consists of essentially three operands consisting of the two source operands (for the multiply) and a prior FMAC result value (i.e, the intermediate used for the accumulate function). The source operands are examined as usual but in conjunction with the selected intermediate result, and any NaN values detected are propagated as defined by Table 3-13.

FFLIM is also a three-operand instruction, though it is ultimately either a two-operand maximum or minimum operation based on the value of the source operand. NaN values detected are propagated as defined by Table 3-12.

Table 3-13. NaN Propagation Priority

Source Operands		Result	Condition	Notes
Fb	Fs			
FPN	sNaN	Quieted sNaN	—	INVAL signaled
FPN	qNaN	qNaN	—	-
sNaN	FPN	Quieted sNaN	—	INVAL signaled
qNaN	FPN	qNaN	—	-
qNaN1	qNaN2	qNaN1	qNaN1 ≥qNaN2	-
		qNaN2	qNaN2 > qNaN1	-
sNaN	qNaN	qNaN	—	INVAL signaled
qNaN	sNaN	qNaN	—	INVAL signaled
sNaN1	sNaN2	Quieted sNaN1	sNaN1 ≥ sNaN2	INVAL signaled
		Quieted sNaN2	sNaN2 > sNaN1	

NaN Propagation Rules

For instructions that generate a result, special propagation rules apply when one or both source operands are NaN values, such that sNaNs can be successfully used as “tracer” values.

When both source operands are NaNs, qNaNs take priority over sNaNs. The appropriate NaN values will be selected as the operation default result as shown in Table 3-13. In the absence of any NaN source operands, any other floating-point numbers will be processed by the FPU module to generate the result.

Note: Source sNaN values will always generate an Invalid exception, but the corresponding quieted sNaN may not always be the operation result.

This magnitude comparison is based on the magnitude of the significand associated with each of these values (the sign is ignored). It is straightforward to implement because:

- The MSb of a sNaN significant is 0 (with any non-zero value in the remaining bits).
- The MSb of a qNaN significant is 1 (with any value in the remaining bits).

An example tracer sNaN propagation is shown in Figure 3-18. When an FPU operation (Op1) executes with a sNaN and a normal floating-point number, the sNaN will be quieted and propagate as the result. In Figure 3-18, this is sNaN1 (the initial tracer) being propagated as qNaN1. Should a subsequent operation (Op2) execute with qNaN1 and, for example, a later sNaN tracer (sNaN2), operand qNaN1 will have priority, thereby maintaining propagation of the original tracer payload. However, should that qNaN1 value then be presented to another FPU operation (Op3) together with another qNaN, the qNaN result could be either of the source qNaNs, depending upon the magnitude of their respective significands.

However, if the significand of the initial sNaN1 tracer is large enough, it will ultimately be able to continue to propagate past all subsequent NaNs and be available to view at the end of the code block, thereby allowing it to be traced back to its source.

Figure 3-18. Tracker sNaN Operand Propagation Example

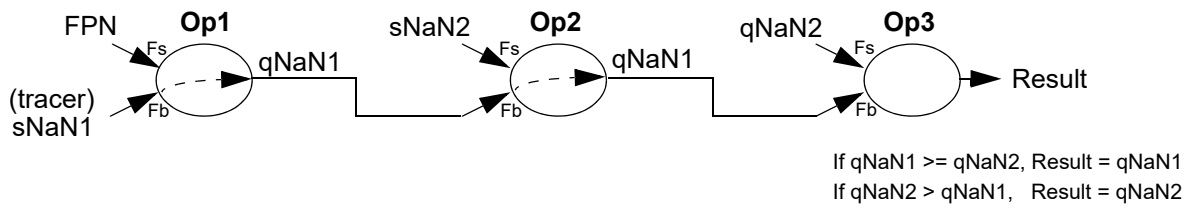


Table 3-14. FMAC NaN Propagation Priority

Multiply Source Operands		Add Source Operands		FMAC Result (Fd)	Notes
Fb or Fs	Fs or Fb	Intermediate Result	Accumulator Source (Fd)		
FP Multiply					
		FP Add			
FPN	FPN	FPN	FPN	FPN	
			qNaN	qNaN	
			sNaN	Quieted sNaN	INVAL signaled
		Distinguished qNaN(3)	FPN	Distinguished qNaN	INVAL signaled
			qNaN1	Distinguished qNaN or qNaN1(2)	INVAL signaled
			sNaN	Distinguished qNaN or Quieted sNaN(2)	INVAL signaled
FPN	sNaN1	Quieted sNaN1	FPN	Quieted sNaN1	INVAL signaled
			qNaN	Quieted sNaN1 or qNaN(2)	INVAL signaled
			sNaN2	Quieted sNaN1(2)	INVAL signaled
FPN	qNaN1	qNaN1	FPN	qNaN1	
			qNaN2	qNaN1 or qNaN22	
			sNaN	qNaN1	INVAL signaled
qNaN1	qNaN2	qNaN1 or qNaN2(2)	FPN	qNaN1 or qNaN2(2)	
			qNaN3	qNaN1 or qNaN2 or qNaN3(2)	
			sNaN	qNaN1 or qNaN2(2)	INVAL signaled
sNaN1	sNaN2	Quieted (sNaN1 or sNaN2)(2)	FPN	Quieted (sNaN1 or sNaN2)(2)	INVAL signaled
			qNaN	Quieted (sNaN1 or sNaN2)(2) or qNaN	INVAL signaled
			sNaN3	Quieted (sNaN1 or sNaN2)(2)	INVAL signaled

Notes:

1. FPN is a floating-point number that is not a NaN.
2. Using significand magnitude comparisons as defined in Table 3-13.
3. Distinguished qNaN intermediate result will arise when operands are 0 and Inf (any sign).

3.6.3 Zero, Infinity, Not a Number (NaN) and Subnormal Values

The IEEE 754-2008/2019 standards reserve data encoding to represent special values, as shown in Figure 3-19.

Zero is conveyed when both exponent and mantissa are all 0's. Zero is a signed value (for some operations) as determined by the Sign bit. Infinity is conveyed by an exponent value of all 1's with an all 0's mantissa. Infinity is a signed value as determined by the Sign bit.

A Signaling NaN is conveyed by an exponent value of all 1's with the MSb of the mantissa set to 0 (remaining mantissa bits may be set to any value). The Quiet Nan (qNaN, see 3.6.3.1. Not a Number (NaN)) is conveyed by an exponent value of all 1's with the MSb of the mantissa set to 1 (remaining mantissa bits may be set to any value). NaN values are not signed, so the Sign bit may be any state.

A Subnormal value (see 3.6.3.2. Subnormal Number) is conveyed by an exponent of all 0's and any non-zero mantissa value. Subnormals are signed values as determined by the Sign bit.

3.6.3.1 Not a Number (NaN)

The Signaling NaN (sNaN) and Quiet NaN (qNaN) are specific data codes that indicate certain situations. In all cases, an exponent value of all 1's with a non-zero mantissa signifies a NaN (an exponent value of all 1's with an all 0's mantissa is used to convey Infinity).

qNaNs may be generated as the result of an invalid operation, such as taking the square root of a negative floating-point number. A qNaN will propagate through subsequent floating-point operations. Operations that will generate an Invalid exception for each instruction are documented in Table 3-18.

sNaNs are reserved input operands which, under default exception handling, will signal an Invalid exception when encountered. This may be used to indicate uninitialized variables, or as debug aids, but they are never generated by arithmetic computations or comparisons. Whenever the source operand of operation is an sNaN, the result will be a qNaN.

Both sNaNs and qNaNs can store "Payloads" in the mantissa bit field. The payload must not affect the MSB of the mantissa. The payload can be used as a debugging aid in tracing through complex arithmetic calculations.

3.6.3.1.1 qNaN and sNaN Propagation

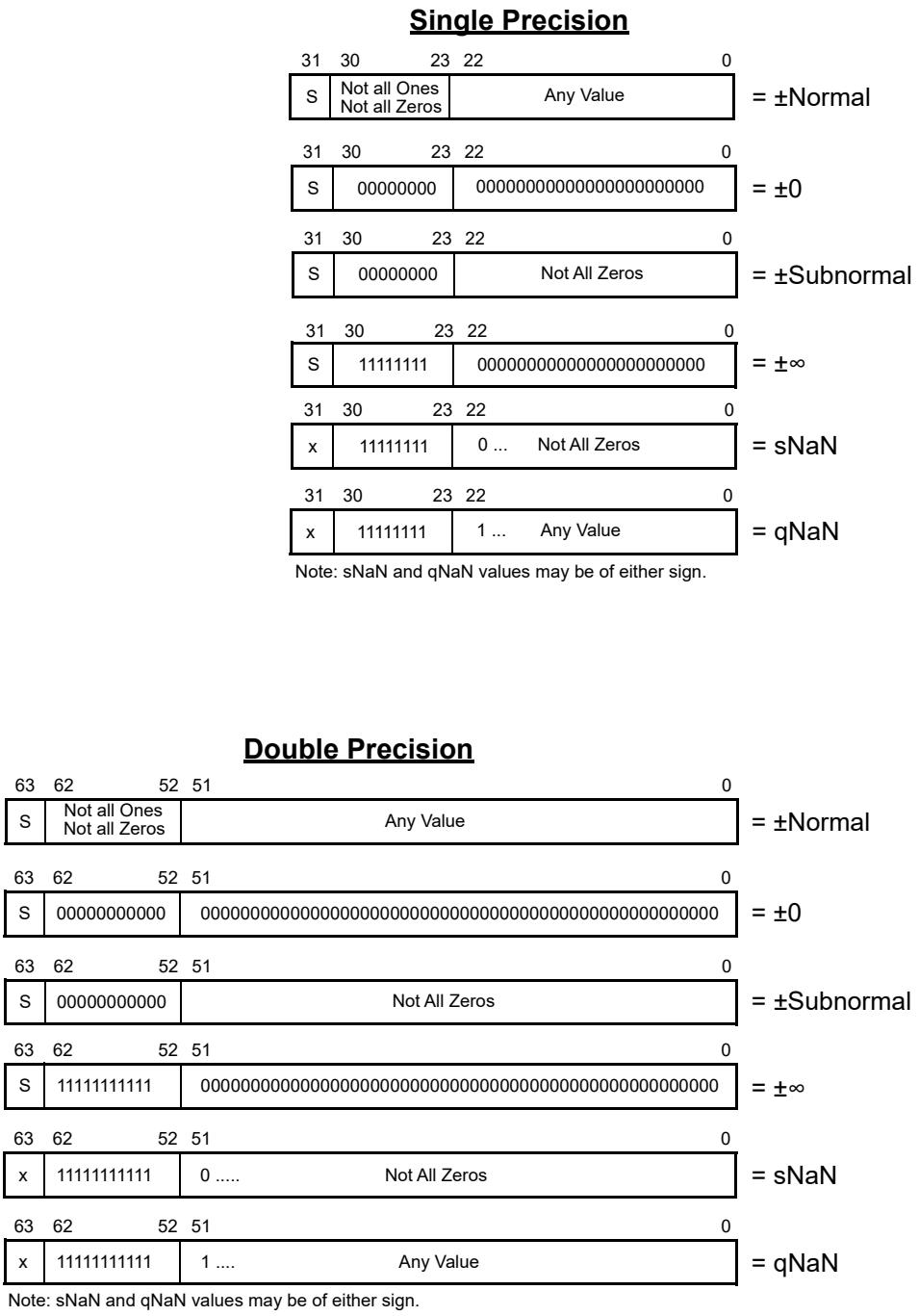
The IEEE 754-2008/2019 standards indicates that source qNaNs should be propagated, including any associated payload. The FPU module does not propagate any source qNaNs, but instead generates fixed distinguished qNaN results.

In keeping with other device floating-point implementations, this module will propagate qNaN and sNaN values where possible. Refer to NaN Propagation for further detail.

For instructions where a source operand qNaN is not available, a distinguished qNaN value as shown below will be provided as the result whenever those instructions suffer a computational error.

- Single Precision: Distinguished qNaN = 0x7FC0_0001
- Double Precision: Distinguished qNaN = 0x7FF8_0000_0000_0001

Figure 3-19. Floating-Point Encodings



3.6.3.1.2 NaN Operands with Float-to-Integer Conversion

The FF2DI and FF2LI are the float-to-integer instructions. These instructions can output Huge Integer in lieu of Invalid when the source is a value that would convert to an integer outside the range of the result format under the applicable rounding attribute. This output is implemented as a new exception, Huge Integer (FSR.HUGI).

The IEEE 754-2008/2019 standard calls for Invalid to be signaled if this situation cannot otherwise be indicated. The FSR.HUGI exception is considered an implementation of “otherwise indicated,” making the FF2DI and FF2LI instructions compliant.

The module drives status output Invalid (and does not drive Huge Integer) when the source is ±NaN, or ±∞ per the IEEE 754-2008/2019 standards. Note that Invalid is also driven for a qNaN input

3.6.3.2 Subnormal Number

A subnormal number (historically also referred to as a denormal number) is a *non-zero* floating-point number with a magnitude of less than that of the smallest normal number representable in the given format. The benefit of subnormal numbers is that they allow for gradual underflow when a result is very small (when compared to that without subnormal numbers). The IEEE 754 standard represents subnormal numbers as a special case.

Using Single Precision data format as an example, the smallest normal numbers around 0 are greater than $+2^{-126}$ or less than -2^{-126}, which occur when the floating-point number exponent is 1 (bearing in mind that the 8-bit exponent is defined with a bias of +127) and the mantissa is all 0's.

The exponent value of 0 is reserved for subnormal numbers. However, the IEEE 754 standard treats subnormal numbers as a special case where the hidden mantissa bit becomes 0 and the exponent bias is changed (by +1) to compensate, such that the datum exponent becomes -126. This allows the subnormal range to surround 0 and be between a little greater than -2^{-126} to a little less than -2^{-126}. That is:

$-2^{-126} < subnormal < +2^{-126}$

The minimum exponent value is referred to as Emin, and is -126 for Single Precision and -1023 for Double Precision formats. A subnormal number would therefore be represented as:

$(-1)^{S} \times [0].m_{base2} \times 2^{Emin}$

where:

-'*S*' indicates the sign of the number (same values as a signed integer value)

-'*m*' represents the fractional mantissa value

For example, the largest SP positive subnormal number will be when all mantissa bits are all set (0x007F_FFFF), and the smallest number will be when all mantissa bits are all clear (0x0000_0000), which is 0.0.

3.6.3.2.1 Subnormal Number Handling

Should any floating-point calculation generate a subnormal result, the FSR.UDF will be set; if it is not already set, the sticky status FSR.UDFS will also be set. In addition, if any instruction is presented with a subnormal operand value, FSR.SUBO will be set. If it is not already set, the sticky status FSR.SUBOS will also be set.

Subnormal Override Functions

Although not IEEE 754 compliant, subnormal operands and/or results may be overridden to improve the performance of some applications that do not require subnormal number precision. Use of the subnormal override function:

- Avoids the consequences of processing or having to deal with subnormal datum.
- Handles result underflows when a result is subnormal, negating the need to handle an underflow exception.

The subnormal override functions consist of two parts, one to flush subnormal input operands to zero (referred to as Subnormals-Are-Zeros, or SAZ mode), and the other to remove subnormal results (referred to as Flush-To-Zero, or FTZ mode).

Note: Subnormal override modes are not applicable to FCPS/FCPQ (no result to override), `FAND`, `FIOR`, `FTST`, `FMOV`, `FMOVC` or any CPU to/from FPU data move instruction.

Subnormals-Are-Zero (SAZ)

Subnormals-Are-Zero (SAZ) mode is enabled when FCR.SAZ is set and will ensure that any subnormal operand input to a Functional Block is replaced with a 0 value of the same sign as

the subnormal value it is replacing. This avoids the consequences of processing or having to deal with subnormal datum. This operation applies to all floating-point instructions except: FMOV, FMOVC, FAND, FIOR, FTST, FLI2F and FDI2F.

Note: SAZ mode is applied to FABS and FNEG instructions to ensure result consistency with that of an equivalent sequence of FPU arithmetic instructions.

Note: Does not apply to FPU to CPU or CPU to FPU move instructions.

3.6.3.2.2 Flush-To-Zero (FTZ)

Flush-To-Zero (FTZ) mode is enabled when both FCR.FTZ and FCR.UDFM are set. If the underflow exception is unmasked (FCR.UDFM = 0), then the FCR.FTZ bit will have no effect. Should a floating-point operation generate an infinitely precise result that is less than the smallest possible subnormal number, then the Functional Block will round this to a result of 0 with the same sign as the subnormal value. This will occur irrespective of whether FTZ mode is enabled or not. Both Underflow (FSR.UDF) and Inexact (FSR.INX) will be signaled (if not already set, sticky status FSR.UDFS and FSR.INXS will also be set). Should a floating-point result be a subnormal number (that the Functional Block has not rounded up to the smallest magnitude normal number), and FTZ mode is enabled, the result will be replaced with 0 of the same sign as the subnormal value it is replacing. Again, both Underflow (FSR.UDF) and Inexact (FSR.INX) will be signaled (if not already set, sticky status FSR.UDFS and FSR.INXS will also be set), though the Underflow exception has to be masked (in order to enabled FTZ mode), so no interrupt will be issued. Forcing the result to 0 allows the user to ignore underflows (though at the expense of some accuracy).

The FCR.FTZ bit is only examined during the WB-stage of an instruction such that it may be modified as late as the cycle before the instruction enters the WB-stage. For example, the following code sequence will only apply the FTZ function to the FSUB instruction:

```
 * Assume FCR.FTZ=0 && FCR.UDFM=1 at entry
FADD.s F0, F1,F2                ;add without FTZ
FIOR #0x0400, FCR               ; set FCR.FTZ
FSUB.s F3, F4,F5                ;sub with FTZ
FAND #0xFBFF, FCR               ;clear FCR.FTZ
FADD.s F2,F6,F7                 ;add without FTZ
```

3.6.3.2.3 Subnormal Operand Exception

Should any affected instruction execute using a subnormal operand, and SAZ mode is disabled, the Subnormal Operand (FSR.SUBO) exception will be signaled. This provides a mechanism to indicate the use of a subnormal value without requiring the operand be tested (FTST).

Should SAZ mode be enabled and a subnormal operand is encountered (and changed to a 0 value), SUBO will not be signaled.

SAZ mode may be enabled irrespective of whether the SUBO exception is masked or not (though when enabled will never signal SUBO).

3.6.4 Floating-Point Data Register (F0-F31)

Name: Fn

Bit	31	30	29	28	27	26	25	24
	Fn[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	Fn[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	Fn[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	Fn[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – Fn[31:0] Floating-Point Data Register bits

3.6.5 Floating-Point Control Register

Name: FCR

Note:

1. Floating-Point Exception Mask bits, FCR [6:0]: Each Exception Mask bit corresponds to an Exception Status flag in the FSR. The Mask bit must be clear to allow the exception event to generate an interrupt to the CPU. The Underflow Mask bit (FCR.UDFM) is also used as part of the Flush-to-Zero (FTZ) mode enable as discussed in 3.6.3.2.2. Flush-To-Zero (FTZ).

   Floating-point rounding mode control, FCR [9:8]: These bits define the global IEEE 754 compatible rounding mode used by the FPU instruction. 3.6.8.9.3. Rounding Modes.

   Floating-point subnormal override mode control, FCR [11:10]: These bits enable the Subnormals-Are-Zero (SAZ) and Flush-to-Zero (FTZ) subnormal override modes supported by the FPU.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					SAZ	FTZ	RND [1:0]	
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
		SUBOM	HUGIM	INXM	UDFM	OVFM	DIV0M	INVALM
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 11 – SAZ Subnormals Are Zero Operand Mode bit

Value	Description
1	Subnormals Are Zero mode is enabled
0	Subnormals Are Zero mode is disabled

Bit 10 – FTZ Flush To Zero Result Mode bit

Value	Description
1	Flush To Zero mode is enabled
0	Flush To Zero mode is disabled

Bits 9:8 – RND [1:0] FPU Rounding Mode bit

Value	Description
11	IEEE Round to Negative Infinity (floor)
10	IEEE Round to Positive Infinity (ceiling)
01	IEEE Round to Zero (truncate)
00	IEEE Round to Nearest (even)

Bit 6 – SUBOM Subnormal Operand Exception Mask bit

Value	Description
1	Subnormal exception is masked
0	Subnormal exception is not masked

Bit 5 – HUGIM Huge Integer Exception Mask bit

Value	Description
1	Huge Integer exception is masked
0	Huge Integer exception is not masked

Bit 4 – INXM Inexact Exception Mask bit

Value	Description
1	Inexact exception is masked
0	Inexact exception is not masked

Bit 3 – UDFM Underflow Exception Mask bit

Value	Description
1	Underflow exception is masked
0	Underflow exception is not masked

Bit 2 – OVFM Overflow Exception Mask bit

Value	Description
1	Overflow exception is masked
0	Overflow exception is not masked

Bit 1 – DIV0M Divide By Zero Exception Mask bit

Value	Description
1	Divide By Zero exception is masked
0	Divide By Zero exception is not masked

Bit 0 – INVALM Invalid Exception Mask bit

Value	Description
1	Invalid exception is masked
0	Invalid exception is not masked

3.6.6 Floating-Point Status Register

Name: FSR

Note: Dynamic floating-point exception status, FSR [6:0]: Dynamic status bits are updated based on the results from each instruction Functional Block and will be updated after execution of each instruction.

Sticky floating-point exception status, FSR [14:8]: Sticky status bits can be set based on the results from each instruction Functional Block but cannot be cleared by hardware (other than at device Reset), and therefore represent a history of status since the last time the sticky bits were cleared. The FSR bits can be cleared through software.

Floating point compare status, FSR [19:16]: Status generated by executing a floating-point compare (FCPQ/FCPS) instruction. Used individually or combined to generate the floating-point branch conditions used by the CPU CBRAn instructions.

Floating-point test status, FSR [27:24]: Floating-point datum characteristic status generated by executing the floating-point test (FTST) instruction.

Bit	31	30	29	28	27	26	25	24
				SUB	INF	FN	FZ	FNAN
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
					GT	LT	EQ	UN
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
		SUBOS	HUGIS	INXS	UDFS	OVFS	DIV0S	INVALS
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		SUBO	HUGI	INX	UDF	OVF	DIV0	INVAL
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 28 – SUB (FTST) Subnormal Status bit

Value	Description
1	Operand is subnormal
0	Operand result is not subnormal

Bit 27 – INF (FTST) Infinite Status bit

Value	Description
1	Operand is infinite
0	Operand is not infinite

Bit 26 – FN (FTST) Negative Status bit

Value	Description
1	Operand is negative
0	Operand is not negative

Bit 25 – FZ (FTST) Zero Status bit

Value	Description
1	Operand is zero
0	Operand is not zero

Bit 24 – FNAN (FTST) Not a Number Status bit

Value	Description
1	Operand is a NaN (qNaN or sNaN) value
0	Operand is not a NaN value

Bit 19 – GT (FCPS/FCPQ) Greater Than Status bit

Value	Description
1	Minuend is greater than the subtrahend (Fb > Fs)
0	Minuend is not greater than the subtrahend (Fb ≤ Fs)

Bit 18 – LT (FCPS/FCPQ) Less Than Status bit

Value	Description
1	Minuend is less than the subtrahend (Fb < Fs)
0	Minuend is not less than the subtrahend (Fb ≥ Fs)

Bit 17 – EQ (FCPS/FCPQ) Equal Status bit

Value	Description
1	Minuend is equal to the subtrahend (Fb = Fs)
0	Minuend is not equal to the subtrahend (Fb != Fs)

Bit 16 – UN (FCPS/FCPQ) Unordered Status bit

Value	Description
1	Either or both operands are NaN values
0	Neither operands are NaN values

Bit 14 – SUBOS Sticky Subnormal Operand Exception Flag bit

Value	Description
1	Subnormal Operand exception has just occurred, or at some time in the past
0	Subnormal Operand exception has not occurred

Bit 13 – HUGIS Sticky Huge Integer Exception Flag bit

Value	Description
1	Huge Integer exception has just occurred, or at some time in the past
0	Huge Integer exception has not occurred

Bit 12 – INXS Sticky Inexact Exception Flag bit

Value	Description
1	Inexact exception has just occurred, or at some time in the past
0	Inexact exception has not occurred

Bit 11 – UDFS Sticky Underflow Exception Flag bit

Value	Description
1	Underflow exception has just occurred, or at some time in the past
0	Underflow exception has not occurred

Bit 10 – OVFS Sticky Overflow Exception Flag bit

Value	Description
1	Overflow exception has just occurred, or at some time in the past
0	Overflow exception has not occurred

Bit 9 – DIV0S Sticky Divide by Zero Exception Flag bit

Value	Description
1	Divide by Zero exception has just occurred, or at some time in the past
0	Divide by Zero exception has not occurred

Bit 8 – INVALS Sticky Invalid Exception Flag bit

Value	Description
1	Invalid exception has just occurred, or at some time in the past
0	Invalid exception has not occurred

Bit 6 – SUBO Subnormal Operand Exception Flag bit

Value	Description
1	Subnormal Operand exception has occurred
0	Subnormal Operand exception has not occurred

Bit 5 – HUGI Huge Integer Exception Flag bit

Value	Description
1	Huge Integer exception has occurred
0	Huge Integer exception has not occurred

Bit 4 – INX Inexact Exception Flag bit

Value	Description
1	Inexact exception has occurred
0	Inexact exception has not occurred

Bit 3 – UDF Underflow Exception Flag bit

Value	Description
1	Underflow exception has occurred
0	Underflow exception has not occurred

Bit 2 – OVF Overflow Exception Flag bit

Value	Description
1	Overflow exception has occurred
0	Overflow exception has not occurred

Bit 1 – DIV0 Divide by Zero Exception Flag bit

Value	Description
1	Divide by Zero exception has occurred
0	Divide by Zero exception has not occurred

Bit 0 – INVAL Invalid Exception Flag bit

Value	Description
1	Invalid exception has occurred
0	Invalid exception has not occurred

3.6.7 Floating-Point Exception Address Capture Register

Name: FEAR

Note:

1. FEAR [1] always set to 1'b0.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	FEAR[22:15]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FEAR[14:7]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FEAR[6:0]							EACE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

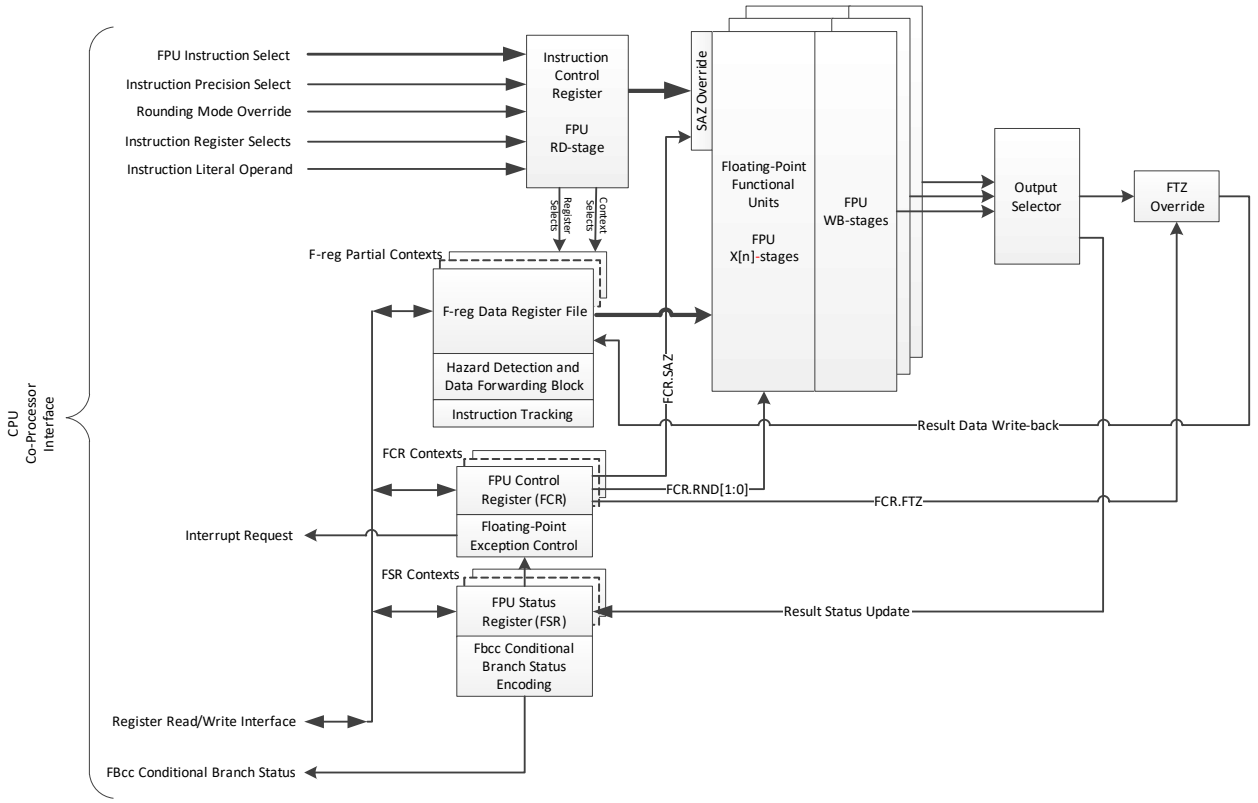
Bits 23:1 – FEAR[22:0] Floating-Point Instruction Exception Address Capture Register bits(1)

Bit 0 – EACE Exception Address Capture Enable bit

Value	Description
1	FEAR register address capture enabled
0	FEAR register address capture disabled (and FEAR [23:0] may contain a captured address)

3.6.8 FPU Module Operation

Figure 3-20. Module Block Diagram



3.6.8.1 Floating-Point Unit Registers

The dsPIC Floating-Point Unit (FPU) provides a large set of working registers (F-regs):

- 32 x 32-bit (Single Precision, F0 ... F31) or
- 16 x 64-bit (Double Precision, F0, F2 ... F28, F30) or
- A mix of the two sizes aligned as shown in Figure 3-21.

In addition to the F-regs, status (FSR) and control (FCR) registers are also supported as shown in Figure 3-21:

- FSR (FPU Status Register, 32-bit): Holds the status of retired floating-point instructions:
  - FSR [6:0]: Instruction “most-recent” exception status
  - FSR [14:8]: Instruction "sticky" exception status
  - FSR [19:16]: FCPS/FCPQ instruction status
  - FSR [28:24]: FTST instruction status
- FCR (FPU Control Register, 16-bit):
  - FCR [6:0]: Exception mask control
  - FCR [9:8]: Rounding mode control
  - FCR [10]: Subnormal result “Flush-to-Zero” (FTZ) control
  - FCR [11]: Subnormal operand “Subnormals-are-Zero” (SAZ) control
- FEAR: (FPU Exception Address Capture Register, 24-bit): Holds the address of the first instruction encountered that causes an exception. All subsequent instructions in the FPU pipeline that

subsequently retire will not affect the FEAR, even if they too generate exceptions. The FEAR is intended for use during debug of the floating-point software.

Note: The FSR msws and lsws may be read/written independently of each other by some instructions.

Note: Although inconsistent with device interrupts, where interrupt controls are referred to as enables (where logic 1 represents enabled), it is more conventional (and in keeping with the IEEE-754 specification) that the FPU exception controls be referred to as masks (where logic 1 represents masked). These bits are all set at Reset, masking exceptions by default.

3.6.8.1.1 FPU Register Access

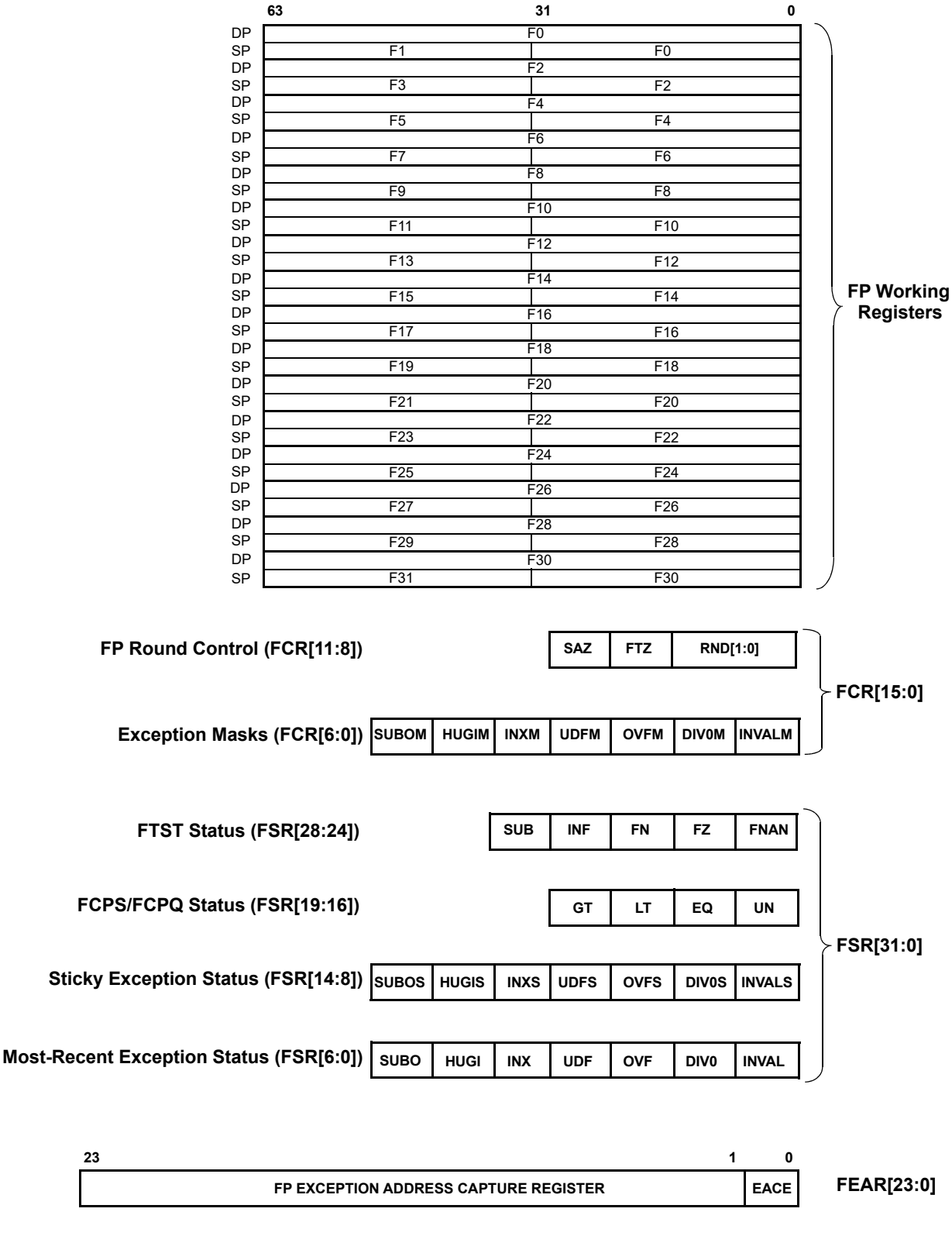
Data may be moved in and out of any FPU register, from OR to W-regs or DS memory, by using dedicated coprocessor register move instructions that execute from within the integer pipeline (refer to `MOVCRW`, `MOVWCR`, `MOVLCR`, `LDWLOCR`, `STWLOCR`, `PUSHCR` and `POPCR` CPU instructions as described in 3.6.8. FPU Module Operation).

All data is moved as 32-bit entities, so Double Precision data moves will require the execution of two instructions (64-bit data moves are not supported in this device).

In addition, the FPU supports `FAND` and `FIOR` instructions that can logically AND or OR a literal value with the FSR (lsw only, exception status), FCR or FEAR (lsw only).

3.6.8.2 FPU Programmer's Model

Figure 3-21. FPU Programmer's Model



3.6.8.3 FPU Register Set

The FPU Programmer's Model of registers is shown in Figure 2-1 and is comprised of floating-point operand registers (F-regs), a floating-point control register (FCR), a floating-point status register (FSR), and a floating-point exception address capture register (FEAR). None of the registers are memory mapped and must be read or written by the CPU using the coprocessor move instructions (`MOVCRW`, `MOVWCR`, `PUSHCR`, `POPCR`, `LDWLOCR`, `STWLOCR`, and `MOVLCR`). The FCR, FSR and FEAR registers may also be subjected to a literal AND or OR operation by the `FAND` and `FIOR` instructions respectively.

3.6.8.3.1 Floating-Point Operand Registers (F-Regs)

To differentiate from the CPU working W-regs, the FPU operand/result data working registers are referred to as F-regs. The FPU supports up to 32 Single Precision values, or up to 16 Double Precision values. Aligned pairs of the F-regs registers (e.g., F1:F0 values may be used to provide data storage for Double Precision values. Single and Double Precision values may be mixed within the register file. Other than data movement in and out of the FPU, all instructions are register-to-register operations within the FPU register set.

The F-regs are not memory-mapped and can only be accessed by the CPU using specific instructions as discussed in 3.6.8.3.3. CPU Access of FPU Registers.

The 32 x 32-bit F-reg array, together with additional register contexts, is implemented as a register file. FPU instructions can have 1, 2 or 3 operands (read sources) and 0 or 1 result destination, and most also update status in the FSR. Registers may be used individually for Single Precision data values or coupled as odd; even pairs (only) should be used to support Double Precision data values (e.g., F1:F0).

Source registers are bound to an instruction when the instruction is issued and are not writable by the CPU until the instruction is committed. At this point, they are clocked into operand registers that drive the target functional block and can therefore be subsequently written. Note that a bound source register may be read at any time.

Destination registers (F-regs and FSR) are bound to an instruction when the instruction is committed and are not accessible by the CPU until the instruction has retired.

Floating-Point Control Register (FCR)

The FCR is comprised of the following bit fields as defined in 3.6.5. FCR.

Floating-Point Exception Mask bits, FCR [6:0]: Each Exception Mask bit corresponds to an Exception Status flag in the FSR. The Mask bit must be clear to allow the exception event to generate an interrupt to the CPU. The Underflow Mask bit (FCR.UDFM) is also used as part of the Flush-to-Zero (FTZ) mode enable as discussed in 3.6.3.2.2. Flush-To-Zero (FTZ).

Floating-Point Rounding mode control, FCR [9:8]: These bits define the global IEEE 754 compatible rounding mode used by the FPU instruction. See 3.6.8.9.3. Rounding Modes.

Floating-Point Subnormal Override mode control, FCR [11:10]: These bits enable the Subnormals-Are-Zero (SAZ) and Flush-to-Zero (FTZ) subnormal override modes supported by the FPU.

3.6.8.3.2 Floating-Point Unit Register Contexts

To speed up real time control systems and other time critical applications, the dsPIC FPU supports multiple register contexts that are tied to Interrupt Priority Levels.

The FPU includes a set of hardware register contexts. Each context includes the FSR, FCR and four register pairs (i.e., F0 through F7). All other F-regs and FEAR are not included and must be saved and restored through software.

The number of supported register contexts matches that of the CPU and is fixed at seven, which represents one context per CPU Interrupt Priority Level (IPL). Should the CPU change context, then the FPU will follow suit, and all subsequent instructions issued to the FPU will execute within that (new) context. However, all FPU instructions issued in a prior context will be allowed to continue to execute and retire within that context, irrespective of the context change within the CPU. Similarly,

any data dependencies that occur within the context of the instruction underway will remain within that context.

As the FSR is part of the register context, exceptions are context specific. Should the FPU change register context, any FPU exceptions generated as a result of the execution of FPU instructions already issued from the prior context will remain pending until the FPU returns to that original context.

Hazard detection is also context based such that each instruction operand and result register is tagged with its own context. Hazards can therefore only exist within the same register context.

This concept extends to the FSR and FCR which have an independent representation within each register context. Consequently, the CPU will not stall (assuming no FSR and/or FCR hazard exists within the current context) if it were to access the FSR or FCR while the FPU continued to execute instructions issued from within a different context. These instructions would have access to their own version of the FSR and FCR.

3.6.8.3.3 CPU Access of FPU Registers

The following CPU instructions are provided specifically to support data movement into and out of the coprocessors. The assembler uses the register declarations to direct encoding of the FPU as the target coprocessor within each instruction op code:

- `MOVCRW`: Move any FPU register to a W-reg or DS memory (using indirect addressing).
- `LDWLOCR`: Move the contents of DS memory (read using register+literal offset addressing (`[Ws + Slit14]`)) to any FPU F-reg register.
- `STWLOCR`: Move any FPU F-reg to DS memory (read using register+literal offset addressing (`[Wd+Slit14]`)).
- `PUSHCR`: 16-bit short instruction dedicated to moving any FPU register onto the system stack.
- `MOVWCR`: Move a W-reg or DS memory value (using indirect addressing) to any FPU register.
- `POPCR`: 16-bit short instruction dedicated to moving a value from the system stack to any FPU register.
- `MOVLCR`: Move a 32-bit literal value to any FPU register.

Note: These instructions are referred as `mov.l`, `push.l` or `pop.l`. Please refer to the dsPIC33A Programmer's Reference Manual for the correct syntax of these instructions.

3.6.8.3.4 Intra-FPU Register Moves and Logical Operations

In addition to CPU to/from FPU data movement, the FPU supports instructions that execute within its own pipeline that perform register to register moves or logical operations:

- `FMOV`: Copy any F-reg or F-reg pair into another F-reg or F-reg pair.
- `FMOVC`: Move one of 32 Single or Double Precision constant values into an F-reg or F-reg pair.
- `FAND`: Logically AND a 16-bit literal value (lit16) with the lsw of the FPU FSR, FCR or FEAR.
- `FIOR`: Logically OR a 16-bit literal value (lit16) with the lsw of the FPU FSR, FCR or FEAR.

Note: To allow subsequent instruction to immediately utilize `FAND` and `FIOR` changes to FCR.RND[1:0] and FCR.SAZ control bits without stalls, these bits are manipulated and updated in the first pipeline stage (RD-stage). However, the remaining FCR bits are not written back until the end of the instruction as usual. Consequently, should the CPU need to read the FCR immediately after modification, it will be stalled by the FPU until the `FAND` or `FIOR` instruction has retired.

3.6.8.4 Data Hazard Management

Read-After-Write (RAW) data hazards can arise due to:

- Data dependencies between FPU instructions

- As the result of a register move from an FPU register to the CPU when an FPU instruction underway has not yet completed its result write (to the same register)

Write-After-Read (WAR) data hazards within the FPU pipeline alone are not possible because the pipeline ensures that instruction reads always precede subsequent instruction writes. However, a WAR hazard can arise when the CPU pipeline writes to an FPU register that has yet to be read by a previously issued but stalled FPU instruction.

Write-After-Write (WAW) data hazards are possible should the CPU attempt to write to an FPU register that is also the target of a prior FPU instruction which has not yet completed its result write.

All hazards are detected within the FPU or CPU (or both) and will be mitigated either through data forwarding or pipeline stalls. Refer to 3.6.8.7. FPU Hazards for further details.

3.6.8.5 FPU and CPU Exceptions

Issued FPU instructions that become committed (accepted by the Execute stage) are always atomic with respect to CPU exceptions. No CPU exception (other than a Reset event) can force the FPU to abandon an instruction that is already underway.

CPU exceptions will result in a register context switch in both the CPU and FPU. Furthermore, FPU exceptions are always context specific. That is, any FPU exception occurring after a context switch will remain pending until the FPU returns to the prior context.

FPU exceptions can only be taken and handled when unmasked (referred to as alternate exception handling). The FPU will return the calculated result of each operation and signal any exception via an interrupt to the CPU.

If FPU exceptions are masked, the FPU will return a default result for each operation that generates an exception as defined in Table 3-15. The exception will be signaled by setting the corresponding bit(s) in the FSR, but no interrupt will be issued to the CPU. This is intended to allow code to execute unhindered by exception handling at the time of execution. If required, exception status may be examined at a later time and appropriate action taken.

3.6.8.5.1 Huge Integer and Subnormal Exceptions

In addition to the IEEE 754-2008/2019 compliant exception support, this macro also offers two additional exceptions and associated masks that some users may find useful.

- Huge Integer: FSR.HUGI
  Exception signaled whenever a Float-to-Integer conversion operation (FF2DI and FF2LI) results in an integer value that is larger than the destination register can represent.
- Subnormal Operand: FSR.SUBO
  Exception signaled whenever an operand of an affected instruction is a subnormal value and Subnormals-Are-Zeros (SAZ) mode is disabled (FCR.SAZ = 0). This is the only exception that can be triggered by an operand source condition (all others are related to result conditions).

Table 3-15. Default Exception Results

Exception	FSR Bit Name	Default Result
Invalid	INVAL(2)	Distinguished qNaN or quieted sNaN or Largest integer result (for FF2DI/FF2LI only)
Divide By Zero	DIV0	Correctly signed Infinity(3)

Notes:

1. Under default exception handling, UDF is only set (along with INX) if the result is an inexact underflow. Applies irrespective of whether FTZ mode is enabled or not.
2. FCPS and FCPQ do not generate a result other than an FSR update. However, INVAL will be set by FCPS if either or both operands are a qNaN or sNaN, or by FCPQ if either or both operands are an sNaN.
3. 0/0 is a special case (where both the dividend and divisor are not finite) which will return the distinguished qNaN as the result. INVAL will be set but DIV0 will not.

.....continued

Exception	FSR Bit Name	Default Result		
Overflow	OVF	Rounding Mode	Nearest (Even)	Infinity with sign of exact result
			Zero	Most positive finite number with sign of exact result
			+Infinity	Positive overflow: +Infinity Negative overflow: Most negative finite number
			-Infinity	Positive overflow: Most positive finite number Negative overflow: -Infinity
Underflow	UDF(1)	FCR.FTZ = 0: Rounded subnormal result		
		FCR.FTZ = 1; Zero with sign of exact result		
Inexact	INX	Rounded (inexact) result		
Huge Integer	HUGI	Largest integer value with sign of input operand		
Subnormal Operator	SUBO	N/A (input operand exception)		

Notes:

1. Under default exception handling, UDF is only set (along with INX) if the result is an inexact underflow. Applies irrespective of whether FTZ mode is enabled or not.
2. FCPS and FCPQ do not generate a result other than an FSR update. However, INVAL will be set by FCPS if either or both operands are a qNaN or sNaN, or by FCPQ if either or both operands are an sNaN.
3. 0/0 is a special case (where both the dividend and divisor are not finite) which will return the distinguished qNaN as the result. INVAL will be set but DIV0 will not.

3.6.8.6 CPU to FPU Interface

The CPU can issue instructions to a coprocessor (FPU), and directly read and write FPU registers. However, coprocessors otherwise operate independently of the CPU instruction pipeline, executing their instructions within their own pipeline hardware.

An FPU can only receive, send and process data that is funneled through (and under the direction of) the CPU. No CPU addressing capability is shared with an FPU. Consequently, an FPU can only support register direct addressing for all instruction source or destination addressing modes that target a FPU register. Data flow to and from each FPU is controlled using dedicated move instructions that execute within the CPU. Because the CPU and FPU pipelines execute independently, data related hazards that may arise when moving data between the CPU and an FPU are mitigated using a simple request/grant bus which will stall the CPU as needed.

The CPU supports speculative execution of instructions that immediately follow a conditional branch. These could be FPU instructions, so a mechanism exists to allow the CPU to cleanly kill these instructions should the branch prediction prove incorrect.

In case an FPU SFR read is killed, all FPU SFR (e.g., status and control registers) are defined such that a read of any SFR is not destructive within itself. This will avoid the possibility of a killed SFR read affecting the state of the FPU.

3.6.8.6.1 FPU Pipeline Operation

The CPU decodes all coprocessor instructions during the F-stage. The source and destination coprocessor registers are extracted from the opcode and supplied to the coprocessor, along with a corresponding instruction select and control signals such that no instruction decode is necessary within the coprocessor.

The FPU pipeline stages consist of Read (RD), Execute (X[n]) and Write-Back (WB) stages. The Read and Write-Back stages consist of a single register and are common to all instructions. The Execute stage consists of as many stages as required to execute the specific instruction (i.e., X [0], X[1]..... X[n]) but at least X [0].

The CPU pipeline F-stage and A-stage fetch can issue FPU instructions respectively as shown in Figure 3-23. The pipeline can suffer both structural and data hazards, as discussed later in 3.6.8.7.1. FPU Structural Hazards and FPU Functional Block Unavailable, respectively, which can result in CPU and FPU pipeline stalls, as shown in the corresponding diagrams.

One instruction may be issued into the RD-stage, where it will remain for one cycle (hazards aside) until dispatched into the X [0] stage. The number of cycles each instruction remains within the execute phase varies depending upon the operation. In order to avoid stalling the pipeline for the duration of any long instruction, up to four instructions may be dispatched into X[0] and executed concurrently (structural hazards aside).

Instructions retire in the same order in which they are issued. As a consequence of being able to execute multiple instructions with varying execution times, the pipeline Instruction/Hazard Tracker logic is designed to ensure that in-order retirement is maintained.

All instructions with an execution latency of four cycles or less are implemented such that the execution stages are fully pipelined. Consequently, assuming no data dependencies (hazards) arise, these instructions can be repeatedly issued at a rate of one per cycle (and receive their results at a rate of one per cycle after an initial execution latency), without incurring a structural hazard stall.

For instructions where the execution latency exceeds four cycles (FDIV and FSQRT), the FPU pipeline will fill the instruction and then stall subsequent instructions (due to a structural hazard) until the required execution resource becomes available.

- FDIV: Floating-point divide is implemented as an iterative operation such that the input data cannot be pipelined until all iterations have completed and the result is passed onto the adjustment stage within the Functional Block. For example, should the CPU issue two sequential FDIV instructions, the second FDIV instruction will stall in the RD-stage until the first FDIV enters the final execution cycle, at which point the second FDIV may be dispatched into execute stage to commence execution.
- FSQRT: Floating-point square root requires 10 (Single Precision) or 13 (Double Precision) cycles to execute. The hazard tracker can handle up to four issued instructions, so an FSQRT followed by up to three sequential FPU instructions (including FSQRT) may be executing at any one time. The CPU may issue one more instruction, but it will remain in the RD-stage until the oldest FSQRT instruction underway enters the WB-stage, six cycles later, and subsequently retires. At this point, one slot within the hazard tracker is now available for use, and the pending FPU instruction will be committed for execution. Another FSQRT instruction will retire in the next cycle, opening another hazard tracker slot for another issued FSQRT instruction, and so forth, until the hazard tracker is full again and the pipeline must wait a further six cycles for the initial FSQRT to retire. For FSQRT alone, the best case block repeat rate is therefore one per cycle for the initial 4 FSQRT instructions issued, with a subsequent four FSQRT instructions to be issued after six (Single Precision) or nine (Double Precision) cycles have passed. This supports an average execution time of (4+6)/4 or 2.5 cycles/instruction (Single Precision) or (4+9)/4 or 3.25 cycles/instruction (Double Precision).

FPU Read Stage

The FPU pipeline RD-stage receives instructions issued by the CPU. The CPU issues FPU instructions from the A-stage into the FPU RD stage which consists of a single register, such that only one FPU instruction can be held at any one time. The instruction is committed when it is dispatched to X[0], where it will start execution. X[0] holds the instruction such that the CPU is free to issue another instruction into the RD-stage.

The RD-stage is also subject to hazard checks and can therefore be stalled. Should a RAW hazard be detected with a prior instruction that is already executing within the FPU pipeline, the hazard will be detected in the RD-stage which will then be stalled until such time that the hazard is resolved.

Should the CPU subsequently attempt to issue additional FPU instructions, the RD-stage will not be able to accept them so will also stall the CPU until such time that the RAW hazard has been resolved. From the CPU perspective, this scenario is viewed as a structural hazard.

The RD-stage will also stall the CPU under the following conditions:

1. Whenever the number of instructions (default value is four) are in their execute X[n] stages, an instruction is pending in the RD-stage, and the CPU is attempting to issue a further instruction. In this situation, the Instruction/Hazard Tracker is full so the FPU cannot dispatch another instruction from the RD-stage into X [0] until one of the instructions currently executing passes into the WB-stage (refer to 3.6.8.7.1. FPU Structural Hazards). Assuming the default value is four, this can occur when the pipeline is executing instructions that take longer than four cycles to execute, and additional FPU instructions are issued while the long instruction is still executing (i.e., not yet in the WB-stage). The longer instruction(s) execute and retire at a rate which is slower than the rate at which the Instruction/Hazard Tracker can be filled, resulting in the CPU being stalled.
2. Whenever the CPU attempts to issue more than two FDIV instructions while a previously issued and dispatched FDIV instruction is still executing (i.e., not yet in the WB-stage). FDIV is a special case where no more than one instance can be executed within the pipeline at any one time. Consequently, executing another FDIV while a prior instance is still executing will cause this second FDIV to be issued but held pending in the RD-stage (i.e., CPU will not stall). But attempting to issue a third FDIV instruction while the pending (second) instance has not yet been dispatched to X [0], will result in a CPU (issue) stall. The RD-stage also includes special logic to support the FAND and FIOR operations (refer to FAND and FIOR Instructions).

FPU Execute Stage

Each instruction may consist of one or more execute stages depending upon the functional block targeted by the operation. When the instruction enters the X [0]-stage, it is registered such that the RD-stage is free to receive another instruction issued by the CPU.

All instructions (other than FDIV) are pipelined through as many X[n] stages as deemed necessary to meet the timing requirements.

The pipeline stages will be added such that the propagation delay of each is as balanced as possible, and that sequential issue of the same instruction may be fully pipelined (i.e., instructions using the same Functional Block may be sequentially issued without incurring a structural hazard in the execute stage).

FPU Write-Back Stage

The WB-stage captures each Single Precision or Double Precision result as they exit the execute stage in dedicated registers. FPU instruction execution time is variable, but only one instruction is permitted to be in the WB-stage at any one time. If more than one instruction has completed execution and is in a position to retire, the pipeline will determine which instruction to retire to maintain instruction execution order and eliminate any WAW hazards. The instruction will then complete the write-back in one cycle during the WB-stage before being retired. The Instruction/Hazard Tracker logic will ensure instructions enter the WB-stage in the same order as they were issued (refer to 3.6.8.7.3. Instruction/Hazard Tracker).

Prior to writing the result, if FTZ mode is enabled (see 3.6.3.2.2. Flush-To-Zero (FTZ)), the result is modified accordingly if subnormal. This final value is also passed onto the RAW hazard mitigation forwarding logic (see Internal RAW Hazards).

FAND and FIOR Instructions

The FAND and FIOR instructions operate with a 16-bit literal, and can only target the FCR, FSR and FEAR. They are considered a special case as they are executed using custom blocks that are implemented within the RD-stage for some FCR bits and the WB-stage for everything else.

To allow subsequent instruction to immediately use FAND and FIOR changes to FCR.RND [1:0] and FCR.SAZ control bits without (RAW hazard) stalls, these bits are modified during the RD-stage, then updated at the end of the RD-stage such that they are available for immediate use by any subsequent instruction.

The remaining FCR bits and all FSR and FEAR bits are read, modified and written back during the WB-stage. Reading the FSR late (i.e., in the WB-stage rather than the RD-stage) avoids a potential RAW hazard arising between a prior instruction FSR update and a subsequent FAND or FIOR FSR operation.

3.6.8.7 FPU Hazards

The coprocessor interface can suffer from structural and data dependencies as described in the following sections. RAW, WAR and WAW data hazards are possible; RAR hazards are not.

3.6.8.7.1 FPU Structural Hazards

When a requested FPU resource is unavailable, a structural hazard will be detected. This may result in the coprocessor stalling the CPU until the hazard is resolved.

Hazards that arise from actions within the FPU are referred to as “internal” hazards. Those that arise due to actions between the CPU and FPU are referred to as “external” hazards. Depending upon how the CPU/FPU pipeline is viewed (separate or conjoined), some of these hazards may be viewed as either structural (i.e., a resource is unavailable) or data related.

FPU Pipeline Full or Busy

When the CPU attempts to issue an instruction to the coprocessor and it is unable to accept it because the pipeline is full or busy, an external structural hazard will result, and the coprocessor will stall the CPU until such time that the instruction can be accepted.

When an issued instruction is stalled in the FPU RD-stage due to a RAW hazard with a prior currently executing instruction, the FPU pipeline is considered busy such that further FPU instructions cannot be accepted. Consequently, should the CPU attempt to issue any additional FPU instructions while the RD-stage is stalled, the FPU will stall the CPU until such time that the hazard resolves, resulting in an external structural hazard as shown in Figure 3-24.

The pipeline is considered full when the Instruction/Hazard Tracker FIFO is full, which occurs when the number of instructions (default value is four) are active within it, including the one waiting in the RD-stage for dispatch into X[0]. The pipeline will remain full until the oldest instruction enters the WB-stage. Should the CPU attempt to issue another FPU instruction, the FPU will stall the CPU until such time that the Instruction/Hazard Tracker FIFO is no longer full.

FPU Functional Block Unavailable

If the FPU pipeline is not full, and the FPU attempts to dispatch an instruction from the RD-stage that uses a functional block that is already in use by a prior instruction, an internal structural hazard will result, and the RD-stage will be stalled until such time that the functional block is no longer in use. If the CPU attempts to issue another FPU instruction before this occurs, the FPU will then stall the CPU until the hazard resolves.

This scenario can arise as a result of in-order retirement where instructions that target the same functional block will be stalled in the pipeline waiting for slower, older instructions to complete execution. An example is shown in Figure 3-3 where a slow instruction (FSIN) is followed by multiple instructions that target the same MISC_SP functional block. The first FMOV will stall in X [0] waiting for the FSIN to retire, resulting in an internal structural hazard. The subsequently issued FMOV will issue but will be stalled in the RD-stage because it cannot progress into X [0] until the first FMOV is able to move into the WB-stage, another internal structural hazard. As the RD-stage is now stalled,

should the CPU attempt to issue any additional FPU `FMOV` or `FMOVC` instructions (which share the same functional block), the FPU will stall the CPU until such time that the pipeline can advance again, causing an external structural hazard.

This scenario will always arise for sequential issue of the multi-cycle iterative FDIV instruction (all other instructions can be pipelined) as shown in Figure 3-4.

FPU Register Unavailable to Read

When the CPU attempts to read a register that is bound to an issued FPU instruction, an external structural hazard will result and the coprocessor will not be able to read until such time that the register becomes available, creating a read stall for the CPU.

FPU Register Unavailable to Write

When the CPU attempts to write to a register that is bound to an issued FPU instruction, the co-processor will not be able to write until such time that the register becomes available, creating a write stall for the CPU (see FPU WAW Hazards).

3.6.8.7.2 FPU Data Hazards

The coprocessor interface can suffer from data dependencies leading to RAW, WAR and WAW data hazards (RAR hazards are not possible). Unlike the CPU integer pipeline, a coprocessor hazard does not necessarily prevent the pipeline from progressing for other coprocessor instructions, unless subject to other hazards.

Internal RAW Hazards

An internal RAW (Read-After-Write) data dependency hazard will occur when the result of an FPU instruction is not available at the time it is selected as the source operand (F-reg) of a subsequent FPU instruction. The affected instruction will be stalled in the RD-stage until such time that the hazard is resolved.

In order to mitigate the hazards, the coprocessor includes data forwarding paths between the FPU execution result data output and the coprocessor RD-stage (as shown in Figure 3-8). This path will forward the write data value should the write and read instructions target a common register. Forwarding as soon as the result data is available (i.e., prior to the FPU register write) will help mitigate the impact of the hazard.

External RAW Hazards

An external RAW (Read-After-Write) data dependency hazard will occur should the contents of an FPU register be unavailable at the time it is read by the CPU because the register is bound to a previously issued FPU instruction. The coprocessor will detect the hazard and read will be stalled until such time that the register becomes available (i.e., after the result has be written), creating a read stall for the CPU.

In addition, an external RAW hazard will occur if:

- A CPU write to a coprocessor register is immediately followed by the CPU issuing a coprocessor instruction that uses the same register as an operand source.

  or

- A CPU write to a coprocessor register is immediately followed by a CPU read of the same register.

In both of these CPU RAW hazard scenarios, the CPU is responsible for detecting the hazard and inserting the necessary stall cycle for the coprocessor to resolve the hazard. Hazard detection is the same for both scenarios.

In order to resolve these hazards, the coprocessor includes data forwarding paths between the CPU W-stage and both the coprocessor RD-stage (as shown in Figure 3-5) and the CPU read data output (as shown in Figure 3-6). These paths will forward the write data value should the write and read instructions target a common register.

Note: When the CPU attempts to write to an F-reg, it is possible that an instruction in the RD-stage is using the same register as an operand source, and it is stalled as the result of an internal RAW hazard. Forwarding the new CPU write data into this register would then be incorrect because the RD-stage instruction was issued prior to the CPU write instruction. Consequently, should the instruction in the RD-stage have been there for more than one cycle (i.e., be stalled), the FPU will disable the forward path and allow the stall mechanism to recognize the hazard as usual. This will prevent the CPU write from completing until such time that the instruction in the RD-stage has been dispatched to start execution.

CPU write data forwarding to the coprocessor RD-stage allows the CPU to issue a coprocessor instruction earlier than would be possible if the CPU coprocessor write had to complete. CPU write data forwarding to the CPU read data path together with a CPU stall cycle (detected and inserted by the CPU) resolves the (unlikely) hazard that arises when a CPU write is followed immediately by a CPU read of the same coprocessor register. The converse scenario where a CPU read of an FPU register into a W-reg is immediately followed by a CPU write of the same W-reg to another F-reg is shown in Figure 3-7.

FPU WAR Hazards

WAR (Write-After-Read) anti-dependency hazard can occur should the pipeline allow read and write execution to be out of (instruction sequence) order. That is, a WAR hazard will arise whenever an instruction writes to a register before the same register is read by a *prior* instruction. That is, the read and write occur out of execution order resulting in the (older) read instruction ultimately using the (later) write data which would be incorrect.

Under normal sequential execution conditions, a WAR hazard should never arise because the read of all older instructions always precedes the writes of later ones. However, a WAR hazard can arise within the coprocessor pipeline should a slow instruction (e.g., `FPU FSIN`) have a result data dependency (RAW hazard) with a later instruction, and that later instruction is followed by a `MOVWCR` or `POPCR` instruction that targets the same register as the dependency. This is because the dependency will force an FPU pipeline stall until the result data is available and the RAW hazard is resolved, but the `MOVWCR` or `POPCR` move instructions (which do not execute using the FPU pipeline) will not be stalled. Consequently, it is possible that the write from the `MOVWCR` or `POPCR` instruction would occur prior to the stalled instruction continuing execution (after the RAW hazard). The write would then be overwritten by the FPU pipeline and therefore lost. This scenario is detected as a WAR hazard and prevented from happening by stalling the most recent write, such that the write order remains correct. CPU to FPU move instructions that do not target the register involved in the stall will still execute as normal (i.e., without stalling).

The coprocessor must therefore detect the possibility of such a hazard and force in-order execution of all dependent instructions by stalling the most recent CPU write instruction in the W-stage until after the prior read is completed.

An example WAR hazard and its resolution is shown in Figure 3-10. A RAW hazard between the `FSIN.s` and `FMOV.s` instructions will stall `FMOV.s` to resolve the hazard (stall cycles shown in green), but this will also set the pipeline up for a possible WAR hazard because the subsequent `MOVWCR` instruction is not prevented from continuing execution.

As is the case in this example, should the `MOVWCR` instruction destination be the same F-reg as that used by the `FMOV.s` as a source, the `MOVWCR` must be prevented from writing until the prior (`FSIN.s`) has been able to forward the write data to the `MOV.s` RD-stage. The `FSIN.s` and `MOVWCR` enter their respective write stages together, and the FPU prioritizes the CPU write, maintaining correct write ordering. This results in a one cycle stall of `MOVWCR` instruction to resolve the hazard.

FPU WAW Hazards

The WAW (Write-After-Write) hazard is a further consequence of allowing instructions to continue execution while others are stalled or taking longer to execute. As is the case for WAR hazards, instruction writes can end up out of order, leaving an incorrect (stale) value in a destination register.

WAW output dependency is possible because once the coprocessor instruction is issued, the CPU and coprocessor pipelines operate independently. A multicycle coprocessor instruction may therefore complete after one or more CPU instructions that were subsequently issued (i.e., out of order). A WAW hazard will exist when the CPU instruction is ready to retire before the coprocessor instruction retires and either:

- The same register is a destination for both the coprocessor instruction and the CPU instruction that follows it.

  or

- The CPU instruction write targets a coprocessor register that is being used by the prior coprocessor instruction.

In both cases, the resource cannot be shared.

An internal WAW hazard can arise between successive FPU instructions that have differing execution time. However, each issued instruction is tracked by pushing its associated functional unit ID into a FIFO, which is emptied in the same order as it is filled when instructions move results from their functional units into the WB-stage. Should an expected (from the FIFO) functional unit result not be ready, this knowledge is used in the write stage to complete the destination write in the correct sequence, stalling those instructions that arrive out of order, thereby eliminating the WAW hazard.

If the CPU and FPU pipelines are viewed as conjoined, a WAW hazard is also possible should the CPU attempt to write a value to the same register as that also targeted by a previously issued FPU instruction whose write has not yet completed. However, access to the write target(s) of an instruction is inhibited as soon as the instruction is committed (see 3.6.8.5. FPU and CPU Exceptions). Consequently, any attempt by the CPU to write to an FPU register that is already bound to a prior FPU instruction being executed will result in the write grant failure (and the CPU write stalling).

An example WAW hazard and its resolution is shown in Figure 3-11 for an FPU instruction that requires three iterations of the execute stage to complete. This results in a two cycle write stall within the CPU instruction pipeline.

Note: For the purpose of WAW hazard detection, the FSR is considered as a single entity.

Note: The FSR is bound to all FPU instructions except for `FMOVC` and `FMOV` (these ops do not update the FSR), and FAND and FIOR unless they will modify the FSR. The FEAR is bound to all FPU instructions except for `FMOVC`, `FMOV` and `FTST`. It is also not bound to FAND or FIOR unless it will be modified by them. Note that this applies irrespective of whether FEAR is enabled or not (i.e., FEAR.EACE is a “don’t care” with respect to FEAR hazard detection).

3.6.8.7.3 Instruction/Hazard Tracker

The Instruction/Hazard Tracker is a mechanism whereby hazard related information required while an instruction is progressing through the execute stages is captured in a FIFO for each issued instruction when that instruction is committed and enters the FPU pipeline X [0]-stage. The FIFO depth (Default is four) defines how many instructions may be sequentially dispatched into the execution stage before it is regarded as full.

Each FIFO entry includes the following information which is used during the X-stages:

1. Entry valid flag.
2. Flags to indicate which Functional Block (function and operation precision) is targeted.
3. Operand source register identification and valid flag such that RAW hazards may be identified as the instruction progresses.
4. Flags to support Single Precision and Double Precision NaN propagation logic.
5. Flag to indicate if instruction is FDIV or FSUB (where the operand order is reversed).
6. Flag to indicate if instruction is FMAC (special case for NaN propagation).

Each FIFO entry requires a 'valid' bit which is clear whenever the entry is empty or after it has been used in the WB-stage. This bit will inhibit any associated hazard detection after an instruction has retired.

Operation precision partially identifies the selected Functional Block but also directs the hazard logic. Single Precision operations need only check for hazards involving single F-regs whereas Double Precision must check F-reg pairs for hazards.

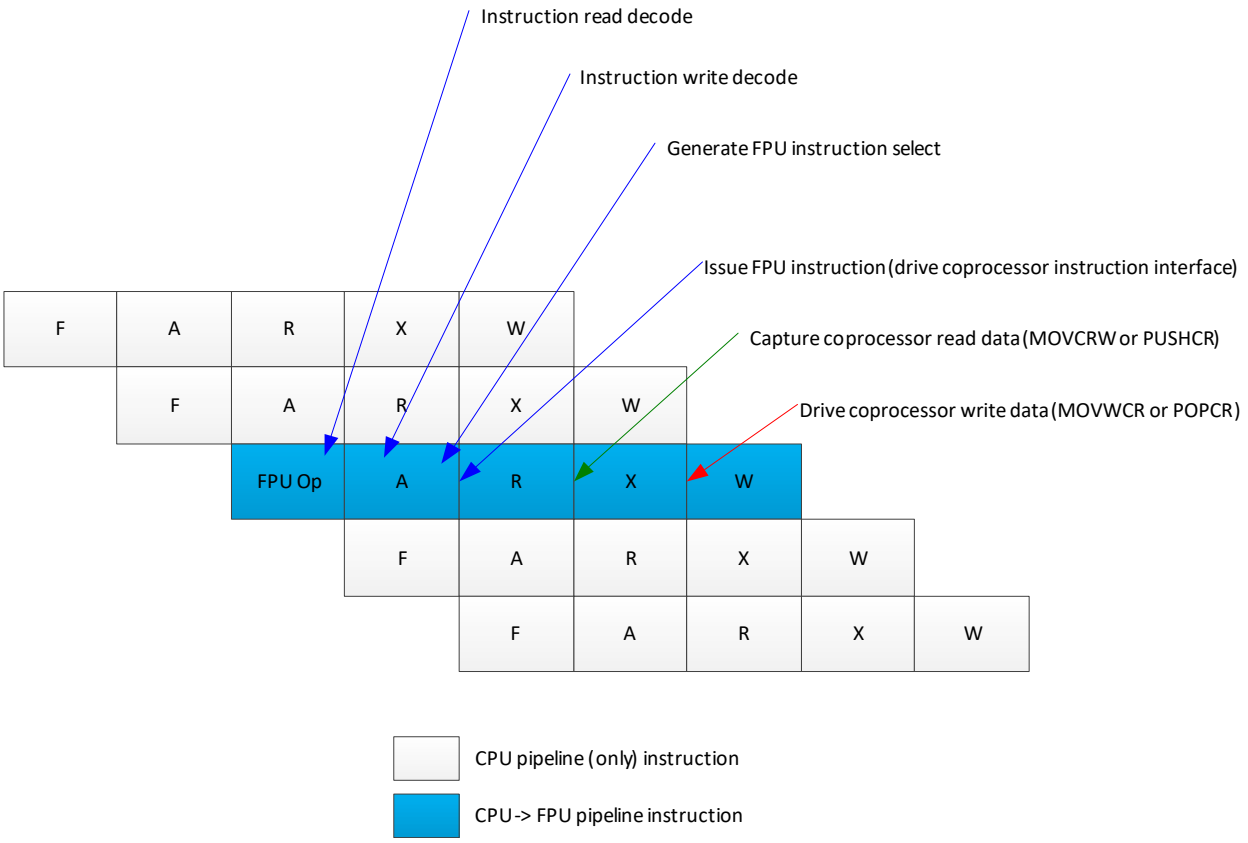
Operand register identification and valid flags log which F-regs are used for operands (not all instructions require all three source operands) for hazard tracking. In addition, each FIFO entry includes the following information (also detected in the RD-stage) which is used during the instruction WB-stage:

1. Flags to indicate if any result is to be written to an F-reg and whether the FSR is to be updated.
2. Result destination register (and context) identification (defined as DP targets). Additional flags select the active registers (i.e., DP F-reg pair or one of two SP F-reg destinations).
3. Flag to indicate if instruction permits FTZ override of result.
4. CPU A-stage instruction address to capture in the FEAR (if enabled) should the instruction generate an exception.
5. Flags to indicate if the instruction is a FAND or IOR, and the associated FSR/FCR/FEAR target register select bits.
6. The presence of a subnormal operand (when SAZ mode is disabled) is captured and used to signal the subnormal exception (i.e., at the same time as any other exceptions the instruction may generate).

3.6.8.7.4 CPU Write Stalls

Whenever the CPU encounters a write stall, the entire integer pipeline is stalled (because the CPU only supports in-order execution). No subsequent instruction is permitted to move into the W-stage to retire until the write stall is resolved. Different Pipeline stages are explained in 3.6.8.6.1. FPU Pipeline Operation.

Figure 3-22. CPU Pipeline Coprocessor Interface Flow



Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-23. CPU Pipeline Coprocessor Issue Flow

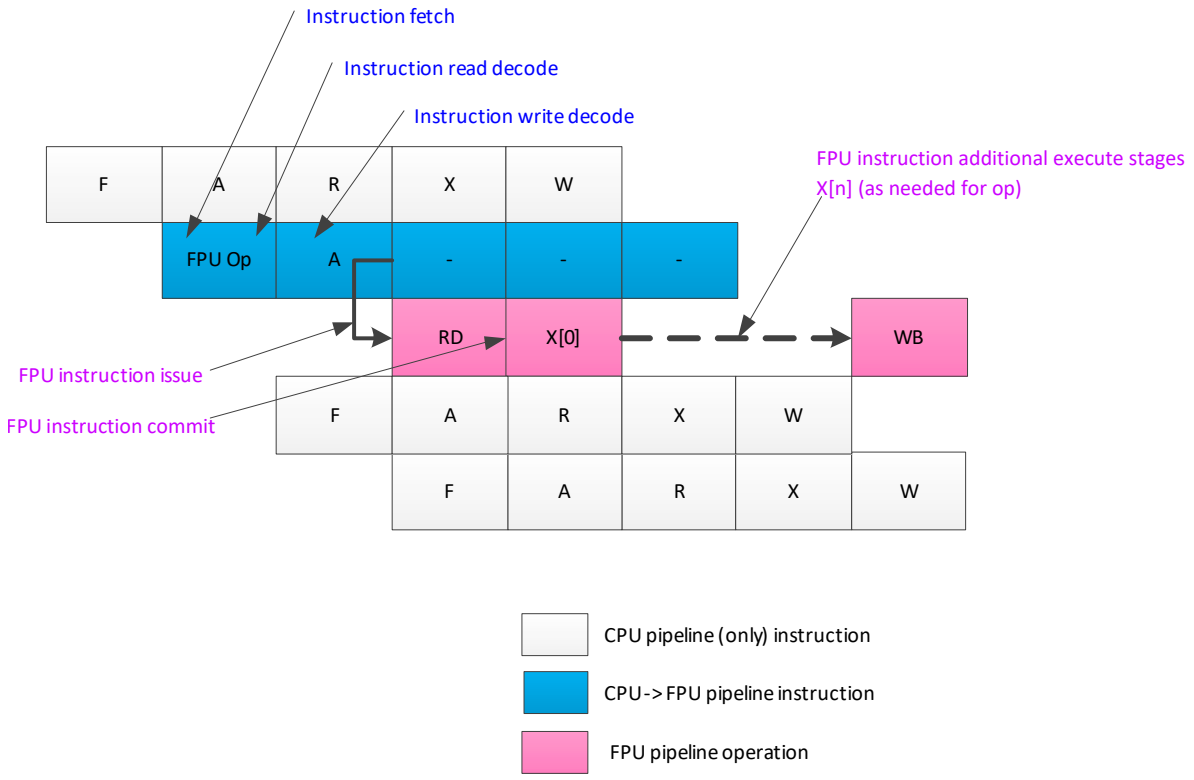
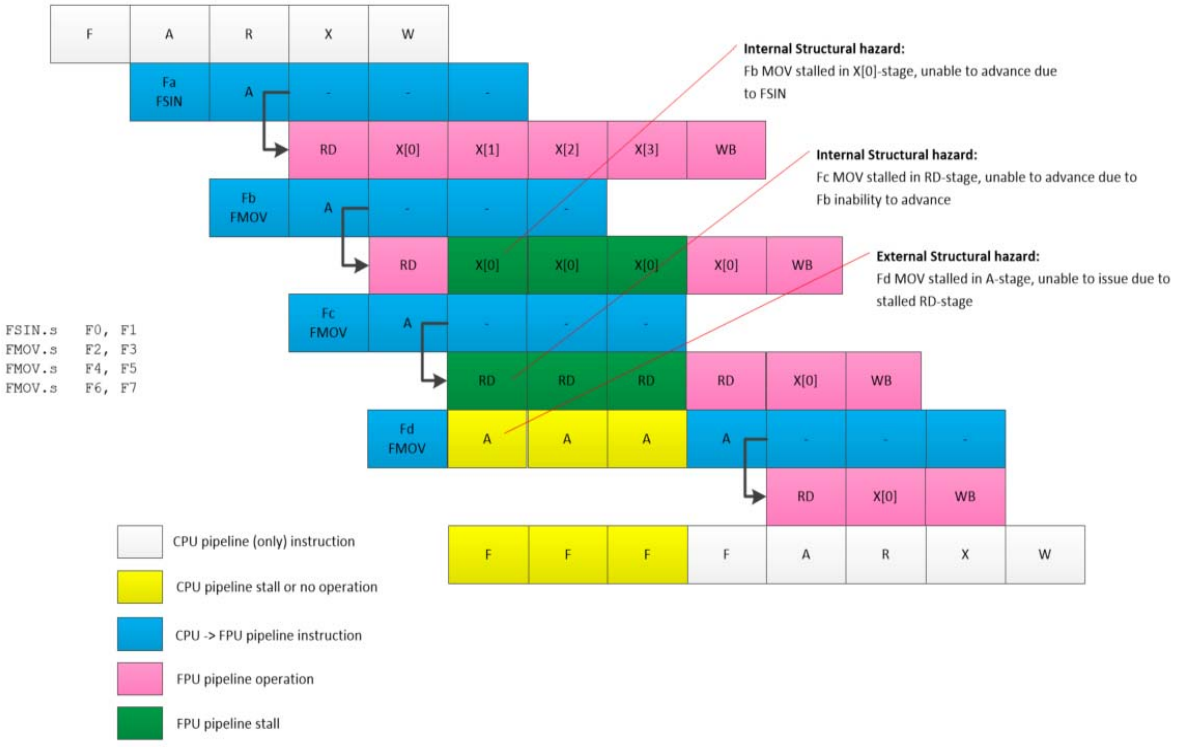
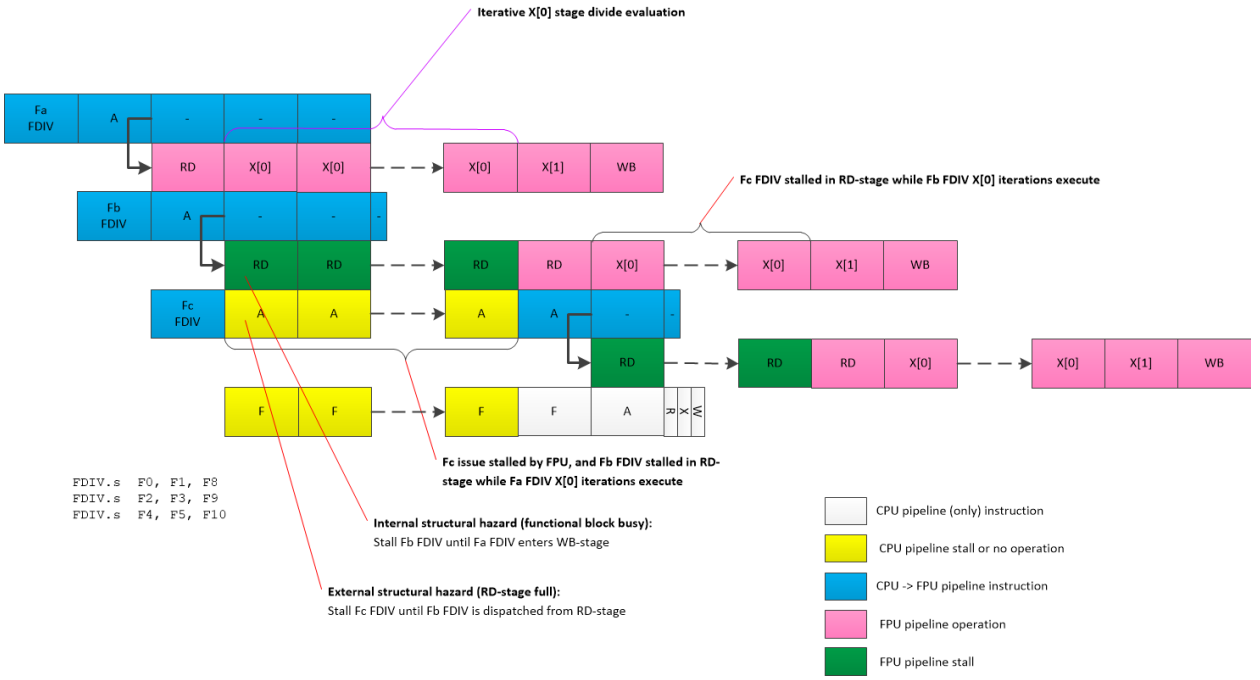


Figure 3-24. Pipeline and Functional Block Busy Internal/External Structural Hazards



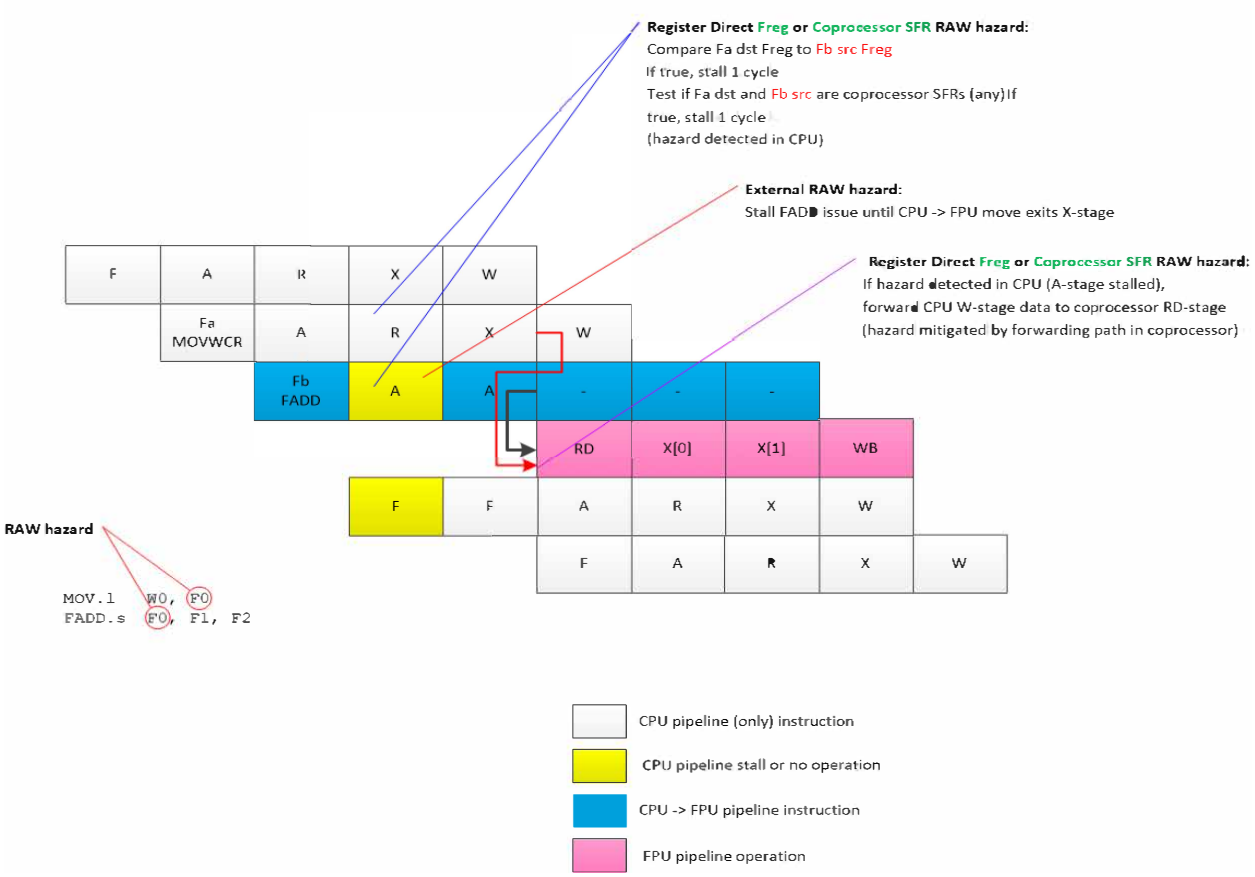
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-25. FDIV Pipeline and Functional Block Busy Internal/External Structural Hazards



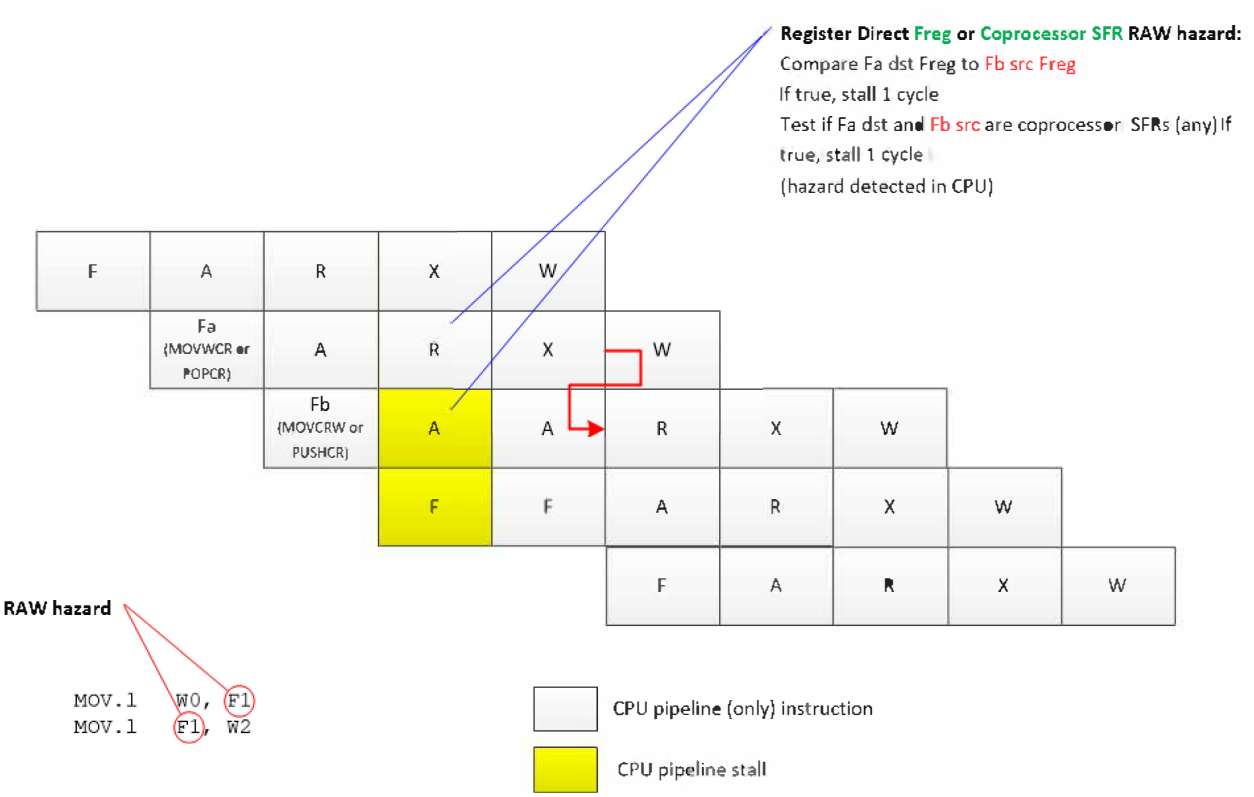
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-26. External RAW Hazard (CPU Write Data to FPU Read Forwarding)



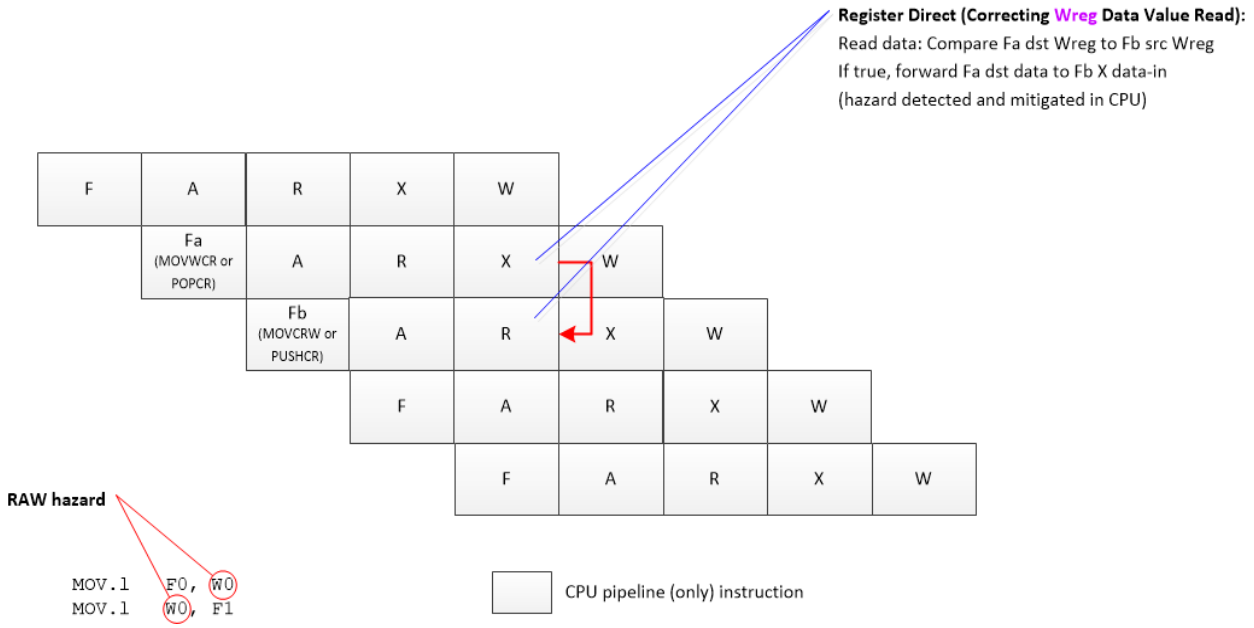
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-27. External Raw Hazard (CPU F-REG Write Data to CPU F-REG Read Forwarding)



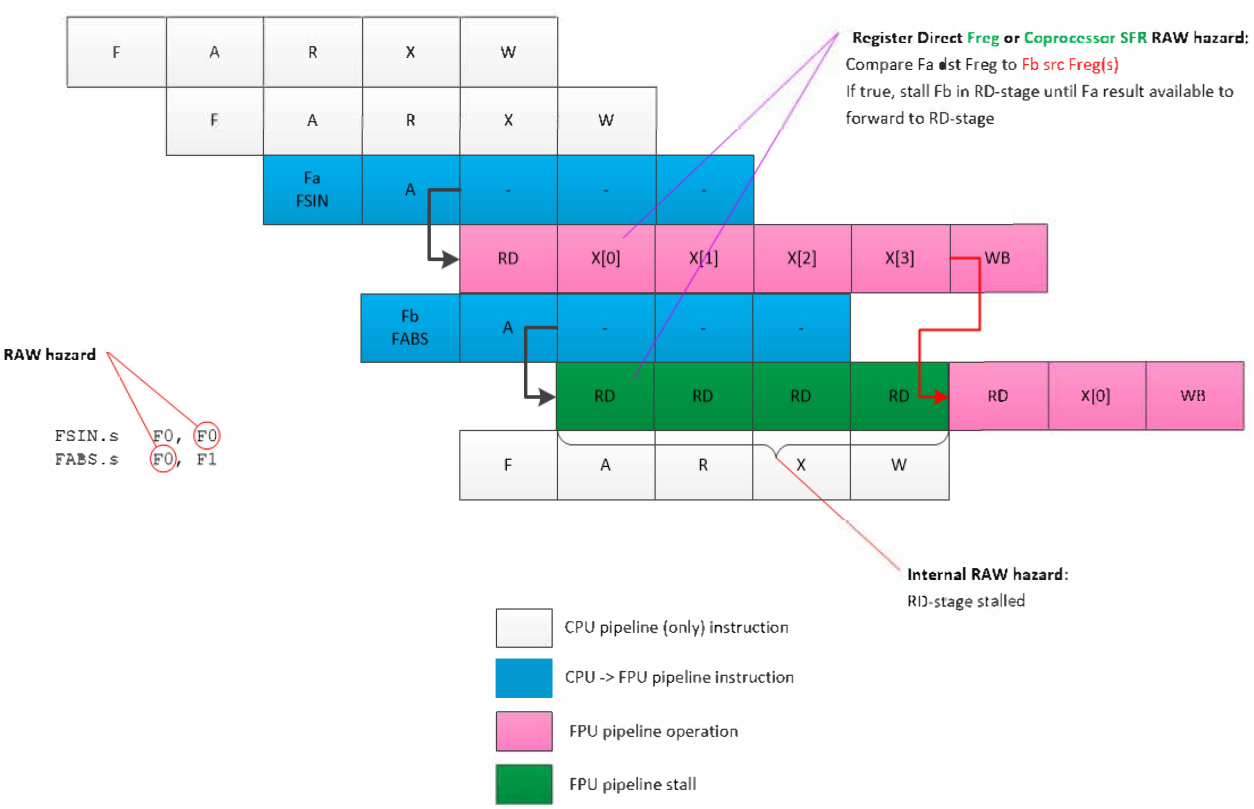
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-28. External Raw Hazard (CPU W-REG Write Data to CPU W-REG Read Forwarding)



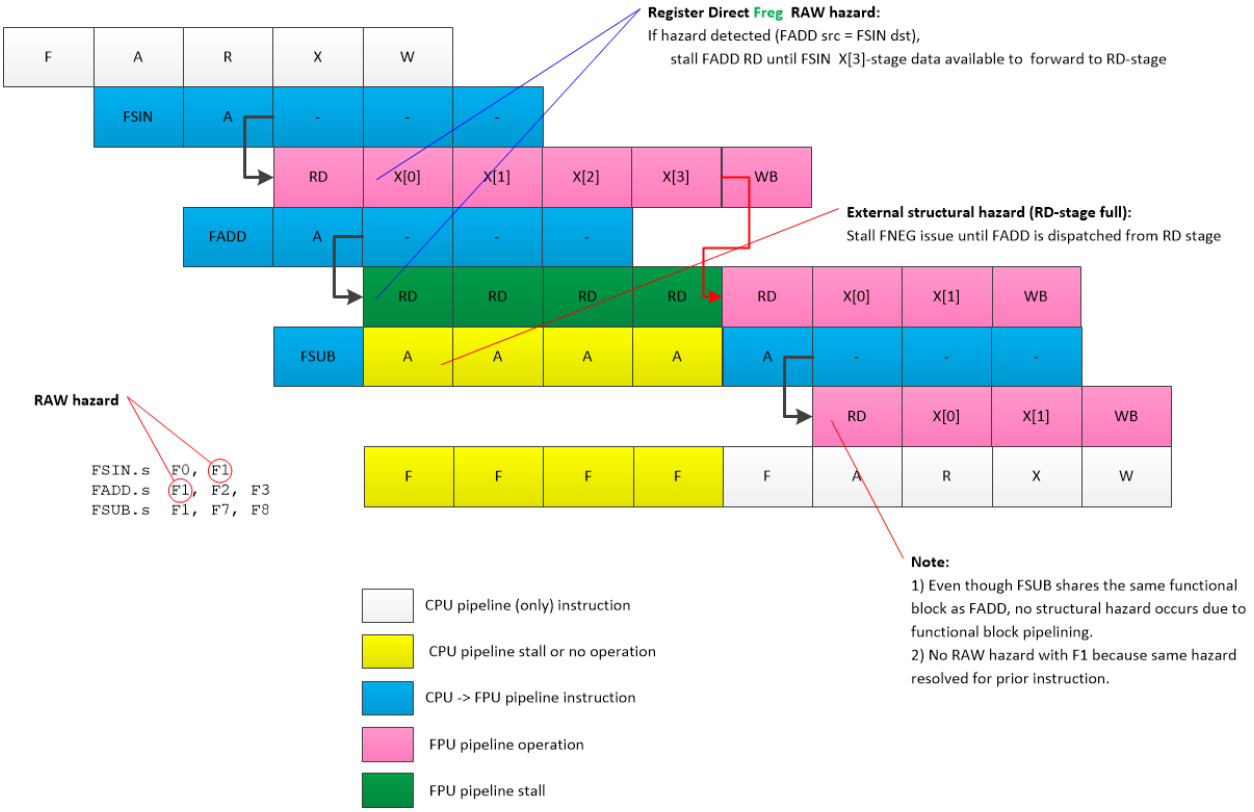
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-29. Internal Raw Hazard (FPU Write Data to FPU Read Forwarding)



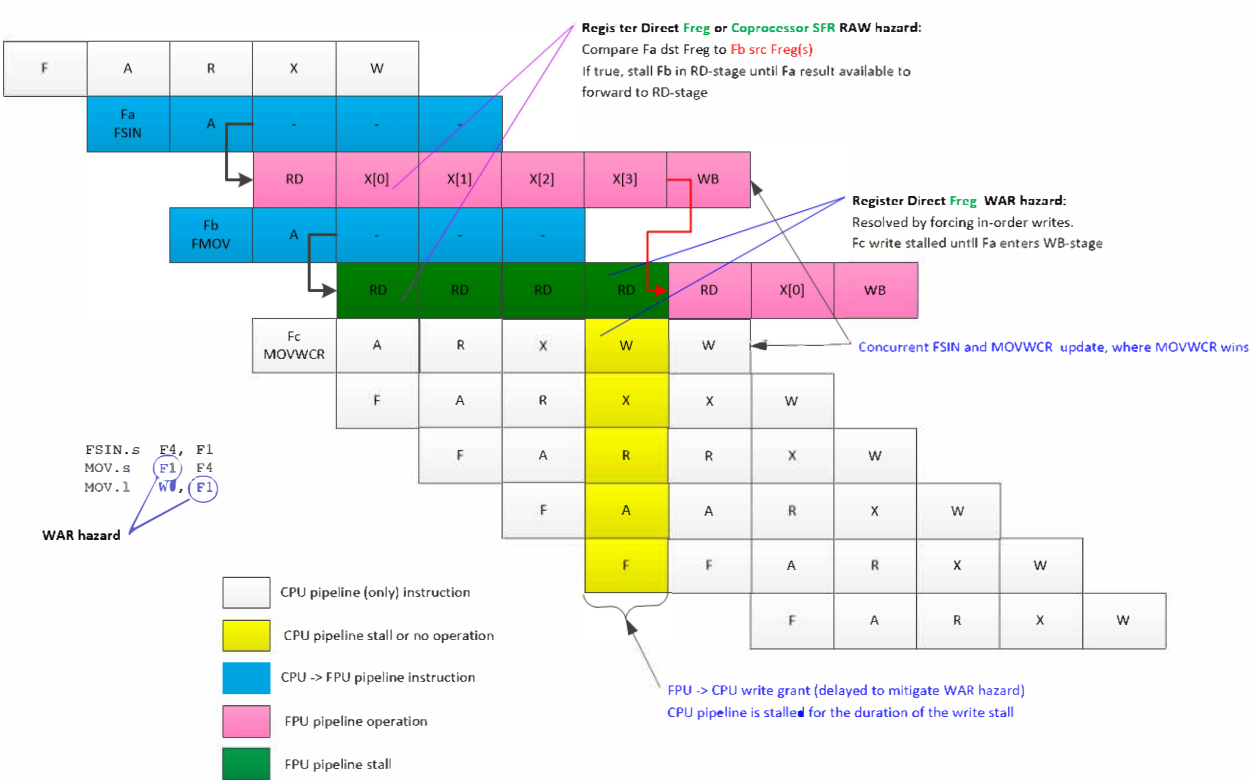
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-30. Internal Raw Hazard, External and Internal Structural Hazards



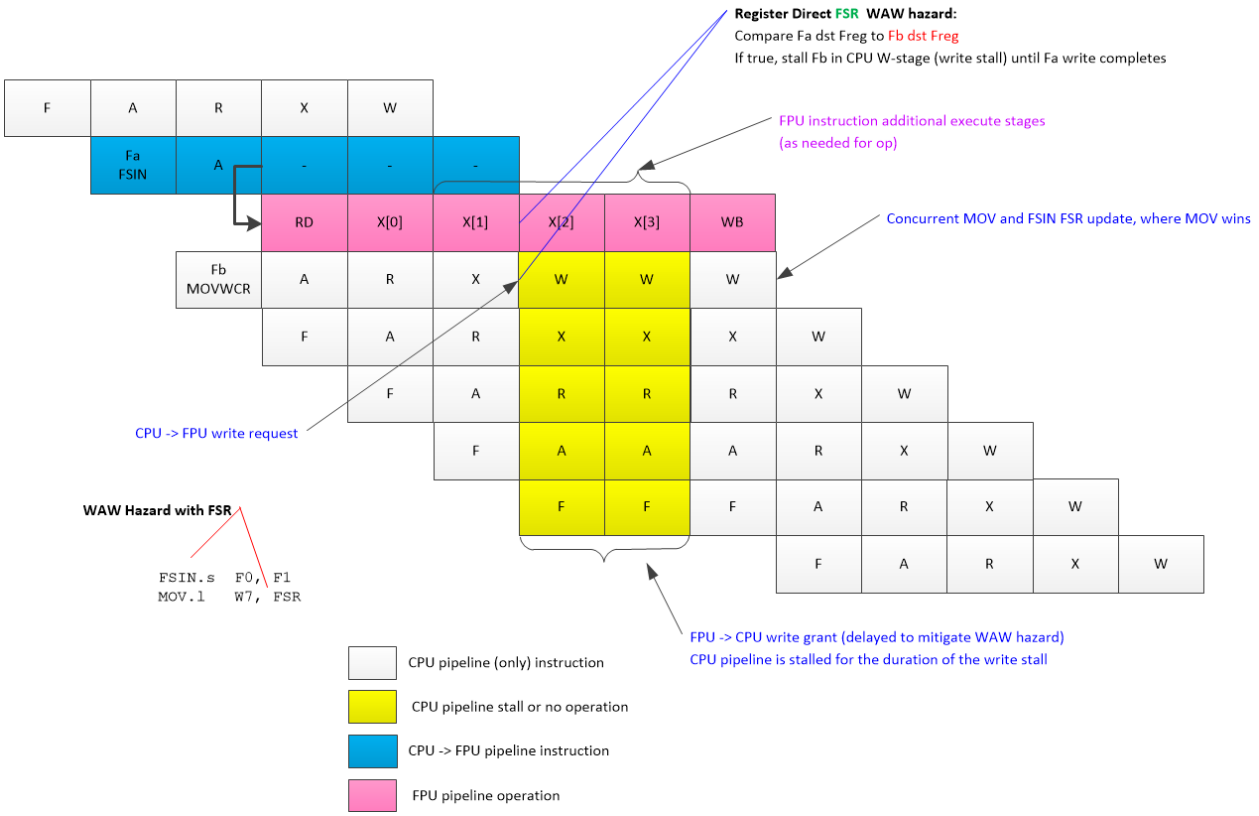
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-31. FPU WAR Hazard



Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-32. CPU/FPU WAW Hazard



Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

3.6.8.8 Operand Pre-Processing

Floating-point operands are subject to examination during the RD-stage in order to implement NaN propagation and the subnormal value override function. This is necessary to apply rules that determine the outcome in the presence of one or more NaN input values and evaluate operands for special conditions.

3.6.8.8.1 NaN Propagation Operand Detection

For instructions that generate a result, special propagation rules apply when one or both source operands are NaN values, such that sNaNs can be successfully used as “tracer” values. Should a NaN be deemed as propagated, then it will replace the operation result.

With reference to NaN Propagation, all instructions will examine the operands for NaN values during the RD-stage:

- Two operand instructions:
  If one or both operands are NaN values, the RD-stage will apply a propagation priority as shown in Table 3-13.
- Three operand FMAC instruction:
  The source operands are examined in the RD-stage as usual but in conjunction with the selected intermediate result, and any NaN values detected are propagated as shown in Table 3-14.
- Three operand FFLIM instruction:
  If one or both limit operands are NaN values, the RD-stage will apply a propagation priority as shown in Table 3-12. If the FFLIM input value is a NaN, the limit values are ignored and the input NaN value is propagated (quieted if an sNaN).

In all cases, if a NaN is to be propagated, the corresponding NaN value is entered as the operand value in the Instruction/Hazard Tracker FIFO entry for that instruction. The instruction FIFO entry also sets a flag to indicate that NaN propagation is enabled.

3.6.8.8.2 Subnormals Operands

The FPU supports a subnormal operand override mode, Subnormals-Are-Zero (SAZ), the functionality of which is defined in 3.6.3.2.3. Subnormal Operand Exception. Subnormals- Are-Zero (SAZ) mode is enabled when FCR.SAZ is set.

Should a subnormal operand be detected when SAZ mode is disabled, the subnormal exception will be signaled by setting FSR.SUBO (and FSR.SUBOS if not already set) during the WB-stage (i.e., at the same time as when all other exceptions are signaled). If SAZ mode is enabled, the subnormal exception will not be signaled.

Note: SAZ mode is not applicable to `FAND`, `FIOR`, `FMOV`, `FMOVC` or any CPU to/from FPU data move instruction, none of which can modify any FPU status. In addition, SAZ mode is ignored by `FTST` such that a subnormal operand will always be recognized as such by the instruction, irrespective of the state of FCR.SAZ. However, SAZ mode can influence FF2LI/FF2DI operands. In these cases, subnormal or zero operands will write the same result (integer value of 0). But if the operand is subnormal and SAZ mode disabled, a subnormal exception will also be signaled. Conversely, if the operand is subnormal and SAZ mode enabled, a subnormal exception will not be signaled.

3.6.8.9 Result Post-Processing

Floating-point results are subject to examination during the WB-stage to implement the subnormal result override Flush-To-Zero (FTZ) mode and NaN propagation results.

3.6.8.9.1 Subnormal Results

The FPU supports a subnormal result override mode, Flush-To-Zero (FTZ), the functionality of which is defined in 3.6.3.2.2. Flush-To-Zero (FTZ). Flush-To-Zero (FTZ) is enabled when both FCR.FTZ and FCR.UDFM are set. Should the underflow exception be unmasked (FCR.UDFM = 0), then the FCR.FTZ bit will have no effect.

This mode is implemented within the WB-stage such that results written to the destination register (and those forwarded) will be adjusted accordingly if FTZ mode is enabled. The FCR.FTZ bit is only examined during the WB-stage of an instruction such that it may be modified as late as the cycle before the instruction enters the WB-stage.

Note: FTZ mode is not applicable to `FAND`, `FIOR`, `FMOV`, `FMOVC` or any CPU to/from FPU data move instruction, none of which can modify any FPU status. It is also not applicable to FTST because the FSR is the only possible destination for this operation. In addition, FTZ mode will have no effect on FF2LI/FF2DI and FLI2F/FDI2F instruction results because - FF2LI/FF2DI results are integers and - FLI2F/FDI2F destination data may be 0 but never subnormal.

3.6.8.9.2 NaN Propagation Result Write

NaN operand values are detected in the RD-stage, prioritized, and then passed (via an Instruction/ Hazard Tracker FIFO entry) to the instruction WB-stage. A valid NaN propagation will cause the operation result from the Execute stage to be ignored, and the propagated NaN value to be written into the result destination instead, as discussed in NaN Propagation.

3.6.8.9.3 Rounding Modes

The rounding mode for each instruction Functional Block is defined by the value written into FCR.RND [1:0] as defined in 3.6.5. FCR. The FPU treats the rounding mode input as an operand supplied from the RD-stage when the instruction is dispatched into the Execute stage.

Note: Rounding Modes are not applicable to `FAND`, `FIOR`, `FCPQ`, `FCPS`, `FTST`, `FABS`, `FNEG`, `FFLIM`, `FMAX`, `FMIN`, `FMAXNUM`, `FMINNUM`, `FMOV`, `FMOVC` or any CPU to/from FPU data move instruction.

There is a 3-bit rounding mode input (rnd [2:0]) to support up to eight different rounding modes for all FPU conversion operations. Setting rnd [2] = 1 and mapping rnd [1:0] to FCR [9:8] will allow user selection of the IEEE 754 compliant modes as defined in 3.6.5. FCR.

The integer/floating-point conversion instructions (FDI2F, FLI2F, FF2DI, FF2LI) may either specify the rounding mode within the instruction syntax or default to that defined in FCR.RND [1:0]. CPU will issue one of these instructions and the FPU will use it to determine the Functional Block Rounding mode as shown in Table 3-16.

Table 3-16. FPU Conversion OP Rounding Modes Control

Rounding Mode Bits in Opcode[2:0]	Functional Block Rounding Mode
111	IEEE Round to Negative Infinity (floor)
110	IEEE Round to Positive Infinity (ceiling)
101	IEEE Round to Zero (truncate)
100	IEEE Round to Nearest (even)
0xx	Global mode (defined by FCR.RND[1:0])

3.6.8.10 Floating Point Status

The FPU generates four types of status:

- Exception condition “most-recent” status from most instructions (see Table 3-19). These bits are located within FSR [6:0]:INX, HUGI, OVF, UDF, DIV0, INVAL, SUBO.
- Exception condition "sticky" status from most instructions (see Table 3-19). These bits are located within FSR [14:8]: INXS, HUGIS, OVFS, UDFS, DIV0S, INVALS, SUBOS.
- Value ordering relations status to indicate the result of the FCPS/FCPQ compare instructions. These bits are located within FSR [19:16]:GT, LT, EQ, UN.
- Operand characteristic status from the FTST datum inspection/classify instruction. These bits are located within FSR [28:24]: SUB, INF, FZ, FN, FNAN.

Operand comparisons are likely to be used frequently, so the compare status bits generated by the FCPS/FCPQ instructions are supported with CPU conditional branch instructions. All other status must be read into the CPU (using the MOVCRW instruction) or pushed onto the stack (using PUSHCR) and then acted upon as necessary.

Note: Irrespective of whether an exception is masked or not, writing a logic 1 to an exception status flag using any instruction that can write 1’s to the FSR will not result in any associated exception being taken.

3.6.8.10.1 Compare Status and Predicates

IEEE 754-2008/2019 standards specify Quiet and Signaling Compare Predicates (equations) as shown in Table 3-17. A “signaling” predicate signals (i.e., attempts to generate an exception) when a Quiet NaN or Signaling NaN (qNaN or sNaN) operand is detected.

A “quiet” predicate will not signal when a qNaN operand is detected.

An sNaN will always signal an exception when detected as an operand for all instructions except those that do not generate any exceptions (FMOV, FMOVC, FABS, FNEG and FTST).

The FPU coprocessor macro implements Signaling and Quiet predicates by supporting two floating-point compare options, one signaling (FCPS), one quiet FCPQ), and a set of floating-point branch operations that test for the required predicates. Each compare instruction will set one of the four mutually exclusive ordering relations (GT, LT, EQ, UN status bits) located in the FSR to indicate the result of the comparison.

- FCPS (signaling compare):
  - qNaN or sNaN: If either or both operands are a qNaN or sNaN value, the compare is considered unordered which will cause the FSR.UN bit to be set. In addition, the FSR.INVAL bit will be set, causing the CPU to be signaled via the Invalid exception (assuming that the exception is not masked).
- FCPQ (quiet compare):

- qNaN: If one or more operands contain a qNaN value, the compare is considered unordered which will cause the FSR.UN bit to be set. A qNaN will not set the FSR.INVAL bit, so no signaling will occur.
- sNaN: If either or both operands are a sNaN value, the compare is considered unordered which will cause the FSR.UN bit to be set. In addition, the FSR.INVAL bit will be set, causing the CPU to be signaled via the Invalid exception (assuming that the exception is not masked).

The compare operation subtracts Fs (subtrahend) from Fb (minuend). The EQ, GT and LT status bits are set as follows:

- If the minuend is equal to the subtrahend (Fb = Fs) the EQ status bit is set.
- If the minuend is greater than the subtrahend (Fb > Fs) the GT status bit is set.
- If the minuend is less than the subtrahend (Fb < Fs) the LT status bit is set.

In addition, the UN status bit is set if one or both operands is a NaN. If this is the case, no other compare status is set (i.e., UN and EQ, GT, LT are mutually exclusive).

Note: The FCPS/FCPQ instructions consider -0 and +0 as equivalent.

Note: Comparing a value to itself should produce an equivalence result. However, UN has precedence over EQ such that, should two values be identical but both NaN, the UN bit will be set but the EQ bit will be cleared.

FPU Status Conditional Branches

The CPU has the ability to conditionally branch off various status bits generated within the coprocessor. In the case of the FPU, an internal status register (FSR) is supported which is updated at the end of each floating-point operation.

The FPU FSR is comprised of instruction exception status and FCPS/FCPQ/FTST instruction status. Conditional branching is supported within the CPU for the FCPS/FCPQ compare instructions only.

The CPU ISA includes a set of generic coprocessor conditional branch instructions, CBRA0 through CBRA15, each of which can operate with any instantiated coprocessor and branch based upon the state of a corresponding bit within a vector supplied by each coprocessor. In the case of the FPU, CBRA0 through CBRA13 are used, each represented as an FBRA instruction with its corresponding assembler attribute, for the FCPS/FCPQ instruction status branch conditions. The FCPS/FCPQ status is held in FSR [19:16] and indicates the comparison result. CBRA[n] timing is the same as any other CPU conditional branch, such that the condition is examined at the end of the CBRA[n] R-stage. If the condition is true, the branch is taken. If the condition is not true, the branch is not taken and sequential execution continues.

As is the case for all conditional branches, the instruction(s) immediately following the branch are speculatively executed, and they will either be part of the taken or the not taken path, based on the direction of the branch. These instructions are permitted to be floating-point operations. This requires that the FPU accommodate the possibility that these instructions could ultimately be killed due to a branch mispredict.

Note that the FPU will not return the result of FBRA instruction until any FCPS/FCPQ instruction already underway in the coprocessor pipeline has retired. The CPU will consequently stall until such time that the msw of the FSR is available to be read (though these are fast operations, so stalls should be minimal). In effect, a CPU conditional branch instruction operation will synchronize the integer and floating-point pipelines with respect to FPU FCPS/FCPQ status.

The LS 3-bits of the branch opcode concatenated with the sub-opcode bit (such that the sub-opcode bit becomes the LSb of this value), may be used by the CPU decoder as a bit pointer into the 16-bit branch status test value to select the corresponding branch predicate result. The branch then decides if the outcome is true (taken) or false (not taken) based on the state of the selected bit (where true is when the bit is set, false when clear).

Note: FCPS/FCPQ and FTST instructions update two different portions of the FSR. Consequently, execution of an FTST instruction (which also updates the FSR) will not inhibit the CPU CBRAn instructions from using the branch status generated from the FSR ordering relation bits.

The FTST instruction will test the operand and update the SUB, INF, FN, FZ, FNAN status bits. No exceptions will be generated by this instruction. Due to the relative infrequent use of this instruction, dedicated conditional branches are not supported by the CPU to test these status bits. The user must read the FSR and then act upon the bits of instruction using existing CPU instructions.

3.6.3.10.2 Operand Characteristic (Test) Status

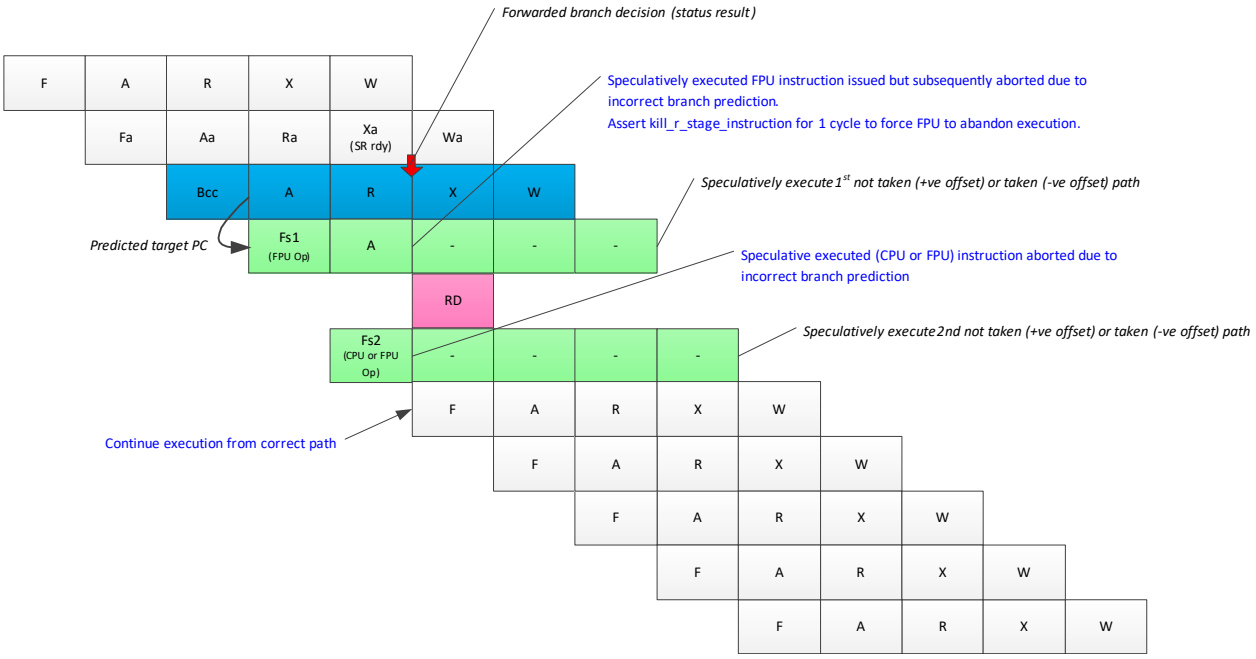
Table 3-17. Floating-Point Conditional Branches and Associated Predicates

Assembler FBRA Attribute	Design Mnemonic CBRA Mapping	Predicates					Assembler FBRA Attribute	Design Mnemonic CBRA Mapping	Negated Predicates				
		Ordering Relation				Definition[1] (Alternative Definition)			Ordering Relation				Definition[2] (Alternative Definition)
		>	<	=	?				>	<	=	?	
EQ	CBRA0	F	F	T	F	Equal	UNE	CBRA1	T	T	F	T	Unordered or Greater Than or Less Than (Unordered or Not Equal)
NE	CBRA2	T	T	F	F	Greater Than or Less Than (Not Equal)	UEQ[3]	CBRA3	F	F	T	T	Unordered or Equal
GT	CBRA4	T	F	F	F	Greater Than	ULE	CBRA5	F	T	T	T	Unordered or Less Than or Equal
GE	CBRA6	T	F	T	F	Greater Than or Equal	ULT	CBRA7	F	T	F	T	Unordered or Less Than
LT	CBRA8	F	T	F	F	Less Than	UGE	CBRA9	T	F	T	T	Unordered or Greater Than or Equal
LE	CBRA10	F	T	T	F	Less Than or Equal	UGT	CBRA11	T	F	F	T	Unordered or Greater Than
OR	CBRA12	T	T	T	F	Ordered	UN	CBRA13	F	F	F	T	Unordered

3.6.8.10.3 FPU Instruction Kill

As is the case for all instructions executed within conditional branch speculative slots, floating-point instructions will be killed if a branch mispredict occurs. The CPU will recognize a mispredict prior to the end of the conditional branch R-stage (i.e., when the prior instruction status is available to forward). If the instruction in the first speculative slot is an FPU instruction, it will be issued to the FPU, but the CPU will assert a signal to kill the instruction for one cycle, forcing the FPU to subsequently abandon execution prior to it being committed. If the instruction in the second speculative slot is an FPU instruction, it will be abandoned prior to being issued to the FPU.

Figure 3-33. FPU Instruction Speculative Execution



3.6.8.11 Generating FP Exceptions

The FPU can generate the five IEEE-754 2008 compliant exceptions. Each exception has a flag associated with it in the FSR register as noted below.

All instructions, except `FTST`, `FMOV` and `FMOVC`, can affect FSR exception status as a consequence of the operation (any exception status flag bits not set will be cleared). The `FTST`, `FMOV` and `FMOVC` instructions will neither set nor clear any exception flags. Refer to the IEEE-754 2008 standard for a more detailed definition.

"Most-recent" exception status only shows the exception status of the most recent instruction executed. It contains no accrued status from prior operations such that if a status bit is not affected or not signaled by the instruction, it will be cleared.

"Sticky" exception status contains accrued status from all instructions executed.

The following summarizes when exceptions are signaled, and the default results for each exception are summarized in Table 3-15. Refer to Table 3-18 for exception conditions and default results for each instruction.

- Invalid: FSR.INVAL Exception signaled whenever an operation generates no usefully definable result. INVAL is set under the following conditions:
  - Any operation on a sNaN input
  - Addition of infinities with opposite signs, or subtraction of infinities with the same sign
  - Multiplication 0*infinity
  - Division 0/0 and infinity/infinity
  - Square root of a negative floating-point value

Note: INVAL is also set by FF2DI and FF2LI Float-to-Integer conversion instructions in the event of a qNaN or Infinity input value.

- Divide by Zero: FSR.DIV0 Exception signaled whenever the FDIV instruction dividend is finite and the divisor is 0.
- Overflow: FSR.OVF Exception signaled whenever an operation results in an overflow, defined as a post-rounded result that exceeds the largest finite number that the destination can represent.
- Underflow: FSR.UDF Exception signaled whenever an operation results in a tiny but non-zero (subnormal) result (see Underflow with an Exact Rounded Result for a special case).
- Inexact: FSR.INX Exception signaled whenever the rounded result of an operation is:
  - Not equal to the same result represented using infinite precision (i.e., has suffered a loss of accuracy)
  - Is subnormal and not an exact zero when FTZ mode is enabled

In addition, this macro also generates two additional exceptions:

- Huge Integer: FSR.HUGI Exception signaled whenever a Float-to-Integer conversion operation (FF2DI and FF2LI) results in an integer value that is larger than the destination register can represent.
- Subnormal Operand: FSR.SUBO Exception signaled whenever an operand of an affected instruction (see Subnormal Override Functions and Table 3-18) is a subnormal value, and Subnormals-Are-Zeros (SAZ) mode is disabled.

3.6.8.11.1 Exception Generation Special Cases

Concurrent Exceptions

The following combinations of exceptions can occur together:

- Huge Integer will always also signal Invalid and Inexact

- Overflow will always also signal Inexact
- Underflow may also signal Inexact

No other exceptions can occur concurrently.

Notes:

1. Inexact can be asserted independently of any other exception.
2. If Overflow, Underflow and Inexact exceptions are all enabled, the exception handler should prioritize Overflow and Underflow above Inexact.

During FF2DI and FF2LI Float-to-Integer conversion instructions, should the HUGI exception be signaled (due to a finite float-point value conversion that results in a value that exceeds the size of the destination register), the INVAL exception will also be signaled. Users may therefore choose to ignore the (not IEEE-754 compliant) HUGI exception.

Underflow with an Exact Rounded Result

An exact subnormal result is not viewed as an Underflow condition by the IEEE 754-2008/2019 standard when operating with default exception handling. Consequently, FPU exceptions operate differently when the Underflow exception is disabled (enabling default exception handling for Underflow).

As summarized below, when operating with default exception handling (Underflow exception masked, FCR.UDFM = 1), the FPU will only signal Underflow (i.e., set FSR.UDF = 1) if the rounded subnormal result suffers from a loss of accuracy (such that the Inexact status will also be set). Otherwise, no status is affected. In both cases, the rounded (default) result will be delivered.

Note: If FTZ mode is active (FCR.FTZ && FCR.UDFM = 1), Inexact is signaled whenever a result is subnormal and not an exact zero.

When operating with alternate exception handling (Underflow exception is unmasked), Underflow will be signaled whenever a subnormal result is detected irrespective of whether it is exact or not. If the result also suffers from any loss of accuracy, Inexact will also be set.

Note: A zero result is not considered an Underflow condition, so will not signal Underflow irrespective of exception handling mode.

- Default exceptions: UDF interrupt is masked (UDFM = 1)
  - If an inexact Underflow occurs, both UDF and INX (and sticky equivalents) are set.
  - If an exact Underflow occurs, no status is set.
  - In both cases, default (rounded) result is delivered, and no interrupts are generated.
- Alternate exceptions: UDF interrupt is unmasked (UDFM = 0)
  - If any Underflow occurs, UDF (and sticky equivalent) is always set and an interrupt occurs. INX (and sticky equivalent) will also be set if it is inexact; it is cleared otherwise.
  - Default (rounded) result is delivered and interrupt is generated

Invalid for a qNaN Input Operand

A qNaN input operand does not typically signal the Invalid exception other than when encountered by the following instructions:

- FF2DI and FF2LI Float-to-Integer conversion instructions (because a qNaN cannot, of course, be represented as an integer value).
- FCPS instruction to signal any NaN input.

For both sNaN and qNaN operands, the FF2DI and FF2LI instructions will deliver the integer indefinite value (most negative number) as the result, and INVAL (but not HUGI) will be signaled.

No result is delivered for the FCPS instruction (other than FCPS/FCPQ status).

3.6.8.11.2 FP Exception Sticky Flags

The INVAL, DIV0, OVF, UDF, INX, HUGI and SUBO exception flags have corresponding “sticky” flags INVALS, DIV0S, OVFS, UDFS, INXS, HUGIS and SUBOS also located within the FSR. These bits are set whenever the corresponding exception flag is set. They will remain set if the exception flag is cleared by a subsequent instruction; therefore, they are considered “sticky.” These bits support the delayed exception handling model required by the IEEE 754-2008 standard and are analogous to conventional interrupt flags.

3.6.8.11.3 Modifying Exception Status Through Software

Software may simultaneously clear both “most-recent” and “sticky” exception flags using the `FAND` instruction. Software may also set both “most-recent” and “sticky” exception flags simultaneously using the `FIOR` instruction, though this operation is of little utility.

Note: Clearing (or setting) a most-recent exception status bit by using the `FAND` (or `FIOR`) instruction will not affect any corresponding sticky exception bits.

3.6.8.11.4 FP Exception Masks

FSR bits INVAL, DIV0, OVF, UDF, INX, HUGI and SUBO have corresponding INVALM, DIV0M, OVFM, UDFM, INXM, HUGIM and SUBOM Exception Mask bits in the FCR. Should a “most-recent” exception flag be set by an operation, the corresponding Exception Mask bit in the FCR must already be clear in order to generate an interrupt.

Should an FCR Exception Mask bit be set when the corresponding FSR exception flag bit is set, no interrupt will be generated. In all cases (except SUBO which is an input operand exception), a default result (see Table 3-15) will be written. However, the corresponding “sticky” exception flag will be set and remain set until manually cleared.

Note: If not masked, interrupts are generated at the time the corresponding flag is set (i.e., they are generated by the leading edge of the flag set operation). However, setting a flag (in FSR [6:0]) when the corresponding exception is enabled will not generate an interrupt. Should forcing an interrupt be required (e.g., during debug), it may be achieved by setting the FPU interrupt flag (in the Interrupt Controller). Conversely, unmasking a previously masked exception when its flag is already set will not generate an interrupt.

Table 3-18. FP Instruction Exception Conditions

Instruction	Exceptions (FSR[6:0])							Conditions	Default Results and Notes
	SUBO	HUGI	INX	UDF	OVF	DIV0	INVAL		
FMOV	—	—	—	—	—	—	—	—	—
FMOVC	—	—	—	—	—	—	—	—	—
Status Bit Set/Clear/Update Instructions									
FAND	⇓	⇓	⇓	⇓	⇓	⇓	⇓	Logical AND with FSR[15:0]	—
FIOR	⇑	⇑	⇑	⇑	⇑	⇑	⇑	Logical OR with FSR[15:0]	No exceptions are generated as a consequence of setting an exception flag
FTST	—	—	—	—	—	—	—	—	No result delivered other than FTST status (FSR[28:24])
Conversion Instructions									
FLI2F	0	0	⇳	0	0	0	0	INX: See 3.6.8.11. Generating FP Exceptions	INX: Destination will be written with inexact floating-point result.
FDI2F	0	0	⇳	0	0	0	0	INX: See 3.6.8.11. Generating FP Exceptions	INX: Destination will be written with inexact floating-point result.

Key: ⇳ = set or cleared, ‘0’ = always cleared, — = unchanged, ⇑ = may be cleared but never set, ⇓ = may be set but never cleared

Notes:

1. In all cases where INVAL is signaled as the result of an sNaN operand, the destination will be written with the quieted sNaN.
2. Underflow result is defined in Table 3-15 when SOV mode enabled and Underflow exception is not enabled.
3. The Subnormal Operand exception (SUBO) is an input operand exception (all other exceptions are related to operation results). SUBO never signaled if SAZ mode enabled.

.....continued

Instruction	Exceptions (FSR[6:0])							Conditions	Default Results and Notes
	SUBO	HUGI	INX	UDF	OVF	DIV0	INVAL		
FF2LI	⇳	⇳	⇳	0	0	0	⇳	HUGI: Result exceeds target register size INX: See 3.6.8.11. Generating FP Exceptions INVAL: ∞, sNaN or qNaN SUBO: Subnormal operand(3)	HUGI: Destination will be written with either most positive or most negative integer, matching sign of input operand. HUGI will also cause INVAL to be set. INX: Destination will be written with inexact integer result. INVAL: Destination will be written with integer indefinite value if HUGI is not set, or value defined above if HUGI is set. Assertion of SUBO will not affect integer result.
FF2DI	⇳	⇳	⇳	0	0	0	⇳	HUGI: Result exceeds target register size INX: See 3.6.8.11. Generating FP Exceptions INVAL: ∞, sNaN or qNaN SUBO: Subnormal operand(3)	HUGI: Destination will be written with either most positive or most negative integer, matching sign of input operand. HUGI will also cause INVAL to be set. **Note:** INX: Destination will be written with inexact integer result. INVAL: Destination will be written with integer indefinite value if HUGI is not set, or value defined above if HUGI is set. Assertion of SUBO will not affect integer result.
Comparison Instructions									
FCPS	⇳	0	0	0	0	0	⇳	INVAL: sNaN or qNaN SUBO: Subnormal operand(3)	No result delivered other than FCPS/FCPQ status (FSR[19:16])
FCPQ	⇳	0	0	0	0	0	⇳	INVAL: sNaN SUBO: Subnormal operand(3)	No result delivered other than FCPS/FCPQ status (GT, LT, EQ or UN)
FFLIM	⇳	0	0	0	0	0	⇳	INVAL: sNaN SUBO: Subnormal operand(3)	Refer to Table 3-12 for result delivered. Operand value of -0 compares to less than +0.
FMAX	⇳	0	0	0	0	0	⇳	INVAL: sNaN SUBO: Subnormal operand(3)	Refer to Table 3-11 for result delivered. Operand value of -0 compares to less than +0.
FMAXNUM	⇳	0	0	0	0	0	⇳	INVAL: sNaN SUBO: Subnormal operand(3)	Refer to Table 3-11 for result delivered. Operand value of -0 compares to less than +0.
FMIN	⇳	0	0	0	0	0	⇳	INVAL: sNaN SUBO: Subnormal operand(3)	Refer to Table 3-11 for result delivered. Operand value of -0 compares to less than +0.
FMINNUM	⇳	0	0	0	0	0	⇳	INVAL: sNaN SUBO: Subnormal operand(3)	Refer to Table 3-11 for result delivered. Operand value of -0 compares to less than +0.
Math Instructions									

Key: ⇳ = set or cleared, '0' = always cleared, — = unchanged, ⇑ = may be cleared but never set, ⇓ = may be set but never cleared

Notes:

1. In all cases where INVAL is signaled as the result of an sNaN operand, the destination will be written with the quieted sNaN.
2. Underflow result is defined in Table 3-15 when SOV mode enabled and Underflow exception is not enabled.
3. The Subnormal Operand exception (SUBO) is an input operand exception (all other exceptions are related to operation results). SUBO never signaled if SAZ mode enabled.

.....continued

Instruction	Exceptions (FSR[6:0])							Conditions	Default Results and Notes
	SUBO	HUGI	INX	UDF	OVF	DIV0	INVAL		
FADD	⇳	0	⇳	⇳	⇳	0	⇳	INX: See 3.6.8.11. Generating FP Exceptions UDF: See 3.6.8.11. Generating FP Exceptions OVF: See 3.6.8.11. Generating FP Exceptions INVAL: ((-∞) + ∞) or sNaN SUBO: Subnormal operand(3)	INX: Will also be set if OVF is set. Will also be set if UDF is set and result is not an exact subnormal. INX/UDF/OVF: Destination will be written with rounded result. 2 **Note:** INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.1
FSUB	⇳	0	⇳	⇳	⇳	0	⇳	INX: See 3.6.8.11. Generating FP Exceptions UDF: See 3.6.8.11. Generating FP Exceptions OVF: See 3.6.8.11. Generating FP Exceptions INVAL: (∞ - ∞) or sNaN SUBO: Subnormal operand(3)	INX: Will also be set if OVF is set. Will also be set if UDF is set and result is not an exact subnormal. INX/UDF/OVF: Destination will be written with rounded result. 2 INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.
FNEG	⇳	—	—	—	—	—	—	SUBO: Subnormal operand(3)	IEEE-754 requires no result exceptions be signaled. Subnormal operand exception signaled or SAZ mode applied to ensure result consistency with that of an equivalent sequence of FPU arithmetic instructions.
FABS	⇳	—	—	—	—	—	—	SUBO: Subnormal operand(3)	IEEE-754 requires no result exceptions be signaled. Subnormal operand exception signaled (or SAZ mode applied) to ensure result consistency with that of an equivalent sequence of FPU arithmetic instructions.
FMUL	⇳	0	⇳	⇳	⇳	0	⇳	INX: See 3.6.8.11. Generating FP Exceptions UDF: See 3.6.8.11. Generating FP Exceptions OVF: See 3.6.8.11. Generating FP Exceptions INVAL: (0 * ∞) or (∞ * 0) or sNaN SUBO: Subnormal operand(3)	INX: Will also be set if OVF is set. Will also be set if UDF is set and result is not an exact subnormal. INX/UDF/OVF: Destination will be written with rounded result. 2 INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.
FMAC	⇳	0	⇳	⇳	⇳	0	⇳	INX: See 3.6.8.11. Generating FP Exceptions UDF: See 3.6.8.11. Generating FP Exceptions OVF: See 3.6.8.11. Generating FP Exceptions INVAL: (0 * ∞)+c or (∞ * 0)+c or sNaN SUBO: Subnormal operand(3)	INX: Will also be set if OVF is set. Will also be set if UDF is set and result is not an exact subnormal. INX/UDF/OVF: Destination will be written with rounded result. 2 INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.1 INVAL will also be signaled if accumulation is a subtraction of infinities.

Key: ⇳ = set or cleared, '0' = always cleared, — = unchanged, ⇑ = may be cleared but never set, ⇓ = may be set but never cleared

Notes:

1. In all cases where INVAL is signaled as the result of an sNaN operand, the destination will be written with the quieted sNaN.
2. Underflow result is defined in Table 3-15 when SOV mode enabled and Underflow exception is not enabled.
3. The Subnormal Operand exception (SUBO) is an input operand exception (all other exceptions are related to operation results). SUBO never signaled if SAZ mode enabled.

.....continued

Instruction	Exceptions (FSR[6:0])							Conditions	Default Results and Notes
	SUBO	HUGI	INX	UDF	OVF	DIV0	INVAL		
FDIV	⇳	0	⇳	⇳	⇳	⇳	⇳	INX: See 3.6.8.11. Generating FP Exceptions UDF: See 3.6.8.11. Generating FP Exceptions OVF: See 3.6.8.11. Generating FP Exceptions DIV0: Finite Dividend with Divisor = 0 INVAL: (0/0) or (∞/∞) or sNaN SUBO: Subnormal operand(3)	INX: Will also be set if OVF is set. Will also be set if UDF is set and result is not an exact subnormal. INX/UDF/OVF: Destination will be written with rounded result. 2 DIV0: Result of (±x/±0) will be ±∞ by default, where sign is XOR of operand signs INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.1 (0/0) or (∞/0) are special cases and will not set DIV0. Result of (∞/0) is correctly signed infinity. Result of (0/0) is the distinguished qNaN with INVAL signaled.
FSQRT	⇳	0	⇳	0	0	0	⇳	INVAL: x < 0 or sNaN SUBO: Subnormal operand(3)	INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.(1) FSQRT(±0) = ±0 (no exception signaled)
FSIN	⇳	0	⇳	⇳	0	0	⇳	INX:See 3.6.8.11. Generating FP Exceptions UDF:See 3.6.8.11. Generating FP Exceptions INVAL: | x | = ∞ or sNaN SUBO: Subnormal operand(3)	INX: Will also be set if UDF is set and result is not an exact subnormal. INX/UDF: Destination will be written with rounded result. (2) INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.(1)
FCOS	⇳	0	⇳	⇳	0	0	⇳	INVAL: | x | = ∞ or sNaN SUBO: Subnormal operand(3)	INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.(1)

Key: ⇳ = set or cleared, '0' = always cleared, — = unchanged, ⇑ = may be cleared but never set, ⇓ = may be set but never cleared

Notes:

1. In all cases where INVAL is signaled as the result of an sNaN operand, the destination will be written with the quieted sNaN.
2. Underflow result is defined in Table 3-15 when SOV mode enabled and Underflow exception is not enabled.
3. The Subnormal Operand exception (SUBO) is an input operand exception (all other exceptions are related to operation results). SUBO never signaled if SAZ mode enabled.

3.6.8.12 FP Instruction Status Effects

Table 3-19 lists the floating-point instructions and their effect on the FPU Status Bits.

- FSR [19:16] can only be updated by the `FCPS`/`FCPQ` instructions.
- FSR [28:24] can only be updated by the `FTST` instruction.
- `FMOV`, `FMOVC`, `FABS` and `FNEG` instructions do not generate any status.

3.6.8.12.1 FP Status Access

The msws and lsws of the FSR are implemented as 16-bit registers which can be independently read and written by the CPU and FPU. This is intended to prevent structural hazards arising between `FTST`/`FCPS`/`FCPQ`/`FBcc` instructions (that access the FSR msw), and all other instructions that could affect the exception status (located in the FSR lsw). Hazards are not expected to exist between `FTST` and `FCPS`/`FCPQ` instructions.

The FCPS/FCPQ/FTST status located in msw of the FSR (FSRH) can be read and stacked by the `PUSHCR` instruction while the lsw of the FSR is written (or blocked to be written). The msws and lsws of the FSR are otherwise concatenated into a single 32-bit value (FSR) for read/write by the `MOVCRW`/`MOVWCR` and `PUSHCR`/`POPCR` instructions.

In addition, the FAND and FIOR instructions operate with a 16-bit literal and can be used to set or clear the bit in the FCR or lsw of the FSR or FEAR. When the FSR is referenced in these ops, users can manipulate the FPU exception status, but access to the FCPS/FCPQ/FTST status in FSRH is not possible. This choice assumes that the need to modify FTST/FCPS/FCPQ status is rare. Consequently, doing so using MOVCRW/MOVWCR or PUSHCR/POPCR is expected to be adequate.

Table 3-19. FP Instruction Status Operations

Instruction	FPU Status Register (FSR)																						
	FSR[28:24]					FSR[19:16]				FSR[14:8]							FSR[6:0]						
	SUB	INF	FN	FZ	FNAN	GT	LT	EQ	UN	SUBOS	HUGIS	INXS	UDFS	OVFS	DIVOS	INVALS							
Move Instructions																							
FMOV	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
FMOVC	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Status Bit Set/Clear/Update Instructions																							
FAND (1)	—	—	—	—	—	—	—	—	—	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓
FIOR (1)	—	—	—	—	—	—	—	—	—	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑
FTST	⇕	⇕	⇕	⇕	⇕	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Conversion Instructions																							
FLI2F	—	—	—	—	—	—	—	—	—	—	—	⇑(2)	—	—	—	—	0	0	⇕(2)	0	0	0	0
FDI2F	—	—	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	0	0	⇕	0	0	0	0
FF2LI	—	—	—	—	—	—	—	—	—	⇑	⇑	⇑	—	—	—	⇑	⇕	⇕	⇕	0	0	0	⇕
FF2DI	—	—	—	—	—	—	—	—	—	⇑	⇑	⇑	—	—	—	⇑	⇕	⇕	⇕	0	0	0	⇕
Comparison Instructions																							
FCPS	—	—	—	—	—	⇕	⇕	⇕	⇕	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
FCPQ	—	—	—	—	—	⇕	⇕	⇕	⇕	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
FFLIM	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
FMAX	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
FMAXNUM	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
FMIN	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
FMINNUM	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
Math Instructions																							
FADD	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	⇑	—	⇑	⇕	0	⇕	⇕	⇕	0	⇕
FSUB	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	⇑	—	⇑	⇕	0	⇕	⇕	⇕	0	⇕
FNEG	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	—	⇕	—	—	—	—	—	—
FABS	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	—	⇕	—	—	—	—	—	—
FMUL	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	⇑	—	⇑	⇕	0	⇕	⇕	⇕	0	⇕
FMAC	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	⇑	—	⇑	⇕	0	⇕	⇕	⇕	0	⇕
FDIV	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	⇑	⇑	⇑	⇕	0	⇕	⇕	⇕	⇕	⇕
FSQRT	—	—	—	—	—	—	—	—	—	⇑	—	⇑	—	—	—	⇑	⇕	0	⇕	0	0	0	⇕

Key: ⇕ set or cleared, '0' always cleared, — unchanged, ⇓ may be cleared but never set, ⇑ may be set but never cleared

Notes:

1. With respect to the FSR, FAND and FIOR can only affect FSR[15:0] (exception status). When set, no exception is signaled.
2. LI2F.s only. Inexact not possible for Long integer to Double Precision float (LI2F.d) conversion.

.....continued

Instruction	FPU Status Register (FSR)																						
	FSR[28:24]					FSR[19:16]				FSR[14:8]							FSR[6:0]						
	SUB	INF	FN	FZ	FNAN	GT	LT	EQ	UN	SUBOS	HUGIS	INXS	UDFS	OVFS	DIVOS	INVALS							
FSIN	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	—	—	⇑	⇳	0	⇳	⇳	0	0	⇳
FCOS	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	—	—	⇑	⇳	0	⇳	⇳	0	0	⇳

Key: ⇳ set or cleared, '0' always cleared, — unchanged, ⇓ may be cleared but never set, ⇑ may be set but never cleared

Notes:

1. With respect to the FSR, FAND and FIOR can only affect FSR[15:0] (exception status). When set, no exception is signaled.
2. LI2F.s only. Inexact not possible for Long integer to Double Precision float (LI2F.d) conversion.

4. Memory Organization

This section describes the Memory Organization and Bus Matrix (BMX) in dsPIC33A devices. The following features are covered:

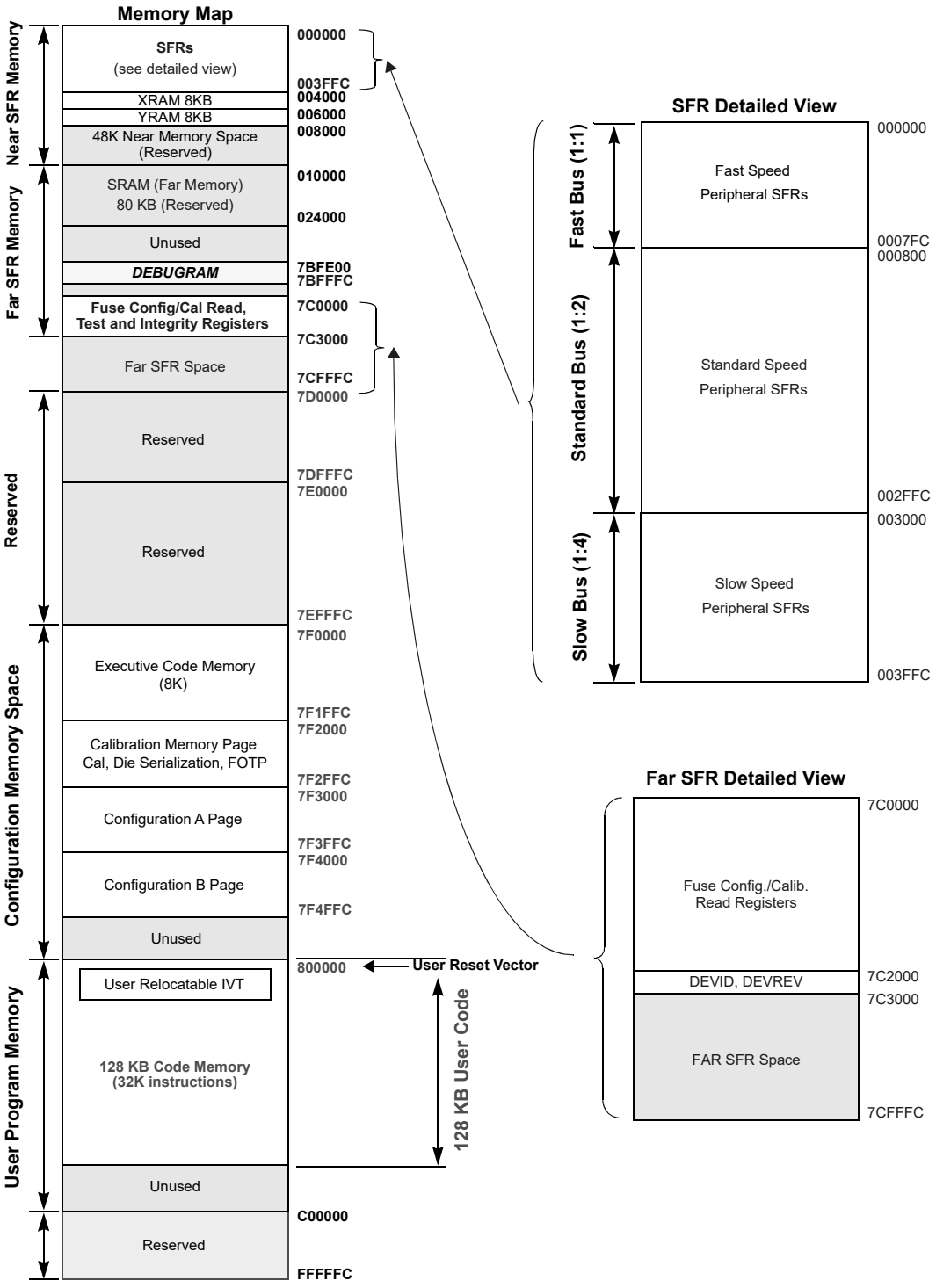
- Memory and SFR Maps
- Modified Harvard Architecture
- Unified Memory Map
- Split Data Bus Speeds
- Bus Matrix (BMX):
  - Establishes communication between initiator and targets
  - Decodes addresses and provides arbitration between multiple initiators requesting access to the same target
  - Provides concurrent accesses to multiple targets from different initiators
  - Generates a bus error exception back to an initiator on any failed access
  - Provides support for the CPU to execute from RAM

4.1 Device-Specific Information

4.1.1 Address Space

The memory map for dsPIC33AK128MC106 devices is shown in Figure 4-1.

Figure 4-1. Memory Map for dsPIC33AK128MC106



Notes:

1. Memory areas are not shown to scale.
2. Calibration data area includes UDID and ICSP™ Write Inhibit locations.

4.1.2 Multiple Speed Peripheral Bus and Clock Overview

The dsPIC33AK128MC106 family of devices implements a multiple speed peripheral bus. Infrequent access and initialization SFR are on the slow bus, whereas critical control loop type SFR are run at full speed relative to the CPU.

Each peripheral bus speed includes an associated peripheral clock. The peripheral bus SFR access and peripheral clock speed are equal, and are arranged in fast (1:1), standard (1/2) and slow speeds (1/4). The fast peripheral clock is equal to the CPU or system clock provided by CLKGEN1. The other two speeds are generated by the peripheral clock divider discussed in 12.2.1. Peripheral Clock Divider.

Table 4-1 maps peripherals to their associated peripheral buses and clock speeds.

Table 4-1. Peripheral Bus Speed to Peripheral Mapping

Fast Speed Bus Peripherals	Standard Speed Bus Peripherals	Slow Speed Bus Peripherals
CPU	PWM	Flash NVM
Interrupt Controller	UART	Clock monitors
GPIO(1)	SPI	WDT
CRC	I2C	PTG
Security Module	SENT	ECC
Bus Matrix control	QEI	GPIO, PPS(2)
ADC	CCP	DMT
	DAC and CMP	PMD
	Timer 1	CLC
	Performance monitor	Op amps
	Peripheral access controller	
	IO Integrity monitor	
	BISS	
	DMA	

Notes:

1. Includes only PORT, LAT, TRIS and change notification registers.
2. Includes remainder of GPIO registers.

4.1.2.1 Unique Device Identifier (UDID)

All dsPIC33AK128MC106 family devices are individually encoded during final manufacturing with a Unique Device Identifier or UDID. The UDID cannot be erased by a bulk erase command or any other user-accessible means. This feature allows for manufacturing traceability of Microchip Technology devices in applications where this is a requirement. It may also be used by the application manufacturer for any number of things that may require unique identification, such as:

- Tracking the device
- Unique identifying number
- Unique security key

The UDID comprises four full 32-bit words. When taken together, these fields form a unique 128-bit identifier.

The UDID is stored in read-only locations, located between 0x7F2120-0x7F212C in the device configuration space. Table 4-2 lists the addresses of the identifier words and shows their contents.

Table 4-2. UDID Address

Address	Description
0x7F2120	UDID Word 1
0x7F2124	UDID Word 2
0x7F2128	UDID Word 3
0x7F212C	UDID Word 4

4.2 Architectural Overview

4.2.1 Program Memory Organization

Program memory addresses are always word-aligned on the lower word, and addresses are incremented or decremented by two during code execution. This arrangement provides compatibility with data memory space addressing and makes data in the program memory space accessible.

4.2.2 Interrupt and Trap Vectors

All dsPIC33AK128MC106 family devices have the user program space starting at 0x800000. The interrupt vector table is placed (by default) at the start of user program Flash memory (0x800004). The user code can then relocate the IVT to any valid address in Flash/RAM by modifying the IVTBASE register. The reset vector into user program space is at 0x800000. When a Reset is asserted, the reset vector read commences and the vector contents are presented to the CPU. The data is then immediately transferred to the program space address bus to create the address of the first instruction to be executed, redirecting the program execution to the appropriate start-up routine.

4.2.3 Data Memory Organization

SFR space and data RAM are mapped starting at address 000000 to allow near memory access from instructions that can encode a memory address. The near address range for most file register instructions is 64KB total. The far address space is any address beyond the first 64KB. The SFR space is 16 KB and is separated into different clock speeds via a peripheral bus splitter:

- The address range 000000 - 0007FF is associated with a fast (1:1 System Clock) peripheral bus. A minimum of SFRs are mapped in this region.
- The address range 000800 - 002FFF is associated with the standard speed (1:2) peripheral bus. The majority of the peripheral SFRs are mapped in this region.
- The address range 003000 - 003FFF is associated with the slow speed (1:4) peripheral bus. Peripheral SFRs that need infrequent access are mapped in this region.
- The address range 7C0000 - 7CFFFF is associated with the slow speed (1:4) peripheral bus dedicated to the calibration and configuration registers in far memory space.

4.2.4 Bus Matrix

The Bus Matrix (BMX) arbitrates memory accesses in the event two independent initiators are trying to access the same targets. Initiators are a set of modules that can initiate a read or write transaction to other modules called targets. The BMX connects the initiator to targets and determines which initiator gets priority (based on fixed priority scheme and the SFR priority). The type of access, program or data, is determined by the initiator bus. It supports concurrent accesses by different initiators as long as they have independent targets. For example, CPU to SFR could be concurrent with DMA to RAM.

Bus Matrix Initiators include:

- Initiator 0 – CPU X Data Bus (CPU XDS)
- Initiator 1 – CPU Y Data Bus (CPU YDS)
- Initiator 2 – DMA

- Initiator 3 – CPU Instruction Bus (CPU IS)
- Initiator 4 – Nonvolatile Memory Controller
- Initiator 5 – In-Circuit Debugger

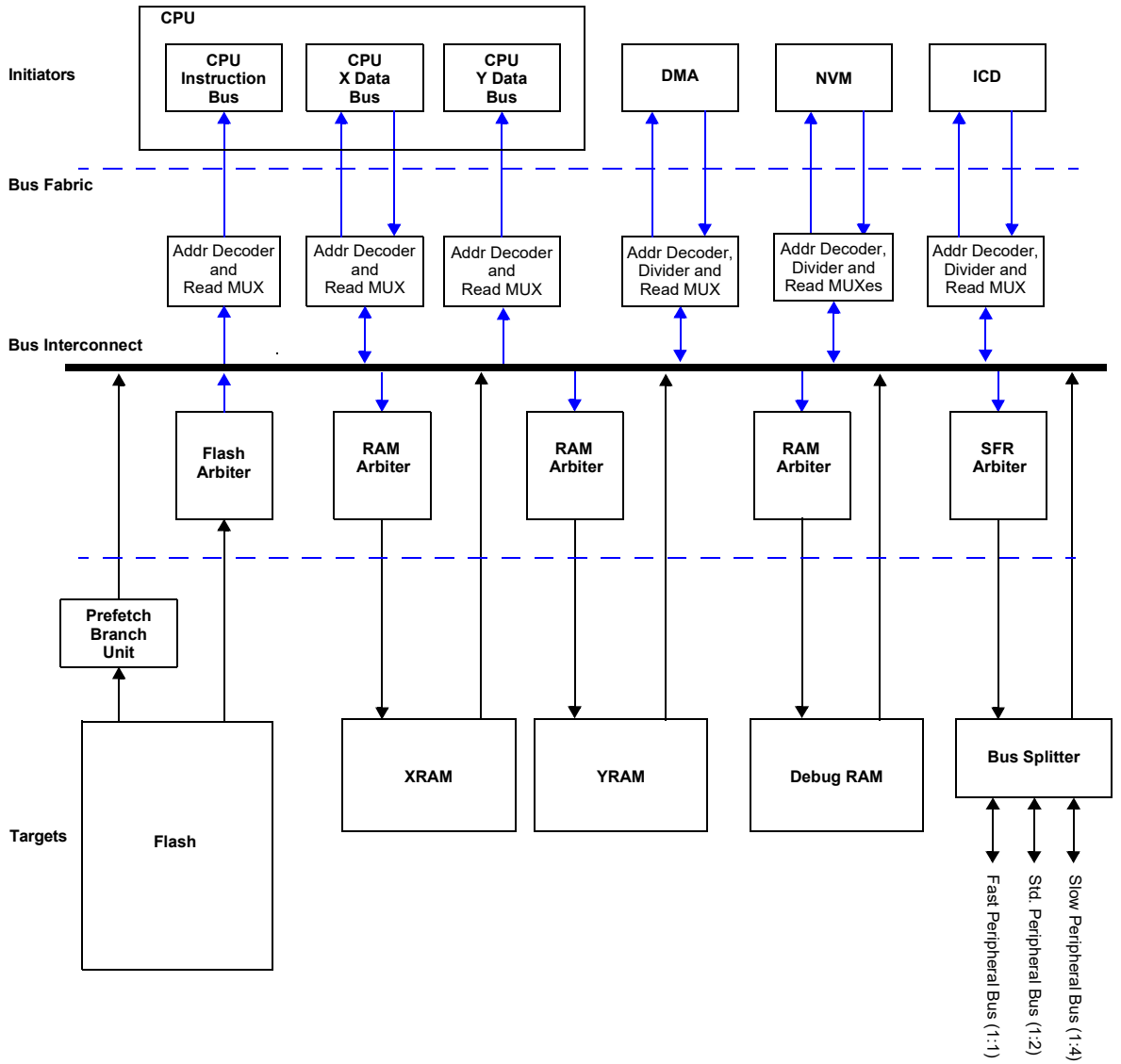
Bus Matrix Targets include:

- Program Flash (data reads)
- Peripheral Buses (through Bus Splitter)
- XRAM interface
- YRAM interface
- Debug RAM

Note: XRAM and YRAM simultaneous access is supported.

Figure 4-2 shows the typical block diagram of the Bus Matrix.

Figure 4-2. Bus Matrix Initiators and Targets



Note: See Table 4-5 for initiator/target options.

4.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0770	BMXINITPR	31:24								
		23:16								
		15:8								
		7:0							NVMPR	DMAPR
0x0774	BMXIRAML	31:24	BMXIRAML[31:24]							
		23:16	BMXIRAML[23:16]							
		15:8	BMXIRAML[15:8]							
		7:0	BMXIRAML[7:2]							
0x0778	BMXIRAMH	31:24	BMXIRAMH[31:24]							
		23:16	BMXIRAMH[23:16]							
		15:8	BMXIRAMH[15:8]							
		7:0	BMXIRAMH[7:2]							
0x077C	BMXXDATERR	31:24								
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8					YRAMRERR	XRAMRERR		PGSPCRERR
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR
0x0780	BMXYDATERR	31:24								
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8					YRAMRERR	XRAMRERR		
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR
0x0784	BMXDMAERR	31:24								
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8					YRAMRERR	XRAMRERR		PGSPCRERR
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR
0x0788	BMXCPUERR	31:24								
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8					YRAMRERR	XRAMRERR		
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR
0x078C	BMXNVMERR	31:24								
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8					YRAMRERR	XRAMRERR		PGSPCRERR
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR
0x0790	BMXICDERR	31:24				DBGWERR				
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8				DBGRERR	YRAMRERR	XRAMRERR		
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR

4.3.1 Bus Initiator Priority Control Register

Name: BMXINITPR
Offset: 0x770

Notes:

1. CPU has the highest priority.
2. The ICD does not have an associated INITPR bit and is always forced to the lowest priority.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
							NVMPR	DMAPR
Access							R/W	R/W
Reset							0	0

Bit 1 – NVMPR NVM Priority Override bit

Value	Description
1	Raise NVM initiator RAM access above CPU
0	No change to NVM initiator RAM access priority

Bit 0 – DMAPR DMA Priority Override bit

Value	Description
1	Raise DMA initiator RAM access above CPU
0	No change to DMA initiator RAM access priority

4.3.2 BMX Instruction RAM Low Address Register

Name: BMXIRAML
Offset: 0x774

Bit	31	30	29	28	27	26	25	24
	BMXIRAML[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BMXIRAML[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BMXIRAML[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BMXIRAML[7:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bits 31:24 – BMXIRAML[31:24] Lower Boundary Address for Instruction RAM bits
Defines the lower boundary address (inclusive) for instruction RAM.

Bits 23:16 – BMXIRAML[23:16] Lower Boundary Address for Instruction RAM bits

Bits 15:8 – BMXIRAML[15:8] Lower Boundary Address for Instruction RAM bits

Bits 7:2 – BMXIRAML[7:2] Lower Boundary Address for Instruction RAM bits

4.3.3 BMX Instruction RAM High Address Register

Name: BMXIRAMH
Offset: 0x778

Bit	31	30	29	28	27	26	25	24
	BMXIRAMH[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BMXIRAMH[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BMXIRAMH[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BMXIRAMH[7:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bits 31:24 – BMXIRAMH[31:24] Upper Boundary Address for Instruction RAM bits
Defines the upper boundary address (non-inclusive) for instruction RAM.

Bits 23:16 – BMXIRAMH[23:16] Upper Boundary Address for Instruction RAM bits

Bits 15:8 – BMXIRAMH[15:8] Upper Boundary Address for Instruction RAM bits

Bits 7:2 – BMXIRAMH[7:2] Upper Boundary Address for Instruction RAM bits

4.3.4 BMX Error Status Register for Initiator

Name: BMXXDATERR
Offset: 0x77C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
					YRAMRERR	XRAMRERR		PGSPCRERR
Access					R/HS/C	R/HS/C		R/HS/C
Reset					0	0		0

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by invalid instruction write outside of IRAM space
0	No IRAM write address errors

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space
0	No unimplemented address write error

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space
0	No invalid target write error

Bit 11 – YRAMRERR YRAM Read Error Flag bits(1)

Value	Description
1	Bus error generated by YRAM read operation
0	No error on YRAM read operation

Bit 10 – XRAMRERR XRAM Read Error Flag bits(1)

Value	Description
1	Bus error generated by XRAM read operation
0	No error on XRAM read operation

Bit 8 – PGSPCRERR Program Space Read Error Flag bit

Value	Description
1	Bus error generated by program space read operation
0	No error on program space read operation

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of IRAM space
0	No IRAM read address errors

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space
0	No unimplemented address write error

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space
0	No invalid target error

4.3.5 BMX Error Status Register for Initiator

Name: BMXYDATERR
Offset: 0x780

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
					YRAMRERR	XRAMRERR		
Access					R/HS/C	R/HS/C		
Reset					0	0		

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by invalid instruction write outside of IRAM space
0	No IRAM write address errors

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space
0	No unimplemented address write error

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space
0	No invalid target write error

Bit 11 – YRAMRERR YRAM Read Error Flag bits(1)

Value	Description
1	Bus error generated by YRAM read operation
0	No error on YRAM read operation

Bit 10 – XRAMRERR XRAM Read Error Flag bits(1)

Value	Description
1	Bus error generated by XRAM read operation
0	No error on XRAM read operation

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of IRAM space
0	No IRAM read address errors

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space
0	No unimplemented address write error

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space
0	No invalid target error

4.3.6 BMX Error Status Register for Initiator

Name: BMXDMAERR
Offset: 0x784

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
					YRAMRERR	XRAMRERR		PGSPCRERR
Access					R/HS/C	R/HS/C		R/HS/C
Reset					0	0		0

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by invalid instruction write outside of IRAM space
0	No IRAM write address errors

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space
0	No unimplemented address write error

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space
0	No invalid target write error

Bit 11 – YRAMRERR YRAM Read Error Flag bits(1)

Value	Description
1	Bus error generated by YRAM read operation
0	No error on YRAM read operation

Bit 10 – XRAMRERR XRAM Read Error Flag bits(1)

Value	Description
1	Bus error generated by XRAM read operation
0	No error on XRAM read operation

Bit 8 – PGSPCRERR Program Space Read Error Flag bit

Value	Description
1	Bus error generated by program space read operation
0	No error on program space read operation

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of IRAM space
0	No IRAM read address errors

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space
0	No unimplemented address write error

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space
0	No invalid target error

4.3.7 BMX Error Status Register for CPU

Name: BMXCPUERR
Offset: 0x788

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
					YRAMRERR	XRAMRERR		
Access					R/HS/C	R/HS/C		
Reset					0	0		

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by invalid instruction write outside of IRAM space
0	No IRAM write address errors

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space
0	No unimplemented address write error

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space
0	No invalid target write error

Bit 11 – YRAMRERR YRAM Read Error Flag bits(1)

Value	Description
1	Bus error generated by YRAM read operation
0	No error on YRAM read operation

Bit 10 – XRAMRERR XRAM Read Error Flag bits(1)

Value	Description
1	Bus error generated by XRAM read operation
0	No error on XRAM read operation

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of IRAM space
0	No IRAM read address errors

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space
0	No unimplemented address write error

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space
0	No invalid target error

4.3.8 BMX Error Status Register for Initiator

Name: BMXNVMERR
Offset: 0x78C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
					YRAMRERR	XRAMRERR		PGSPCRERR
Access					R/HS/C	R/HS/C		R/HS/C
Reset					0	0		0

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by invalid instruction write outside of IRAM space
0	No IRAM write address errors

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space
0	No unimplemented address write error

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space
0	No invalid target write error

Bit 11 – YRAMRERR YRAM Read Error Flag bits(1)

Value	Description
1	Bus error generated by YRAM read operation
0	No error on YRAM read operation

Bit 10 – XRAMRERR XRAM Read Error Flag bits(1)

Value	Description
1	Bus error generated by XRAM read operation
0	No error on XRAM read operation

Bit 8 – PGSPCRERR Program Space Read Error Flag bit

Value	Description
1	Bus error generated by program space read operation
0	No error on program space read operation

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of IRAM space
0	No IRAM read address errors

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space
0	No unimplemented address write error

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space
0	No invalid target error

4.3.9 BMX Error Status Register for Initiator x

Name: BMXICDERR
Offset: 0x790

Bit	31	30	29	28	27	26	25	24
				DBGWERR				
Access				R/HS/C				
Reset				0				

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
				DBGRERR	YRAMRERR	XRAMRERR		
Access				R/HS/C	R/HS/C	R/HS/C		
Reset				0	0	0		

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 28 – DBGWERR Debug RAM Write Error bit

Value	Description
1	Bus error generated by debug RAM write operation
0	No error on debug RAM write operation

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by invalid instruction write outside of IRAM space
0	No IRAM write address errors

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space
0	No unimplemented address write error

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space
0	No invalid target write error

Bit 12 – DBGRERR Debug RAM Read Error bit

Value	Description
1	Bus error generated by debug RAM read operation
0	No error on debug RAM read operation

Bit 11 – YRAMRERR YRAM Read Error Flag bits(1)

Value	Description
1	Bus error generated by YRAM read operation
0	No error on YRAM read operation

Bit 10 – XRAMRERR XRAM Read Error Flag bits(1)

Value	Description
1	Bus error generated by XRAM read operation
0	No error on XRAM read operation

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of IRAM space
0	No IRAM read address errors

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space
0	No unimplemented address write error

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space
0	No invalid target error

4.4 BMX Operation

The purpose of the BMX is to decode addresses and provide arbitration between multiple initiators requesting access to the same target, and to provide concurrent accesses to multiple targets from different initiators.

4.4.1 Arbitration

BMX supports a decentralized fixed priority arbitration scheme. Each target has an independent arbitrator which will grant read and write requests to an initiator when that target is available. The default priority of an initiator is fixed and determined by the initiator's index on the bus. The priorities are shown in Table 4-3.

Table 4-3. Initiator Priority

Priority	Initiator Index	Type
Highest	0	CPU X Data Bus (CPU XDS)
	1	CPU Y Data Bus (CPU YDS)
	2	DMA
	3	CPU Instruction Bus (CPU IS)
	4	Flash Controller
Lowest	5	In-Circuit Debugger

4.4.1.1 Priority Overrides

The BMXINITPR register can be used to temporarily override the default priority scheme when accessing RAM targets. When the corresponding bit is set, an initiator will have its priority raised above the native priorities of the CPU and other initiators. The ICD bus master does not support priority overrides; it will always be the lowest priority and does not have an associated BMXINITPR bit.

If multiple override bits are set, priority between the overridden initiators is determined by their natural priority order. For initiators that do not win arbitration, the BMX will stall the initiator of the lower priority transaction until a subsequent cycle, after the target access completes, when the initiator does win arbitration. Losing initiators will be forced to stall until no higher priority initiators are requesting the target.

4.4.2 Concurrency

As the Bus Matrix has independent address decode for each initiator and independent arbiters for each target, it supports concurrent accesses. As long as each initiator is accessing a unique target, all reads and writes can proceed simultaneously.

4.4.3 Write Buffers

As the targets can operate on slower clock sources than the initiators, the Bus Matrix requires targets have the ability to buffer writes. The target is then responsible for completing the write. Posting the write improves performance since the write completes sooner from the perspective of the CPU.

Posting writes improves single write performance only. Back-to-back writes can cause stalls to initiators if the clocking for target is not set to a 1:1 ratio. Write buffers are only one deep, so repeated writes will be held by the Bus Matrix until the buffer is empty and the previous write completes. All transactions to a target complete in order, therefore reads are never allowed to pass writes.

4.4.4 Debug RAM

The BMX supports a RAM target for use in Debug mode only. The debug RAM space is at a fixed location and is from 0x7BFE00 to 0x7BFFFF.

The ICD initiator can always access debug RAM. For other initiators, both read and write access to debug scratch pad RAM is disallowed unless the device is in Debug mode. Access to the debug RAM when outside of Debug mode will generate a bus error and set the ADDRERR (BMXERRx[1]) bit. Additionally, debug RAM is not a valid target for the CPU instruction bus regardless of the Debug state and will cause a bus error.

Note that because the CPU does not handle traps while in Debug mode, in some circumstances, bus errors caused in Debug mode may result in a ‘soft lock’ situation that will require a device Reset to resolve.

4.4.5 Bus Error

The Bus Matrix generates a bus error exception back to an initiator on any failed access. Failed accesses can occur for a number of reasons:

- Unimplemented memory in a valid target space. For example, the DMA accessing XRAM higher than allowed by RAM array size.
- Any location in an invalid target space. For example, ICD addressing Program Space.
- Instruction bus reads outside of the memory range defined by either of the BMXIRAML/H registers. Note, the BMX will not generate bus errors for instruction bus accesses targeting Flash. The Flash controller will generate the bus error and indicate status in NVMCON register.
- Instruction bus reads of debug RAM target, regardless of the BMXIRAML/H setting.
- A RAM write request is generated to the instruction RAM space defined by the BMXIRAML/H registers.
- Flash or RAM read results in an ECC Double-Bit Error (ECC DED). This bus error is generated by the target and passed up to the initiator, which will cause the initiator's Bus Error trap.
- Target indicates a bus error (root causes are target-specific).

When a bus error is generated within the BMX, it will set the relevant bit within the BMXxERR register for the initiator which generated the transaction. The user can read error registers from some combination of the initiator that caused the trap, from BMXxERR, and from the target error registers, to diagnose the error. See Table 4-4 for initiator indexes.

As the BMX supports simultaneous read and write operations to some targets, bus errors are split into separate read or write operation errors. This ensures that even if two simultaneous errors occur, they can both be captured.

Table 4-4. Target to Bus Error Index Mapping

TGTRERRy (BMXxERR[13:8])	Target
0	Program Space Read
1	Bus Splitter/SFRs
2	XRAM
3	YRAM
4	Debug RAM Target

4.4.5.1 Valid Targets

Not all targets are valid destinations for each initiator. Refer to Table 4-5 for details on which targets are valid for each initiator.

Accessing an invalid target will generate a bus error and set the BADTGTWERR (BMXERRx[16]) or BADTGTRERR (BMXERRx[0] bit.

Table 4-5. Valid Targets by Initiator

Initiators	Targets				
	PS Read	XRAM	YRAM	Debug RAM	SFRs
CPU X Data	✔	✔	✔	If Debug mode is enabled	✔
CPU Y Data	—	✔	✔	If Debug mode is enabled	—
DMA	✔	✔	✔	—	✔
CPU Instruction	—	✔	✔	—	—
NVM	✔	—	—	—	—
ICD	—	✔	✔	✔	✔

4.4.5.2 Initiator Bus Error Handling

The BMX does not directly generate any trap or interrupt signals on error events. Instead, the corresponding initiator will generate the event. Similarly, the BMX will not alter any data.

While initiators will handle bus errors differently, depending on individual module requirements, the most typical handling of a bus error is as follows:

- Ignore or discard returned read data, retain failed write data
- Abort or halt operation in progress
- Generate interrupt or trap signal to the interrupt controller indicating a Bus Error state

4.4.5.3 Target Bus Error Handling

A target will generate a bus error in any situation where it is unable to deliver data back to the BMX on a read event or unable to latch data on a write event. The target will retain relevant error status information which can be used to determine the cause of the bus error.

The BMX will set the Write Target Error bits or Read Target Error bits for the specific target that caused the error within the Error register for the initiator that generated the bus transaction.

For example, an ECC DED error on an XRAM read by the CPU would set the XRAMERR (BMXXDATERR) bit. Refer to Table 4-4 for target indices.

4.4.5.4 BMXERRx Registers

To properly deal with speculative data fetches, the BMXERRx registers have somewhat different behavior than standard error registers in other modules. When a bus error occurs, the BMX will always set the corresponding bit in the initiator's BMXERRx register and set the bus error signal going into that initiator. However, the initiator may choose to do nothing about the bus error. This is most common in the case of the CPU when it is performing speculative fetches for data that never end up getting used. If the erroneous data fetch is unused, there will be no trap or interrupt error event.

As a result, the BMX does not have clear information on when the BMXERRx error flags can safely be cleared, nor can it ensure a software routine will run that will be able to read and clear set flags. Therefore, to prevent past unused BMX error flags from staying set, the BMXERRx register will clear any previous flags whenever a new error event happens.

This effectively turns the BMXERRx register into a 'one-hot' register, where there will generally only be a single bit set at a time, reflecting the most recent error event. It is only possible for two bits to be set in the event that read and write errors occur simultaneously.

4.4.6 Execute from RAM

BMX provides support for the CPU to execute from RAM. Execute from RAM will use the same priority and arbitration scheme as described earlier in this chapter.

4.4.6.1 Instruction RAM Window Registers

The BMXIRAML and BMXIRAMH registers define the valid address range for instruction RAM. Instruction bus accesses to RAM outside of this window will generate a bus error.

BMXIRAML defines the lower address of the window, including the address defined within BMXIRAML. The upper valid address in the window is BMXIRAMH – 1. Note that this means the BMXIRAMH register is exclusive, instead of inclusive, which differs from the BMXIRAML register. This is done to provide an easy disable mechanism. Both registers will default to the same value (e.g., '0's), effectively disabling execute from RAM at start-up.

4.4.6.2 Write and Read Protection of Instruction RAM Space

To ensure that the RAM allocated for execution is not accidentally modified, writes to RAM locations between BMXIRAML and BMXIRAMH are prevented by the BMX. A bus error will be generated and the IRAMWRERR (BMXERRx[18]) bit will be set if an initiator attempts to write to instruction RAM space.

Though IRAM space is executable memory, it will not support any read protection via code protect. Data in the IRAM area are readable as normal RAM space.

Attempts to execute RAM space outside of the IRAM window will generate a bus error and set the IRAMRDERR (BMXERRx[2]) bit.

4.4.6.3 Instruction RAM Configuration Sequence

The proper sequence for configuring and using IRAM is as follows:

1. Write BMXIRAML/BMXIRAMH to '0' to ensure no RAM locations are write protected. This step is optional if the sequence is done at Reset as both registers reset to a value of '0'.
2. Copy instructions into desired RAM locations from Flash or other storage mechanism (e.g., bootloader).
3. Set the IRAM start and end addresses by writing to the BMXIRAML and BMXIRAMH registers.
4. Vector to any RAM location within the IRAM window to begin execution from RAM.

The BMXIRAML and BMXIRAMH registers are implemented with write protection to ensure that users have the ability to lock the designated IRAM range after configuration. The BMX does not

support aborts on RAM accesses. The CPU will be responsible for ignoring unneeded instruction data delivered from the RAM access.

When switching from executing from NVM to executing from RAM, the following will occur:

1. Any active NVM fetches would be aborted by the PBU. The new RAM instruction fetch will be immediately sent to the CPU as soon as data are available.
2. To prevent cache thrashing, PBU is put into a 'Standby' state while running from RAM.

5. Data Memory

This section describes the data memory in dsPIC33A devices. The dsPIC33AK128MC106 has a data width of 32 bits. All internal registers and data space memory are organized as 32 bits wide. The data spaces can be accessed as one 16 Kbyte linear address range. The data memory is accessed using Address Generation Units (AGUs) and separate data paths.

The dsPIC33AK128MC106 device family incorporates ECC support for data read/write and forced Fault Injection capability for use by Functional Safety focused customers. The ECC controller provides interrupt output for single and double-bit errors.

The following data memory features are implemented:

- Three Address Generation Units: two for read and one for write
- Two SRAM data blocks for independent read and write
- ECC support for data read and write
- ECC provides interrupt output for single and double bit errors
- Forced Fault Injection capability for use by Functional Safety focused customers

5.1 Device-Specific Information

Table 5-1 summarizes the available data memory size and the Start and End address for X and Y RAM spaces.

Table 5-1. Data Memory

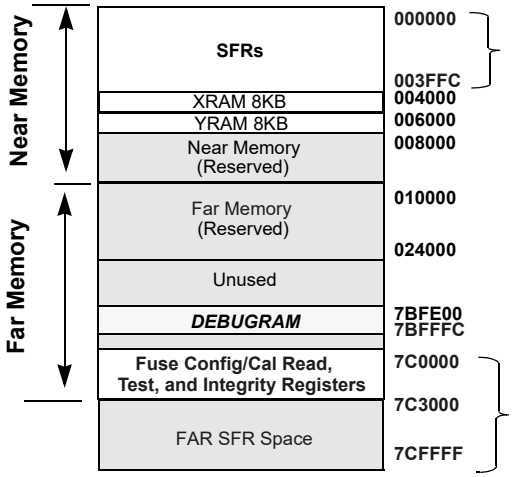
Device Variant	Data Memory	X RAM			Y RAM		
		Memory	Start Address	End Address	Memory	Start Address	End Address
dsPIC33AK128 MC1XX	16 Kbytes	8 Kbytes	0x004000	0x006000	8 Kbytes	0x006000	0x008000
dsPIC33AK64 MC1XX	16 Kbytes	8 Kbytes	0x004000	0x006000	8 Kbytes	0x006000	0x008000
dsPIC33AK32 MC1XX	8 Kbytes	8 Kbytes	0x004000	0x005000	8 Kbytes	0x005000	0x006000
Note: Any initiator can access both X and Y RAM.							

5.2 Architectural Overview

This section describes the features of the data memory. dsPIC33AK128MC106 devices have a 16 Kbyte data memory space. Figure 5-1 shows a typical memory map for dsPIC33AK128MC106 family devices.

The dsPIC33AK128MC106 family of devices implements X and Y RAM in equal sizes. RAM width is 32-bit in addition to seven parity bits to implement ECC. The DEBUGRAM is 512 bytes.

Figure 5-1. Data Memory Map for dsPIC33AK128MC106 Family Devices



Note:

1. Memory areas are not shown to scale.
2. The dsPIC33AK128MC106 devices will have a unified memory map with non-overlapping Program and Data address spaces as shown in Figure 5-1. The address space is 8 MB total starting at 000000 and ending at 0x7CFFFF.

SFR space and data RAM are mapped starting at address 000000 to allow near memory access from instructions that can encode a memory address. The near address range for most file register instructions is 64KB total. The SFR space is 16 KB and is separated into different clock speeds via a peripheral bus splitter:

- The address range 000000 - 0007FF is associated with a fast (1:1 System Clock) peripheral bus. A minimum of SFRs are mapped into this region.
- The address range 000800 - 002FFF is associated with the standard speed (1:2 or slower) peripheral bus. The majority of the peripheral SFRs will be mapped into this region.
- The address range 003000 - 003FFF is associated with the slow speed (1:4 or slower) peripheral bus. Peripheral SFRs that require infrequent access will be mapped into this region.
- The address range 7C0000 - 7CFFFF is associated with the slow speed (1:4 or slower) peripheral bus dedicated to the calibration and configuration registers in “far” memory space.

5.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3580	BMXECCxCON (= x or y)	31:24								
		23:16								
		15:8	ECCEN							
		7:0								FLTINJ
0x3584	BMXECCxSTAT (= x or y)	31:24								
		23:16								
		15:8				ESEL				PWBNE
		7:0			SECO	SEC			DEDO	DED
0x3588	BMXECCxFPTR (= x or y)	31:24								
		23:16								
		15:8			FLTPTR2[5:0]					
		7:0			FLTPTR1[5:0]					
0x358C	BMXECCxFADDR (= x or y)	31:24								
		23:16	ADDR[23:16]							
		15:8	ADDR[15:8]							
		7:0	ADDR[7:2]							
0x3590	BMXECCxEADDR (x = x or y)	31:24								
		23:16	ADDR[23:16]							
		15:8	ADDR[15:8]							
		7:0	ADDR[7:2]							
0x3594	BMXECCxEDATA (= x or y)	31:24								
		23:16	DATA[23:16]							
		15:8	DATA[15:8]							
		7:0	DATA[7:0]							
0x3598	BMXECCxVAL (= x or y)	31:24								
		23:16								
		15:8								
		7:0		ECC[6:0]						
0x359C	BMXECCxSYND (x = x or y)	31:24								
		23:16								
		15:8								
		7:0		SYND[6:0]						
0x35A0	PWBECCxCON (= x or y)	31:24								
		23:16								
		15:8	ECCEN							
		7:0								FLTINJ
0x35A4	PWBECCxSTAT (= x or y)	31:24								
		23:16								
		15:8				ESEL				PWBNE
		7:0			SECO	SEC			DEDO	DED
0x35A8	PWBECCxFPTR (= x or y)	31:24								
		23:16								
		15:8			FLT2LPTR[5:0]					
		7:0			FLT1LPTR[5:0]					
0x35AC	PWBECCxFADDR (= x or y)	31:24								
		23:16	ADDR[23:16]							
		15:8	ADDR[15:8]							
		7:0	ADDR[7:2]							
0x35B0	PWBECCxEADDR (= x or y)	31:24								
		23:16	ADDR[23:16]							
		15:8	ADDR[15:8]							
		7:0	ADDR[7:2]							
0x35B4	PWBECCxEDATA (= x or y)	31:24								
		23:16	DATA[23:16]							
		15:8	DATA[15:8]							
		7:0	DATA[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x35B8	PWBECCxVAL (= x or y)	31:24								
		23:16								
		15:8								
		7:0		ECC[6:0]						
0x35BC	PWBECCxSYND (= x or y)	31:24								
		23:16								
		15:8								
		7:0		SYND[6:0]						
0x35C0	BMXECCyCON (= x or y)	31:24								
		23:16								
		15:8	ECCEN							
		7:0								FLTINJ
0x35C4	BMXECCySTAT (= x or y)	31:24								
		23:16								
		15:8				ESEL				PWBNE
		7:0			SECO	SEC			DEDO	DED
0x35C8	BMXECCyFPTR (= x or y)	31:24								
		23:16								
		15:8			FLTPTR2[5:0]					
		7:0			FLTPTR1[5:0]					
0x35CC	BMXECCyFADDR (= x or y)	31:24								
		23:16	ADDR[23:16]							
		15:8	ADDR[15:8]							
		7:0	ADDR[7:2]							
0x35D0	BMXECCxEADDR (y = x or y)	31:24								
		23:16	ADDR[23:16]							
		15:8	ADDR[15:8]							
		7:0	ADDR[7:2]							
0x35D0	PWBECCyEADDR (= x or y)	31:24								
		23:16	ADDR[23:16]							
		15:8	ADDR[15:8]							
		7:0	ADDR[7:2]							
0x35D4	BMXECCyEDATA (= x or y)	31:24								
		23:16	DATA[23:16]							
		15:8	DATA[15:8]							
		7:0	DATA[7:0]							
0x35D8	BMXECCyVAL (= x or y)	31:24								
		23:16								
		15:8								
		7:0		ECC[6:0]						
0x35DC	BMXECCxSYND (y = x or y)	31:24								
		23:16								
		15:8								
		7:0		SYND[6:0]						
0x35E0	PWBECCyCON (= x or y)	31:24								
		23:16								
		15:8	ECCEN							
		7:0								FLTINJ
0x35E4	PWBECCySTAT (= x or y)	31:24								
		23:16								
		15:8				ESEL				PWBNE
		7:0			SECO	SEC			DEDO	DED
0x35E8	PWBECCyFPTR (= x or y)	31:24								
		23:16								
		15:8			FLT2LPTR[5:0]					
		7:0			FLT1LPTR[5:0]					
0x35EC	PWBECCyFADDR (= x or y)	31:24								
		23:16	ADDR[23:16]							
		15:8	ADDR[15:8]							
		7:0	ADDR[7:2]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x35F0 ... 0x35F3	Reserved									
0x35F4	PWBECCyEDATA (= x or y)	31:24								
		23:16	DATA[23:16]							
		15:8	DATA[15:8]							
		7:0	DATA[7:0]							
0x35F8	PWBECCyVAL (= x or y)	31:24								
		23:16								
		15:8								
		7:0		ECC[6:0]						
0x35FC	PWBECCySYND (= x or y)	31:24								
		23:16								
		15:8								
		7:0		SYND[6:0]						

5.3.1 BMX ECC RAM Control Register

Name: BMXECCaCON (a = x or y)
Offset: 0x3580, 0x35C0

Note:

1. This bit can be set by software, but not cleared. It is cleared by any device Reset.

Legend: R = Readable bit; W = Writable bit; S = Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ECCEN							
Access	R/S							
Reset	0							

Bit	7	6	5	4	3	2	1	0
								FLTINJ
Access								R/W
Reset								0

Bit 15 – ECCEN Enable ECC functionality.(1)

Write and Read for X or Y data space functionality of the ECC block is enabled, depending on the instance of this register. By default, ECC blocks are disabled on device Reset. It is the software responsibility to initialize the RAM locations with valid data, then set this bit to "1". Once set, the software cannot clear this bit.

Value	Description
1	ECC is enabled
0	ECC is disabled

Bit 0 – FLTINJ Fault Injection Enable bit

Value	Description
1	Fault injection is enabled when the content of read Data RAM[23:2] matches ECCFADDR[23:2]
0	Fault injection is disabled

5.3.2 BMX ECC RAM Status Register

Name: BMXECCaSTAT (a = x or y)
Offset: 0x3584, 0x35C4

Legend: R = Readable bit; W = Writable bit; C = Clearable bit; HS = Hardware Settable bit
Note:

1. This bit selects whether the ECCaEADDR, ECCaEDATA, ECCaVAL and ECCaSYND registers display the SEC or DED error event information.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				ESEL				PWBNE
Access				R/W				R
Reset				0				0

Bit	7	6	5	4	3	2	1	0
			SECO	SEC			DEDO	DED
Access			R/C/HS	R/C/HS			R/C/HS	R/C/HS
Reset			0	0			0	0

Bit 12 – ESEL Error Reporting Select bit(1)

Value	Description
1	Show SEC error event information
0	Show DED error event information

Bit 8 – PWBNE Posted Write Buffer Not Empty Status bit

Value	Description
1	PWB has to perform at least one read or data merge or write operation, i.e., it is not empty
0	PWB is empty. All data committed to RAM

Bit 5 – SECO Single Error Correction Event Overflow Status bit

Value	Description
1	SEC event not captured due to overflow
0	No SEC event overflow detected

Bit 4 – SEC Single Error Correction Status bit

Value	Description
1	Single bit error detected
0	Single bit error not detected

Bit 1 – DEDO Double Error Detection Event Overflow Status bit

Value	Description
1	DED event not captured due to overflow
0	No DED event overflow detected

Bit 0 – DED Double Error Detection Indicator Status bit

Value	Description
1	Double bit error detected
0	Double bit error not detected

5.3.3 BMX ECC RAM Fault Pointer Register

Name: BMXECCaFPTR (a = x or y)
Offset: 0x3588, 0x35C8

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			FLTPTR2[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			FLTPTR1[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 13:8 – FLTPTR2[5:0] ECC Fault Injection Bit Pointer 2

Value	Description
111111-100111	No fault injection
100110	Fault injection (bit inversion) occurs on bit 38 of ECC bit order
...	
000001	Fault injection (bit inversion) occurs on bit 1 of ECC bit order
000000	Fault injection (bit inversion) occurs on bit 0 of ECC bit order

Bits 5:0 – FLTPTR1[5:0] ECC Fault Injection Bit Pointer 1

Value	Description
111111-100111	No fault injection
100110	Fault injection (bit inversion) occurs on bit 38 of ECC bit order
...	
000001	Fault injection (bit inversion) occurs on bit 1 of ECC bit order
000000	Fault injection (bit inversion) occurs on bit 0 of ECC bit order

5.3.4 BMX ECC RAM Fault Injection Address Register

Name: BMXECCaFADDR (a = x or y)
Offset: 0x358C, 0x35CC

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	ADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[7:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bits 23:16 – ADDR[23:16] ECC Fault Injection Address bits

Bits 15:8 – ADDR[15:8] ECC Fault Injection Address bits

Bits 7:2 – ADDR[7:2] ECC Fault Injection Address bits
These register bits are compared against the RAM address read during fault injection cycle. These register bits' values represent the relative address from the start address of X or Y data spaces depending on the Read or Write instance of this register. The start address should be added to this relative address to know the corresponding data RAM address.

5.3.5 BMX ECC RAM Error Address Register

Name: BMXECCxEADDR (a = x or y)
Offset: 0x3590, 0x35D0

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	ADDR[23:16]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDR[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[7:2]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC		
Reset	0	0	0	0	0	0		

Bits 23:16 – ADDR[23:16] ECC RAM Read Data Address bits

Bits 15:8 – ADDR[15:8] ECC RAM Read Data Address bits

Bits 7:2 – ADDR[7:2] ECC RAM Read Data Address bits
These bits register the location of the RAM read address when any of the following bits are set high: ECCSTAT.SEC or ECCSTAT.DED. These register bits' values represent the relative address from the start address of X or Y data spaces depending on the instance of this register. The start address should be added to this relative address to know the corresponding system bus address.

5.3.6 BMX ECC RAM Error Data Register

Name: BMXECCaEDATA (a = x or y)
Offset: 0x3594, 0x35D4

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – DATA[23:0] RAM Read Error Data bits
The data captured when an error occurs, i.e., when any of the following bits are set high: ECCSTAT.SEC or ECCSTAT.DED.

5.3.7 BMX ECC Value Register

Name: BMXECCaVAL (a = x or y)
Offset: 0x3598, 0x35D8

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		ECC[6:0]						
Access		R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset		0	0	0	0	0	0	0

Bits 6:0 – ECC[6:0] Error Correcting Code bits
These bits register the Error Correcting Code associated with the RAM read data when any of the following bits are set high: ECCSTAT.SEC or ECCSTAT.DED.

5.3.8 BMX ECC Syndrome Register

Name: BMXECCxSYND (a = x or y)
Offset: 0x359C, 0x35DC

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		SYND[6:0]						
Access		R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset		0	0	0	0	0	0	0

Bits 6:0 – SYND[6:0] Syndrome Value bits
These bits register the syndrome value associated with the RAM read data when any of the following bits are set high: ECCSTAT.SEC or ECCSTAT.DED.

5.3.9 PWB ECC RAM Control Register

Name: PWBECCaCON (a = x or y)
Offset: 0x35A0, 0x35E0

Note:

1. This bit can be set by software, but not cleared. It is cleared by any device Reset.

Legend: R = Readable bit; W = Writable bit; S = Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ECCEN							
Access	R/S							
Reset	0							

Bit	7	6	5	4	3	2	1	0
								FLTINJ
Access								R/W
Reset								0

Bit 15 – ECCEN Enable ECC functionality.(1)

Write and Read for X or Y data space functionality of the ECC block is enabled, depending on the instance of this register. By default, ECC blocks are disabled on device Reset. It is the software responsibility to initialize the RAM locations with valid data, then set this bit to "1". Once set, the software cannot clear this bit.

Value	Description
1	ECC is enabled
0	ECC is disabled

Bit 0 – FLTINJ Fault Injection Enable bit

Value	Description
1	Fault injection is enabled when the content of read Data RAM[23:2] matches ECCFADDR[23:2]
0	Fault injection is disabled

5.3.10 PWB ECC RAM Status Register

Name: PWBECCaSTAT (a = x or y)
Offset: 0x35A4, 0x35E4

Legend: R = Readable bit; W = Writable bit; C = Clearable bit; HS = Hardware Settable bit
Note:

1. This bit selects whether the ECCaEADDR, ECCaEDATA, ECCaVAL and ECCaSYND registers display the SEC or DED error event information.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				ESEL				PWBNE
Access				R/W				R
Reset				0				0

Bit	7	6	5	4	3	2	1	0
			SECO	SEC			DEDO	DED
Access			R/C/HS	R/C/HS			R/C/HS	R/C/HS
Reset			0	0			0	0

Bit 12 – ESEL Error Reporting Select bit(1)

Value	Description
1	Show SEC error event information
0	Show DED error event information

Bit 8 – PWBNE Posted Write Buffer Not Empty Status bit

Value	Description
1	PWB has to perform at least one read or data merge or write operation, i.e., it is not empty
0	PWB is empty, all data committed to RAM

Bit 5 – SECO Single Error Correction Event Overflow Status bit

Value	Description
1	SEC event not captured due to overflow
0	No SEC event overflow detected

Bit 4 – SEC Single Error Correction Status bit

Value	Description
1	Single bit error detected
0	Single bit error not detected

Bit 1 – DEDO Double Error Detection Event Overflow Status bit

Value	Description
1	DED event not captured due to overflow
0	No DED event overflow detected

Bit 0 – DED Double Error Detection Indicator Status bit

Value	Description
1	Double bit error detected
0	Double bit error not detected

5.3.11 PWB ECC RAM Fault Pointer Register

Name: PWBECCaFPTR (a = x or y)
Offset: 0x35A8, 0x35E8

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			FLT2LPTR[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			FLT1LPTR[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 13:8 – FLT2LPTR[5:0] ECC Fault Injection Bit Pointer 2

Value	Description
111111-100111	No fault injection
100110	Fault injection (bit inversion) occurs on bit 38 of ECC bit order
...	
000001	Fault injection (bit inversion) occurs on bit 1 of ECC bit order
000000	Fault injection (bit inversion) occurs on bit 0 of ECC bit order

Bits 5:0 – FLT1LPTR[5:0] ECC Fault Injection Bit Pointer 1

Value	Description
111111-100111	No fault injection
100110	Fault injection (bit inversion) occurs on bit 38 of ECC bit order
...	
000001	Fault injection (bit inversion) occurs on bit 1 of ECC bit order
000000	Fault injection (bit inversion) occurs on bit 0 of ECC bit order

5.3.12 PWB ECC RAM Fault Injection Address Register

Name: PWBECCaFADDR (a = x or y)
Offset: 0x35AC, 0x35EC

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	ADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[7:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bits 23:16 – ADDR[23:16] ECC Fault Injection Address bits

Bits 15:8 – ADDR[15:8] ECC Fault Injection Address bits

Bits 7:2 – ADDR[7:2] ECC Fault Injection Address bits
These register bits are compared against the RAM address read during the fault injection cycle. These register bits' values represent the relative address from the start address of X or Y data spaces depending on the read or write instance of this register. The start address should be added to this relative address to know the corresponding data RAM address.

5.3.13 PWB ECC RAM Error Address Register

Name: PWBECCaEADDR (a = x or y)
Offset: 0x35B0, 0x35D0

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	ADDR[23:16]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDR[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[7:2]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC		
Reset	0	0	0	0	0	0		

Bits 23:16 – ADDR[23:16] ECC RAM Read Data Address bits

Bits 15:8 – ADDR[15:8] ECC RAM Read Data Address bits

Bits 7:2 – ADDR[7:2] ECC RAM Read Data Address bits
These bits register the location of the RAM read address when any of the following bits are set high: ECCSTAT.SEC or ECCSTAT.DED. These register bits' values represent the relative address from the start address of X or Y data spaces depending on the instance of this register. The start address should be added to this relative address to know the corresponding system bus address.

5.3.14 PWB ECC RAM Error Data Register

Name: PWBECCaEDATA (a = x or y)
Offset: 0x35B4, 0x35F4

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – DATA[23:0] RAM Read Error Data bits
The data captured when an error occurs, i.e., when any of the following bits are set high: ECCSTAT.SEC or ECCSTAT.DED.

5.3.15 PWB ECC Value Register

Name: PWBECCaVAL (a = x or y)
Offset: 0x35B8, 0x35F8

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		ECC[6:0]						
Access		R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset		0	0	0	0	0	0	0

Bits 6:0 – ECC[6:0] Error Correcting Code bits
These bits register the Error Correcting Code associated with the RAM read data when any of the following bits are set high: ECCSTAT.SEC or ECCSTAT.DED.

5.3.16 PWB ECC Syndrome Register

Name: PWBECCaSYND (a = x or y)
Offset: 0x35BC, 0x35FC

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		SYND[6:0]						
Access		R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset		0	0	0	0	0	0	0

Bits 6:0 – SYND[6:0] Syndrome Value bits
These bits register the syndrome value associated with the RAM read data when any of the following bits are set high: ECCSTAT.SEC or ECCSTAT.DED.

5.4 Operation

Data memory addresses between 0x0000 and 0x3FFF are reserved for the device Special Function Registers (SFRs). The SFRs include control and status bits for the CPU and peripherals on the device.

An example data space memory map is shown in Figure 5-1.

The RAM begins at address 0x4000 and is split into two blocks, X and Y data space. For data writes, the X and Y data memory spaces are always accessed as a single, linear data space. For data reads, the X and Y data memory spaces can be accessed independently or as a single, linear space. Data reads for MCU class instructions always access the X and Y data spaces as a single combined data space. Dual source operand DSP instructions, such as the MAC instruction, access the X and Y data spaces separately to support simultaneous reads for the two source operands.

MCU instructions can use any W register as an address pointer for a data read or write operation.

5.4.1 Near Data Memory

A 64 Kbyte address space, referred to as near data memory, is reserved in the data memory space between 0x00_0000 and 0x01_0000. Near data memory is directly addressable through a 16-bit absolute address field within all file register instructions.

The memory regions included in the near data region will depend on the amount of data memory implemented for each dsPIC33AK128MC106 device variant. At a minimum, the near data region will include all of the SFRs and some of the X data memory. For devices that have smaller amounts of data memory, the near data region can include all of X memory space and possibly some or all of Y memory space. Refer to Figure 5-1 for more details.

Note: The entire 64 Kbyte near data space can be addressed directly using the `MOV` instruction. Refer to the *"dsPIC33A Programmer's Reference Manual"*, available from www.microchip.com, for further details.

5.4.2 Data Space Address Generation Units (AGUs)

dsPIC33AK128MC106 family devices contain three independent address generator units. The X RAGU and X WAGU support byte (.b), word (.w) and long (.l) word sized data space reads and writes, respectively, for MCU instructions, and word or long word reads and writes for DSP instructions. The Y AGU supports word and long word sized data reads for the DSP MAC-class of instructions only. The AGUs are each capable of supporting two types of data addressing:

- Linear Addressing
- Modulo (circular) Addressing

In addition, the X WAGU can support Bit-Reversed Addressing.

Linear and Modulo Data Addressing modes can be applied to any address within the unified address space. Although Bit- Reversed Addressing will work with any EA calculation, by definition it is only applicable to data space.

Data space memory is organized as 32-bit words; all Effective Addresses (EAs) point to bytes. Instructions can thus access any byte or aligned word (data words at an even byte address) or aligned long word (data words at an even 32-bit word address).

Misaligned accesses are not supported, and if attempted they will initiate an address error trap. The least significant 2 bits of the EA is used to determine the byte or upper/lower 16-bit word access. EA[0] will always be 1'b0 for word accesses, and EA[1:0] will always be 2'b00 for long word accesses.

SFRs and RAM support byte, word, and double word read or write operations.

When executing instructions that require just one source operand to be fetched from (and one result to be written back to) data space, the X RAGU and X WAGU are used to calculate the EAs of the source and destination, respectively. The AGUs can generate an address to point to anywhere in the 16 Mbyte address space. They support all MCU addressing modes and Modulo Addressing for low overhead circular buffers. The X WAGU also supports Bit-Reversed Addressing to facilitate FFT data reorganization.

When executing instructions which require two source operands to be concurrently fetched (i.e. the MAC class of DSP instructions), both the X RAGU and Y AGU are used simultaneously.

The dsPIC33AK128MC106 device family contains an X AGU and a Y AGU for generating data memory addresses. Both X and Y AGUs can generate any EA within the available data memory range. However, EAs that are outside of the physical memory provided return all zeros for data reads and writes to those locations and therefore have no effect. Furthermore, an address error trap will be generated. For more information on address error traps, refer to 10. Interrupt Controller.

5.4.3 Data Alignment

The ISA supports long word (32-bit), word (16-bit) and byte (8-bit) sized operations. Data is aligned in data memory and registers as long words, but all data space EAs resolve to bytes. Data word and byte reads will read the complete 32-bit word that contains the word or byte, using the LSbs of any EA to determine which word or byte to select within the CPU. The selected word or byte is placed onto the lsw or byte of the X data path (no byte accesses are possible from the Y data path as the MAC-class of instruction can only fetch words or long words). That is, data memory and registers are organized as four parallel byte-wide entities with a shared (long word) address decode but separate write lines. Data byte writes will only write to the corresponding side of the array or register which matches the byte address.

Note: Byte reads will always read the entire word, so mechanisms to clear or set peripheral status bits when read (e.g. quick flag clearing mechanisms) are not allowed.

As a consequence of this byte addressability, all EA calculations must be scaled to step through long word aligned memory. For example, the core must recognize that post modified register indirect addressing mode, [Ws]+=1, will result in a value of Ws+1 for byte operations, Ws+2 for word operations, and Ws+4 for long word operations.

Misaligned word or long word accesses are not supported. For word accesses, the LSb of the EA must be 1'b0. For long word accesses, the least significant 2 bits of the EA must be 2'b00. Therefore, care must be taken when mixing operations of different data widths or translating from 16-bit dsPIC code. Should a misaligned read or write be attempted, an address error trap will be forced. If the fault occurs during a read access, the read will be allowed to complete. If the fault occurs during a write access, the write will also be allowed to complete (inhibiting the write would have been possible but inconsistent with other situations where an errant write could not be inhibited). In both cases, the address error trap will be asserted. The next instruction (already pre-fetched and underway) will be executed while the exception is arbitrated and acknowledged. When this instruction completes, the trap will then be taken, allowing the system and/or user to examine the machine state subsequent to execution of the address fault.

Note: Byte and word ALU operations can produce results in excess of a byte or a word. However, to maintain 16-bit dsPIC backwards code compatibility, the ALU result destination write from all operations maintains the same width as that of the source operands (i.e. MSbs of the destination are not modified) and the SR is updated based only upon the state of the result data.

A sign extend (SE) instruction is provided to allow users to translate 8-bit to16-bit, and 16-bit to 32-bit signed values. Alternatively, for unsigned data, users can clear the MS portion of any W register through executing a byte or word zero extend (ZE).

Note: Care must be taken when mixing byte and word size instructions/operands.

Although most instructions are capable of operating on long word, word or byte data sizes, it should be noted that the DSP and some other instructions operate on long word or word sized data only.

Figure 5-2. Data Alignment

31	23	15	7	0	Address
Byte 3	Byte2	Byte1	Byte 0		24'h00_0000
Byte 7	Byte6	Byte5	Byte 4		24'h00_0004
Byte 11	Byte10	Byte9	Byte 8		24'h00_0008

5.4.4 X Address Generation Unit

The X AGU is used by all instructions and supports all addressing modes. The X AGU consists of a read AGU (X RAGU) and a write AGU (X WAGU), which operate independently on separate read and write buses during different phases of the instruction cycle. The X read data bus is the return data path for all instructions that view data space as a combined X and Y address space. It is also the X address space data path for the dual operand read instructions (DSP instruction class). The X write data bus is the only write path to the combined X and Y data space for all instructions.

The X AGU supports linear addressing through all of the address space. It can therefore generate EAs within the range 0x000000 to 0xFFFFFF.

The X RAGU starts its EA calculation during the prior instruction cycle, using information derived from the just pre-fetched instruction. The X RAGU EA is presented to the address bus at the beginning of the instruction cycle.

The X WAGU starts its EA calculation at the beginning of the instruction cycle. The EA is presented to the address bus during the write phase of the instruction.

Both the X RAGU and the X WAGU support Modulo Addressing.

Bit-Reversed Addressing is supported by the X WAGU only.

5.4.5 Y Address Generation Unit

The Y data memory space has one AGU that supports data reads from the Y data memory space. The Y memory bus is never used for data writes. The function of the Y AGU and Y memory bus is to support concurrent data reads for DSP class instructions.

The Y AGU can generate an EA within the data space address range 0x0000 to 0xFFFFFF.

The Y AGU timing is identical to that of the X RAGU, in that its EA calculation starts prior to the instruction cycle, using information derived from the pre-fetched instruction. The EA is presented to the address bus at the beginning of the instruction cycle.

The Y AGU supports Modulo Addressing and Post-Modification Addressing modes for the DSP class of instructions that use it.

Note: The Y AGU does not support data writes. All data writes occur via the X WAGU to the combined X and Y data spaces. The Y AGU is only used during data reads for dual source operand DSP instructions.

5.4.6 Error Correcting Code (ECC)

To improve data memory performance and reliability, the dsPIC33AK128MC106 device family includes Error Correcting Code (ECC) functionality as an integral part of the data memory. ECC can determine the presence of single-bit errors in data memory, including which bit is in error, and correct the data without user intervention. Both X and Y memory support ECC and have their own control registers.

Each 32-bit data word in SRAM is completed by seven additional ECC bits, which are not accessible by the user.

Upon any 8/16/32-bit write in the memory, the seven ECC bits are computed and stored along with the data (the 8/16-bits writes are actually composed of an atomic read 32 bits, modify, write 32 bits).

Upon any 8/16/32-bit read in the memory, if the ECC feature is disabled, then single or double errors are not detected and are not corrected. If the ECC feature is enabled, the ECC syndrome is computed on the related 32 data bits + 7 ECC bits.

Single-bit errors are automatically identified and corrected on read back. An interrupt is generated if enabled. Double-bit errors will be identified but not corrected. Either Bus Error or Generic Error traps are generated based on read or write path double bit errors.

The user controls the ECC fault injection through the ECCxCON, ECCxFPTR, ECCxFADDR and ECCxSTAT SFRs. Users may either create intentional faults in data read from the data RAM, or in data written into the data RAM. Single or double-bit faults may be injected into any location within the data word (i.e., any data bit including the ECC parity bits).

5.4.6.1 Enabling ECC Fault Injection

The ECC fault injection logic is enabled by setting ECCxCONbits.FLTINJ = 1.

5.4.6.2 Performing Fault Injection

The following sequence should be followed to inject faults:

1. Load data RAM target address into ECCxFADDR register.

2. Select first fault bit determined by ECCxFPTRbits.FLT1PTR[5:0]. The data RAM target bit is inverted to create the fault.

3. If a double fault is desired, select second fault bit determined by ECCxFPTRbits.FLT2PTR[5:0], otherwise set ECCxFPTR.FLT2PTR[5:0] = 6'h3F.

4. Perform a read of the data RAM target address.

The ECCxSTAT is updated whenever a read occurs and the read address matches ECCxFADDR. The read address is the relative address from start address of X or Y data spaces. Expected usage of data read fault injection is to attempt to read a predefined data set with known good ECC, and inject an error immediately prior to the ECC evaluation, thereby testing the response of the ECC block under different data stimulus.

5.4.7 Data Memory (MBIST) Overview

The dsPIC33AK128MC106 family features a data memory Built-In Self-Test (MBIST) that has the option to be run at start-up or run time. The memory test checks that all memory locations are functional and provides a pass/fail status of the RAM that can be used by software to take action if needed. If a failure is reported, the specific location(s) are not identified.

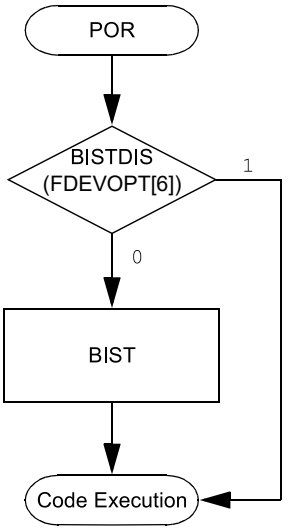
The MBISTCON register contains control and status bits for BIST operation. The MBISTDONE bit (MBISTCON[7]) indicates if a BIST was run since the last Reset and the MBISTSTAT bit (MBISTCON[4]) provides the pass fail result.

BIST will always run on FRC+PLL with PLL settings resulting in a 200 MHz clock rate.

5.4.7.1 BIST at Start-up

The BIST can be configured to automatically run on a POR-type Reset, as shown in Figure 5-3. By default, when BISTDIS (FDEVOPT[6]) = 1, the BIST is disabled and will not be part of device start-up. If the BISTDIS bit is cleared during device programming, the BIST will run after all Configuration registers have been loaded and before code execution begins.

Figure 5-3. BIST Flowchart



5.4.7.2 Fault Simulation

A mechanism is available to simulate a BIST failure to allow testing of Fault handling software. When the FLTINJ bit is set during a run-time BIST, the MBISTSTAT bit will be set regardless of the test result. The procedure for a BIST Fault simulation is as follows:

1. Set the MBISTEN bit (MBISTCON[0]).
2. Set the FLTINJ bit (MBISTCON[8]).
3. Execute a Software Reset command.
4. Verify a Software Reset has occurred by reading SWR (RCON[6]) (optional).
5. Verify the MBISTDONE, MBSITSTAT and FLTINJ bits are all set.

5.4.7.3 BIST at Run Time

The BIST can also be run at any time during code execution. Note that a BIST will corrupt all of the RAM contents, including the Stack Pointer, and requires a subsequent Reset. The system should be prepared for a Reset before a BIST is performed. The BIST is invoked by setting the MBISTEN bit (MBISTCON[0]). The MBISTCON register is protected against accidental writes and requires an unlock sequence prior to writing. Only one bit can be set per unlock sequence. The procedure for a run-time BIST is as follows:

1. Write 0x0001 to the MBISTCON SFR.
2. Execute a Software Reset command.
3. Verify a Software Reset has occurred by reading SWR (RCON[6]) (optional).
4. Verify that the MBISTDONE bit is set.
5. Take action depending on test result indicated by MBISTSTAT.

5.4.7.4 MBIST Control Register

Name: MBISTCON
Offset: 03AA0

Notes:

1. Resets only on a true POR Reset.
2. This bit will self-clear when the MBIST test is complete.

Legend: HS = Hardware Settable bit; HC = Hardware Clearable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
								FLTINJ
Access								R/W
Reset								0

Bit	7	6	5	4	3	2	1	0
	MBISTDONE			MBISTSTAT				MBISTEN
Access	R/W/HS			R				R/W/HC
Reset	0			0				0

Bit 8 – FLTINJ MBIST Fault Inject Control bit(1)

Value	Description
1	The MBIST test will complete and sets MBISTSTAT = 1, simulating an SRAM test failure
0	The MBIST test will execute normally

Bit 7 – MBISTDONE MBIST Done Status bit

Value	Description
1	An MBIST operation has been executed
0	No MBIST operation has occurred on the last Reset sequence

Bit 4 – MBISTSTAT MBIST Status bit

Value	Description
1	The last MBIST failed
0	The last MBIST passed; all memory may not have been tested

Bit 0 – MBISTEN MBIST Enable bit(2)

Value	Description
1	MBIST test is armed; an MBIST test will execute at the next device Reset
0	MBIST test is disarmed

6. Flash Program Memory

The dsPIC33A family devices contain an internal programmable Flash for storing and executing application code. The high-endurance Flash provides great flexibility in code development and storage, combining a long retention life with a high number of read/write cycles. The memory is readable, writable and erasable during normal operation over the entire V_{DD} range.

Flash program memory has the following features:

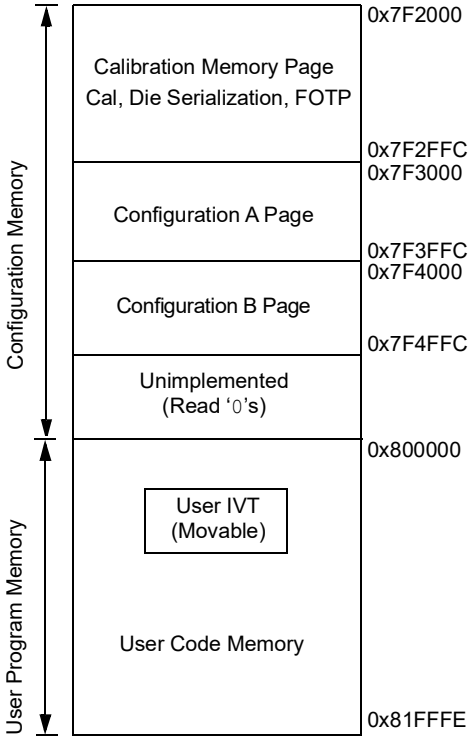
- 128-bit data Flash word programming
- Bus Mastered Row programming
- Page Erase operation
- ECC support for data read/write. Includes forced fault injection capability for use by Functional Safety focused customers.
- Trap event output for ECC double-bit errors
- Interrupt output for ECC single-bit errors
- New ECC registers to improve the customer's interface for safety relation application by improving the fault injection information
- CRC feature to verify the Flash contents

6.1 Device-Specific Information

Table 6-1. Device Memory Attributes

Device	Flash User Code Size (Kbytes)	Address Range	Pages	Page Size		Row Size		Rows Per Page
dsPIC33AK32MC10X	32	0x800000 – 0x807FFF	8	256, quad instruction words (128 bits)	1024, instruction words (32 bits)	32, quad instruction words (128 bits)	128, instruction words (32 bits)	8
dsPIC33AK64MC10X	64	0x800000 – 0x80FFFF	16					
dsPIC33AK128MC10X	128	0x800000 – 0x81FFFF	32					

Figure 6-1. dsPIC33A Program Memory Map[1,2]



Notes:

1. Memory areas are not shown to scale.
2. Refer to **"Memory Organization"** for the exact memory range of device.

6.2 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3000	NVMCON	31:24								
		23:16				WRRE		WREC[2:0]		
		15:8	WR	WREN	WRERR	NVMPIDL	Reserved[1:0]			
		7:0	LOCK	DRBV			MVMOP[3:0]			
0x3004	NVMADR	31:24								
		23:16	NVMADR[23:16]							
		15:8	NVMADR[15:8]							
		7:0	NVMADR[7:4]							
0x3008	NVMDATA0	31:24	DATA0[31:24]							
		23:16	DATA0[23:16]							
		15:8	DATA0[15:8]							
		7:0	DATA0[7:0]							
0x300C	NVMDATA1	31:24	DATA1[63:56]							
		23:16	DATA1[55:48]							
		15:8	DATA1[47:40]							
		7:0	DATA1[39:32]							
0x3010	NVMDATA2	31:24	DATA2[95:88]							
		23:16	DATA2[87:80]							
		15:8	DATA2[79:72]							
		7:0	DATA2[71:64]							
0x3014	NVMDATA3	31:24	DATA3[127:120]							
		23:16	DATA3[119:112]							
		15:8	DATA3[111:104]							
		7:0	DATA3[103:96]							
0x3018	NVMSRCADR	31:24								
		23:16	SRCADR[23:16]							
		15:8	SRCADR[15:8]							
		7:0	SRCADR[7:2]							
0x301C	NVMECCCON	31:24								
		23:16								
		15:8					ECCLOCK			
		7:0								FLTINJ
0x3020	NVMECCSTAT	31:24								
		23:16								
		15:8				ESEL			FLEC[1:0]	
		7:0			SECO	SEC			DEDO	DED
0x3024	NVMECCFPTR	31:24								
		23:16								
		15:8	FLT2PTR[7:0]							
		7:0	FLT1PTR[7:0]							
0x3028	NVMECCFADDR	31:24								
		23:16	ADDR[23:16]							
		15:8	ADDR[15:8]							
		7:0	ADDR[7:4]							
0x302C	NVMECCEADDR	31:24								
		23:16	ADDR[23:16]							
		15:8	ADDR[15:8]							
		7:0	ADDR[7:4]							
0x3030	NVMECCEDATA0	31:24	ECCEDATA0[31:24]							
		23:16	ECCEDATA0[23:16]							
		15:8	ECCEDATA0[15:8]							
		7:0	ECCEDATA0[7:0]							
0x3034	NVMECCEDATA1	31:24	ECCEDATA1[63:56]							
		23:16	ECCEDATA1[55:48]							
		15:8	ECCEDATA1[47:40]							
		7:0	ECCEDATA1[39:32]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3038	NVMECCEDATA2	31:24	ECCEDATA2[95:88]							
		23:16	ECCEDATA2[87:80]							
		15:8	ECCEDATA2[79:72]							
		7:0	ECCEDATA2[71:64]							
0x303C	NVMECCEDATA3	31:24	ECCEDATA3[127:120]							
		23:16	ECCEDATA3[119:112]							
		15:8	ECCEDATA3[111:104]							
		7:0	ECCEDATA3[103:96]							
0x3040	NVMECCVAL	31:24								
		23:16								
		15:8								ECCVAL[8]
		7:0	ECCVAL[7:0]							
0x3044	NVMECCSYND	31:24								
		23:16								
		15:8								
		7:0	ECCSYND[7:0]							
0x3048	NVMCRCCON	31:24								
		23:16	DELAY[7:0]							
		15:8	CRCEN	START					CRCIDL[1:0]	
		7:0							CRCEC[1:0]	
0x304C	NVMCRCST	31:24								
		23:16	CRCST[23:16]							
		15:8	CRCST[15:8]							
		7:0	CRCST[7:0]							
0x3050	NVMCRCEND	31:24								
		23:16	CRCEND[23:16]							
		15:8	CRCEND[15:12]				CRCEND[11:8]			
		7:0	CRCEND[7:0]							
0x3054	NVMCRCSEED	31:24	CRCSEED[31:24]							
		23:16	CRCSEED[23:16]							
		15:8	CRCSEED[15:8]							
		7:0	CRCSEED[7:0]							
0x3058	NVMCRCDATA	31:24	CRCDATA[31:24]							
		23:16	CRCDATA[23:16]							
		15:8	CRCDATA[15:8]							
		7:0	CRCDATA[7:0]							

6.2.1 Nonvolatile Memory (NVM) Control Register

Name: NVMCON
Offset: 0x3000

Notes:

1. A BOR event Reset will indirectly clear WR by forcing the FSM to terminate the operation underway. The WR bit cannot be set if the operation target address held in the NVMADRx register falls within unimplemented address space.
2. Reset occurs only on POR or BOR, but the actual initial state of P2ACTIV that is visible to the user will depend upon which panel is determined to be Active after the Reset exit.
3. A BMX address error is likely due to a bad NVMSRCADR value. The same error will generate a bus error TRAP via the interrupt controller. This will aid the software in diagnosing the issue.
4. WRERR bit will remain set if an attempt is made to execute (WR=1) a reserved PROGOP command. WR bit will not remain set.
5. "Word" is defined to be a 128-bit data value plus ECC (140 bits total). However, each word program command may consist of a sequence of fractional word programming operations.
6. Reserved when in Single Boot mode (DUAL_BOOTPRESENT=0).

Legend: C = R= Readable bit; HC = Hardware Settable bit; Clearable bit; SO = Settable Only bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
				WRRE		WREC[2:0]		
Access				R/HS/HC		R/HS/HC	R/HS/HC	R/HS/HC
Reset				0		0	0	0

Bit	15	14	13	12	11	10	9	8
	WR	WREN	WRERR	NVMPIDL	Reserved[1:0]			
Access	R/S/HC	R/W	R/C/HS	R/W	R/HC/HS	R/HC/HS		
Reset	0	0	0	0	0	0		

Bit	7	6	5	4	3	2	1	0
	LOCK	DRBV			MVMOP[3:0]			
Access	R/W	R/C/HS			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit 20 – WRRE Program/Erase Reset Event bit

Value	Description
1	Warm Reset request during program/erase operation
0	No event reported Write 0 to clear, writing 1 has no effect

Bits 18:16 – WREC[2:0] Program/Erase Error Code bit

Value	Description
101	Row programming operation not completed due to warm Reset
100	System bus error during row program operation
011	Error reported by Flash panel control logic

Value	Description
010	Security access control violation
001	Invalid program/erase operation (PROGOP)
000	No error Unused codes are reserved Read as '000' if WRERR=0

Bit 15 – WR Write Control bit(1)

Value	Description
1	Initiates a memory or fuse element program or erase operation
0	Program or erase operation is complete and inactive

Bit 14 – WREN Program/Erase Enable bit

Value	Description
1	Allows program/erase cycles
0	Inhibits programming/erasing of memory or fuse elements; This bit cannot be updated if either the LOCK bit is set or the WR bit is set

Bit 13 – WRERR Sequence Error Flag bit

Value	Description
1	Indicates an improper program or erase termination due to: • An attempt to execute a reserved PROGOP command • An error detected by the Smart Write module such as Flash failures • If a BMX address error is detected during a row programming operation This bit cannot be set by software
0	Either a POR or BOR has occurred or software cleared the WRERR bit

Bit 12 – NVMPIDL NVM Power Down in Idle Enable bit

Value	Description
1	Flash panels enter a Sleep mode (very Low-Power mode) when device enters Idle mode
0	Keep Flash and Fuse panels powered in Standby mode when device enters Idle mode

Bits 11:10 – Reserved[1:0]

Bit 7 – LOCK Lock bit

Value	Description
1	Program/erase functions are disabled until after the next Reset
0	Program/erase functions are not disabled Write 1 to set, writing 0 has no effect

Bit 6 – DRBV Data Read Buffer Valid bit

Value	Description
1	Data read buffer holds valid data
0	Data read buffer invalid Write 0 to clear, writing 1 has no effect

Bits 3:0 – MVMOP[3:0] NVM Operation Select bits(4,5,6)

Value	Description
0111-0100	Reserved
0011	The next WR command will perform a memory page erase operation

Value	Description
0010	The next WR command will perform a row program 1 operation
0001	The next WR command will perform a word program 1, 3 operation (data source: NVMDATAx)
0000	Reserved

6.2.2 NVM Address Register(1)

Name: NVMADR
Offset: 0x3004

Note:
1. This register is not writable when WR = 1.

Legend: x = Bit is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	NVMADR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NVMADR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	NVMADR[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bits 23:16 – NVMADR[23:16]

Bits 15:8 – NVMADR[15:8]

Bits 7:4 – NVMADR[7:4] NVM Address Register bits(1)
NVM Address register used to program a Flash word or Flash row or perform a page erase. During row programming, the address may point to any Flash word within the Flash row. The row programming always starts at the beginning of the Flash row. NVMADR [3:0] is hard coded to logic '0000'.

6.2.3 NVM Write Data 0 Register

Name: NVMDATA0
Offset: 0x3008
Reset: 0
Property: R/W

Notes:

1. This register is not writable when WR = 1.
2. This register is readable only when NVMCON.WREN = 0.
3. This register is not readable when NVMCON.WREN = 1. An attempted read will return (by convention) all '1's.
4. This register is cleared after a Flash write operation completes (when NVMCON.WR returns to '0').
5. This register is also mapped into user address space as SLVDATAL.

Bit	31	30	29	28	27	26	25	24
	DATA0[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA0[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA0[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA0[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA0[31:0] NVM Write Data Register(1,2,3,4,5)

6.2.4 NVM Write Data 1 Register

Name: NVMDATA1
Offset: 0x300C
Reset: 0
Property: R/W

Notes:

1. This register is not writable when WR = 1.
2. This register is readable only when NVMCON.WREN = 0.
3. This register is not readable when NVMCON.WREN = 1. An attempted read will return (by convention) all '1's.
4. This register is cleared after a Flash write operation completes (when NVMCON.WR returns to '0').
5. This register is also mapped into user address space as SLVDATAL.

Bit	31	30	29	28	27	26	25	24
	DATA1[63:56]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA1[55:48]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA1[47:40]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA1[39:32]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA1[63:32] NVM Write Data Register(1,2,3,4,5)

6.2.5 NVM Write Data 2 Register

Name: NVMDATA2
Offset: 0x3010
Reset: 0
Property: R/W

Notes:

1. This register is not writable when WR = 1.
2. This register is readable only when NVMCON.WREN = 0.
3. This register is not readable when NVMCON.WREN = 1. An attempted read will return (by convention) all '1's.
4. This register is cleared after a Flash write operation completes (when NVMCON.WR returns to '0').
5. This register is also mapped into user address space as SLVDATAL.

Bit	31	30	29	28	27	26	25	24
	DATA2[95:88]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA2[87:80]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA2[79:72]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA2[71:64]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA2[95:64] NVM Write Data Register(1,2,3,4,5)

6.2.6 NVM Write Data 3 Register

Name: NVMDATA3
Offset: 0x3014
Reset: 0
Property: R/W

Notes:

1. This register is not writable when WR = 1.
2. This register is readable only when NVMCON.WREN = 0.
3. This register is not readable when NVMCON.WREN = 1. An attempted read will return (by convention) all '1's.
4. This register is cleared after a Flash write operation completes (when NVMCON.WR returns to '0').
5. This register is also mapped into user address space as SLVDATAL.

Bit	31	30	29	28	27	26	25	24
	DATA3[127:120]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA3[119:112]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA3[111:104]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA3[103:96]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA3[127:96] NVM Write Data Register(1,2,3,4,5)

6.2.7 NVM Source Data Address Register(1,2)

Name: NVMSRCADR
Offset: 0x3018

Notes:

1. This register is not writable when WR = 1.
2. This address must be aligned to a RAM word address (word-aligned).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	SRCADR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SRCADR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SRCADR[7:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bits 23:16 – SRCADR[23:16] RAM Base Address register for row programming
The address is always on 32-bit word boundaries.

Bits 15:8 – SRCADR[15:8] RAM Base Address register for row programming
The address is always on 32-bit word boundaries.

Bits 7:2 – SRCADR[7:2] RAM Base Address register for row programming
The address is always on 32-bit word boundaries.

6.2.8 NVM ECC Control Register

Name: NVMECCCON
Offset: 0x301C

Notes:

1. The LOCK bit is settable by software and can only be cleared via a Reset.
2. If the ECCLOCK=1, then the FLTINJ bit is a don't care and the fault injection function is disabled.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					ECCLOCK			
Access					R/W			
Reset					0			

Bit	7	6	5	4	3	2	1	0
								FLTINJ
Access								R/W
Reset								0

Bit 11 – ECCLOCK ECC Fault Inject LOCK bit(1)

Value	Description
1	ECC fault injection function is disabled
0	ECC fault injection functionality is enabled

Bit 0 – FLTINJ Fault Injection Sequence Enable bit(2)

Value	Description
1	Enable at fault injection to occur on NVM address match with ECCFADR
0	Fault injection is disabled

6.2.9 NVM ECC Status Register

Name: NVMECCSTAT
Offset: 0x3020

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				ESEL			FLEC[1:0]	
Access				R/W			R	R
Reset				0			0	0

Bit	7	6	5	4	3	2	1	0
			SECO	SEC			DEDO	DED
Access			R/C/HS	R/C/HS			R/C/HS	R/C/HS
Reset			0	0			0	0

Bit 12 – ESEL Error Reporting Select bit

Value	Description
1	Select SEC information for ECCEADDR, ECCEDATAx, ECCVAL, ECCSYN registers
0	Select DED information for ECCEADDR, ECCEDATAx, ECCVAL, ECCSYN registers

Bits 9:8 – FLEC[1:0] Fuse Load Error Code

Value	Description
11	One or more DED errors; One or more default fuse values used
10	One or more DED errors; No default fuse values used
01	One or more SEC errors; No DED errors
00	No ECC bit errors

Bit 5 – SECO SEC Event Overflow bit

Value	Description
1	SEC error not captured due to overflow
0	No SEC error overflow reported

Bit 4 – SEC SEC Event Reported bit

Value	Description
1	Single bit error reported
0	Single bit error not reported

Bit 1 – DEDO DED Event Overflow bit

Value	Description
1	DED error not captured due to overflow
0	No DED error overflow reported

Bit 0 – DED DED Event Reported bit

Value	Description
1	Double bit error reported
0	Double bit error not reported

6.2.10 NVM ECC Fault Injection Pointer Register

Name: NVMECCFPTR
Offset: 0x3024

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	FLT2PTR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FLT1PTR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – FLT2PTR[7:0] ECC Fault Injection Bit Pointer 2

Value	Description
11111111–10001001	No fault injection occurs
10001000	Fault injection (bit inversion) occurs on bit 136 of ECC bit order
...	
00000001	Fault injection (bit inversion) occurs on bit 1 of ECC bit order
00000000	Fault injection (bit inversion) occurs on bit 0 of ECC bit order

Bits 7:0 – FLT1PTR[7:0] ECC Fault Injection Bit Pointer 1

Value	Description
11111111–10001001	No fault injection occurs
10001000	Fault injection (bit inversion) occurs on bit 136 of ECC bit order
...	
00000001	Fault injection (bit inversion) occurs on bit 1 of ECC bit order
00000000	Fault injection (bit inversion) occurs on bit 0 of ECC bit order

6.2.11 NVM ECC Fault Injection Address Register

Name: NVMECCFADDR
Offset: 0x3028

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	ADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bits 23:16 – ADDR[23:16]

Bits 15:8 – ADDR[15:8]

Bits 7:4 – ADDR[7:4] ECC Fault Injection Address bits
Address of targeted Flash word for fault injection

6.2.12 NVM ECC Error Address Register

Name: NVMECCEADDR
Offset: 0x302C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	ADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bits 23:16 – ADDR[23:16]

Bits 15:8 – ADDR[15:8]

Bits 7:4 – ADDR[7:4] ECC Address of Read Data
These bits register the location of the NVM read data when any of the following bits are set high: ECCSTAT.SEC or ECCSTAT.DED.

6.2.13 NVM ECC Error Data 0 Register

Name: NVMECCEDATA0
Offset: 0x3030

Legend: SO = Settable Only bit

Bit	31	30	29	28	27	26	25	24
	ECCEDATA0[31:24]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ECCEDATA0[23:16]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ECCEDATA0[15:8]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ECCEDATA0[7:0]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ECCEDATA0[31:0] NVM Read Error Data bits
These bits register the NVM read data (taking fault injections into account) when the SEC or DED bit is set in the NVMECCSTAT register.

6.2.14 NVM ECC Error Data 1 Register

Name: NVMECCEDATA1
Offset: 0x3034

Legend: SO = Settable Only bit

Bit	31	30	29	28	27	26	25	24
	ECCEDATA1[63:56]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ECCEDATA1[55:48]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ECCEDATA1[47:40]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ECCEDATA1[39:32]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ECCEDATA1[63:32] NVM Read Error Data bits
These bits register the NVM read data (taking fault injections into account) when the SEC or DED bit is set in the NVMECCSTAT register.

6.2.15 NVM ECC Error Data 2 Register

Name: NVMECCEDATA2
Offset: 0x3038

Legend: SO = Settable Only bit

Bit	31	30	29	28	27	26	25	24
	ECCEDATA2[95:88]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ECCEDATA2[87:80]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ECCEDATA2[79:72]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ECCEDATA2[71:64]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ECCEDATA2[95:64] NVM Read Error Data bits
These bits register the NVM read data (taking fault injections into account) when the SEC or DED bit is set in the NVMECCSTAT register.

6.2.16 NVM ECC Error Data 3 Register

Name: NVMECCEDATA3
Offset: 0x303C

Legend: SO = Settable Only bit

Bit	31	30	29	28	27	26	25	24
	ECCEDATA3[127:120]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ECCEDATA3[119:112]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ECCEDATA3[111:104]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ECCEDATA3[103:96]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ECCEDATA3[127:96] NVM Read Error Data bits
These bits register the NVM read data (taking fault injections into account) when any of the ECCSTAT. SEC or ECCSTAT. DED bits are set high.

6.2.17 NVM ECC Value Register

Name: NVMECCVAL
Offset: 0x3040

Legend: SO = Settable Only bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
								ECCVAL[8]
Access								R/HS
Reset								0

Bit	7	6	5	4	3	2	1	0
	ECCVAL[7:0]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bits 8:0 – ECCVAL[8:0] Error Correcting Code for the Data bits
These bits register the Error Correcting Code associated with the NVM read data (taking fault injections into account) when the SEC or DED bit is set in the NVMECCSTAT register.

6.2.18 NVM ECC Syndrome Register

Name: NVMECCSYND
Offset: 0x3044

Note:

1. The syndrome bit encoding indicates the Hamming code bit locations. This value is meaningful in single bit detection only.

Legend: SO = Settable Only bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	ECCSYND[7:0]							
Access	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS	R/HS
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – ECCSYND[7:0] Syndrome Value bits(1)
These bits register the syndrome value associated with the NVM read data (taking fault injections into account) when the SEC or DED bit is set in the NVMECCSTAT register.

6.2.19 NVM CRC Control Register

Name: NVMCRCCON
Offset: 0x3048

Legend: C = Clearable bit; SO = Settable Only bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	DELAY[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCEN	START					CRCIDL[1:0]	
Access	R/W	R/W					R/W	R/W
Reset	0	0					0	0

Bit	7	6	5	4	3	2	1	0
							CRCEC[1:0]	
Access							R/HS/HC	R/HS/HC
Reset							0	0

Bits 23:16 – DELAY[7:0] Delay Between CRC Accesses bits
Delay count in system clock cycles between CRC accesses

Bit 15 – CRCEN Enable CRC Function bit

Value	Description
1	CRC function enabled
0	CRC function disabled

Bit 14 – START Start CRC Calculation bit

Value	Description
1	Start CRC calculation (CRC in progress)
0	CRC calculation complete (CRC function idle)

Bits 9:8 – CRCIDL[1:0] Idle Operation Control bits

Value	Description
11	Reserved
10	CRC operates in Idle mode and is stopped in CPU Run mode
01	CRC operates in CPU Run mode and is stopped in Idle mode
00	CRC operates in CPU Run mode and Idle mode

Bits 1:0 – CRCEC[1:0] CRC Error Code bits

Value	Description
11	Invalid address (start address greater than end address)
10	Flash ECC DED error
01	Security access control violation

Value	Description
00	No error

6.2.20 NVM CRC Start Address Register

Name: NVMCRCST
Offset: 0x304C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	CRCST[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCST[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CRCST[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – CRCST[23:0] Start Address Offset LSb
CRC start address (least significant bits), fixed value 0x000

6.2.21 NVM CRC End Address Register

Name: NVMCRCEND
Offset: 0x3050

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	CRCEND[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCEND[15:12]				CRCEND[11:8]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	CRCEND[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bits 23:16 – CRCEND[23:16]

Bits 15:12 – CRCEND[15:12]

Bits 11:0 – CRCEND[11:0] End Address LSb
CRC end address (least significant bits) fixed value 0xFFF

6.2.22 NVM CRC Seed Register

Name: NVMCRCSEED
Offset: 0x3054

Bit	31	30	29	28	27	26	25	24
	CRCSEED[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CRCSEED[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCSEED[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CRCSEED[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CRCSEED[31:0] 32-Bit Seed Value for CRC Calculation bit

6.2.23 NVM CRC Output Data Register

Name: NVMCRCDATA
Offset: 0x3058

Bit	31	30	29	28	27	26	25	24
	CRCDATA[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CRCDATA[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCDATA[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CRCDATA[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CRCDATA[31:0] NVM CRC Data Output bit

6.3 Operation

6.3.1 Reading Program Memory

Program memory is read linearly, similar to how data memory is read. The `MOV` instruction is used to read the program memory.

Previously, dsPIC33 devices used the Table Read feature to read the Flash program memory. However, in dsPIC33A devices, TBLRDL/TBLRDH features are unimplemented.

6.3.2 Programming Memory Writes

There are two methods that the user application can use to program Flash memory:

- Run-Time Self-Programming (RTSP)
- In-Circuit Serial Programming™ (ICSP™)

6.3.2.1 Run-Time Self-Programming (RTSP)

RTSP allows the user application to modify Flash program memory contents. The program/erase operations use NVMCON, NVMADR, NVMSRCADR, and NVMDATA (0-3) registers to modify the Flash. With RTSP, the user application can erase a single page of Flash memory and program either a word or row of Flash memory.

Flash is programmed using either a Word Program (128-bits) or Row Program (512 bytes) operation. Flash must be erased before programming. While programming operations are very reliable, best practice is to verify Flash contents after programming. This can be done using either the integrated CRC function or with a direct comparison of programmed Flash with the source data. A 128-bit Flash word is the smallest unit of Flash memory that can be programmed. The ECC parity bits associated with each Flash word are programmed automatically for both word and row

programming. Programing a Flash word more than once will generally result in the word having incorrect ECC parity bit values.

All erase and program operations using the classic dsPIC33 PROGOP interface must check the status of the WR bit and the WRERR bit, to determine the completion of the requested erase or programming operation. Alternatively, software can use an interrupt to monitor completion of the programming operation. A completion interrupt is generated if the operation completed successfully or if the operation was terminated by an error. Any outdated software that made assumptions using fixed time periods for programming operations must be reviewed and updated if applicable.

6.3.2.1.1 Status Bits

Program Flash memory has two status bits in the NVMCON register to check the status of the NVM operation initiated by the user: WRERR bit and URERR bit. These bits ensure that the programming sequence has completed successfully.

WRERR Error Flag Bit

When a program or erase operation is initiated using the NVMCON.WR bit, the NVMCON.WRERR bit will indicate if the operation was properly completed. The user may check the state of the WRERR bit at any time after WR is set. Typically, the application will wait for WR to be cleared (indicating the program or erase sequence has completed), then check WRERR to confirm if the sequence completed successfully or not.

If WRERR = 0 at that time, the operation completed successfully. If WRERR = 1 at that time, the operation did not complete successfully. The WRERR bit is cleared on a cold Reset and should be cleared by firmware before initiating a program/erase operation.

WREC Bit

The WREC field indicates the error type when the WRERR bit is set. Program/erase operation errors include:

- Row programming not completed due to warm Reset
- System bus error during row program operation
- Error reported by flash panel control logic
- Security access control violation
- Invalid program/erase operation (PROGOP)
- WREC is cleared on a cold Reset and when WRERR=0.

WRRE Bit

The WRRE bit indicates if there was a warm Reset (6.3.3.1.2. Implications of Device Reset) request during a program/erase operation.

The PROGOP, WRERR, WREC and WRRE fields are not cleared on a warm Reset. This provides the error status for a program/erase operation initiated before the warm Reset.

URERR Error Flag Bit

When URERR= 1, status bit indicates:

1. A BOR terminates a program/erase op.
2. $\overline{\text{MCLR}}$ terminates row programming.
3. A BMX error during row programing

Note: A BMX error during row programming sets both URERR = 1 and WRERR = 1.

6.3.2.1.2 LOCK Bit

The dsPIC33A devices have a “one-way” LOCK feature implemented through its NVMCON register. Once software sets the LOCK bit, only a reset will clear the LOCK bit. The LOCK bit, if set, prevents all

erase and programming operations. If the LOCK bit is set while an erase or programming operation is in progress, the operation can continue to completion.

A “one-way” ECCLOCK has been implemented in the ECCCON register. Once the software sets the ECCLOCK bit, only a Reset will clear the ECCLOCK bit. If the ECCLOCK bit is set, the fault injection feature is disabled.

6.3.2.2 Flash Programming Operations

A program or erase operation is necessary for programming or erasing the internal Flash program memory. The following operations can be used to modify the Flash contents.

1. Word program
2. Row program
3. Page erase

Setting the WR bit (NVMCON[15]) starts the operation. The WR bit is automatically cleared when the operation is finished.

The CPU stalls until the programming operation is finished. The CPU will not execute any instructions or respond to interrupts during this time. If any interrupts occur during the programming cycle, they will remain pending until the cycle completes.

Note: In the event of any Reset other than a POR or BOR, the erase or program operation underway can complete while the device resets.

6.3.2.2.1 Flash Page Erase

A page erase is necessary before programming the Flash memory. A page erase will ensure that the old contents of the page are erased and that it is programmed with all ‘1s before programming with new content. Prior to a page erase operation, the target page is selected by writing an address within the page into the Write Address register, NVMADR. A page erase will erase eight rows of data concurrently.

User Page Erase Sequence

Executing a page erase sequence typically includes the following user code steps, subject to the constraints externally imposed by the security module:

1. Load NVMADR with an address within the page to be erased.
2. Configure NVMCON to page erase:
   - WREN = 1
   - PROGOP = page erase (see 6.2.1. NVMCON)
3. Set the WR bit.
4. The program sequence completes and the WR bit is cleared by hardware.
5. Clear WREN bit.
6. Test the WRERR bit to ensure the erase sequence completed successfully.

6.3.2.2.2 Word Programming

The smallest block of data that can be programmed by the user is one Flash word (128 bits+ECC). The data to be word programmed must be loaded into NVMDATAx before the word programming sequence is initiated. For the word programming PROGOP, the Flash word at the location pointed to by NVMADR is programmed. Note that NVMADR is a word address, so NVMADR [3:0] is defined as 4’b0000 to force Flash word alignment of the operation. To keep the contents secure, the NVMDATAx register is cleared upon completion of any Flash write operation.

User Word Programming Sequence

A word programming sequence typically includes the following steps, subject to the constraints externally imposed by the security module:

1. Write data to be programmed to NVMDATAx.
2. Load NVMADR with the Flash address to be programmed.
3. Configure NVMCON to word program:
   - WREN = 1
   - PROGOP = Word program (see 6.2.1. NVMCON)
4. Set the WR bit.
5. The program sequence completes and the WR bit is cleared by hardware.
6. Clear WREN bit.
7. Test the WRERR bit to ensure the program sequence completed successfully.

Note: Writes to NVMDATAx registers are inhibited while WR = 1.

6.3.2.2.3 Bus Mastered Row Programming

In dsPIC33A devices, row programming is performed directly from a buffer space in data RAM. The location of the RAM buffer is determined by the NVMSRCADR register(s), which are loaded with the data RAM address containing the first word of program data to be written. Prior to performing the program operation, the buffer space in RAM must be loaded with the row of data to be programmed.

One row consists of 32 Flash words (128 bit+ECC). Row programming starts at the lowest address in the row and proceeds up to the highest address in the row. The module is not specified to operate when the operation is somewhere in the middle of the row and then wraps around.

The row program command automatically sequences through a row worth of write data stored (by the user) in local memory. This module implements row programming operations as a sequence of 32, 140-bit word programming sequences.

During a row programming operation, the Flash controller becomes the bus initiator and system RAM is used to store the row programming data. The user loads the system RAM prior to the program operation. The user loads the NVMSRCADR address register with the location of the data in RAM. This address is word aligned to 32-bit RAM addresses. The NVMADR address must be aligned to a Flash word address boundary where (address [3:0] = 4'b0000). During row programming, the NVMADR address may point to any Flash word within the Flash row.

BMX Bus Error During Row Programming

If the NVMSRCADR value is invalid, and the NVM is performing a row programming operation, the BMX (Bus Matrix) will not be able to connect to the RAM containing the needed data. The BMX will generate an error if it cannot access the address specified by the NVMSRCADR register. This error triggers the NVM to terminate the row programming operation and set the WRERR and the URERR status bits. The NVM Module will generate a NVM Trap signal to the Interrupt Controller to alert the user software that a bus error has occurred.

User Row Programming Sequence

A row programming sequence typically includes the following steps, subject to the constraints externally imposed by the security module:

1. Load NVMSRCADR with the address of data in device RAM.
2. Load NVMADR with the Flash address to be programmed.
3. Configure NVMCON to row program:
   - WREN =1
   - PROGOP = 0x2
4. Set the WR bit.
5. The program sequence completes and the WR bit is cleared by hardware.

6. Clear WREN bit to avoid accidental writes.
7. Test the WRERR bit to ensure the program sequence completed successfully.

Note: Writes to RAM used during bus mastered row programming cannot be protected. Consequently, the user must be aware that inadvertent corruption of this area of RAM is possible during programming.

6.3.3 Flash ECC

To improve program memory performance and durability, select dsPIC33A devices include ECC functionality as an integral part of the Flash memory controller. ECC can determine the presence of single-bit errors in program data, including which bit is in error, and correct the data without user intervention. When implemented, ECC is automatic and cannot be disabled.

Additionally, users can inject single-bit errors or double-bit errors into the ECC calculation. This allows Functional Safety focused users to be able to test the ECC calculation and fault generation logic at run time to verify there are no latent faults present in the system.

When data is written to program memory, ECC generates a 9-bit Hamming code parity value, 8-bit Single Error Correction (SEC) value and one extra bit to support Double Error Correction (DED) for every 128-bit input data value. Parity data is not memory-mapped and is inaccessible. When the data is read back, the ECC calculates parity on it and compares it to the previously stored parity value. If a parity mismatch occurs, there are two possible outcomes:

- Single-bit errors are automatically identified and corrected on read back. An optional device-level interrupt is also generated.
- Double-bit errors will generate a soft trap. If special exception handling for the trap is not implemented, a device Reset will also occur.

The user controls the ECC fault injection through the ECCCON, ECCFPTR, ECCFADDR and the ECCSTAT SFRs. In keeping with all other NVM Controller SFRs, writes to these registers are inhibited during any Flash write sequence. Users may either create intentional faults in data read from the Flash or in data written into the Flash. Single-bit or double-bit faults may be injected into any location within the data word (i.e., any data bit but also including the ECC parity bits).

6.3.3.1 Enabling ECC Fault Injection

The ECC fault injection logic is enabled by setting EECCON.FLTINJ = 1. To help avoid unintentional enabling, the EECCON.ECCLOCK bit can be set to disable all fault injection operations. The ECCLOCK bit is only cleared via a Reset.

6.3.3.1.1 Performing Fault Injection

The following sequence should be followed to inject faults:

1. Load Flash target address into ECCFADDR register.
2. Select first fault bit determined by ECCCON.FLT1PTR [7:0]. The target bit is inverted to create the fault.
3. If a double fault is desired, select the second fault bit determined by ECCCON.FLT2PTR [7:0], otherwise set ECCCON.FLT2PTR [7:0] = 8'hFF.
4. Perform a read of the Flash target address.

The ECCSTAT is updated whenever a read occurs and the read address matches ECCFADDR. The expected usage of a data read fault injection is to attempt to read a predefined data set with known good ECC and inject an error immediately prior to the ECC evaluation, thereby testing the response of the ECC block under different data stimuli.

6.3.3.1.2 Implications of Device Reset

Cold Reset

A loss of power (Cold Reset) during a program/erase operation immediately terminates the operation. In this case, the targeted Flash may be partially programmed/erased and reads of the affected Flash may return invalid data or generate ECC errors. There is no status register indication that a program or erase was terminated by a Cold Reset. This needs to be determined based on the state of Flash. The NVM Controller CRC function can be used to evaluate the integrity of Flash after a Cold Reset.

Warm Reset

If a Warm Reset request occurs during a word program or page erase operation, the operation continues with the rest of the device held in Reset until the operation completes. The row programming operation cannot be completed due to its use of the RAM, which is not accessible in Reset. The WRRE bit is set when a Warm Reset request occurs during a program/erase operation.

The fault injection capability of the ECC macro is a customer accessible feature intended for use only at run-time. It is not possible to inject errors when the Flash based fuses are read (i.e., during the Reset sequence). Should a Reset (other than BOR or POR) occur during a fault injection write, the write will be allowed to complete just like any other Flash write. Should the Reset be due to a BOR or POR, the write will have been aborted because power was removed from the panel. Flash Controller and ECC fault injection registers related to Flash writes are subject only to POR. Unless the Reset was a POR, the user may check the state of NVMCON.WRERR to determine if a write had failed prior to Reset.

6.3.4 NVM CRC Function

In dsPIC33A devices, NVM includes a CRC function that generates a 32-bit CRC value on the contents of the Flash memory. If desired, use the CRC feature to verify the Flash contents with improved Flash data protection.

6.3.4.1 Usage

The user programs the start and end address of the Flash memory to be CRC'ed into CRCST and CRCEND, respectively. The CRC module operates on Flash page boundaries (4 KB). The user then sets the CRCON bit in the CRCCON register to start the CRC calculation. When the CRC sequence has completed, and the calculated CRC data is available, the DONE bit is asserted in the CRCCON register, and interrupt is generated. To begin another CRC calculation, the CRCON bit must be cleared and then asserted after the CRCST and CRCEND registers are loaded with the desired start and end addresses.

6.4 Application Example

Example 6-1. Word-Write

```
long DataWord1;
long DataWord2;
long DataWord3;
long DataWord4;

long TargetWriteAddress;

 NVMCONbits.NVMOP = 1;
 NVMCONbits.WREN = 1;
 NVMADR =  TargetWriteAddress ;
 NVMDATA0 =  DataWord1;
 NVMDATA1 =  DataWord2;
 NVMDATA2 =  DataWord3;
 NVMDATA3 =  DataWord4;

 NVMCONbits.WR = 1;
 while (NVMCONbits.WR == 1);
```

7. Configuration Bits

In dsPIC33AK128MC106 family devices, the Configuration Words are implemented as volatile memory. This means that configuration data will get loaded to volatile memory (from the Flash Configuration Words) each time the device is powered up. There are two types of regions, CFGA for general purpose, and CFGB for security. Both regions also have a backup copy and are summarized in Table 7-1. The configuration data are automatically loaded from the Flash Configuration Words to the proper Configuration Shadow registers during device Resets. The locations of the CFGA and CFGB registers are shown in Table 7-2.

In the dsPIC33AK128MC106 family, the configuration and calibration data is duplicated in Flash for improved robustness. If a double ECC error is detected during the start-up sequence, the device will restart the loading of the calibration and configuration from the backup set in Flash. The backup addresses are shown in Table 7-2.

To maintain the integrity of the stored configuration values, logic performs on-going bit value checks. All device configuration bits are implemented as a complementary set of register bits.

Table 7-1. Configuration Regions

Region	Configuration Regions
CFGA	0x7F3000 - 0x7F3800
CFGA Backup	0x7F3800 - 0x7F4000
CFGB	0x7F4000 - 0x7F4800
CFGB Backup	0x7F4800 - 0x7F5000

Table 7-2. dsPIC33AK128MC106 Configuration Addresses

Register Name	Address	Register Name	Address
FCP	0x7F3000	FPR4ST	0x7F4044
FICD	0x7F3010	FPR4END	0x7F4048
FDEVOPT1	0x7F3020	FPR5CTRL	0x7F4050
FWDT	0x7F3030	FPR5ST	0x7F4054
FPR0CTRL	0x7F4000	FPR5END	0x7F4058
FPR0ST	0x7F4004	FPR6CTRL	0x7F4060
FPR0END	0x7F4008	FPR6ST	0x7F4064
FPR1CTRL	0x7F4010	FPR6END	0x7F4068
FPR1ST	0x7F4014	FPR7CTRL	0x7F4070
FPR1END	0x7F4018	FPR7ST	0x7F4074
FPR2CTRL	0x7F4020	FPR7END	0x7F4078
FPR2ST	0x7F4024	FIRT	0x7F4080
FPR2END	0x7F4028	FSECDBG	0x7F4090
FPR3CTRL	0x7F4030	FPED	0x7F40A0
FPR3ST	0x7F4034	FEPUCB	0x7F40B0
FPR3END	0x7F4038	FWPUCB	0x7F40C0
FPR4CTRL	0x7F4040		

Table 7-3. dsPIC33AK128MC106 Backup Configuration Addresses

Register Name	Address	Register Name	Address
FCPBKUP	0x7F3800	FPR4STBKUP	0x7F4844
FICDBKUP	0x7F3810	FPR4ENDBKUP	0x7F4848
FDEVOPT1BKUP	0x7F3820	FPR5CTRLBKUP	0x7F4850
FWDTBKUP	0x7F3830	FPR5STBKUP	0x7F4854

.....continued			
Register Name	Address	Register Name	Address
FPR0CTRLBKUP	0x7F4800	FPR5ENDBKUP	0x7F4858
FPR0STBKUP	0x7F4804	FPR6CTRLBKUP	0x7F4860
FPR0ENDBKUP	0x7F4808	FPR6STBKUP	0x7F4864
FPR1CTRLBKUP	0x7F4810	FPR6ENDBKUP	0x7F4868
FPR1STBKUP	0x7F4814	FPR7CTRLBKUP	0x7F4870
FPR1ENDBKUP	0x7F4818	FPR7STBKUP	0x7F4874
FPR2CTRLBKUP	0x7F4820	FPR7ENDBKUP	0x7F4878
FPR2STBKUP	0x7F4824	FIRTBKUP	0x7F4880
FPR2ENDBKUP	0x7F4828	FSECDBGBKUP	0x7F4890
FPR3CTRLBKUP	0x7F4830	FPEDBKUP	0x7F48A0
FPR3STBKUP	0x7F4834	FEPUCBBKUP	0x7F48B0
FPR3ENDBKUP	0x7F4838	FWPUCBBKUP	0x7F48C0
FPR4CTRLBKUP	0x7F4840		

Table 7-4. Device ID and Revision Addresses

Register	Address
DEVID	0x7C2000
DEVREV	0x7C0004

Table 7-5. Family Device Identifier

Device	Device ID Value	JTAG ID Value
dsPIC33AK32MC102	0x9D00	0x09D00053
dsPIC33AK32MC103	0x9D01	0x09D01053
dsPIC33AK32MC105	0x9D02	0x09D02053
dsPIC33AK32MC106	0x9D03	0x09D03053
dsPIC33AK64MC102	0x9D10	0x09D10053
dsPIC33AK64MC103	0x9D11	0x09D11053
dsPIC33AK64MC105	0x9D12	0x09D12053
dsPIC33AK64MC106	0x9D13	0x09D13053
dsPIC33AK128MC102	0x9D20	0x09D20053
dsPIC33AK128MC103	0x9D21	0x09D21053
dsPIC33AK128MC105	0x9D22	0x09D22053
dsPIC33AK128MC106	0x9D23	0x09D23053

Note: The register summary covers CGA and CFGB areas. The backup copies in CFGA Backup and CFGB Backup are identical and not listed here.

7.1 Configuration Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7F3000	FCP	31:24								
		23:16								
		15:8								
		7:0					WPUCA[1:0]		$\overline{\text{CRC}}$	CP
0x7F3004 ... 0x7F300F	Reserved									
0x7F3010	FICD	31:24								
		23:16								
		15:8								
		7:0	Reserved		JTAGEN					
0x7F3014 ... 0x7F301F	Reserved									
0x7F3020	FDEVOPT1	31:24								
		23:16								
		15:8			SPI2PIN					
		7:0		BISTDIS		ALTI2C2	ALTI2C1			
0x7F3024 ... 0x7F302F	Reserved									
0x7F3030	FWDT	31:24								
		23:16								WDTRSTEN
		15:8	WDTEN	WDTWIN[1:0]		RWDTPS[4:0]				
		7:0	RCLKSEL[1:0]		SWDTPS[4:0]					WINDIS
0x7F3034 ... 0x7F3FFF	Reserved									
0x7F4000	FPR0CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x7F4004	FPR0ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x7F4008	FPR0END	31:24								
		23:16		END[22:16]						
		15:8	END[15:8]							
		7:0	END[7:0]							
0x7F400C ... 0x7F400F	Reserved									
0x7F4010	FPR1CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x7F4014	FPR1ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x7F4018	FPR1END	31:24								
		23:16		END[22:16]						
		15:8	END[15:8]							
		7:0	END[7:0]							
0x7F401C ... 0x7F401F	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7F4020	FPR2CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x7F4024	FPR2ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x7F4028	FPR2END	31:24								
		23:16		END[22:16]						
		15:8	END[15:8]							
		7:0	END[7:0]							
0x7F402C ... 0x7F402F	Reserved									
0x7F4030	FPR3CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x7F4034	FPR3ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x7F4038	FPR3END	31:24								
		23:16		END[22:16]						
		15:8	END[15:8]							
		7:0	END[7:0]							
0x7F403C ... 0x7F403F	Reserved									
0x7F4040	FPR4CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x7F4044	FPR4ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x7F4048	FPR4END	31:24								
		23:16		END[22:16]						
		15:8	END[15:8]							
		7:0	END[7:0]							
0x7F404C ... 0x7F404F	Reserved									
0x7F4050	FPR5CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x7F4054	FPR5ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x7F4058	FPR5END	31:24								
		23:16		END[22:16]						
		15:8	END[15:8]							
		7:0	END[7:0]							
0x7F405C ... 0x7F405F	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7F4060	FPR6CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x7F4064	FPR6ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x7F4068	FPR6END	31:24								
		23:16		END[22:16]						
		15:8	END[15:8]							
		7:0	END[7:0]							
0x7F406C ... 0x7F4073	Reserved									
0x7F4074	FPR7ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x7F4078	FPR7END	31:24								
		23:16		END[22:16]						
		15:8	END[15:8]							
		7:0	END[7:0]							
0x7F407C ... 0x7F407F	Reserved									
0x7F4080	FIRT	31:24								
		23:16								
		15:8								
		7:0								IRT
0x7F4084 ... 0x7F408F	Reserved									
0x7F4090	FSECDBG	31:24								
		23:16								
		15:8								
		7:0								SECDBG
0x7F4094 ... 0x7F409F	Reserved									
0x7F40A0	FPED	31:24								
		23:16								
		15:8								
		7:0								ICSPPED
0x7F40A4 ... 0x7F40AF	Reserved									
0x7F40B0	FEPUCB	31:24	FEPUCB[31:24]							
		23:16	FEPUCB[23:16]							
		15:8	FEPUCB[15:8]							
		7:0	FEPUCB[7:0]							
0x7F40B4 ... 0x7F40BF	Reserved									
0x7F40C0	FWPUCB	31:24	FWPUCB[31:24]							
		23:16	FWPUCB[23:16]							
		15:8	FWPUCB[15:8]							
		7:0	FWPUCB[7:0]							

7.1.1 FCP Configuration Register

Name: FCP
Offset: 0x7F3000

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					WPUCA[1:0]		$\overline{CRC}$	CP
Access					R/W	R/W	R/W	R/W
Reset					1	1	1	1

Bits 3:2 – WPUCA[1:0] Write Protect bits

Value	Description
11	UCA1 and UCA2 not write protected
10	Reserved
01	UCA1 and UCA2 write protected
00	Reserved

Bit 1 – $\overline{CRC}$ Code Protect User Program CRC bit

Value	Description
1	CRC is disallowed in test modes when code protection is enabled
0	CRC is allowed in test modes when code protection is enabled

Bit 0 – CP Code Protect Enable bit

Value	Description
1	Code protection disabled
0	Code protection is enabled

7.1.2 FICD Configuration Register

Name: FICD
Offset: 0x7F3010

Legend:R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	Reserved		JTAGEN					
Access	R/W		R/W					
Reset	1		0					

Bit 7 – Reserved Reserved: Maintain as '1'

Bit 5 – JTAGEN JTAG Enable bit

Value	Description
1	JTAG port is enabled
0	JTAG port is disabled

7.1.3 FDEVOPT Configuration Register

Name: FDEVOPT1
Offset: 0x7F3020

Note:

1. Fixed pin option is only available for 64-pin packages.

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			SPI2PIN					
Access			R/W					
Reset			1					

Bit	7	6	5	4	3	2	1	0
		BISTDIS		ALTI2C2	ALTI2C1			
Access		R/W		R/W	R/W			
Reset		1		1	1			

Bit 13 – SPI2PIN SPI 2 Fast I/O Pad Disable bit(1)

Value	Description
1	SPI2 uses PPS (IO Remap) to make connects with device pins
0	SPI2 uses direct connections with specified device pins

Bit 6 – BISTDIS Memory BIST Feature Disable bit

Value	Description
1	mBIST on Reset feature disabled
0	mBIST on Reset feature enabled

Bit 4 – ALTI2C2 Alternate I2C2 Pin Mapping bit

Value	Description
1	Default location for SCL2/SDA2 pins
0	Alternate location for SCL2/SDA2 pins (ASCL2/ASDA2)

Bit 3 – ALTI2C1 Alternate I2C1 Pin Mapping bit

Value	Description
1	Default location for SCL1/SDA1 pins
0	Alternate location for SCL1/SDA1 pins (ASCL1/ASDA1)

7.1.4 FWDT Configuration Register

Name: FWDT
Offset: 0x0x7F3030

Legend:R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
								WDTRSTEN
Access								R/W
Reset								1

Bit	15	14	13	12	11	10	9	8
	WDTEN	WDTWIN[1:0]		RWDTPS[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	1	0	0	1	0

Bit	7	6	5	4	3	2	1	0
	RCLKSEL[1:0]		SWDTPS[4:0]					WINDIS
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	1	0	1	0	0

Bit 16 – WDTRSTEN WDT Reset Enable bit

Value	Description
1	Enables WDT Reset; Run time WDT event will cause a device Reset
0	Disables WDT Reset; Run time WDT event generates a Trap

Bit 15 – WDTEN Watchdog Timer Enable bit

Value	Description
1	WDT is enabled in hardware
0	WDT controlled via the ON bit (WDTCONL[15])

Bits 14:13 – WDTWIN[1:0] Watchdog Timer Window Select bits

Value	Description
11	WDT window is 25% of the WDT period
10	WDT window is 37.5% of the WDT period
01	WDT window is 50% of the WDT period
00	WDT Window is 75% of the WDT period

Bits 12:8 – RWDTPS[4:0] Run Mode Watchdog Timer Period Select bits

Value	Description
11111	Divide by 2 ^ 31 = 2,147,483,648
11110	Divide by 2 ^ 30 = 1,073,741,824
...	
00001	Divide by 2 ^ 1 = 2
00000	Divide by 2 ^ 0 = 1

Bits 7:6 – RCLKSEL[1:0] Watchdog Timer Clock Select bits

Value	Description
11	LPRC clock
10	Uses BFRC when system clock is not INTOSC/LPRC and device is not in Sleep; otherwise, uses INTOSC/LPRC
01	Reserved
00	Uses peripheral clock (F_{OSC}/4) when device is not in Sleep; otherwise, uses LPRC

Bits 5:1 – SWDTPS[4:0] Sleep Mode Watchdog Timer Period Select bits

Value	Description
11111	Divide by 2 ^ 31 = 2,147,483,648
11110	Divide by 2 ^ 30 = 1,073,741,824
. . .	
00001	Divide by 2 ^ 1 = 2
00000	Divide by 2 ^ 1 = 2

Bit 0 – WINDIS Watchdog Window Disable bit

Value	Description
1	Window mode disabled
0	Window mode enabled

7.1.5 FPRxCTRL Configuration Register

Name: FPRxCTRL
Offset: 0x7F4000, 0x7F4010, 0x7F4020, 0x7F4030, 0x7F4040, 0x7F4050, 0x7F4060, 07F4070

Legend: Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			Reserved[1:0]				RTYPE[1:0]	
Access			R/W	R/W			R/W	R/W
Reset			1	1			1	1

Bit	7	6	5	4	3	2	1	0
	CRC	WR	RD	EX			ERAO	RDIS
Access	R	R/W	R/W	R/W			R/W	R/W
Reset	0	1	1	1			1	1

Bits 13:12 – Reserved[1:0] Maintain as '11'

Bits 9:8 – RTYPE[1:0] Region Type bits

Value	Description
11	Firmware Configurable Region
10	One Time Programmable (OTP) Region
01	Immutable Root of Trust (IRT) Region
00	Reserved

Bit 7 – CRC Flash CRC Permission bit

Value	Description
1	Flash CRC calculation permitted
0	Flash CRC calculation not permitted

Bit 6 – WR Write Permission bit

Value	Description
1	Write (program/erase) permitted
0	Write (program/erase) not permitted

Bit 5 – RD Read Permission bit

Value	Description
1	Read (data read) permitted
0	Read (data read) not permitted

Bit 4 – EX Execute Permission bit

Value	Description
1	Execute (instruction fetch) permitted
0	Execute (instruction fetch) not permitted

Bit 1 – ERAO ECC Report Address Only on ECC Error bit

Value	Description
1	ECC error reporting information restricted to address only
0	No ECC error reporting restrictions

Bit 0 – RDIS Region Disable bit

Value	Description
1	Region disabled
0	Region enabled

7.1.6 FPRxST Configuration Register

Name: FPRxST
Offset: 0x7F4004, 0x7F4014, 0x7F4024, 0x7F4034, 0x7F4044, 0x7F4054, 0x7F4064, 0x7F4074

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		START[22:16]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	START[15:12]				START[11:8]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	1	1	1	1	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	START[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 22:16 – START[22:16] Start Address Offset MSb bits
Protection region start address offset (most significant bits)

Bits 15:12 – START[15:12] Start Address Offset MSb bits
Protection region start address offset (most significant bits)

Bits 11:0 – START[11:0] Start Address Offset LSb bits
Protection region start address offset (least significant bits), fixed value 0x0

7.1.7 FPRxEND Configuration Register

Name: FPRxEND
Offset: 0x7F4008, 0x7F4018, 0x7F4028, 0x7F4038, 0x7F4048, 0x7F4058, 0x7F4068, 0x7F4078

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		END[22:16]						
Access		R	R	R	R	R	R	R
Reset		1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	END[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	END[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bits 22:16 – END[22:16]

Bits 15:8 – END[15:8]

Bits 7:0 – END[7:0] End Address Offset LSb bits
Protection region end address offset (least significant bits) fixed value 0xFFF

7.1.8 FIRT Configuration Register

Name: FIRT
Offset: 0x7F4080

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								IRT
Access								R/W
Reset								1

Bit 0 – IRT Immutable Root of Trust Enable bit

Value	Description
1	IRT enabled
0	IRT disabled

7.1.9 FSECDBG Configuration Register

Name: FSECDBG
Offset: 0x7F4090

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								SECDBG
Access								R/W
Reset								1

Bit 0 – SECDBG Secure Debug Mode Enable bit

Value	Description
1	SECDBG enabled
0	SECDBG disabled

7.1.10 FPED Configuration Register

Name: FPED
Offset: 0x7F40A0

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								ICSPPED
Access								R/W
Reset								1

Bit 0 – ICSPPED ICSP Program/Erase Disable bit

Value	Description
1	ICSP Program/Erase disabled
0	ICSP Program/Erase enabled

7.1.11 FEPUCB Configuration Register

Name: FEPUCB
Offset: 0x7F40B0

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	FEPUCB[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FEPUCB[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FEPUCB[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FEPUCB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 31:0 – FEPUCB[31:0] UCB Erase Protect Register bits
UCB Erase Protect Code = 32'h84C1_F396

7.1.12 FWPUCB Configuration Register

Name: FWPUCB
Offset: 0x7F40C0

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	FWPUCB[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FWPUCB[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FWPUCB[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FWPUCB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 31:0 – FWPUCB[31:0] UCB Write Protect Register bits
UCB Write Protect Code = 32'h5B9B_12E4

7.2 Device Calibration and Identification

The dsPIC33AK128MC106 devices have two Identification registers near the end of configuration memory space that store the Device ID (DEVID) and Device Revision (DEVREV). These registers are used to determine the mask, variant and manufacturing information about the device. Addresses are listed in Table 7-4 and device specific values are listed in Table 7-5. These registers are read-only and are shown in 7.2.2. DEVID and 7.2.1. DEVREV.

7.2.1 Device Revision Register

Name: DEVREV

Legend: R = Read-only bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	Reserved							
Access	r							
Reset	0							

Bit	7	6	5	4	3	2	1	0
					DEVREV[3:0]			
Access					R	R	R	R
Reset								

Bit 15 – Reserved Maintain as '0'

Bits 3:0 – DEVREV[3:0] Device Revision bits

7.2.2 Device ID Register

Name: DEVID

Legend: R = Read-only bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	FAMID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset								

Bit	7	6	5	4	3	2	1	0
	DEV[7:0]							
Access	R	R	R	R	R	R	R	R
Reset								

Bits 15:8 – FAMID[7:0] Device Family Identifier bits
See Table 7-4 for the list of FAMID Identifier bits.

Bits 7:0 – DEV[7:0] Individual Device Identifier bits
See Table 7-5 for the list of Device Identifier bits.

8. Security Module

The security module protects the operation of the device and intellectual property from unauthorized access, use and modification.

The following security features are available on the dsPIC33AK128MC106 family of devices:

- Secure Boot
- Secure Debug
- Immutable Root of Trust (IRT)
- Code Protect
- ICSP Program/Erase Disable (Entire Flash OTP by ICSP Write Inhibit)
- Firmware IP Protection
- Flash Write Protection

The security features can be broken down into four categories:

- Device Locking
- Immutable Root of Trust (IRT)
- Configurable Flash Protection Regions
- Flash Access Control

Device Locking prevents unauthorized external access via debugger or programmer ICSP interfaces (local attacks). Device locking features include Code Protect, Entire Flash OTP by ICSP Write Inhibit and Secure Debug.

The **Immutable Root of Trust** (IRT) partition protects IRT firmware and data for implementation of Secure Boot, Secure Debug, Device Attestation and other security functions.

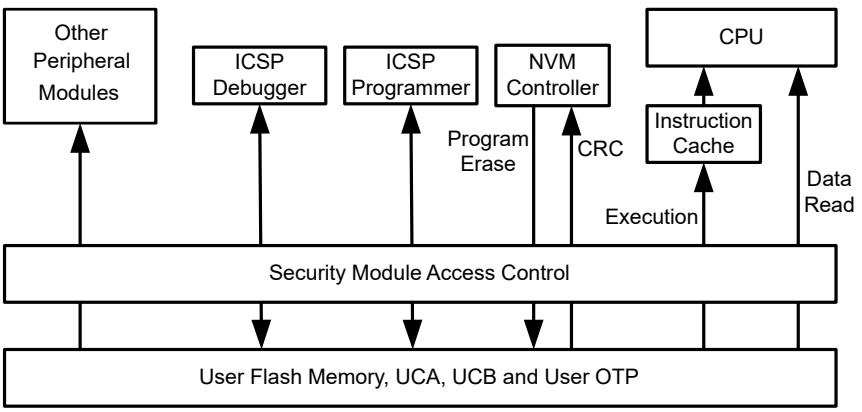
Eight **Configurable Protection Regions** provide flexible user program Flash access control. The protection regions include: IRT partition, immutable device firmware (OTP), firmware IP protection (execute-only memory), Flash write-protection and code Flash partitioning.

Flash Access Control is provided for the executive, user OTP and user configuration Flash spaces.

8.1 Architectural Overview

A simplified block diagram of the security module is illustrated in Figure 8-1

Figure 8-1. Security Module Block Diagram



Security module access control is provided for the entire system address space and is based on the address of the access, type of access and the device mode of operation. Access control is enforced

in all modes of operation for the CPU, NVM and for all other modules on the peripheral system bus. Accesses can be denied due to a variety of reasons including:

- Code-protect violation
- Flash protection region violation
- Access to reserved space or a Flash space-specific violation (e.g., attempted execution from user configuration space).

An access is allowed only if permitted by all access controls. An access control evaluation is done at the time the Flash memory is accessed. Accesses to the Flash space returned from the instruction cache, prefetch buffer or a NVM read data buffer are not checked.

8.1.1 Access Types

Access control is based on the type of access. There are four main access types:

- Execute (Instruction Fetch)
- Data Read
- Write (Program/Erase)
- CRC

The **Execute Access** type indicates a CPU instruction fetch.

The **Data Read Access** type indicates either a CPU operand or a peripheral system bus data access.

The **Write Access** type indicates either a Flash programming or erase operation. In some instances, access control differentiates between program and erase operations.

The **CRC Access** type indicates an access by CRC hardware integrated into the NVM Controller. The integrated CRC function allows firmware integrity checking without exposing the region contents outside the NVM Controller. This capability can be used for integrity checking of execute-only memory. NVM CRC hardware is limited to calculating a 32-bit CRC over one or more consecutive four Kbyte Flash pages.

8.1.2 Modes of Operation

Access control is based on the device mode of operation. There are three device modes of operation related to security:

- Mission mode
- Debug mode
- ICSP programming mode

Mission Mode is the default device mode of operation. After Reset, code in the user program Flash is executed. After booting, Mission mode also supports code execution from RAM. Execution from RAM is not visible to the Security Module and does not compromise Security Module functions.

Debug Mode can be entered either at Reset or upon hitting a debug breakpoint. It executes debug executive code stored in a dedicated Flash space separate from user program Flash to support in-circuit debugging. In this document, the term *"debug"* generically refers to in-circuit debugging using external development tools.

ICSP Programming Mode provides the device access using the external ICSP programming tools.

8.1.3 Configuration

The Security module is configured with the Flash configuration bits and SFRs. The configuration bits values are loaded from Flash at Reset and are maintained until the next Reset. Updates to the configuration values in Flash take effect on the next Reset. The Protection Region Descriptor SFRs Reset values are loaded from the configuration bits also. These Protection Region Descriptors

include locking capability (modification is disabled). This allows for several configuration options including:

- Dynamic configuration by firmware
- Static configuration by boot firmware
- Permanent configuration using the Flash configuration bits

Writes to a locked (read-only) Protection Region Descriptor register are ignored.

8.2 Security Module Register Summary

The UCPROT register contains bits to select the access to the User Configuration Areas A and B.

The IRTCTRL register controls the Immutable Root of Trust.

The IRTSTAT register contains the Immutable Root of Trust firmware defined status word.

The PRxCTRL register enables protection regions and defines the access level for each region.

The PRxST and PRxEND registers define start and end addresses of each protection region.

The PRxLOCK registers lock settings for the protection region.

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x02E0	UCPROT	31:24								
		23:16								
		15:8								
		7:0						WPUCA	WPUCB	EPUCB
0x02E4	IRTCTRL	31:24								
		23:16								
		15:8								
		7:0			IACT	PLCK		DONE	DBG	EAA
0x02E8	IRTSTAT	31:24	STATUS[31:24]							
		23:16	STATUS[23:16]							
		15:8	STATUS[15:8]							
		7:0	STATUS[7:0]							
0x02EC ... 0x02FF	Reserved									
0x0300	PR0CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0304	PR0ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0308	PR0END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x030C	PR0LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0310	PR1CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0314	PR1ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0318	PR1END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x031C	PR1LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0320	PR2CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0324	PR2ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0328	PR2END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x032C	PR2LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0330	PR3CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0334	PR3ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0338	PR3END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x033C	PR3LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0340	PR4CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0344	PR4ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0348	PR4END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x034C	PR4LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0350	PR5CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0354	PR5ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0358	PR5END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x035C	PR5LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0360	PR6CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0364	PR6ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0368	PR6END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x036C	PR6LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0370	PR7CTRL	31:24								
		23:16								
		15:8			Reserved[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0374	PR7ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0378	PR7END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x037C	PR7LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0380 ... 0x1E83	Reserved									
0x1E84	PACCON2	31:24								
		23:16		MBISTCONWR		RPCONWR	WDTCONWR	CM2RANGEWR	CM2CONWR	CM3RANGEWR
		15:8								
		7:0		MBISTCONLK		RPCONLK	WDTCONLK	CM4RANGELK	CM4CONLK	CM3RANGELK

8.2.1 User Configuration Areas Protection Register

Name: UCPROT
Offset: 0x2E0

Legend: x = Loaded from Configuration bits, S = Set Only bit, R = Readable bit

Notes:

1. WPUCA Reset value is based on the WPUCA[1:0] (FCP[3:2]) Configuration bits settings in the UCA area. WPUCA Write Protection bit does not apply in ICSP programming mode.
2. WPUCB is reset to 1 if the FWPUCB Configuration Word is programmed in the UCB area.
3. EPUCB is reset to 1 if the FEPUCB Configuration Word is programmed in the UCB area.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						WPUCA	WPUCB	EPUCB
Access						R/S	R/S	R/S
Reset						x	x	x

Bit 2 – WPUCA UCA Write Protect Enable bit(1)
Write 1 to set; Writing 0 has no effect

Value	Description
1	UCA is write protected (cannot be programmed or erased)
0	UCA is not write protected

Bit 1 – WPUCB UCB Write Protect Enable bit(2)
Write 1 to set; Writing 0 has no effect

Value	Description
1	UCB is write protected (cannot be programmed or erased)
0	UCB is not write protected

Bit 0 – EPUCB UCB Erase Protect Enable bit(3)
Write 1 to set; Writing 0 has no effect

Value	Description
1	UCB is erase protected (cannot be programmed or erased)
0	UCB is not erase protected

8.2.2 IRT Control Register

Name: IRTCTRL
Offset: 0x2E4

Legend: HS = Hardware Settable bit, S = Set Only bit, R = Readable bit

Notes:

1. Register is read only when PLCK bit is set.
2. If PLCK bit is set then the access to IRT regions is disabled and the IRTCTRL and IRTSTAT registers are read-only.
3. DBG bit controls debug access to the IRT partition when IRT is enabled.
4. EAA controls the external access via debug and ICSP interfaces when IRT and secure debug are enabled. Setting of EAA bit allows debug and ICSP programmer access; otherwise, these functions are disabled.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			IACT	PLCK		DONE	DBG	EAA
Access			R/HS	R/HS		R/S	R/W	R/W
Reset			0	0		0	0	0

Bit 5 – IACT

Value	Description
1	
0	

Bit 4 – PLCK IRT Partition Lock Status bit (1,2)

Value	Description
1	IRT partition is locked
0	IRT partition is not locked

Bit 2 – DONE UCA Write Protect Enable bit
Write 1 to set; Writing 0 has no effect

Value	Description
1	IRT execution is done
0	IRT execution is not finished

Bit 1 – DBG IRT Debug Enable bit (3)
Reset on POR or BOR only

Value	Description
1	Debug access to IRT partition is allowed
0	Debug access to IRT partition is disabled

Bit 0 – EAA External Access (Debugger or Programmer) Enable Bit(4)
Reset on POR or BOR only

Value	Description
1	Debug access to IRT partition is allowed
0	Debug access to IRT partition is disabled

8.2.3 IRT Status Word Register

Name: IRTSTAT
Offset: 0x2E8

Legend: R = Readable bit, W = Writable bit

Note:

1. Register is read-only when PLCK bit (IRTCTRL[4]) is set.

Bit	31	30	29	28	27	26	25	24
	STATUS[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	STATUS[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	STATUS[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	STATUS[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – STATUS[31:0] IRT Status Bits(1)
Reset only on POR or BOR. IRT firmware defined status word.

8.2.4 Protection Region n Control Register

Name: PRnCTRL
Offset: 0x300, 0x310, 0x320, 0x330, 0x340, 0x350, 0x360, 0x370

Legend: n = Region number; x = Loaded from Configuration Bits; HS = Hardware Settable bit, S = Set Only bit, R = Readable bit

Notes:

1. Register is read-only unless RTYPE[1:0] bits are '11' and LOCK[1:0] bits (PRnLOCK[1:0]) are '11'. Firmware configurable region descriptors are locked (read-only) at Reset and must be unlocked with a write to PRnLOCK before being modified.
2. IRT regions: all access disabled when the PLCK bit (IRTCRTL[4]) is set. For the OTP regions, the region write permissions are disabled, regardless of the WR bit. OTP and IRT are permanent regions.
3. The WR bit is not used for OTP regions; write permissions are always disabled for OTP regions.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			Reserved[1:0]				RTYPE[1:0]	
Access			R	R			R	R
Reset			x	x			x	x

Bit	7	6	5	4	3	2	1	0
	CRC	WR	RD	EX			ERAO	RDIS
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	x	x	x	x			x	x

Bits 13:12 – Reserved[1:0]

Bits 9:8 – RTYPE[1:0] Protection Region Type Selection bits (1,2)

Value	Description
11	Firmware Configurable Region
10	One Time Programmable (OTP) Region
01	Immutable Root of Trust (IRT) Region
00	Reserved

Bit 7 – CRC CRC Permission Enable bit

Value	Description
1	NVM Controller CRC is permitted
0	NVM Controller CRC is disabled

Bit 6 – WR

Write Permission Enable bit (3)

Value	Description
1	Write (program/erase) is permitted
0	Write (program/erase) is disabled

Bit 5 – RD Read Permission Enable bit

Value	Description
1	Read is permitted
0	Read is disabled

Bit 4 – EX Execute Permission Enable bit

Value	Description
1	Execute (instruction fetch) is permitted
0	Execute (instruction fetch) is disabled

Bit 1 – ERAO Error Report Address Only

Value	Description
1	ECC error reporting information restricted to address only
0	No ECC error reporting restrictions

Bit 0 – RDIS Protection Region Disable bit

Value	Description
1	Region is disabled
0	Region is enabled

8.2.5 Protection Region n Start Address Offset Register

Name: PRnST
Offset: 0x304, 0x314, 0x324, 0x334, 0x344, 0x354, 0x364, 0x374

Legend: n = Region number, x = Loaded from Configuration bits, R = Readable bit, W= Writable bit

Notes:

1. Register is read-only unless RTYPE[1:0] bits are '11' and LOCK[1:0] bits (PRLOCK[1:0]) are '11'. Firmware configurable region descriptors are locked (read-only) at Reset and must be unlocked with a write to PRLOCK before being modified.
2. Start address offset is a first byte offset from the beginning of the user program Flash (0x800000 address).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		START[22:16]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	x

Bit	15	14	13	12	11	10	9	8
	START[15:12]				START[11:8]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	x	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	START[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 22:16 – START[22:16] Most Significant bits of the Protection Region First Byte Address Offset(1,2)

Bits 15:12 – START[15:12] Most Significant bits of the Protection Region First Byte Address Offset(1,2)

Bits 11:8 – START[11:8] Least Significant bits of the Protection Region First Byte Address Offset. Fixed value 0x000.

Bits 7:0 – START[7:0] Least Significant bits of the Protection Region First Byte Address Offset. Fixed value 0x000.

8.2.6 Protection Region n End Address Offset Register

Name: PRnEND
Offset: 0x308, 0x318, 0x328, 0x338, 0x348, 0x358, 0x368, 0x378

Legend: n = Region number, x = Loaded from Configuration bits, R = Readable bit, W = Writable bit

Note:

1. Register is read-only unless RTYPE[1:0] bits are '11' and LOCK[1:0] bits (PRLOCK[1:0]) are '11'. Firmware configurable region descriptors are locked (read-only) at Reset and must be unlocked with a write to PRLOCK before being modified.
2. End address offset is a last byte offset from the beginning of the user program Flash (0x800000 address).
3. END value is loaded from Configuration Words.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		END[22:16]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	x

Bit	15	14	13	12	11	10	9	8
	END[15:12]				END[11:8]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	x	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	END[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bits 22:16 – END[22:16] Most Significant bits of the Protection Region Last Byte Address Offset(1,2,3)

Bits 15:12 – END[15:12] Most Significant bits of the Protection Region Last Byte Address Offset(1,2)

Bits 11:8 – END[11:8] Least Significant bits of the Protection Region Last Byte Address Offset. Fixed value 0xFFF.

Bits 7:0 – END[7:0] Least Significant bits of the Protection Region Last Byte Address Offset. Fixed value 0xFFF.

8.2.7 Protection Region n Lock Control Register

Name: PRnLOCK
Offset: 0x30C, 0x31C, 0x32C, 0x33C, 0x34C, 0x35C, 0x36C, 0x37C

Legend: n = Region number, x = Loaded from Configuration bits, R = Readable bit, W = Writable bit

Note:

1. The LOCK field does not apply to OTP and IRT region descriptors that cannot be modified by firmware.

Bit	31	30	29	28	27	26	25	24
	KEY[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	KEY[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
							LOCK[1:0]	
Access							R/W	R/W
Reset							0	0

Bits 31:16 – KEY[15:0] Write Key bits

Register is read-only if LOCK[1:0] bits are set to '01'. Writes to the PRxLOCK registers must be 32 bits with the 0xB737 key field value; otherwise, the write is ignored. The region descriptors are locked (read-only) on Reset to prevent inadvertent writes from corrupting a descriptor. The region descriptor registers are unlocked (writes enabled) by setting LOCK[1:0] bits to '11'.

Bits 1:0 – LOCK[1:0] Protection Region Lock Option bits

Register is read-only if LOCK[1:0] bits are set to '01'. Writes to the PRxLOCK registers must be 32 bits with the 0xB737 key field value; otherwise, the write is ignored. The region descriptors are locked (read-only) on Reset to prevent inadvertent writes from corrupting a descriptor. The region descriptor registers are unlocked (writes enabled) by setting LOCK[1:0] bits to '11'.

Value	Description
11	Region descriptor is unlocked (can be modified in SFRs)
10	Reserved
01	Region descriptor is locked, cannot be unlocked until after the next reset
00	Region descriptor is locked

8.3 Flash Memory Map

The Flash memory is divided into several spaces as shown in Table 8-1

Table 8-1. Flash Memory Map

Base Address	Space	Size
0x7F2C00	User OTP (One-Time-Programmable)	1 Kbyte
0x7F3000	UCA (User Configuration A)	4 Kbytes
0x7F4000	UCB (User Configuration B)	4 Kbytes
0x800000	Program (User Code)	Device dependent

8.3.1 User OTP

User OTP space is available for permanent storage of application specific data. User OTP Flash programming, data read and CRC access is permitted in all modes. Execution from User OTP Flash is not permitted. User OTP space is not erased on a chip erase. User OTP Flash words should only be programmed once. This must be enforced by firmware; there is no hardware access control that prohibits programming a Flash word multiple times. The User OTP space is separate from the Flash protection region OTP capability.

8.3.2 User Configuration A (UCA)

The UCA Flash space is used for storage device configuration bits, which includes code-protect and background debugger enable bits. UCA Configuration Words are loaded at Reset. Data read and CRC access to UCA space are permitted in all modes. Execution from UCA space is not permitted. Write permissions are restricted in Mission mode and Debug mode by the UCA Write-Protect WPUCA (UCPROT[2]) bit and/or by the UCA Write-Protect Configuration WPUCA[1:0] (FCP[3:2]) bits.

Subject to other access restrictions, a UCA space may be erased and then programmed in ICSP Programming mode even if the write-protect in Configuration bits WFUCA[1:0] (FCP[3:2]) is enabled. The write access to UCA spaces is not allowed if code-protect and/or Entire Flash OTP by ICSP Write Inhibit is enabled. Row programming is not allowed for UCA space. Programming of the write-protect Configuration bits WPUCA[1:0] (FCP[3:2]), prevents UCA updates in Mission mode and Debug mode. In this case, UCA can only be updated in an ICSP Programming mode. The WPUCA (UCPROT[2]) bit may also be programmed by application firmware to enable write-protection after the next Reset. UCA is always erased on a chip erase, even when write-protected. A chip erase disables UCA write protection and code protection.

8.3.3 User Configuration B (UCB)

The UCB stores security and boot Configuration bits including the Flash protection region descriptor configurations. UCB data read and CRC access is permitted in all modes. Execution from UCB space is not permitted. Write permission is restricted in all modes by the UCB Write Protect WPUCB bit (UCPROT[1]). The UCB write protection is enabled after Reset if the FWPUCB Configuration Word is programmed with a value of 0x5B9B12E4. Also, the UCB area can be erase-protected by the EPUCB bit (UCPROT[0]).

The UCB erase protection is enabled after Reset if the FEPUCB Configuration Word is programmed with a value of 0x84C1F396. Write protection disables both programming and erase. Erase protection only disables erase. The UCB write and erase protect Configuration Words (FWPUCB and FEPUCB) are 32-bit OTP Flash locations that can only be programmed to their specified values (0x5B9B12E4 and 0x84C1F396). Row programming is not allowed for UCB. UCB is erased on a chip erase unless it is either erase protected when EPUCB (UCPROT[0]) = '1' or write-protected when WPUCB (UCPROT[1]) = '1'. If the UCB erase protection Configuration Word is programmed, all programmed UCB Configuration Words are protected from modification. This allows multiple parties to program firmware and data into user program Flash and permanently protect it from modification using IRT or OTP regions.

An aspect of this capability is UCB overwrite protection. UCB overwrite protection ensures Flash Configuration Words in UCB are only programmed once after each UCB page erase. Once any valid Configuration bit in a UCB 128-bit Flash word is programmed to '0', further programming is not

allowed for that Flash word without a page erase. UCB overwrite protection is only provided for a word that has been programmed after the next Reset because the overwrite protection is based on the configuration values loaded at Reset.

Once all protection region descriptors and other UCB Configuration Words are programmed, UCB can be permanently write- protected by programming the UCB write-protect FWPUCB Configuration Word. Typically, the UCB write-protect Configuration bit should be programmed before a device is deployed in a system design. The UCB Write-Protect bit must be programmed to enable the Entire Flash OTP by ICSP Write Inhibit feature and/or the Secure Debug. During development or the system production process, the EPUCB (UCPROT[0]) and WPUCB (UCPROT[1]) bits can be set by firmware to provide UCB protection without programming the UCB write or erase-protect Configuration Words.

8.3.4 Program Memory

User program Flash stores code and data for Mission mode execution. Unless restricted by an access control, there is full access (execute, data read, write, CRC) to user program Flash in all modes. User program Flash access controls include the protection regions and code-protect.

Programming the UCB write protect word permanently prevents modification of all UCB configuration bits settings. Protection region and code protect access controls apply to all of user program space. The chip erase operation erases all of user program Flash except for permanent (OTP and IRT) protection regions. If permanent protection regions are configured and UCB is either erase protected (EPUCB (UCPROT[0]) = '1') or write protected (WPUCB (UCPROT[1]) = '1'), only the user program memory Flash outside the permanent regions is erased. In this case, UCB, which stores the permanent region descriptors and the permanent region contents, is not affected by the chip erase. If UCB is not erase protected (EPUCB (UCPROT[0]) = '0') and not write protected (WPUCB (UCPROT[1]) = '0'), then UCB and the entire user program and user data Flash are erased by a chip erase. UCA is always erased by a chip erase. Chip erase is only allowed in ICSP Programming Mode. If Secure Debug is enabled, IRT firmware authorization is required for a chip erase. If Entire Flash OTP by ICSP Write Inhibit is enabled, chip erase is permanently disabled in all modes. Chip erase overrides firmware configurable (non-permanent) protection region write protections. Firmware configurable region contents are erased on a chip erase. Regardless of UCB erase or write protections, if permanent protection regions are configured, only user program Flash outside the permanent regions is erased.

8.3.5 Security Configuration Words

A summary of security related Configuration Words is shown in Table 8-2. Each entry in the table is a 128-bit Flash word that can be independently programmed. Not all 128 bits are used for any Configuration Word; unused bits are reserved. The table shows the primary Configuration Words and their addresses. The Configuration Words have primary and secondary locations, with the secondary location used in case of an uncorrectable bit error when loading the primary location. UCA and UCB secondary word addresses are the primary address + 0x800. FEPUCB and FWPUCB are OTP words that have 32-bit values to prevent inadvertent programming. These words can only be programmed to the specified value (0x84C1F396 for FEPUCB and 0x5B9B12E4 for FWPUCB). If the word has the specified 32-bit value, it is considered programmed and that option is enabled; otherwise, the option is disabled. UCB Configuration Words may not be programmed if any valid bit/field from that Flash word had previously been programmed (not all ones). The UCB overwrite protection is based on the fuse values loaded at Reset, therefore it only takes effect for a word that has been programmed after the next Reset.

Table 8-2. Security Configuration Words

Primary Configuration Words		Secondary (Backup) Configuration Words	
Address	**Name**	**Address**	**Name**
User Configuration A			
0x7F3000	FCP	0x7F3800	FCPBKUP
0x7F3010	FICD	0x7F3810	FICDBKUP

.....continued			
Primary Configuration Words		Secondary (Backup) Configuration Words	
Address	Name	Address	Name
0x7F3020	FDEVOPT	0x7F3820	FDEVOPTBKUP
User Configuration B			
0x7F4000	FPR0CTRL	0x7F4800	FPR0CTRLBKUP
0x7F4004	FPR0ST	0x7F4804	FPR0STBKUP
0x7F4008	FPR0END	0x7F4808	FPR0ENDBKUP
0x7F4010	FPR1CTRL	0x7F4810	FPR1CTRLBKUP
0x7F4014	FPR1ST	0x7F4814	FPR1STBKUP
0x7F4018	FPR1END	0x7F4818	FPR1ENDBKUP
0x7F4020	FPR2CTRL	0x7F4820	FPR2CTRLBKUP
0x7F4024	FPR2ST	0x7F4824	FPR2STBKUP
0x7F4028	FPR2END	0x7F4828	FPR2ENDBKUP
0x7F4030	FPR3CTRL	0x7F4830	FPR3CTRLBKUP
0x7F4034	FPR3ST	0x7F4834	FPR3STBKUP
0x7F4038	FPR3END	0x7F4838	FPR3ENDBKUP
0x7F4040	FPR4CTRL	0x7F4840	FPR4CTRLBKUP
0x7F4044	FPR4ST	0x7F4844	FPR4STBKUP
0x7F4048	FPR4END	0x7F4848	FPR4ENDBKUP
0x7F4050	FPR5CTRL	0x7F4850	FPR5CTRLBKUP
0x7F4054	FPR5ST	0x7F4854	FPR5STBKUP
0x7F4058	FPR5END	0x7F4858	FPR5ENDBKUP
0x7F4060	FPR6CTRL	0x7F4860	FPR6CTRLBKUP
0x7F4064	FPR6ST	0x7F4864	FPR6STBKUP
0x7F4068	FPR6END	0x7F4868	FPR6ENDBKUP
0x7F4070	FPR7CTRL	0x7F4870	FPR7CTRLBKUP
0x7F4074	FPR7ST	0x7F4874	FPR7STBKUP
0x7F4078	FPR7END	0x7F4878	FPR7ENDBKUP
0x7F4080	FIRT	0x7F4880	FIRTBKUP
0x7F4090	FSECDBG	0x7F4890	FSECDBGBKUP
0x7F40A0	FTPED	0x7F48A0	FTPEDBKUP
0x7F40B0	FEPUCB	0x7F48B0	FEPUCBBKUP
0x7F40C0	FWPUCB	0x7F48C0	FWPUCBBKUP

8.4 Device Locking

8.4.1 Code Protect

Code-protect is a device locking option that prevents external read-out and modification of the user program. The code protection is enabled with a Configuration CP (FCP[0]) bit in the UCA configuration area. Enabling code protection does the following:

- Disables debug
- Disables Configuration Words override in ICSP Programming mode
- Disables UCA write permissions in ICSP Programming mode
- Restricts access to Program Memory in ICSP Programming mode
- Restricts ECC error reporting for the Program Memory in the ICSP Programming mode

Code protection disables user program Flash execute, read and write permissions in ICSP Programming mode. The Program Memory CRC Configuration CRC (FCP[1]) bit can be programmed to allow user program memory CRC access in ICSP Programming mode when code-protect is enabled. The Flash protection regions may impose CRC access restrictions on all or part of user program Flash that applies in all modes. Code-protect also disables write permissions to UCA spaces in ICSP Programming mode. Code-protect is disabled after a chip erase; this allows a device to be reprogrammed without revealing the original user program or user data Flash contents. The permanent (OTP and IRT) regions are not erased on a chip erase. The Entire Flash OTP by ICSP Write Inhibit can be enabled to prevent external tools from chip erasing the device. UCA may be write-protected to prevent code-protect from being disabled in Mission mode. Enabling code-protect does not disable ICSP Programming Mode. This potentially allows access to data stored in RAM and registers that are not cleared on a Reset. Secure debug provides protection from this vulnerability.

If code-protect is enabled with CRC allowed, the ICSP Programming Mode may calculate the CRC over user program Flash pages. To prevent access to the protected spaces using ECC error injection and reporting, ECC error reporting data is restricted to address only for user program in ICSP Programming Mode when code protect is enabled.

8.4.2 ICSP Program/Erase Disable (Entire Flash OTP by ICSP™ Write Inhibit)

ICSP Program/Erase Disable (Entire Flash OTP by ICSP Write Inhibit) permanently disables unsecured external chip erase and Flash programming. Entire Flash OTP by ICSP Write Inhibit is enabled with the TPED (FTPED[0]) Configuration bit and takes effect when the UCB write-protect FWPUCB word is also programmed with 0x5B9B12E4 value. Enabling Entire Flash OTP by ICSP Write Inhibit does the following:

- Disables chip erase in all modes
- Disables Flash programming and erase in ICSP Programming mode

Entire Flash OTP by ICSP Write Inhibit is intended to be used with the code-protect to permanently disable external readout or modification of the user program via unsecured programming or debug interfaces. When both code-protect and Entire Flash OTP by ICSP Write Inhibit are enabled, external tools cannot enable debug, access user program, modify Flash or disable code-protect. Programming and erase of the Flash can only be done in Mission mode. Mission mode write permissions are not affected by Entire Flash OTP by ICSP Write Inhibit. IRT firmware may support secure programming when Entire Flash OTP by ICSP Write Inhibit is enabled.

8.4.3 Secure Debug

Secure debug is a device locking option that provides IRT firmware control over external access to the device. Secure debug is enabled with a Configuration bit SECDBG (FSECDBG[0]) in the UCB area. The IRT and UCB write-protect Configuration bits also need to be programmed for secure debug access controls to take effect. When secure debug is enabled, external access is controlled by the EAA (IRTCTRL[0]) bit which is only reset on a cold Reset (POR or BOR). Only IRT firmware can write to IRTCTRL unless the DBG (IRTCTRL[1]) bit is set. When EAA (IRTCTRL[0]) bit = '0', the device is locked. All external access via debug and programming interfaces is disabled.

When the JTAG port is enabled, only the boundary scan function is allowed when the device is locked. When EAA (IRTCTRL[0]) bit = '1', debug and test access is allowed. Unless DBG (IRTCTRL[1]) bit = '1', IRT Flash regions remain protected from the external and debug access. However, a properly timed Reset during IRT firmware execution may leak IRT information in RAM and registers, which retain their state through the Reset. So, when EAA bit = '1', IRT firmware should not access symmetric (secret) or private operational keys if these need to be protected from the external access when the device is unlocked. Public keys may be accessed as disclosure of these keys is not a security issue. IRT firmware may use any means for secure debug external access authorization. The IRT firmware may implement various non-volatile external access configuration options, including access based on an authenticated unlock token, permanently disabling access and unrestricted

access. In any case, when secure debug is enabled, the device must boot after a cold Reset for the IRT firmware to enable access by setting the EAA bit.

Both the Entire Flash OTP by ICSP Write Inhibit and UCB write protect configuration bits must be programmed for Entire Flash OTP by ICSP Write Inhibit access controls to be enabled. UCB write-protect is an OTP Configuration Word. Once Entire Flash OTP by ICSP Write Inhibit is enabled, it cannot be disabled.

IRT, secure debug and UCB write-protect Configuration bits must all be programmed for secure debug to be enabled.

UCB write-protect (FWPUCB) is an OTP Configuration word. Once secure debug is enabled, it cannot be disabled. Secure debug can be used either with or without code protection. When secure debug is enabled without code protection, once authorization is given, debug and test access are allowed with full access to user program and user data Flash (subject to protection region restrictions). When both secure debug and code-protect are enabled, both IRT authorization and a chip erase are required for external access to user program Flash. This allows user program to be changed without revealing its original contents and only with IRT authorization. The permanent regions (IRT and OTP) are not erased on a chip erase. When secure debug is enabled, there is no means to disable or bypass secure debug access controls. For development, secure debug can be emulated by external development tools (debugger). To emulate the secure debug, the secure debug Configuration bit (SECDBG) should not be programmed. The external development tools follow the normal procedure for enabling external access as if secure debug is enabled. The IRT firmware sets the EAA bit state accordingly. The external development tools check the EAA bit to determine if external access would be permitted if the secure debug was enabled.

8.4.4 Immutable Root of Trust (IRT)

Device security functions including secure boot, secure debug and device attestation require a root of trust. The root of trust executes at boot time and may include multiple stages forming a chain of trust. The chain of trust operates by having each stage authenticate the next stage before transferring control to the next stage. The first stage in the chain of trust is inherently trusted and must have immutable firmware. This first and possibly only root of trust stage is referred to as the Immutable Root of Trust (IRT).

In Mission mode, IRT firmware is the first firmware to execute after a Reset. The root of trust consisting of one or more stages with immutable and optionally updatable firmware components may provide any number of security services including secure boot, secure debug, secure programming, over-the-air (OTA) firmware update and device attestation.

8.4.4.1 IRT Partition

A designated portion of the user program is used for the IRT partition. The IRT Flash space is specified using Flash protection regions. One or more permanent IRT regions may be created for firmware, data, and cryptographic keys. Once IRT is enabled, IRT regions are only accessible during IRT execution except for debug access when authorized by the IRT firmware. An IRT region may have write permissions disabled making it immutable (generally for firmware) or have write permissions enabled (generally for configuration data and keys) allowing it to be updated.

IRT is enabled by the IRT (FIRT[0]) Configuration bit in the UCB area. By default (erased UCB), IRT is disabled. When IRT is disabled, IRT regions (if any) are not protected and are accessible to firmware outside the IRT partition. IRT regions are only truly permanent once either the UCB erase-protect and/or UCB write-protect words are programmed. For development, the UCB erase and write-protect words may be left unprogrammed. When IRT is enabled, a permanent immutable protection region with IRT firmware must be configured for the beginning of user program Flash. This region is functionally the boot ROM for the device and may be either an OTP region or IRT region with write permission disabled.

OTP region firmware can be shared between the IRT and other firmware components. IRT firmware execution starts after resetting into Mission mode. IRT execution ends when the IRT firmware sets

the DONE (IRTCTRL[2]) bit and there is a subsequent instruction fetch from user program Flash not within an IRT region. This includes an attempted instruction fetch that has an error, such as an access privilege violation or an uncorrectable bit error.

IRT firmware may set the DONE bit (IRTCTRL[2]) and transfer control to the application (non-secure) code while executing from an IRT region. IRT firmware must ensure that the application code is not prefetched before the DONE bit is set. This ensures that the start of the application code is fetched after the DONE bit is set, disabling the IRT partition before the application code can execute. This requirement can be met by having at least 32 bytes of separation between the end of IRT firmware and the start of application firmware. IRT firmware must ensure the application start address is in user program Flash and not within an IRT region. For the IRT partition to be secured, either the UCB write-protect word or the UCB erase-protect word must be programmed. For development, the UCB erase and write-protect words may be left unprogrammed. Permission to access the IRT partition is indicated by the partition lock PLCK (IRTCTRL[4]) bit. Partition access is allowed (PLCK bit = '0') when IRT is disabled, during IRT execution and for debug access when the DBG (IRTCTRL<1>) bit is set. Otherwise, partition access is disabled (PLCK bit = '1').

The IRT partition implementation is based on the principle of temporal isolation. Since IRT firmware is the first firmware to execute after Reset, it controls execution of other firmware components. Sensitive IRT data is cleared from registers and RAM before executing any application (non-secure firmware). Likewise, the IRT partition is disabled (PLCK bit = '1') before executing the application (non-secure firmware), preventing it from accessing or corrupting IRT Flash regions. If IRT firmware code needs to be protected from access by non-security firmware, the instruction cache should be invalidated and disabled before transferring control to the application firmware. The IVT base address should be set to application space before attempting to transfer control to the application firmware. This ensures that application firmware exceptions, starting with the first application code instruction, can be handled by application trap handlers.

Secure debug must be enabled and external access must be disabled (EAA (IRTCTRL[0]) bit = '0') for the IRT partition to be fully protected from the external access. When secure debug is not enabled or when it is enabled but external access is allowed (EAA bit = '1'), the external access to internal registers and RAM values is possible via ICSP interface. A properly timed Reset during IRT execution may reveal sensitive information in RAM and registers that retain their state through the Reset.

IRT partition access may be extended to updatable root of trust firmware components. One or more updatable components (stages) are authenticated directly by IRT firmware or indirectly with the chain of trust process previously described. The updatable root of trust components executes with the IRT partition enabled. The IRT partition is locked by the IRT or updatable root of trust firmware by setting the DONE bit before starting execution of the application (non-secure code). Alternatively, updatable root of trust components may have an independent security partition. In this case, the IRT disables the IRT partition before transferring control to an updatable root of trust component in a non-IRT region. A separate (non-IRT) protection region can be set up for the updatable root of trust components to store cryptographic keys and configuration data. Access to this region is disabled before transferring control to the application (non-secure firmware). Unlike the IRT partition, this partition is erased on a chip erase and can be accessed by debug without a specific authorization. IRT firmware may implement a security life cycle state using IRT region storage. This is separate from the higher-level device life cycle state controlled by the UCB write protect word. Additionally, IRT firmware may implement non-volatile Secure Debug configuration options.

IRT partition memory is not erased on a chip erase and debug access to IRT memory is controlled by IRT firmware.

8.4.4.2 IRT Control Register

The IRT control register IRTCTRL includes the EAA, DBG, IACT, DONE and PLCK bits. IRTCTRL is read only except when PLCK bit = '0'. The EAA bit controls debug entry when secure debug is enabled. The DBG bit controls debug access to the IRT partition when IRT and debug are enabled. The DBG bit is cleared by a cold Reset (POR or BOR) and must be set by IRT firmware to enable the IRT firmware debug.

When the DBG bit = '0', there is no debug access to the IRT partition. CPU breakpoints and debugger memory accesses are disabled during IRT execution. IRT partition access is only enabled (PLCK bit = '0') during IRT execution. When the DBG bit = '1', debug access to the IRT partition is enabled. CPU breakpoints and debugger memory accesses are allowed during IRT execution. IRT partition access is enabled (PLCK bit = '0') when the device resets into Debug mode.

The DONE bit is set by the IRT (or other root of trust) firmware upon completion of root of trust execution. The IRT partition is locked (PLCK bit = '1') when the DONE bit is set and there is a subsequent instruction fetch from the user program Flash not within an IRT region. This includes an attempted instruction fetch which has an error, such as an access privilege violation or an uncorrectable bit error. The PLCK bit is a read-only bit that indicates if the IRT partition is locked (access disabled). If the PLCK bit = '1', access to IRT regions is disabled and the IRTCTRL and IRTSTAT registers are read- only.

8.4.4.3 IRT Status Register

The IRT status register (IRTSTAT) is a 32-bit read/write register available for IRT firmware status and is only reset on a cold Reset (POR or BOR). The IRTSTAT register usage is defined by the IRT firmware. It can be used to store secure boot and other status information across the device Resets. The IRTSTAT register is read-only except when the PLCK bit = '0'.

8.5 Flash Protection Regions

Eight configurable protection regions are available for user program and user data Flash access control. A protection region can cover one or more consecutive 4 Kbyte pages anywhere in user program Flash. Protection regions provide access control based on the type of access for the designated portion of Flash. Access control is enforced in all modes of operation. Regions may overlap with the most restrictive permissions taking precedence. Subject to other access controls, there is full access (execute, data read, write, CRC) for user program Flash pages not included within any protection region.

Flash protection regions use include:

- Flash write protection
- OTP Flash
- IRT partition
- Firmware IP protection (execute only memory)
- Flash code partitioning.

8.5.1 Region Descriptor

Each protection region x has a region descriptor comprised of four SFRs: Control (PRnCTRL), Start Address (PRnST), End Address (PRnEND) and Lock (PRnLOCK). Most region descriptor register bit/field Reset values can be configured with Configuration bits (FPRnCTRL, FPRnST, FPRnEND and FPRnLOCK).

Region descriptor configuration options include dynamic configuration by firmware, static configuration by boot firmware and permanent configuration using fuses. This last option enables creating permanent OTP and IRT regions in program Flash that cannot be modified even with a chip erase. Firmware configurable region descriptors also include a locking mechanism. The PRnLOCK register must be configured to allow descriptor writes; otherwise, writes to other descriptor registers are ignored.

8.5.2 Region Control Register

The region descriptor PRnCTRL register includes the region disable bit: RDIS (PRnCTRL[0]), access permission bits: CRC, WR, RD, EX (PRnCTRL[7:4]) and region type field: RTYPE[1:0] (PRnCTRL[9:8]). When RDIS bit = '1', the region is disabled, and it has no effect. When RDIS bit = '0', the region is enabled, and the specified portion of user program space is subject to access control based on the region type and permissions settings.

When read access is disabled, the ECC error reporting is restricted for the region to address information only. The data, parity and syndrome information are not reported. This prevents access to region firmware using ECC error injection and reporting.

There are four permission bits, each corresponds to a Flash access type as follows:

- CRC - NVM Controller CRC
- WR - Write (Program/Erase)
- RD - Read (Data Read)
- EX - Execute (Instruction Fetch)

Access to the region is controlled based on the access type. If the permission bit associated with the access type is set, the access is allowed. Otherwise, the access is not allowed. The OTP regions never allow write operations regardless of the WR bit setting. The CRC access type refers to accesses by CRC hardware integrated into the NVM Controller. The NVM Controller CRC hardware is designed to allow for firmware integrity verification without exposing the contents of the region.

The RTYPE field which is read-only can be one of three values:

- Firmware Configurable
- One Time Programmable (OTP)
- Immutable Root of Trust (IRT)

OTP and IRT regions are permanent regions configured by Configuration bits and their region descriptors cannot be modified. The permanent (OTP and IRT) region descriptors in UCB and the contents of the regions in user program space are not erased on a chip erase if either EPUCB (UCPROT[2]) bit = '1' or WPUCB (UCPROT[1]) = '1'. To make OTP and IRT regions truly permanent, either (or both) the FEPUCB or FWPUCB Configuration words must be programmed.

Firmware configurable region descriptors can be modified. Firmware configurable is the default type for regions when FPRnCTRL Configuration words are not programmed.

8.5.3 Region Start and End Address Registers

The region n descriptor registers, PRnST and PRnEND, define the start address offset and end address offset for the region. The address offsets are on four Kbyte Flash page boundaries. The address offsets are an offset from the beginning of either the selected user program partition or user data Flash.

If the start address is greater than the end address, then the region size is zero and the region access controls do not apply to any portion of Flash.

8.5.4 Region Lock Register

The region x descriptor PRnLOCK register has two fields: a 16-bit KEY[15:0] field and 2-bit LOCK[1:0] field.

Writes to the register must be 32 bits (long word) and have the correct KEY value of 0xB737. Otherwise, the write is ignored. The LOCK field has three options:

- Region descriptor unlocked (LOCK[1:0] bits = '11')
- Region descriptor locked, cannot be unlocked until after the next reset (LOCK[1:0] bits = '01')
- Region descriptor locked (LOCK[1:0] bits = '00')

If the descriptor is locked, the PRnCTRL, PRnST and PRnEND registers are read-only. The Reset value for the LOCK[1:0] field is region descriptor locked, so a PRnLOCK register write is required to unlock the descriptor. The PRnLOCK register is not used for OTP and IRT region descriptors, which cannot be modified.

8.5.5 Permanent Regions

Multiple parties can create permanent OTP and/or IRT Flash regions in a device. If either the UCB erase-protect and/or UCB write-protect Configuration Words are programmed, OTP and IRT region descriptors cannot be modified even with a chip erase.

OTP regions can be used to prevent firmware tampering for devices that do not require firmware updates. This eliminates the need for secure boot in these devices.

Execute-only OTP regions can be used for firmware IP protection. These regions are configured to not allow data reads but allow execution and CRC access for integrity checking without exposing the protected code.

IRT protection regions designate the root of trust partition within the user program space. IRT regions can be used for IRT firmware, data and cryptographic keys. OTP regions can also be used for IRT firmware.

8.5.6 Code Partitioning

Protection regions configured exclusively for code (execute-only) or data (no execute) can be used to partition user program Flash into separate code and data spaces. This partitioning prevents inadvertent execution from data tables or data accesses to code images. Protection regions can be configured to disable all access for unused portions of Flash.

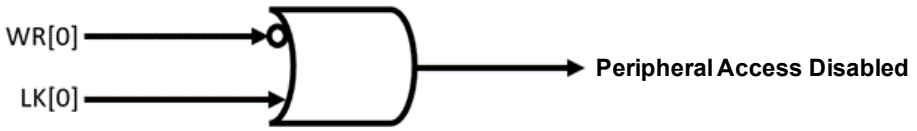
8.6 Peripheral Access Controller (PAC)

Legacy dsPIC/PIC devices used a locking mechanism where the user wrote 0xAA and 0x55 into NVMKEY in a sequence. This was to prevent accidental enabling or disabling of critical peripherals. This feature has been replaced with a dedicated module called the Peripheral Access Controller (PAC).

Register Locking and Unlocking

The module implements an OR gate between the associated peripheral's PACCONx lock bit and the inverse of the PACCONx write enable bit. See Figure 8-2 for a diagram of the PAC locking behavior. If the target peripheral LK bit or the inverse of the WR bit is set, the target register or registers cannot be written to, only read access is allowed.

Figure 8-2. PAC Locking Behavior



If the LK bit is set OR the WR is clear, peripheral access is DISABLED.

Individual and Range Mode

The PAC module can lock/unlock individual registers and lock/unlock a range of registers depending on which registers are the target registers. If the PAC module uses range mode, the entire range of target peripheral registers are covered by that lock and write enable bit. Table 8-3 below is used to determine which registers use individual or range mode.

Table 8-3. Individual and Range Mode

Register	Individual or Range Mode
IVTBASE	Individual
IVTCREG	Individual
BMXIRAML	Individual

.....continued

Register	Individual or Range Mode
BMXIRAMH	Individual
PCLKCON	Individual
IOIMCON1	Individual
IOIMCON2	Individual
IOIMCON3	Individual
IOIMCON4	Individual
NVMCON	Individual
OSCCTRL	Individual
CM1CON	Individual
CM1RANGE	Range
CM2CON	Individual
CM2RANGE	Range
CM3CON	Individual
CM3RANGE	Range
CM4CON	Individual
CM4RANGE	Range
WDTCON	Individual
RPCON	Individual
MBISTCON	Individual

8.6.1 Peripheral Access Control Register 1

Name: PACCON1

Legend: R = Readable bit, W = Writable bit, S = One Way Settable bit

Bit	31	30	29	28	27	26	25	24
	CM3CONWR	CM2RANGEWR	CM2CONWR	CM1RANGEWR	CM1CONWR	OSCCTRLWR	NVMCONWR	IOIM4CONWR
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	IOIM3CONWR	IOIM2CONWR	IOIM1CONWR	PCLKCONWR	BMXIRAMHWR	BMXIRAMLWR	IVTCREGWR	IVTBASEWR
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	CM3CONLK	CM2RANGELK	CM2CONLK	CM1RANGELK	CM1CONLK	OSCCTRLLK	NVMCONLK	IOIM4CONLK
Access	S/R	S/R	S/R	S/R	S/R	S/R	S/R	S/R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	IOIM3CONLK	IOIM2CONLK	IOIM1CONLK	PCLKCONLK	BMXIRAMHLK	BMXIRAMLLK	IVTCREGLK	IVTBASELK
Access	S/R	S/R	S/R	S/R	S/R	S/R	S/R	S/R
Reset	0	0	0	0	0	0	0	0

Bit 31 – CM3CONWR Clock Monitor 3 Control Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 30 – CM2RANGEWR Clock Monitor 2 Range (CM2WINPR - CM2LWARN) Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 29 – CM2CONWR Clock Monitor 2 Control Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 28 – CM1RANGEWR Clock Monitor 1 Range (CM1WINPR - CM1LWARN) Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 27 – CM1CONWR Clock Monitor 1 Control Register Write Enable bit

Value	Description
1	Register is writable

Value	Description
0	Register is not writable

Bit 26 – OSCCTRLWR System Clock Control Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 25 – NVMCONWR Nonvolatile Memory Control Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 24 – IOIM4CONWR IOIM 4 Control Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 23 – IOIM3CONWR IOIM 3 Control Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 22 – IOIM2CONWR IOIM 2 Control Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 21 – IOIM1CONWR IOIM 1 Control Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 20 – PCLKCONWR PWM Clock Control Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 19 – BMXIRAMHWR BMX Instruction RAM High Address Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 18 – BMXIRAMLWR BMX Instruction RAM Low Address Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 17 – IVTCREGWR Interrupt Vector Collapse Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 16 – IVTBASEWR Interrupt Vector Base Address Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 15 – CM3CONLK Clock Monitor 3 Control Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 14 – CM2RANGELK Clock Monitor 2 Range (CM2WINPR - CM2LWARN) Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 13 – CM2CONLK Clock Monitor 2 Control Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 12 – CM1RANGELK Clock Monitor 1 Range (CM1WINPR - CM1LWARN) Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 11 – CM1CONLK Clock Monitor 1 Control Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 10 – OSCCTRLLK System Clock Control Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 9 – NVMCONLK Nonvolatile Memory Control Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 8 – IOIM4CONLK IOIM 4 Control Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 7 – IOIM3CONLK IOIM 3 Control Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 6 – IOIM2CONLK IOIM 2 Control Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 5 – IOIM1CONLK IOIM 1 Control Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 4 – PCLKCONLK PWM Clock Control Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 3 – BMXIRAMHLK BMX Instruction RAM High Address Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 2 – BMXIRAMLLK BMX Instruction RAM Low Address Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 1 – IVTCREGLK Interrupt Vector Collapse Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 0 – IVTBASELK Interrupt Vector Base Address Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

8.6.2 Peripheral Access Control Register 2

Name: PACCON2
Offset: 0x1E84

Legend: R = Readable bit, W = Writable bit, S = One Way Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		MBISTCONWR		RPCONWR	WDTCONWR	CM2RANGEWR	CM2CONWR	CM3RANGEWR
Access		R/W		R/W	R/W	R/W	R/W	R/W
Reset		1		1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		MBISTCONLK		RPCONLK	WDTCONLK	CM4RANGELK	CM4CONLK	CM3RANGELK
Access		S/R		S/R	R/W	R/W	R/W	R/W
Reset		0		0	1	1	1	1

Bit 22 – MBISTCONWR MBIST Control Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 20 – RPCONWR Peripheral Remapping Configuration Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 19 – WDTCONWR Watchdog Timer Control Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 18 – CM2RANGEWR Clock Monitor 2 Range (CM2WINPR - CM2LWARN) Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 17 – CM2CONWR Clock Monitor 2 Control Register Write Enable bit

Value	Description
1	Register is writable

Value	Description
0	Register is not writable

Bit 16 – CM3RANGEWR Clock Monitor 3 Range (CM3WINPR - CM3LWARN) Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 6 – MBISTCONLK MBIST Control Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 4 – RPCONLK Peripheral Remapping Configuration Register Lock bit

Value	Description
1	Register is write locked
0	Register is not write locked

Bit 3 – WDTCONLK Watchdog Timer Control Register Lock bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 2 – CM4RANGELK Clock Monitor 4 Range (CM4WINPR - CM4LWARN) Lock bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 1 – CM4CONLK Clock Monitor 4 Control Register Lock bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 0 – CM3RANGELK Clock Monitor 3 Range (CM3WINPR - CM3LWARN) Lock bit

Value	Description
1	Register is writable
0	Register is not writable

Bit 0 – IOIM12CONWR IOIM 12 Control Register Write Enable bit

Value	Description
1	Register is writable
0	Register is not writable

9. Resets

The dsPIC33A family of devices implements a Reset module to safely control device start up, faults and external conditions. There are two types of Resets: cold Reset and warm Reset.

A cold Reset is the result of a Power-on Reset (POR) or Brown-out Reset (BOR). A warm Reset is the result of all other Reset sources, including the `RESET` instruction.

The device is kept in a Reset state until the system power supplies have stabilized at appropriate levels and the oscillator clock is ready. When the oscillator clock is ready, the processor begins execution from the Reset location. The user application programs a `GOTO` instruction at the Reset address, which redirects program execution to the appropriate start-up routine.

When the device exits the Reset condition (begins normal operation), the device operating parameters (voltage, frequency, temperature, etc.) must be within their operating ranges, otherwise the device may not function correctly.

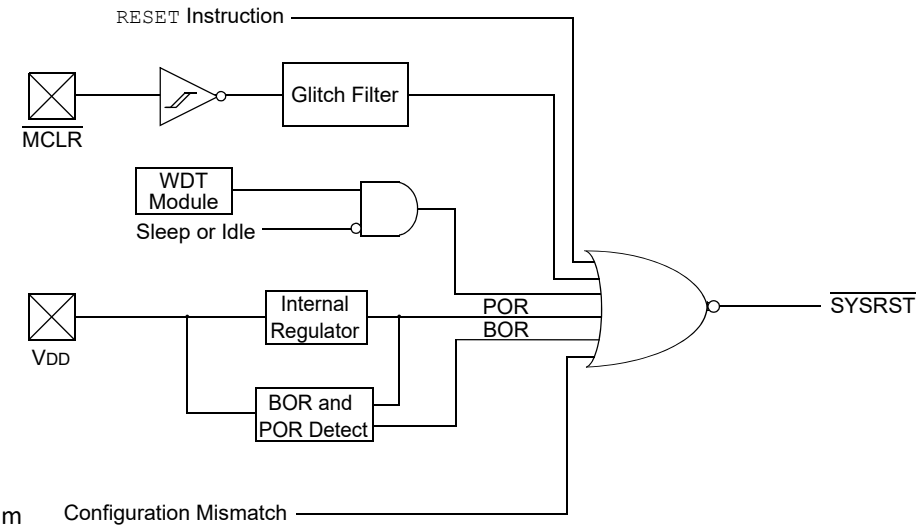
The Resets module combines all Reset sources and controls the system level Reset signal SYSRST. The following is a list of device Reset sources:

- Power-on Reset (POR)
- Brown-out Reset (BOR)
- Master Clear Reset (MCLR)
- Watchdog Time-out Reset (WDT)
- Software Reset (SWR)
- Configuration Mismatch Reset (CM)

9.1 Architectural Overview

A simplified block diagram of the Reset module is shown in Figure 9-1. Any active source of Reset will make the system Reset signal active. Many registers associated with the CPU and peripherals are forced to a known “Reset state.” Most registers are unaffected by a Reset; their status is unknown on POR and unchanged by all other Resets.

Figure 9-1. System Reset Block



Diagram

Note: Multiple regulators may be instantiated with discrete status bits.

9.2 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3198	RCON	31:24								
		23:16			VREG3R	VREG2R	VREG1R			
		15:8							CM	
		7:0	EXTR	SWR		WDTO	SLEEP	IDLE	BOR	POR

9.2.1 Reset Control Register

Name: RCON
Offset: 0x3198

Legend: R = Readable bit, C = Clearable bit, HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			VREG3R	VREG2R	VREG1R			
Access			R/C/HS	R/C/HS	R/C/HS			
Reset			0	0	0			

Bit	15	14	13	12	11	10	9	8
							CM	
Access							R/C/HS	
Reset							0	

Bit	7	6	5	4	3	2	1	0
	EXTR	SWR		WDTO	SLEEP	IDLE	BOR	POR
Access	R/C/HS	R/C/HS		R/C/HS	R/C/HS	R/C/HS	R/C/HS	R/C/HS
Reset	0	0		0	0	0	1	1

Bit 21 – VREG3R Voltage Domain 3 Regulation Loss Flag bit

Value	Description
1	Voltage Domain 3 lost voltage regulation
0	Voltage Regulation in Domain 3 has not been lost

Bit 20 – VREG2R Voltage Domain 2 Regulation Loss Flag bit

Value	Description
1	Voltage Domain 2 lost voltage regulation
0	Voltage Regulation in Domain 2 has not been lost

Bit 19 – VREG1R Voltage Domain 1 Regulation Loss Flag bit

Value	Description
1	Voltage Domain 1 lost voltage regulation
0	Voltage Regulation in Domain 1 has not been lost

Bit 9 – CM Configuration Mismatch Flag bit

Value	Description
1	A Configuration Mismatch Reset has occurred.
0	A Configuration Mismatch Reset has not occurred

Bit 7 – EXTR External Reset ($\overline{MCLR}$) Pin bit

Value	Description
1	A Master Clear (pin) Reset has occurred
0	A Master Clear (pin) Reset has not occurred

Bit 6 – SWR Software RESET (Instruction) Flag bit

Value	Description
1	A RESET instruction has been executed
0	A RESET instruction has not been executed

Bit 4 – WDTO Watchdog Timer Time-out Flag bit

Value	Description
1	Device reset has occurred due to WDT time-out
0	WDT time-out has not occurred

Bit 3 – SLEEP Wake-up from Sleep Flag bit

Value	Description
1	Device has been in Sleep mode
0	Device has not been in Sleep mode

Bit 2 – IDLE Wake-up from Idle Flag bit

Value	Description
1	Device has been in Idle mode
0	Device has not been in Idle mode

Bit 1 – BOR Brown-out Reset Flag bit

Value	Description
1	A Brown-out Reset has occurred
0	A Brown-out Reset has not occurred. Set by hardware at detection of a BOR event

Bit 0 – POR Power-on Reset Flag bit

Value	Description
1	A Power-on Reset has occurred
0	A Power-on Reset has not occurred. Set by hardware at detection of a POR event

9.3 Operation

9.3.1 System Reset (SYSRST)

The dsPIC33A Internal System Reset (SYSRST) can be generated from multiple Reset sources, such as:

- Power-on Reset (POR)
- Brown-out Reset (BOR)
- Master Clear Reset (MCLR)
- Watchdog Time-out Reset (WDT) or Trap
- Software Reset (SWR)
- Configuration Mismatch Reset (CM)

A system Reset is active at the first POR and asserted until device configuration settings are loaded and the oscillator clock sources become stable. The system Reset is then deasserted allowing the CPU to start fetching code after eight system clock cycles (SYSCLK).

BOR, $\overline{\text{MCLR}}$ and WDTO Resets are asynchronous events, and to avoid SFR and RAM corruptions, the system Reset is synchronized with the system clock. All other Reset events are synchronous.

9.3.2 Power-On Reset (POR)

A Power-On Reset (POR) circuit ensures the device is reset from power-on. The POR circuit is active until V_{DD} crosses the V_{POR} threshold.

A power-on event generates an internal POR pulse when a V_{DD} rise is detected above V_{POR}. The device supply voltage characteristics must meet the specified starting voltage and rise rate requirements to generate the POR pulse. In particular, V_{DD} must fall below V_{POR} before a new POR is initiated. For more information on the V_{POR} and V_{DD} rise-rate specifications, refer to the **"Electrical Characteristics"** chapter. A POR event can also be generated when the 1.1 V core voltage drops below a safe operating condition.

The POR bit in the Reset Control (RCON[0]) register is set to indicate the Power-On Reset.

Note: When the device exits the Reset condition (begins normal operation), the device operating parameters (voltage, frequency, temperature, etc.) must be within their operating ranges; otherwise, the device will not function correctly. The user software must ensure that the delay between the time power is first applied and the time the system Reset is released is adequate to get all operating parameters within the specification.

9.3.3 Master Clear Reset ($\overline{MCLR}$)

Whenever the master clear pin ($\overline{MCLR}$) is driven low, the Reset event is synchronized with the System Clock (SYSCLK) before asserting the System Reset (SYSRST), provided the input pulse on $\overline{MCLR}$ is longer than a certain minimum width, as specified in the **"Electrical Characteristics"** chapter.

The $\overline{MCLR}$ pin provides a filter to minimize the effects of noise and to avoid unwanted Reset events. The Status bit, EXTR (RCON[7]), is set to indicate the $\overline{MCLR}$ Reset.

9.3.4 Software Reset (SWR)

Whenever the `RESET` instruction is executed, the device will enter a warm Reset state. This Reset state will not reinitialize the clock. The clock source in effect prior to the `RESET` instruction will remain in effect. The device will be released from a Reset state at the next instruction cycle, and the Reset vector fetch will commence.

The Software Reset (Instruction) Flag bit (SWR) in the Reset Control register (RCON[6]) is set to indicate the software Reset.

9.3.5 Watchdog Timer Reset (WDT)

Whenever a watchdog time-out occurs, the device will asynchronously assert SYSRT and the clock source will remain unchanged. A WDT time-out during Sleep or Idle mode will wake-up the processor, but will not reset the processor.

The Watchdog Time-out Flag bit (WDT) in the Reset Control register (RCON[4]) is set to indicate the watchdog Reset.

9.3.6 Brown-out Reset (BOR)

The BOR module generates a device Reset when a brown-out condition occurs to protect against code mis-execution. The Brown-out Reset (BOR) module is based on an internal voltage reference circuit that monitors V_{DD}. Brown-out conditions are generally caused by glitches on the AC mains (for example, missing portions of the AC cycle waveform due to bad power transmission lines or voltage sags due to excessive current draw when a large inductive load is turned on).

A BOR generates a Reset pulse which resets the device. The BOR status bit (RCON[1]) is set to indicate that a BOR has occurred. The BOR circuit continues to operate while in Sleep or Idle mode and resets the device should V_{DD} fall below the BOR threshold voltage. The BOR threshold voltage is detailed in the 37. Electrical Characteristics section.

9.3.7 On-Chip Voltage Regulators

dsPIC33AK128MC106 family devices have multiple internal voltage regulators to supply power to the core at 1.1V (typical) and to other system resources.

There are three core voltage domains to provide isolation of power supplies to minimize noise to peripherals. Each of the followings domains have their own linear regulator configured to 1.1V.

- VREG1 -- CPU core and system logic
- VREG2 -- ADC
- VREG3 -- PLLs and FRC

In the event of a device Reset due to the voltage regulators, the RCON[21:19] register bits will report which domain lost voltage regulation.

9.3.8 Configuration Mismatch Reset (CM)

To maintain the integrity of the stored configuration values, all device Configuration bits are loaded and implemented as a complementary set of bits. As the Configuration Words are being loaded, for each bit loaded as '1', a complementary value of '0' is stored into its corresponding background word location and vice versa. The bit pairs are compared every time the Configuration Words are loaded, including Sleep mode. During this comparison, if the Configuration bit values are not found opposite to each other, a configuration mismatch event is generated, which causes a device Reset.

If a device Reset occurs as a result of a configuration mismatch, the CM Status bit (RCON[9]) is set.

9.3.9 Using the RCON Status Bits

The user software can read the RCON register after any system Reset to determine the cause of the Reset. Table 9-1 provides a summary of the Reset flag bit operation.

Note: The Status bits in the RCON register should be cleared after they are read so that the next RCON register value after a device Reset will be meaningful.

Table 9-1. Reset Flag Bit Operation

Flag Bit(1)	Set by	Cleared by
POR (RCON[0])	POR	User Software
BOR (RCON[1])	POR, BOR	User Software
IDLE (RCON[2])	`PWRSAV #IDLE` instruction	User Software, POR, BOR
SLEEP (RCON[3])	`PWRSAV #SLEEP` instruction	User Software, POR, BOR
WDTO (RCON[4])	WDT time-out	User Software, POR, BOR
SWR (RCON[6])	`Software Reset` command	User Software, POR, BOR
EXTR (RCON[7])	MCLR Reset	User Software, POR, BOR
CM (RCON[9])	Configuration mismatch	User Software, POR, BOR
VREGxR (RCON[21:19])	Voltage Domain Regulation lost	User Software

Note:
1. All Reset flag bits may be set or cleared by the user software.

9.4 Application Examples

After a device Reset, the RCON register can be examined by initialization code to confirm the source of the Reset. In certain applications, this information can be used to take appropriate action to correct the problem that caused the Reset to occur.

All Reset status bits in the RCON register should be cleared after reading them to ensure the RCON value will provide meaningful results after the next device Reset.

Example 9-1 illustrates how to determine the source of device Reset using the RCON register.

Example 9-1. Determining the Source of Device Reset

```
int main(void)
{
//... perform application specific startup tasks
// next, check the cause of the Reset
if(RCON & 0x0003)
{
// execute a Power-on Reset handler
// ...
}
else if(RCON & 0x0002)
{
// execute a Brown-out Reset handler
// ...
}
else if(RCON & 0x0080)
{
// execute a Master Clear Reset handler
// ...
}
else if(RCON & 0x0040)
{
// execute a Software Reset handler
// ...
}
else if (RCON & 0x0200)
{
// execute a Configuration Mismatch Reset handler
// ...
}
else if (RCON & 0x0010)
{
// execute Watchdog Time-out Reset handler
// ...
}
// ... perform other application-specific tasks
while(1);
}
```

9.5 Effects of Reset

The Reset value for the Reset Control register, RCON, will depend on the type of device Reset, as indicated in Table 9-2.

Table 9-2. Status Bits, Their Significance and the Initialization Condition for RCON Register

Condition	Program Counter	EXTR	SWR	WDTO	SLEEP	IDLE	CM	BOR	POR
Power-on Reset or $\overline{MCLR}$ set as POR	0x00000000_0000	0	0	0	0	0	0	1	1
Brown-out Reset		0	0	0	0	0	0	1	u
$\overline{MCLR}$ Reset during Run Mode		1	u	u	u	u	u	u	u
$\overline{MCLR}$ Reset during Idle Mode		1	u	u	u	1	u	u	u
$\overline{MCLR}$ Reset during Sleep Mode		1	u	u	1	u	u	u	u
Software Reset Command		u	1	u	u	u	u	u	u
Configuration Word Mismatch Reset		u	u	u	u	u	1	u	u
WDT Time-out Reset during Run Mode		u	u	0	u	u	u	u	u

.....continued

Condition	Program Counter	EXTR	SWR	WDTO	SLEEP	IDLE	CM	BOR	POR
WDT Time-out Reset during Idle Mode	PC+2	u	u	0	u	1	u	u	u
WDT Time-out Reset during Sleep Mode		u	u	1	1	u	u	u	u
Interrupt Exit from Idle Mode	PC+2 or Interrupt Vector	u	u	u	u	1	u	u	u
Interrupt Exit from Sleep Mode		u	u	u	1	u	u	u	u

Legend: u = unchanged

Note: The Program Counter (PC) is loaded with PC + 2 if the interrupt priority is less than or equal to the CPU interrupt priority level. The PC is loaded with the hardware vector address if the interrupt priority is greater than the CPU interrupt priority level.

9.5.1 Special Function Register (SFR) Reset States

Most of the SFRs associated with the dsPIC33A CPU and peripherals are reset to a particular value at a device Reset. This also applies to a Reset due to Run mode WDT timeout, which is treated as a full device Reset by the dsPIC33A CPU and peripherals. Refer to register details of the specific peripheral or system module for SFR Reset values.

The Reset value for the Reset Control register, RCON, will depend on the type of device Reset, as described in Table 9-2.

9.5.2 Configuration Word Register Reset States

All Reset conditions force the configuration settings to be reloaded. The POR sets all the Configuration Word register locations to a '1' before loading the configuration settings. For all other Reset conditions, the Configuration Word register locations are not reset prior to being reloaded. This difference in behavior accommodates $\overline{\text{MCLR}}$ assertions during Debug mode without affecting the state of the debug operations.

Independent of the source of a Reset, the system clock is always reloaded and is specified by the FNOSC[2:0] bits (DEVCFG[2:0]). When the device is executing code, the user software may change the primary system clock source by using the OSCCON register.

10. Interrupt Controller

The dsPIC33AK128MC106 family interrupt controller reduces the numerous peripheral interrupt request signals to a single interrupt request signal to the dsPIC33AK128MC106 family CPU. The core supports a prioritized interrupt and trap exception scheme.

The interrupt controller has the following features:

- Interrupt Vector Tables (IVT) for User memory
- Parameterized Number of Interrupt Sources
- Reset vector (not part of IVT)
- 8 processor traps
- Up to 31 generic traps + 1 software trap
- 7 user selectable priority levels
- A unique vector for each interrupt or exception in full IVT mode
- A collapsed vector (logical OR) for all peripheral Interrupts
- Fixed priority within a specified user priority level
- Predictable latencies
- Software can generate any peripheral interrupt
- Alternate IVT availability via IVTBASE
- Support for Testability via INTTREG

10.1 Device-Specific Information

Table 10-1. Interrupt Vector Details

IRQ #	Interrupt Source	MPLAB® ISR Name
0	Common collapsed interrupt	_COMMONInterrupt
1	CPU and FPU	_CPUFPUInterrupt
2	X RAM ECC single error	_XRAMECCInterrupt
3	Y RAM ECC single error	_YRAMECCInterrupt
4	PBU parity error	_PBUEInterrupt
5	NVM ECC single error	_NVMECCInterrupt
6	NVM program or erase operation completed	_NVMInterrupt
7	NVM CRC operation completed	_NVMCRCInterrupt
8	Reserved	—
9	Combined clock fail	_CLKFInterrupt
10	Combined clock error	_CLKEInterrupt
11	Clock 1 failure	_CLK1FInterrupt
12	Clock 1 warming complete	_CLK1WInterrupt
13	Clock 1 monitor error	_CLK1MInterrupt
14	Clock 1 ready	_CLK1RInterrupt
15	Clock 2 failure	_CLK2FInterrupt
16	Clock 2 warming complete	_CLK2WInterrupt
17	Clock 2 monitor error	_CLK2MInterrupt
18	Clock 2 ready	_CLK2RInterrupt
19	Clock 3 failure	_CLK3FInterrupt

.....continued

IRQ #	Interrupt Source	MPLAB® ISR Name
20	Clock 3 warming complete	_CLK3WInterrupt
21	Clock 3 monitor error	_CLK3MInterrupt
22	Clock 3 ready	_CLK3RInterrupt
23	Clock 4 failure	_CLK4FInterrupt
24	Clock 4 warming complete	_CLK4WInterrupt
25	Clock 4 monitor error	_CLK4MInterrupt
26	Clock 4 ready	_CLK4RInterrupt
27	Reserved	—
28	Wake up from WDT	_WDTInterrupt
29-32	Reserved	—
33	External Interrupt 0	_INT0Interrupt
34	External Interrupt 1	_INT1Interrupt
35	External Interrupt 2	_INT2Interrupt
36	External Interrupt 3	_INT3Interrupt
37	External Interrupt 4	_INT4Interrupt
38	PWM EVENT A	_PEVTAInterrupt
39	PWM EVENT B	_PEVTBInterrupt
40	PWM EVENT C	_PEVTCInterrupt
41	PWM EVENT D	_PEVTDInterrupt
42	PWM EVENT E	_PEVTEInterrupt
43	PWM EVENT F	_PEVTFInterrupt
44	PWM Generator 1	_PWM1Interrupt
45	PWM Generator 2	_PWM2Interrupt
46	PWM Generator 3	_PWM3Interrupt
47	PWM Generator 4	_PWM4Interrupt
48	Timer 1 Interrupt	_T1Interrupt
49	CCP 1 timer	_CCT1Interrupt
50	CCP 1 input capture or output compare	_CCP1Interrupt
51	CCP 2 timer	_CCT2Interrupt
52	CCP 2 input capture or output compare	_CCP2Interrupt
53	CCP 3 timer	_CCT3Interrupt
54	CCP 3 input capture or output compare	_CCP3Interrupt
55	CCP 4 timer	_CCT4Interrupt
56	CCP 4 input capture or output compare	_CCP4Interrupt
57-62	Reserved	—
63	SPI 1 Rx	_SPI1RXInterrupt
64	SPI 1 Tx	_SPI1TXInterrupt
65	SPI 1 General	_SPI1GInterrupt
66	SPI 2 Rx	_SPI2RXInterrupt
67	SPI 2 Tx	_SPI2TXInterrupt
68	SPI 2 General	_SPI2GInterrupt
69	SPI 3 Rx	_SPI3RXInterrupt
70	SPI 3 Tx	_SPI3TXInterrupt
71	SPI 3 General	_SPI3GInterrupt
72	DMA Channel 0	_DMA0Interrupt

.....continued

IRQ #	Interrupt Source	MPLAB® ISR Name
73	DMA Channel 1	_DMA1Interrupt
74	DMA Channel 2	_DMA2Interrupt
75	DMA Channel 3	_DMA3Interrupt
76	Analog Comparator Interrupt 1	_CMP1Interrupt
77	Analog Comparator Interrupt 2	_CMP2Interrupt
78	Analog Comparator Interrupt 3	_CMP3Interrupt
79	Reserved	—
80	I2C 1 Error Interrupt	_I2C1EInterrupt
81	I2C 1 General Interrupt	_I2C1Interrupt
82	I2C 1 RX Buffer Full Interrupt	_I2C1RXInterrupt
83	I2C 1 TX Buffer Full Interrupt	_I2C1TXInterrupt
84	I2C 2 Error Interrupt	_I2C2EInterrupt
85	I2C 2 General Interrupt	_I2C2Interrupt
86	I2C 2 RX Buffer Full Interrupt	_I2C2RXInterrupt
87	I2C 2 TX Buffer Full Interrupt	_I2C2TXInterrupt
88	Reserved	—
89	UART 1 RX	_U1RXInterrupt
90	UART 1 TX	_U1TXInterrupt
91	UART1 error	_U1EInterrupt
92	UART 1 event	_U1EVTInterrupt
93	UART 2 RX	_U2RXInterrupt
94	UART 2 TX	_U2TXInterrupt
95	UART 2 error	_U2EInterrupt
96	UART 2 event	_U2EVTInterrupt
97	UART 3 RX	_U3RXInterrupt
98	UART 3 TX	_U3TXInterrupt
99	UART 3 error	_U3EInterrupt
100	UART 3 event	_U3EVTInterrupt
101-104	Reserved	—
105	SENT 1 RX and TX	_SENT1Interrupt
106	SENT 1 error	_SENT1EInterrupt
107	SENT 2 RX and TX	_SENT2Interrupt
108	SENT 2 error	_SENT2EInterrupt
109	DMA 4 Interrupt	_DMA4Interrupt
110	DMA 5 Interrupt	_DMA5Interrupt
111-112	Reserved	—
113	Change notice A	_CNAInterrupt
114	Change notice B	_CNBInterrupt
115	Change notice C	_CNCInterrupt
116	Change notice D	_CNDInterrupt
117-124	Reserved	—
125	QEI 1 position counter compare	_QEI1Interrupt
126-128	Reserved	—
129	BISS 1 Transmission Error Interrupt	_BISS1EInterrupt
130	BISS 1 Transmission Finished Interrupt	_BISS1Interrupt

.....continued

IRQ #	Interrupt Source	MPLAB® ISR Name
131	CRC	_CRCInterrupt
132	ICD - ICD App In	—
133	Reserved	—
134	PTG Step	_PTGSTEPInterrupt
135	PTG WDT	_PTGWDTInterrupt
136	PTG trigger 0	_PTG0Interrupt
137	PTG trigger 1	_PTG1Interrupt
138	PTG trigger 2	_PTG2Interrupt
139	PTG trigger 3	_PTG3Interrupt
140-145	Reserved	—
146	ADC 1 data channel 0 done	_AD1CH0Interrupt
147	ADC 1 digital comparator 0	_AD1CMP0Interrupt
148	ADC 1 data channel 1 done	_AD1CH1Interrupt
149	ADC 1 digital comparator 1	_AD1CMP1Interrupt
150	ADC 1 data channel 2 done	_AD1CH2Interrupt
151	ADC 1 digital comparator 2	_AD1CMP2Interrupt
152	ADC 1 data channel 3 done	_AD1CH3Interrupt
153	ADC 1 digital comparator 3	_AD1CMP3Interrupt
154	ADC 1 data channel 4 done	_AD1CH4Interrupt
155	ADC 1 digital comparator 4	_AD1CMP4Interrupt
156	ADC 1 data channel 5 done	_AD1CH5Interrupt
157	ADC 1 digital comparator 5	_AD1CMP5Interrupt
158	ADC 1 data channel 6 done	_AD1CH6Interrupt
159	ADC 1 digital comparator 6	_AD1CMP6Interrupt
160	ADC 1 data channel 7 done	_AD1CH7Interrupt
161	ADC 1 digital comparator 7	_AD1CMP7Interrupt
162	ADC 1 data channel 8 done	_AD1CH8Interrupt
163	ADC 1 digital comparator 8	_AD1CMP8Interrupt
164	ADC 1 data channel 9 done	_AD1CH9Interrupt
165	ADC 1 digital comparator 9	_AD1CMP9Interrupt
166	ADC 1 data channel 10 done	_AD1CH10Interrupt
167	ADC 1 digital comparator 10	_AD1CMP10Interrupt
168	ADC 1 data channel 11 done	_AD1CH11Interrupt
169	ADC 1 digital comparator 11	_AD1CMP11Interrupt
170	ADC 1 data channel 12 done	_AD1CH12Interrupt
171	ADC 1 digital comparator 12	_AD1CMP12Interrupt
172	ADC 1 data channel 13 done	_AD1CH13Interrupt
173	ADC 1 digital comparator 13	_AD1CMP13Interrupt
174	ADC 1 data channel 14 done	_AD1CH14Interrupt
175	ADC 1 digital comparator 14	_AD1CMP14Interrupt
176	ADC 1 data channel 15 done	_AD1CH15Interrupt
177	ADC 1 digital comparator 15	_AD1CMP15Interrupt
178	ADC 1 data channel 16 done	_AD1CH16Interrupt
179	ADC 1 digital comparator 16	_AD1CMP16Interrupt
180	ADC 1 data channel 17 done	_AD1CH17Interrupt

.....continued

IRQ #	Interrupt Source	MPLAB® ISR Name
181	ADC 1 digital comparator 17	_AD1CMP17Interrupt
182	ADC 1 data channel 18 done	_AD1CH18Interrupt
183	ADC 1 digital comparator 18	_AD1CMP18Interrupt
184	ADC 1 data channel 19 done	_AD1CH19Interrupt
185	ADC 1 digital comparator 19	_AD1CMP19Interrupt
186-187	Reserved	—
188	ADC 2 data channel 0 done	_AD2CH0Interrupt
189	ADC 2 digital comparator 0	_AD2CMP0Interrupt
190	ADC 2 data channel 1 done	_AD2CH1Interrupt
191	ADC 2 digital comparator 1	_AD2CMP1Interrupt
192	ADC 2 data channel 2 done	_AD2CH2Interrupt
193	ADC 2 digital comparator 2	_AD2CMP2Interrupt
194	ADC 2 data channel 3 done	_AD2CH3Interrupt
195	ADC 2 digital comparator 3	_AD2CMP3Interrupt
196	ADC 2 data channel 4 done	_AD2CH4Interrupt
197	ADC 2 digital comparator 4	_AD2CMP4Interrupt
198	ADC 2 data channel 5 done	_AD2CH5Interrupt
199	ADC 2 digital comparator 5	_AD2CMP5Interrupt
200	ADC 2 data channel 6 done	_AD2CH6Interrupt
201	ADC 2 digital comparator 6	_AD2CMP6Interrupt
202	ADC 2 data channel 7 done	_AD2CH7Interrupt
203	ADC 2 digital comparator 7	_AD2CMP7Interrupt
204	ADC 2 data channel 8 done	_AD2CH8Interrupt
205	ADC 2 digital comparator 8	_AD2CMP8Interrupt
206	ADC 2 data channel 9 done	_AD2CH9Interrupt
207	ADC 2 digital comparator 9	_AD2CMP9Interrupt
208	ADC 2 data channel 10 done	_AD2CH10Interrupt
209	ADC 2 digital comparator 10	_AD2CMP10Interrupt
210	ADC 2 data channel 11 done	_AD2CH11Interrupt
211	ADC 2 digital comparator 11	_AD2CMP11Interrupt
212	ADC 2 data channel 12 done	_AD2CH12Interrupt
213	ADC 2 digital comparator 12	_AD2CMP12Interrupt
214	ADC 2 data channel 13 done	_AD2CH13Interrupt
215	ADC 2 digital comparator 13	_AD2CMP13Interrupt
216	ADC 2 data channel 14 done	_AD2CH14Interrupt
217	ADC 2 digital comparator 14	_AD2CMP14Interrupt
218	ADC 2 data channel 15 done	_AD2CH15Interrupt
219	ADC 2 digital comparator 15	_AD2CMP15Interrupt
220	ADC 2 data channel 16 done	_AD2CH16Interrupt
221	ADC 2 digital comparator 16	_AD2CMP16Interrupt
222	ADC 2 data channel 17 done	_AD2CH17Interrupt
223	ADC 2 digital comparator 17	_AD2CMP17Interrupt
224	ADC 2 data channel 18 done	_AD2CH18Interrupt
225	ADC 2 digital comparator 18	_AD2CMP18Interrupt
226	ADC 2 data channel 19 done	_AD2CH19Interrupt

.....continued		
IRQ #	Interrupt Source	MPLAB® ISR Name
227	ADC 2 digital comparator 19	_AD2CMP19Interrupt
228-231	Reserved	—
232	CLC 1 positive edge	_CLC1PInterrupt
233	CLC 1 negative edge	_CLC1NInterrupt
234	CLC 2 positive edge	_CLC2PInterrupt
235	CLC 2 negative edge	_CLC2NInterrupt
236	CLC 3 positive edge	_CLC3PInterrupt
237	CLC 3 negative edge	_CLC3NInterrupt
238	CLC 4 positive edge	_CLC4PInterrupt
239	CLC 4 negative edge	_CLC4NInterrupt
240-274	Reserved	—
275	GPIO integrity monitor 1	_IOIM1Interrupt
276	GPIO integrity monitor 2	_IOIM2Interrupt
277	GPIO integrity monitor 3	_IOIM3Interrupt
278	GPIO integrity monitor 4	_IOIM4Interrupt
279-501	Reserved	—

10.2 Architectural Overview

The interrupt controller module assembles all the interrupt request signals from the peripherals and assigns both a fixed “natural order” priority and a user assigned priority to each signal. The highest level unmasked interrupt request is then presented to the processor core along with a vector number, which represents an offset into the Interrupt Vector Table.

The interrupt controller provides up to 502 interrupt sources (unused sources are reserved for future use) that can be programmed with different priority levels along with eight processor traps and other generic traps.

10.2.1 System Traps and Interrupts

- CPU
  - Address error exception
  - Stack error exception
  - Math error trap due to arithmetic divide by zero
  - Math error trap due to arithmetic Accumulator A Overflow
  - Math error trap due to arithmetic Accumulator B Overflow
  - Math error trap due to arithmetic Accumulator A Catastrophic Overflow
  - Math error trap due to arithmetic Accumulator B Catastrophic Overflow
  - Math error trap due to arithmetic attempted out of range SFTAC
  - Program memory bus error
  - X-space read or write bus error
  - Y-space read or write bus error
  - Illegal instruction trap
- NVM Controller
  - SEC interrupt
  - Erase programming complete/error interrupt CRC done

- DMA
  - DMA bus error trap
- ICD
  - ICD bus error
- DMT
  - DMT event generic trap
- WDT
  - WDT Sleep/Idle interrupt
  - WDT Run event generic trap
- PBU Cache
  - Cache parity error interrupt
- XRAM Controller
  - Interrupt event ECC SEC
  - Interrupt PWB error
- YRAM Controller
  - Interrupt event ECC SEC
  - Interrupt PWB error

The interrupt controller is responsible for pre-processing the peripheral interrupts and processor exceptions prior to them being presented to the processor core. The interrupts and traps are enabled, prioritized and controlled using centralized special function registers.

10.3 Interrupt Vector Table

The Interrupt Vector Table (IVT) resides in the program memory. The IVT contains up to 502 interrupt vectors plus six processor trap vectors. In general, each interrupt source has its own vector. Each interrupt vector contains a 24-bit wide address. The value programmed into each interrupt vector location is the starting address of the associated Interrupt Service Routine (ISR). See Table 10-1 for interrupt vector details.

The processor core is responsible for performing the interrupt bus cycle: the reading of the IVT and transferring the address contained in the interrupt vector to the program counter. The interrupt vector is transferred from the program data bus into the program counter via a 24-bit wide multiplexer on the input of the program counter.

The peripheral IVT is relocatable using a SFR (IVTBASE) to set the base address. At a device Reset, the base address value is “0x800000”. The IVT starts at address 0x800000, and code execution begins at the address specified in the Reset vector at 0x800000. The IVT is shown in Figure 10-1.

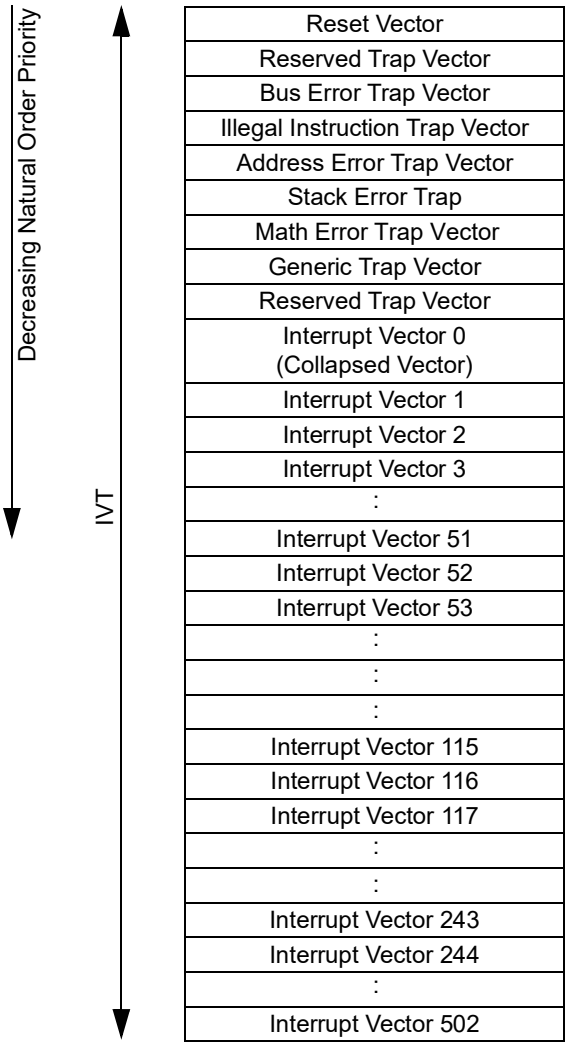
The user can choose an alternate IVT table based on IVTBASE value. This allows the erasing and reprogramming of user code without affecting the vector table. This is useful in various software update scenarios.

10.3.1 Remappable Interrupt Table

The interrupt controller has the capability to relocate the Interrupt Vector Table location using the IVTBASE register. This feature is referred to as the Remappable Interrupt Vector Table (RIVT). The RIVT proves beneficial in instances where there is a necessity to modify the current vector table, such as updating vector address values, or to redefine the vectors for alternative uses, including debugging processes or different application programs.

Additionally, IVT can be relocated anywhere within Flash/RAM by defining IVT in a new location and modifying the IVTBASE register accordingly.

Figure 10-1. Interrupt Vector Table (IVTC = 0, Default)

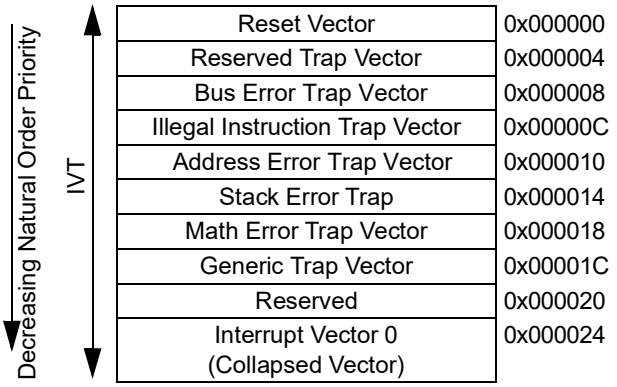


10.3.2 Collapsed Interrupt Vector

The IVT (Interrupt Vector Table) can be collapsed by configuring the IVTC bit within the IVTCREG register. While trap handlers remain unaffected by this setting, all peripheral interrupts are directed to a singular location that follows trap vectors. When the IVTC bit is set to 1, all peripheral interrupts utilize a common vector, which is located at the offset address 0x000024.

The collapsed peripheral interrupt vector is placed in the reserved interrupt location. The TRAP interrupts are not collapsed and the peripheral interrupts are pointed to one location, which is placed after the TRAP's interrupt location, as shown in Figure 10-2.

Figure 10-2. Interrupt Vector Table (IVTC = 1)



The potential application of RIVT and CIVT is in scenarios where the processor is operating within a secure or boot memory segment and encounters a trap or an interrupt. In such cases, the secure boot software is required to assign the interrupt base address to a designated interrupt vector address located within the secure or boot memory segment. Upon completion of its operations, the secure boot software will then assign predetermined values to specific memory segments. Additionally, the secure boot software has the option to set the Interrupt Vector Table Collapse (IVTC) to "1", thereby consolidating the interrupt vector for all peripherals into a single entry. Once the secure boot software has finalized its processes, it may reset this bit, effectively deactivating the collapsed IVT feature.

10.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x70	INTCON1	31:24	NSTDIS							
		23:16								
		15:8	GIE							
		7:0				STKERR	ADDRERR	BADOPERR		
0x74	INTCON2	31:24								
		23:16								
		15:8								
		7:0				INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
0x78	INTCON3	31:24								
		23:16								
		15:8								
		7:0					CPUBET	DMABET	YRAMBET	XRAMBET
0x7C	INTCON4	31:24								
		23:16			OVATE	OVBTE	COVTE			
		15:8								
		7:0			OVAERR	OVBERR	COVAERR	COVBERR	SFTACERR	DIV0ERR
0x80	INTCON5	31:24	SOFT							
		23:16								
		15:8								
		7:0					YPWBDED	XPWBDED	WDTE	DMTE
0x84	INTTREG	31:24								
		23:16	IRQCPU		VHOLD					
		15:8			ILR[3:0]					VECNUM[8]
		7:0	VECNUM[7:0]							
0x88	IVTBASE	31:24	IVTBASE[31:24]							
		23:16	IVTBASE[23:16]							
		15:8	IVTBASE[15:8]							
		7:0	IVTBASE[7:6]							
0x8C	IVTCREG	31:24								
		23:16								
		15:8								
		7:0								IVTC
0x90	IFS0	31:24				WDTIF		C4RDYIF	C4MONIF	C4WARMIF
		23:16	C4FAILIF	C3RDYIF	C3MONIF	C3WARMIF	C3FAILIF	C2RDYIF	C2MONIF	C2WARMIF
		15:8	C2FAILIF	C1RDYIF	C1MONIF	C1WARMIF	C1FAIL1IF	CLKERRIF	CLKFAILIF	
		7:0	NVMCRCIF	NVMIF	NVMECCIF	PBERRIF	YRAMECCIF	XRAMECCIF	CPUFPUIF	
0x94	IFS1	31:24	SPI1RXIF							CCP4IF
		23:16	CCT4IF	CCP3IF	CCT3IF	CCP2IF	CCT2IF	CCP1IF	CCT1IF	T1IF
		15:8	PCG4IF	PCG3IF	PCG2IF	PCG1IF	PEVTFIF	PEVTEIF	PEVTDIF	PEVTCIF
		7:0	PEVTBIF	PEVTAIF	INT4IF	INT3IF	INT2IF	INT1IF	INT0IF	
0x98	IFS2	31:24	U2EIF	U2TXIF	U2RXIF	U1EVTIF	U1EIF	U1TXIF	U1RXIF	
		23:16	I2C2TXIF	I2C2RXIF	I2C2IF	I2C2EIF	I2C1TXIF	I2C1RXIF	I2C1IF	I2C1EIF
		15:8		CMP3IF	CMP2IF	CMP1IF	DMA3IF	DMA2IF	DMA1IF	DMA0IF
		7:0	SPI3GIF	SPI3TXIF	SPI3RXIF	SPI2GIF	SPI2TXIF	SPI2RXIF	SPI1GIF	SPI1TXIF
0x9C	IFS3	31:24			QEI1IF					
		23:16				CNDIF	CNCIF	CNBIF	CNAIF	
		15:8		DMA5IF	DMA4IF	SENT2EIF	SENT2IF	SENT1EIF	SENT1IF	PWM6IF
		7:0	U4EVTIF			U3EVTIF	U3EIF	U3TXIF	U3RXIF	U2EVTIF
0xA0	IFS4	31:24	AD1CMP6IF	AD1CH6IF	AD1CMP5IF	AD1CH5IF	AD1CMP4IF	AD1CH4IF	AD1CMP3IF	AD1CH3IF
		23:16	AD1CMP2IF	AD1CH2IF	AD1CMP1IF	AD1CH1IF	AD1CMP0IF	AD1CH0IF		
		15:8					PTG3IF	PTG2IF	PTG1IF	PTG0IF
		7:0	PTGWDTIF	PTGSTEPIF		ICDIF	CRCIF	BISS1EIF		
0xA4	IFS5	31:24	AD2CMP1IF	AD2CH1IF	AD2CMP0IF	AD2CH0IF			AD1CMP19IF	AD1CH19IF
		23:16	AD1CMP18IF	AD1CH18IF	AD1CMP17IF	AD1CH17IF	AD1CMP16IF	AD1CH16IF	AD1CMP15IF	AD1CH15IF
		15:8	AD1CMP14IF	AD1CH14IF	AD1CMP13IF	AD1CH13IF	AD1CMP12IF	AD1CH12IF	AD1CMP11IF	AD1CH11IF
		7:0	AD1CMP10IF	AD1CH10IF	AD1CMP9IF	AD1CH9IF	AD1CMP8IF	AD1CH8IF	AD1CMP7IF	AD1CH7IF

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0xA8	IFS6	31:24	AD2CMP17IF	AD2CH17IF	AD2CMP16IF	AD2CH16IF	AD2CMP15IF	AD2CH15IF	AD2CMP14IF	AD2CH14IF
		23:16	AD2CMP13IF	AD2CH13IF	AD2CMP12IF	AD2CH12IF	AD2CMP11IF	AD2CH11IF	AD2CMP10IF	AD2CH10IF
		15:8	AD2CMP9IF	AD2CH9IF	AD2CMP8IF	AD2CH8IF	AD2CMP7IF	AD2CH11IF	AD2CMP10IF	AD2CH10IF
		7:0	AD2CMP5IF	AD2CH5IF	AD2CMP4IF	AD2CH4IF	AD2CMP4IF	AD2CH3IF	AD2CMP2IF	AD2CH2IF
0xAC	IFS7	31:24								
		23:16								
		15:8	CLC4NIF	CLC4PIF	CLC3NIF	CLC3PIF	CLC2NIF	CLC2PIF	CLC1NIF	CLC1PIF
		7:0					ADC2CMP19IF	AD2CH19IF	ADC2CMP18IF	AD2CH18IF
0xB0	IFS8	31:24								
		23:16		IOM4IF	IOM3IF	IOM2IF	IOM1IF			
		15:8								
		7:0								
0xB4	IEC0	31:24						WDTIE	C4MONIE	C4WARMIE
		23:16	C4FAILIE	C3RDYIE	C3MONIE	C3WARMIE	C3FAILIE	C2RDYIE	C2MONIE	C2WARMIE
		15:8			C2FAILIE	C1WARMIE	C1FAILIE	CLKERRIE	CLKFAILIE	
		7:0	NVMCRCIE	NVMIE	NVMECCIE	PBERRIE		XRAMECCIE	CPUFPUIE	
0xB8	IEC1	31:24	SPI1RXIE							CCP4IE
		23:16	CCT4IE	CCP3IE	CCT3IE	CCP2IE	CCT2IE	CCP1IE	CCT1IE	T1IE
		15:8	PCG4IE	PCG3IE	PCG2IE	PCG1IE	PEVTFIE	PEVTEIE	PEVTDIE	PEVTCIE
		7:0	PEVTBIE	PEVTAIE	INT4IE	INT3IE	INT2IE	INT1IE	INT0IE	
0xBC	IEC2	31:24	U2EIE	U2TXIE	U2RXIE	U1EVTIE	U1EIE	U1TXIE	U1RXIE	
		23:16	I2C2TXIE	I2C2RXIE	I2C2IE	I2C2EIE	I2C1TXIE	I2C1RXIE	I2C1IE	I2C1EIE
		15:8		CMP3IE	CMP2IE	CMP1IE	DMA3IE	DMA2IE	DMA1IE	DMA0IE
		7:0	SPI3GIE	SPI3TXIE	SPI3RXIE	SPI2GIE	SPI2TXIE	SPI2RXIE	SPI1GIE	SPI1TXIE
0xC0	IEC3	31:24			QEI1IE					
		23:16				CNDIE	CNCIE	CNBIE	CNAIE	
		15:8		DMA5IE	DMA4IE	SENT2EIE	SENT2IE	SENT1EIE	SENT1IE	
		7:0				U3EVTIE	U3EIE	U3TXIE	U3RXIE	U2EVTIE
0xC4	IEC4	31:24	AD1CMP6IE	AD1CH6IE	AD1CMP5IE	AD1CH5IE	AD1CMP4IE	AD1CH4IE	AD1CMP3IE	AD1CH3IE
		23:16	AD1CMP2IE	AD1CH2IE	AD1CMP1IE	AD1CH1IE	AD1CMP0IE	AD1CH0IE		
		15:8					PTG3IE	PTG2IE	PTG1IE	PTG0IE
		7:0	PTGWDIE	PTGSTEPIE	ICDIE		CRCIE	BISS1EIE	BISS1TXIE	
0xC8	IEC5	31:24	AD2CMP1IE	AD2CH1IE	AD2CMP0IE	AD2CH0IE			AD1CMP19IE	AD1CH19IE
		23:16	AD1CMP18IE	AD1CH18IE	AD1CMP17IE	AD1CH17IE	AD1CMP16IE	AD1CH16IE	AD1CMP15IE	AD1CH15IE
		15:8	AD1CMP14IE	AD1CH14IE	AD1CMP13IE	AD1CH13IE	AD1CMP12IE	AD1CH12IE	AD1CMP11IE	AD1CH11IE
		7:0	AD1CMP10IE	AD1CH10IE	AD1CMP9IE	AD1CH9IE	AD1CMP8IE	AD1CH8IE	AD1CMP7IE	AD1CH7IE
0xCC	IEC6	31:24	AD2CMP17IE	AD2CH17IE	AD2CMP16IE	AD2CH16IE	AD2CMP15IE	AD2CH15IE	AD2CMP14IE	AD2CH14IE
		23:16	AD2CMP13IE	AD2CH13IE	AD2CMP12IE	AD2CH12IE	AD2CMP11IE	AD2CH11IE	AD2CMP10IE	AD2CH10IE
		15:8	AD2CMP9IE	AD2CH9IE	AD2CMP8IE	AD2CH8IE	AD2CMP7IE	AD2CH7IE	AD2CMP6IE	AD2CH6IE
		7:0	AD2CMP5IE	AD2CH5IE	AD2CMP4IE	AD2CH4IE	AD2CMP3IE	AD2CH3IE	AD2CMP2IE	AD2CH2IE
0xD0	IEC7	31:24								
		23:16								
		15:8	CLC4NIE	CLC4PIE	CLC3NIE	CLC3PIE	CLC2NIE	CLC2PIE	CLC1NIE	CLC1PIE
		7:0					AD2CMP19IE	AD2CH19IE	AD2CMP18IE	AD2CH18IE
0xD4	IEC8	31:24								
		23:16		IOMON4IE	IOMON3IE	IOMON2IE	IOMON1IE			
		15:8								
		7:0								
0xD8	IPC0	31:24		NVMCRCIP[2:0]				NVMIP[2:0]		
		23:16		NVMECCIP[2:0]				PBERRIP[2:0]		
		15:8		YRAMECCIP[2:0]				XRAMECCIP[2:0]		
		7:0		CPUFPUIP[2:0]						
0xDC	IPC1	31:24		C2FAILIP[2:0]				C1RDYIP[2:0]		
		23:16						C1MONIP[2:0]		
		15:8		C1FAILIP[2:0]				CLKERRIP[2:0]		
		7:0		CLKFAILIP[2:0]						

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0xE0	IPC2	31:24			C4FAILIP[2:0]				C3RDYIP[2:0]	
		23:16							CWARM3IP[2:0]	
		15:8			CLKFAIL3IP[2:0]				C2RDYIP[2:0]	
		7:0			C2MONIP[2:0]				C2WARMIP[2:0]	
0xE4	IPC3	31:24								
		23:16							WDTIP[2:0]	
		15:8							C4RDYIP[2:0]	
		7:0			C4MONIP[2:0]				C4WARMIP[2:0]	
0xE8	IPC4	31:24			PEVTBIP[2:0]				PEVTAIP[2:0]	
		23:16			INT4IP[2:0]				INT3IP[2:0]	
		15:8			INT2IP[2:0]				INT1IP[2:0]	
		7:0			INT0IP[2:0]					
0xEC	IPC5	31:24			PWM4IP[2:0]				PWM3IP[2:0]	
		23:16			PWM2IP[2:0]				PWM1IP[2:0]	
		15:8			PEVTFIP[2:0]				PEVTEIP[2:0]	
		7:0			PEVTDIP[2:0]				PEVTCIP[2:0]	
0xF0	IPC6	31:24			CCT4IP[2:0]				CCP3IP[2:0]	
		23:16			CCT3IP[2:0]					CCP2IP[2:0]
		15:8			CCT2IP[2:0]				CCP1IP[2:0]	
		7:0			CCT1IP[2:0]				T1IP[2:0]	
0xF4	IPC7	31:24			SPI1RXIP[2:0]					
		23:16								
		15:8								
		7:0							CCP4IP[2:0]	
0xF8	IPC8	31:24			SPI3GIP[2:0]				SPI3TXIP[2:0]	
		23:16			SPI3RXIP[2:0]				SPI2GIP[2:0]	
		15:8			SPI2TXIP[2:0]				SPI2RXIP[2:0]	
		7:0			SPI1GIP[2:0]				SPI1TXIP[2:0]	
0xFC	IPC9	31:24								
		23:16			CMP2IP[2:0]				CMP1IP[2:0]	
		15:8			DMA3IP[2:0]				DMA2IP[2:0]	
		7:0			DMA1IP[2:0]				DMA0IP[2:0]	
0x0100	IPC10	31:24			I2C2TXIP[2:0]				I2C2RXIP[2:0]	
		23:16							I2C2EIP[2:0]	
		15:8			I2C1TXIP[2:0]				I2C1RXIP[2:0]	
		7:0			I2C1IP[2:0]				I2C1EIP[2:0]	
0x0104	IPC11	31:24			U2EIP[2:0]				U2TXIP[2:0]	
		23:16			U2RXIP[2:0]				U1EVTIP[2:0]	
		15:8			U1EIP[2:0]				U1TXIP[2:0]	
		7:0			U1RXIP[2:0]					
0x0108	IPC12	31:24								
		23:16							U3EVTIP[2:0]	
		15:8			U3EIP[2:0]				U3TXIP[2:0]	
		7:0			U3RXIP[2:0]				U2EVTIP[2:0]	
0x010C	IPC13	31:24								
		23:16			DMA4IP[2:0]				SENT2EIP[2:0]	
		15:8			SENT2IP[2:0]				SENT1EIP[2:0]	
		7:0			SENT1IP[2:0]					
0x0110	IPC14	31:24								
		23:16							CNDIP[2:0]	
		15:8			CNCIP[2:0]				CNBIP[2:0]	
		7:0			CNAIP[2:0]					
0x0114	IPC15	31:24								
		23:16			QEI1IP[2:0]					
		15:8								
		7:0								
0x0118	IPC16	31:24			PTGWDIP[2:0]				PTGSTEPIP[2:0]	
		23:16							ICDIP[2:0]	
		15:8			CRCIP[2:0]				BISS1EIP[2:0]	
		7:0			BISS1TXIP[2:0]					

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x011C	IPC17	31:24								
		23:16								
		15:8		PTG3IP[2:0]				PTG2IP[2:0]		
		7:0		PTG1IP[2:0]				PTG0IP[2:0]		
0x0120	IPC18	31:24		AD1CMP2IP[2:0]				AD1CH2IP[2:0]		
		23:16		AD1CMP1IP[2:0]				AD1CH1IP[2:0]		
		15:8		AD1CMP0IP[2:0]				AD1CH0IP[2:0]		
		7:0								
0x0124	IPC19	31:24		AD1CMP6IP[2:0]				AD1CH6IP[2:0]		
		23:16		AD1CMP5IP[2:0]				AD1CH5IP[2:0]		
		15:8		AD1CMP4IP[2:0]				AD1CH4IP[2:0]		
		7:0		AD1CMP3IP[2:0]				AD1CH3IP[2:0]		
0x0128	IPC20	31:24		AD1CMP10IP[2:0]				AD1CH10IP[2:0]		
		23:16		AD1CMP9IP[2:0]				AD1CH9IP[2:0]		
		15:8		AD1CMP8IP[2:0]				AD1CH8IP[2:0]		
		7:0		AD1CMP7IP[2:0]				AD1CH7IP[2:0]		
0x012C	IPC21	31:24		AD1CMP14IP[2:0]				AD1CH14IP[2:0]		
		23:16		AD1CMP13IP[2:0]				AD1CH13IP[2:0]		
		15:8		AD1CMP12IP[2:0]				AD1CH12IP[2:0]		
		7:0		AD1CMP11IP[2:0]				AD1CH11IP[2:0]		
0x0130	IPC22	31:24		AD1CMP18IP[2:0]				AD1CH18IP[2:0]		
		23:16		AD1CMP17IP[2:0]				AD1CH17IP[2:0]		
		15:8		AD1CMP16IP[2:0]				AD1CH16IP[2:0]		
		7:0		AD1CMP15IP[2:0]				AD1CH15IP[2:0]		
0x0134	IPC23	31:24		AD2CMP1IP[2:0]				AD2CH1IP[2:0]		
		23:16		AD2CMP0IP[2:0]				AD2CH0IP[2:0]		
		15:8								
		7:0		AD1CMP19IP[2:0]				AD1CH19IP[2:0]		
0x0138	IPC24	31:24		AD2CMP5IP[2:0]				AD2CH5IP[2:0]		
		23:16		AD2CMP4IP[2:0]				AD2CH4IP[2:0]		
		15:8						AD2CH3IP[2:0]		
		7:0		AD2CMP2IP[2:0]				AD2CH2IP[2:0]		
0x013C	IPC25	31:24		AD2CMP9IP[2:0]				AD2CH9IP[2:0]		
		23:16		AD2CMP8IP[2:0]				AD2CH8IP[2:0]		
		15:8		AD2CMP7IP[2:0]				AD2CH7IP[2:0]		
		7:0		AD2CMP6IP[2:0]				AD2CH6IP[2:0]		
0x0140	IPC26	31:24		AD2CMP13IP[2:0]				AD2CH13IP[2:0]		
		23:16		AD2CMP12IP[2:0]				AD2CH12IP[2:0]		
		15:8		AD2CMP11IP[2:0]				AD2CH11IP[2:0]		
		7:0		AD2CMP10IP[2:0]				AD2CH10IP[2:0]		
0x0144	IPC27	31:24		AD2CMP17IP[2:0]				AD2CH17IP[2:0]		
		23:16		AD2CMP16IP[2:0]				AD2CH16IP[2:0]		
		15:8		AD2CMP15IP[2:0]				AD2CH15IP[2:0]		
		7:0		AD2CMP14IP[2:0]				AD2CH14IP[2:0]		
0x0148	IPC28	31:24								
		23:16								
		15:8		AD2CMP19IP[2:0]				AD2CH19IP[2:0]		
		7:0		AD2CMP18IP[2:0]				AD2CH18IP[2:0]		
0x014C	IPC29	31:24		CLC4NIP[2:0]				CLC4PIP[2:0]		
		23:16		CLC3NIP[2:0]				CLC3PIP[2:0]		
		15:8		CLC2NIP[2:0]				CLC2PIP[2:0]		
		7:0		CLC1NIP[2:0]				CLC1PIP[2:0]		
0x0150 ... 0x015F	Reserved									
0x0160	IPC34	31:24						IOM4IP[2:0]		
		23:16		IOM3IP[2:0]				IOM2IP[2:0]		
		15:8		IOM1IP[2:0]						
		7:0								

10.4.1 Interrupt Control Register 1

Name: INTCON1
Offset: 0x70

Note:
1. The user is responsible for clearing this bit by writing a zero to it.

Bit	31	30	29	28	27	26	25	24
	NSTDIS							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	GIE							
Access	R/W							
Reset	1							

Bit	7	6	5	4	3	2	1	0
				STKERR	ADDRERR	BADOPERR		
Access				R/W	R/W	R/W		
Reset				0	0	0		

Bit 31 – NSTDIS Interrupt Nesting Disable bit

Value	Description
1	Interrupt nesting is disabled
0	Interrupt nesting is enabled

Bit 15 – GIE Global Interrupt Enable bit

Value	Description
1	Interrupts are enabled (assuming associated IE bits are enabled)
0	Interrupts are disabled (traps are still enabled)

Bit 4 – STKERR Stack Error Trap status bit(1)

Value	Description
1	Stack Error Trap has occurred
0	Stack Error Trap has not occurred

Bit 3 – ADDRERR Address Error Trap status bit(1)

Value	Description
1	Address Error Trap has occurred
0	Address Error Trap has not occurred

Bit 2 – BADOPERR Illegal Op-code Error Trap status bit(1)

Value	Description
1	Illegal Opcode Error Trap has occurred
0	Illegal Opcode Error Trap has not occurred

10.4.2 Interrupt Control Register 2

Name: INTCON2
Offset: 0x74

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit 4 – INT4EP External Interrupt 4 Edge Detect Polarity Select bit

Value	Description
1	Interrupt on negative edge
0	Interrupt on positive edge

Bit 3 – INT3EP External Interrupt 3 Edge Detect Polarity Select bit

Value	Description
1	Interrupt on negative edge
0	Interrupt on positive edge

Bit 2 – INT2EP External Interrupt 2 Edge Detect Polarity Select bit

Value	Description
1	Interrupt on negative edge
0	Interrupt on positive edge

Bit 1 – INT1EP External Interrupt 1 Edge Detect Polarity Select bit

Value	Description
1	Interrupt on negative edge
0	Interrupt on positive edge

Bit 0 – INT0EP External Interrupt 0 Edge Detect Polarity Select bit

Value	Description
1	Interrupt on negative edge
0	Interrupt on positive edge

10.4.3 Interrupt Control Register 3

Name: INTCON3
Offset: 0x78

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					CPUBET	DMABET	YRAMBET	XRAMBET
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – CPUBET CPU Instruction Bus Error Trap Status bit

Value	Description
1	CPU Instruction-bus error has occurred
0	CPU Instruction-bus error has not occurred

Bit 2 – DMABET DMA Bus Error Trap Status bit

Value	Description
1	DMA bus error has occurred
0	DMA bus error has not occurred

Bit 1 – YRAMBET CPU Y Data Bus Error Trap Status bit

Value	Description
1	CPU Y Data bus error has occurred
0	CPU Y Data bus error has not occurred

Bit 0 – XRAMBET CPU X Data Bus Error Trap Status bit

Value	Description
1	CPU X Data bus error has occurred
0	CPU X Data bus error has not occurred

10.4.4 Interrupt Control Register 4

Name: INTCON4
Offset: 0x7C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			OVATE	OVBTE	COVTE			
Access			R/W	R/W	R/W			
Reset			0	0	0			

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			OVAERR	OVBERR	COVAERR	COVBERR	SFTACERR	DIV0ERR
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 21 – OVATE Accumulator A Overflow Trap Enable bit

Value	Description
1	Enable Accumulator A overflow trap (OVAERR)
0	Trap is disabled

Bit 20 – OVBTE Accumulator B Overflow Trap Enable bit

Value	Description
1	Enable Accumulator B overflow trap (OVBERR)
0	Trap is disabled

Bit 19 – COVTE Catastrophic Overflow A/B Trap Enable bit

Value	Description
1	Enable trap on Catastrophic overflow of Accumulator A/B (COVAERR/COVBERR)
0	Trap is disabled

Bit 5 – OVAERR Accumulator A Overflow Trap Flag bit

Value	Description
1	Trap was caused by overflow of Accumulator A
0	Trap was not caused by overflow of Accumulator A

Bit 4 – OVBERR Accumulator B Overflow Trap Flag bit

Value	Description
1	Trap was caused by overflow of Accumulator B
0	Trap was not caused by overflow of Accumulator B

Bit 3 – COVAERR Accumulator A Catastrophic Overflow Trap Flag bit

Value	Description
1	Trap was caused by catastrophic overflow of Accumulator A
0	Trap was not caused by catastrophic overflow of Accumulator A

Bit 2 – COVBERR Accumulator B Catastrophic Overflow Trap Flag bit

Value	Description
1	Trap was caused by catastrophic overflow of Accumulator B
0	Trap was not caused by catastrophic overflow of Accumulator B

Bit 1 – SFTACERR Shift Accumulator Error Status bit

Value	Description
1	Math error trap was caused by an invalid accumulator shift
0	Math error trap was not caused by an invalid accumulator shift

Bit 0 – DIV0ERR Arithmetic Divide-By-Zero Error Status bit

Value	Description
1	Math error trap occurred due to arithmetic divide by zero
0	Math error trap due to divide-by-zero has not occurred

10.4.5 Interrupt Control Register 5

Name: INTCON5
Offset: 0x80

Bit	31	30	29	28	27	26	25	24
	SOFT							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					YPWBDED	XPWBDED	WDTE	DMTE
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 31 – SOFT Software Generated Soft Trap Status bit

Value	Description
1	Raise software generated Soft Trap
0	Soft trap has not occurred

Bit 3 – YPWBDED YRAM PWB DED Trap Status bit

Value	Description
1	YRAM PWB DED error event has occurred
0	YRAM PWB DED error event has not occurred

Bit 2 – XPWBDED XRAM PWB DED Trap Status bit

Value	Description
1	XRAM PWB DED error event has occurred
0	XRAM PWB DED error event has not occurred

Bit 1 – WDTE WDT Run Mode Event Status bit

Value	Description
1	WDT event has occurred
0	WDT event has not occurred

Bit 0 – DMTE DMT Event Status bit

Value	Description
1	DMT event has occurred
0	DMT event has not occurred

10.4.6 Interrupt Control and Status Register

Name: INTTREG
Offset: 0x84

Notes:

1. During debug mode, the ICD can force the interrupts to be disabled. This allows the user code to progress with single stepping even when there are interrupts pending.
2. The bit-fields in the INTTREG register correspond to the value of vector number, interrupt level and IRQ request flag, when an interrupt is presented to the CPU.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	IRQCPU		VHOLD					
Access	R		R/W					
Reset	0		0					

Bit	15	14	13	12	11	10	9	8
			ILR[3:0]					VECNUM[8]
Access			R	R	R	R		R
Reset			0	0	0	0		0

Bit	7	6	5	4	3	2	1	0
	VECNUM[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 23 – IRQCPU Interrupt Request from Interrupt Controller to CPU bit(1,2)

Bit 21 – VHOLD Vector Number Capture Enable bit

Value	Description
1	VECNUM[8:0] bits read current value of vector number encoding tree (i.e., highest priority pending interrupt)
0	Vector number latched into VECNUM[8:0] at Interrupt Acknowledge and retained until next IACK

Bits 13:10 – ILR[3:0] CPU Interrupt Priority Level bits(2)

Bits 8:0 – VECNUM[8:0] Vector Number of Pending Interrupt bits(2)
Vector number of pending interrupt or last acknowledged interrupt. VECNUM = IRQ + 9.

10.4.7 Interrupt Vector Base Address Register

Name: IVTBASE
Offset: 0x88

Note:

1. IVTBASE[5:0] is hard coded to 1'b000000.

Bit	31	30	29	28	27	26	25	24
	IVTBASE[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	IVTBASE[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	IVTBASE[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	IVTBASE[7:6]							
Access	R/W	R/W						
Reset	0	0						

Bits 31:24 – IVTBASE[31:24]

Bits 23:16 – IVTBASE[23:16] Interrupt

Bits 15:8 – IVTBASE[15:8]

Bits 7:6 – IVTBASE[7:6] Interrupt Vector Table Base Address bits(1)

10.4.8 Interrupt Vector Collapse Register

Name: IVTCREG
Offset: 0x8C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								IVTC
Access								R/W
Reset								0

Bit 0 – IVTC Interrupt Vector Table Collapse bit

Value	Description
1	Enable Interrupt Vector table collapse
0	Interrupt Vector table collapse disabled

10.4.9 Interrupt Request Flags Register 0

Name: IFS0
Offset: 0x90

Bit	31	30	29	28	27	26	25	24
				WDTIF		C4RDYIF	C4MONIF	C4WARMIF
Access				RW		RW	RW	RW
Reset				0		0	0	0

Bit	23	22	21	20	19	18	17	16
	C4FAILIF	C3RDYIF	C3MONIF	C3WARMIF	C3FAILIF	C2RDYIF	C2MONIF	C2WARMIF
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	C2FAILIF	C1RDYIF	C1MONIF	C1WARMIF	C1FAIL1IF	CLKERRIF	CLKFAILIF	
Access	RW	RW	RW	RW	RW	RW	RW	
Reset	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0
	NVMCRCIF	NVMIF	NVMECCIF	PBERRIF	YRAMECCIF	XRAMECCIF	CPUFPUIF	
Access	RW	RW	RW	RW	RW	RW	RW	
Reset	0	0	0	0	0	0	0	

Bit 28 – WDTIF Sleep/Idle Mode WDT Interrupt Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 26 – C4RDYIF Clock 4 Ready Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 25 – C4MONIF Clock 4 Monitor Error Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 24 – C4WARMIF Clock 4 Warming Complete Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 23 – C4FAILIF Clock 4 Failure Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 22 – C3RDYIF Clock 3 Ready Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 21 – C3MONIF Clock 3 Monitor Error Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 20 – C3WARMIF Clock 3 Warming Complete Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 19 – C3FAILIF Clock 3 Failure Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 18 – C2RDYIF Clock 2 Ready Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 17 – C2MONIF Clock 2 Monitor Error Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 16 – C2WARMIF Clock 2 Warming Complete Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 15 – C2FAILIF Clock 2 Failure Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 14 – C1RDYIF Clock 1 Ready Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 13 – C1MONIF Clock 1 Monitor Error Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 12 – C1WARMIF Clock 1 Warming Complete Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 11 – C1FAIL1IF Clock 1 Failure Interrupt Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 10 – CLKERRIF Combined Clock Error Interrupt Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 9 – CLKFAILIF Combined Clock Failure Interrupt Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 7 – NVMCRCIF NVM CRC Operation Complete Interrupt Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 6 – NVMIF NVM Program or Erase Complete Interrupt Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 5 – NVMECCIF NVM ECC Single Error Interrupt Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 4 – PBERRIF PBU Parity Error Interrupt Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 3 – YRAMECCIF Y-RAM ECC Single Error Interrupt Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 2 – XRAMECCIF X-RAM ECC Single Error Interrupt Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 1 – CPUFPUIF FPU Interrupt Status Flag bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

10.4.10 Interrupt Request Flags Register 1

Name: IFS1
Offset: 0x94

Bit	31	30	29	28	27	26	25	24
	SPI1RXIF							CCP4IF
Access	R							R/W
Reset	0							0

Bit	23	22	21	20	19	18	17	16
	CCT4IF	CCP3IF	CCT3IF	CCP2IF	CCT2IF	CCP1IF	CCT1IF	T1IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PCG4IF	PCG3IF	PCG2IF	PCG1IF	PEVTFIF	PEVTEIF	PEVTDIF	PEVTCIF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PEVTBIF	PEVTAIF	INT4IF	INT3IF	INT2IF	INT1IF	INT0IF	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	

Bit 31 – SPI1RXIF SPI1 Receive Interrupt Flag

Value	Description
1	Interrupt request has occurred
0	Interrupt request has not occurred

Bit 24 – CCP4IF Input Capture/Output Compare 4 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 23 – CCT4IF Capture/Compare/Timer 4 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 22 – CCP3IF Input Capture/Output Compare 3 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 21 – CCT3IF Capture/Compare/Timer 3 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 20 – CCP2IF Input Capture/Output Compare 2 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 19 – CCT2IF Capture/Compare/Timer 2 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 18 – CCP1IF Input Capture/Output Compare 1 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 17 – CCT1IF Capture/Compare/Timer 1 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 16 – T1IF Timer1 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 15 – PCG4IF PWM1 Generator 4 Interrupt Status bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 14 – PCG3IF PWM1 Generator 3 Interrupt Status bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 13 – PCG2IF PWM1 Generator 2 Interrupt Status bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 12 – PCG1IF PWM1 Generator 1 Interrupt Status bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 11 – PEVTFIF PWM Event F Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 10 – PEVTEIF PWM Event E Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 9 – PEVTDIF PWM Event D Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 8 – PEVTCIF PWM Event C Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 7 – PEVTBIF PWM Event B Interrupt bit

Value	Description
1	Interrupt request has occurred
0	Interrupt request has not occurred

Bit 6 – PEVTAIF PWM Event A Interrupt bit

Value	Description
1	Interrupt request has occurred
0	Interrupt request has not occurred

Bit 5 – INT4IF External Interrupt 4 bit

Value	Description
1	Interrupt request has occurred
0	Interrupt request has not occurred

Bit 4 – INT3IF External Interrupt 3 bit

Value	Description
1	Interrupt request has occurred
0	Interrupt request has not occurred

Bit 3 – INT2IF External Interrupt 2 bit

Value	Description
1	Interrupt request has occurred
0	Interrupt request has not occurred

Bit 2 – INT1IF External Interrupt 1 bit

Value	Description
1	Interrupt request has occurred
0	Interrupt request has not occurred

Bit 1 – INT0IF External Interrupt 0 bit

Value	Description
1	Interrupt request has occurred
0	Interrupt request has not occurred

10.4.11 Interrupt Request Flags Register 2

Name: IFS2
Offset: 0x98

Bit	31	30	29	28	27	26	25	24
	U2EIF	U2TXIF	U2RXIF	U1EVTIF	U1EIF	U1TXIF	U1RXIF	
Access	R	R	R	R/W	R	R	R	
Reset	0	0	0	0	0	0	0	

Bit	23	22	21	20	19	18	17	16
	I2C2TXIF	I2C2RXIF	I2C2IF	I2C2EIF	I2C1TXIF	I2C1RXIF	I2C1IF	I2C1EIF
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		CMP3IF	CMP2IF	CMP1IF	DMA3IF	DMA2IF	DMA1IF	DMA0IF
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SPI3GIF	SPI3TXIF	SPI3RXIF	SPI2GIF	SPI2TXIF	SPI2RXIF	SPI1GIF	SPI1TXIF
Access	R/W	R	R	R/W	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – U2EIF UART2 Framing Error Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 30 – U2TXIF UART2 Transmit Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 29 – U2RXIF UART2 Receive Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 28 – U1EVTIF UART1 Event Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 27 – U1EIF UART1 Framing Error Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 26 – U1TXIF UART1 Transmit Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 25 – U1RXIF UART1 Receive Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 23 – I2C2TXIF I2C2 Transmit Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 22 – I2C2RXIF I2C2 Receive Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 21 – I2C2IF I2C2 Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 20 – I2C2EIF I2C2 Error Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 19 – I2C1TXIF I2C1 Transmit Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 18 – I2C1RXIF I2C1 Receive Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 17 – I2C1IF I2C1 Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 16 – I2C1EIF I2C1 Error Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 14 – CMP3IF Comparator 3 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 13 – CMP2IF Comparator 2 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 12 – CMP1IF Comparator 1 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 11 – DMA3IF Direct Memory Access 3 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 10 – DMA2IF Direct Memory Access 2 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 9 – DMA1IF Direct Memory Access 1 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 8 – DMA0IF Direct Memory Access 0 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 7 – SPI3GIF SPI3 General Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 6 – SPI3TXIF SPI3 Transmit Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 5 – SPI3RXIF SPI3 Receive Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 4 – SPI2GIF SPI2 General Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 3 – SPI2TXIF SPI2 Transmit Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 2 – SPI2RXIF SPI2 Receive Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 1 – SPI1GIF SPI1 General Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 0 – SPI1TXIF SPI1 Transmit Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

10.4.12 Interrupt Request Flags Register 3

Name: IFS3
Offset: 0x9C

Bit	31	30	29	28	27	26	25	24
			QEI1IF					
Access			R/W					
Reset			0					

Bit	23	22	21	20	19	18	17	16
				CNDIF	CNCIF	CNBIF	CNAIF	
Access				R/W	R/W	R/W	R/W	
Reset				0	0	0	0	

Bit	15	14	13	12	11	10	9	8
		DMA5IF	DMA4IF	SENT2EIF	SENT2IF	SENT1EIF	SENT1IF	PWM6IF
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	U4EVTIF			U3EVTIF	U3EIF	U3TXIF	U3RXIF	U2EVTIF
Access	R/W			R/W	R	R	R	R/W
Reset	0			0	0	0	0	0

Bit 29 – QEI1IF QEI1 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 20 – CNDIF Change Notice D Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 19 – CNCIF Change Notice C Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 18 – CNBIF Change Notice B Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 17 – CNAIF Change Notice A Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 14 – DMA5IF Direct Memory Access 5 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 13 – DMA4IF Direct Memory Access 4 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 12 – SENT2EIF SENT2 Error Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 11 – SENT2IF SENT2 TX/RX Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 10 – SENT1EIF SENT1 Error Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 10 – PWM8IF PWM Generator 8 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 9 – SENT1IF SENT1 TX/RX Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 9 – PWM7IF PWM Generator 7 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 8 – PWM6IF PWM Generator 6 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 7 – U4EVTIF UART4 Event Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 4 – U3EVTIF UART3 Event Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 3 – U3EIF UART3 Framing Error Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 2 – U3TXIF UART3 Transmit Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 1 – U3RXIF UART3 Receive Interrupt Flag

Value	Description
1	Interrupt has occurred
0	Interrupt event has not occurred

Bit 0 – U2EVTIF UART2 Event Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

10.4.13 Interrupt Request Flags Register 4

Name: IFS4
Offset: 0xA0

Bit	31	30	29	28	27	26	25	24
	AD1CMP6IF	AD1CH6IF	AD1CMP5IF	AD1CH5IF	AD1CMP4IF	AD1CH4IF	AD1CMP3IF	AD1CH3IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	AD1CMP2IF	AD1CH2IF	AD1CMP1IF	AD1CH1IF	AD1CMP0IF	AD1CH0IF		
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bit	15	14	13	12	11	10	9	8
					PTG3IF	PTG2IF	PTG1IF	PTG0IF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGWDTIF	PTGSTEPIF		ICDIF	CRCIF	BISS1EIF		
Access	R/W	R/W		R/W	R/W/HS	R/W/HS		
Reset	0	0		0	0	0		

Bit 31 – AD1CMP6IF ADC 1 Digital Comparator 6 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 30 – AD1CH6IF ADC 1 Channel 6 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 29 – AD1CMP5IF ADC 1 Digital Comparator 5 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 28 – AD1CH5IF ADC 1 Channel 5 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 27 – AD1CMP4IF ADC 1 Digital Comparator 4 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 26 – AD1CH4IF ADC 1 Channel 4 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 25 – AD1CMP3IF ADC 1 Digital Comparator 3 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 24 – AD1CH3IF ADC 1 Channel 3 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 23 – AD1CMP2IF ADC 1 Digital Comparator 2 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 22 – AD1CH2IF ADC 1 Channel 2 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 21 – AD1CMP1IF ADC 1 Digital Comparator 1 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 20 – AD1CH1IF ADC 1 Channel 1 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 19 – AD1CMP0IF ADC 1 Digital Comparator 0 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 18 – AD1CH0IF ADC 1 Channel 0 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 11 – PTG3IF PTG3 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 10 – PTG2IF PTG2 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 9 – PTG1IF PTG1 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 8 – PTG0IF PTG0 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 7 – PTGWDTIF PTG Watchdog Timer Time-out Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 6 – PTGSTEPIF PTG Step Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 4 – ICDIF In-Circuit Debugger Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 3 – CRCIF CRC Interrupt Flag

Value	Description
1	Interrupt has occurred (must be cleared by software)
0	Interrupt event has not occurred

Bit 2 – BISS1EIF BiSS 1 Error Interrupt bit

Value	Description
1	Interrupt has occurred (must be cleared by software)
0	Interrupt event has not occurred

Bit 2 – BISS1TXIF BiSS 1 Transmit Interrupt bit

Value	Description
1	Interrupt has occurred (must be cleared by software)
0	Interrupt event has not occurred

10.4.14 Interrupt Request Flags Register 5

Name: IFS5
Offset: 0xA4

Bit	31	30	29	28	27	26	25	24
	AD2CMP1IF	AD2CH1IF	AD2CMP0IF	AD2CH0IF			AD1CMP19IF	AD1CH19IF
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	23	22	21	20	19	18	17	16
	AD1CMP18IF	AD1CH18IF	AD1CMP17IF	AD1CH17IF	AD1CMP16IF	AD1CH16IF	AD1CMP15IF	AD1CH15IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	AD1CMP14IF	AD1CH14IF	AD1CMP13IF	AD1CH13IF	AD1CMP12IF	AD1CH12IF	AD1CMP11IF	AD1CH11IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	AD1CMP10IF	AD1CH10IF	AD1CMP9IF	AD1CH9IF	AD1CMP8IF	AD1CH8IF	AD1CMP7IF	AD1CH7IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – AD2CMP1IF ADC 2 Digital Comparator 1 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 30 – AD2CH1IF ADC 2 Channel 1 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 29 – AD2CMP0IF ADC 2 Digital Comparator 0 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 28 – AD2CH0IF ADC 2 Channel 0 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 25 – AD1CMP19IF ADC 1 Digital Comparator 19 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 24 – AD1CH19IF ADC 1 Channel 19 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 23 – AD1CMP18IF ADC 1 Digital Comparator 18 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 22 – AD1CH18IF ADC 1 Channel 18 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 21 – AD1CMP17IF ADC 1 Digital Comparator 17 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 20 – AD1CH17IF ADC 1 Channel 17 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 19 – AD1CMP16IF ADC 1 Digital Comparator 16 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 18 – AD1CH16IF ADC 1 Channel 16 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 17 – AD1CMP15IF ADC 1 Digital Comparator 15 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 16 – AD1CH15IF ADC 1 Channel 15 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 15 – AD1CMP14IF ADC 1 Digital Comparator 14 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 14 – AD1CH14IF ADC 1 Channel 14 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 13 – AD1CMP13IF ADC 1 Digital Comparator 13 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 12 – AD1CH13IF ADC 1 Channel 13 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 11 – AD1CMP12IF ADC 1 Digital Comparator 12 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 10 – AD1CH12IF ADC 1 Channel 12 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 9 – AD1CMP11IF ADC 1 Digital Comparator 11 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 8 – AD1CH11IF ADC 1 Channel 11 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 7 – AD1CMP10IF ADC 1 Digital Comparator 10 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 6 – AD1CH10IF ADC 1 Channel 10 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 5 – AD1CMP9IF ADC 1 Digital Comparator 9 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 4 – AD1CH9IF ADC 1 Channel 9 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 3 – AD1CMP8IF ADC 1 Digital Comparator 8 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 2 – AD1CH8IF ADC 1 Channel 8 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 1 – AD1CMP7IF ADC 1 Digital Comparator 7 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 0 – AD1CH7IF ADC 1 Channel 7 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

10.4.15 Interrupt Request Flags Register 6

Name: IFS6
Offset: 0xA8

Bit	31	30	29	28	27	26	25	24
	AD2CMP17IF	AD2CH17IF	AD2CMP16IF	AD2CH16IF	AD2CMP15IF	AD2CH15IF	AD2CMP14IF	AD2CH14IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	AD2CMP13IF	AD2CH13IF	AD2CMP12IF	AD2CH12IF	AD2CMP11IF	AD2CH11IF	AD2CMP10IF	AD2CH10IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	AD2CMP9IF	AD2CH9IF	AD2CMP8IF	AD2CH8IF	AD2CMP7IF	AD2CH11IF	AD2CMP10IF	AD2CH10IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	AD2CMP5IF	AD2CH5IF	AD2CMP4IF	AD2CH4IF	AD2CMP4IF	AD2CH3IF	AD2CMP2IF	AD2CH2IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – AD2CMP17IF ADC 2 Digital Comparator 17 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 30 – AD2CH17IF ADC 2 Channel 17 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 29 – AD2CMP16IF ADC 2 Digital Comparator 16 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 28 – AD2CH16IF ADC 2 Channel 16 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 27 – AD2CMP15IF ADC 2 Digital Comparator 15 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 26 – AD2CH15IF ADC 2 Channel 15 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 25 – AD2CMP14IF ADC 2 Digital Comparator 14 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 24 – AD2CH14IF ADC 2 Channel 14 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 23 – AD2CMP13IF ADC 2 Digital Comparator 13 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 22 – AD2CH13IF ADC 2 Channel 13 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 21 – AD2CMP12IF ADC 2 Digital Comparator 12 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 20 – AD2CH12IF ADC 2 Channel 12 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 19 – AD2CMP11IF ADC 2 Digital Comparator 11 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 18 – AD2CH11IF ADC 2 Channel 11 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 17 – AD2CMP10IF ADC 2 Digital Comparator 10 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 16 – AD2CH10IF ADC 2 Channel 10 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 15 – AD2CMP9IF ADC 2 Digital Comparator 9 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 14 – AD2CH9IF ADC 2 Channel 9 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 13 – AD2CMP8IF ADC 2 Digital Comparator 8 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 12 – AD2CH8IF ADC 2 Channel 8 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 11 – AD2CMP7IF ADC 2 Digital Comparator 7 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 10 – AD2CH11IF ADC 2 Channel 11 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 9 – AD2CMP10IF ADC 2 Digital Comparator 10 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 8 – AD2CH10IF ADC 2 Channel 10 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 7 – AD2CMP5IF ADC 2 Digital Comparator 5 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 6 – AD2CH5IF ADC 2 Channel 5 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 5 – AD2CMP4IF ADC 2 Digital Comparator 4 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 4 – AD2CH4IF ADC 2 Channel 4 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 3 – AD2CMP4IF ADC 2 Digital Comparator 5 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 2 – AD2CH3IF ADC 2 Channel 3 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 1 – AD2CMP2IF ADC 2 Digital Comparator 2 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 0 – AD2CH2IF ADC 2 Channel 2 Conversion Ready bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

10.4.16 Interrupt Request Flags Register 7

Name: IFS7
Offset: 0xAC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CLC4NIF	CLC4PIF	CLC3NIF	CLC3PIF	CLC2NIF	CLC2PIF	CLC1NIF	CLC1PIF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
					ADC2CMP19IF	AD2CH19IF	ADC2CMP18IF	AD2CH18IF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 15 – CLC4NIF CLC4 Negative Edge Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 14 – CLC4PIF CLC4 Positive Edge Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 13 – CLC3NIF CLC3 Negative Edge Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 12 – CLC3PIF CLC4 Positive Edge Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 11 – CLC2NIF CLC2 Negative Edge Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 10 – CLC2PIF CLC2 Positive Edge Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 9 – CLC1NIF CLC1 Negative Edge Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 8 – CLC1PIF CLC1 Positive Edge Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 3 – ADC2CMP19IF ADC 2 Comparator 19 Interrupt bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 2 – AD2CH19IF ADC 2 Channel 19 Conversion Ready bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 1 – ADC2CMP18IF ADC 2 Comparator 18 Interrupt bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 0 – AD2CH18IF ADC 2 Channel 18 Conversion Ready bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

10.4.17 Interrupt Request Flags Register 8

Name: IFS8
Offset: 0xB0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		IOM4IF	IOM3IF	IOM2IF	IOM1IF			
Access		R/W	R/W	R/W	R/W			
Reset		0	0	0	0			

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 22 – IOM4IF IOM 4 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 21 – IOM3IF IOM 3 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 20 – IOM2IF IOM 2 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 19 – IOM1IF IOM 1 Interrupt bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

10.4.18 Interrupt Enable Register 0

Name: IEC0
Offset: 0xB4

Bit	31	30	29	28	27	26	25	24
						WDTIE	C4MONIE	C4WARMIE
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	23	22	21	20	19	18	17	16
	C4FAILIE	C3RDYIE	C3MONIE	C3WARMIE	C3FAILIE	C2RDYIE	C2MONIE	C2WARMIE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
			C2FAILIE	C1WARMIE	C1FAILIE	CLKERRIE	CLKFAILIE	
Access			R/W	R/W	R/W	R/W	R/W	
Reset			0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0
	NVMCRCIE	NVMIE	NVMECCIE	PBERRIE		XRAMECCIE	CPUFPUIE	
Access	R/W	R/W	R/W	R/W		R/W	R/W	
Reset	0	0	0	0		0	0	

Bit 26 – WDTIE Watchdog Timer Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 26 – C4RDYIE Clock 4 Ready Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 25 – C4MONIE Clock 4 Monitor Error Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 24 – C4WARMIE Clock 4 Warming Complete Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 23 – C4FAILIE Clock 4 Failure Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 22 – C3RDYIE Clock 3 Ready Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 21 – C3MONIE Clock3 Monitor Error Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 20 – C3WARMIE Clock 3 Warming Complete Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 19 – C3FAILIE Clock 3 Failure Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 18 – C2RDYIE Clock 2 Ready Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 17 – C2MONIE Clock 2 Monitor Error Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 16 – C2WARMIE Clock 2 Warming Complete Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 13 – C2FAILIE Clock 2 Failure Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 13 – C1RDYIE Clock 1 Ready Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 13 – C1MONIE Clock 1 Monitor Error Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 12 – C1WARMIE Clock 1 Warming Complete Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 11 – C1FAILIE Clock 1 Failure Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 10 – CLKERRIE Clock Error (Combined) Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 9 – CLKFAILIE Clock Failure (Combined) Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 7 – NVMCRCIE NVM CRC Operation Complete Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 6 – NVMIE NVM Erase/Program Operation Complete Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 5 – NVMECCIE NVM ECC Single Error Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 4 – PBERRIE PBU Parity Error Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 2 – XRAMECCIE X RAM ECC Single Error Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 2 – YRAMECCIE Y RAM ECC Single Error Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 1 – CPUFPUIE CPU Floating Point Unit Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

10.4.19 Interrupt Enable Register 1

Name: IEC1
Offset: 0xB8

Bit	31	30	29	28	27	26	25	24
	SPI1RXIE							CCP4IE
Access	R/W							R/W
Reset	0							0

Bit	23	22	21	20	19	18	17	16
	CCT4IE	CCP3IE	CCT3IE	CCP2IE	CCT2IE	CCP1IE	CCT1IE	T1IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PCG4IE	PCG3IE	PCG2IE	PCG1IE	PEVTFIE	PEVTEIE	PEVTDIE	PEVTCIE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PEVTBIE	PEVTAIE	INT4IE	INT3IE	INT2IE	INT1IE	INT0IE	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	

Bit 31 – SPI1RXIE SPI1 Receive Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is disabled

Bit 24 – CCP4IE Input Capture/Output Compare 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 23 – CCT4IE Capture/Compare/Timer 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 22 – CCP3IE Input Capture/Output Compare 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 21 – CCT3IE Capture/Compare/Timer 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 20 – CCP2IE Input Capture/Output Compare 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 19 – CCT2IE Capture/Compare/Timer 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 18 – CCP1IE Input Capture/Output Compare 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 17 – CCT1IE Capture/Compare/Timer 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 16 – T1IE Timer 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 15 – PCG4IE PWM 1 Generator 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 14 – PCG3IE PWM 1 Generator 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 13 – PCG2IE PWM 1 Generator 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 12 – PCG1IE PWM 1 Generator 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 11 – PEVTFIE PWM Event F Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 10 – PEVTEIE PWM Event E Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 9 – PEVTDIE PWM Event D Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 8 – PEVTCIE PWM Event C Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 7 – PEVTBIE PWM Event B Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 6 – PEVTAIE PWM Event A Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 5 – INT4IE External Interrupt 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 4 – INT3IE External Interrupt 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 3 – INT2IE External Interrupt 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 2 – INT1IE External Interrupt 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 1 – INT0IE External Interrupt 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

10.4.20 Interrupt Enable Register 2

Name: IEC2
Offset: 0xBC

Bit	31	30	29	28	27	26	25	24
	U2EIE	U2TXIE	U2RXIE	U1EVTIE	U1EIE	U1TXIE	U1RXIE	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	

Bit	23	22	21	20	19	18	17	16
	I2C2TXIE	I2C2RXIE	I2C2IE	I2C2EIE	I2C1TXIE	I2C1RXIE	I2C1IE	I2C1EIE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		CMP3IE	CMP2IE	CMP1IE	DMA3IE	DMA2IE	DMA1IE	DMA0IE
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SPI3GIE	SPI3TXIE	SPI3RXIE	SPI2GIE	SPI2TXIE	SPI2RXIE	SPI1GIE	SPI1TXIE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – U2EIE UART2 Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 30 – U2TXIE UART2 Transmit Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 29 – U2RXIE UART2 Receive Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 28 – U1EVTIE UART1 Event Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 27 – U1EIE UART1 Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 26 – U1TXIE UART1 Transmit Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 25 – U1RXIE UART1 Receive Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 23 – I2C2TXIE I2C2 Transmit Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 22 – I2C2RXIE I2C2 Receive Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 21 – I2C2IE I2C2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 20 – I2C2EIE I2C2 Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 19 – I2C1TXIE I2C2 Transmit Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 18 – I2C1RXIE I2C2 Receive Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 17 – I2C1IE I2C1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 16 – I2C1EIE I2C1 Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 14 – CMP3IE Comparator 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 13 – CMP2IE Comparator 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 12 – CMP1IE Comparator 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 11 – DMA3IE Direct Memory Access 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 10 – DMA2IE Direct Memory Access 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 9 – DMA1IE Direct Memory Access 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 8 – DMA0IE Direct Memory Access 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 7 – SPI3GIE SPI3 General Interrupt Enable bit

Value	Description
1	Interrupt request enabled
0	Interrupt request not enabled

Bit 6 – SPI3TXIE SPI3 Transmitter Interrupt Enable bit

Value	Description
1	Interrupt request enabled
0	Interrupt request not enabled

Bit 5 – SPI3RXIE SPI3 Receiver Interrupt Enable bit

Value	Description
1	Interrupt request enabled
0	Interrupt request not enabled

Bit 4 – SPI2GIE SPI2 General Interrupt Enable bit

Value	Description
1	Interrupt request enabled
0	Interrupt request not enabled

Bit 3 – SPI2TXIE SPI2 Transmitter Interrupt Enable bit

Value	Description
1	Interrupt request enabled
0	Interrupt request not enabled

Bit 2 – SPI2RXIE SPI2 Receiver Interrupt Enable bit

Value	Description
1	Interrupt request enabled
0	Interrupt request not enabled

Bit 1 – SPI1GIE SPI1 General Interrupt Enable bit

Value	Description
1	Interrupt request enabled
0	Interrupt request not enabled

Bit 0 – SPI1TXIE SPI1 Transmitter Interrupt Enable bit

Value	Description
1	Interrupt request enabled
0	Interrupt request not enabled

10.4.21 Interrupt Enable Register 3

Name: IEC3
Offset: 0xC0

Bit	31	30	29	28	27	26	25	24
			QEI1IE					
Access			R/W					
Reset			0					

Bit	23	22	21	20	19	18	17	16
				CNDIE	CNCIE	CNBIE	CNAIE	
Access				R/W	R/W	R/W	R/W	
Reset				0	0	0	0	

Bit	15	14	13	12	11	10	9	8
		DMA5IE	DMA4IE	SENT2EIE	SENT2IE	SENT1EIE	SENT1IE	
Access		R/W	R/W	R/W	R/W	R/W	R/W	
Reset		0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0
				U3EVTIE	U3EIE	U3TXIE	U3RXIE	U2EVTIE
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit 29 – QEI1IE QEI1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 20 – CNDIE Change Notice D Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 19 – CNCIE Change Notice C Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 18 – CNBIE Change Notice B Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 17 – CNAIE Change Notice A Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 14 – DMA5IE Direct Memory Access 5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 13 – DMA4IE Direct Memory Access 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 12 – SENT2EIE SENT2 Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 11 – SENT2IE SENT2 TX/RX Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 10 – SENT1EIE SENT1 Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 9 – SENT1IE SENT1 TX/RX Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 4 – U3EVTIE UART3 Event Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 3 – U3EIE UART3 Framing Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 2 – U3TXIE UART3 Transmit Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 1 – U3RXIE UART3 Receive Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 0 – U2EVTIE UART2 Event Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

10.4.22 Interrupt Enable Register 4

Name: IEC4
Offset: 0xC4

Bit	31	30	29	28	27	26	25	24
	AD1CMP6IE	AD1CH6IE	AD1CMP5IE	AD1CH5IE	AD1CMP4IE	AD1CH4IE	AD1CMP3IE	AD1CH3IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	AD1CMP2IE	AD1CH2IE	AD1CMP1IE	AD1CH1IE	AD1CMP0IE	AD1CH0IE		
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bit	15	14	13	12	11	10	9	8
					PTG3IE	PTG2IE	PTG1IE	PTG0IE
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGWDIE	PTGSTEPIE	ICDIE		CRCIE	BISS1EIE	BISS1TXIE	
Access	R/W	R/W	R/W		R/W	R/W	R/W	
Reset	0	0	0		0	0	0	

Bit 31 – AD1CMP6IE ADC 1 Digital Comparator 6 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 30 – AD1CH6IE ADC 1 Channel 6 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 29 – AD1CMP5IE ADC 1 Digital Comparator 5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 28 – AD1CH5IE ADC 1 Channel 5 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 27 – AD1CMP4IE ADC 1 Digital Comparator 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 26 – AD1CH4IE ADC 1 Channel 4 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 25 – AD1CMP3IE ADC 1 Digital Comparator 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 24 – AD1CH3IE ADC 1 Channel 3 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 23 – AD1CMP2IE ADC 1 Digital Comparator 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 22 – AD1CH2IE ADC 1 Channel 2 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 21 – AD1CMP1IE ADC 1 Digital Comparator 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 20 – AD1CH1IE ADC 1 Channel 1 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 19 – AD1CMP0IE ADC 1 Digital Comparator 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 18 – AD1CH0IE ADC 1 Channel 0 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 11 – PTG3IE PTG3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 10 – PTG2IE PTG2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 9 – PTG1IE PTG1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 8 – PTG0IE PTG1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 7 – PTGWDIE PTG WDT Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 6 – PTGSTEPIE PTG Step Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 5 – ICDIE ICD Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 3 – CRCIE CRC Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 2 – BISS1EIE BiSS 1 Transmission Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 1 – BISS1TXIE BiSS 1 Transmission Complete Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

10.4.23 Interrupt Enable Register 5

Name: IEC5
Offset: 0xC8

Bit	31	30	29	28	27	26	25	24
	AD2CMP1IE	AD2CH1IE	AD2CMP0IE	AD2CH0IE			AD1CMP19IE	AD1CH19IE
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	23	22	21	20	19	18	17	16
	AD1CMP18IE	AD1CH18IE	AD1CMP17IE	AD1CH17IE	AD1CMP16IE	AD1CH16IE	AD1CMP15IE	AD1CH15IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	AD1CMP14IE	AD1CH14IE	AD1CMP13IE	AD1CH13IE	AD1CMP12IE	AD1CH12IE	AD1CMP11IE	AD1CH11IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	AD1CMP10IE	AD1CH10IE	AD1CMP9IE	AD1CH9IE	AD1CMP8IE	AD1CH8IE	AD1CMP7IE	AD1CH7IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – AD2CMP1IE ADC 2 Digital Comparator 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 30 – AD2CH1IE ADC 2 Channel 1 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 29 – AD2CMP0IE ADC 2 Digital Comparator 0 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 28 – AD2CH0IE ADC 2 Channel 0 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 25 – AD1CMP19IE ADC 1 Digital Comparator 19 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 24 – AD1CH19IE ADC 1 Channel 19 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 23 – AD1CMP18IE ADC 1 Digital Comparator 18 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 22 – AD1CH18IE ADC 1 Channel 18 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 21 – AD1CMP17IE ADC 1 Digital Comparator 17 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 20 – AD1CH17IE ADC 1 Channel 17 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 19 – AD1CMP16IE ADC 1 Digital Comparator 16 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 18 – AD1CH16IE ADC 1 Channel 16 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 17 – AD1CMP15IE ADC 1 Digital Comparator 16 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 16 – AD1CH15IE ADC 1 Channel 15 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 15 – AD1CMP14IE ADC 1 Digital Comparator 14 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 14 – AD1CH14IE ADC 1 Channel 14 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 13 – AD1CMP13IE ADC 1 Digital Comparator 13 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 12 – AD1CH13IE ADC 1 Channel 13 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 11 – AD1CMP12IE ADC 1 Digital Comparator 12 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 10 – AD1CH12IE ADC 1 Channel 12 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 9 – AD1CMP11IE ADC 1 Digital Comparator 16 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 8 – AD1CH11IE ADC 1 Channel 11 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 7 – AD1CMP10IE ADC 1 Digital Comparator 10 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 6 – AD1CH10IE ADC 1 Channel 10 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 5 – AD1CMP9IE ADC 1 Digital Comparator 9 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 4 – AD1CH9IE ADC 1 Channel 9 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 3 – AD1CMP8IE ADC 1 Digital Comparator 8 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 2 – AD1CH8IE ADC 1 Channel 8 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 1 – AD1CMP7IE ADC 1 Digital Comparator 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 0 – AD1CH7IE ADC 1 Channel 7 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

10.4.24 Interrupt Enable Register 6

Name: IEC6
Offset: 0xCC

Bit	31	30	29	28	27	26	25	24
	AD2CMP17IE	AD2CH17IE	AD2CMP16IE	AD2CH16IE	AD2CMP15IE	AD2CH15IE	AD2CMP14IE	AD2CH14IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	AD2CMP13IE	AD2CH13IE	AD2CMP12IE	AD2CH12IE	AD2CMP11IE	AD2CH11IE	AD2CMP10IE	AD2CH10IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	AD2CMP9IE	AD2CH9IE	AD2CMP8IE	AD2CH8IE	AD2CMP7IE	AD2CH7IE	AD2CMP6IE	AD2CH6IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	AD2CMP5IE	AD2CH5IE	AD2CMP4IE	AD2CH4IE	AD2CMP3IE	AD2CH3IE	AD2CMP2IE	AD2CH2IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – AD2CMP17IE ADC 2 Digital Comparator 17 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 30 – AD2CH17IE ADC 2 Channel 17 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 29 – AD2CMP16IE ADC 2 Digital Comparator 16 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 28 – AD2CH16IE ADC 2 Channel 16 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 27 – AD2CMP15IE ADC 2 Digital Comparator 15 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 26 – AD2CH15IE ADC 2 Channel 15 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 25 – AD2CMP14IE ADC 2 Digital Comparator 14 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 24 – AD2CH14IE ADC 2 Channel 14 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 23 – AD2CMP13IE ADC 2 Digital Comparator 13 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 22 – AD2CH13IE ADC 2 Channel 13 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 21 – AD2CMP12IE ADC 2 Digital Comparator 12 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 20 – AD2CH12IE ADC 2 Channel 12 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 19 – AD2CMP11IE ADC 2 Digital Comparator 11 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 18 – AD2CH11IE ADC 2 Channel 11 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 17 – AD2CMP10IE ADC 2 Digital Comparator 10 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 16 – AD2CH10IE ADC 2 Channel 10 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 15 – AD2CMP9IE ADC 2 Digital Comparator 9 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 14 – AD2CH9IE ADC 2 Channel 9 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 13 – AD2CMP8IE ADC 2 Digital Comparator 8 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 12 – AD2CH8IE ADC 2 Channel 8 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 11 – AD2CMP7IE ADC 2 Digital Comparator 7 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 10 – AD2CH7IE ADC 2 Channel 7 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 9 – AD2CMP6IE ADC 2 Digital Comparator 6 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 8 – AD2CH6IE ADC 2 Channel 6 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 7 – AD2CMP5IE ADC 2 Digital Comparator 5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 6 – AD2CH5IE ADC 2 Channel 5 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 5 – AD2CMP4IE ADC 2 Digital Comparator 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 4 – AD2CH4IE ADC 2 Channel 4 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 3 – AD2CMP3IE ADC 2 Digital Comparator 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 2 – AD2CH3IE ADC 2 Channel 3 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 1 – AD2CMP2IE ADC 2 Digital Comparator 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 0 – AD2CH2IE ADC 2 Channel 2 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

10.4.25 Interrupt Enable Register 7

Name: IEC7
Offset: 0xD0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CLC4NIE	CLC4PIE	CLC3NIE	CLC3PIE	CLC2NIE	CLC2PIE	CLC1NIE	CLC1PIE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
					AD2CMP19IE	AD2CH19IE	AD2CMP18IE	AD2CH18IE
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 15 – CLC4NIE CLC4 Negative Edge Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 14 – CLC4PIE CLC4 Positive Edge Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 13 – CLC3NIE CLC3 Negative Edge Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 12 – CLC3PIE CLC7 Positive Edge Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 11 – CLC2NIE CLC2 Negative Edge Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 10 – CLC2PIE CLC2 Positive Edge Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 9 – CLC1NIE CLC1 Negative Edge Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 8 – CLC1PIE CLC1 Positive Edge Interrupt Enable bit

Value	Description
1	Interrupt enabled
0	Interrupt not enabled

Bit 3 – AD2CMP19IE ADC 2 Digital Comparator 19 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 2 – AD2CH19IE ADC 2 Channel 19 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 1 – AD2CMP18IE ADC 2 Digital Comparator 18 Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

Bit 0 – AD2CH18IE ADC 2 Channel 18 Conversion Ready bit

Value	Description
1	Interrupt is enabled
0	Interrupt is not enabled

10.4.26 Interrupt Enable Register 8

Name: IEC8
Offset: 0xD4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		IOMON4IE	IOMON3IE	IOMON2IE	IOMON1IE			
Access		R/W	R/W	R/W	R/W			
Reset		0	0	0	0			

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 22 – IOMON4IE IOM 4 Interrupt Enable bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 21 – IOMON3IE IOM 3 Interrupt Enable bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 20 – IOMON2IE IOM 2 Interrupt Enable bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bit 19 – IOMON1IE IOM 1 Interrupt Enable bit

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

10.4.27 Interrupt Priority Register 0

Name: IPC0
Offset: 0xD8

Bit	31	30	29	28	27	26	25	24
		NVMCRCIP[2:0]				NVMIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		NVMECCIP[2:0]				PBERRIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		YRAMECCIP[2:0]				XRAMECCIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		CPUFPUIP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bits 30:28 – NVMCRCIP[2:0] NVM CRC Complete Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – NVMIP[2:0] NVM Erase/Program Operation Complete Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – NVMECCIP[2:0] NVM ECC Single Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – PBERRIP[2:0] PBU Parity Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – YRAMECCIP[2:0] Y RAM ECC Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – XRAMECCIP[2:0] X RAM ECC Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – CPUFPUIP[2:0] FPU Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.28 Interrupt Priority Register 1

Name: IPC1
Offset: 0xDC

Bit	31	30	29	28	27	26	25	24
		C2FAILIP[2:0]				C1RDYIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
						C1MONIP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	15	14	13	12	11	10	9	8
		C1FAILIP[2:0]				CLKERRIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		CLKFAILIP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bits 30:28 – C2FAILIP[2:0] Clock 2 Failure Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – C1RDYIP[2:0] Clock 1 Ready Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – C1MONIP[2:0] Clock 1 Monitor Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – C1FAILIP[2:0] Clock 1 Failure Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – CLKERRIP[2:0] Clock Error (combined) Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – CLKFAILIP[2:0] Clock Fail (Combined) Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.29 Interrupt Priority Register 2

Name: IPC2
Offset: 0xE0

Bit	31	30	29	28	27	26	25	24
		C4FAILIP[2:0]				C3RDYIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
						CWARM3IP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	15	14	13	12	11	10	9	8
		CLKFAIL3IP[2:0]				C2RDYIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		C2MONIP[2:0]				C2WARMIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – C4FAILIP[2:0] Clock 4 Failure Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – C3RDYIP[2:0] Clock 3 Ready Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – CWARM3IP[2:0] Clock 3 Warming Complete Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – CLKFAIL3IP[2:0] Clock-3 Fail Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – C2RDYIP[2:0] Clock 2 Ready Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – C2MONIP[2:0] Clock 2 Monitor Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – C2WARMIP[2:0] Clock 2 Warming Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.30 Interrupt Priority Register 3

Name: IPC3
Offset: 0xE4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						WDTIP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	15	14	13	12	11	10	9	8
						C4RDYIP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	7	6	5	4	3	2	1	0
		C4MONIP[2:0]				C4WARMIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 18:16 – WDTIP[2:0] Sleep/Idle Mode WDT Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – C4RDYIP[2:0] Clock 4 Ready Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – C4MONIP[2:0] Clock 4 Monitor Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – C4WARMIP[2:0] Clock 4 Warming Complete Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.31 Interrupt Priority Register 4

Name: IPC4
Offset: 0xE8

Bit	31	30	29	28	27	26	25	24
		PEVTBIP[2:0]				PEVTAIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		INT4IP[2:0]				INT3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		INT2IP[2:0]				INT1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		INT0IP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bits 30:28 – PEVTBIP[2:0] PWM Event B Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – PEVTAIP[2:0] PWM Event A Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – INT4IP[2:0] External Interrupt 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – INT3IP[2:0] External Interrupt 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – INT2IP[2:0] External Interrupt 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – INT1IP[2:0] External Interrupt 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – INT0IP[2:0] External Interrupt 0 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.32 Interrupt Priority Register 5

Name: IPC5
Offset: 0xEC

Bit	31	30	29	28	27	26	25	24
		PWM4IP[2:0]				PWM3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		PWM2IP[2:0]				PWM1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		PEVTFIP[2:0]				PEVTEIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		PEVTDIP[2:0]				PEVTCIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – PWM4IP[2:0] PWM4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – PWM3IP[2:0] PWM3 Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – PWM2IP[2:0] PWM2 Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – PWM1IP[2:0] PWM1 Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – PEVTFIP[2:0] PWM Event F Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – PEVTEIP[2:0] PWM Event E Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – PEVTDIP[2:0] PWM Event D Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – PEVTCIP[2:0] PWM Event C Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.33 Interrupt Priority Register 6

Name: IPC6
Offset: 0xF0

Bit	31	30	29	28	27	26	25	24
		CCT4IP[2:0]				CCP3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		0	0	1

Bit	23	22	21	20	19	18	17	16
		CCT3IP[2:0]						CCP2IP[2:0]
Access		R/W	R/W	R/W				R/W
Reset		1	0	0				1

Bit	15	14	13	12	11	10	9	8
		CCT2IP[2:0]				CCP1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		0	0	1

Bit	7	6	5	4	3	2	1	0
		CCT1IP[2:0]				T1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – CCT4IP[2:0] Capture/Compare/Timer 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – CCP3IP[2:0] CCP3 Interrupt Priority bit

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – CCT3IP[2:0] Capture/Compare/Timer 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bit 16 – CCP2IP[2:0] CCP2 Interrupt Priority bit

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – CCT2IP[2:0] Capture/Compare/Timer 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – CCP1IP[2:0] CCP1 Interrupt Priority bit

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – CCT1IP[2:0] Capture/Compare/Timer 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – T1IP[2:0] Timer 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.34 Interrupt Priority Register 7

Name: IPC7
Offset: 0xF4

Bit	31	30	29	28	27	26	25	24
		SPI1RXIP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						CCP4IP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bits 30:28 – SPI1RXIP[2:0] SPI1 Receive Done Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – CCP4IP[2:0] Input Capture/Output Compare 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.35 Interrupt Priority Register 8

Name: IPC8
Offset: 0xF8

Bit	31	30	29	28	27	26	25	24
		SPI3GIP[2:0]				SPI3TXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		SPI3RXIP[2:0]				SPI2GIP[2:0]		
Access		R/W	R/W	R/W		RW	RW	RW
Reset		1	0	0		0	0	0

Bit	15	14	13	12	11	10	9	8
		SPI2TXIP[2:0]				SPI2RXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		SPI1GIP[2:0]				SPI1TXIP[2:0]		
Access		RW	RW	RW		R/W	R/W	R/W
Reset		0	0	0		1	0	0

Bits 30:28 – SPI3GIP[2:0] SPI3 General Interrupt Priority bit

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – SPI3TXIP[2:0] SPI3 Transfer Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – SPI3RXIP[2:0] SPI3 Receive Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – SPI2GIP[2:0] SPI2 General Interrupt Priority bit

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – SPI2TXIP[2:0] SPI2 Transfer Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – SPI2RXIP[2:0] SPI2 Receive Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – SPI1GIP[2:0] SPI1 General Interrupt Priority bit

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – SPI1TXIP[2:0] SPI1 Transfer Done Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.36 Interrupt Priority Register 9

Name: IPC9
Offset: 0xFC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		CMP2IP[2:0]				CMP1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		DMA3IP[2:0]				DMA2IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		DMA1IP[2:0]				DMA0IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 22:20 – CMP2IP[2:0] Comparator 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – CMP1IP[2:0] Comparator 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – DMA3IP[2:0] Direct Memory Access 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – DMA2IP[2:0] Direct Memory Access 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – DMA1IP[2:0] Direct Memory Access 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – DMA0IP[2:0] Direct Memory Access 0 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.37 Interrupt Priority Register 10

Name: IPC10
Offset: 0x100

Bit	31	30	29	28	27	26	25	24
		I2C2TXIP[2:0]				I2C2RXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	1		0	0	1

Bit	23	22	21	20	19	18	17	16
						I2C2EIP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	15	14	13	12	11	10	9	8
		I2C1TXIP[2:0]				I2C1RXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	1		0	0	1

Bit	7	6	5	4	3	2	1	0
		I2C1IP[2:0]				I2C1EIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – I2C2TXIP[2:0] I2C2 Transmit Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – I2C2RXIP[2:0] I2C2 Receive Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – I2C2EIP[2:0] I2C2 Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – I2C1TXIP[2:0] I2C1 Transmit Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – I2C1RXIP[2:0] I2C1 Receive Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – I2C1IP[2:0] I2C1 Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – I2C1EIP[2:0] I2C1 Error Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.38 Interrupt Priority Register 11

Name: IPC11
Offset: 0x104

Bit	31	30	29	28	27	26	25	24
		U2EIP[2:0]				U2TXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		U2RXIP[2:0]				U1EVTIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		U1EIP[2:0]				U1TXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		U1RXIP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bits 30:28 – U2EIP[2:0] UART2 Framing Error Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – U2TXIP[2:0] UART2 Transmit Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – U2RXIP[2:0] UART2 Receive Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – U1EVTIP[2:0] UART1 Event Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – U1EIP[2:0] UART1 Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – U1TXIP[2:0] UART1 Transmit Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – U1RXIP[2:0] UART 1 Receive Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.39 Interrupt Priority Register 12

Name: IPC12
Offset: 0x108

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						U3EVTIP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	15	14	13	12	11	10	9	8
		U3EIP[2:0]				U3TXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		U3RXIP[2:0]				U2EVTIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 18:16 – U3EVTIP[2:0] UART3 Event Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – U3EIP[2:0] UART3 Framing Error Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – U3TXIP[2:0] UART3 Transmit Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – U3RXIP[2:0] UART 3 Receive Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – U2EVTIP[2:0] UART2 Event Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.40 Interrupt Priority Register 13

Name: IPC13
Offset: 0x10C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		DMA4IP[2:0]				SENT2EIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		SENT2IP[2:0]				SENT1EIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		SENT1IP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bits 22:20 – DMA4IP[2:0] DMA4 Interrupt Priority bits

Value	Description
1	High priority
0	Low priority

Bits 18:16 – SENT2EIP[2:0] SENT2 External Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – SENT2IP[2:0] SENT2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – SENT1EIP[2:0] SENT1 External Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – SENT1IP[2:0] SENT1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.41 Interrupt Priority Register 14

Name: IPC14
Offset: 0x110

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						CNDIP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	15	14	13	12	11	10	9	8
		CNCIP[2:0]				CNBIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		CNAIP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bits 18:16 – CNDIP[2:0] Change Notification D Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – CNCIP[2:0] Change Notification C Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – CNBIP[2:0] Change Notification B Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – CNAIP[2:0] Change Notification A Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.42 Interrupt Priority Register 15

Name: IPC15
Offset: 0x114

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		QEI1IP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 22:20 – QEI1IP[2:0] QEI1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.43 Interrupt Priority Register 16

Name: IPC16
Offset: 0x118

Bit	31	30	29	28	27	26	25	24
		PTGWDIP[2:0]				PTGSTEPIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
						ICDIP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	15	14	13	12	11	10	9	8
		CRCIP[2:0]				BISS1EIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		BISS1TXIP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bits 30:28 – PTGWDIP[2:0]

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – PTGSTEPIP[2:0] PTG Step Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – ICDIP[2:0] ICD Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – CRCIP[2:0] CRC Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – BISS1EIP[2:0] BiSS 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – BISS1TXIP[2:0] BiSS 1 Transmit Interrupt Priority bits

Value	Description
1	High priority
0	Low priority

10.4.44 Interrupt Priority Register 17

Name: IPC17
Offset: 0x11C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
		PTG3IP[2:0]				PTG2IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		PTG1IP[2:0]				PTG0IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 14:12 – PTG3IP[2:0] Peripheral Trigger Generator 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – PTG2IP[2:0] Peripheral Trigger Generator 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – PTG1IP[2:0] Peripheral Trigger Generator 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – PTG0IP[2:0] Peripheral Trigger Generator 0 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.45 Interrupt Priority Register 18

Name: IPC18
Offset: 0x120

Bit	31	30	29	28	27	26	25	24
		AD1CMP2IP[2:0]				AD1CH2IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD1CMP1IP[2:0]				AD1CH1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD1CMP0IP[2:0]				AD1CH0IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 30:28 – AD1CMP2IP[2:0] ADC 1 Digital Comparator 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD1CH2IP[2:0] ADC 1 Channel 2 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD1CMP1IP[2:0] ADC 1 Digital Comparator 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD1CH1IP[2:0] ADC 1 Channel 1 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD1CMP0IP[2:0] ADC 1 Digital Comparator 0 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD1CH0IP[2:0] ADC 1 Channel 0 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.46 Interrupt Priority Register 19

Name: IPC19
Offset: 0x124

Bit	31	30	29	28	27	26	25	24
		AD1CMP6IP[2:0]				AD1CH6IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD1CMP5IP[2:0]				AD1CH5IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD1CMP4IP[2:0]				AD1CH4IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		AD1CMP3IP[2:0]				AD1CH3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – AD1CMP6IP[2:0] ADC 1 Digital Comparator 6 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD1CH6IP[2:0] ADC 1 Channel 6 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD1CMP5IP[2:0] ADC 1 Digital Comparator 5 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD1CH5IP[2:0] ADC 1 Channel 5 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD1CMP4IP[2:0] ADC 1 Digital Comparator 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD1CH4IP[2:0] ADC 1 Channel 4 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD1CMP3IP[2:0] ADC 1 Digital Comparator 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD1CH3IP[2:0] ADC 1 Channel 3 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.47 Interrupt Priority Register 20

Name: IPC20
Offset: 0x128

Bit	31	30	29	28	27	26	25	24
		AD1CMP10IP[2:0]				AD1CH10IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD1CMP9IP[2:0]				AD1CH9IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD1CMP8IP[2:0]				AD1CH8IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		AD1CMP7IP[2:0]				AD1CH7IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – AD1CMP10IP[2:0] ADC 1 Digital Comparator 10 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD1CH10IP[2:0] ADC 1 Channel 10 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD1CMP9IP[2:0] ADC 1 Digital Comparator 9 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD1CH9IP[2:0] ADC 1 Channel 9 Conversion Ready bit

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD1CMP8IP[2:0] ADC 1 Digital Comparator 8 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD1CH8IP[2:0] ADC 1 Channel 8 Conversion Ready bit

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD1CMP7IP[2:0] ADC 1 Digital Comparator 7 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD1CH7IP[2:0] ADC 1 Channel 7 Conversion Ready bit

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.48 Interrupt Priority Register 21

Name: IPC21
Offset: 0x12C

Bit	31	30	29	28	27	26	25	24
		AD1CMP14IP[2:0]				AD1CH14IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD1CMP13IP[2:0]				AD1CH13IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD1CMP12IP[2:0]				AD1CH12IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		AD1CMP11IP[2:0]				AD1CH11IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – AD1CMP14IP[2:0] ADC 1 Digital Comparator 14 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD1CH14IP[2:0] ADC 1 Channel 14 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD1CMP13IP[2:0] ADC 1 Digital Comparator 13 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD1CH13IP[2:0] ADC 1 Channel 13 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD1CMP12IP[2:0] ADC 1 Digital Comparator 12 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD1CH12IP[2:0] ADC 1 Channel 12 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD1CMP11IP[2:0] ADC 1 Digital Comparator 11 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD1CH11IP[2:0] ADC 1 Channel 11 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.49 Interrupt Priority Register 22

Name: IPC22
Offset: 0x130

Bit	31	30	29	28	27	26	25	24
		AD1CMP18IP[2:0]				AD1CH18IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD1CMP17IP[2:0]				AD1CH17IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD1CMP16IP[2:0]				AD1CH16IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		AD1CMP15IP[2:0]				AD1CH15IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – AD1CMP18IP[2:0] ADC 1 Digital Comparator 18 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD1CH18IP[2:0] ADC 1 Channel 18 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD1CMP17IP[2:0] ADC 1 Digital Comparator 17 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD1CH17IP[2:0] ADC 1 Channel 17 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD1CMP16IP[2:0] ADC 1 Digital Comparator 16 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD1CH16IP[2:0] ADC 1 Channel 16 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD1CMP15IP[2:0] ADC 1 Digital Comparator 15 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD1CH15IP[2:0] ADC 1 Channel 15 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.50 Interrupt Priority Register 23

Name: IPC23
Offset: 0x134

Bit	31	30	29	28	27	26	25	24
		AD2CMP1IP[2:0]				AD2CH1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD2CMP0IP[2:0]				AD2CH0IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		AD1CMP19IP[2:0]				AD1CH19IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – AD2CMP1IP[2:0] ADC 2 Digital Comparator 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD2CH1IP[2:0] ADC 2 Channel 1 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD2CMP0IP[2:0] ADC 2 Digital Comparator 0 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD2CH0IP[2:0] ADC 2 Channel 0 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD1CMP19IP[2:0] ADC 1 Digital Comparator 19 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD1CH19IP[2:0] ADC 1 Channel 19 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.51 Interrupt Priority Register 24

Name: IPC24
Offset: 0x138

Bit	31	30	29	28	27	26	25	24
		AD2CMP5IP[2:0]				AD2CH5IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD2CMP4IP[2:0]				AD2CH4IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
						AD2CH3IP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	7	6	5	4	3	2	1	0
		AD2CMP2IP[2:0]				AD2CH2IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – AD2CMP5IP[2:0] ADC 2 Digital Comparator 5 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD2CH5IP[2:0] ADC 2 Channel 5 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD2CMP4IP[2:0] ADC 2 Digital Comparator 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD2CH4IP[2:0] ADC 2 Channel 4 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD2CH3IP[2:0] ADC 2 Channel 3 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD2CMP2IP[2:0] ADC 2 Digital Comparator 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD2CH2IP[2:0] ADC 2 Channel 2 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.52 Interrupt Priority Register 25

Name: IPC25
Offset: 0x13C

Bit	31	30	29	28	27	26	25	24
		AD2CMP9IP[2:0]				AD2CH9IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD2CMP8IP[2:0]				AD2CH8IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD2CMP7IP[2:0]				AD2CH7IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		AD2CMP6IP[2:0]				AD2CH6IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – AD2CMP9IP[2:0] ADC 2 Digital Comparator 9 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD2CH9IP[2:0] ADC 2 Channel 9 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD2CMP8IP[2:0] ADC 2 Digital Comparator 8 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD2CH8IP[2:0] ADC 2 Channel 8 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD2CMP7IP[2:0] ADC 2 Digital Comparator 7 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD2CH7IP[2:0] ADC 2 Channel 7 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD2CMP6IP[2:0] ADC 2 Digital Comparator 6 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD2CH6IP[2:0] ADC 2 Channel 6 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.53 Interrupt Priority Register 26

Name: IPC26
Offset: 0x140

Bit	31	30	29	28	27	26	25	24
		AD2CMP13IP[2:0]				AD2CH13IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD2CMP12IP[2:0]				AD2CH12IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD2CMP11IP[2:0]				AD2CH11IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		AD2CMP10IP[2:0]				AD2CH10IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – AD2CMP13IP[2:0] ADC 2 Digital Comparator 13 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD2CH13IP[2:0] ADC 2 Channel 13 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD2CMP12IP[2:0] ADC 2 Digital Comparator 12 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD2CH12IP[2:0] ADC 2 Channel 12 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD2CMP11IP[2:0] ADC 2 Digital Comparator 11 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD2CH11IP[2:0] ADC 2 Channel 11 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD2CMP10IP[2:0] ADC 2 Digital Comparator 10 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD2CH10IP[2:0] ADC 2 Channel 10 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.54 Interrupt Priority Register 27

Name: IPC27
Offset: 0x144

Bit	31	30	29	28	27	26	25	24
		AD2CMP17IP[2:0]				AD2CH17IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD2CMP16IP[2:0]				AD2CH16IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD2CMP15IP[2:0]				AD2CH15IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		AD2CMP14IP[2:0]				AD2CH14IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – AD2CMP17IP[2:0] ADC 2 Digital Comparator 17 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD2CH17IP[2:0] ADC 2 Channel 17 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD2CMP16IP[2:0] ADC 2 Digital Comparator 16 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD2CH16IP[2:0] ADC 2 Channel 16 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD2CMP15IP[2:0] ADC 2 Digital Comparator 15 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD2CH15IP[2:0] ADC 2 Channel 15 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD2CMP14IP[2:0] ADC 2 Digital Comparator 14 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD2CH14IP[2:0] ADC 2 Channel 14 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.55 Interrupt Priority Register 28

Name: IPC28
Offset: 0x148

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
		AD2CMP19IP[2:0]				AD2CH19IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		AD2CMP18IP[2:0]				AD2CH18IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 14:12 – AD2CMP19IP[2:0] ADC 2 Digital Comparator 19 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD2CH19IP[2:0] ADC 2 Channel 19 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD2CMP18IP[2:0] ADC 2 Digital Comparator 14 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD2CH18IP[2:0] ADC 2 Channel 18 Conversion Ready bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.56 Interrupt Priority Register 29

Name: IPC29
Offset: 0x14C

Bit	31	30	29	28	27	26	25	24
		CLC4NIP[2:0]				CLC4PIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		CLC3NIP[2:0]				CLC3PIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		CLC2NIP[2:0]				CLC2PIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		CLC1NIP[2:0]				CLC1PIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – CLC4NIP[2:0]

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – CLC4PIP[2:0]

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – CLC3NIP[2:0]

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – CLC3PIP[2:0]

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – CLC2NIP[2:0]

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – CLC2PIP[2:0]

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – CLC1NIP[2:0]

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – CLC1PIP[2:0]

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.57 Interrupt Priority Register 34

Name: IPC34
Offset: 0x160

Bit	31	30	29	28	27	26	25	24
						IOM4IP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	23	22	21	20	19	18	17	16
		IOM3IP[2:0]				IOM2IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		IOM1IP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 26:24 – IOM4IP[2:0]

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – IOM3IP[2:0]

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – IOM2IP[2:0]

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – IOM1IP[2:0]

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.5 Operation

10.5.1 Reset Vector

The Reset vector is a true vector without the legacy jump instruction (GOTO). The processor clears its registers in response to a Reset, which forces the PC to 0x800000. The processor then begins program execution at the fixed Reset vector at 0x800000. The Reset vector at the Reset address can redirect program execution to the appropriate start-up routine.

When a Reset is asserted, the user vector read commences. When available, the Reset vector contents are presented to the CPU. The data is then immediately transferred to the PS address bus to create the address of the first instruction to be executed.

10.5.2 Trap Vector Details

Note: Any unimplemented or unused vector locations in the IVT should be programmed with the address of a default interrupt handler routine that contains a RESET instruction.

Table 10-2. Trap Vector Details

Vector Source	MPLAB® XC-DSC Trap ISR Name	IRQ #	VECNUM	IVT Offset	Interrupt Bit Location		
					Flag	Enable	Priority
Bus Error - CPU X Data Bus error	_BusErrorTrap	N/A	2	0x8	INTCON3[0]	—	14
Bus Error - CPU Y Data Bus Error	_BusErrorTrap	N/A	2	0x8	INTCON3[1]	—	14
Bus error - DMA buserror	_BusErrorTrap	N/A	2	0x8	INTCON3[2]	—	14
Bus Error -CPU Instruction Data Bus Error	_BusErrorTrap	N/A	2	0x8	INTCON3[3]	—	14
Illegal Instruction	_IllegalInstructionTrap	N/A	3	0xC	INTCON1[2]	—	13
Address Error	_AddressErrorTrap	N/A	4	0x10	INTCON1[3]	—	12
Stack Error	_StackErrorTrap	N/A	5	0x14	INTCON1[4]	—	11
Math Error - Divide by Zero	_MathErrorTrap	N/A	6	0x18	INTCON4[0]	—	10

.....continued

Vector Source	MPLAB® XC-DSC Trap ISR Name	IRQ #	VECNUM	IVT Offset	Interrupt Bit Location		
					Flag	Enable	Priority
Math Error - Accumulator Shift Error	_MathErrorTrap	N/A	6	0x18	INTCON4[1]	—	10
Math Error - AccumulatorB Catastrophic Overflow	_MathErrorTrap	N/A	6	0x18	INTCON4[2]	—	10
Math Error - AccumulatorA Catastrophic Overflow	_MathErrorTrap	N/A	6	0x18	INTCON4[3]	—	10
Math Error - Accumulator B Overflow	_MathErrorTrap	N/A	6	0x18	INTCON4[4]	—	10
Math Error - AccumulatorA Overflow	_MathErrorTrap	N/A	6	0x18	INTCON4[5]	—	10
General error-DMT Event Trap	_GeneralTrap	N/A	7	0x1C	INTCON5[0]	—	9
General Error-WDT Run Event Trap	_GeneralTrap	N/A	7	0x1C	INTCON5[1]	—	9
General Error-XRAM PWB DED Error	_GeneralTrap	N/A	7	0x1C	INTCON5[2]	—	9
General Error-YRAM PWB DED Error	_GeneralTrap	N/A	7	0x1C	INTCON5[3]	—	9
General Error-SOFT Trap	_GeneralTrap	N/A	7	0x1C	INTCON5[31]	—	9

10.6 Interrupt Control and Status Registers

The dsPIC33AK128MC106 family devices implement the following registers for the interrupt controller:

- INTCON1
- INTCON2
- INTCON3
- INTCON4
- INTCON5
- INTTREG

10.6.1 INTCON1 through INTCON4

Global interrupt control functions are controlled from INTCON1, INTCON2, INTCON3 and INTCON4.

INTCON1 contains the Interrupt Nesting Disable bit (NSTDIS) as well as the control and status flags for the processor trap sources.

The INTCON2 register controls external interrupt request signal behavior, contains the Global Interrupt Enable bit (GIE) and the Alternate Interrupt Vector Table Enable bit (AIVTEN).

INTCON3 contains the status flags for the Auxiliary PLL and DO stack overflow status trap sources.

The INTCON4 register contains the Software Generated Hard Trap Status bit (SGHT).

10.6.2 IFSx

The IFSx registers maintain all of the interrupt request flags. Each source of interrupt has a status bit, which is set by the respective peripherals or external signal and is cleared via software.

10.6.3 IECx

The IECx registers maintain all of the interrupt enable bits. These control bits are used to individually enable interrupts from the peripherals or external signals.

10.6.4 IPCx

The IPCx registers are used to set the Interrupt Priority Level (IPL) for each source of interrupt. Each user interrupt source can be assigned to one of seven priority levels.

10.6.5 INTTREG

When an interrupt is presented to the CPU, associated details of the interrupt are latched onto the INTTREG register. INTTREG[IRQCPU] represents the interrupt IRQ request status, while INTTREG[ILR] holds the priority level of the presented interrupt. If INTTREG[VHOLD] is equal to 1, the value of INTTREG[VECNUM] represents the vector number of the presented interrupt. Otherwise, it represents the vector number of the last acknowledged interrupt.

INTTREG holds the value of ILR and VECNUM, until the next interrupt in presented to the CPU.

The interrupt sources are assigned to the IFSx, IECx and IPCx registers in the same sequence as they are listed in Table 10-1. For example, the FPU exception is shown as having a vector number of 10 and an IRQ number of 1. Thus, the FPUIF bit is placed at IFS0[1], the FPUIE bit in IEC0[1] and FPUIP[2:0] bits in IPC0(IPC0[6:4]).

10.7 Priority

10.7.1 CPU Priority

An interrupt or a trap source must have a priority level greater than the current CPU priority to initiate an exception process. The CPU priority level is defined by a 4-bit value, SR.IPL [3:0]. The IPL [3] bit can be read at any time and may be cleared by software to allow trap handlers to jump to another process without having to execute a RETFIE.

IPL3: MSb of CPU Priority Level Nibble

1 = CPU Priority ≥ 8 (trap exception underway)

0 = CPU Priority < 8 (no trap exception underway)

IPL [2:0]: CPU Interrupt Priority Level status bits

111 = All interrupts disabled

110 = Level 7 interrupts enabled

101 = Level 6 and 7 interrupts enabled

100 = Level 5 through 7 interrupts enabled

011 = Level 4 through 7 interrupts enabled

010 = Level 3 through 7 interrupts enabled

001 = Level 2 through 7 interrupts enabled

000 = Level 1 through 7 interrupts enabled

The SR.IPL[2:0] status bits are readable and writable, so the user application can modify these bits to disable all sources of interrupts below a given priority level. For example, if IPL = 011, the CPU would not be interrupted by any source with a programmed priority level of one, two or three. All user interrupt sources can be disabled by setting SR.IPL[2:0] = 111.

Trap events have a higher priority than any user interrupt source. When the IPL3 bit is set, a trap event is in progress. The IPL3 bit can be cleared, but not set, by the user application. In some applications, the IPL3 bit will need to be cleared when a trap has occurred, and it will need to be branched to an instruction other than the instruction immediately after the one that originally caused the trap to occur.

The CPU interrupt priority is automatically modified during exception processing. However, provided interrupt nesting is enabled, (INTCON1.NSTDIS= 0), IPL [2:0] are read/write bits and may also be manipulated by the user to dynamically modify the CPU interrupt priority. If interrupt nesting is disabled (INTCON1.NSTDIS = 1), IPL [2:0] shall by default be set to 0x7 and become read-only bits to prevent the user from inadvertently dropping the CPU interrupt priority (and causing any pending interrupts to nest).

10.7.2 Interrupt Priority

Each peripheral interrupt source can be assigned to one of the seven priority levels. The user assignable interrupt priority control bits for each individual interrupt are located in the Least Significant three bits of each nibble within the IPCx registers. Bit 3 of each nibble is not used and is read as a '0'. These bits define the priority level assigned to a particular interrupt. The usable priority levels are one (lowest priority) through seven (highest priority). If all the IPCx bits associated with an interrupt source are cleared, the interrupt source is effectively disabled.

More than one interrupt request source can be assigned to a specific priority level. To resolve priority conflicts within a given user-assigned level, each source of an interrupt has a natural priority order based on its location in the IVT. The lower IRQ numbered interrupt vectors have higher natural priority, while the higher numbered vectors have lower natural priority (refer to Table 10-1 for IRQ numbers of interrupt). The overall priority level for any pending source of an interrupt is first determined by the user application-assigned priority of that source in the IPCx register, then by the natural order priority within the IVT/AIVT.

Natural order priority is used only to resolve conflicts between simultaneous pending interrupts with the same user application-assigned priority level. Once the priority conflict is resolved and the exception process begins, the CPU can be interrupted only by a source with a higher user application-assigned priority. Interrupts with the same user application-assigned priority, but a higher natural order priority that becomes pending during the exception process, remain pending until the current exception process completes.

Each interrupt source can be assigned to one of seven priority levels. This enables the user application to assign a low natural order priority and a very high overall priority level to an interrupt. For example, the UART1 RX Interrupt can be assigned to priority level seven, and the External Interrupt 0 (INT0) can be assigned to priority level one, thereby giving it a very low effective priority.

Note: The user selectable priority levels start at one as the lowest priority and level seven as the highest priority. If an interrupt is programmed with a priority of zero, interrupt is disabled.

10.8 Interrupt Sequence

All interrupt event flags are sampled in the system clock by the IFSx registers. A pending interrupt request (IRQ) is indicated by the flag bit being equal to a '1' in an IFSx register. The IRQ will cause the interrupt to occur if the corresponding bit in the Interrupt Enable (IECx) register is set. For the next clock cycle, the priorities of all pending interrupt requests are evaluated.

No instruction is aborted when the CPU responds to the IRQ. When the IRQ is sampled, the instruction in progress is completed before the Interrupt Service Routine (ISR) is executed.

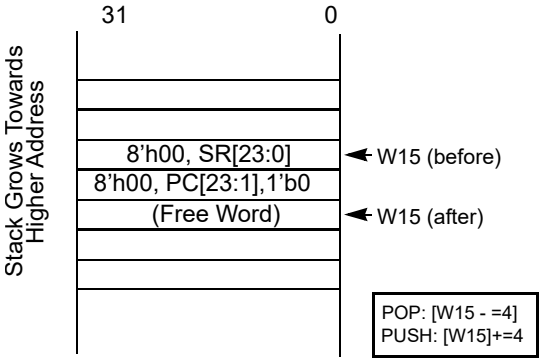
If there is a pending IRQ with a priority level greater than the current processor priority level in the processor Status register, an interrupt will be presented to the processor.

The interrupt request, its associated vector number, and the new Interrupt Priority Level are latched into ILR, VECNUM, and IRQCPU bit fields in the INTTREG register to keep them stable through the interrupt process. The new Interrupt Priority Level is the priority of the pending interrupt. The

processor reacts to the interrupt request by asserting the IACK (interrupt Acknowledge) signal to prevent the ILR, VECNUM and IRQCPU bits from changing during the interrupt process.

The processor then stacks the current program counter and the low byte of the processor status register. The low byte of the status registers contains the processor priority level at the time prior to the beginning of the interrupt cycle.

Figure 10-3. Exception Stack Frame



The processor then takes the priority level for this interrupt and loads it into the processor status register. This action will disable all lower priority interrupts until the completion of the interrupt service routine.

After servicing the interrupt, the RETFIE (Return from Interrupt) instruction will unstack the Program Counter and status registers to return the processor to its state prior to the interrupt sequence.

The processor Interrupt Priority Level (IPL) register is a 4-bit register. The IPL bits are available in the Processor Status Register (SR).

The MSB of the SR.IPL register is set if a trap is being processed. There are seven levels of user Interrupt Priority Levels, and eight priority levels for traps (two levels are reserved + six traps). If the user sets the IPL [3:0] bits to a value of seven, then all user interrupts are disabled, but the traps will still be processed. The user cannot write a value greater than seven into the processor IPL bits.

10.8.1 Peripheral Interrupt Vector Address

The vector address is determined by the value of the interrupt request number and the value of the base register bits. The base register - IVTBASE value is used to provide an offset for the base address of the interrupt vector, so that interrupt vector addresses can be mapped anywhere in user Flash or anywhere in the user RAM.

To calculate the address in the vector table where a particular peripheral interrupt vector resides, use the following equation:

address = (Base (Hex) + (IRQ number x4 (decimal)) + 24 (hex)

An example for Interrupt #21 with Base address = 0x80A000:

address = (0x80A000 (hex)+ (21x4 (dec)) +24 (hex) = 0x80A078 (hex)

An example for Interrupt #21 after reset:

address = ('800000 (hex)+ (21x4 (dec)) +24 (hex) = 800078 (hex)

10.8.2 Interrupt Nesting

Interrupts are nestable by default. Any ISR in progress can be interrupted by another source of interrupt with a higher user application-assigned priority level. Interrupt nesting can be disabled by setting the Interrupt Nesting Disable bit (NSTDIS) in the INTCON1 register. When the NSTDIS control

bit is set, all interrupts in progress force the CPU priority to Level 7 by setting IPL = 0b111. This action effectively masks all other sources of interrupt until a RETFIE instruction is executed. When interrupt nesting is disabled, the user application-assigned Interrupt Priority Levels (IPLs) have no effect except to resolve conflicts between simultaneous pending interrupts. The IPL bits (SR) become read-only when interrupt nesting is disabled. This prevents the user application from setting IPL to a lower value, which would effectively re-enable interrupt nesting.

10.9 Non-Maskable Traps

Traps are non-maskable, nestable interrupts that adhere to a fixed priority structure. Traps provide a means to correct erroneous operation during debugging and the operation of the application. If the user application does not intend to correct a trap error condition, these vectors must be loaded with the address of a software routine to reset the device. Otherwise, the user application must program the trap vector with the address of a service routine that corrects the trap condition.

The following sources of non-maskable traps are implemented in dsPIC33A devices:

- Bus Error & ECC DED Trap
- Illegal Opcode Error Trap
- CPU Address Error Trap
- CPU Stack Error Trap
- CPU Math Error Trap
- Generic Trap

For many of the trap conditions, the instruction that caused the trap is allowed to complete before exception processing begins. Therefore, the user application may have to correct the action of the instruction that caused the trap. Each trap source has a fixed priority as defined by its position in the IVT/IVTC. A bus error trap has the highest priority, while a generic trap has the lowest priority. Refer Table 10-2 for trap vector and priority details.

10.9.1 Soft Traps

Soft traps can be treated like non-maskable sources of an interrupt that adhere to the priority assigned by their position in the IVT. Soft traps are processed like interrupts and require two cycles to be sampled and acknowledged prior to exception processing. Therefore, additional instructions may be executed before a soft trap is acknowledged.

10.9.1.1 Bus Error Traps

This trap is generated if the requested operation cannot be completed due to errors on the bus. Bus matrix initiator modules will generate this error if the BMX or target macro report an error during the transaction (e.g., address errors, ECC DED errors). A bus error occurs when any of the bits within the INTCON3 register are set. Each bit within the INTCON3 register is assigned to a specific bus error condition.

10.9.1.2 Illegal Opcode Error

An illegal opcode error trap is asserted when an attempt is made to execute an illegal opcode; it is conditional upon the commitment of a speculatively executed instruction (so it cannot be asserted until the R-stage). An illegal opcode trap is asserted for:

- an attempt to execute any opcode slot within the opcode map that is not allocated an instruction. In addition to unused opcode slots, this also includes any unused sub-opcode slots.
- an attempt to PFC to the second word of a two-word instruction, using the instruction word identifier bit.
- an attempt to execute any unimplemented coprocessor instruction including any opcode that includes coprocessor select opcode bits ('zz' bits) for an unimplemented coprocessor.
- an attempt to PFC to an odd 16-bit instruction address within what is defined as a 32-bit instruction word (Most Significant opcode bit set).

10.9.1.3 CPU Address Error Trap

This trap will be taken when any of the following circumstances occurs:

1. If a W-reg (W0 through W14) is used by an AGU as a source or destination and a misaligned data word access (long word access at an odd address or word access at an odd byte address) is attempted.
   Note: For misaligned writes, the read or write is *not* inhibited. Address alignment is forced such that read or write is always aligned.
2. If a W-reg (W0 through W15) is used by an AGU:
   - as a source or destination and contains a value where the Most Significant 8-bits are not 8'h00

   or

   - as an offset where permitted and contains a value where the Most Significant 8-bits are not 8'h00 or 8'hFF (offset can be a signed negative value).
     Note: The CPU will detect X address access to SFRs and treat these as a special case. However, Y space reads of SFR or PS addresses must be detected by the BMX and result in an address error trap.
3. If the MSB of a computed PFC address (`BRAW`, `RCALLW`, `CALLW`, `GOTOW`) is not 8'h00.

10.9.1.4 Stack Error Trap

The stack is initialized to start of data RAM address (0x4000) during Reset. A stack error trap is generated if the Stack Pointer effective address (EA) is less than the initial stack value (0x4000). Stack underflow detection is to provide protection to SFR space from getting modified by the Stack Pointer. A Stack Limit register (SPLIM) associated with the Stack Pointer is uninitialized at Reset. The stack overflow check is not enabled until a word is written to the SPLIM register.

The stack error status bit (STKERR) in the INTCON1 register gets set whenever a stack error occurs. To avoid re-entry into the Trap Service Routine (TSR), the STKERR status flag must be cleared in software.

The stack error trap will be taken only when the following circumstances occur:

1. A Stack Pointer (W15) based access is attempted with an EA that is less than the start address of data RAM.
2. Stack overflow protection is enabled and a Stack Pointer-based access is attempted with an EA which is greater than the (user programmable) limit value written into the SPLIM register.
3. A pre/post increment/decrement operation is performed on W15 (including Stack Pointer modification during exception processing) that results in EA[1:0] != 2'b00 (i.e., not long word aligned). This will detect byte and word pre/post increment /decrement operations that are otherwise considered aligned but would result in a misaligned Stack Pointer.

10.9.1.5 Math Error Traps

The math error trap will execute under the circumstances listed below. The associated math error status and enable bits are located in INTCON4 register.

1. Divide By Zero: Should an attempt be made to divide by zero, the PC stack will point to the iterated instruction being executed at the time the error is detected. The divide iterations, up to the point that exception processing occurs, will execute as usual (though with meaningless results).
2. DSP Overflow: If the following conditions are all true, an Overflow of AccA (OVA) math error trap will be taken.
   a. OVATE bit is set (INTCON4[21])
   b. Accumulator A is operating with 1.63 saturation disabled or in 9.63 mode
   c. An arithmetic operation caused an overflow from bit 63 of accumulator A

3. DSP Overflow: If the following conditions are all true, an Overflow of AccB (OVB) math error trap will be taken.
   a. OVBTE bit is set (INTCON4[20])
   b. Accumulator B is operating with 1.63 saturation disabled or in 9.63 mode
   c. An arithmetic operation caused an overflow from bit 63 of accumulator B
4. DSP Overflow: If the following conditions are all true, an accumulator Catastrophic Overflow (COV) math error trap will be taken.
   a. COVTE bit is set (INTCON4[19])
   b. Either Accumulator A or B is operating with all saturation disabled
   c. An arithmetic operation caused an overflow from bit 71 (catastrophic overflow) of the accumulator with all saturation disabled
5. DSP Shift Out of Range: If an attempt is made to execute SFTAC with a shift value of greater than 32 or less than -32, the instruction will complete (without a result write) and a math trap will be generated.

10.9.1.6 Generic Trap

A generic trap occurs when any of the bits within the INTCON5 register are set. Each bit within the INTCON5 register is assigned to a specific trap error condition. Generic traps include WDT (run mode), DMT, DMA address error and software generated traps.

10.9.1.7 PCTRAP

In any trap event, the Trap Origination Address register (PCTRAP) is loaded with the value of the PC associated with the instruction that caused the trap. The origination address of a trap is captured in the PCTRAP register, given that the current CPU IPL at the time of the trap is less than eight.

Further PCTRAP updates are blocked after the first PCTRAP address capture, preventing newer traps from overwriting the source address of older ones. The register needs to be written with value 24'h000000 for trap address capture to be re-enabled.

This feature is primarily intended to aid system debugging and to locate the cause of system traps.

Note: PCTRAP captures the trap origination address of the system error caused by CPU only.

10.10 Interrupt Operations

10.10.1 Disabling Interrupts

10.10.1.1 DISI Inhibition Of Interrupts

The 16-bit CPU DISI instruction is replaced with DISICTL instruction in the dsPIC33A core devices, and they provide a means to disable interrupts around blocks of critical code. In addition, they allow the user to select an IPL threshold (IPLT) at which interrupts are disabled. The IPLT may be set to any value between IPL 0 (no interrupts disabled) and IPL 7 (all interrupts disabled). Interrupt requests at an IPL that is at or below the selected IPLT will be inhibited. These requests will remain pending until such time that the IPLT is lowered to a level less than the IPL of the pending interrupts. The IPLT may be defined using a 3-bit literal or register direct Wns source.

10.10.1.2 Global Interrupt Disable

A Global Interrupt Enable bit (GIE) is used to enable or disable all interrupts globally. When the GIE bit is cleared, it causes the interrupt controller to behave as if the CPU's SR.IPL bits are set to seven and disables all interrupts except the traps. When the GIE bit is set again, the interrupt controller acts based on the actual value of SR.IPL; the system will return to the previous operating state, depending on the prior interrupt priority bit settings.

Setting the interrupt level literal to 7 will disable all interrupts, and therefore has essentially the same effect (hazards and operational timing aside) as clearing the Global Interrupt Enable (INTCON1.GIE) bit within the interrupt controller.

10.10.2 External Interrupt Requests

The interrupt controller supports up to five external interrupt request signals INT0 - INT4. These inputs are edge sensitive, they require a low to high, or a high to low transition to create an interrupt request. The INTCON2 register has five bits INT0EP - INT4EP that select the polarity of the edge detection circuitry. Each external interrupt pin can be programmed to interrupt the CPU on a rising edge or falling edge event. INT0-INT2 can also be used as channel trigger sources for DMA or synchronization sources for CCP.

10.10.3 Wake-Up From Sleep, Idle

When an interrupt or trap request is received by the interrupt controller, a wake-up signal will be presented to the processor to wake the processor up. The processor will wake up from Sleep or Idle mode and resume operation.

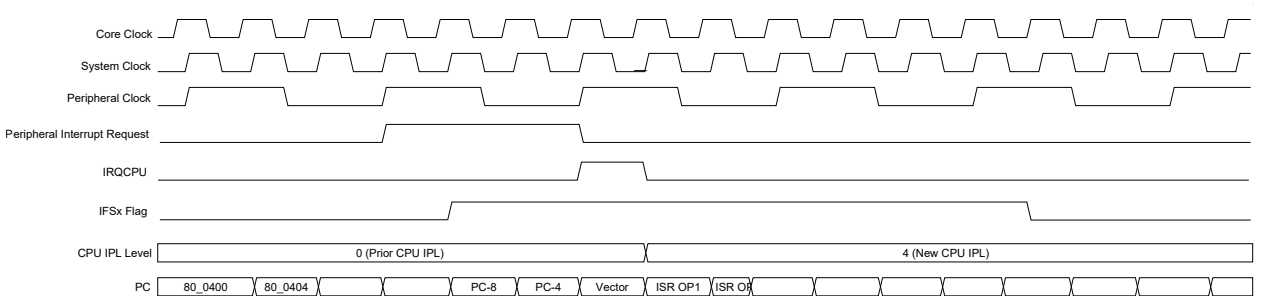
When the device wakes from Sleep or Idle mode, one of two actions occur:

- If the IPL for that source is greater than the current CPU priority level, the processor will process the interrupt and branch to the ISR for the interrupt source.
- If the user application-assigned IPL for the source is lower than or equal to the current CPU priority level, the processor will continue execution, starting with the instruction immediately following the `PWRSAV` instruction that previously put the CPU in Sleep or Idle mode.

10.10.4 Interrupt Processing Timing

The interrupt is sampled in the interrupt controller on the rising edge of the system clock into the IFSx registers. The priority resolution occurs during the next two clock cycles. The resolved interrupt request signal from the interrupt controller to the processor is then presented in the next system clock cycle. The interrupt is then acknowledged during the next system clock cycle.

Figure 10-4. Interrupt Latency



10.10.4.1 Interrupt Latency

- **Interrupt Latency: CPU Highest Priority Bus Main:**

If the CPU is set to be the highest priority RAM bus Main (Bus Matrix register BMXINITPR[31:0] = 32'b0), the CPU will offer a variable latency response for all exceptions solely based upon the execution time of the instructions underway (that are completed) at the time of the exception. The interrupt latency from the time when the system clock samples the pending interrupts to the time the first instruction of the ISR has been fetched will be:

max. $latency_{VAR=1} = t_{arb} + \rho + \eta + \Delta$ cycles

min. $latency_{VAR=1} = t_{arb} + 1 + \eta + \Delta$ cycles

Where:

t_{arb} Arbitration time (cycles)

ρ = Total instruction execution time during exception processing (cycles)

η = Vector memory access time (cycles)

Δ = Program memory access time (cycles)

The above relationship applies for exceptions occurring during any instruction, including during a PS access. The latency is expressed as a range for any given instruction because the interrupt may arrive at the beginning or end of an instruction.

- **Interrupt Latency: CPU Not Highest Priority Main:**

If the CPU is not the highest priority RAM bus Main, the CPU will offer a variable latency response for all exceptions that may also include additional delays resulting from higher priority bus master RAM access requests. That is, when the CPU is not operating as the highest priority bus Main, exception processing is no longer an atomic operation and may be stalled as necessary to provide bus access to another Main.

10.10.4.1.1 Interrupt Return Latency

To return from an interrupt, the program must call the RETFIE instruction. The RETFIE instruction in the dsPIC33A family of devices is expected to take around seven to ten cycles with PBU enabled.

During the first two cycles of a RETFIE instruction, the contents of the PC and the SR register are popped from the stack. After unstacking the SR (in the second cycle), the RETFIE instruction will change context and CPU.IPL level to that defined by the unstacked SR value. The new context will be immediately available at the start of execution of the next instruction.

The third instruction cycle is used to fetch the instruction addressed by the updated Program Counter. This cycle is executed as a `NOP` instruction. On the fourth cycle, program execution resumes at the point where the interrupt occurred.

10.10.5 Interrupt Setup Procedures

10.10.5.1 Initialization

To configure an interrupt source, complete the following steps:

1. If nested interrupts are not desired, set the NSTDIS control bit (INTCON1).
2. Select the user application-assigned priority level for the interrupt source by writing to the control bits in the appropriate IPCx control register. The priority level depends on the specific application and type of the interrupt source. If multiple priority levels are not desired, the IPCx register control bits for all enabled interrupt sources can be programmed to the same non-zero value.
3. Clear the interrupt flag status bit associated with the peripheral in the associated IFSx status register.
4. Enable the interrupt source by setting the interrupt enable control bit associated with the source in the appropriate IECx control register.

10.10.5.2 Interrupt Service Routine

The method used to declare an ISR and initialize the IVT/AIVT with the correct vector address depends on the programming language (C or Assembly) and the language development tool suite used to develop the application.

In general, the user application must clear the interrupt flag in the appropriate IFSx register for the source of the interrupt that the ISR handles. Otherwise, the application will re-enter the ISR immediately after it exits the routine. If the ISR is coded in Assembly language, it must be terminated using a RETFIE instruction to unstack the saved PC value, SRL value and old CPU priority level.

10.10.5.3 Trap Service Routine

A Trap Service Routine is coded like an ISR, except that the appropriate trap status flag in the INTCONx register must be cleared to avoid re-entry into the Trap Service Routine.

10.10.5.4 Code Examples

Example 10-1. Enable Interrupts

```
void enableInterrupts (void)
{
/* Enable level 1-7 interrupts */
/* No restoring of previous CPU IPL state performed here */
INTCON1bits.GIE = 1;
}
```

Example 10-2. Disable Interrupts

```
void disableInterrupts (void)
{
/* Disable level 1-7 interrupts */
/* No restoring of previous CPU IPL state performed here */
INTCON1bits.GIE = 0;
}
```

Example 10-3. ISR for Timer 1 Interrupt

```
void __attribute__((__interrupt__)) _T1Interrupt(void)
{
/* Insert ISR Code Here*/
/* Clear Timer1 interrupt */
IFS1bits.T1IF = 0;
 }
```

11. I/O Ports with Edge Detect

The dsPIC33A family devices include multiple general purpose I/O ports to interface to external circuitry. The ports are highly configurable for a wide range of digital and analog applications. The edge detect feature allows sensing pin changes while in sleep, or it can generate an interrupt to remove the need for the CPU polling pin state. The ports also include an integrity monitor function to verify the pin state in critical applications. The I/O integrity check feature is available on a subset of pins.

The key features of the I/O ports with edge detect module are:

- Pin control and pin configuration
- Flexible remapping of input and output signals
- Open Drain Operation
- Configurable Pin pull up or pull down capability
- Change Notification:
  - Monitors for state change on device pins
  - Detects change with respect to the last PORT value
  - Detects positive and/or negative edge events
  - Individual change status for each pin
  - Generates an interrupt event per port
  - Operates in Sleep mode
- Slew Rate Control

The key features of the I/O integrity module are:

- Fault detection:
  - Detect short circuits in the signal path
  - Detect open circuits in the signal path
  - Detect faulty signal conditions
  - Detect external tampering events
  - Device interrupt, trap or reset upon fault detection
- Configurable blanking delay
- Fault injection capability

11.1 Device-Specific Information

Table 11-1. PORTA Availability

Device	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
64-Pin	15:8					✔	✔	✔	✔
	7:0	✔	✔	✔	✔	✔	✔	✔	✔
48-Pin	15:8							✔	✔
	7:0	✔	✔	✔	✔	✔	✔	✔	✔
36-Pin	15:8								
	7:0		✔	✔	✔	✔	✔	✔	✔
28-Pin	15:8								
	7:0				✔	✔	✔	✔	✔

Table 11-2. PORTB Availability

Device	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
64-Pin	15:8					✔	✔	✔	✔
	7:0	✔	✔	✔	✔	✔	✔	✔	✔
48-Pin	15:8							✔	✔
	7:0	✔	✔	✔	✔	✔	✔	✔	✔
36-Pin	15:8								
	7:0	✔	✔	✔	✔	✔	✔	✔	✔
28-Pin	15:8								
	7:0				✔	✔	✔	✔	✔

Table 11-3. PORTC Availability

Device	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
64-Pin	15:8					✔	✔	✔	✔
	7:0	✔	✔	✔	✔	✔	✔	✔	✔
48-Pin	15:8								
	7:0	✔	✔	✔	✔	✔	✔	✔	✔
36-Pin	15:8								
	7:0			✔	✔	✔	✔	✔	✔
28-Pin	15:8								
	7:0				✔	✔	✔	✔	✔

Table 11-4. PORTD Availability

Device	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
64-Pin	15:8				✔	✔	✔	✔	✔
	7:0	✔	✔	✔	✔	✔	✔	✔	✔
48-Pin	15:8								✔
	7:0	✔	✔	✔	✔	✔	✔	✔	✔
36-Pin	15:8					✔			
	7:0				✔	✔	✔	✔	✔
28-Pin	15:8								
	7:0					✔	✔	✔	✔

Table 11-5. ANSELA Availability

Name	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
ANSELA	15:8					✔	✔	✔	✔
	7:0	✔	✔	✔	✔	✔	✔	✔	✔

Note:

1. ANSEL bit availability is dependent on existence of the port bit given a device variant as indicated in Table 11-1 through Table 11-4.

Table 11-6. ANSELB Availability

Name	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
ANSELB	15:8							✔	✔
	7:0	✔	✔	✔	✔	✔	✔	✔	✔

Note:

1. ANSEL bit availability is dependent on existence of the port bit given a device variant as indicated in Table 11-1 through Table 11-4.

Table 11-7. ANSELC Availability

Name	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
ANSELB	15:8								
	7:0								

Note:

1. ANSEL bit availability is dependent on existence of the port bit given a device variant as indicated in Table 11-1 through Table 11-4.

Table 11-8. ANSELD Availability

Name	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
ANSELB	15:8								
	7:0								

Note:

1. ANSEL bit availability is dependent on existence of the port bit given a device variant as indicated in Table 11-1 through Table 11-4.

Table 11-9. PPS Availability by Package

64-Pin	48-Pin	36-Pin	28-Pin
RP1-RP12	RP1-RP10	RP1-RP7	RP1-RP5
RP17-RP28	RP17-RP24	RP17-RP24	RP17-RP21
RP33-RP44	RP33-RP40	RP33-RP38	RP33-RP37
RP49-RP61	RP49-RP57	RP49-RP53	RP49-RP52

Table 11-10. Selectable Input Sources (Maps Input to Function)

Input Name(1)	Function Name	Register	Register Bitfield
External Interrupt 1	INT1	RPINR0	INT1R[7:0]
External Interrupt 2	INT2	RPINR0	INT2R[7:0]
External Interrupt 3	INT3	RPINR0	INT3R[7:0]
External Interrupt 4	INT4	RPINR1	INT4R[7:0]
Timer1 External Clock	T1CK	RPINR1	T1CKR[7:0]
Reference Clock Input 1	REFI1	RPINR1	REFI1R[7:0]
Reference Clock Input 2	REFI2	RPINR1	REFI1R[7:0]
SCCP Input Capture 1	ICM1	RPINR2	ICM1R[7:0]
SCCP Input Capture 2	ICM2	RPINR2	ICM2R<7:0>
SCCP Input Capture 3	ICM3	RPINR2	ICM3R[7:0]

Note:

1. Unless otherwise noted, all inputs use the Schmitt Trigger input buffers.

.....continued

Input Name[1]	Function Name	Register	Register Bitfield
SCCP Input Capture 4	ICM4	RPINR2	ICM4R[7:0]
SCCP Fault A	OCFA	RPINR5	OCFAR[7:0]
SCCP Fault B	OCFB	RPINR5	OCFBR[7:0]
SCCP Fault C	OCFC	RPINR5	OCFCR[7:0]
SCCP Fault D	OCFD	RPINR5	OCFDR[7:0]
PWM Input 8	PCI8	RPINR6	PCI8R[7:0]
PWM Input 9	PCI9	RPINR6	PCI9R[7:0]
PWM Input 10	PCI10	RPINR6	PCI10R[7:0]
PWM Input 11	PCI11	RPINR6	PCI11R[7:0]
QEI1 Input A	QEIA1	RPINR7	QEIA1R[7:0]
QEI1 Input B	QEIB1	RPINR7	QEIB1R[7:0]
QEI Index 1 Input	QEINDX1	RPINR7	QEINDX1R[7:0]
QEI Home 1 Input	QEIHOM1	RPINR7	QEIHOM1R[7:0]
UART1 Receive	U1RX	RPINR9	U1RXR[7:0]
UART1 Data-Set-Ready	U1DSR	RPINR9	U1DSRR[7:0]
UART2 Receive	U2RX	RPINR9	U2RXR[7:0]
UART2 Data-Set-Ready	U2DSR	RPINR9	U2DSRR[7:0]
SPI1 Data Input	SDI1	RPINR10	SDI1R[7:0]
SPI1 Clock Input	SCK1IN	RPINR10	SCK1R[7:0]
SPI1 Client Select	SS1	RPINR10	SS1R[7:0]
SPI2 Data Input	SDI2	RPINR11	SDI2R[7:0]
SPI2 Clock Input	SCK2IN	RPINR11	SCK2R[7:0]
SPI2 Client Select	SS2	RPINR11	SS2R[7:0]
UART3 Receive	U3RX	RPINR13	U3RXR[7:0]
UART3 Data-Set-Ready	U3DSR	RPINR13	U3DSRR[7:0]
SENT1 Input	SENT1	RPINR14	SENT1R[7:0]
SENT2 Input	SENT2	RPINR14	SENT2R[7:0]
SPI3 Data Input	SDI3	RPINR15	SDI3R[7:0]
SPI3 Clock Input	SCK3IN	RPINR15	SCK3R[7:0]
SPI3 Client Select	SS3	RPINR15	SS3R[7:0]
PWM Input 12	PCI12	RPINR17	PCI12R[7:0]
PWM Input 13	PCI13	RPINR17	PCI13R[7:0]
PWM Input 12	PCI14	RPINR17	PCI17R[7:0]
PWM Input 15	PCI15	RPINR17	PCI17R[7:0]
PWM Input 16	PCI16	RPINR18	PCI16R[7:0]
PWM Input 17	PCI17	RPINR18	PCI17R[7:0]
PWM Input 18	PCI18	RPINR18	PCI18R[7:0]
ADC Trigger 31 Input	ADTRIG31	RPINR18	ADTRIG31R[7:0]
BiSS Return Input	BISS1SL	RPINR19	BISS1SLR[7:0]
BiSS Get Sense Input	BISS1GS	RPINR19	BISS1GSR[7:0]
CLC Input A	CLCINA	RPINR20	CLCINAR[7:0]
CLC Input B	CLCINB	RPINR20	CLCINBR[7:0]
CLC Input C	CLCINC	RPINR20	CLCINCR[7:0]

Note:

1. Unless otherwise noted, all inputs use the Schmitt Trigger input buffers.

.....continued			
Input Name(1)	**Function Name**	**Register**	**Register Bitfield**
CLC Input D	CLCIND	RPINR20	CLCINDR[7:0]
UART1 Clear to Send	U1CTS	RPINR21	U1CTS[7:0]
UART2 Clear to Send	U2CTS	RPINR21	U2CTS[7:0]
UART3 Clear to Send	U3CTS	RPINR21	U3CTS[7:0]

Note:
1. Unless otherwise noted, all inputs use the Schmitt Trigger input buffers.

Table 11-11. Pin Correlation to Input Remap #(1)

Index	Source Output Signal	Remap Input #
193	RPV15	RPI193
192	RPV14	RPI192
191	RPV13	RPI191
190	RPV12	RPI190
189	RPV11	RPI189
188	RPV10	RPI188
187	RPV9	RPI187
186	RPV8	RPI186
185	RPV7	RPI185
184	RPV6	RPI184
183	RPV5	RPI183
182	RPV4	RPI182
181	RPV3	RPI181
180	RPV2	RPI180
179	RPV1	RPI179
178	RPV0	RPI178
177	DAC3 pwm_req_off	RPI177
176	DAC3 pwm_req_on	RPI176
175	DAC2 pwm_req_off	RPI175
174	DAC2 pwm_req_on	RPI174
173	DAC1 pwm_req_off	RPI173
172	DAC1 pwm_req_on	RPI172
171	PEVTF	RPI171
170	PEVTE	RPI170
169	PEVTD	RPI169
168	PTG TRIG[27]	RPI168
167	PTG TRIG[26]	RPI167
166-164	Reserved	
163	CMP3	RPI163
162	CMP2	RPI162
161	CMP1	RPI161
160-62	Reserved	
61	RD12	RPI61
60	RD11	RPI60
59	RD10	RPI59
58	RD9	RPI58

.....continued

Index	Source Output Signal	Remap Input #
57	RD8	RPI57
56	RD7	RPI56
55	RD6	RPI55
54	RD5	RPI54
53	RD4	RPI53
52	RD3	RPI52
51	RD2	RPI51
50	RD1	RPI50
49	RD0	RPI49
48-45	Reserved	
44	RC11	RPI44
43	RC10	RPI43
42	RC9	RPI42
41	RC8	RPI41
40	RC7	RPI40
39	RC6	RPI39
38	RC5	RPI38
37	RC4	RPI37
36	RC3	RPI36
35	RC2	RPI35
34	RC1	RPI34
33	RC0	RPI33
32-29	Reserved	
28	RB11	RPI28
27	RB10	RPI27
26	RB9	RPI26
25	RB8	RPI25
24	RB7	RPI24
23	RB6	RPI23
22	RB5	RPI22
21	RB4	RPI21
20	RB3	RPI20
19	RB2	RPI19
18	RB1	RPI18
17	RB0	RPI17
16-13	Reserved	
12	RA11	RPI12
11	RA10	RPI11
10	RA9	RPI10
9	RA8	RPI9
8	RA7	RPI8
7	RA6	RPI7
6	RA5	RPI6
5	RA4	RPI5
4	RA3	RPI4

.....continued

Index	Source Output Signal	Remap Input #
3	RA2	RPI3
2	RA1	RPI2
1	RA0	RPI1
0	tied to 1'b0	—

Note:
1. This list of output signals can be mapped to any of the peripheral inputs listed in Table 11-10.

Table 11-12. Virtual Outputs to Remappable Ouput Registers(1)

Virtual Outputs	Remappable Output Register	Register Bitfield
RPV0	RPOR16	RP65R[6:0]
RPV1	RPOR16	RP66R[6:0]
RPV2	RPOR16	RP67R[6:0]
RPV3	RPOR16	RP68R[6:0]
RPV4	RPOR17	RP69R[6:0]
RPV5	RPOR17	RP70R[6:0]
RPV6	RPOR17	RP71R[6:0]
RPV7	RPOR17	RP72R[6:0]
RPV8	RPOR18	RP73R[6:0]
RPV9	RPOR18	RP74R[6:0]
RPV10	RPOR18	RP75R[6:0]
RPV11	RPOR18	RP76R[6:0]
RPV12	RPOR19	RP77R[6:0]
RPV13	RPOR19	RP78R[6:0]
RPV14	RPOR19	RP79R[6:0]
RPV15	RPOR19	RP80R[6:0]

Note:
1. This list of virtual output signals can be mapped to any of the peripheral inputs listed in Table 11-10.

Table 11-13. Output Selection for Remappable Pins (RPn)

Function	RPnR[5:0]	Output Name
Default PORT	0	RPn tied to Default Pin
PWM1H	1	RPn tied to PWM1H Output
PWM1L	2	RPn tied to PWM1L Output
PWM2H	3	RPn tied to PWM2H Output
PWM2L	4	RPn tied to PWM2L Output
PWM3H	5	RPn tied to PWM3H Output
PWM3L	6	RPn tied to PWM3L Output
PWM4H	7	RPn tied to PWM4H Output
PWM4L	8	RPn tied to PWM4L Output
U1TX	9	RPn tied to UART1 Transmit
U1RTS	10	RPn tied to UART1 Request-to-Send
U2TX	11	RPn tied to UART2 Transmit
U2RTS	12	RPn tied to UART2 Request-to-Send
SDO1	13	RPn tied to SPI1 Data Output
SCK1OUT	14	RPn tied to SPI1 Clock Output
SS1OUT	15	RPn tied to SPI1 Client Select

.....continued		
Function	**RPnR[5:0]**	**Output Name**
SDO2	16	RPn tied to SPI2 Data Output
SCK2OUT	17	RPn tied to SPI2 Clock Output
SS2OUT	18	RPn tied to SPI2 Client Select
SDO3	19	RPn tied to SPI3 Data Output
SCK3OUT	20	RPn tied to SPI3 Clock Output
SS3OUT	21	RPn tied to SPI3 Client Select
REFO1	22	RPn tied to Reference Clock 1 Output
REFO2	23	RPn tied to Reference Clock 2 Output
OCM1	24	RPn tied to CCP1 Output Compare Event
OCM2	25	RPn tied to CCP2 Output Compare Event
OCM3	26	RPn tied to CCP3 Output Compare Event
OCM4	27	RPn tied to CCP4 Enable Output Compare Event
CMP1	32	RPn tied to Comparator 1 Output
CMP2	33	RPn tied to Comparator 2 Output
CMP3	34	RPn tied to Comparator 2 Output
U3TX	36	RPn tied to UART3 Transmit
U3RTS	37	RPn tied to UART Request-to-Send
PEVTA	43	RPn tied to PWM Event A Output
PEVTB	44	RPn tied to PWM Event B output
QEICMP1	45	RPn tied to QEI Comparator 1 Output
CLC1OUT	47	RPn tied to CLC1 Output
CLC2OUT	48	RPn tied to CLC2 Output
PEVTC	51	RPn tied to PWM Event C Output
PEVTD	52	RPn tied to PWM Event D Output
PEVTE	53	RPn tied to PWM Event E Output
PEVTF	54	RPn tied to PWM Event F Output
PTG TRIG 24	55	RPn tied to PTG Trigger 24 Output
PTG TRIG 25	56	RPn tied to PTG Trigger 25 Output
SENT1OUT	57	RPn tied to SENT1 Output
SENT2OUT	58	RPn tied to SENT2 Output
BISS1MO	63	RPn tied to BiSS Output
BISS1MA	64	RPn tied to BiSS CLK
CLC3OUT	65	RPn tied to CLC3 Output
CLC4OUT	66	RPn tied to CLC4 Output
U1DTRn	67	RPn tied to UART1 Data Terminal Ready Output
U2DTRn	68	RPn tied to UART2 Data Terminal Ready Output
U3DTRn	69	RPn tied to UART3 Data Terminal Ready Output

11.2 Architectural Overview

A general purpose I/O port that shares a pin with a peripheral is generally subservient to the peripheral. Once enabled, the peripheral selects whether the peripheral or the associated port has ownership of the I/O pin.

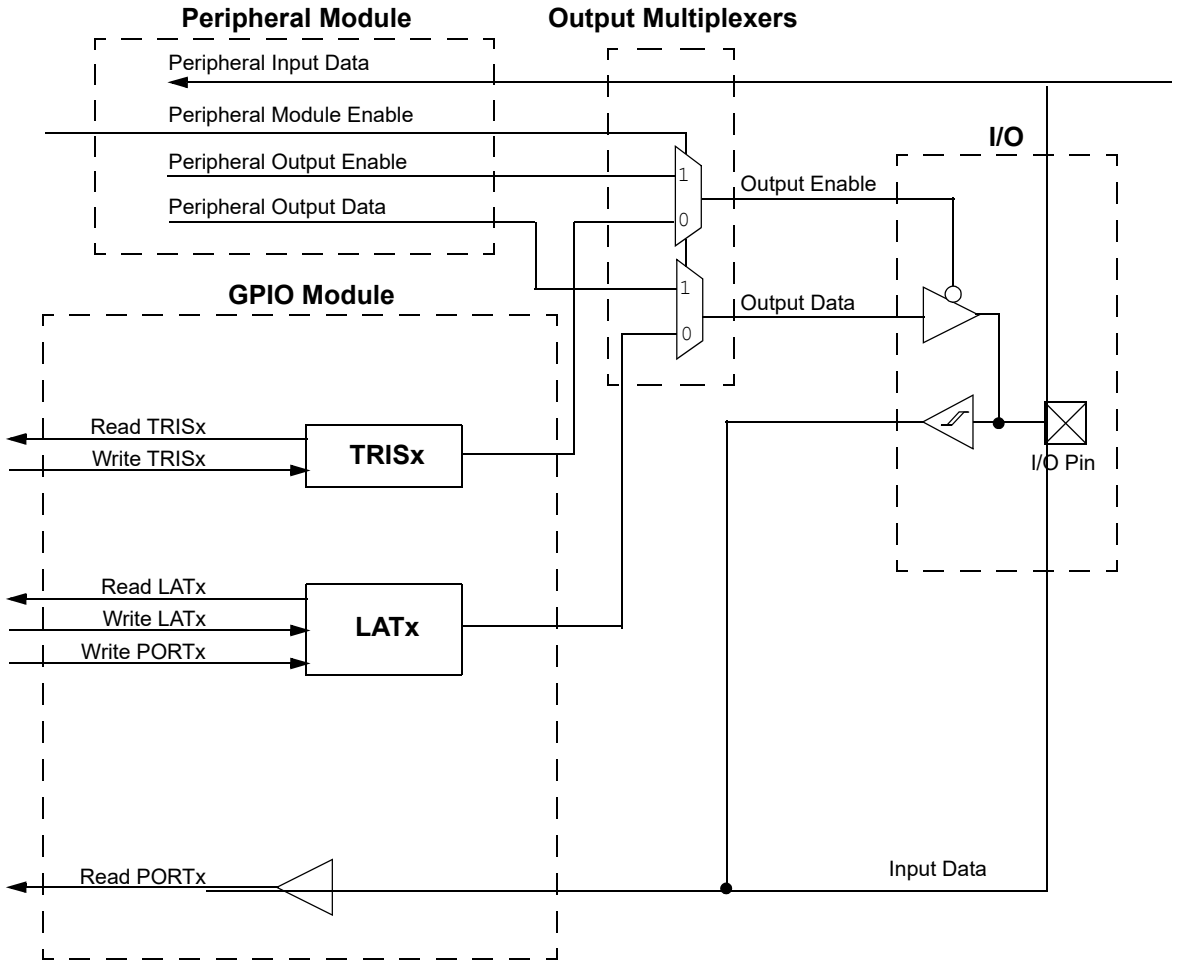
An MUX and its associated logic controls interaction between peripherals and port logic. When a peripheral is enabled but the peripheral is not actively driving a pin, the port is still allowed to drive the pin. This is useful for “loop through”, in which a port’s digital output drives the input of a peripheral that shares the same pin.

When a peripheral is enabled and the peripheral is actively driving an associated pin, the IO MUX disables the use of the pin as a general purpose output. The I/O pin value may be read by the port, but the LATx[n] output value for the port is ignored.

A block diagram of a typical I/O port structure is shown in Figure 11-1.

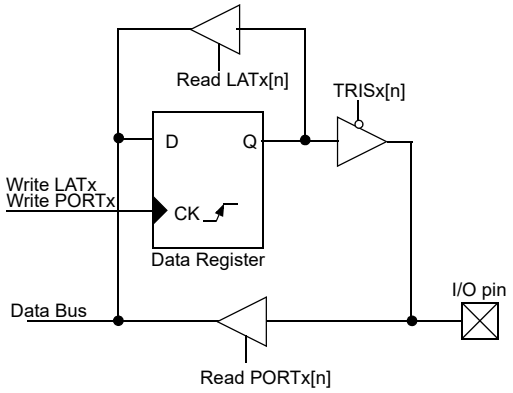
11.2.1 Port Overview

Figure 11-1. Typical Port Structure Block Diagram



11.2.1. Port Overview is a simplified version of the port itself without peripheral multiplexing.

Figure 11-2. Simplified Block Diagram of PORT/LAT/TRIS



11.2.2 Peripheral Pin Select (PPS) Overview

The PPS feature allows remapping of pin and port functions and can be configured to best suit an application. This simplifies board design and allows peripherals to be connected to one another with or without a pin.

Figure 11-3 shows the structure of a remappable input. The control logic is based on the peripheral, through selection of which Remappable Pin (RPn) is used as the input source. Each peripheral is associated with a register bit field to select RPn. More than one peripheral can select the same input pin.

Figure 11-3. Remappable Input for U1RX

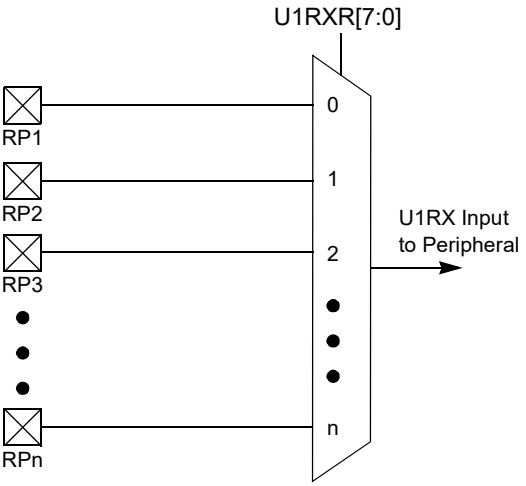
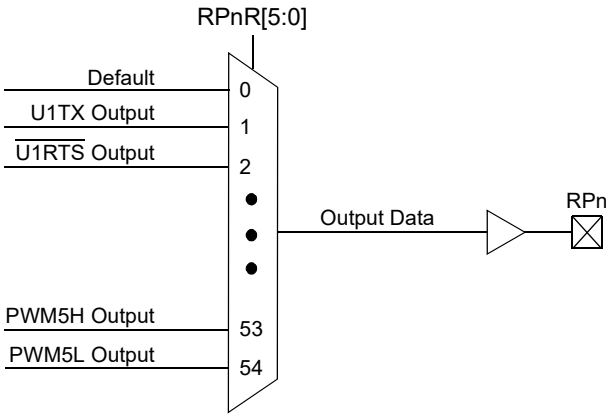


Figure 11-4 shows the structure of a remappable output. Unlike the inputs, the control logic is based on the remappable pin. Each RPn pin has an associated register bit field to select which peripheral output is routed to the pin. The output of one peripheral can be mapped to multiple RPn pins.

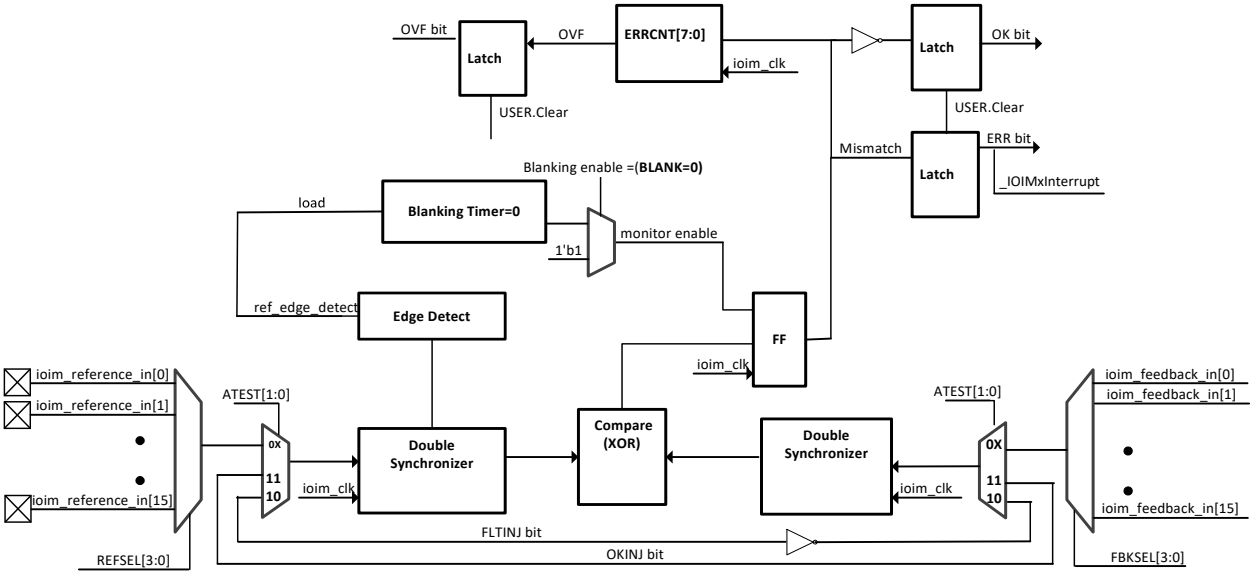
Figure 11-4. Multiplexing of Remappable Outputs for RPn



11.2.3 I/O Integrity Monitor (IOIM) Overview

The port module includes integrity monitoring circuity, as shown in Figure 11-5, to validate I/O functionality in critical applications by comparing a device output signal against a reference signal. If a mismatch is detected, an event is generated to allow software to take action as needed for the application. A programmable blanking timer is included to account for the feedback path delay. The timer is reset on a change of state in the reference signal. A counter is included to keep track of the number of mismatch events.

Figure 11-5. IOIM Block Diagram



11.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0200	PORTA	31:24	ALTPORTA[15:8]							
		23:16	ALTPORTA[7:0]							
		15:8	PORTA[15:8]							
		7:0	PORTA[7:0]							
0x0204	LATA	31:24	ALTLATA[15:8]							
		23:16	ALTLATA[7:0]							
		15:8	LATA[15:8]							
		7:0	LATA[7:0]							
0x0208	TRISA	31:24	ALTTRISA[15:8]							
		23:16	ALTTRISA[7:0]							
		15:8	TRISA[15:8]							
		7:0	TRISA[7:0]							
0x020C	CNSTATA	31:24	CNSTATA[15:8]							
		23:16	CNSTATA[7:0]							
		15:8	CNSTATA[15:8]							
		7:0	CNSTATA[7:0]							
0x0210	CNFA	31:24	ALTCNFA[15:8]							
		23:16	ALTCNFA[7:0]							
		15:8	CNFA[15:8]							
		7:0	CNFA[7:0]							
0x0214	PORTB	31:24	ALTPORTB[15:8]							
		23:16	ALTPORTB[7:0]							
		15:8	PORTB[15:8]							
		7:0	PORTB[7:0]							
0x0218	LATB	31:24	ALTLATB[15:8]							
		23:16	ALTLATB[7:0]							
		15:8	LATB[15:8]							
		7:0	LATB[7:0]							
0x021C	TRISB	31:24	ALTTRISB[15:8]							
		23:16	ALTTRISB[7:0]							
		15:8	TRISB[15:8]							
		7:0	TRISB[7:0]							
0x0220	CNSTATB	31:24	CNSTATB[15:8]							
		23:16	CNSTATB[7:0]							
		15:8	CNSTATB[15:8]							
		7:0	CNSTATB[7:0]							
0x0224	CNFB	31:24	ALTCNFB[15:8]							
		23:16	ALTCNFB[7:0]							
		15:8	CNFB[15:8]							
		7:0	CNFB[7:0]							
0x0228	PORTC	31:24	ALTPORTC[15:8]							
		23:16	ALTPORTC[7:0]							
		15:8	PORTC[15:8]							
		7:0	PORTC[7:0]							
0x022C	LATC	31:24	ALTLATC[15:8]							
		23:16	ALTLATC[7:0]							
		15:8	LATC[15:8]							
		7:0	LATC[7:0]							
0x0230	TRISC	31:24	ALTTRISC[15:8]							
		23:16	ALTTRISC[7:0]							
		15:8	TRISC[15:8]							
		7:0	TRISC[7:0]							
0x0234	CNSTATC	31:24	CNSTATC[15:8]							
		23:16	CNSTATC[7:0]							
		15:8	CNSTATC[15:8]							
		7:0	CNSTATC[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0238	CNFC	31:24	ALTCNFC[15:8]							
		23:16	ALTCNFC[7:0]							
		15:8	CNFC[15:8]							
		7:0	CNFC[7:0]							
0x023C	PORTD	31:24	ALTPORTD[15:8]							
		23:16	ALTPORTD[7:0]							
		15:8	PORTD[15:8]							
		7:0	PORTD[7:0]							
0x0240	LATD	31:24	ALTLATD[15:8]							
		23:16	ALTLATD[7:0]							
		15:8	LATD[15:8]							
		7:0	LATD[7:0]							
0x0244	TRISD	31:24	ALTTRISD[15:8]							
		23:16	ALTTRISD[7:0]							
		15:8	TRISD[15:8]							
		7:0	TRISD[7:0]							
0x0248	CNSTATD	31:24	CNSTATD[15:8]							
		23:16	CNSTATD[7:0]							
		15:8	CNSTATD[15:8]							
		7:0	CNSTATD[7:0]							
0x024C	CNFD	31:24	ALTCNFD[15:8]							
		23:16	ALTCNFD[7:0]							
		15:8	CNFD[15:8]							
		7:0	CNFD[7:0]							
0x0250 ... 0x1E8F	Reserved									
0x1E90	IOIM1CON	31:24					FLTINJ	OKINJ	ATEST[1:0]	
		23:16	EOVFV[7:0]							
		15:8	ON		SLPEN	SIDL		EXTCLK		
		7:0	FBKSEL[3:0]				REFSEL[3:0]			
0x1E94	IOIM1BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1E98	IOIM1STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1E9C	IOIM2CON	31:24					FLTINJ	OKINJ	ATEST[1:0]	
		23:16	EOVFV[7:0]							
		15:8	ON		SLPEN	SIDL		EXTCLK		
		7:0	FBKSEL[3:0]				REFSEL[3:0]			
0x1EA0	IOIM2BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1EA4	IOIM2STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1EA8	IOIM3CON	31:24					FLTINJ	OKINJ	ATEST[1:0]	
		23:16	EOVFV[7:0]							
		15:8	ON		SLPEN	SIDL		EXTCLK		
		7:0	FBKSEL[3:0]				REFSEL[3:0]			
0x1EAC	IOIM3BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1EB0	IOIM3STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1EB4	IOIM4CON	31:24					FLTINJ	OKINJ	ATEST[1:0]	
		23:16	EOVFV[7:0]							
		15:8	ON		SLPEN	SIDL		EXTCLK		
		7:0	FBKSEL[3:0]				REFSEL[3:0]			
0x1EB8	IOIM4BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1EBC	IOIM4STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1EC0 ... 0x1EC3	Reserved									
0x1EC4	IOIM5BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1EC8	IOIM5STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1ECC ... 0x1ECF	Reserved									
0x1ED0	IOIM6BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1ED4	IOIM6STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1ED8 ... 0x1EDB	Reserved									
0x1EDC	IOIM7BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1EE0	IOIM7STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1EE4 ... 0x1EE7	Reserved									
0x1EE8	IOIM8BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1EEC	IOIM8STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1EF0 ... 0x1EF3	Reserved									
0x1EF4	IOIM9BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1EF8	IOIM9STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1EFC ... 0x1EFF	Reserved									
0x1F00	IOIM10BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1F04	IOIM10STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1F08 ... 0x1F0B	Reserved									
0x1F0C	IOIM11BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1F10	IOIM11STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1F14 ... 0x1F17	Reserved									
0x1F18	IOIM12BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1F1C	IOIM12STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1F20 ... 0x1F23	Reserved									
0x1F24	IOIM13BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1F28	IOIM13STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1F2C ... 0x1F2F	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1F30	IOIM14BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1F34	IOIM14STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1F38 ... 0x1F3B	Reserved									
0x1F3C	IOIM15BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1F40	IOIM15STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1F44 ... 0x1F47	Reserved									
0x1F48	IOIM16BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1F4C	IOIM16STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1F50 ... 0x363F	Reserved									
0x3640	ANSELA	31:24	ALTANSELA[15:8]							
		23:16	ALTANSELA[7:0]							
		15:8	ANSELA[15:8]							
		7:0	ANSELA[7:0]							
0x3644	ODCA	31:24	ALTODCA[15:8]							
		23:16	ALTODCA[7:0]							
		15:8	ODCA[15:8]							
		7:0	ODCA[7:0]							
0x3648	CNPUA	31:24	ALTCNPUA[15:8]							
		23:16	ALTCNPUA[7:0]							
		15:8	CNPUA[15:8]							
		7:0	CNPUA[7:0]							
0x364C	CNPDA	31:24	ALTCNPDA[15:8]							
		23:16	ALTCNPDA[7:0]							
		15:8	CNPDA[15:8]							
		7:0	CNPDA[7:0]							
0x3650	CNCONA	31:24								
		23:16								
		15:8	ON				CNSTYLE	PORT32		
		7:0								
0x3654	CNEN0A	31:24	ALTCNEN0A[15:8]							
		23:16	ALTCNEN0A[7:0]							
		15:8	CNEN0A[15:8]							
		7:0	CNEN0A[7:0]							
0x3658	CNEN1A	31:24	ALTCNEN1A[15:8]							
		23:16	ALTCNEN1A[7:0]							
		15:8	CNEN1A[15:8]							
		7:0	CNEN1A[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x365C ... 0x3663	Reserved									
0x3664	ANSELB	31:24	ALTANSELB[15:8]							
		23:16	ALTANSELB[7:0]							
		15:8	ANSELB[15:8]							
		7:0	ANSELB[7:0]							
0x3668	ODCB	31:24	ALTODCB[15:8]							
		23:16	ALTODCB[7:0]							
		15:8	ODCB[15:8]							
		7:0	ODCB[7:0]							
0x366C	CNPUB	31:24	ALTCNPUB[15:8]							
		23:16	ALTCNPUB[7:0]							
		15:8	CNPUB[15:8]							
		7:0	CNPUB[7:0]							
0x3670	CNPDB	31:24	ALTCNPDB[15:8]							
		23:16	ALTCNPDB[7:0]							
		15:8	CNPDB[15:8]							
		7:0	CNPDB[7:0]							
0x3674	CNCONB	31:24								
		23:16								
		15:8	ON				CNSTYLE	PORT32		
		7:0								
0x3678	CNEN0B	31:24	ALTCNEN0B[15:8]							
		23:16	ALTCNEN0B[7:0]							
		15:8	CNEN0B[15:8]							
		7:0	CNEN0B[7:0]							
0x367C	CNEN1B	31:24	ALTCNEN1B[15:8]							
		23:16	ALTCNEN1B[7:0]							
		15:8	CNEN1B[15:8]							
		7:0	CNEN1B[7:0]							
0x3680 ... 0x368B	Reserved									
0x368C	ODCC	31:24	ALTODCC[15:8]							
		23:16	ALTODCC[7:0]							
		15:8	ODCC[15:8]							
		7:0	ODCC[7:0]							
0x3690	CNPUC	31:24	ALTCNPUC[15:8]							
		23:16	ALTCNPUC[7:0]							
		15:8	CNPUC[15:8]							
		7:0	CNPUC[7:0]							
0x3694	CNPDC	31:24	ALTCNPDC[15:8]							
		23:16	ALTCNPDC[7:0]							
		15:8	CNPDC[15:8]							
		7:0	CNPDC[7:0]							
0x3698	CNCONC	31:24								
		23:16								
		15:8	ON				CNSTYLE	PORT32		
		7:0								
0x369C	CNEN0C	31:24	ALTCNEN0C[15:8]							
		23:16	ALTCNEN0C[7:0]							
		15:8	CNEN0C[15:8]							
		7:0	CNEN0C[7:0]							
0x36A0	CNEN1C	31:24	ALTCNEN1C[15:8]							
		23:16	ALTCNEN1C[7:0]							
		15:8	CNEN1C[15:8]							
		7:0	CNEN1C[7:0]							
0x36A4 ... 0x36AF	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x36B0	ODCD	31:24	ALTODCD[15:8]							
		23:16	ALTODCD[7:0]							
		15:8	ODCD[15:8]							
		7:0	ODCD[7:0]							
0x36B4	CNPUD	31:24	ALTCNPUD[15:8]							
		23:16	ALTCNPUD[7:0]							
		15:8	CNPUD[15:8]							
		7:0	CNPUD[7:0]							
0x36B8	CNPDD	31:24	ALTCNPDD[15:8]							
		23:16	ALTCNPDD[7:0]							
		15:8	CNPDD[15:8]							
		7:0	CNPDD[7:0]							
0x36BC	CNCOND	31:24								
		23:16								
		15:8	ON				CNSTYLE	PORT32		
		7:0								
0x36C0	CNEN0D	31:24	ALTCNEN0D[15:8]							
		23:16	ALTCNEN0D[7:0]							
		15:8	CNEN0D[15:8]							
		7:0	CNEN0D[7:0]							
0x36C4	CNEN1D	31:24	ALTCNEN1D[15:8]							
		23:16	ALTCNEN1D[7:0]							
		15:8	CNEN1D[15:8]							
		7:0	CNEN1D[7:0]							
0x36C8 ... 0x38FF	Reserved									
0x3900	RPCON	31:24								
		23:16								
		15:8					IOLOCK			
		7:0								
0x3904	RPINR0	31:24	INT3R[7:0]							
		23:16	INT2R[7:0]							
		15:8	INT1R[7:0]							
		7:0								
0x3908	RPINR1	31:24	REFI2R[7:0]							
		23:16	REFI1R[7:0]							
		15:8	T1CKR[7:0]							
		7:0	INT4R[7:0]							
0x390C	RPINR2	31:24	ICM4R[7:0]							
		23:16	ICM3R[7:0]							
		15:8	ICM2R[7:0]							
		7:0	ICM1R[7:0]							
0x3910 ... 0x3917	Reserved									
0x3918	RPINR5	31:24	OCFDR[7:0]							
		23:16	OCFCR[7:0]							
		15:8	OCFB[7:0]							
		7:0	OCFA[7:0]							
0x391C	RPINR6	31:24	PCI11R[7:0]							
		23:16	PCI10R[7:0]							
		15:8	PCI9R[7:0]							
		7:0	PCI8R[7:0]							
0x3920	RPINR7	31:24	QEIHOME1R[7:0]							
		23:16	QEIINDX1R[7:0]							
		15:8	QEIB1R[7:0]							
		7:0	QEIA1R[7:0]							
0x3924 ... 0x3927	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3928	RPINR9	31:24	U2CTSR[7:0]							
		23:16	U2RXR[7:0]							
		15:8	U1CTSR[7:0]							
		7:0	U1RXR[7:0]							
0x392C	RPINR10	31:24								
		23:16	SS1R[7:0]							
		15:8	SCK1R[7:0]							
		7:0	SDI1R[7:0]							
0x3930	RPINR11	31:24								
		23:16	SS2R[7:0]							
		15:8	SCK2R[7:0]							
		7:0	SDI2R[7:0]							
0x3934 ... 0x3937	Reserved									
0x3938	RPINR13	31:24								
		23:16								
		15:8	U3CTSR[7:0]							
		7:0	U3RXR[7:0]							
0x393C	RPINR14	31:24	SENT2R[7:0]							
		23:16	SENT1R[7:0]							
		15:8								
		7:0								
0x3940	RPINR15	31:24								
		23:16	SS3R[7:0]							
		15:8	SCK3R[7:0]							
		7:0	SDI3R[7:0]							
0x3944 ... 0x3947	Reserved									
0x3948	RPINR17	31:24	PCI15R[7:0]							
		23:16	PCI14R[7:0]							
		15:8	PCI13R[7:0]							
		7:0	PCI12R[7:0]							
0x394C	RPINR18	31:24	ADTRIG31R[7:0]							
		23:16	PCI18R[7:0]							
		15:8	PCI17R[7:0]							
		7:0	PCI16R[7:0]							
0x3950	RPINR19	31:24								
		23:16								
		15:8	BISSGL1R[7:0]							
		7:0	BISSSL1R[7:0]							
0x3954	RPINR20	31:24	CLCIND[7:0]							
		23:16	CLCINC[7:0]							
		15:8	CLCINB[7:0]							
		7:0	CLCINA[7:0]							
0x3958	RPINR21	31:24								
		23:16	U3DCDR[7:0]							
		15:8	U2DCDR[7:0]							
		7:0	U1DCDR[7:0]							
0x395C ... 0x397F	Reserved									
0x3980	RPOR0	31:24		RP4R[6:0]						
		23:16		RP3R[6:0]						
		15:8		RP2R[6:0]						
		7:0		RP1R[6:0]						
0x3984	RPOR1	31:24		RP8R[6:0]						
		23:16		RP7R[6:0]						
		15:8		RP6R[6:0]						
		7:0		RP5R[6:0]						

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3988	RPOR2	31:24					RP12R[6:0]			
		23:16					RP11R[6:0]			
		15:8					RP10R[6:0]			
		7:0					RP9R[6:0]			
0x398C ... 0x398F	Reserved									
0x3990	RPOR4	31:24					RP20R[6:0]			
		23:16					RP19R[6:0]			
		15:8					RP18R[6:0]			
		7:0					RP17R[6:0]			
0x3994	RPOR5	31:24					RP24R[6:0]			
		23:16					RP23R[6:0]			
		15:8					RP22R[6:0]			
		7:0					RP21R[6:0]			
0x3998	RPOR6	31:24					RP28R[6:0]			
		23:16					RP27R[6:0]			
		15:8					RP26R[6:0]			
		7:0					RP25R[6:0]			
0x399C ... 0x399F	Reserved									
0x39A0	RPOR8	31:24					RP36R[6:0]			
		23:16					RP35R[6:0]			
		15:8					RP34R[6:0]			
		7:0					RP33R[6:0]			
0x39A4	RPOR9	31:24					RP40R[6:0]			
		23:16					RP39R[6:0]			
		15:8					RP38R[6:0]			
		7:0					RP37R[6:0]			
0x39A8	RPOR10	31:24					RP44R[6:0]			
		23:16					RP43R[6:0]			
		15:8					RP42R[6:0]			
		7:0					RP41R[6:0]			
0x39AC ... 0x39AF	Reserved									
0x39B0	RPOR12	31:24					RP52R[6:0]			
		23:16					RP51R[6:0]			
		15:8					RP50R[6:0]			
		7:0					RP49R[6:0]			
0x39B4	RPOR13	31:24					RP56R[6:0]			
		23:16					RP55R[6:0]			
		15:8					RP54R[6:0]			
		7:0					RP53R[6:0]			
0x39B8	RPOR14	31:24					RP60R[6:0]			
		23:16					RP59R[6:0]			
		15:8					RP58R[6:0]			
		7:0					RP57R[6:0]			
0x39BC	RPOR15	31:24								
		23:16								
		15:8								
		7:0					RP61R[6:0]			
0x39C0	RPOR16	31:24					RP68R[6:0]			
		23:16					RP67R[6:0]			
		15:8					RP66R[6:0]			
		7:0					RP65R[6:0]			
0x39C4	RPOR17	31:24					RP72R[6:0]			
		23:16					RP71R[6:0]			
		15:8					RP70R[6:0]			
		7:0					RP69R[6:0]			

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x39C8	RPOR18	31:24					RP76R[6:0]			
		23:16					RP75R[6:0]			
		15:8					RP74R[6:0]			
		7:0					RP73R[6:0]			
0x39CC	RPOR19	31:24					RP80R[6:0]			
		23:16					RP79R[6:0]			
		15:8					RP78R[6:0]			
		7:0					RP76R[6:0]			

11.3.1 Input Data for PORTx Register

Name: PORTx
Offset: 0x0200, 0x0214, 0x0228, 0x023C

Note: SFR bit availability is defined in Table 11-1 through Table 11-4 for each device variant and port, respectively.

Bit	31	30	29	28	27	26	25	24
	ALTPORTx[15:8]							
Access	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ALTPORTx[7:0]							
Access	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PORTx[15:8]							
Access	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PORTx[7:0]							
Access	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC
Reset	0	0	0	0	0	0	0	x

Bits 31:16 – ALTPORTx[15:0] Alternate PORTx Data Input Value bits

Bits 15:0 – PORTx[15:0] PORTx Data Input Value bits

11.3.2 Output Data for LATx Register

Name: LATx
Offset: 0x0204, 0x0218, 0x022C, 0x0240

Note: See Table 11-1 through Table 11-4 for bit availability for a given device variant.

Bit	31	30	29	28	27	26	25	24
	ALTLATx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ALTLATx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LATx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LATx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	x

Bits 31:16 – ALTLATx[15:0] PORTx Data Output Value bits

Bits 15:0 – LATx[15:0] PORTx Data Output Value bits

11.3.3 Output Enable for TRISx Register

Name: TRISx
Offset: 0x0208, 0x021C, 0x0230, 0x0244

Note: See Table 11-1 through Table 11-4 for bit availability for a given device variant.

Bit	31	30	29	28	27	26	25	24
	ALTTRISx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ALTTRISx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TRISx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TRISx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 31:16 – ALTTRISx[15:0] Output Enable for PORTx bits

Value	Description
1	LATx[n] is not driven on the PORTx[n] pin
0	LATx[n] is driven on the PORTx[n] pin

Bits 15:0 – TRISx[15:0] Output Enable for PORTx bits

Value	Description
1	LATx[n] is not driven on the PORTx[n] pin
0	LATx[n] is driven on the PORTx[n] pin

11.3.4 Interrupt Change Notification Status for CNSTATx Register

Name: CNSTATx
Offset: 0x020C, 0x0220, 0x0234, 0x0248

Note: See Table 11-1 through Table 11-4 for bit availability for a given device variant.

Bit	31	30	29	28	27	26	25	24
	CNSTATx[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CNSTATx[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNSTATx[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNSTATx[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CNSTATx[15:0] Interrupt Change Notification Status for PORTx bits
When CNSTYLE (CNCONx[11]) = 0:

Value	Description
1	Change occurred on PORTx[n] since last read of PORTx[n]
0	Change did not occur on PORTx[n] since last read of PORTx[n]

Bits 15:0 – CNSTATx[15:0] Interrupt Change Notification Status for PORTx bits
When CNSTYLE (CNCONx[11]) = 0:

Value	Description
1	Change occurred on PORTx[n] since last read of PORTx[n]
0	Change did not occur on PORTx[n] since last read of PORTx[n]

11.3.5 Interrupt Change Notification Flag for CNFx Register

Name: CNFx
Offset: 0x0210, 0x0224, 0x0238, 0x024C

Note: See Table 11-1 through Table 11-4 for bit availability for a given device variant.

Bit	31	30	29	28	27	26	25	24
	ALTCNFx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ALTCNFx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNFx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNFx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – ALTCNFx[15:0] Interrupt Change Notification Flag for PORTx bits
When CNSTYLE (CNCONx[11]) = 1:

Value	Description
1	An enabled edge event occurred on the PORTx[n] pin
0	An enabled edge event did not occur on the PORTx[n] pin

Bits 15:0 – CNFx[15:0] Interrupt Change Notification Flag for PORTx bits
When CNSTYLE (CNCONx[11]) = 1:

Value	Description
1	An enabled edge event occurred on the PORTx[n] pin
0	An enabled edge event did not occur on the PORTx[n] pin

11.3.6 Analog Select for ANSELx Register

Name: ANSELx
Offset: 0x3640, 0x3664

Note: See Table 11-1 through Table 11-8 for bit availability for a given device variant.

Bit	31	30	29	28	27	26	25	24
	ALTANSELx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ALTANSELx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ANSELx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ANSELx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 31:16 – ALTANSELx[15:0] Analog Select for PORTx bits

Value	Description
1	Analog input is enabled and digital input is disabled on the PORTx[n] pin
0	Analog input is disabled and digital input is enabled on the PORTx[n] pin

Bits 15:0 – ANSELx[15:0] Analog Select for PORTx bits

Value	Description
1	Analog input is enabled and digital input is disabled on the PORTx[n] pin
0	Analog input is disabled and digital input is enabled on the PORTx[n] pin

11.3.7 Open-Drain Enable for ODCx Register

Name: ODCx
Offset: 0x3644, 0x3668, 0x368C, 0x36B0

Note: See Table 11-1 through Table 11-4 for bit availability for a given device variant.

Bit	31	30	29	28	27	26	25	24
	ALTODCx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ALTODCx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ODCx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ODCx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – ALTODCx[15:0] PORTx Open-Drain Enable bits

Value	Description
1	Open-drain is enabled on the PORTx pin
0	Open-drain is disabled on the PORTx pin

Bits 15:0 – ODCx[15:0] PORTx Open-Drain Enable bits

Value	Description
1	Open-drain is enabled on the PORTx pin
0	Open-drain is disabled on the PORTx pin

11.3.8 Change Notification Pull-up Enable for CNPUx Register

Name: CNPUx
Offset: 0x3648, 0x366C, 0x3690, 0x36B4

Note: See Table 11-1 through Table 11-4 for bit availability for a given device variant.

Bit	31	30	29	28	27	26	25	24
	ALTCNPUx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ALTCNPUx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNPUx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNPUx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – ALTCNPUx[15:0] Change Notification Pull-up Enable for PORTx bits

Value	Description
1	The pull-up for PORTx[n] is enabled – takes precedence over the pull-down selection
0	The pull-up for PORTx[n] is disabled

Bits 15:0 – CNPUx[15:0] Change Notification Pull-up Enable for PORTx bits

Value	Description
1	The pull-up for PORTx[n] is enabled – takes precedence over the pull-down selection
0	The pull-up for PORTx[n] is disabled

11.3.9 Change Notification Pull-Down Enable for CNPDx Register

Name: CNPDx
Offset: 0x364C, 0x3670, 0x3694, 0x36B8

Note: See Table 11-1 through Table 11-4 for bit availability for a given device variant.

Bit	31	30	29	28	27	26	25	24
	ALTCNPDx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ALTCNPDx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNPDx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNPDx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – ALTCNPDx[15:0] Change Notification Pull-Down Enable for PORTx bits

Value	Description
1	The pull-down for PORTx[n] is enabled (if the pull-up for PORTx[n] is not enabled)
0	The pull-down for PORTx[n] is disabled

Bits 15:0 – CNPDx[15:0] Change Notification Pull-Down Enable for PORTx bits

Value	Description
1	The pull-down for PORTx[n] is enabled (if the pull-up for PORTx[n] is not enabled)
0	The pull-down for PORTx[n] is disabled

11.3.10 Change Notification Control for CNCONx Register

Name: CNCONx
Offset: 0x3650, 0x3674, 0x3698, 0x36BC

Note: See Table 11-1 through Table 11-4 for bit availability for a given device variant.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON				CNSTYLE	PORT32		
Access	R/W				R/W	R/W		
Reset	0				0	0		

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 15 – ON Change Notification (CN) Control for PORTx On bit

Value	Description
1	CN is enabled
0	CN is disabled

Bit 11 – CNSTYLE Change Notification Style Selection bit

Value	Description
1	Edge style (detects edge transitions, CNFx[15:0] bits are used for a Change Notification event)
0	Mismatch style (detects change from last port read, CNSTATx[15:0] bits are used for a Change Notification event)

Bit 10 – PORT32 Selects between 16-bit and 32-bit control of Port SFRs

Value	Description
1	32-bit Access selected
0	16-bit Access selected

11.3.11 Interrupt Change Notification Enable for CNEN0x Register

Name: CNEN0x
Offset: 0x3654, 0x3678, 0x369C, 0x36C0

Note: See Table 11-1 through Table 11-4 for bit availability for a given device variant.

Bit	31	30	29	28	27	26	25	24
	ALTCNEN0x[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ALTCNEN0x[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNEN0x[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNEN0x[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – ALTCNEN0x[15:0] Interrupt Change Notification Enable for PORTx bits

Value	Description
1	Interrupt-on-change (from the last read value) is enabled for PORTx[n]
0	Interrupt-on-change is disabled for PORTx[n]

Bits 15:0 – CNEN0x[15:0] Interrupt Change Notification Enable for PORTx bits

Value	Description
1	Interrupt-on-change (from the last read value) is enabled for PORTx[n]
0	Interrupt-on-change is disabled for PORTx[n]

11.3.12 Interrupt Change Notification Edge Select for CNEN1x Register

Name: CNEN1x
Offset: 0x3658, 0x367C, 0x36A0, 0x36C4

Note: See Table 11-1 through Table 11-4 for bit availability for a given device variant.

Bit	31	30	29	28	27	26	25	24
	ALTCNEN1x[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ALTCNEN1x[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNEN1x[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNEN1x[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – ALTCNEN1x[15:0] Interrupt Change Notification Edge Select for PORTx bits

Bits 15:0 – CNEN1x[15:0] Interrupt Change Notification Edge Select for PORTx bits

11.3.13 Peripheral Remapping Configuration Register(1)

Name: RPCON
Offset: 0x3900

Note:

1. Writing to this register needs an unlock sequence.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					IOLOCK			
Access					R/W			
Reset					0			

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 11 – IOLOCK Peripheral Remapping Register Lock bit

Value	Description
1	All Peripheral Remapping registers are locked and cannot be written
0	All Peripheral Remapping registers are unlocked and can be written

11.3.14 Peripheral Pin Select Input Register 0

Name: RPINR0
Offset: 0x3904

Bit	31	30	29	28	27	26	25	24
	INT3R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INT2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INT1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 31:24 – INT3R[7:0] Assign External Interrupt 3 (INT3) to the Corresponding RPn Pin bits

Bits 23:16 – INT2R[7:0] Assign External Interrupt 2 (INT2) to the Corresponding RPn Pin bits

Bits 15:8 – INT1R[7:0] Assign External Interrupt 1 (INT1) to the Corresponding RPn Pin bits

11.3.15 Peripheral Pin Select Input Register 1

Name: RPINR1
Offset: 0x3908

Bit	31	30	29	28	27	26	25	24
	REFI2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	REFI1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	T1CKR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INT4R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – REFI2R[7:0] Reference clock input 2

Bits 23:16 – REFI1R[7:0] Reference clock input 1

Bits 15:8 – T1CKR[7:0] Assign Timer1 External Clock (T1CK) to the Corresponding RPn Pin bits

Bits 7:0 – INT4R[7:0] Assign External Interrupt 4 (INT4) to the Corresponding RPn Pin bits

11.3.16 Peripheral Pin Select Input Register 2

Name: RPINR2
Offset: 0x390C

Bit	31	30	29	28	27	26	25	24
	ICM4R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ICM3R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ICM2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ICM1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – ICM4R[7:0] Assign SCCP Capture 4 (ICM4) Input to the Corresponding RPn Pin bits

Bits 23:16 – ICM3R[7:0] Assign SCCP Capture 3 (ICM3) Input to the Corresponding RPn Pin bits

Bits 15:8 – ICM2R[7:0] Assign SCCP Capture 2 (ICM2) Input to the Corresponding RPn Pin bits

Bits 7:0 – ICM1R[7:0] Assign SCCP Capture 1 (ICM1) Input to the Corresponding RPn Pin bits

11.3.17 Peripheral Pin Select Input Register 5

Name: RPINR5
Offset: 0x3918

Bit	31	30	29	28	27	26	25	24
	OCFDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	OCFCR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	OCFB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	OCFA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – OCFDR[7:0] Assign SCCP Fault D (OCFD) Input to the Corresponding RPn Pin bits

Bits 23:16 – OCFCR[7:0] Assign SCCP Fault C (OCFC) Input to the Corresponding RPn Pin bits

Bits 15:8 – OCFB[7:0] Assign SCCP Fault B (OCFB) Input to the Corresponding RPn Pin bits

Bits 7:0 – OCFA[7:0] Assign SCCP Fault A (OCFA) Input to the Corresponding RPn Pin bits

11.3.18 Peripheral Pin Select Input Register 6

Name: RPINR6
Offset: 0x391C

Bit	31	30	29	28	27	26	25	24
	PCI11R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PCI10R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PCI9R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PCI8R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – PCI11R[7:0] Assign PWM Input 11 (PCI11) to the Corresponding RPn Pin bits

Bits 23:16 – PCI10R[7:0] Assign PWM Input 10 (PCI10) to the Corresponding RPn Pin bits

Bits 15:8 – PCI9R[7:0] Assign PWM Input 9 (PCI9) to the Corresponding RPn Pin bits

Bits 7:0 – PCI8R[7:0] Assign PWM Input 8 (PCI8) to the Corresponding RPn Pin bits

11.3.19 Peripheral Pin Select Input Register 7

Name: RPINR7
Offset: 0x3920

Bit	31	30	29	28	27	26	25	24
	QEIHOME1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	QEIINDX1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	QEIB1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	QEIA1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – QEIHOME1R[7:0]

Bits 23:16 – QEIINDX1R[7:0]

Bits 15:8 – QEIB1R[7:0] Assign QEI Input B (QEIB1) to the Corresponding RPn Pin bits

Bits 7:0 – QEIA1R[7:0] Assign QEI Input A (QEIA1) to the Corresponding RPn Pin bits

11.3.20 Peripheral Pin Select Input Register 9

Name: RPINR9
Offset: 0x3928

Bit	31	30	29	28	27	26	25	24
	U2CTSR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	U2RXR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	U1CTSR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	U1RXR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – U2CTSR[7:0] Assign UART2 Clear-to-Send ($\overline{U2CTS}$) to the Corresponding RPn Pin bits

Bits 23:16 – U2RXR[7:0] Assign UART2 Receive (U2RX) to the Corresponding RPn Pin bit

Bits 15:8 – U1CTSR[7:0] Assign UART1 Clear-to-Send ($\overline{U1CTS}$) to the Corresponding RPn Pin bits

Bits 7:0 – U1RXR[7:0] Assign UART1 Receive (U1RX) to the Corresponding RPn Pin bit

11.3.21 Peripheral Pin Select Input Register 10

Name: RPINR10
Offset: 0x392C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	SS1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SCK1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SDI1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – SS1R[7:0] Assign SPI1 Client Select ($\overline{SS1}$) to the Corresponding RPn Pin bits

Bits 15:8 – SCK1R[7:0] Assign SPI1 Clock Input (SCK1IN) to the Corresponding RPn Pin bits

Bits 7:0 – SDI1R[7:0] Assign SPI1 Data Input (SDI1) to the Corresponding RPn Pin bits

11.3.22 Peripheral Pin Select Input Register 11

Name: RPINR11
Offset: 0x3930

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	SS2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SCK2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SDI2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – SS2R[7:0] Assign SPI1 Client Select ($\overline{SS2}$) to the Corresponding RPn Pin bits

Bits 15:8 – SCK2R[7:0] Assign SPI2 Clock Input (SCK2IN) to the Corresponding RPn Pin bits

Bits 7:0 – SDI2R[7:0] Assign SPI1 Data Input (SDI2) to the Corresponding RPn Pin bits

11.3.23 Peripheral Pin Select Input Register 13

Name: RPINR13
Offset: 0x3938

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	U3CTSR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	U3RXR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – U3CTSR[7:0] Assign UART2 Clear-to-Send (U2CTS) to the Corresponding RPn Pin bits

Bits 7:0 – U3RXR[7:0] Assign UART3 Receive (U3RX) to the Corresponding RPn Pin bit

11.3.24 Peripheral Pin Select Input Register 14

Name: RPINR14
Offset: 0x393C

Bit	31	30	29	28	27	26	25	24
	SENT2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SENT1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 31:24 – SENT2R[7:0] Assign SENT2 Input (SENT2) to the Corresponding RPn Pin bits

Bits 23:16 – SENT1R[7:0] Assign SENT1 Input (SENT1) to the Corresponding RPn Pin bits

11.3.25 Peripheral Pin Select Input Register 15

Name: RPINR15
Offset: 0x3940

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	SS3R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SCK3R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SDI3R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – SS3R[7:0] Assign SPI3 Client Select ($\overline{SS3}$) to the Corresponding RPn Pin bits

Bits 15:8 – SCK3R[7:0] Assign SPI3 Clock Input (SCK3IN) to the Corresponding RPn Pin bits

Bits 7:0 – SDI3R[7:0] Assign SPI3 Data Input (SDI3) to the Corresponding RPn Pin bits

11.3.26 Peripheral Pin Select Input Register 17

Name: RPINR17
Offset: 0x3948

Bit	31	30	29	28	27	26	25	24
	PCI15R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PCI14R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PCI13R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PCI12R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – PCI15R[7:0] Assign PWM Input 15 (PCI15) to the Corresponding RPn Pin bits

Bits 23:16 – PCI14R[7:0] Assign PWM Input 14 (PCI14) to the Corresponding RPn Pin bits

Bits 15:8 – PCI13R[7:0] Assign PWM Input 13 (PCI13) to the Corresponding RPn Pin bits

Bits 7:0 – PCI12R[7:0] Assign PWM Input 12 (PCI12) to the Corresponding RPn Pin bits

11.3.27 Peripheral Pin Select Input Register 18

Name: RPINR18
Offset: 0x394C

Bit	31	30	29	28	27	26	25	24
	ADTRIG31R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PCI18R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PCI17R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PCI16R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – ADTRIG31R[7:0]

Bits 23:16 – PCI18R[7:0] Assign PWM Input 18 (PCI18) to the Corresponding RPn Pin bits

Bits 15:8 – PCI17R[7:0] Assign PWM Input 17 (PCI17) to the Corresponding RPn Pin bits

Bits 7:0 – PCI16R[7:0] Assign PWM Input 16 (PCI16) to the Corresponding RPn Pin bits

11.3.28 Peripheral Pin Select Input Register 19

Name: RPINR19
Offset: 0x3950

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BISSGL1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BISSSL1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BISSGL1R[7:0]

Bits 7:0 – BISSSL1R[7:0]

11.3.29 Peripheral Pin Select Input Register 20

Name: RPINR20
Offset: 0x3954

Bit	31	30	29	28	27	26	25	24
	CLCIND[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CLCINC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CLCINB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLCINA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – CLCIND[7:0]

Bits 23:16 – CLCINC[7:0]

Bits 15:8 – CLCINB[7:0]

Bits 7:0 – CLCINA[7:0]

11.3.30 Peripheral Pin Select Input Register 21

Name: RPINR21
Offset: 0x3958

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	U3DCDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	U2DCDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	U1DCDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – U3DCDR[7:0] UARTx Data Carrier Detect

Bits 15:8 – U2DCDR[7:0] UARTx Data Carrier Detect

Bits 7:0 – U1DCDR[7:0] UARTx Data Carrier Detect

11.3.31 Peripheral Pin Select Output Register 0

Name: RPOR0
Offset: 0x3980

Bit	31	30	29	28	27	26	25	24
		RP4R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP3R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP2R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP1R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP4R[6:0] Peripheral Output Function is Assigned to RP4 Output Pin bits

Bits 22:16 – RP3R[6:0] Peripheral Output Function is Assigned to RP3 Output Pin bits

Bits 14:8 – RP2R[6:0] Peripheral Output Function is Assigned to RP2 Output Pin bits

Bits 6:0 – RP1R[6:0] Peripheral Output Function is Assigned to RP1 Output Pin bits

11.3.32 Peripheral Pin Select Output Register 1

Name: RPOR1
Offset: 0x3984

Bit	31	30	29	28	27	26	25	24
		RP8R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP7R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP6R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP5R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP8R[6:0] Peripheral Output Function is Assigned to RP8 Output Pin bits

Bits 22:16 – RP7R[6:0] Peripheral Output Function is Assigned to RP7 Output Pin bits

Bits 14:8 – RP6R[6:0] Peripheral Output Function is Assigned to RP6 Output Pin bits

Bits 6:0 – RP5R[6:0] Peripheral Output Function is Assigned to RP5 Output Pin bits

11.3.33 Peripheral Pin Select Output Register 2

Name: RPOR2
Offset: 0x3988

Bit	31	30	29	28	27	26	25	24
		RP12R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP11R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP10R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP9R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP12R[6:0] Peripheral Output Function is Assigned to RP12 Output Pin bits

Bits 22:16 – RP11R[6:0] Peripheral Output Function is Assigned to RP11 Output Pin bits

Bits 14:8 – RP10R[6:0] Peripheral Output Function is Assigned to RP10 Output Pin bits

Bits 6:0 – RP9R[6:0] Peripheral Output Function is Assigned to RP9 Output Pin bits

11.3.34 Peripheral Pin Select Output Register 4

Name: RPOR4
Offset: 0x3990

Bit	31	30	29	28	27	26	25	24
		RP20R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP19R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP18R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP17R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP20R[6:0] Peripheral Output Function is Assigned to RP20 Output Pin bits

Bits 22:16 – RP19R[6:0] Peripheral Output Function is Assigned to RP19 Output Pin bits

Bits 14:8 – RP18R[6:0] Peripheral Output Function is Assigned to RP18 Output Pin bits

Bits 6:0 – RP17R[6:0] Peripheral Output Function is Assigned to RP17 Output Pin bits

11.3.35 Peripheral Pin Select Output Register 5

Name: RPOR5
Offset: 0x3994

Bit	31	30	29	28	27	26	25	24
		RP24R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP23R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP22R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP21R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP24R[6:0] Peripheral Output Function is Assigned to RP24 Output Pin bits

Bits 22:16 – RP23R[6:0] Peripheral Output Function is Assigned to RP23 Output Pin bits

Bits 14:8 – RP22R[6:0] Peripheral Output Function is Assigned to RP22 Output Pin bits

Bits 6:0 – RP21R[6:0] Peripheral Output Function is Assigned to RP21 Output Pin bits

11.3.36 Peripheral Pin Select Output Register 6

Name: RPOR6
Offset: 0x3998

Bit	31	30	29	28	27	26	25	24
		RP28R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP27R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP26R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP25R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP28R[6:0] Peripheral Output Function is Assigned to RP28 Output Pin bits

Bits 22:16 – RP27R[6:0] Peripheral Output Function is Assigned to RP27 Output Pin bits

Bits 14:8 – RP26R[6:0] Peripheral Output Function is Assigned to RP26 Output Pin bits

Bits 6:0 – RP25R[6:0] Peripheral Output Function is Assigned to RP25 Output Pin bits

11.3.37 Peripheral Pin Select Output Register 8

Name: RPOR8
Offset: 0x39A0

Bit	31	30	29	28	27	26	25	24
		RP36R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP35R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP34R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP33R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP36R[6:0] Peripheral Output Function is Assigned to RP36 Output Pin bits

Bits 22:16 – RP35R[6:0] Peripheral Output Function is Assigned to RP35 Output Pin bits

Bits 14:8 – RP34R[6:0] Peripheral Output Function is Assigned to RP34 Output Pin bits

Bits 6:0 – RP33R[6:0] Peripheral Output Function is Assigned to RP33 Output Pin bits

11.3.38 Peripheral Pin Select Output Register 9

Name: RPOR9
Offset: 0x39A4

Bit	31	30	29	28	27	26	25	24
		RP40R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP39R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP38R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP37R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP40R[6:0] Peripheral Output Function is Assigned to RP40 Output Pin bits

Bits 22:16 – RP39R[6:0] Peripheral Output Function is Assigned to RP39 Output Pin bits

Bits 14:8 – RP38R[6:0] Peripheral Output Function is Assigned to RP38 Output Pin bits

Bits 6:0 – RP37R[6:0] Peripheral Output Function is Assigned to RP37 Output Pin bits

11.3.39 Peripheral Pin Select Output Register 10

Name: RPOR10
Offset: 0x39A8

Bit	31	30	29	28	27	26	25	24
		RP44R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP43R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP42R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP41R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP44R[6:0] Peripheral Output Function is Assigned to RP44 Output Pin bits

Bits 22:16 – RP43R[6:0] Peripheral Output Function is Assigned to RP43 Output Pin bits

Bits 14:8 – RP42R[6:0] Peripheral Output Function is Assigned to RP42 Output Pin bits

Bits 6:0 – RP41R[6:0] Peripheral Output Function is Assigned to RP41 Output Pin bits

11.3.40 Peripheral Pin Select Output Register 12

Name: RPOR12
Offset: 0x39B0

Bit	31	30	29	28	27	26	25	24
		RP52R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP51R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP50R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP49R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP52R[6:0] Peripheral Output Function is Assigned to RP52 Output Pin bits

Bits 22:16 – RP51R[6:0] Peripheral Output Function is Assigned to RP51 Output Pin bits

Bits 14:8 – RP50R[6:0] Peripheral Output Function is Assigned to RP50 Output Pin bits

Bits 6:0 – RP49R[6:0] Peripheral Output Function is Assigned to RP49 Output Pin bits

11.3.41 Peripheral Pin Select Output Register 13

Name: RPOR13
Offset: 0x39B4

Bit	31	30	29	28	27	26	25	24
		RP56R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP55R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP54R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP53R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP56R[6:0] Peripheral Output Function is Assigned to RP56 Output Pin bits

Bits 22:16 – RP55R[6:0] Peripheral Output Function is Assigned to RP55 Output Pin bits

Bits 14:8 – RP54R[6:0] Peripheral Output Function is Assigned to RP54 Output Pin bits

Bits 6:0 – RP53R[6:0] Peripheral Output Function is Assigned to RP53 Output Pin bits

11.3.42 Peripheral Pin Select Output Register 14

Name: RPOR14
Offset: 0x39B8

Bit	31	30	29	28	27	26	25	24
		RP60R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP59R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP58R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP57R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP60R[6:0] Peripheral Output Function is Assigned to RP60 Output Pin bits

Bits 22:16 – RP59R[6:0] Peripheral Output Function is Assigned to RP59 Output Pin bits

Bits 14:8 – RP58R[6:0] Peripheral Output Function is Assigned to RP58 Output Pin bits

Bits 6:0 – RP57R[6:0] Peripheral Output Function is Assigned to RP57 Output Pin bits

11.3.43 Peripheral Pin Select Output Register 15

Name: RPOR15
Offset: 0x39BC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		RP61R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 6:0 – RP61R[6:0] Peripheral Output Function is Assigned to RP61 Output Pin bits

11.3.44 Peripheral Pin Select Output Register 16

Name: RPOR16
Offset: 0x39C0

Bit	31	30	29	28	27	26	25	24
		RP68R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP67R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP66R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP65R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP68R[6:0] Peripheral Output Function is Assigned to Virtual Output RP68

Bits 22:16 – RP67R[6:0] Peripheral Output Function is Assigned to Virtual Output RP67

Bits 14:8 – RP66R[6:0] Peripheral Output Function is Assigned to Virtual Output RP66

Bits 6:0 – RP65R[6:0] Peripheral Output Function is Assigned to Virtual Output RP65

11.3.45 Peripheral Pin Select Output Register 17

Name: RPOR17
Offset: 0x39C4

Bit	31	30	29	28	27	26	25	24
		RP72R[6:0]						
Access		RW	RW	RW	RW	RW	RW	RW
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP71R[6:0]						
Access		RW	RW	RW	RW	RW	RW	RW
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP70R[6:0]						
Access		RW	RW	RW	RW	RW	RW	RW
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP69R[6:0]						
Access		RW	RW	RW	RW	RW	RW	RW
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP72R[6:0] Peripheral Output Function is Assigned to Virtual Output RP72

Bits 22:16 – RP71R[6:0] Peripheral Output Function is Assigned to Virtual Output RP71

Bits 14:8 – RP70R[6:0] Peripheral Output Function is Assigned to Virtual Output RP70

Bits 6:0 – RP69R[6:0] Peripheral Output Function is Assigned to Virtual Output RP69

11.3.46 Peripheral Pin Select Output Register 18

Name: RPOR18
Offset: 0x39C8

Bit	31	30	29	28	27	26	25	24
		RP76R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP75R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP74R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP73R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP76R[6:0] Peripheral Output Function is Assigned to Virtual Output RP76

Bits 22:16 – RP75R[6:0] Peripheral Output Function is Assigned to Virtual Output RP75

Bits 14:8 – RP74R[6:0] Peripheral Output Function is Assigned to Virtual Output RP74

Bits 6:0 – RP73R[6:0] Peripheral Output Function is Assigned to Virtual Output RP73

11.3.47 Peripheral Pin Select Output Register 19

Name: RPOR19
Offset: 0x39CC

Bit	31	30	29	28	27	26	25	24
		RP80R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP79R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP78R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP76R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP80R[6:0] Peripheral Output Function is Assigned to Virtual Output RP80

Bits 22:16 – RP79R[6:0] Peripheral Output Function is Assigned to Virtual Output RP79

Bits 14:8 – RP78R[6:0] Peripheral Output Function is Assigned to Virtual Output RP78

Bits 6:0 – RP76R[6:0] Peripheral Output Function is Assigned to Virtual Output RP76

11.3.48 IOIM x Control Register

Name: IOIMxCON
Offset: 0x1E90, 0x1E9C, 0x1EA8, 0x1EB4

Bit	31	30	29	28	27	26	25	24
					FLTINJ	OKINJ	ATEST[1:0]	
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EOVFV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SLPEN	SIDL		EXTCLK		
Access	R/W		R/W	R/W		R/W		
Reset	0		0	0		0		

Bit	7	6	5	4	3	2	1	0
	FBKSEL[3:0]				REFSEL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 27 – FLTINJ Fault Injection bit
A write of '1' to this bit will simulate a comparison mismatch event. The ERR bit will become set, the ERRCNT will increment to EOVFV and the OVF will become set if the ERRCNT overflows.

Bit 26 – OKINJ OK Inject bit
A write of '1' to this bit will simulate a transition on the reference signal input, blanking time insertion and a good comparison between reference and feedback signals. The OK bit will become set.

Bits 25:24 – ATEST[1:0] Artificial Test Enable bit

Value	Description
11	Artificial OKINJ Test is enabled
10	Artificial FLTINJ Test is enabled
01	Artificial Test disabled
00	Artificial Test disabled

Bits 23:16 – EOVFV[7:0] Error-counter Overflow Value bits

Bit 15 – ON Module Enable bit

Value	Description
1	IOIM module is enabled
0	IOIM module is disabled

Bit 13 – SLPEN Module Sleep Enable bit

Value	Description
1	Module operates in Sleep mode
0	Module disabled in Sleep mode

Bit 12 – SIDL Module Stop in Idle Mode Enable bit

Value	Description
1	Module disabled in Idle mode
0	Module operates in Idle mode

Bit 10 – EXTCLK External Clock Source Enable bit

Value	Description
1	CLKGEN13 (200 MHz)
0	T_{CY} (Instruction Cycle) (default)

Bits 7:4 – FBKSEL[3:0] Feedback Input Mux Selection bits

Value	Description
1111	Feedback input [15] is selected
...	
0001	Feedback input [1] is selected
0000	Feedback input [0] is selected

Bits 3:0 – REFSEL[3:0] Reference Input Mux Selection bits

Value	Description
1111	Reference input [15] is selected
...	
0001	Reference input [1] is selected
0000	Reference input [0] is selected

11.3.49 IOIM x Blanking Time Register

Name: IOIMxBCON
Offset: 0x1E94, 0x1EA0, 0x1EAC, 0x1EB8, 0x1EC4, 0x1ED0, 0x1EDC, 0x1EE8, 0x1EF4, 0x1F00, 0x1F0C, 0x1F18, 0x1F24, 0x1F30, 0x1F3C, 0x1F48

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BLANK[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BLANK[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – BLANK[15:0] Blanking Time Register bits
The 16-bit value is to be the banking time value.

11.3.50 IOIM x Status Register

Name: IOIMxSTAT
Offset: 0x1E98, 0x1EA4, 0x1EB0, 0x1EBC, 0x1EC8, 0x1ED4, 0x1EE0, 0x1EEC, 0x1EF8, 0x1F04, 0x1F10, 0x1F1C, 0x1F28, 0x1F34, 0x1F40, 0x1F4C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ERRCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bits 15:8 – ERRCNT[7:0] Error Counter bits
Indicates the number of mismatches between the reference and feedback inputs after the blanking time has expired.

Bit 7 – FFEDGE Feedback Falling Edge Status bit

Value	Description
1	At least one fall transition has been detected on the feedback input
0	No reference fall edge has occurred

Bit 6 – FREDGE Feedback Rising Edge Status bit

Value	Description
1	At least one rise transition has been detected
0	No reference rise edge has occurred on the feedback input

Bit 5 – RFEDGE Reference Falling Edge Status bit

Value	Description
1	At least one fall transition has been detected on the reference input
0	No reference fall transition has occurred

Bit 4 – RREDGE Reference Rise Edge Status bit

Value	Description
1	At least one rise transition has been detected on the reference input
0	No reference fall transition has occurred

Bit 2 – OVF Overflow bit

Value	Description
1	Error counter has overflowed since the last time it was cleared
0	Error counter has not overflowed

Bit 1 – ERR Error bit

Value	Description
1	At least one mismatch has occurred between the reference and feedback inputs
0	No mismatches have occurred

Bit 0 – OK OK bit

Value	Description
1	At least one transition has been detected on the reference input with no mismatch
0	No transitions detected on the reference input

11.4 Operation

11.4.1 I/O Port Control

Before reading and writing any I/O port, the desired pin or pins should be properly configured for the application. Each I/O port has nine registers directly associated with the operation of the port and one control register. Each I/O port pin has a corresponding bit in these registers. Throughout this section, the letter 'x' denotes any or all port module instances. For example, TRISx would represent TRISA, TRISB, TRISC and so on. Any bit and its associated data and control registers that are not valid for a particular device will be disabled and will read as zeros.

The TRISx registers configure the data direction flow through port I/O pins. The TRISx register bits determine whether a PORTx I/O pin is an input or an output:

- If a data direction bit is '1', the corresponding I/O port pin is an input.
- If a data direction bit is '0', the corresponding I/O port pin is an output.

A read from a TRISx register reads the last value written to that register. All I/O port pins are defined as inputs after a Power-on Reset (POR).

Note: It is recommended to make the pin an output and drive to zero (TRISx = 0,LATx = 0) prior to making the I/O pin an input (TRISx = 1); this will help in discharging the parasitic capacitance internal to the I/O pin.

The PORTx registers allow I/O pins to be accessed; a write to a PORTx register writes to the corresponding LATx register (PORTx data latch). The I/O port pin(s) configured as outputs are updated. A write to a PORTx register is effectively the same as a write to a LATx register. A read from a PORTx register reads the synchronized signal applied to the port I/O pins.

The LATx registers hold data written to port I/O pins; a write to a LATx register latches data to the corresponding port I/O pins. The I/O port pins configured as outputs are updated. A read from a LATx register reads the data held in the PORTx data latch, not from the port I/O pins.

11.4.2 Open-Drain Configuration (ODCx)

Each I/O pin can be individually configured for either normal digital output or open-drain output. This is controlled by the Open-Drain Control register, ODCx, associated with each I/O pin. If the ODCx bit for an I/O pin is a '1', the pin acts as an open-drain output. If the ODCx bit for an I/O pin is a '0', the pin is configured for a normal digital output (the ODCx bit is valid only for output pins). After a Reset, the status of all the bits of the ODCx register is set to '0'.

The open-drain feature allows the generation of outputs higher than Vdd (e.g., 5V) on any desired 5V tolerant pins by using external pull-up resistors. The maximum open-drain voltage allowed is the same as the maximum Vih specification. The ODCx register setting takes effect in all of the I/O modes, allowing the output to behave as an open-drain, even if a peripheral is controlling the pin.

Although the user could achieve the same effect by manipulating the corresponding LATx and TRISx bits, this procedure will not allow the peripheral to operate in Open-Drain mode (except for the default operation of the I^2C pins). Since I^2C pins are already open-drain pins, the ODCx settings do not affect the I^2C pins. Also, the ODCx settings do not affect the JTAG output characteristics as the JTAG scan cells are inserted between the ODCx logic and the I/O.

11.4.3 Configuring Analog and Digital Port Pins (ANSELx)

The ANSELx register controls the operation of the analog port pins. The port pins that are to function as analog inputs must have their corresponding ANSELx and TRISx bits set. To use port pins for I/O functionality with digital modules, such as timers, UARTs, etc., the corresponding ANSELx bit must be cleared.

The ANSELx register has a default value of 0xFFFF; therefore, all pins that share analog functions are, by default, analog and *not* digital.

If the TRISx bit is cleared (output) while the ANSELx bit is set, the digital output level (V_{OH} or V_{OL}) is converted by an analog peripheral, such as the ADC module or the comparator module.

When the PORTx register is read, all pins configured as analog input channels are read as cleared (a low level).

Pins configured as digital inputs do not convert an analog input. Analog levels on any pin defined as a digital input (including the ANx pins) can cause the input buffer to consume current that exceeds the device specifications.

Table 11-14. ANSEL Pin States

ANSELx Bit	TRISx Bit	Pin State
1	0	The output is from analog module
1	1	The port always reads as low
0	0	Digital Output
0	1	Digital Input

11.4.4 Edge Control Rate

The SR1x/SR0x registers are used to control the edge rate of the pin when it is driving a digital output. The Reset value of these register bits (00) results in the fastest edge rate (i.e., no slew rate control).

11.4.5 Peripheral Pin Select (PPS)

The Peripheral Pin Select Input registers, RPINRx, and the Peripheral Pin Select Output registers, RPORx, provide control for PPS input and output mapping. See 11.4.5.3. Input Mapping and 11.4.5.4. Output Mapping for detailed information on configuring these registers.

11.4.5.1 Available Pins

The number of available pins is dependent on the particular device and its pin count. Pins that support the Peripheral Pin Select feature include the designation, “RPn”, in their full pin designation, where “RP” designates a remappable peripheral and “n” is the remappable pin number. Note that on a device’s schematic symbol, remappable input pins are designated as RPIn and remappable output pins are designated as RPn.

11.4.5.2 Available Peripherals

The peripherals managed by the Peripheral Pin Select are all digital-only peripherals. These include general serial communications (UART and SPI), general purpose timer clock inputs, timer-related peripherals (input capture and output compare) and interrupt-on-change inputs.

In comparison, some digital-only peripheral modules are never included in the Peripheral Pin Select feature. This is because the peripheral’s function requires special I/O circuitry on a specific port and

cannot be easily connected to multiple pins. These modules include I^2C and the motor control PWM. A similar requirement excludes all modules with analog inputs, such as an A/D Converter.

A key difference between remappable and non-remappable peripherals is that remappable peripherals are not associated with a default I/O pin. The peripheral must always be assigned to a specific I/O pin before it can be used. In contrast, non-remappable peripherals are always available on a default pin, assuming that the peripheral is active and not conflicting with another peripheral.

When a remappable peripheral is active on a given I/O pin, it takes priority over all other digital I/Os and digital communication peripherals associated with the pin. Priority is given regardless of the type of peripheral that is mapped. Remappable peripherals never take priority over any analog functions associated with the pin.

11.4.5.3 Input Mapping

The inputs of the Peripheral Pin Select options are mapped on the basis of the peripheral; that is, a control register associated with a peripheral dictates the pin it will be mapped to. The RPINRx registers are used to configure peripheral input mapping. Each register contains sets of 8-bit fields, with each set associated with one of the remappable peripherals. Programming a given peripheral's bit field with an appropriate 8-bit value maps the RPn pin with the corresponding value to that peripheral.

Example: Interrupt 1 input (from Table 11-10) mapped to PWM Event 4 (from Table 11-11)

_INT1R[7:0] = 168; // 168 = PWM EVENT 4

11.4.5.4 Output Mapping

In contrast to inputs, the outputs of the PPS options are mapped on the basis of the pin. In this case, a control register associated with a particular pin dictates the peripheral output to be mapped. The RPORx registers are used to control output mapping. Each register contains sets of six-bit fields, with each set associated with one RPn pin. The value of the bit field corresponds to one of the peripherals and that peripheral's output is mapped to the pin (see Table 11-13).

A null output is associated with the Output Register Reset value of '0'. This is done to ensure that, by default, remappable outputs remain disconnected from all output pins.

11.4.5.5 Mapping Limitations

The control scheme of the peripheral select pins is not limited to a small range of fixed peripheral configurations. There are no mutual or hardware enforced lockouts between any of the peripheral mapping SFRs. Any combination of peripheral mappings across any or all of the RPn pins is possible. This includes both many-to-one and one-to-many mappings of peripheral inputs and outputs to pins.

11.4.6 Controlling Configuration Changes

Because peripheral mapping can be changed during run time, some restrictions on peripheral remapping are needed to prevent accidental configuration changes. All dsPIC/PIC devices include a control register lock bit to prevent alterations to the peripheral map.

Under normal operation, writes to the RPINRx and RPORx registers are not allowed. Attempted writes will appear to execute normally, but the contents of the registers will remain unchanged. To change these registers, they must be unlocked in hardware. The register lock is controlled by the IOLOCK bit (RPCON[11]). Setting IOLOCK prevents writes to the control registers; clearing IOLOCK allows writes.

11.4.7 Considerations for Peripheral Pin Selection

The ability to control Peripheral Pin Selection introduces several considerations into application design.

Peripheral Pin Selects are not available on default pins in the device's default (Reset) state. More specifically, because all RPINRx registers reset to '1's and RPORx registers reset to '0's, this means

all PPS inputs are tied to Vss, while all PPS outputs are disconnected. This means that before any other application code is executed, the user must initialize the device with the proper peripheral configuration. Because the IOLOCK bit resets in the unlocked state, it is not necessary to execute the unlock sequence after the device has come out of Reset. For application safety, however, it is always better to set IOLOCK and lock the configuration after writing to the control registers.

Choosing the configuration requires a review of all Peripheral Pin Selects and their pin assignments, particularly those that will not be used in the application. In all cases, unused pin-selectable peripherals should be disabled completely. The RPn functions will be cleared as default.

The assignment of a peripheral to a pin does not perform any other configuration of the pin's I/O circuitry. This means that adding a pin-selectable output to a pin may mean inadvertently driving an existing peripheral input when the output is driven. Users must be familiar with the behavior of other fixed peripherals that share a remappable pin and know when to enable or disable them. To be safe, fixed digital peripherals that share the same pin should be disabled when not in use.

Along these lines, configuring a remappable pin for a specific peripheral does not automatically turn that feature on. The peripheral must be specifically configured for operation and enabled as if it were tied to a fixed pin. Where this happens in the application code (immediately following device Reset and peripheral configuration or inside the main application routine) depends on the peripheral and its use in the application.

A final consideration is that Peripheral Pin Select functions neither override analog inputs nor reconfigure pins with analog functions for digital I/Os. If a pin is configured as an analog input on device Reset, it must be explicitly reconfigured as a digital I/O when used with a Peripheral Pin Select.

Example 11-1 provides a configuration for bidirectional communication with flow control using UART1. The following input and output functions are used:

- Input Functions: U1RX, $\overline{\text{U1CTS}}$
- Output Functions: U1TX, $\overline{\text{U1RTS}}$

Example 11-1. Configuring UART1 Input and Output Functions

```
//Configure Input Functions(See Table 11-10)

// Assign U1Rx To Pin RP35
_U1RXR = 35;

// Assign U1CTS To Pin RP36
_U1CTSR = 36;

// Configure Output Functions
// Assign U1Tx To Pin RP37
_RP37 = 9;                      // 9= U1TX (see Table 11-13)

// Assign U1RTS To Pin RP38
_RP38 = 10;                      // 10 = U1RTS (see Table 11-13)
```

11.4.8 Virtual Output Pins

The virtual pins (RPVn) enable the user to connect internal peripherals, whose signals may be of significant use to other peripherals, but these outputs may not need to be presented to a device pin.

The concept of "virtual pins" enables device features to be mapped to other peripherals that do not otherwise have an dedicated connection.

A simple example use is a fault input to the PWM from the CLC as shown in example 3-x. The CLC output is mapped to a virtual output pin (RPVn), and the PWM input is mapped to the virtual pin. Virtual pins are associated with an input index from Table 11-11.

Example 11-2. Virtual PPS connection

```
// Map CLC output to virtual pin RP176
_RP176R = 0x28;          // 28 = CLC2OUT

// Map Input PWM PCI 8 to virtual output pin RP176
_PCI8R = 176
```

11.4.9 Peripheral Multiplexing

When a peripheral is enabled, the associated pin output drivers are typically module controlled, while a few are user-settable. The I/O pin may be read through the input data path, but the output driver for the I/O port bit is generally disabled.

Note: Some ports are shared with analog module pins. The corresponding bits in the ANSELx registers, if present, must be set to '0' for I/O port functionality.

11.4.10 I/O Multiplexing with Multiple Peripherals

For some dsPIC/PIC devices, particularly those with a small number of I/O pins, multiple peripheral functions may be multiplexed on each I/O pin.

The name of the I/O pin defines the priority of each function associated with the pin. The conceptual I/O pin has two multiplexed peripherals, Peripheral A and Peripheral B, and is named as PERA/PERB/PIO.

The I/O pin name is chosen because the user-assigned application can easily determine the priority of the functions assigned to the pin. Peripheral A has the highest priority for control of the pin. If Peripheral A and Peripheral B are enabled at the same time, Peripheral A will take control of the I/O pin.

11.4.10.1 Software Input Pin Control

Some of the functions assigned to an I/O pin may be input functions that do not take control of the pin output driver. An example of one such peripheral is the input capture module. If the I/O pin associated with the input capture is configured as an output, using the appropriate TRISx control bit, the user can manually affect the state of the input capture pin through its corresponding PORTx register. This behavior can be useful in some situations, especially for testing purposes, when no external signal is connected to the input pin.

The organization of the peripheral multiplexers will determine if the peripheral input pin can be manipulated in software using the PORTx register.

In general, the following peripherals allow their input pins to be controlled manually through the PORTx registers:

- External Interrupt Pins
- Timer Clock Input Pins
- Input Capture Pins
- PWM Fault Pins

Most serial communication peripherals, when enabled, take control of the I/O pin so that the input pins associated with the peripheral cannot be affected through the corresponding PORTx registers. Example peripherals include the following:

- SPI
- I^2C
- DCI
- UART

- CAN FD
- QEI

11.4.10.2 Pin Control Summary

When a peripheral is enabled, the associated pin output drivers are typically module controlled, while a few are user-settable. The term, “module control”, means that the associated port pin output driver is disabled, and the pin can only be controlled and accessed by the peripheral. The term, “user settable”, means that the associated peripheral port pin output driver is user-configurable in software through the associated TRISx Special Function Register (SFR). The TRISx register must be set for the peripheral to function properly. For “user-settable” peripheral pins, the actual port pin state can always be read through the PORTx SFR.

An input capture peripheral provides an example of a user-settable peripheral. The user application must write the associated TRISx register to configure the input capture pin as an input. Because the I/O pin circuitry is still active when the input capture is enabled, the following method can be used to manually produce capture events using software:

- The input capture pin is configured as an output using the associated TRISx register.
- Then, the software can write values to the corresponding LATx register drive to internally control the input capture pin and force capture events.

As another example, an INTx pin can be configured as an output, and then by writing to the associated LATx bit, an INTx interrupt, if enabled, can be generated.

The UART is an example of a module control peripheral. When the UART is enabled, the PORTx and TRISx registers have no effect and cannot be used to read or write the RX and TX pins. Most communication peripheral functions available on the dsPIC/PIC devices are module control peripherals.

For example, the SPI module can be configured for Host mode, in which only the SDO pin is required. In this scenario, the SDI pin can be configured as a general purpose output pin by clearing (setting to a logic ‘0’) the associated TRISx bit. For more information on how pins can be configured for a module, refer to the specific module section.

11.4.10.3 Multiplexing Digital Input Peripheral

The following conditions are characteristic of a multiplexed digital input peripheral:

- Peripheral does not control the TRISx register. Some peripherals require the pin be configured as an input by setting the corresponding TRISx bit = 1.
- Peripheral input path is independent of I/O input path and uses an input buffer that is dependent on the peripheral.
- PORTx register data input path is not affected and is able to read the pin value.

11.4.10.4 Multiplexing Digital Output Peripheral

The following conditions are characteristic of a multiplexed digital output peripheral:

- Peripheral controls the output data. Some peripherals require the pin be configured as an output by setting the corresponding TRISx bit = 0.
- If a peripheral pin has an automatic tri-state feature (e.g., PWM outputs), the peripheral has the ability to tri-state the pin.
- Pin output driver type could be affected by the peripheral (e.g., drive strength, slew rate, etc.).
- PORTx register output data has no effect.

11.4.10.5 Multiplexing Digital Bidirectional Peripheral

The following conditions are characteristic of a multiplexed digital bidirectional peripheral:

- Peripheral automatically configures the pin as an output, but not as an input. Some peripherals require the pin be configured as an input by setting the corresponding TRISx bit = 1 .

- Peripherals control the output data.
- Pin output driver type could be affected by the peripheral (e.g., drive strength, slew rate, etc.).
- PORTx register data input path is not affected and is able to read the pin value.
- PORTx register output data has no effect.

11.4.10.6 Multiplexing Analog Input Peripheral

The following condition is characteristic of a multiplexed analog input peripheral:

- All digital port input buffers are disabled and PORTx registers read '0' to prevent "crowbar" current.

11.4.10.7 Multiplexing Analog Output Peripheral

The following conditions are characteristic of a multiplexed analog output peripheral:

- All digital port input buffers are disabled and PORTx registers read '0' to prevent crowbar current.
- Analog output is driven onto the pin independent of the associated TRISx setting.

Note: To use pins that are multiplexed with the ADC module for digital I/Os, the corresponding bits in the ANSELx register, if present, must be set to '0', even if the ADC module is turned off.

11.4.11 Change Notice (CN)

The Change Notification pins provide dsPIC devices the ability to generate interrupt requests to the processor in response to a Change-of-State (COS) on selected input pins (corresponding TRISx bits must be = 1).

The enabled pin values are compared with the values sampled during the last read operation of the designated PORTx register. If the pin value is different from the last value read, a mismatch condition is generated. The mismatch condition can occur on any of the enabled input pins. The mismatches are "ORed" together to provide a single interrupt-on-change signal. The enabled pins are sampled on every internal system clock cycle, SYSCLK.

11.4.11.1 CN Configuration and Operation

The CN pins are configured as follows:

1. Disable CPU interrupts.
2. Set the desired CN I/O pin as an input by setting the corresponding TRISx register bits = 1.

   Note: If the I/O pin is shared with an analog peripheral, it may be necessary to configure this pin as digital input.
3. Enable the CN Module by setting the ON bit (CNCONx[15]) = 1.
4. Enable individual CN input pins; enable optional pull-ups or pull-downs.
5. Read the corresponding PORTx registers to clear the CN interrupt.
6. Configure the CNx Interrupt Priority bits, CNxIP[2:0].
7. Clear the CNx Interrupt Flag bit by setting the CNxIF bit (IFSx register) = 0.
8. Configure the CNx pin interrupt for either Mismatch mode or Edge Detect mode using the CNSTYLE bit (CNCONx[11]). If Mismatch mode is selected, enable the individual CNx function using the CNEN0x bits. If Edge Detect mode is selected, use the CNEN0x bits to enable positive edge detection and the CNEN1x bits to enable negative edge detection.
9. Enable the CNx Interrupt Enable bit by setting the CNxIE bit (IECx register) = 1.
10. Enable CPU interrupts.

    The CNSTATx/CNFx registers indicate whether a change occurred on the corresponding pin since the last read of the PORTx bit.

    The CNFx registers indicate a valid edge detect event has occurred when CNSTYLE = 1. CNFx bits need to be cleared by the user to set up the CN logic to detect the next edge transition. In

Edge Detect mode, a CN interrupt can be controlled to occur only during a rising or falling edge condition on a pin. The CNSTATx are read-only registers that indicate a valid Mismatch mode event has occurred when CNSTYLE = 0.

When a CN interrupt occurs in Mismatch mode, the user should read the PORTx register associated with the CN pins. This will clear the mismatch condition and set up the CN logic to detect the next pin change. The CN pins have a minimum input pulse-width specification. Refer to the **"Electrical Characteristics**" chapter to learn more.

11.4.12 I/O integrity Module (IOIM)

The purpose of the I/O Integrity Monitor Macro is to provide safety components to a PICmicro device. To keep up with the safety demand, this macro is primarily designed to monitor the connectivity of a selected reference signal against a feedback signal with user configurable path delay.

Typical uses include:

- To increase protection for customers who are building safety sensitive applications
- To simplify structural tests for safety applications
- To ensure the pin has been connected correctly in its final environment by using structural tests

The I/O Integrity Monitor is a change detector circuit that compares a reference signal with a selected feedback signal. The module detects a change in the reference signal to start the blanking timer, and the blanking timer is used to account for the feedback path delay. After a programmable amount of time, the state of the selected feedback input is compared with the state of the reference input.

If a mismatch is sampled at any time or a configured time after the blanking period, then an error event is generated. The module gives an Error event interrupt on a mismatch event.

Every error event is used to increment the user-defined Error Counter value; the module gives an overflow status bit once the Error Counter reaches the overflow state.

A typical application for the I/O Integrity Monitor is to compare the data supplied to an output pin with the data read from the output pin driver. The output pin driver data is supplied to the monitor via an input buffer. If desired, a second pin can be used to monitor the output pin data at a remote location, somewhere on the PCB or far from the chip.

The I/O Integrity Monitor is typically used to monitor the integrity of a signal generated by the MCU. However, it can also be used to monitor any signal generated outside of the MCU by connecting the reference input and feedback inputs to the appropriate external monitoring points.

11.4.12.1 Enabling the Module

The module is enabled when IOIMCON[ON] = 1.

When IOIMCON[ON] = 0, the module is considered disabled and consumes minimum power.

11.4.12.2 Configuration of IOIMCON

IOIMCON selects the IO monitor option and determines comparison pulse duration, like high pulse, low pulse or both low and high pulse period comparison. ON must be set for the IOIM to operate. All registers can be programmed while ON is clear. The I/O pin must be configured as a digital output for the IO monitor to be checked, and the presence of the port is subject to package variant.

11.4.12.2.1 Clock Selection

The IOIMCON.EXTCLK bit determines the clock source that operates the timebase and other module logic.

When EXTCLK = 0, the module timebase and logic are clocked from the system instruction cycle clock, T_{CY}.

The external clock selection is dependent on the device definition. The external clock source may or may not be synchronous to the Peripheral Clock domain. Synchronization logic is required between the SFR register space (Peripheral Clock domain) and the module logic (IO Monitor Clock domain) to allow the module to operate from high speed asynchronous clock sources.

11.4.12.2.2 Reference Input Selection

The feedback input is selected from a sixteen input multiplexer as shown in Figure 11-5. The value for the reference pin will be taken only from the output of that particular selected pin. For example, if the selected pin is the pin out of PWM, the PWM out is taken as the reference input; this can also be manually updated by setting LAT of the selected pin to 0/1.

11.4.12.2.3 Feedback Input Selection

The feedback input is selected from a sixteen input multiplexer as shown in Figure 11-5. The value for the feedback pin can be taken from both output and input of the particular pin. For example, the feedback can be input from an external value or can be manually updated by setting LAT of the selected pin to 0/1.

11.4.12.2.4 Write Protection

In this implementation, the assigned LOCK and WREN are register bits located in the PACCON1 register responsible for all of the device's peripheral access control.

11.4.12.2.5 Blanking Time

IOIMBTCON.BLANK sets the blanking time. The mismatch event is valid after the blanking timer has counted down to zero. The blanking time is introduced to account for I/O pad delays or associated routing path delays in the feedback input with respect to reference input. Blanking time is reloaded into the blanking timer (as shown in Example 11-3) every time the reference pin is toggled to high.

Example 11-3. IOIM Code

```
#include "xc.h"

int main(void)
{
    IOIM1CONbits.ON = 0;          // module is turned off.
    IOIM1STAT = 0;                // clearing all status flags

    // both of the above steps clears the module status bits.

    IOIM1CONbits.TESTEN = 0x10;  // set 10 for FAULT injection mode and 11 for OK
injection mode.
    IOIM1BCONbits.BLANK = 100;   // set to make sure the required delay is given between
reference and feedback.

    // we dont need to select any any reference and feedback pins as it is internal

    IOIM1CONbits.EOVFVAL = 10;   // set err count overflow value

    _IOM1IF = 0; //clear interrupt flag
    _IOM1IE = 1; //enable interrupt

    IOIM1CONbits.FLTINJ = 0;     // make sure fault inject is clear
    //IOIM1CONbits.OKINJ = 0;    // make sure ok inject is clear

    IOIM1CONbits.ON = 1;         // module is turned on

    IOIM1CONbits.FLTINJ = 1;     // set to inject fault condition
//    IOIM1CONbits.OKINJ = 1;    // set to inject ok condition

    while(1);

    return 0;
}

void __attribute__((interrupt, no_auto_psv)) _IOIM1Interrupt(void)
{

    _IOM1IF = 0;                // clear interrupt flag
    IOIM1STATbits.ERR = 0;
}
```

11.4.12.2.6 Self-Test

The IOIMCON SFR contains the FLTINJ bit and OKINJ bit, which allow user software to simulate a IOIM failure/OK. This feature allows the user to fully test fault handling software (see 11.3.48. IOIMxCON). The mode for artificial test must be set accordingly.

Table 11-15. IOIM Test Conditions

Operating Mode	ATEST Bit	OKINJ Bit	FLTINJ Bit	Register Bits Valid				Comments
				OK	ERR	ERRCNT	OVF	
Functional mode	00/01	x	x	Yes	Yes	Yes	Yes	Reference and feedback signals are taken from external signals.
	11	1	0	1	0	0	0	It is a module health test. Feedback and reference signals are injected from the register bit (OKINJ) by user.
	10	0	1	0	1	Yes	Yes	It is a module health test. Feedback and reference signals are injected from the register bit (FLTINJ) by user.

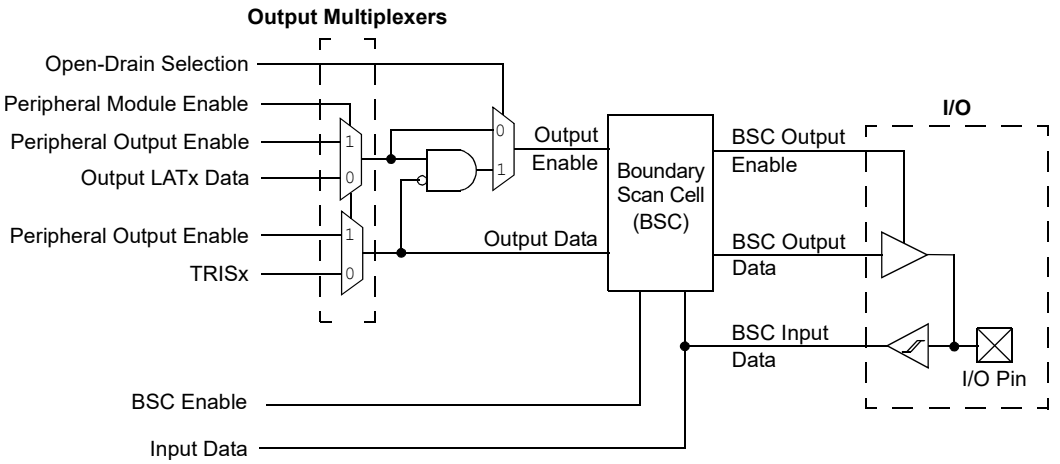
Notes:

1. It is not recommended to set the OKINJ and FLTINJ bits at the same time.
2. OK bit is not set when FLTINJ = 1.
3. It is recommended to disable and then enable the module using the ON bit after mode changes.

11.4.13 Boundary Scan Cell Connections

The dsPIC33A family devices support JTAG boundary scan. A Boundary Scan Cell (BSC) is inserted between the internal I/O logic circuit and the I/O pin, as shown in Figure 11-6. Most of the I/O pads have Boundary Scan Cells; however, JTAG pads do not. For normal I/O operation, the BSC is disabled, and therefore, is bypassed. The output enable input of the BSC is directly connected to the BSC output enable, and the output data input of the BSC is directly connected to the BSC output data. The pads that do not have BSC are the power supply pads (V_{DD}, V_{SS} and V_{CAP}/V_{CORE}) and the JTAG pads (TCK, TDI, TDO and TMS).

Figure 11-6. Boundary Scan Cell Connections

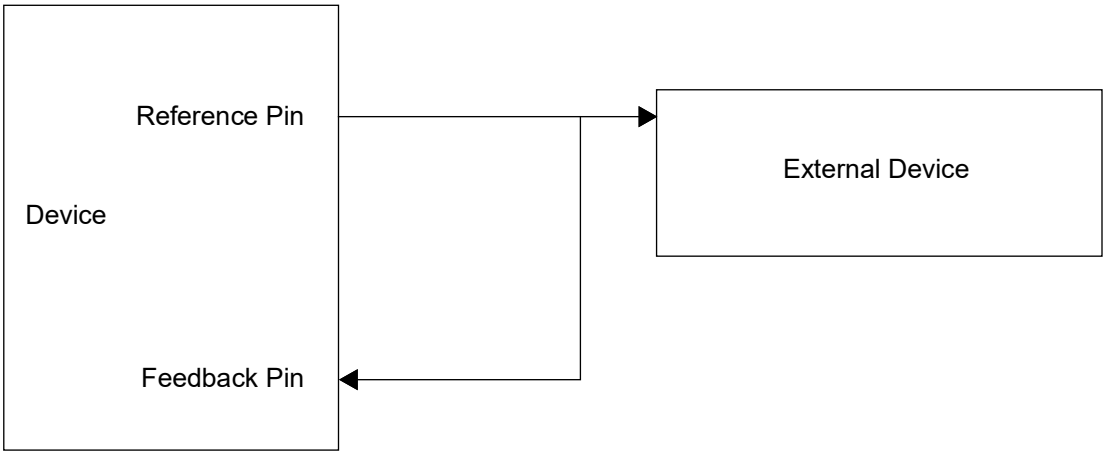


11.5 Applications

11.5.1 Application Use of IOIM

The setup in code example is used to showcase the general application of IOIM. The reference pin will be in output configuration and feedback will be input. The delay between the signal leaving the reference pin and being received in the feedback pin is taken into consideration using the blanking timer.

Figure 11-7. General IOIM Application



Example 11-4.

```
#include "xc.h"

int main(void)
{
    IOIM1CONbits.ON = 0;                 // module is turned off.
    IOIM1STAT = 0;                       // clearing all status flags

    // both of the above steps clears the module status bits.

    IOIM1CONbits.TESTEN = 0x00;         // set 00/01 to disable artificial test.
    IOIM1BCONbits.BLANK = 100; // set to make sure the required delay is given between
reference and feedback.

    IOIM1CONbits.REFSEL = 8;            // this is according to datasheet (pin rc8)
    IOIM1CONbits.FBKSEL = 7;            // this is according to datasheet (pin rc7)
    IOIM1CONbits.EOVFVAL = 10;          // set err count overflow value

    _IOM1IF = 0;                        // clear interrupt flag
    _IOM1IE = 1;                        // enable interrupt

    _TRISC8 = 0;                        // set the reference pin to output
    _LATC8 = 0;                         // set the reference pin value

    IOIM1CONbits.ON = 1;                // module is turned on

    while(1);

    return 0;
}

void __attribute__((interrupt, no_auto_psv)) _IOIM1Interrupt(void)
{

    _IOM1IF = 0;                        // clear interrupt flag
    IOIM1STATbits.ERR = 0;
    if(IOIM1STATbits.OVF)
    {
        IOIM1STATbits.OVF = 0;
        // your overflow application code goes here
    }
}
```

11.6 Interrupts

I/O Modules have Change Notification interrupts.

Interrupt Type	Condition	Flag
Change Notification	Whenever the data to the pin is in accordance with the Change notification configuration the interrupt is triggered. There is only one flag for each PORT and the exact pin should be checked in the specified registers.	CNxIF

IO Integrity Module has Error Notification interrupt.

Interrupt Type	Condition	Flag
IOIM Interrupt	Whenever the ERR bit is set during the working of IOIM, this interrupt is triggered. There is one interrupt for every instance of the module.	IOIMxIF

11.7 Operation in Power-Saving Modes

11.7.1 I/O Port Operation in Sleep Mode

As the device enters Sleep mode, the system clock is disabled; however, the CN module continues to operate asynchronously. If one of the enabled CN pins changes state, the CNxIF bit (IFSx register) will be set. If the CNxIE bit (IECx register) is set, and its priority is greater than the current CPU priority, the device will wake from Sleep mode and execute the CN Interrupt Service Routine (ISR).

If the assigned priority level of the CN interrupt is less than or equal to the current CPU priority level, the CPU will not be awakened and the device will enter Idle mode.

11.7.2 I/O Port Operation in Idle Mode

As the device enters Idle mode, the system clock sources remain functional and the CN module continues to operate synchronously. If one of the enabled CN pins changes state, the CNxIF bit (IFSx register) will be set. If the CNxIE bit (IECx register) is set, and its priority is greater than the current CPU priority, the device will wake from Idle mode and execute the CN Interrupt Service Routine (ISR).

11.7.3 I/O Integrity Module Operations in Sleep Mode

When the device enters Sleep mode, the Peripheral Clock is disabled. The module can be selected to run with an external clock source. The Sleep Enable bit (IOIMCON.SLPEN) selects whether the monitor function will stop in Sleep mode or continue to operate.

- If SLPEN = 1, the module is enabled during Sleep and will continue to monitor I/Os
- If SLPEN = 0, the module is not enabled during Sleep and will not monitor I/Os

Notes:

1. If the module is enabled during Sleep and requests an external clock source that can remain active in Sleep mode, the module will continue operation on that clock source in Sleep.
2. The monitor module must be clocked from a source available in Sleep for continued operation in Sleep mode.
3. An Interrupt event will wake the device from Sleep when the module interrupt is enabled.

11.7.4 I/O Integrity Module Operations in Idle Mode

When the device enters Idle mode, the peripheral clock sources remain functional and the CPU stops executing code. The Sleep in Idle bit (IOIMCON.SIDL) selects whether the monitor function will stop in Idle mode or continue to operate in Idle mode.

- If IOIMCON.SIDL = 1'b0, the module will continue normal operation in Idle mode.
- If IOIMCON.SIDL = 1'b1, the module will stop when the device is in Idle mode. The module will perform the same procedures when stopped in Idle mode as for Sleep mode.

11.7.5 Interrupt Operation

This module generates a single interrupt on a mismatch event. The IOIMSTAT.ERR bits reflect the interrupt.

The mismatch/error is generated when the mismatch is greater than two IOIM clocks. The mismatch is generated on every mismatch event and the OVF is set when the number of mismatches becomes equal to the defined overflow value; the OVF status bit doesn't generate an interrupt. The Error status bits reflection takes seven IOIM clock latencies, and it is a user responsibility to clear Status bits.

11.8 Effects of Various Resets

11.8.1 Device Reset

All I/O registers are forced to their Reset states upon a device Reset.

11.8.2 Power-on Reset

All I/O registers are forced to their Reset states upon a Power-on Reset (POR).

11.8.3 Watchdog Timer Reset

All I/O registers are unchanged upon a Watchdog Timer Reset.

12. Oscillator and Clocking Module

This section describes the oscillator, the clocking system and how to configure them. The oscillators, PLLs and clock generators supply clocking to the system, including the CPU and peripherals. The components are configured to work together to produce the required clocks for the system. The following features are covered in this section:

- Clock Generator (CLKGENn)
- PLL Generator (PLLGENn)
- Clock sources including FRC, Primary, Auxiliary, BFRC.
- Fail Safe Clock monitor (FSCM)
- REFO Clock
- Power-save mode

12.1 Device-Specific Information

Table 12-1. Oscillator Summary

Number of Clock Generators	Number of PLLs
13	2

Table 12-2. Clock Generator Input and Destination

Clock Generator	CLKGEN Input Source	Module Clock Destination
CLKGEN1	All Sources	System Clock and Peripheral Clock
CLKGEN2	FRC Only	FRC
CLKGEN3	BFRC Only	BFRC
CLKGEN4	All Sources	RAM BIST and NVM BIST
CLKGEN5	All Sources	PWM
CLKGEN6	All Sources	ADC
CLKGEN7	All Sources	PDM DAC
CLKGEN8	All Sources	UART
CLKGEN9	All Sources	SPI
CLKGEN10	All Sources	PTG
CLKGEN11	All Sources	BiSS
CLKGEN12	All Sources	CCP and REFO1
CLKGEN13	All Sources	CLC, IOIM and REFO2
Note: CLKGEN 1 is always ON.		

Table 12-3. Clock Generator Input Clock Selections

Value	Description
1111-1011	Reserved
1010	REFI1 – Device REFI1 pin through PPS
1001	REFI2 – Device REFI2 pin through PPS
1000	PLL2 VCO DIV output
0111	PLL1 VCO DIV output
0110	PLL2 F_{OUT} output
0101	PLL1 F_{OUT} output
0100	LPRC as BFRC/244
0011	POSC – Primary crystal oscillator (4-32 MHz)

.....continued	
Value	**Description**
0010	BFRC – Internal Backup 8 MHz RC oscillator
0001	FRC – Internal 8 MHz RC oscillator
0000	ICSP clock (PGC)

Table 12-4. PLL Input Clock Selections

Value	Description
1111-1011	Reserved
1010	REFI1 – Device REFI1 pin through PPS
1001	REFI2 – Device REFI2 pin through PPS
1000-0100	Reserved
0011	POSC – Primary crystal oscillator (4-32 MHz)
0010	BFRC – Internal Backup 8 MHz RC oscillator
0001	FRC – Internal 8 MHz RC oscillator
0000	ICSP clock (PGC)

Table 12-5. Clock Monitor CNTSEL/WINSEL Input Clock Selections

Value	Description
0x11010	REFI2 - Device REFI1 pin through PPS
0x11001	REFI1 - Device REFI2 pin through PPS
0x11000-0x10101	Reserved
0x10100	BRFC/244
0x10011	POSC - Primary crystal oscillator (4-32 MHz)
0x10010	BFRC - Internal Backup 8 MHz RC oscillator
0x10001	FRC - Internal 8 MHz RC oscillator
0x10000	ISCP clock (PGC)
0x01111	Reserved
0x01110	PLL2 VCO DIV output
0x01101	PLL2 F_{OUT} output
0x01100	PLL1 VCO DIV output
0x01011	PLL1 F_{OUT} output
0x01010	CLKGEN11
0x01001	CLKGEN10
0x01000	CLKGEN9
0x00111	CLKGEN8
0x00110	CLKGEN7
0x00101	CLKGEN6
0x00100	CLKGEN5
0x00011	CLKGEN4
0x00010	CLKGEN3
0x00001	CLKGEN2
0x00000	CLKGEN1

12.2 Architectural Overview

The oscillator module is based on a number of clock generators (CLKGENs) and PLL generators (PLLGENs). The CLKGENs are preassigned to the CPU and peripherals. Some of the CLKGENs are shared by more than one consumer. Each clock generator selects a clock source from the available oscillators or PLLs and passes it to a selectable divider circuit. The CLKGENs utilize a change request

mechanism to prevent unintentional modifications. After the settings are written, a change request bit is set, and when clear, the operation is complete. In the event of a clock failure, the backup clock source is then used by the CLKGEN.

The Oscillator has the following main functions/modules:

- Multiple Clock Generators, Each With:
  - Selectable clock source
  - A backup clock source
  - Fail safe clock monitors
- Multiple PLLs, Each With:
-
  - Selectable clock source
  - A backup clock source
  - PLL FOUT primary divider output
  - PLL VCO secondary divider output
  - Fail safe clock monitors
- Clock Monitor Module That Compares Clock Against Reference Clock:
  - Clock failure detection
  - Clock frequency drift detection
  - Selectable threshold limits for warning and/or failing
  - Fault injection capability
  - Frequency, pulse width and duty cycle measurements

A high level block diagram of the dsPIC33A oscillator system is shown in Figure 12-1.

Figure 12-1. Oscillator Module Block Diagram

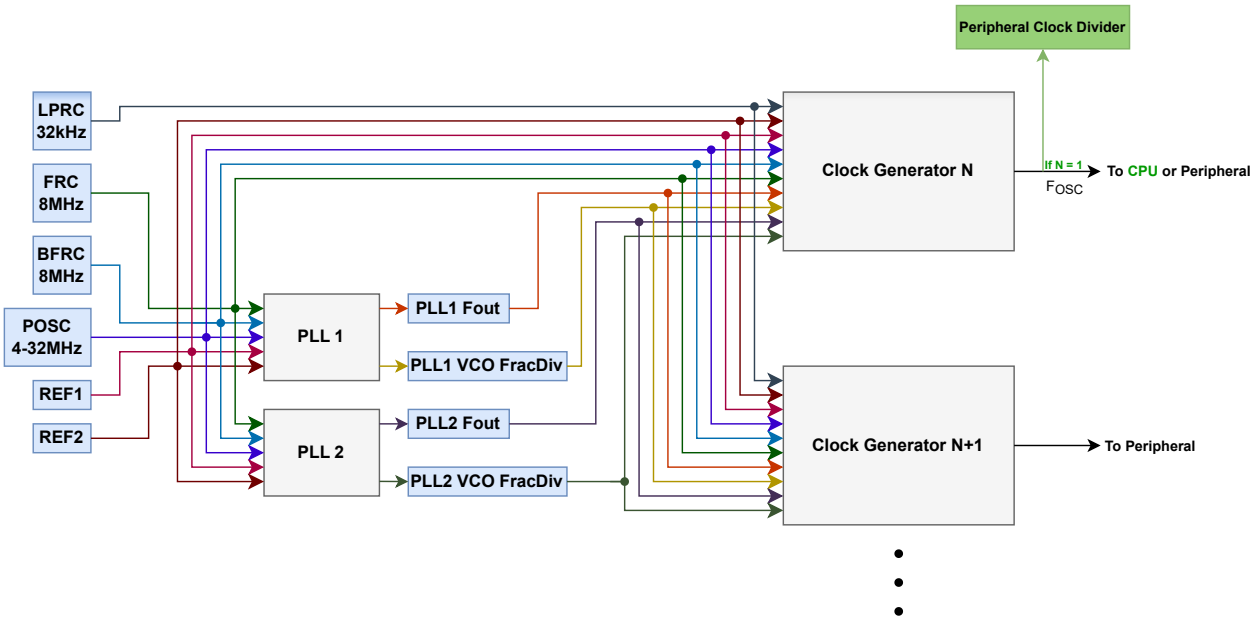
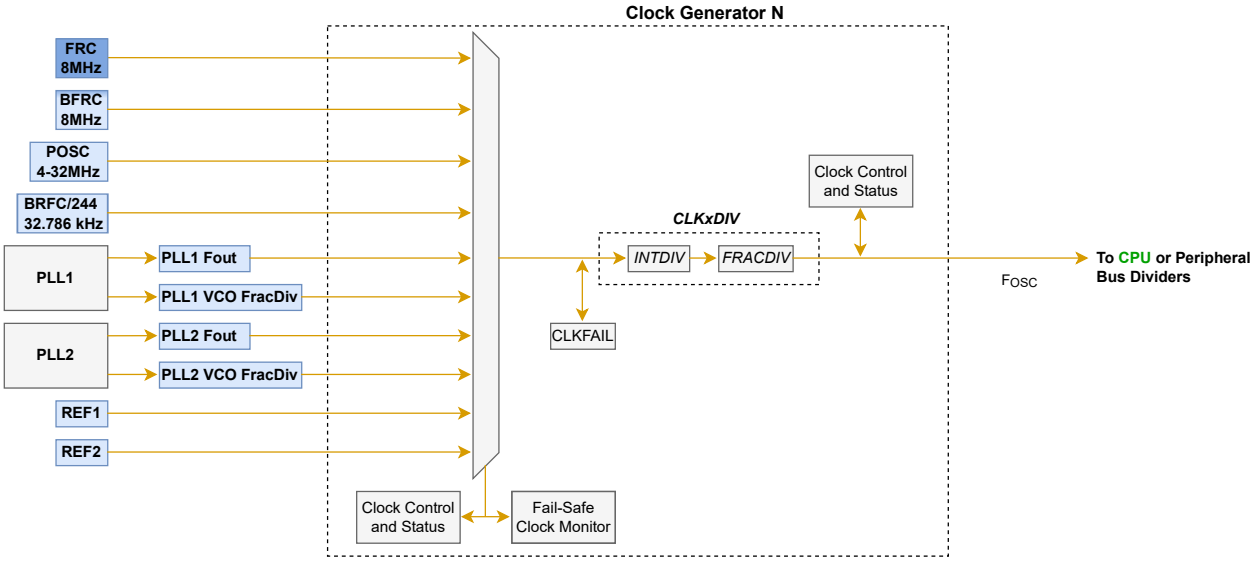


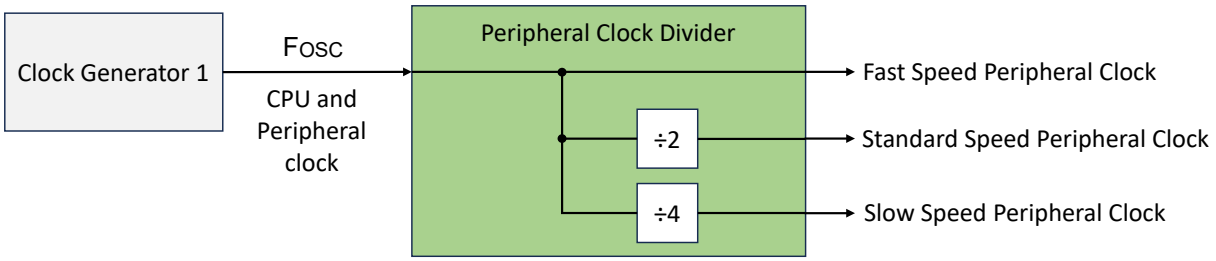
Figure 12-2. Clock Generator



12.2.1 Peripheral Clock Divider

To support multiple peripheral bus speeds, the CPU clock from CLKGEN1 is further divided to provide two additional clk rates: standard and slow. The fast-speed peripheral clock is the same as the CPU, the standard-speed peripheral clock is half, and the slow-speed peripheral clock is one-quarter speed of the peripheral clock, as shown in Figure 12-3.

Figure 12-3. Peripheral Clock Divider Block Diagram



12.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3100	OSCCTRL	31:24	DIAGLOCK							
		23:16	CLKLOCK							
		15:8	PLL2RDY	PLL1RDY	LPRCRDY			POSCRDY	BFRCRDY	FRCRDY
		7:0	PLL2EN	PLL1EN	LPRCEN			POSCEN	BFRCEN	FRCEN
0x3104	OSCCFG	31:24								
		23:16							FRCLPWR[1:0]	
		15:8								
		7:0			POSCIOFNC	PXTALCFG[2]	PXTALCFG[1:0]		POSCMD[1:0]	
0x3108	CLKFAIL	31:24	SCSMSMCH						PLLFAIL2	PLLFAIL1
		23:16								
		15:8				CLKFAIL13	CLKFAIL12	CLKFAIL11	CLKFAIL10	CLKFAIL9
		7:0	CLKFAIL8	CLKFAIL7	CLKFAIL6	CLKFAIL5	CLKFAIL4	CLKFAIL3	CLKFAIL2	CLKFAIL1
0x3108	SCSFAIL	31:24							PLLSCS2	PLLSCS1
		23:16								
		15:8				CLKSCS13	CLKSCS12	CLKSCS11	CLKSCS10	CLKSCS9
		7:0	CLKSCS8	CLKSCS7	CLKSCS6	CLKSCS5		CLKSCS3	CLKSCS2	CLKSCS1
0x310C ... 0x3117	Reserved									
0x3118	CLK1CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x311C	CLK1DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3120	CLK2CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x3124	CLK2DIV	31:24	Reserved[31:24]							
		23:16	Reserved[23:16]							
		15:8	Reserved[15:8]							
		7:0	Reserved[7:0]							
0x3128	CLK3CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x312C	CLK3DIV	31:24	Reserved[31:24]							
		23:16	Reserved[23:16]							
		15:8	Reserved[15:8]							
		7:0	Reserved[7:0]							
0x3130	CLK4CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x3134	CLK4DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3138	CLK5CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x313C	CLK5DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3140	CLK6CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x3144	CLK6DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3148	CLK7CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x314C	CLK7DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3150	CLK8CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x3154	CLK8DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3158	CLK9CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x315C	CLK9DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3160	CLK10CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x3164	CLK10DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3168	CLK11CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x316C	CLK11DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3170	CLK12CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x3174	CLK12DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3178	CLK13CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x317C	CLK13DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3180	PLL1CON	31:24	CLKRDY	PLLSWEN	RIS	FOUTSWEN	EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x3184	PLL1DIV	31:24					PLLPRE[3:0]			
		23:16								
		15:8	PLLFBDIV[7:0]							
		7:0			POSTDIV1[2:0]			POSTDIV2[2:0]		
0x3188	VCO1DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8								
		7:0								
0x318C	PLL2CON	31:24	CLKRDY	PLLSWEN	RIS	FOUTSWEN	EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0					COSC[3:0]			
0x3190	PLL2DIV	31:24					PLLPRE[3:0]			
		23:16								
		15:8	PLLFBDIV[7:0]							
		7:0			POSTDIV1[2:0]			POSTDIV2[2:0]		
0x3194	VCO2DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8								
		7:0								
0x3198	RCON	31:24								
		23:16			VREG3R	VREG2R	VREG1R			
		15:8							CM	
		7:0	EXTR	SWR		WDTO	SLEEP	IDLE	BOR	POR
0x319C	CLKDIAG	31:24	FLTIJEN	STOPPLL2	STOPPLL1	GENSEL[4:0]				
		23:16					SCSFLTDATA[3:0]			
		15:8				STOPGEN13	STOPGEN12	STOPGEN11	STOPGEN10	STOPGEN9
		7:0	STOPGEN8	STOPGEN7	STOPGEN6	STOPGEN5	STOPGEN4	STOPGEN3	STOPGEN2	STOPGEN1
0x31A0 ... 0x31FF	Reserved									
0x3200	CM1CON	31:24								
		23:16								
		15:8	ON		SIDL	SLPEN				
		7:0			CNTDIV[1:0]		FLTINJ[1:0]			WIDTH
0x3204	CM1STAT	31:24								
		23:16								
		15:8					HWT	LWT	HFT	LFT
		7:0						TRIG	SATD	BUFV
0x3208	CM1WINPR	31:24	WINPR[31:24]							
		23:16	WINPR[23:16]							
		15:8	WINPR[15:8]							
		7:0	WINPR[7:0]							
0x320C	CM1SEL	31:24								
		23:16								
		15:8	CNTSEL[7:0]							
		7:0	WINSEL[7:0]							
0x3210	CM1BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3214	CM1SAT	31:24	SAT[31:24]							
		23:16	SAT[23:16]							
		15:8	SAT[15:8]							
		7:0	SAT[7:0]							
0x3218	CM1HFAIL	31:24	HFAIL[31:24]							
		23:16	HFAIL[23:16]							
		15:8	HFAIL[15:8]							
		7:0	HFAIL[7:0]							
0x321C	CM1LFAIL	31:24	LFAIL[31:24]							
		23:16	LFAIL[23:16]							
		15:8	LFAIL[15:8]							
		7:0	LFAIL[7:0]							
0x3220	CM1HWARN	31:24	HWARN[31:24]							
		23:16	HWARN[23:16]							
		15:8	HWARN[15:8]							
		7:0	HWARN[7:0]							
0x3224	CM1LWARN	31:24	LWARN[31:24]							
		23:16	LWARN[23:16]							
		15:8	LWARN[15:8]							
		7:0	LWARN[7:0]							
0x3228 ... 0x322F	Reserved									
0x3230	CM2CON	31:24								
		23:16								
		15:8	ON		SIDL	SLPEN				
		7:0			CNTDIV[1:0]		FLTINJ[1:0]			WIDTH
0x3234	CM2STAT	31:24								
		23:16								
		15:8					HWT	LWT	HFT	LFT
		7:0						TRIG	SATD	BUFV
0x3238	CM2WINPR	31:24	WINPR[31:24]							
		23:16	WINPR[23:16]							
		15:8	WINPR[15:8]							
		7:0	WINPR[7:0]							
0x323C	CM2SEL	31:24								
		23:16								
		15:8	CNTSEL[7:0]							
		7:0	WINSEL[7:0]							
0x3240	CM2BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x3244	CM2SAT	31:24	SAT[31:24]							
		23:16	SAT[23:16]							
		15:8	SAT[15:8]							
		7:0	SAT[7:0]							
0x3248	CM2HFAIL	31:24	HFAIL[31:24]							
		23:16	HFAIL[23:16]							
		15:8	HFAIL[15:8]							
		7:0	HFAIL[7:0]							
0x324C	CM2LFAIL	31:24	LFAIL[31:24]							
		23:16	LFAIL[23:16]							
		15:8	LFAIL[15:8]							
		7:0	LFAIL[7:0]							
0x3250	CM2HWARN	31:24	HWARN[31:24]							
		23:16	HWARN[23:16]							
		15:8	HWARN[15:8]							
		7:0	HWARN[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3254	CM2LWARN	31:24	LWARN[31:24]							
		23:16	LWARN[23:16]							
		15:8	LWARN[15:8]							
		7:0	LWARN[7:0]							
0x3258 ... 0x325F	Reserved									
0x3260	CM3CON	31:24								
		23:16								
		15:8	ON		SIDL	SLPEN				
		7:0			CNTDIV[1:0]		FLTINJ[1:0]			WIDTH
0x3264	CM3STAT	31:24								
		23:16								
		15:8					HWT	LWT	HFT	LFT
		7:0						TRIG	SATD	BUFV
0x3268	CM3WINPR	31:24	WINPR[31:24]							
		23:16	WINPR[23:16]							
		15:8	WINPR[15:8]							
		7:0	WINPR[7:0]							
0x326C	CM3SEL	31:24								
		23:16								
		15:8	CNTSEL[7:0]							
		7:0	WINSEL[7:0]							
0x3270	CM3BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x3274	CM3SAT	31:24	SAT[31:24]							
		23:16	SAT[23:16]							
		15:8	SAT[15:8]							
		7:0	SAT[7:0]							
0x3278	CM3HFAIL	31:24	HFAIL[31:24]							
		23:16	HFAIL[23:16]							
		15:8	HFAIL[15:8]							
		7:0	HFAIL[7:0]							
0x327C	CM3LFAIL	31:24	LFAIL[31:24]							
		23:16	LFAIL[23:16]							
		15:8	LFAIL[15:8]							
		7:0	LFAIL[7:0]							
0x3280	CM3HWARN	31:24	HWARN[31:24]							
		23:16	HWARN[23:16]							
		15:8	HWARN[15:8]							
		7:0	HWARN[7:0]							
0x3284	CM3LWARN	31:24	LWARN[31:24]							
		23:16	LWARN[23:16]							
		15:8	LWARN[15:8]							
		7:0	LWARN[7:0]							
0x3288 ... 0x328F	Reserved									
0x3290	CM4CON	31:24								
		23:16								
		15:8	ON		SIDL	SLPEN				
		7:0			CNTDIV[1:0]		FLTINJ[1:0]			WIDTH
0x3294	CM4STAT	31:24								
		23:16								
		15:8					HWT	LWT	HFT	LFT
		7:0						TRIG	SATD	BUFV

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3298	CM4WINPR	31:24	WINPR[31:24]							
		23:16	WINPR[23:16]							
		15:8	WINPR[15:8]							
		7:0	WINPR[7:0]							
0x329C	CM4SEL	31:24								
		23:16								
		15:8	CNTSEL[7:0]							
		7:0	WINSEL[7:0]							
0x32A0	CM4BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x32A4	CM4SAT	31:24	SAT[31:24]							
		23:16	SAT[23:16]							
		15:8	SAT[15:8]							
		7:0	SAT[7:0]							
0x32A8	CM4HFAIL	31:24	HFAIL[31:24]							
		23:16	HFAIL[23:16]							
		15:8	HFAIL[15:8]							
		7:0	HFAIL[7:0]							
0x32AC	CM4LFAIL	31:24	LFAIL[31:24]							
		23:16	LFAIL[23:16]							
		15:8	LFAIL[15:8]							
		7:0	LFAIL[7:0]							
0x32B0	CM4HWARN	31:24	HWARN[31:24]							
		23:16	HWARN[23:16]							
		15:8	HWARN[15:8]							
		7:0	HWARN[7:0]							
0x32B4	CM4LWARN	31:24	LWARN[31:24]							
		23:16	LWARN[23:16]							
		15:8	LWARN[15:8]							
		7:0	LWARN[7:0]							
0x32B8 ... 0x32BF	Reserved									
0x32C0	FRCTUN	31:24								
		23:16								
		15:8								
		7:0			TUN[5:0]					
0x32C4	BFRCTUN	31:24								
		23:16								
		15:8								
		7:0			TUN[5:0]					

12.3.1 System Clock Control Register

Name: OSCCTRL
Offset: 0x3100

Legend: HS = Hardware Settable bit, HC = Hardware Clearable bit, R = Readable bit

Bit	31	30	29	28	27	26	25	24
	DIAGLOCK							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
	CLKLOCK							
Access	R/W							
Reset	0							

Bit	15	14	13	12	11	10	9	8
	PLL2RDY	PLL1RDY	LPRCRDY			POSCRDY	BFRCRDY	FRCRDY
Access	HS/HC/R	HS/HC/R	HS/HC/R			HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	x			0	1	1

Bit	7	6	5	4	3	2	1	0
	PLL2EN	PLL1EN	LPRCEN			POSCEN	BFRCEN	FRCEN
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bit 31 – DIAGLOCK Clock Diagnostic Lock Enable bit

Value	Description
1	Clock diagnostic register (CLK_DIAG) is locked
0	Clock diagnostic register is not locked, configurations may be modified

Bit 23 – CLKLOCK Clock Lock Enable bit

Value	Description
1	Clock registers are locked
0	Clock registers are not locked, configurations may be modified

Bit 15 – PLL2RDY PLL2 Ready Status bit

Value	Description
1	PLL2 Ready
0	PLL2 Not ready

Bit 14 – PLL1RDY PLL1 Ready Status bit

Value	Description
1	PLL1 Ready
0	PLL1 Not Ready

Bit 13 – LPRCRDY LPRC Ready Status bit

Value	Description
1	Low power 32 KHz RC oscillator ready
0	Low power 32 KHz RC oscillator not ready

Bit 10 – POSCRDY Primary Crystal Oscillator Ready Status bit

Value	Description
1	Primary crystal/resonator oscillator ready
0	Primary crystal/resonator oscillator not ready

Bit 9 – BFRCRDY Backup FRC Ready Status bit

Value	Description
1	Backup FRC oscillator ready
0	Backup FRC oscillator not ready

Bit 8 – FRCRDY 8 MHz FRC Ready Status bit

Value	Description
1	FRC oscillator ready
0	FRC oscillator not ready

Bit 7 – PLL2EN PLL2 Enable bit

Value	Description
1	Enable PLL2
0	Disable PLL2

Bit 6 – PLL1EN PLL1 Enable bit

Value	Description
1	Enable PLL1
0	Disable PLL1

Bit 5 – LPRCEN LPRC Clock Enable bit

Value	Description
1	Enable low power 32 KHz RC oscillator
0	Disable low power 32 KHz RC oscillator

Bit 2 – POSCEN Primary Crystal Clock Enable bit

Value	Description
1	Enable primary crystal/resonator oscillator
0	Disable primary crystal/resonator oscillator

Bit 1 – BFRCEN Backup FRC Clock Enable bit

Value	Description
1	Enable backup FRC oscillator
0	Disable backup FRC oscillator

Bit 0 – FRCEN 8 MHz FRC Clock Enable bit

Value	Description
1	Enable FRC oscillator
0	Disable FRC oscillator

12.3.2 Oscillator Configuration Register

Name: OSCCFG
Offset: 0x3104

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
							FRCLPWR[1:0]	
Access							R/W	R/W
Reset							0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			POSCIOFNC	PXTALCFG[2]	PXTALCFG[1:0]		POSCMD[1:0]	
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	1	1

Bits 17:16 – FRCLPWR[1:0] FRC Low-Power Mode Enable bits

Value	Description
11	Reserved
10	Reserved
01	FRC Low-Power mode
00	FRC Standard-Power mode

Bit 5 – POSCIOFNC Primary CLKO Enable Configuration bit

Value	Description
1	CLKO output signal active on the OSCO pin; primary oscillator must be disabled or configured for the External Clock mode
0	CLKO output disabled

Bit 4 – PXTALCFG[2] Kick-Starter Programmability for Primary Oscillator bit

Value	Description
1	Boost the kick start
0	Default kick start

Bits 3:2 – PXTALCFG[1:0] Current Gain Programmability for Oscillator (Output Drive) bits G3>G2>G1>G0

Value	Description
11	Gain is G3 (use for 24-32 MHz crystals)
10	Gain is G2 (use for 16-24 MHz crystals)
01	Gain is G1 (use for 8-16 MHz crystals)
00	Gain is G0 (use for 4-8 MHz crystals)

Bits 1:0 – POSCMD[1:0] Primary Oscillator Selection bit

Value	Description
11	OFF (use pin as GPIO)
10	HS Oscillator mode selected (10 MHz-32 MHz)
01	MS Oscillator mode selected (3.5 MHz-10 MHz)
00	External clock

12.3.3 Reference Clock Fail Status Register

Name: CLKFAIL
Offset: 0x3108

Legend: HS = Hardware Settable bit, R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	SCSMSMCH						PLLFAIL2	PLLFAIL1
Access	R/W						R/W	R/W
Reset	0						0	0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				CLKFAIL13	CLKFAIL12	CLKFAIL11	CLKFAIL10	CLKFAIL9
Access				HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLKFAIL8	CLKFAIL7	CLKFAIL6	CLKFAIL5	CLKFAIL4	CLKFAIL3	CLKFAIL2	CLKFAIL1
Access	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – SCSMSMCH Source Clock Select Mismatch Indicator bit

Value	Description
1	Clock failure occurred due to SCS integrity check mismatch (or FLTINJ). Check SCSFAIL register for details
0	Clock failure occurred due to non-integrity check related failure

Bit 25 – PLLFAIL2 PLL #2 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the PLL generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 24 – PLLFAIL1 PLL #1 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the PLL generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 12 – CLKFAIL13 Clock Generator #13 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 11 – CLKFAIL12 Clock Generator #12 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 10 – CLKFAIL11 Clock Generator #11 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 9 – CLKFAIL10 Clock Generator #10 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 8 – CLKFAIL9 Clock Generator #9 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 7 – CLKFAIL8 Clock Generator #8 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 6 – CLKFAIL7 Clock Generator #7 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 5 – CLKFAIL6 Clock Generator #6 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 4 – CLKFAIL5 Clock Generator #5 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 3 – CLKFAIL4 Clock Generator #4 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 2 – CLKFAIL3 Clock Generator #3 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 1 – CLKFAIL2 Clock Generator #2 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to selected backup reference clock
0	Selected reference clock is functioning

Bit 0 – CLKFAIL1 Clock Generator #1 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to selected backup reference clock
0	Selected reference clock is functioning

12.3.4 Source Clock Selection Fail Status Register

Name: SCSFAIL
Offset: 0x3108

Legend: HS = Hardware Settable bit, R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
							PLLSCS2	PLLSCS1
Access							HS/R/W	HS/R/W
Reset							0	0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				CLKSCS13	CLKSCS12	CLKSCS11	CLKSCS10	CLKSCS9
Access				HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLKSCS8	CLKSCS7	CLKSCS6	CLKSCS5		CLKSCS3	CLKSCS2	CLKSCS1
Access	HS/R/W	HS/R/W	HS/R/W	HS/R/W		HS/R/W	HS/R/W	HS/R/W
Reset	0	0	0	0		0	0	0

Bit 25 – PLLSCS2 PLL #2 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct
0	Circuit is correctly selecting the chosen source reference clock

Bit 24 – PLLSCS1 PLL #1 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct
0	Circuit is correctly selecting the chosen source reference clock

Bit 12 – CLKSCS13 Clock Generator #13 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it can not correct
0	Circuit is correctly selecting the chosen source reference clock

Bit 11 – CLKSCS12 Clock Generator #12 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it can not correct
0	Circuit is correctly selecting the chosen source reference clock

Bit 10 – CLKSCS11 Clock Generator #11 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it can not correct
0	Circuit is correctly selecting the chosen source reference clock

Bit 9 – CLKSCS10 Clock Generator #10 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it can not correct
0	Circuit is correctly selecting the chosen source reference clock

Bit 8 – CLKSCS9 Clock Generator #9 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it can not correct
0	Circuit is correctly selecting the chosen source reference clock

Bit 7 – CLKSCS8 Clock Generator #8 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it can not correct
0	Circuit is correctly selecting the chosen source reference clock

Bit 6 – CLKSCS7 Clock Generator #7 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it can not correct
0	Circuit is correctly selecting the chosen source reference clock

Bit 5 – CLKSCS6 Clock Generator #6 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it can not correct
0	Circuit is correctly selecting the chosen source reference clock

Bit 4 – CLKSCS5 Clock Generator #5 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it can not correct
0	Circuit is correctly selecting the chosen source reference clock

Bit 2 – CLKSCS3 Clock Generator #3 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it can not correct
0	Circuit is correctly selecting the chosen source reference clock

Bit 1 – CLKSCS2 Clock Generator #2 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it can not correct
0	Circuit is correctly selecting the chosen source reference clock

Bit 0 – CLKSCS1 Clock Generator #1 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it can not correct
0	Circuit is correctly selecting the chosen source reference clock

12.3.5 Clock Generator Control Register

Name: CLKxCON
Offset: 0x3118, 0x3120, 0x3128, 0x3130, 0x3138, 0x3140, 0x3148, 0x3150, 0x3158, 0x3160, 0x3168, 0x3170, 0x3178

Legend: HS = Hardware Settable bit, R = Readable bit, W = Writable bit

Note:

1. CLKGEN 1, 2 and 3 have default value of '1'

Bit	31	30	29	28	27	26	25	24
	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
Access	R/HS/HC		R/W		R/W	R/W	R/W	R/W
Reset	1		0		0	0	0	0

Bit	23	22	21	20	19	18	17	16
	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	1	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	OE	NOSC[3:0]			
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
					COSC[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	1

Bit 31 – CLKRDY Output Clock is Ready bit
This bit cannot be reset.

Value	Description
1	Clock output is ready
0	Clock output is not ready, maybe due to source clock not ready, source selection change is in progress, divider change is in progress, or a clock failure has been detected and the backup clock has not yet occurred.

Bit 29 – RIS Run In Sleep bit

Value	Description
1	Clock Generator block will continue to operate if SLEEP mode is entered
0	Clock Generator block will stop when SLEEP mode is entered

Bit 27 – EXTCFEN External Clock Fail Event Enable bit

Value	Description
1	External clock fail detection is enabled
0	External clock fail detection is disabled

Bits 26:24 – EXTCFSEL[2:0] External Clock Fail Event Select bits(1)

Value	Description
[0]	External clock fail detection module #1
[1]	External clock fail detection module #2
[2]	External clock fail detection module #3

Value	Description
[3]	External clock fail detection module #4
[4]	External clock fail detection module #5
[5]	External clock fail detection module #6
[6]	External clock fail detection module #7
[7]	External clock fail detection module #8

Bit 23 – OSWEN Oscillator Switch Enable bit

Value	Description
1	Request oscillator switch to selection specified by NOSC[3:0] bits
0	Oscillator switch is complete

Bit 22 – DIVSWEN Clock RODIV/ROTRIM Switch Enable bit

Value	Description
1	Clock Divider Switching currently in progress
0	Clock Divider Switch has completed

Bit 20 – FSCMEN Fail-Safe Clock Monitor Enable bit

Value	Description
1	Fail-Safe Clock Monitor is enabled
0	FSCM is disabled

Bits 19:16 – BOSC[3:0] Backup Reference Clock Select bits
See Table 12-3.

Bit 15 – ON Enable Clock Generator bit(1)

Value	Description
1	Clock Generator is enabled
0	Clock Generator is disabled

Bit 13 – SIDL Stop in Idle bit

Value	Description
1	Clock Generator block will stop when IDLE mode is entered
0	Clock Generator block will continue to operate when IDLE mode is entered

Bit 12 – OE Output Enable bit

Value	Description
1	Clock output is enabled to be output on a device pin
0	Clock output on a pin is disabled

Bits 11:8 – NOSC[3:0] New Reference Clock Select bits
See Table 12-3.

Bits 3:0 – COSC[3:0] Current Reference Clock Selection bits
See Table 12-3

12.3.6 Clock Generator Divider Register

Name: CLKxDIV
Offset: 0x311C, 0x3134, 0x313C, 0x3144, 0x314C, 0x3154, 0x315C, 0x3164, 0x316C, 0x3174, 0x317C

Note: The FRC/BFRC variants of the CLKGEN2/3 do not implement the clock divider; the divider ratio for the FRC CLKFEN is fixed at 1x.

Bit	31	30	29	28	27	26	25	24
		INTDIV[14:8]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INTDIV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FRACDIV[8:1]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FRACDIV[0]							
Access	R/W							
Reset	0							

Bits 30:16 – INTDIV[14:0] Integer Divider bit
Integer Divider (INTDIV):
Number of Source Clocks in each 1/2 period of the Divided Clock.
Period Ex:
Divided Clock Period = [Source Clock Period] * INT * 2
Frequency Ex:
111111111111111 = Clock Generator Divided Clock = Source Clock divided by 65,534 (32,767 *2)
111111111111110 = Clock Generator Divided Clock Clock = Source Clock divided by 65,532 (32,766 *2)

Value	Description
000000000000011	Clock Generator Divided Clock = Source Clock divided by 6 (3*2)
000000000000010	Clock Generator Divided Clock = Source Clock divided by 4 (2*2)
000000000000001	Clock Generator Divided Clock = Source Clock divided by 2 (1*2)
000000000000000	Clock Generator Divided Clock = Source Clock (no divider)

Bits 15:7 – FRACDIV[8:0] Fractional Divider bit
Number of source clocks periods added over 512 source clocks, as equally as possible to each half period of the output clock.
Provides fractional additive value for 1/2 period of the output clock

Value	Description
1111_1111_0	510/512 (0.99609375) divisor added to Integer value
1111_1111_1	511/512 (0.998046875) divisor added to Integer value
100000000	256/512 (0.5000) divisor added to Integer value
0000_0001_0	2/512 (0.00390625) divisor added to Integer value
0000_0000_1	1/512 (0.001953125) divisor added to Integer value
0000_0000_0	0/512 (0.0) divisor added to Integer value

12.3.7 Clock Generator Divider Register

Name: CLK2DIV
Offset: 0x3124

Note: The FRC/BFRC variants of the CLKGEN2/3 do not implement the clock divider; the divider ratio for the FRC CLKFEN is fixed at 1x. The associated FRACDIV macro has an active register that contains the current value of INTDIV[14:0] and FRACDIV[8:0]. The CLKxDIV contents are transferred into the FRACDIV macro active registers after the associated DIVSWEN bit is set. The DIVSWEN bit is cleared when the transfer is completed.

Bit	31	30	29	28	27	26	25	24
	Reserved[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	Reserved[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	Reserved[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	Reserved[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – Reserved[31:0]

12.3.8 Clock Generator Divider Register

Name: CLK3DIV
Offset: 0x312C

Note: The FRC/BFRC variants of the CLKGEN2/3 do not implement the clock divider; the divider ratio for the FRC CLKFEN is fixed at 1x. The associated FRACDIV macro has an active register that contains the current value of INTDIV[14:0] and FRACDIV[8:0]. The CLKxDIV contents are transferred into the FRACDIV macro active registers after the associated DIVSWEN bit is set. The DIVSWEN bit is cleared when the transfer is completed.

Bit	31	30	29	28	27	26	25	24
	Reserved[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	Reserved[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	Reserved[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	Reserved[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – Reserved[31:0]

12.3.9 PLL Control Register

Name: PLLxCON
Offset: 0x3180, 0x318C

Legend: HS = Hardware Settable bit, C = Clearable bit, R = Readable bit, W = Writable bit

Note:
1. The number of external clock fail detection modules is device dependent.

Bit	31	30	29	28	27	26	25	24
	CLKRDY	PLLSWEN	RIS	FOUTSWEN	EXTCFEN	EXTCFSEL[2:0]		
Access	R/HS/HC	R/S/HC	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	OE	NOSC[3:0]			
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
					COSC[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 31 – CLKRDY Output Clock is Ready bit
This bit cannot be reset.

Value	Description
1	Clock output is ready
0	Clock output is not ready, may be due to source clock not ready, source selection change is in progress, divider change is in progress, or a clock failure has been detected and the backup clock has not yet occurred.

Bit 30 – PLLSWEN PLL Input and Feedback Divider Switch Enabled bit

Value	Description
1	Enable PLL input and feedback divider update
0	Divider Switch has completed

Bit 29 – RIS Run In Sleep bit

Value	Description
1	PLL block will continue to operate if SLEEP mode is entered
0	PLL block will stop when SLEEP mode is entered

Bit 28 – FOUTSWEN Clock Divider Switch Enabled bit

Value	Description
1	Enable PLL output divider update
0	Divider switch has completed

Bit 27 – EXTCFEN External Clock Fail Event Enable bit

Value	Description
1	External clock fail detection is enabled
0	External clock fail detection is disabled

Bits 26:24 – EXTCFSEL[2:0] External Clock Fail Event Select bits(1)

Value	Description
0011	External clock fail detection module #4
0010	External clock fail detection module #3
0001	
0000	External clock fail detection module #1

Bit 23 – OSWEN Oscillator Switch Enable bit

Value	Description
1	Request oscillator switch to selection specified by NOSC[3:0] bits
0	Oscillator switch is complete

Bit 22 – DIVSWEN Clock RODIV/ROTRIM Switch Enable bit

Value	Description
1	Clock Divider Switching currently in progress
0	Clock Divider Switching has completed

Bit 20 – FSCMEN Fail-Safe Clock Monitor Enable bit

Value	Description
1	Fail-Safe Clock Monitor is enabled
0	Fail-Safe Clock Monitor is disabled

Bits 19:16 – BOSC[3:0] Backup Reference Clock Select bits
See Table 12-4.

Bit 15 – ON Enable PLL Generator bit

Value	Description
1	PLL Generator is enabled
0	PLL Generator is disabled

Bit 13 – SIDL Stop in Idle bit

Value	Description
1	Clock Generator block will stop when IDLE mode is entered
0	Clock Generator block will continue to operate when IDLE mode is entered

Bit 12 – OE Output Enable bit

Value	Description
1	Clock output is enabled to be output on a device pin
0	Clock output on a pin is disabled

Bits 11:8 – NOSC[3:0] New Reference Clock Select bit
See Table 12-4.

Bits 3:0 – COSC[3:0] New Reference Clock Select bit
See Table 12-4.

12.3.10 PLL Divider Register

Name: PLLxDIV
Offset: 0x3184, 0x3190

Bit	31	30	29	28	27	26	25	24
					PLLPRE[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	1

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PLLFBDIV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	0	0	1	0	0	0

Bit	7	6	5	4	3	2	1	0
			POSTDIV1[2:0]			POSTDIV2[2:0]		
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			1	1	1	0	0	1

Bits 27:24 – PLLPRE[3:0] PLLx Reference Clock Prescale bits

Value	Description
1111	16x divide
1110	15x divide
. . .	
0010	2x divide
0001	1x divide
0000	undefined, not allowed

Bits 15:8 – PLLFBDIV[7:0] PLLx Feedback Divider bit

Value	Description
11111111	256x divide
11111110	255x divide
. . .	
00000010	2x divide
00000001	1x divide
00000000	undefined, not allowed

Bits 5:3 – POSTDIV1[2:0] PLLx Post Divider #1 bit

Value	Description
111	7x divide
110	6x divide
. . .	
010	2x divide
001	1x divide
000	undefined, not allowed

Bits 2:0 – POSTDIV2[2:0] PLLx Post Divider #2 bit

Value	Description
111	7x divide
110	6x divide
...	
010	2x divide
001	1x divide
000	undefined, not allowed

12.3.11 PLL VCO Divider Register

Name: VCOxDIV
Offset: 0x3188, 0x3194

Bit	31	30	29	28	27	26	25	24
		INTDIV[14:8]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INTDIV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 30:16 – INTDIV[14:0] PLL VCO Integer Divider bit
Integer Divider (INTDIV):
Number of Source Clocks in each 1/2 period of the Divided Clock.
Period Ex:
Divided Clock Period = [Source Clock Period] * INT * 2
Frequency Ex:
111111111111111 = Clock Generator Divided Clock = Source Clock divided by 65,534 (32,767 *2)

Value	Description
000000000000011	Clock Generator Divided Clock = Source Clock divided by 6 (3*2)
000000000000010	Clock Generator Divided Clock = Source Clock divided by 4 (2*2)
000000000000001	Clock Generator Divided Clock = Source Clock divided by 2 (1*2)
000000000000000	Clock Generator Divided Clock = Source Clock (no divider)

12.3.12 Reset Control Register

Name: RCON
Offset: 0x3198

Legend: R = Readable bit, C = Clearable bit, HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			VREG3R	VREG2R	VREG1R			
Access			R/C/HS	R/C/HS	R/C/HS			
Reset			0	0	0			

Bit	15	14	13	12	11	10	9	8
							CM	
Access							R/C/HS	
Reset							0	

Bit	7	6	5	4	3	2	1	0
	EXTR	SWR		WDTO	SLEEP	IDLE	BOR	POR
Access	R/C/HS	R/C/HS		R/C/HS	R/C/HS	R/C/HS	R/C/HS	R/C/HS
Reset	0	0		0	0	0	1	1

Bit 21 – VREG3R Voltage Domain 3 Regulation Loss Flag bit

Value	Description
1	Voltage Domain 3 lost voltage regulation
0	Voltage Regulation in Domain 3 has not been lost

Bit 20 – VREG2R Voltage Domain 2 Regulation Loss Flag bit

Value	Description
1	Voltage Domain 2 lost voltage regulation
0	Voltage Regulation in Domain 2 has not been lost

Bit 19 – VREG1R Voltage Domain 1 Regulation Loss Flag bit

Value	Description
1	Voltage Domain 1 lost voltage regulation
0	Voltage Regulation in Domain 1 has not been lost

Bit 9 – CM Configuration Mismatch Flag bit

Value	Description
1	A Configuration Mismatch Reset has occurred
0	A Configuration Mismatch Reset has not occurred

Bit 7 – EXTR External Reset ($\overline{MCLR}$) Pin bit

Value	Description
1	A Master Clear (pin) Reset has occurred
0	A Master Clear (pin) Reset has not occurred

Bit 6 – SWR Software RESET (Instruction) Flag bit

Value	Description
1	A RESET instruction has been executed
0	A RESET instruction has not been executed

Bit 4 – WDTO Watchdog Timer Time-out Flag bit

Value	Description
1	Device Reset has occurred due to WDT time-out
0	WDT time-out has not occurred

Bit 3 – SLEEP Wake-up from Sleep Flag bit

Value	Description
1	Device has been in Sleep mode
0	Device has not been in Sleep mode

Bit 2 – IDLE Wake-up from Idle Flag bit

Value	Description
1	Device has been in Idle mode
0	Device has not been in Idle mode

Bit 1 – BOR Brown-out Reset Flag bit

Value	Description
1	A Brown-out Reset has occurred
0	A Brown-out Reset has not occurred; Set by hardware at detection of a BOR event

Bit 0 – POR Power-on Reset Flag bit

Value	Description
1	A Power-on Reset has occurred
0	A Power-on Reset has not occurred; Set by hardware at detection of a POR event

12.3.13 User Clock Diagnostics Control Register

Name: CLKDIAG
Offset: 0x319C

Bit	31	30	29	28	27	26	25	24
	FLTIJEN	STOPPLL2	STOPPLL1	GENSEL[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
					SCSFLTDATA[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
				STOPGEN13	STOPGEN12	STOPGEN11	STOPGEN10	STOPGEN9
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	STOPGEN8	STOPGEN7	STOPGEN6	STOPGEN5	STOPGEN4	STOPGEN3	STOPGEN2	STOPGEN1
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – FLTIJEN SCS[x][3:0]Fault Injection Enable bit

Value	Description
1	Fault is inserted
0	Fault insertion is disabled

Bit 30 – STOPPLL2 PLL2 Reference Clock Monitor Disable bit

Value	Description
1	Selected reference clock for PLL2 is disconnected from the associated clock monitor
0	Selected reference clock for PLL2 is connected to the associated clock monitor

Bit 29 – STOPPLL1 PLL1 Reference Clock Monitor Disable bit

Value	Description
1	Selected reference clock for PLL1 is disconnected from the associated clock monitor
0	Selected reference clock for PLL1 is connected to the associated clock monitor

Bits 28:24 – GENSEL[4:0] Select the Clock Generator or PLL Generator for Fault Injection bits

Value	Description
11111	Reserved for future PLL generators
11110	PLLGEN2
11101	PLLGEN1
10100-01011	Reserved for future clock generators
01010	CLKGEN11
01001	CLKGEN10
01000	CLKGEN9
00111	CLKGEN8
00110	CLKGEN7

Value	Description
00101	CLKGEN6
00100	CLKGEN5
00011	CLKGEN4
00010	CLKGEN3
00001	CLKGEN2
00000	CLKGEN1

Bits 19:16 – SCSFLTDATA[3:0] Fault Data to be Injected bits
Ones invert the SCS[x] data bit
Zeros do not effect the SCS[x] data

Bit 12 – STOPGEN13 Stops the selected reference clock to the clock monitor for CLKGEN13 bit

Value	Description
1	Selected reference clock for CLKGEN13 is disconnected from the associated clock monitor
0	Selected reference clock for CLKGEN13 is connected to the associated clock monitor

Bit 11 – STOPGEN12 Stops the selected reference clock to the clock monitor for CLKGEN12 bit

Value	Description
1	Selected reference clock for CLKGEN12 is disconnected from the associated clock monitor
0	Selected reference clock for CLKGEN12 is connected to the associated clock monitor

Bit 10 – STOPGEN11 Stops the selected reference clock to the clock monitor for CLKGEN11 bit

Value	Description
1	Selected reference clock for CLKGEN11 is disconnected from the associated clock monitor
0	Selected reference clock for CLKGEN11 is connected to the associated clock monitor

Bit 9 – STOPGEN10 Stops the selected reference clock to the clock monitor for CLKGEN10 bit

Value	Description
1	Selected reference clock for CLKGEN10 is disconnected from the associated clock monitor
0	Selected reference clock for CLKGEN10 is connected to the associated clock monitor

Bit 8 – STOPGEN9 Stops the selected reference clock to the clock monitor for CLKGEN9 bit

Value	Description
1	Selected reference clock for CLKGEN9 is disconnected from the associated clock monitor
0	Selected reference clock for CLKGEN9 is connected to the associated clock monitor

Bit 7 – STOPGEN8 Stops the selected reference clock to the clock monitor for CLKGEN8 bit

Value	Description
1	Selected reference clock for CLKGEN8 is disconnected from the associated clock monitor
0	Selected reference clock for CLKGEN8 is connected to the associated clock monitor

Bit 6 – STOPGEN7 Stops the selected reference clock to the clock monitor for CLKGEN7 bit

Value	Description
1	Selected reference clock for CLKGEN7 is disconnected from the associated clock monitor
0	Selected reference clock for CLKGEN7 is connected to the associated clock monitor

Bit 5 – STOPGEN6 Stops the selected reference clock to the clock monitor for CLKGEN6 bit

Value	Description
1	Selected reference clock for CLKGEN6 is disconnected from the associated clock monitor
0	Selected reference clock for CLKGEN6 is connected to the associated clock monitor

Bit 4 – STOPGEN5 Stops the selected reference clock to the clock monitor for CLKGEN5 bit

Value	Description
1	Selected reference clock for CLKGEN5 is disconnected from the associated clock monitor
0	Selected reference clock for CLKGEN5 is connected to the associated clock monitor

Bit 3 – STOPGEN4 Stops the selected reference clock to the clock monitor for CLKGEN4 bit

Value	Description
1	Selected reference clock for CLKGEN4 is disconnected from the associated clock monitor
0	Selected reference clock for CLKGEN4 is connected to the associated clock monitor

Bit 2 – STOPGEN3 Stops the selected reference clock to the clock monitor for CLKGEN3 bit

Value	Description
1	Selected reference clock for CLKGEN3 is disconnected from the associated clock monitor
0	Selected reference clock for CLKGEN3 is connected to the associated clock monitor

Bit 1 – STOPGEN2 Stops the selected reference clock to the clock monitor for CLKGEN2 bit

Value	Description
1	Selected reference clock for CLKGEN2 is disconnected from the associated clock monitor
0	Selected reference clock for CLKGEN2 is connected to the associated clock monitor

Bit 0 – STOPGEN1 Stops the selected reference clock to the clock monitor for CLKGEN1 bit

Value	Description
1	Selected reference clock for CLKGEN1 is disconnected from the associated clock monitor
0	Selected reference clock for CLKGEN1 is connected to the associated clock monitor

12.3.14 Clock Monitor Control Register

Name: CMxCON
Offset: 0x3200, 0x3230, 0x3260, 0x3290

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	SLPEN				
Access	R/W		R/W	R/W				
Reset	1		0	0				

Bit	7	6	5	4	3	2	1	0
			CNTDIV[1:0]		FLTINJ[1:0]			WIDTH
Access			R/W	R/W	R/W	R/W		R/W
Reset			0	0	0	0		0

Bit 15 – ON Clock Monitor Enable bit

Value	Description
1	Clock Monitor is enabled
0	Clock Monitor is disabled

Bit 13 – SIDL Stop in Idle bit

Value	Description
1	Clock Monitor block will stop when IDLE mode is entered
0	Clock Monitor block will continue to operate when IDLE mode is entered

Bit 12 – SLPEN Sleep Mode Enable bit

Value	Description
1	Module continues to operate in Sleep modes
0	Module does not operate in Sleep modes

Bits 5:4 – CNTDIV[1:0] Counter Divider Selection bits(1)
These selection bits allow dividing down the input clock to the counter. The function is intended to detect over-clocking in Clock Monitor mode of operation.

Value	Description
11	Reserved
10	Divide-by 4
01	Divide-by 2
00	Divide-by 1

Bits 3:2 – FLTINJ[1:0] Fault Injection Enable Control bits(2)

These control bits allow injecting an artificially faulty condition into the macro for the purpose of inoperation reliability self-testing.

Value	Description
11	Artificial catastrophic fault injected into the macro by blanking the monitored clock
10	High-frequency drift fault injected into the macro by halving the reference clock
01	Low-frequency drift fault injected into the macro by halving the monitored clock
00	No artificial fault injected

Bit 0 – WIDTH Time Window Selection Control bit(3,4)

This control bit selects whether the Time Window Generator's clock high pulse defines the accumulation time window period.

Value	Description
1	Rising Edge to Next Falling Edge
0	Rising Edge to Rising Edge, see WINPR[31:0]

12.3.15 Clock Monitor Control Register

Name: CMxSTAT
Offset: 0x3204, 0x3234, 0x3264, 0x3294

Note:

1. Do not change the value while in operation.
2. CMxHWT/CMxLWT and CMxHFT/CMxLFT must all be cleared before the fault can be injected.
3. When set high, the function is intended for the pulse width measurement feature where the monitored clock (as opposed to the reference clock) is expected to clock the time window generator responsible for defining the measurement time period. Conversely, the reference clock is used to clock the counter at a higher rate of switching frequency.
4. When set high, the content of WINPR[31:0] prescaler is ignored while both the time window generator and the counter are reset.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					HWT	LWT	HFT	LFT
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
						TRIG	SATD	BUFV
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 11 – HWT High Warning Threshold Status bit
This status bit is set by hardware when the captured count value exceeds that of the high warning threshold register content. In other words, the monitored clock is at a higher frequency than desired (or the reference clock is running slower than expected).

Value	Description
1	High threshold warning
0	No warning

Bit 10 – LWT Low Warning Threshold Status bit
This status bit is set by hardware when the captured count value is less than that of the low warning threshold register content. In other words, the monitored clock is at a lower frequency than desired (or the reference clock is running faster than expected).

Value	Description
1	Low threshold warning
0	No warning

Bit 9 – HFT High Failing Threshold Status bit

This status bit is set by hardware when the captured count value exceeds that of the high failing threshold register content. In other words, the monitored clock is at a higher frequency than allowed (or the reference clock is running slower than allowed). When instanced as a Fail Safe Clock Monitor, a hardware switchover to the backup clock source will occur.

Value	Description
1	High threshold failing
0	No failure

Bit 8 – LFT Low Failing Threshold Status bit

This status bit is set by hardware when the captured count value is less than that of the low failing threshold register content. In other words, the monitored clock is at a lower frequency than allowed (or the reference clock is running faster than allowed). When instanced as a Fail Safe Clock Monitor, a hardware switchover to the backup clock source will occur.

Value	Description
1	Low threshold failing
0	No failure

Bit 2 – TRIG Time Window Generator Trigger Status bit

This status bit indicates the start of the accumulation time window. When set high by hardware, the condition implies that the selected reference clock is toggling properly. Clock Monitor Saturation Interrupt is invoked.

Value	Description
1	The accumulation time window has (re)started/stopped
0	The accumulation time window has not started

Bit 1 – SATD Counter Saturated Status bit

This status bit indicates the counter has saturated without being captured.

Value	Description
1	The counter has saturated at SAT[31:0]
0	The counter has not saturated

Bit 0 – BUFV Buffer Valid Status bit

This status bit indicates successful count capturing into the Data Buffer register. Reset to 1'b0, this bit is set high by hardware when the current count value of the counter is captured into BUF[31:0] at the end of each accumulation time window. As a result, the Count Ready Interrupt (upbs_event[4]) is invoked.

Value	Description
1	Accumulated count has been captured into BUF[31:0] and is ready for software use
0	Accumulated count has not been captured into BUF[31:0] yet, or has been obsoleted by catastrophic failure of the monitored clock for more than one accumulation window

12.3.16 Clock Monitor Prescaler Register

Name: CMxWINPR
Offset: 0x3208, 0x3238, 0x3268, 0x3298

Bit	31	30	29	28	27	26	25	24
	WINPR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	23	22	21	20	19	18	17	16
	WINPR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	15	14	13	12	11	10	9	8
	WINPR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	7	6	5	4	3	2	1	0
	WINPR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 31:0 – WINPR[31:0] Clock Monitor Prescaler
Prescaler value to alter ratio of reference and monitored clocks.

12.3.17 Clock Monitor Input Selection Register

Name: CMxSEL
Offset: 0x320C, 0x323C, 0x326C, 0x329C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CNTSEL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	7	6	5	4	3	2	1	0
	WINSEL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 15:8 – CNTSEL[7:0] Counter clock source
Selects the monitored clock source. See Table 12-5.

Bits 7:0 – WINSEL[7:0] Window clock source
Selects reference clock source. See Table 12-5.

12.3.18 Clock Monitor Buffer Register

Name: CMxBUF
Offset: 0x3210, 0x3240, 0x3270, 0x32A0

Bit	31	30	29	28	27	26	25	24
	BUF[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	23	22	21	20	19	18	17	16
	BUF[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	15	14	13	12	11	10	9	8
	BUF[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	7	6	5	4	3	2	1	0
	BUF[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 31:0 – BUF[31:0] Monitored Clock Count Value.
The Clock Monitor Data Buffer register contains the final accumulated count recorded by the counter during the previous accumulation window. Its content feeds the threshold limit comparators.

12.3.19 Clock Monitor Saturation Register

Name: CMxSAT
Offset: 0x3214, 0x3244, 0x3274, 0x32A4

Bit	31	30	29	28	27	26	25	24
	SAT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	23	22	21	20	19	18	17	16
	SAT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	15	14	13	12	11	10	9	8
	SAT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	7	6	5	4	3	2	1	0
	SAT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 31:0 – SAT[31:0] Clock Monitor Counter Saturation
The Clock Monitor Counter Saturation register contains the accumulated count value which causes the counter to saturate. If the counter has reached the count value programmed into this register before being captured, the CMxSATD bit is set with interrupt invoked.

12.3.20 Clock Monitor High Threshold Failing Register

Name: CMxHFAIL
Offset: 0x3218, 0x3248, 0x3278, 0x32A8

Bit	31	30	29	28	27	26	25	24
	HFAIL[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	23	22	21	20	19	18	17	16
	HFAIL[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	15	14	13	12	11	10	9	8
	HFAIL[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	7	6	5	4	3	2	1	0
	HFAIL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 31:0 – HFAIL[31:0] Clock Monitor High Threshold Failing bits
The Clock Monitor High Threshold Failing register contains the upper failing threshold limit against which the captured count is compared, see HWT. Failure is signaled when CMxBUF[31:0] > CMxHFAIL, so this register defines the fastest monitored frequency, slowest scaled reference frequency, or longest reference pulse width that can be measured before triggering a failure. Setting CMxHFAIL = 0xFFFFFFFF will have the effect of disabling this threshold.

12.3.21 Clock Monitor Low Threshold Failing Register

Name: CMxLFAIL
Offset: 0x321C, 0x324C, 0x327C, 0x32AC

Bit	31	30	29	28	27	26	25	24
	LFAIL[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	23	22	21	20	19	18	17	16
	LFAIL[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	15	14	13	12	11	10	9	8
	LFAIL[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	7	6	5	4	3	2	1	0
	LFAIL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 31:0 – LFAIL[31:0] Clock Monitor Low Threshold Failing bits
The Clock Monitor Low Threshold Failing register contains the lower failing threshold limit against which the captured count is compared (see LFT). Failure is signaled when CMxBUF[31:0] < CMxLFAIL, so this register defines the slowest monitored frequency, fastest scaled reference frequency or shortest reference pulse width that can be measured before triggering a failure. Setting CMxLFAIL = 0x00000000 will have the effect of disabling this threshold.

Value	Description
1	
0	

12.3.22 Clock Monitor High Threshold Warning Register

Name: CMxHWARN
Offset: 0x3220, 0x3250, 0x3280, 0x32B0

Bit	31	30	29	28	27	26	25	24
	HWARN[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	23	22	21	20	19	18	17	16
	HWARN[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	15	14	13	12	11	10	9	8
	HWARN[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	7	6	5	4	3	2	1	0
	HWARN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 31:0 – HWARN[31:0] Clock Monitor High Threshold Warning bits
The Clock Monitor High Threshold Warning register contains the upper warning threshold limit against which the captured count is compared (see HFT).
Warning is signaled when CMxBUF[31:0] > CMxHWARN, so this register defines the fastest monitored frequency, slowest scaled reference frequency or longest reference pulse width that can be measured before triggering a warning. Setting CMxHWARN = 0xFFFFFFFF will have the effect of disabling this threshold.

12.3.23 Clock Monitor Low Threshold Warning Register

Name: CMxLWARN
Offset: 0x3224, 0x3254, 0x3284, 0x32B4

Bit	31	30	29	28	27	26	25	24
	LWARN[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	23	22	21	20	19	18	17	16
	LWARN[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	15	14	13	12	11	10	9	8
	LWARN[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	7	6	5	4	3	2	1	0
	LWARN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 31:0 – LWARN[31:0] Clock Monitor Low Threshold Warning bits
The Clock Monitor Low Threshold Warning register contains the lower warning threshold limit against which the captured count is compared (see LWT).
Warning is signaled when CMxBUF[31:0] < CMxLWARN, so this register defines the slowest monitored frequency, fastest scaled reference frequency or shortest reference pulse width that can be measured before triggering a warning. Setting CMxLWARN = 0x00000000 will have the effect of disabling this threshold.

12.3.24 8 MHz FRC Controller Register

Name: FRCTUN
Offset: 0x32C0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			TUN[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 5:0 – TUN[5:0] Internal Fast RC Oscillator Tuning bits
This bit field specifies the user tuning capability for the internal fast RC oscillator.

Value	Description
011111	Maximum Frequency
011110	
...	
000001	
000000	Center Frequency, oscillator is running at calibrated frequency
111111	
111110	
...	
100001	
100000	Minimum Frequency

12.3.25 BFRC Controller Register

Name: BFRCTUN
Offset: 0x32C4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			TUN[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 5:0 – TUN[5:0] Internal Fast RC Oscillator Tuning bits
This bit field specifies the user tuning capability for the internal fast RC oscillator.

Value	Description
011111	Maximum Frequency
011110	
...	
000001	
000000	Center Frequency, oscillator is running at calibrated frequency
111111	
111110	
...	
100001	
100000	Minimum Frequency

12.4 Operation

12.4.1 Clock Generators (CLKGEN)

The dsPIC33AK128MC106 family of devices contains multiple clock generators. Each clock generator can be configured to use the input clock sources listed in Table 12-3. Clock generator 1 is the system clock, and it will always be enabled.

An expanded view of the CLKGEN is detailed in Figure 12-3.

The assignment of CLKGEN to peripherals is listed in Table 12-2.

The other clock generators can be configured for any clock source. For example, to reference FRC, set the COSC[3:0] bits to FRC source and enable clock switching.

Each generator can be enabled either via setting the ON bit in the CLKxCON register or when requested by a consumer (e.g. a peripheral).

The ON bit is logically ORed with clock request signals from other peripherals to enable the CLKGEN or PLL, and it can only be used to disable the CLKGEN or PLL when there are no active clock requests. In addition, clearing the OSCCTRL.PLLxEN bit disables PLLx.

Each clock generator has a dedicated divider (CLKxDIV) and control bits.

Example 12-1.

```
CLK11CONbits.NOSC = clkSource;        // Clock source selection
   CLK11CONbits.OSWEN = 1;                    // Request clock switch
   while(CLK11CONbits.OSWEN);                 // Verify switch complete

   CLK11DIVbits.INTDIV = 2;
   CLK11CONbits.DIVSWEN = 1;                  // Request divider switch
   while(CLK11CONbits.DIVSWEN);;              // Verify switch complete

   CLK11CONbits.OE = 1;                       // Enable output
   CLK11CONbits.ON = 1;                       // Enable GLKGEN

   while(!CLK11CONbits.CLKRDY);               // Verify CLKGEN ready
```

12.4.1.1 Fail-Safe Clock Monitor (FSCM)

Each CLKGEN has a Fail Safe Clock Monitor (FSCM) built inside to provide clock safety. Each FSCM is enabled via CLKxCON.FSCMEN. To provide faster response to a clock failure, the FSCM uses the 8 MHz BFRC as the reference clock.

In the event of an oscillator failure, the FSCM will generate an Oscillator Fail Interrupt, set the associated CLKFAIL/PLLFAIL flag, and switch the clock over to the specified backup Fail-Safe clock source which is selected via CLKxCON.BOSC bits. The Fail-Safe condition is exited with either a Reset or by completing a clock switch to a new stable clock input for the CLKGEN.

A clock monitor external to the CLKGENs is also included and discussed in 12.4.7. Clock Monitor

12.4.1.2 Setup for Using Clock Generator with 8 MHz Internal FRC

The following process is used to set up the CLKGEN1 to operate the device with an 8 MHz Internal FRC:

1. To set up the fail-safe for the clock generator, follow these steps:
   a. Select the backup clock source by selecting the BOSC bits in CLKxCON register.
   b. Enable fail-safe clock failure by setting the FSCMEN bit in CLKxCON register.
   c. Enable ClkFailInterrupt to generate an interrupt during clock failure.
2. To switch to a new oscillator for the clock generator, follow these steps:
   a. Select clock source to switch by writing to NOSC bits in the CLKxCON register.
   b. Enable switching by writing to the OSWEN bit in the CLKxCON register.
3. To further divide the clock out of the clock generator using CLKxDIV, follow these steps:
   a. Set the integer divide factor bit setting INTDIV bits in the CLKxDIV register.
   b. Set the fractional divide factor bit setting FRACDIV bits in the CLKxDIV register. FRACDIV will not work if INTDIV is configured to 0.
   c. Set the DIVSWEN bit in the CLKxCON register to enable divide factors to get updated.
4. Enable the clock generator (if not enabled by default) by setting the ON bit in CLKxCON register.

Example 12-2. Code Example for Clock Generator 1 Switching to FRC

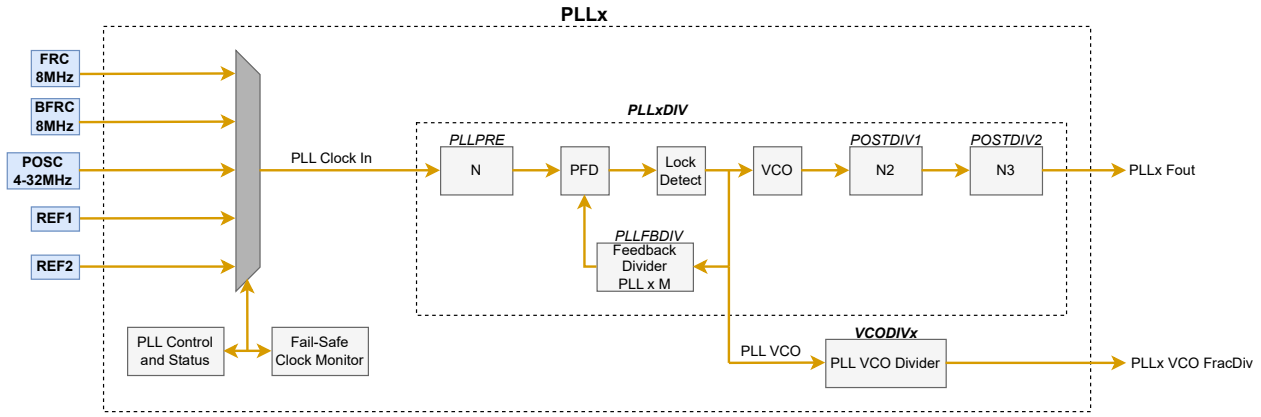
```
//enable clock generator
    CLK1CONbits.ON = 1;
    CLK1CONbits.OE = 1;

    //configure backup oscillator in case of failure
```

```
    CLK1CONbits.BOSC = 2; //BFRC
                    //select backup clock source:
                    //[1] = FRC - Internal 8 MHz RC oscillator
                    //[2] = BFRC - Internal Backup 8 MHz RC oscillator
                    //[3] = POSC - Primary crystal oscillator (4-32 MHz)
                    //[4] = LPRC
                    //[5] = PLL1 Fout output
                    //[6] = PLL2 Fout output
                    //[7] = PLL1 VCO FracDiv output
                    //[8] = PLL2 VCO FracDiv output
                    //[9] = REFI1 - user definable clock source
                    //[10] = REFI2 - user definable clock source


    CLK1CONbits.FSCMEN = 1; //enable fail safe


    //configure clock divide
    CLK1DIVbits.INTDIV = 1; //integer divide factor
    CLK1DIVbits.FRACDIV = 0x0080; //fractional divide factor
    CLK1CONbits.DIVSWEN = 1; //enable divide factors to get updated
    while (CLK1CONbits.DIVSWEN); //hardware cleared
    // Fdiv = Fin / 2*(INTDIV+(FRCDIV/512))

    //enable clock switching
    CLK1CONbits.NOSC = 1 ; //select clock source
                     //[1] = FRC - Internal 8 MHz RC oscillator
                     //[2] = BFRC - Internal Backup 8 MHz RC oscillator
                     //[3] = POSC - Primary crystal oscillator (4-32 MHz)
                     //[4] = LPRC
                     //[5] = PLL1 Fout output
                     //[6] = PLL2 Fout output
                     //[7] = PLL1 VCO FracDiv output
                     //[8] = PLL2 VCO FracDiv output
                     //[9] = REFI1 - user definable clock source
                    //[10] = REFI2 - user definable clock source

    CLK1CONbits.OSWEN = 1; //enable clock switching
    while (CLK1CONbits.OSWEN); //wait for switching(hardware clear))


    IFS0bits.CLKFAILIF = 0;    // enable clock failure interrupt
    IEC0bits.CLKFAILIE = 1;
}
```

12.4.2 Phase-Locked Loop (PLL)

The Primary Oscillator and Internal FRC Oscillator sources can optionally use an on-chip PLL to obtain higher operating speeds. Figure 12-4 illustrates a block diagram of the PLL module.

Figure 12-4. PLL Block Diagram



For PLL operation, the following requirements must be met at all times without exception:

- The PLL Input Frequency (FPLLI) must be in the range specified in Electrical Characteristics.

- The VCO frequency must be in the range specified in Electrical Characteristics.
- The feedback divider must be in the range specified in Electrical Characteristics.
- The output of the PLL module is in the range specified in Electrical Characteristics. The first output divider (POSTDIV1) should be larger than the value for the second output divider (POSTDIV2).

The PLL Phase Detector Input Divider Select bits (PLLPRE[3:0]) in the PLL Divider register (PLLxDIV[29:24]) specify the input divider ratio (N1), which is used to scale down the input clock ($_{FPLLI}$) to meet the PFD input frequency range.

The PLL Feedback Divider bits (PLLFBDIV[7:0]) in the PLL Divider register (PLLFBD[7:0]) specify the divider ratio (M), which scales down the VCO Output Frequency (F_{VCO}) for feedback to the PFD input. The VCO Frequency (F_{VCO}) is 'M' times the PFD Input Frequency (F_{PFD}).

There are two PLL VCO output dividers configured through the POSTDIV1[2:0] and POSTDIV2[2:0] select bits. These bits are located in the PLL Divider register (PLLxDIV[6:4] and PLLxDIV[2:0]) and specify the divider ratios (N2 and N3) that limit the PLL Output Frequency (PLL FOUT). Please refer to Table 12-2 for max frequencies related to each peripheral.

Equation 12-1 provides the relationship between the PLL Input Frequency ($_{FPLLI}$) and VCO Output Frequency (F_{VCO}).

Equation 12-1. F_{VCO} Calculation

$$FVCO = FPLLI \times \left(\frac{M}{N1}\right) = FPLLI \times \left(\frac{PLLFBDIV[7:0]}{PLLPRE[3:0]}\right)$$

Equation 12-2 provides the relationship between the PLL Input Frequency (F_{PLLI}) and PLL Output Frequency (F_{PLLO}).

Equation 12-2. F_{PLLO} Calculation

$$FPLLO = FPLLI \times \left(\frac{M}{N1 \times N2 \times N3}\right) = FPLLI \times \left(\frac{PLLFBDIV[7:0]}{PLLPRE[3:0] \times POSTDIV1[2:0] \times POSTDIV2[2:0]}\right)$$

Where:
$M = PLLFBDIV[7:0]$
$N1 = PLLPRE[3:0]$
$N2 = POSTDIV1[2:0]$
$N3 = POSTDIV2[2:0]$

12.4.2.1 Input Clock Limitation at Start-up for PLL Mode

Table 13-9 provides the default values of the PLL prescaler, PLL feedback divider and both PLL postscalers at Power-on Reset (POR).

Table 12-6. PLL Mode Defaults

Register	Bit Field	Value at POR Reset	PLL Divider Ratio
PLLxDIV	PLLPRE[3:0]	0001	N1 = 1
PLLxDIV	POSTDIV1[2:0]	111	N2 = 7
PLLxDIV	POSTDIV2[2:0]	001	N3 = 1
PLLxDIV	PLLFBDIV[7:0]	11001000	M = 200

Given these Reset values, the following equations provide the PLL Input Frequency (F_{PLLI}) and VCO Output Frequency (F_{VCO}) at Power-on Reset.

Equation 12-3. F_{VCO} at Power-on Reset

$$FVCO = FPLLI\left(\frac{M}{N1}\right) = FPLLI\left(\frac{200}{1}\right) = 200\ FPLLI$$

Equation 12-4. FPLLO at Power-on Reset

$$FPLLO = FPLLI\ \left(\frac{M}{N1 \times N2 \times N3}\right) = FPLLI\ \left(\frac{200}{1 \times 7 \times 1}\right) = 28.5\ FPLLI$$

To use the PLL with settings other than the default settings and to ensure all PLL requirements are met, follow this process:

1. Power up the device with the Internal FRC.
2. Change the FBDIV, PLLPRE, POSTDIV1 and POSTDIV2 bit values, based on the input frequency, to meet these PLL requirements:

   The PLL Input Frequency (F_{PLLI}) must be in the range specified in Electrical Characteristics.

   The VCO Output Frequency (F_{VCO}) must be in the range specified in Electrical Characteristics.
3. Writing PLLxCON:
   a) Enable PLL Input and Feedback Divider update by setting the PLLSWEN bit in the PLLxCON register.
   b) The first output divider (POSTDIV1) should be larger than the value for the second output divider (POSTDIV2). The output dividers POSTDIV1 and POSTDIV2 should not be changed while the PLL is operating. The input reference clock divider and the feedback divider may be updated during PLL operation.
   c) Enable PLL Output Divider update by setting the FOUTSWEN bit in the PLLxCON register.
   d) Select clock source by setting the NOSC[3:0] bits in the PLLxCON register.
   e) Enable clock switching by setting the OSWEN bit in the PLLxCON register.

   Note:
   It is recommended to change clock divider settings before the initial clock switch occurs. The reference clock (PLLPRE) and feedback dividers (FBDIV) can be changed during PLL operation, but care should be taken not to generate an invalid clock frequency.

   When switching from low-speed clock to a high-speed clock, the DIVSW process should be done before a clock switch to prevent undivided high speed clocks from passing through. The opposite holds true for a high-to-low switch, the clock switch should occur before a DIVSW.

12.4.2.2 PLL Lock Status

Whenever the PLL input frequency, the PLL prescaler or the PLL feedback divider is changed, the PLL requires a finite amount of time (T_{LOCK}) to synchronize to the new settings.

The PLLxRDY bit in the Oscillator Control register (OSCCTRL[5]) and the CLKRDY bit in the PLL Control register PLLxCON[31] are read-only status bits that indicate the PLL ready status of the PLL. The LOCK bit is cleared at a Power-on Reset and on a clock switch operation when the PLL is selected as the destination clock source. It remains clear when any clock source not using the PLL is selected. After a clock switch event in which the PLL is enabled, it is advisable to wait for the LOCK bit to be set before executing other code.

12.4.2.3 PLL Setup

12.4.2.3.1 Setup for Using PLL with the Primary Oscillator (POSC)

The following process is used to set up the PLL to operate the device at 200 MIPS with a 10 MHz external crystal:

1. Set the configuration for the primary oscillator to HS mode using POSCMD[1:0] bits in the OSCCFG register.
2. Setting up divider settings: to execute instructions at 200 MHz, a PLL output frequency of 200 MHz will be required. To set up the PLL and meet the requirements of the PLL, follow these steps:
   - i. Select the PLL prescaler.
     - Select a PLL prescaler value of N1 = 1
     - F_{PLLI} = 10 MHz
     - F_{PFD} = 10 MHz(1/N1) = 10 MHz(1) = 10 MHz
   - ii. Select the feedback divider to meet the VCO output frequency requirement, as well as achieve the desired F_{VCO} frequency.
     - Select a feedback divider value of M = 100
     - F_{VCO} = F_{PLLI} x (M/N1) = 10 MHz x (100/1) = 1 GHz
   - iii. Select values for the first and second PLL postscalers to achieve the required FPLLO frequency.
     - Select values for the first and second postscalers of N2 = 5 and N3 = 1
     - FPLL F_{OUT}/FPLLO = F_{VCO}/(N2 x N3) = 1 GHz/5 = 200 MHz
   - iv. Enable the PLL Input and Feedback Divider update by setting PLLSWEN bit in the PLLxCON register. Enable the PLL Output Divider update by setting FOUTSWEN bit in the PLLxCON register.
   - v. Select clock source by setting NOSC[3:0] bits in the PLLxCON register to Primary (3).
   - vi. Enable clock switching by setting OSWEN bit in the PLLxCON register.
3. Writing PLLxCON (can be done in one PLLxCON write):
   - i. Enable the PLL input and Feedback Divider update by setting PLLSWEN bit in the PLLxCON register. Enable the PLL Output Divider update by setting FOUTSWEN bit in the PLLxCON register.
   - ii. Select clock source by setting NOSC[3:0] bits in the PLLxCON register to Primary (3).
   - iii. Enable clock switching by setting OSWEN bit in the PLLxCON register.

Example 12-3 illustrates code for using the PLL with the Primary Oscillator.

Note: PLL1CON writes written out for clarity, These writes can be achieved in a single PLL1CON write.

Example 12-3. Code Example for Using PLL with the Primary Oscillator (POSC)

```
// code example for 200MIPS PLL clock using 10MHz crystal primary oscillator

//set crystal configuration using configuration register
  OSCCFGbits.POSCMD = 2; //set desired Primary oscillator mode
    // 0b00 : EC mode
    // 0b10 : HS mode
    // 0b01 : XT mode
    // 0b11 : Disabled

    //configure backup oscillator in case of failure
    PLL1CONbits.BOSC = 2; //select BFRC backup clock source:

    PLL1CONbits.FSCMEN = 1; //enable fail safe

    //configure PLL values
    PLL1DIVbits.FBDIV = 100; //Feedback Divider
    PLL1DIVbits.PLLPRE = 1; //Reference Clock Divider
    PLL1DIVbits.POSTDIV1 = 5; //Post Divider #1
    PLL1DIVbits.POSTDIV2 = 1; //Post Divider #2

    // PLL Fout = Fin*FBDIV / (PLLPRE * POSTDIV1 * POSTDIV2)
```

```
// PLL Fout = 10M*100/(1*5*1) =200MHz

//Enable PLL Input and Feedback Divider update
PLL1CONbits.PLLSWEN = 1;
while (PLL1CONbits.PLLSWEN == 1);
//Enable PLL Output Divider update
PLL1CONbits.FOUTSWEN = 1;
while (PLL1CONbits.FOUTSWEN);

//select clock switching clock source
PLL1CONbits.NOSC = 3; //select POSC clock source
                //[1] = FRC - Internal 8 MHz RC oscillator
                //[2] = BFRC - Internal Backup 8 MHz RC oscillator
                //[3] = POSC - Primary crystal oscillator (4-32 MHz)
                //[9] = REFI1 - user definable clock source
                //[10] = REFI2 - user definable clock source

PLL1CONbits.OSWEN = 1; //enable clock switching
while (PLL1CONbits.OSWEN); //wait for switching(hardware clear)

while(!OSCCTRLbits.PLL1RDY) //wait for clock to be ready

//Enable PLL clock generator
PLL1CONbits.ON = 1;
PLL1CONbits.OE = 1;
```

12.4.2.3.2 Setup for Using PLL with 8 MHz Internal FRC

The following process is used to set up the PLL to operate the device at 100 MIPS with an 8 MHz Internal FRC and configure PLL VCO output to 250 MHz.

1. To execute instructions at 100 MHz, a PLL output frequency of 100 MHz will be required.
2. To set up the PLL and meet the requirements of the PLL, follow these steps:
   - i. Select the PLL prescaler to meet the PFD input frequency requirement.
     - Select a PLL prescaler value of N1 = 1
     - F_{PLLI} = 8 MHz
     - F_{PFD} = 8 MHz(1/N1) = 8 MHz(1) = 8 MHz
   - ii. Select the feedback divider to meet the VCO output frequency requirement as well as achieve the desired F_{VCO} frequency.
     - Select a feedback divider value of M = 125
     - F_{VCO} = F_{PLLI} x (M/N1) = 8 MHz x (125/1) = 1 GHz
   - iii. Select values for the first and second PLL postscalers to achieve the required F_{PLLO} frequency.
     - Select values for the first and second postscalers of N2 = 5 and N3 = 2
     - F_{PLLO} = F_{VCO}/(N2 x N3) = 1 GHz/10 = 100 MHz
3. Writing PLLxCON (can be done in one PLLxCON write):
   - i. Enable the PLL Input and Feedback Divider update by setting PLLSWEN bit in the PLLxCON register. Enable PLL Output Divider update by setting FOUTSWEN bit in the PLLxCON register.
   - ii. Select clock source by setting NOSC<3:0> bits in the PLLxCON register to FRC (1).
   - iii. Enable clock switching by setting OSWEN bit in the PLLxCON register.
4. PLL VCO output setup is optional for normal PLL output. PLL VCODIV output is necessary if PLL VCODIV will be chosen as a CLKGEN input. For PLL VCO output, the PLLxDIV register needs to be configured. INTDIV bits of the PLLxDIV register are configured as two.
   - FVCO / INTDIV*2 = 1 GHz/ 2*2 =250 MHz

   Example 12-4 illustrates code for using the PLL with an 8 MHz Internal FRC Oscillator.

Note: PLL1CON writes written out for clarity. These writes can be achieved in a single PLL1CON write.

Example 12-4. Code Example for Using PLL with 8 MHz Internal FRC

```
// code example for 100MIPS PLL clock using 10MHz crystal primary oscillator

    //configure backup oscillator in case of failure
    PLL1CONbits.BOSC = 2;       //BFRC as backup clock source

    PLL1CONbits.FSCMEN = 1;     //enable fail safe

    //configure PLL values
    PLL1DIVbits.PLLPRE = 1;     //Reference Clock Divider
    PLL1DIVbits.POSTDIV1 = 5;   //Post Divider #1
    PLL1DIVbits.POSTDIV2 = 2;   //Post Divider #2
    // PLL Fout = Fin*FBDIV / (PLLPRE * POSTDIV1 * POSTDIV2)
    // PLL Fout = 8M*125/(1*5*2) =100MHz

    //Enable PLL Input and Feedback Divider update
    PLL1CONbits.PLLSWEN = 1;
    while (PLL1CONbits.PLLSWEN == 1);
    //Enable PLL Output Divider update
    PLL1CONbits.FOUTSWEN = 1;
    while (PLL1CONbits.FOUTSWEN);
    //select clock switching clock source
    PLL1CONbits.NOSC = 1;    //select FRC clock source
                   //[1] = FRC - Internal 8 MHz RC oscillator
                   //[2] = BFRC - Internal Backup 8 MHz RC oscillator
                   //[3] = POSC - Primary crystal oscillator (4-32 MHz)
                   //[9] = REFI1 - user definable clock source
                   //[10] = REFI2 - user definable clock source

    PLL1CONbits.OSWEN = 1;     //enable clock switching
    while (PLL1CONbits.OSWEN); //wait for switching(hardware clear)

    while(!OSCCTRLbits.PLL1RDY) //wait for clock to be ready

   //Enable PLL clock generator
    PLL1CONbits.ON = 1;
    PLL1CONbits.OE = 1;

   //configure PLL VCO Divider registers for PLL1 VCO Div clock source
    VCO1DIVbits.INTDIV = 2;    //integer divide factor
    // PLL VCO DIV = PLL VCO clock / 2* INTDIV

    //Enable PLL VCO Divider update
    PLL1CONbits.DIVSWEN = 1;
    while (PLL1CONbits.DIVSWEN);
```

12.4.3 Primary Oscillator (POSC)

The dsPIC33A devices contain one instance of the Primary Oscillator (POSC). The Primary Oscillator is available on the OSCI and OSCO pins of the dsPIC33A devices. This connection enables an external crystal (or ceramic resonator) to provide the clock to the device. The Primary Oscillator provides three modes of operation:

- Medium Speed Oscillator (MS Mode)
  - The MS mode is a Medium Gain, Medium Frequency mode used to work with crystal frequencies of 3.5 MHz to 10 MHz.
- High-Speed Oscillator (HS Mode)
  - The HS mode is a High Gain, High-Frequency mode used to work with crystal frequencies of 10 MHz to 32 MHz.
- External Clock Source Operation (EC Mode)

– If the on-chip oscillator is not used, the EC mode allows the internal oscillator to be bypassed. The device clocks are generated from an external source (see Table 37-24 for constraints) and input on the OSCI pin.

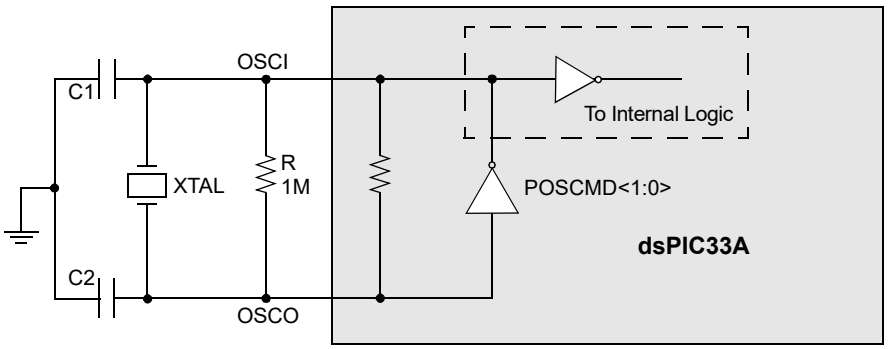
The POSCMD[1:0] bits in the Oscillator Configuration register (OSCCFG) specify the Primary Oscillator mode. Table 12-7 provides the options selected by specific bit configurations, which are programmed at the time of device programming.

Table 12-7. Primary Oscillator Clock Source Options

POSCMD[1:0] Value	Primary Oscillator Source and Mode
11	Disabled (GPIO function)
10	Primary Oscillator: High-Frequency Mode (HS)
01	Primary Oscillator: Medium Frequency Mode (MS)
00	External Clock Mode (EC)

Figure 12-5 is a recommended crystal oscillator circuit diagram for the dsPIC33A devices. Capacitors C1 and C2 form the Load Capacitance (Cl) for the crystal. The optimum Load Capacitance for a given crystal is specified by the crystal manufacturer. Load Capacitance can be calculated as shown in Equation 12-5.

Figure 12-5. Crystal or Ceramic Resonator Operation (XT or HS Oscillator Mode)



Equation 12-5. Crystal Load Capacitance

$$C_L = C_S + \frac{C1 \times C2}{C1 + C2}$$

Note: Where *Cs* is the stray capacitance.

Assuming C1 = C2, Equation 12-6 gives the capacitor value (C1, C2) for a given load and stray capacitance.

Equation 12-6. External Capacitor for Crystal

$$C1 = C2 = 2 \times (C_L - C_S)$$

For more information on crystal oscillators and their operation, refer to “Related Application Notes”.

12.4.3.1 Oscillator Start-up Time

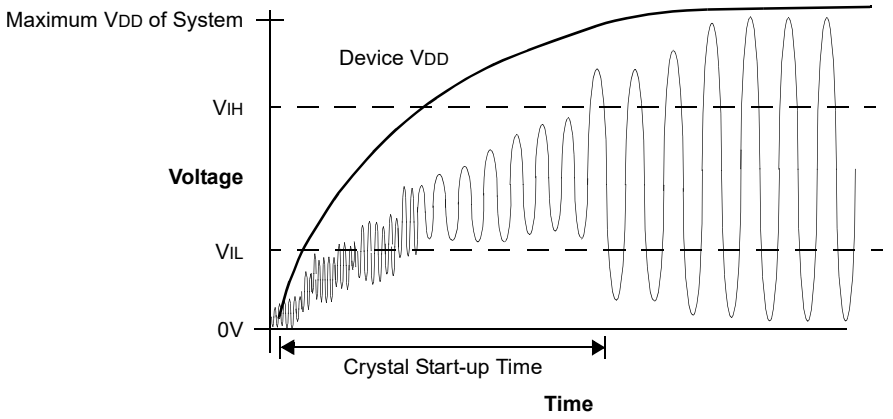
As the device voltage increases from V_{SS}, the oscillator will start its oscillations. The time required for the oscillator to start oscillating depends on these factors:

- Crystal and resonator frequency
- Capacitor values used (C1 and C2 in Figure 12-5)

- Device V_{DD} rise time
- System temperature
- Series resistor value and type if used
- Oscillator mode selection of device (selects the gain of the internal oscillator inverter)
- Crystal quality
- Oscillator circuit layout
- System noise

Figure 12-6 illustrates a plot of a typical oscillator and resonator start-up.

Figure 12-6. Example Oscillator and Resonator Start-up Characteristics



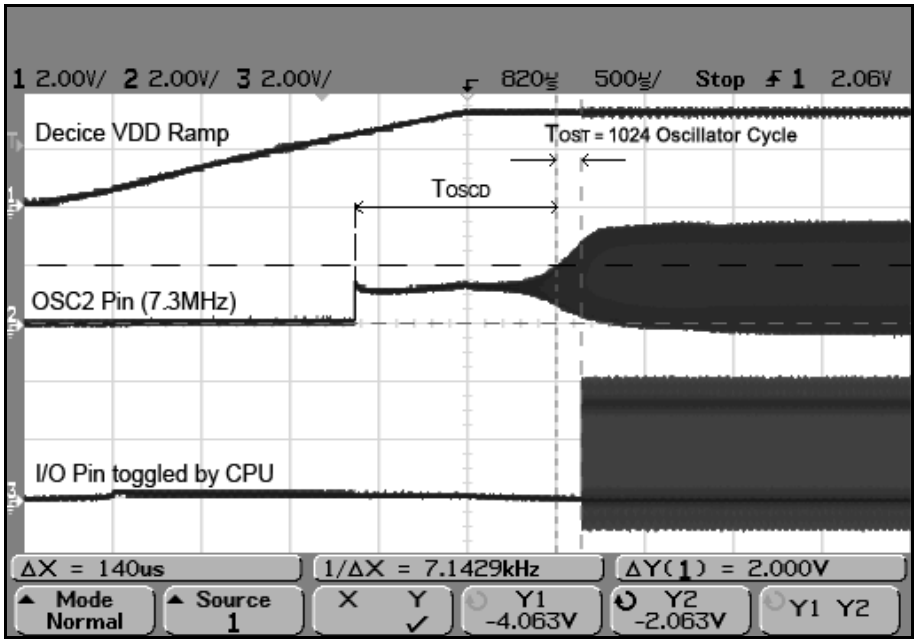
To ensure that a crystal oscillator (or ceramic resonator) has started and stabilized, an Oscillator Start-up Timer (OST) is provided with the Primary Oscillator (POSC). The OST is a simple, 10-bit counter that counts 1024 cycles before releasing the oscillator clock to the rest of the system. This time-out period is denoted as T_{OST}.

The amplitude of the oscillator signal must reach the V_{IL} and V_{IH} thresholds for the oscillator pins before the OST can begin to count cycles. The T_{OST} interval is required every time the oscillator restarts (that is, on POR, BOR and wake-up from Sleep mode) when XT or HS mode is selected in the Configuration Words. The T_{OST} timer does not exist when EC mode is selected.

After the Primary Oscillator is enabled, it takes a finite amount of time to start oscillating. This delay is denoted as T_{OSCD}. After T_{OSCD}, the OST timer takes 1024 clock cycles (T_{OST}) to release the clock. The total delay for the clock to be ready is: $T_{OSCD} + T_{OST}$. If the PLL is used, an additional delay is required for the PLL to lock. For more information, see 12.4.2. Phase-Locked Loop (PLL).

Primary Oscillator start-up behavior is illustrated in Figure 12-7, where the CPU begins toggling an I/O pin when it starts execution after the $T_{OSCD} + T_{OST}$ interval.

Figure 12-7. Oscillator Start-up Characteristics



12.4.3.2 Primary Oscillator Pin Functionality

The Primary Oscillator pins (OSCI and OSCO) can be used for other functions when the oscillator is not being used. The POSCMD[1:0] bits in the Oscillator Configuration register (OSCCFG) determine the oscillator pin function. The POSCIOFNC bit (OSCCFG) determines the OSCO pin function.

POSCIOFNC: OSCO Pin Function bit (except in XT and HS modes):

- 1 = OSCO is the clock output and the instruction cycle (F_{CY}) clock is output on the OCSO pin (see Figure 12-8)
- 0 = OSCO is a general purpose digital I/O pin

The oscillator pin functions are provided in Table 12-8.

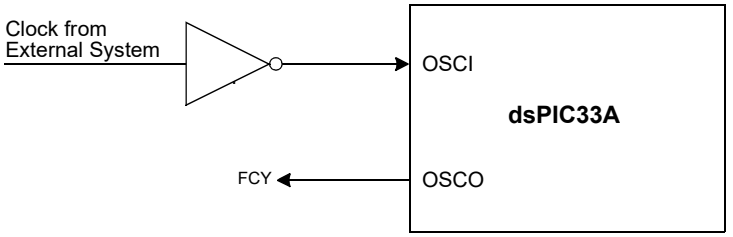
Table 12-8. Clock Pin Function Selection

Oscillator Source	POSCIOFNC Value	POSCMD[1:0] Value	OSCI Pin Function(1)	OSCO Pin Function(2)
Primary Oscillator Disabled	1	11	Digital I/O	Clock Output
Primary Oscillator Disabled	0	11	Digital I/O	Digital I/O
HS	x	10	OSCI	OSCO
MS	x	01	OSCI	OSCO
EC	1	00	OSCI	Clock Output
EC	0	00	—	Digital I/O

Notes:

1. OSCI pin function is determined by the Primary Oscillator Mode Selection (POSCMOD[1:0]) Configuration bits.
2. OSCO pin function is determined by the Primary Oscillator Mode Selection (POSCMOD[1:0]), OSCIOFNC Configuration bits.

Figure 12-8. OSCO Pin for Clock Output (in EC Mode)



12.4.4 Internal Fast RC (FRC) Oscillator

The dsPIC33A devices contain one instance of the Internal Fast RC (FRC) Oscillator. The FRC Oscillator provides a nominal 8 MHz clock without requiring an external crystal or ceramic resonator, which results in system cost savings for applications that do not require a precise clock reference.

The application software can tune the frequency of the oscillator using the FRC Oscillator Tuning bits (TUN[5:0]) in the FRC Oscillator Tuning register (FRCTUN[5:0]).

The Internal FRC Oscillator starts quickly. Unlike a crystal oscillator, which can take several milliseconds to begin oscillation, the Internal FRC starts oscillating immediately.

12.4.5 BFRC Oscillator

The dsPIC33A devices contain one instance of the Internal Backup Fast RC (BFRC) Oscillator and can function as system clock in the event of a FRC failure. The BFRC Oscillator provides a nominal 8 MHz clock without requiring an external crystal or ceramic resonator, which results in system cost savings for applications that do not require a precise clock reference.

The application software can tune the frequency of the oscillator using the FRC Oscillator Tuning bits (TUN[5:0]) in the FRC Oscillator Tuning register (BFRCTUN[5:0]).

12.4.6 Internal Low-Power RC (LPRC) Oscillator

The dsPIC33A devices contain one instance of the Internal LPRC oscillator. The LPRC oscillator is implemented as the BFRC divided by 244 to yield a clock frequency of 32.768 kHz.

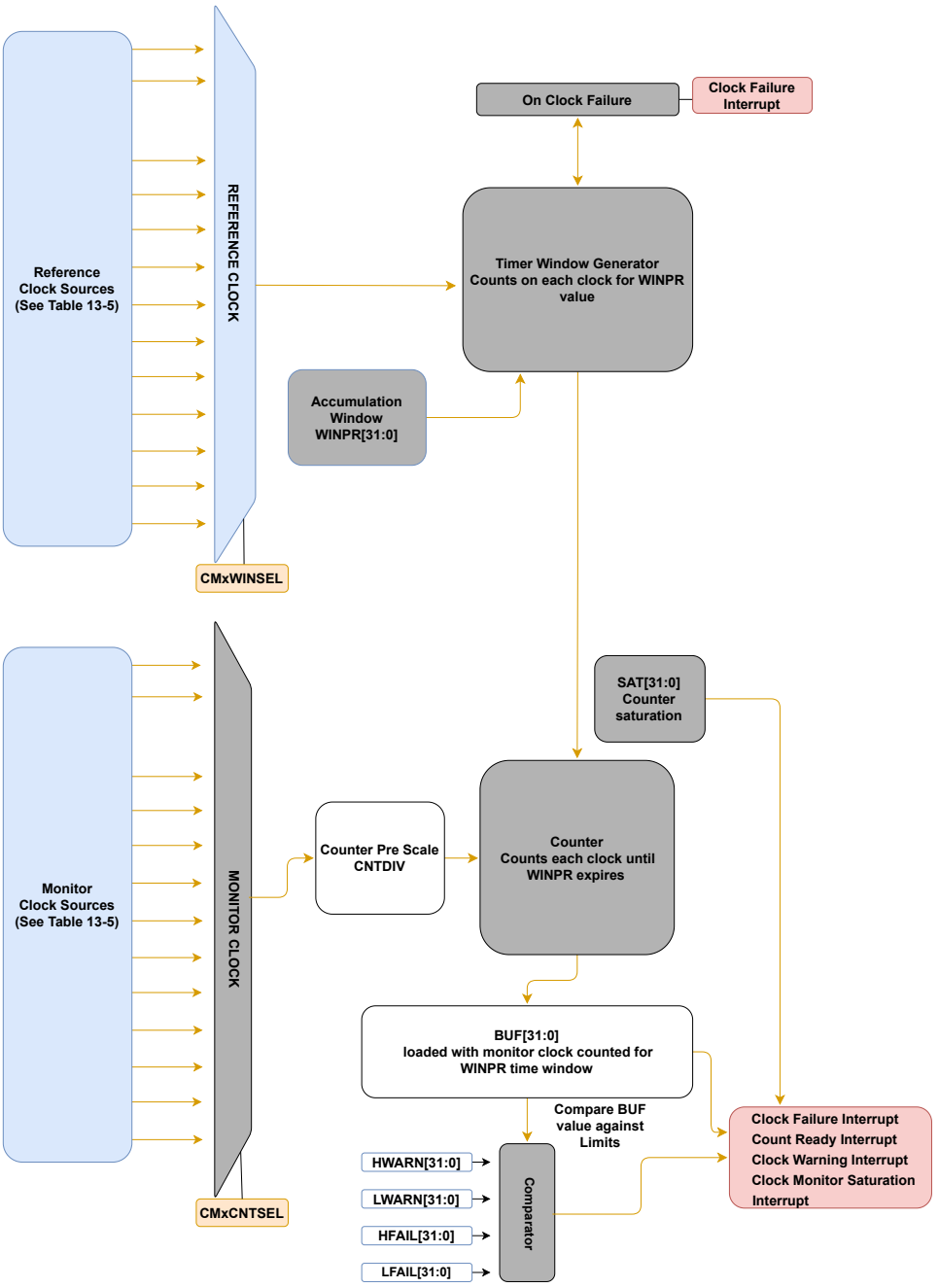
12.4.7 Clock Monitor

The clock monitor's purpose is to detect system clock failure or anomalies and allow the system to take action. The clock monitor uses a second clock (reference) source to compare to the clock being monitored. Clock monitors are assigned to a clock source such as a CLKGEN, PLL output or other system clocks including FRC. The clock monitor provides the following features:

- Configurable Clock Sources:
  - Monitored clock sources
  - Reference clock sources
- Flexible Detection Capability
- Clock Frequency Drift Detection
- Selectable Threshold Limits for Warning and/or Failing
- Monitored Clock Catastrophic Detection
- Reference Clock Catastrophic Detection
- User Optimizable Accuracy via:
  - Accumulation time adjustment
  - Selectable clock sources

- Register Write Protection
- Fault Injection Capability
- Frequency Measurement
- Pulse Width and Duty Cycle Measurement

Figure 12-9. Clock Monitor Architecture



12.4.7.1 Clock Monitor Overview

The clock monitor is primarily used for constant clock monitoring in real time with the following added features:

- Monitored Clock Frequency Drift Detection

- Under/Over Clocking Warning Detection
- Under/Over Clocking Failing Detection
- Monitored Clock Catastrophic Detection
- Reference Clock Detection
- Unable to Start
- Failure After Normal Start

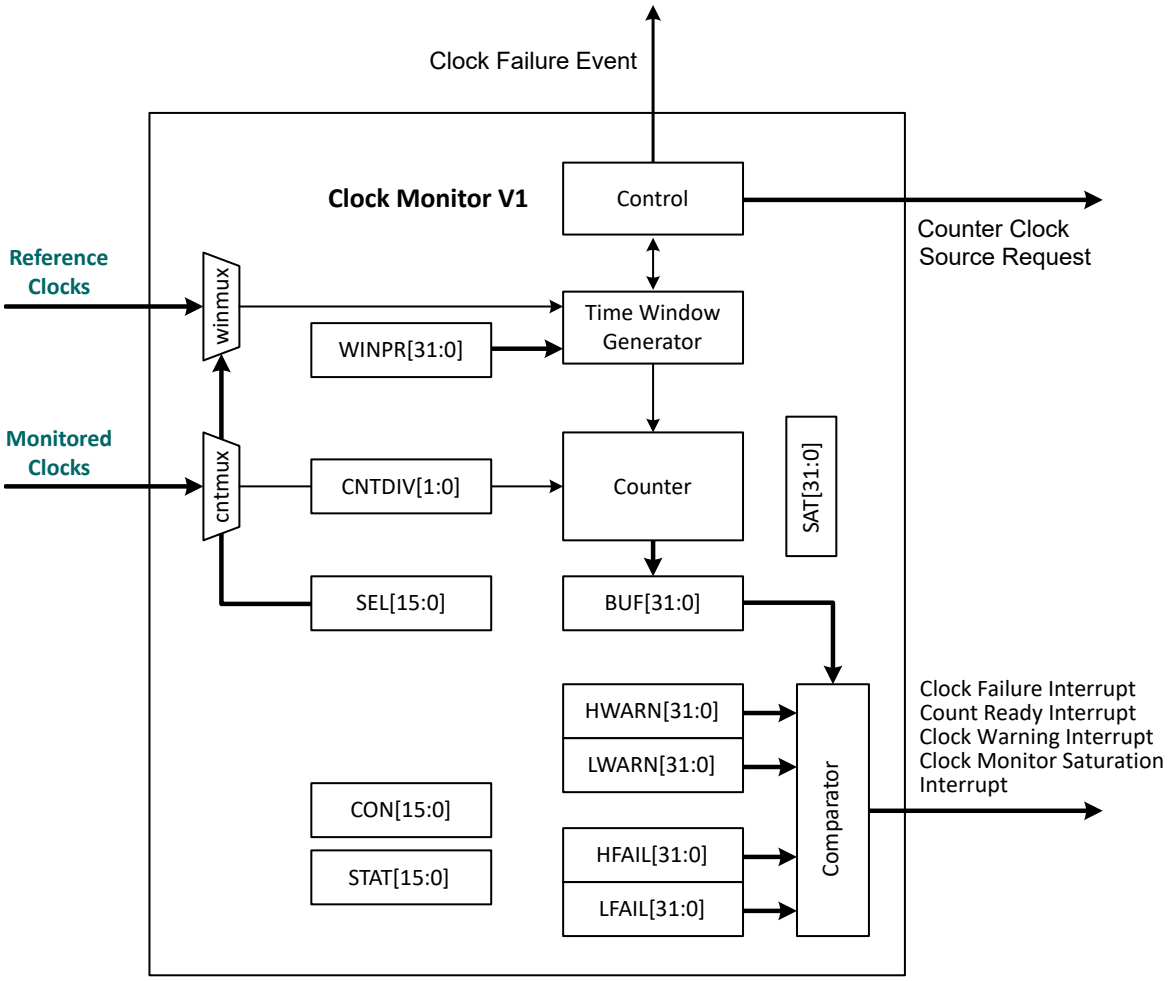
Clock monitoring operates on two different clock sources to accomplish its task, both of which are user selectable from various different clock inputs. Refer to WINSEL and CNTSEL within CMxSEL.

WINSEL selects a reference clock, and CNTSEL selects a monitored clock. The reference clock is used to generate a user-programmable time window defining a known accumulation time period, specified by WINPR[31:0] bits in the CMxWINPR register, over which the monitored clock is allowed to advance the internal counter, thus accumulating its count value.

As each accumulation time period ends, the monitored clock count value is captured into a separate Data Buffer register, see BUF[31:0] in register CMxBUF, while the new accumulation time period begins with a cleared accumulator.

Simultaneously, the ClkxReadyInterrupt is invoked to notify software of the new BUF[31:0] capture contents. The buffer contents are then evaluated by a digital comparator, where various detections are made based on the set and user-defined criteria. Four independent threshold limit registers are employed to allow the user to define the acceptable ranges of the monitored clock frequency. The process is repeated until the module is disabled.

Figure 12-10. Monitor Function

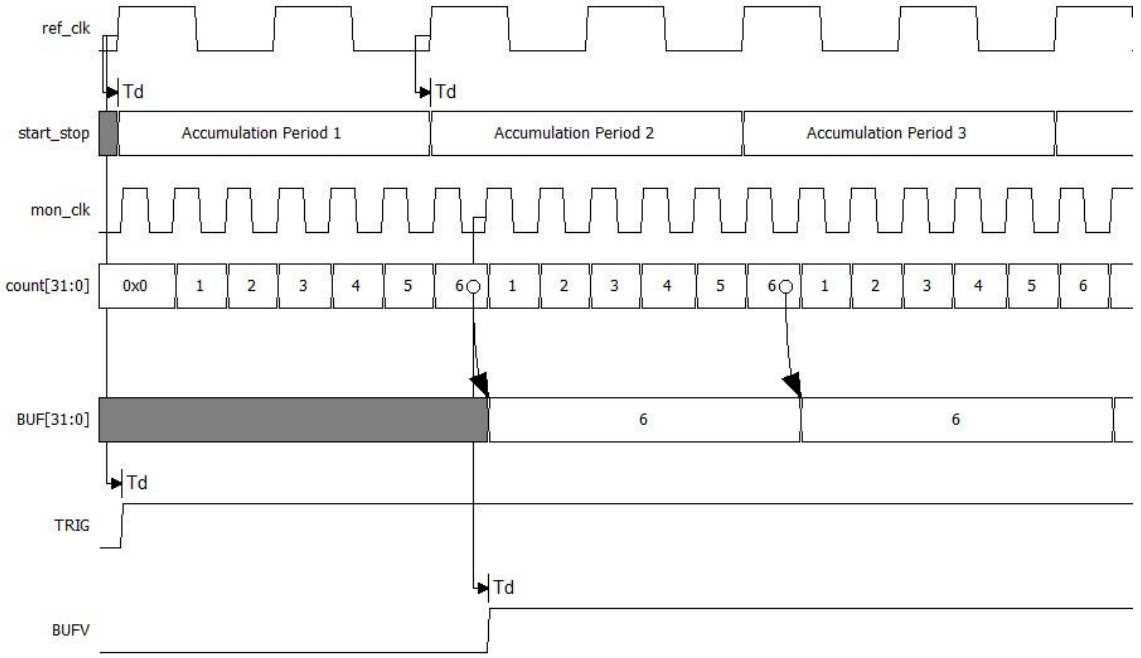


Based on the user selected timings on the reference clock, the start and stop points are chosen to define the desirable time window to accumulate the count value of the counter. The start time initializes the counter to zero and begins adding monitored clock rising edges to itself, whereas the stop time initiates transfer of the accumulated count into the BUF[31:0] register for use by the comparators and software.

Note: Longer time windows provide more available time for the counter to accumulate monitored clocks, resulting in higher precision.

Note: Choosing the source of the reference clock wisely can significantly improve the performance of the module. If monitored clock frequency is significantly higher than that of the reference clock or frequencies are close to each other, then performance may be affected.

Figure 12-11. Internal Timing Diagram – WINPR[31:0] = 1



Note: Synchronization is intended for illustration only and may not be drawn to scale per actual design implementation.

Note:
Upon successful count value capturing into the Data Buffer register:

- Count ready interrupt output based on the capturing of the counter invoked
- BUFV set high

12.4.7.2 Detection on Monitored Clock

The clock monitor can detect faulty conditions of both the clock sources being monitored, and it can also be used as the reference.

12.4.7.2.1 Frequency Drift Detection

The captured count value enters the comparator logic block, and its value is simultaneously compared against the contents of four limit registers. This essentially forms four independent thresholds confining the monitored clock frequency's deviation into two acceptable ranges defined by two pairs of high/low limit registers. Monitored clock frequency drift is, therefore, detectable per user defined tolerance. Upon detection of a threshold violation, the user is notified via an interrupt event.

- Clock failure interrupt output based on fail threshold limit and catastrophic failures invoked
- ON bit cleared
- Clock Fail Event signal is provided for system

12.4.7.2.2 Catastrophic Failure Detection

Monitored Clock Failure

The monitored clock experiences a catastrophic failure and decreases its toggling rate significantly or ceases to toggle completely.

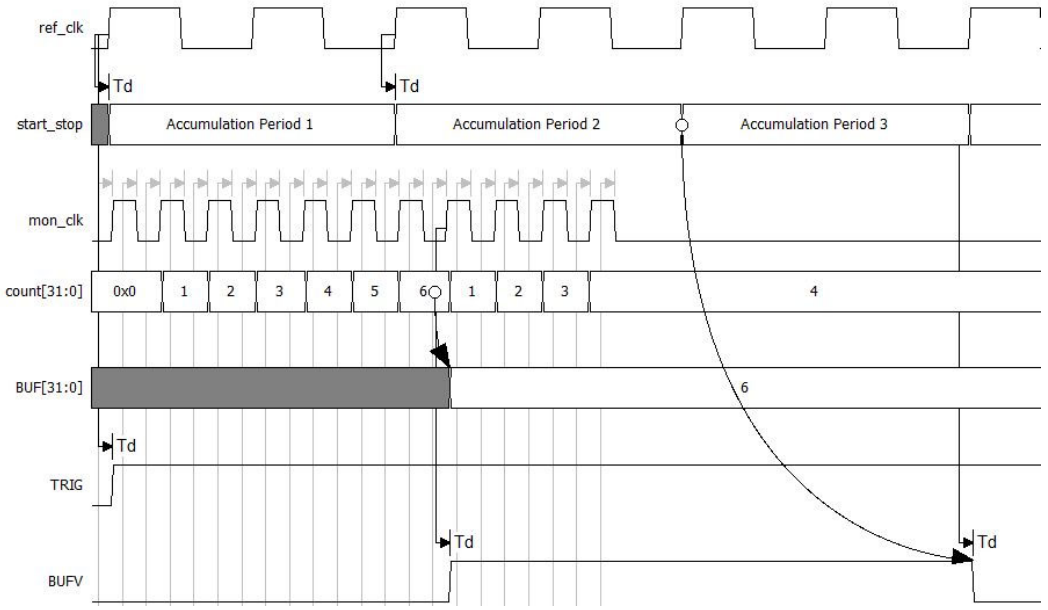
The total latency of the monitored clock frequency drift detection does not only depend on the duration of the accumulation time window on which it operates, it also depends on the latency of its clock domain crossing logic.

Detection happens on timely updates of the Data Buffer register with the captured count as the periodic accumulation time expires. At the next accumulation time expiration, if BUF[31:0] has not been updated, the monitored clock is considered to have experienced a catastrophic failure event given that the reference clock is still toggling reliably.

The module's response is similar to that of the clock frequency drifting beyond the user defined catastrophic tolerance limit described above. By asserting a clock fail event, the clock monitor module provides the system with a fast hardware response time often required to deal with a clock failure in control loop applications. The clock fail event triggers the clock fail interrupt with the conditions reflected below:

- Clock failure interrupt output based on fail threshold limit and catastrophic failures invoked
- ON bit cleared
- Clock Fail Event signal is provided for system

Figure 12-12. Catastrophic Failure Detection on Monitored Clock — Missing Edges on MON_CLK



Reference Clock Failure

The clock monitor module can detect reference clock failure; the reference clock can fail in the following conditions:

1. If the selected reference clock is completely missing from the start, making the signaling event representing the start of the accumulation time window absent, the TRIG bit remains clear as a result. This condition can be captured by the user software's observing the logic value of this bit once the system has booted up - no interrupt and/or other signaling event generated.
2. However, if the reference clock starts out toggling, setting the TRIG bit high, but subsequently slows down significantly or ceases to toggle completely at some point, the signaling event representing the start/stop of the accumulation time window is also affected by the same proportion. This condition effectively extends the accumulation time window causing the

counter to eventually saturate before capturing in some cases. As a result, the SATD bit is set high - interrupt invoked.

Note: SATD being set high does not exclusively reflect this condition. Also, the reference clock is required to set SATD.

12.4.7.3 Measurement Function

The clock monitor module is also capable of functioning as a frequency measurement unit, usually when employed as a standalone peripheral.

12.4.7.3.1 Software Assisted Frequency Measurement

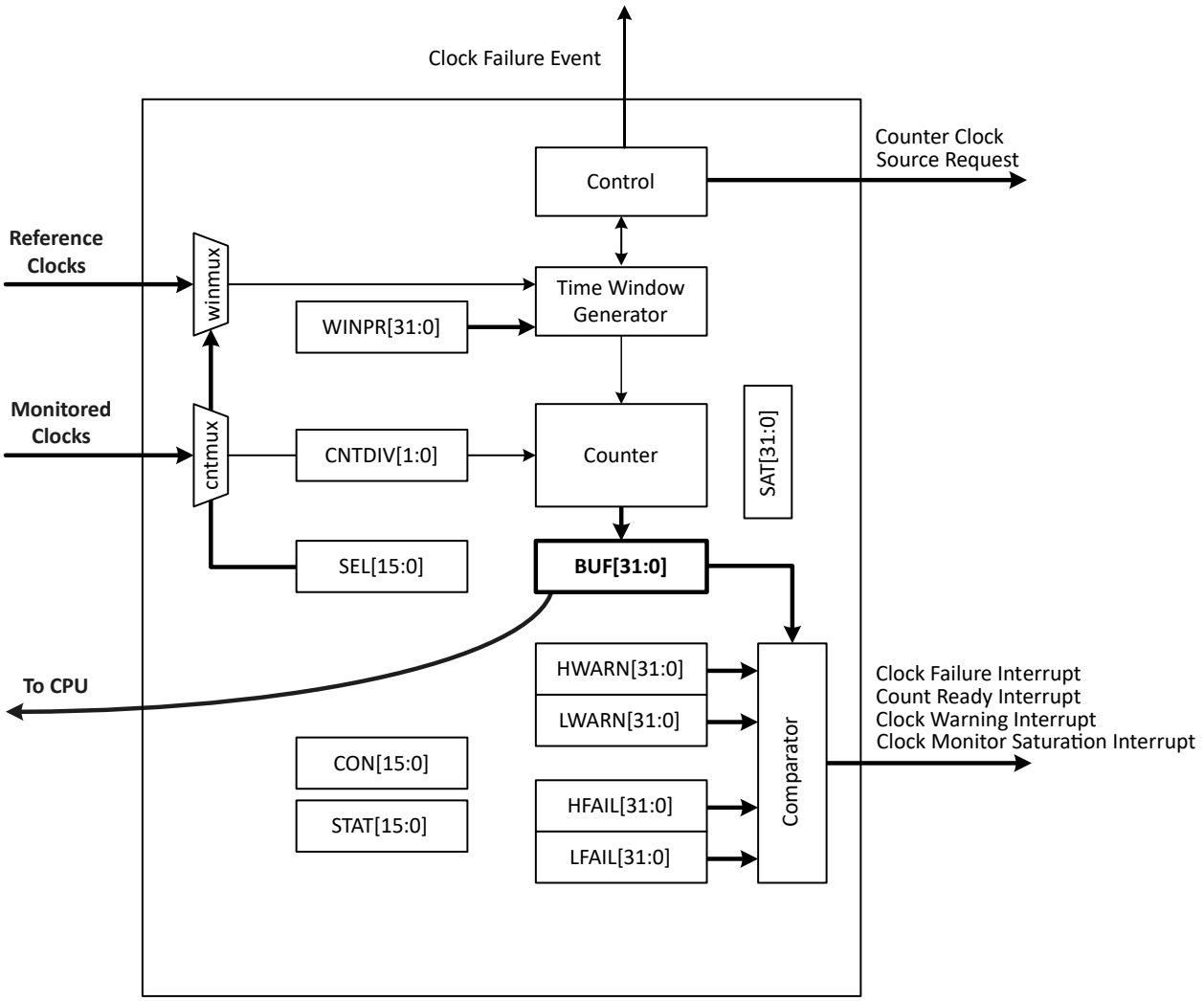
By allowing the CPU access to the accumulated results, firmware can convert the measured counts into frequencies based on existing knowledge of the reference clock frequency or period. The captured count's corresponding register, CMxBUF, is therefore memory mapped in this peripheral. Figure 12-13 shows the basic block diagram of this mode of operation.

Clocking Configurations

Reference Clock ≥ Time Window Generator

Monitored Clock ≥ Counter

Figure 12-13. Frequency Measurement Function – CPU



If Reference clock is Clock generator 6 FRC 8 MHz Monitor Clock is Clock generator 5 EC 8 MHz Accumulation Time = 0x8000 cycles of reference clock. Monitor clock is divided by 2, its 4MHz Time window = 0x8000/8M = 4ms

So, for 4ms with 4MHz clock speed what is the expected accumulation.

0.004096 * 4M = 0x4000

The CMxBUF is expected to hold 0x4000 value on every time window interval.

If Reference clock is Clock generator 6

FRC 8MHz

Monitor Clock is Clock generator 5

EC 8MHz

Accumulation Time = 0x8000 cycles of reference clock

Monitor clock is divided by 2, its 4MHz

Time window = 0x8000/8M = 4ms

So, for 4ms with 4MHz clock speed what is the expected accumulation.

0.004096 * 4M = 0x4000

The CMxBUF is expected to hold 0x4000 value on every time window interval.

Example 12-5. Frequency Measurement Function

```
//Configure clock generators before enabling monitoring

   IFS0bits.C2FAILIF = 0;     // enable clock Monitor2 failure interrupt
   IEC0bits.C2FAILIE = 1;

   IFS0bits.C2MONIF = 0;     // enable clock saturation failure interrupt
   IEC0bits.C2MONIE = 1;

   IFS0bits.C2WARMIF = 0;     // enable clock Warning failure interrupt
   IEC0bits.C2WARMIE = 1;

   IFS0bits.C2RDYIF = 0;     // enable clock ready  failure interrupt
   IEC0bits.C2RDYIE = 1;

   CM2CONbits.ON = 0;        // disable clock monitor

   //Select monitor clock source
   CM2SELbits.CNTSEL = 4;    //Monitoring clock clock-gen-5
   //[0] = clkgen_clk[1] = System clocks
   //[1] = clkgen_clk[2]
   //...
   //[10] = clkgen_clk[11]
   //[11] = pll_fout_clk[1]
   //[12] = pll_vcodiv_clk[1]
   //[13] = pll_fout_clk[2]
   //[14] = pll_vcodiv_clk[2]
   //[15] = 1?b0 (reserved)
   //[16] = Serial Test Mode clock (PGC)
   //[17] = FRC - Internal 8 MHz RC oscillator
   //[18] = BFRC - Internal Backup 8 MHz RC oscillator
   //[19] = POSC - Primary crystal oscillator (4-32 MHz)
   //[25] = REFI1 - user definable clock source
   //[26] = REFI2 - user definable clock source

   //Select reference clock source
   CM2SELbits.WINSEL = 5; //reference clock clock-gen-6
   //[0] = clkgen_clk[1] = System clocks
   //[1] = clkgen_clk[2]
   //...
   //[10] = clkgen_clk[11]
   //[11] = pll_fout_clk[1]
   //[12] = pll_vcodiv_clk[1]
```

```
    //[13] = pll_fout_clk[2]
    //[14] = pll_vcodiv_clk[2]
    //[15] = 1?b0 (reserved)
    //[16] = Serial Test Mode clock (PGC)
    //[17] = FRC - Internal 8 MHz RC oscillator
    //[18] = BFRC - Internal Backup 8 MHz RC oscillator
    //[19] = POSC - Primary crystal oscillator (4-32 MHz)
    //[25] = REFI1 - user definable clock source
    //[26] = REFI2 - user definable clock source


    //Counter Divider Selection
    CM2CONbits.CNTDIV = 1;
    //    10 = Divide-by 4
    //    01 = Divide-by 2
    //    00 = Divide-by 1

    CM2WINPR = 0x8000; //monitor clock pre scale
    //Accumulation Time (in cycles) = WINPR[31:0] + 1

    CM2HFAIL = 0x4500;     //CLOCK MONITOR HIGH THRESHOLD FAILING
    CM2LFAIL = 0x3500;     //CLOCK MONITOR LOW THRESHOLD FAILING

    CM2HWARN = 0x4100;     //CLOCK MONITOR HIGH THRESHOLD WARNING
    CM2LWARN = 0x3600;     //CLOCK MONITOR LOW THRESHOLD WARNING

    CM2SAT = 0x5000;      //CLOCK MONITOR COUNTER SATURATION


    CM2CONbits.WIDTH = 0;    // 0 - Frequency Measurement
                             // 1 - Pulse Width and Duty Cycle Measurement

    CM2CONbits.ON = 1;     //enable clock monitor
    while (CM2CONbits.ON);
```

12.4.7.3.2 Pulse Width and Duty Cycle Measurement

The clock monitored module is also capable of measuring the pulse width of the monitored clock source. By allowing only the high pulse time period of the monitored clock source to define the accumulation time window, see WIDTH, the pulse width can be measured in the number of reference clock cycles. This is given that the reference clock frequency is higher than the monitored clock frequency, so the operation is similar to that of the CCP module.

Clocking Configurations:

Monitored Clock ≥ Time Window Generator

Reference Clock ≥ Counter

In pulse measurement, give slower clock to time window (reference clock) and faster clock to counter (monitor clock).

There is no accumulation window (WINPR= 0), Rising Edge to Next Falling Edge of the reference clock will be the window for CMxBUF to get loaded.

Example 12-6 shows LPRC as reference clock and FRC as monitor clock.

Each Rising Edge to Next Falling Edge of LPRC (32KHz) will be 15.625 µs.

15.625 µs * 8M = 0x007D is the expected count in CMxBUF register.

Similarly, clock period measurement can also be accomplished by allowing the time period of the monitored clock source to define the accumulation time window.

With the clock set-up given above, the pulse width or clock period represented by the number of reference clock cycles and captured at every accumulation lapsed time can simply be read out of the Data Buffer register, BUF[31:0], for further processing as follows:

- Conversion into unit of time
- Duty cycle calculation (with period measurement)

Table 12-9. Module Control

Function	Clock Source		Control
	win_clk[*:0]	cnt_clk[*:0]	ON/WIDTH
Monitor	reference	monitored	1/0
Measurement - SW	reference	reference	1/0
Measurement - PW	monitored	reference	1/1

Example 12-6. Pulse Width

```
IFS0bits.C2FAILIF = 0;     // enable clock Monitor2 failure interrupt
    IEC0bits.C2FAILIE = 1;

    IFS0bits.C2MONIF = 0;          // enable clock saturation failure interrupt
    IEC0bits.C2MONIE = 1;

    IFS0bits.C2WARMIF = 0;         // enable clock Warning failure interrupt
    IEC0bits.C2WARMIE = 1;

    IFS0bits.C2RDYIF = 0;         // enable clock ready  failure interrupt
    IEC0bits.C2RDYIE = 1;

    CM2CONbits.ON = 0;            // disable clock monitor

    //Select monitor clock source
    CM2SELbits.CNTSEL = 4;        // Monitoring clock clock-gen-5
    //[0] = cru_clkgen_clk[1] = System clocks
    //[1] = cru_clkgen_clk[2]
    //...
    //[10] = cru_clkgen_clk[11]
    //[11] = cru_pll_fout_clk[1]
    //[12] = cru_pll_vcodiv_clk[1]
    //[13] = cru_pll_fout_clk[2]
    //[14] = cru_pll_vcodiv_clk[2]
    //[15] = 1?b0 (reserved)
    //[16] = Serial Test Mode clock (PGC)
    //[17] = FRC - Internal 8 MHz RC oscillator
    //[18] = BFRC - Internal Backup 8 MHz RC oscillator
    //[19] = POSC - Primary crystal oscillator (4-32 MHz)
    //[20] = LPRC or BRFC/256
    //[25] = REFI1 - user definable clock source
    //[26] = REFI2 - user definable clock source

    //Select reference clock source
    CM2SELbits.WINSEL = 20; //reference clock LPRC
    //[0] = cru_clkgen_clk[1] = System clocks
    //[1] = cru_clkgen_clk[2]
    //...
    //[10] = cru_clkgen_clk[11]
    //[11] = cru_pll_fout_clk[1]
    //[12] = cru_pll_vcodiv_clk[1]
    //[13] = cru_pll_fout_clk[2]
    //[14] = cru_pll_vcodiv_clk[2]
    //[15] = 1?b0 (reserved)
    //[16] = Serial Test Mode clock (PGC)
    //[17] = FRC - Internal 8 MHz RC oscillator
    //[18] = BFRC - Internal Backup 8 MHz RC oscillator
    //[19] = POSC - Primary crystal oscillator (4-32 MHz)
    //[20] = LPRC or BRFC/256
    //[25] = REFI1 - user definable clock source
    //[26] = REFI2 - user definable clock source


    //Counter Divider Selection
    CM2CONbits.CNTDIV = 0;
    //    10 = Divide-by 4
    //    01 = Divide-by 2
    //    00 = Divide-by 1

    CM2WINPR = 0x0000; //monitor clock pre scale
    //Accumulation Time (in cycles) = WINPR[31:0] + 1

    CM2HFAIL = 0x90;         //CLOCK MONITOR HIGH THRESHOLD FAILING
```

```
CM2LFAIL = 0x70;          //CLOCK MONITOR LOW THRESHOLD FAILING

CM2HWARN = 0x85;          //CLOCK MONITOR HIGH THRESHOLD WARNING
CM2LWARN = 0x75;          //CLOCK MONITOR LOW THRESHOLD WARNING

CM2SAT = 0x100;          //CLOCK MONITOR COUNTER SATURATION


CM2CONbits.WIDTH = 1;    // 0 - Frequency Measurement
// 1 - Pulse Width and Duty Cycle Measurement

CM2CONbits.ON = 1;      //enable clock monitor
while (CM2CONbits.ON);
```

12.4.7.4 Write Protection

To avoid erroneous software, write which impact result of the module has false detection.

The LOCK and WREN are register bits located in module to control the register write.

Table 12-10. Write Access Control

LOCK	WREN	Access Permission
0	0	Registers write access disabled(1)
0	1	Registers write access enabled(2)
1	x	Registers write access disabled(1)

Notes:

1. Access restriction enforced until reset.
2. Default setting.

12.4.7.5 Fault Injection

The fault injection feature allows the user to inject an artificial fault into the module and observe the expected response to ensure the module's proper operation. To be effective, fault injection is required to be performed randomly and periodically with minimal interference to the module's normal operation. The module is not disabled upon detection of any fault injection.

12.4.7.5.1 Catastrophic Fault Injection

Catastrophic fault injection is used to confirm that the module is capable of detecting the condition where the monitored clock completely ceases to toggle, also known as catastrophic failure, discussed earlier.

When the artificial catastrophic fault is injected into the module, the module's counter no longer gets clocked by the selected monitored clock. It is internally driven to ground via hardware. As a result, the counter is unable to continue accumulating, and the current count value stalls inside the counter when the function is invoked. Without a valid clock edge, the counter is unable to transfer its accumulated count into the Data Buffer register at the end of the accumulation time window.

Catastrophic fault injection can be activated by setting FLTINJ[1:0] to 2'b11 at an arbitrary point, subject to LOCK/WREN.

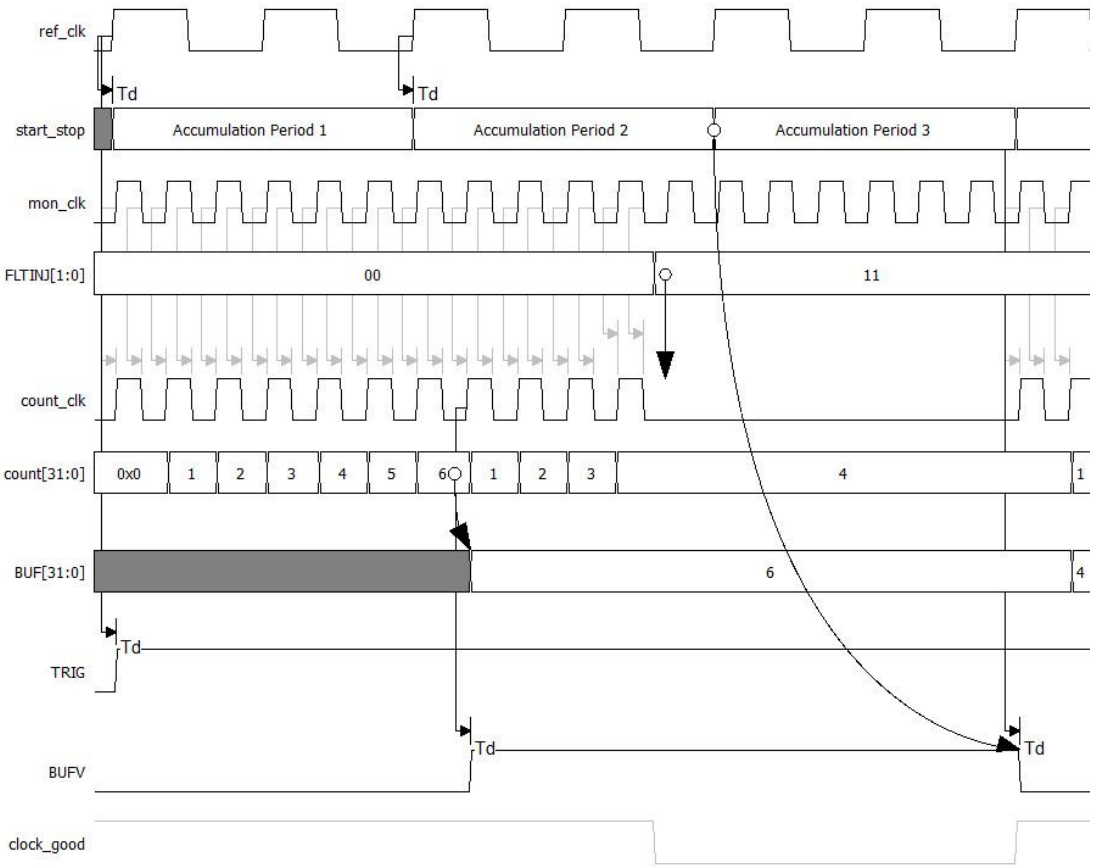
The module's response to catastrophic fault injection:

- Clock failure interrupt output based on fail threshold limit and catastrophic failures invoked
- ON bit is NOT cleared
- Clock Fail Event signal is NOT provided for the system

Notes:

1. FLTINJ[1:0] bits are not self-cleared by hardware. They maintain their programmed value until cleared by software to assist the ISR handler with its discovery process.
2. A persisting real fault is detected in the following accumulation cycle regardless of FLTINJ[1:0] bits being cleared.

Figure 12-14. Catastrophic Fault Injection



12.4.7.5.2 Low Frequency Drift Fault Injection

To mimic low frequency drift, the selected monitored clock input is purposely divided down by two before being allowed to clock the counter, resulting in lowering the final accumulation value to half the expected figure.

The count value, once captured, is then compared with the contents of the four threshold limit registers to determine its current deviation against the set criteria.

As expected,

- If BUF[31:0] < LWARN[31:0], LWT interrupt flag is set high with clock warning interrupt output based on the warning threshold limit, active high invoked.
- If BUF[31:0] < LFAIL[31:0], LFT interrupt flag is also set high with clock failure interrupt output based on fail threshold limit and catastrophic failures invoked.

Low frequency drift fault injection can be activated by setting FLTINJ[1:0] to 2'b01 at random, subject to LOCK/WREN.

Notes:

1. FLTINJ[1:0] bits are not self-cleared by hardware. They maintain their programmed value until cleared by software to assist the ISR handler with its discovery process.
2. Detection of this artificial fault will not occur if the limit tolerances are set too high. The function is designed for the detection of 50% margin below nominal or tighter.

12.4.7.5.3 High Frequency Drift Fault Injection

To mimic high frequency drift, the selected reference clock input is purposely divided down by two before entering the time window generator, thus allowing twice as much time for the counter to accumulate its count and double the expected figure.

The count value, once captured, is then compared against the contents of the four threshold limit registers to determine its current deviation against the set criteria. As expected:

- If BUF[31:0] > HWARN[31:0], HWT interrupt flag is set high with clock warning interrupt output based on the warning threshold limit invoked.
- If BUF[31:0] > HFAIL[31:0], HFT interrupt flag is also set high with clock failure interrupt output based on fail threshold limit and catastrophic failures invoked.
- If the counter saturates before the current accumulation time period ends, SATD status bit is set high with clock monitor saturation interrupt output based on the counter having saturated

High frequency drift fault injection can be activated by setting FLTINJ[1:0] to 2'b10 at random, subject to LOCK/WREN

Notes:

1. FLTINJ[1:0] bits are not self-cleared by hardware. They maintain their programmed value until cleared by software to assist the ISR handler with its discovery process.
2. Detection of this artificial fault will not occur if the limit tolerances are set too high. The function is designed for the detection of 100% margin above nominal or tighter.

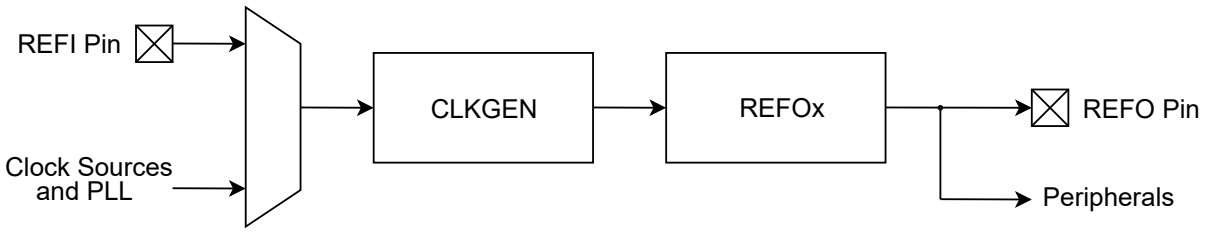
Table 12-11. Module Control

Fault	Clock Connection		Control
	win_clk[*:0]	cnt_clk[*:0]	ON/WIDTH/FLTINJ[1:0]
Catastrophic Fault Injection	reference	monitored	1/0/11
Low Drift Fault Injection	reference	monitored	1/0/01
High Drift Fault Injection	reference	monitored	1/0/10

12.4.8 Reference Clock Output (REFOx)

The dsPIC33AK128MC106 family features one or more reference clock output (REFOx) modules. The REFOx module clock source is one of the CLKGENs (see Table 12-2 for assignments of CLKGENs to REFOx). The input clock to the REFOx is therefore the clock source of the associated CLKGENx, which can be any of the sources defined in Table 12-3. This selection includes reference clock inputs (REFIx) that are mappable to device pins using peripheral pin select (PPS). The REFO output signal is internally routed to select peripherals and can be routed to a device pin using PPS. Figure 12-15 illustrates REFO interconnections.

Figure 12-15. CLKGEN to REFO Assignments



12.4.9 Power-Saving Mode

Oscillator module supports multiple Sleep and Idle modes for power reduction:

For clock generator 1

- Idle mode - System clock stops, peripheral clock, peripheral clock divided run
- Sleep mode - System clock, peripheral clock, peripheral clock divided stop

Other clock and PLL generators

- Stop in Idle - CLKGEN/PLL(x) stop in Idle
- Run in Sleep - CLKGEN/PLL(x) run in Sleep

The Stop in Idle (SIDL) and Run In Sleep (RIS) options are user selectable via the SIDLE and RIS bits in the CLKxCON/PLLCONx registers.

If the SIDLE bit is set in a CLKxCON/PLLCONx register, the associated clock will stop if CPU IDLE is asserted. The SIDLE bit has no effect if CPU Sleep is asserted.

If the RIS bit is set in a CLKxCON/PLLCONx register, the associated clock will continue to run even if CPU Sleep is asserted. The RIS bit has no effect if CPU Idle is asserted.

Clock generator 1 is the clock source for the system clock (sys_clk) and peripheral clock. The PWRSAV instruction that initiates Idle and Sleep mode is primarily responsible for reducing processor and system power consumption.

In Sleep and Idle mode, the processor clock, system and fast peripheral clock, and peripheral clock are all disabled

13. Direct Memory Access (DMA) Controller

The Direct Memory Access (DMA) controller is designed to handle high data throughput peripherals on the SFR bus by enabling direct access to data memory to reduce the need for intensive CPU management. The DMA controller is structured with multiple channels, each of which can be connected to a selectable peripheral module. When a peripheral module triggers its interrupt, the corresponding DMA channel responds by accessing the SRAM according to its programming without requiring CPU intervention. This direct access not only frees up the CPU to handle other tasks, but also optimizes overall system efficiency.

Each DMA channel can interrupt the CPU once the DMA session is complete, or if other interrupt conditions are met. This mechanism ensures that the CPU is only engaged when necessary, enhancing system performance and making efficient use of resources. Additionally, this setup helps decrease dynamic power dissipation, especially in applications with lower data throughput demands. By offloading the data transfer tasks from the CPU, the DMA controller significantly contributes to improved system functionality and energy efficiency in a variety of applications.

The DMA Controller has these features:

- Six Independent Channels
- Concurrent Operation with the CPU (No DMA Caused Wait States)
- DMA Bus Arbitration Using Fixed Priority and Round Robin Scheme
- Four Address Modes
- Four Transfer Modes
- Ping-Pong Mode (Automatic Switch Between Two Channels After Each Block Transfer Completes)
- 8-Bit, 16-Bit or 32-Bit Word Support for Data Transfer
- 24-Bit Source and Destination Address Register for Each Channel, Dynamically Updated and Independently Reloadable
- 32-Bit Transaction Count Register, Dynamically Updated and Independently Reloadable
- Upper and Lower Address Limit Registers
- Counter Half/Full Level Interrupt
- Software Triggered Transfer
- Null Write Mode for Symmetric Buffer Operations
- DMA Request for Each Channel can be Selected From Any Supported Interrupt Source
- Support for Daisy Chaining of Channel Called Channel Chaining (One Channel Triggered by Another Channel)
- Set/Clear/Invert Bit Manipulation Capability
- Pattern Match
- Bus Read/Write Error Fault Indication

13.1 Device-Specific Information

Table 13-1. DMA Summary Table

DMA Module Instances	Channels per Instance	Clock Source	Peripheral Bus Speed
1	6	Standard Speed Peripheral Clock	Standard

Table 13-2. DMA Channel Trigger Sources

CHSEL[7:0]	Trigger (Interrupt)	CHSEL[7:0]	Trigger (Interrupt)	CHSEL[7:0]	Trigger (Interrupt)
00h	INT0 – External Interrupt 0	23h	PCG1 – PWM Generator 1	44h	ADC2 – AN4
01h	INT1 – External Interrupt 1	24h	PCG2 – PWM Generator 2	45h	ADC2 – AN5
02h	INT2 – External Interrupt 2	25h	PCG3 – PWM Generator 3	46h	ADC2 – AN6
03h	NVM – NVM Write complete	26h	PCG4 – PWM Generator 4	47h	ADC2 – AN7
04h	CRC – CRC Generator Interrupt	27h	(Reserved, do not use)	48h	ADC2 – AN8
05h	TMR1 – Timer 1 interrupt	28h	SENT1	49h	ADC2 – AN9
06h	SPI1RX – SPI 1 Receiver	29h	SENT2	4Ah	ADC2 – AN10
07h	SPI1TX – SPI 1 Transmitter	2Ah	BiSS	4Bh	ADC2 – AN11
08h	SPI2RX – SPI 2 Receiver	2Bh	ADC1 – AN0	4Ch	ADC2 – AN12
09h	SPI2TX – SPI 2 Transmitter	2Ch	ADC1 – AN1	4Dh	ADC2 – AN13
0Ah	SPI3RX – SPI 3 Receiver	2Dh	ADC1 – AN2	4Eh	ADC2 – AN14
0Bh	SPI3TX – SPI 3 Transmitter	2Eh	ADC1 – AN3	4Fh	ADC2 – AN15
0Ch-0Eh	(Reserved, do not use)	2Fh	ADC1 – AN4	50h	ADC2 – AN16
0Fh	U1RX – UART1 Receiver	30h	ADC1 – AN5	51h	ADC2 – AN17
10h	U1TX – UART1 Transmitter	31h	ADC1 – AN6	52h	ADC2 – AN18
11h	U2RX – UART2 Receiver	32h	ADC1 – AN7	53h	ADC2 – AN19
12h	U2TX – UART2 Transmitter	33h	ADC1 – AN8	54h-5Ah	(Reserved, do not use)
13h	U3RX – UART3 Receiver	34h	ADC1 – AN9	5Bh	DMA0 Interrupt
14h	U3TX – UART3 Transmitter	35h	ADC1 – AN10	5Ch	DMA1 Interrupt
15h	U4RX – UART4 Receiver	36h	ADC1 – AN11	5Dh	DMA2 Interrupt
16h	U4TX – UART4 Transmitter	37h	ADC1 – AN12	5Eh	DMA3 Interrupt
17h	(Reserved, do not use)	38h	ADC1 – AN13	5Fh	DMA4 Interrupt

.....continued

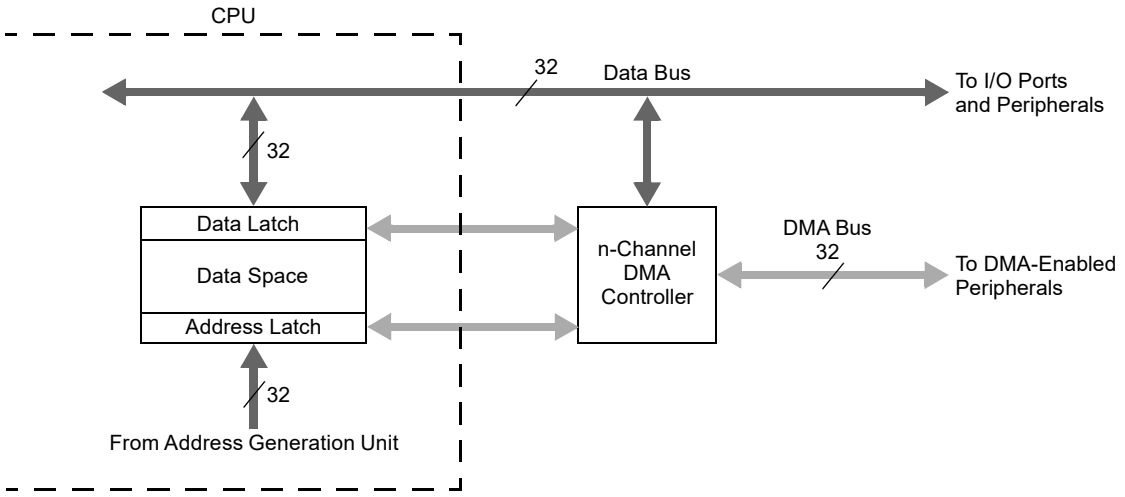
CHSEL[7:0]	Trigger (Interrupt)	CHSEL[7:0]	Trigger (Interrupt)	CHSEL[7:0]	Trigger (Interrupt)
18h	CCP1 IC/OC – Input Capture/ Output Compare	39h	ADC1 – AN14	60h	DMA5 Interrupt
19h	CCP2 IC/OC – Input Capture/ Output Compare	3Ah	ADC1 – AN15	61h-6Ah	(Reserved, do not use)
1Ah	CCP3 IC/OC – Input Capture/ Output Compare	3Bh	ADC1 – AN16	6Bh	I2C1 – Generic Interrupt
1Bh	CCP4 IC/OC – Input Capture/ Output Compare	3Ch	ADC1 – AN17	6Ch	I2C1 – Receive Buffer Interrupt
1Ch	CCP5 IC/OC – Input Capture/ Output Compare	3Dh	ADC1 – AN18	6Dh	I2C1 – Transmit Buffer Interrupt
1Dh	CCP6 IC/OC – Input Capture / Output Compare	3Eh	ADC1 – AN19	6Eh	I2C2 – Generic Interrupt
1Eh	CCP7 IC/OC – Input Capture/ Output Compare	3Fh	(Reserved, do not use)	6Fh	I2C2 – Receive Buffer Interrupt
1Fh	CCP8 IC/OC – Input Capture/ Output Compare	40h	ADC2 – AN0	70h	I2C2 – Transmit Buffer Interrupt
20h	(Reserved, do not use)	41h	ADC2 – AN1	71h-74h	(Reserved, do not use)
21h	PWM EVENT A	42h	ADC2 – AN2		
22h	PWM EVENT B	43h	ADC2 – AN3		

13.2 Architectural Overview

The DMA Controller functions both as a peripheral and a direct extension of the CPU. It is located on the microcontroller data bus between the CPU and DMA-enabled peripherals, with direct access to data space (Figure 13-1). This partitions the SFR bus into two buses, allowing the DMA Controller access to the DMA-capable peripherals located on the new DMA SFR bus. This also lowers bus loading for less power consumption per access. The controller serves as a host device on the DMA SFR bus, controlling data flow from DMA-capable peripherals.

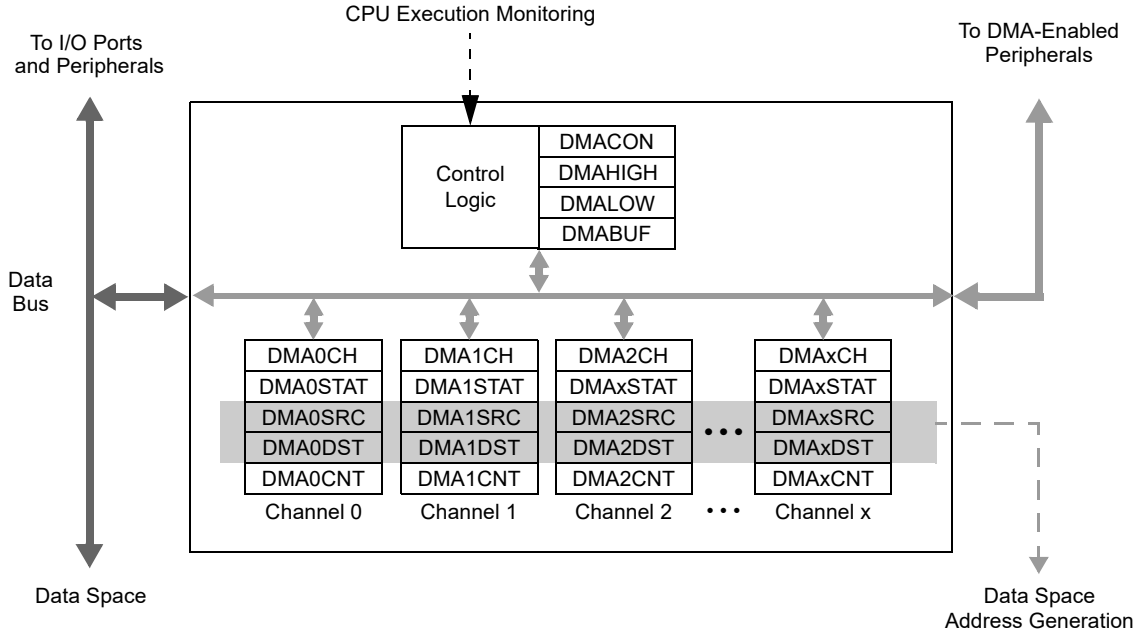
When the CPU is servicing peripherals that are not on the DMA bus, the DMA Controller is free to service peripherals on the DMA bus while the CPU is performing its operations. In this way, the effective bandwidth for handling data is increased. At the same time, DMA operations can proceed without causing a processor Stall. When the CPU and DMA are accessing the SFR simultaneously, the CPU gets priority and the DMA will have to wait until the CPU completes the task.

Figure 13-1. DMA Location Block Diagram



The DMA Controller itself is composed of multiple independent DMA channel controllers, or simply channels (Figure 13-2). Each channel can be independently programmed to transfer data between different areas of the data space, move data between single or multiple addresses, use a wide range of hardware triggers to initiate transfers and conduct programmed transactions once or many times. Multiple channels may even be programmed to work together to carry out more complex data transfers without CPU intervention. The top-level controller sets the boundary addresses for all DMA operations, regardless of the channel. It also arbitrates data bus access between the channels based on a user-selectable priority scheme and determines how DMA will operate in power-saving modes.

Figure 13-2. DMA Channel Controllers Block Diagram



13.3 DMA Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2300	DMACON	31:24								
		23:16								
		15:8	ON		SIDL					
		7:0								PRIORITY
0x2304	DMABUF	31:24	DMABUF[31:24]							
		23:16	DMABUF[23:16]							
		15:8	DMABUF[15:8]							
		7:0	DMABUF[7:0]							
0x2308	DMALOW	31:24								
		23:16	LADDR[23:16]							
		15:8	LADDR[15:8]							
		7:0	LADDR[7:0]							
0x230C	DMAHIGH	31:24								
		23:16	HADDR[23:16]							
		15:8	HADDR[15:8]							
		7:0	HADDR[7:0]							
0x2310	DMA0CH	31:24			PPEN	PCHAEN		RELOADC	RELOADD	RELOADS
		23:16								RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	INTOEN	MATEN	HALFEN	CHEN
0x2314	DMA0SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x2318	DMA0STAT	31:24								
		23:16								
		15:8							DMAFLT[3]	DMAFLT[2]
		7:0	DMAFLT[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
0x231C	DMA0SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							
0x2320	DMA0DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x2324	DMA0CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x2328	DMA0CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							
0x232C	DMA0SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							
0x2330	DMA0INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x2334	DMA0MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2338	DMA0PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							
0x233C	DMA1CH	31:24			PPEN	PCHAEN		RELOADC	RELOADD	RELOADS
		23:16								RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	INTOEN	MATEN	HALFEN	CHEN
0x2340	DMA1SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x2344	DMA1STAT	31:24								
		23:16								
		15:8							DMAFLT[3]	DMAFLT[2]
		7:0	DMAFLT[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
0x2348	DMA1SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							
0x234C	DMA1DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x2350	DMA1CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x2354	DMA1CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							
0x2358	DMA1SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							
0x235C	DMA1INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x2360	DMA1MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							
0x2364	DMA1PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							
0x2368	DMA2CH	31:24			PPEN	PCHAEN		RELOADC	RELOADD	RELOADS
		23:16								RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	INTOEN	MATEN	HALFEN	CHEN
0x236C	DMA2SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x2370	DMA2STAT	31:24								
		23:16								
		15:8							DMAFLT[3]	DMAFLT[2]
		7:0	DMAFLT[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2374	DMA2SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							
0x2378	DMA2DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x237C	DMA2CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x2380	DMA2CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							
0x2384	DMA2SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							
0x2388	DMA2INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x238C	DMA2MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							
0x2390	DMA2PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							
0x2394	DMA3CH	31:24			PPEN	PCHAEN		RELOADC	RELOADD	RELOADS
		23:16								RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	INTOEN	MATEN	HALFEN	CHEN
0x2398	DMA3SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x239C	DMA3STAT	31:24								
		23:16								
		15:8							DMAFLT[3]	DMAFLT[2]
		7:0	DMAFLT[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
0x23A0	DMA3SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							
0x23A4	DMA3DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x23A8	DMA3CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x23AC	DMA3CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x23B0	DMA3SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							
0x23B4	DMA3INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x23B8	DMA3MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							
0x23BC	DMA3PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							
0x23C0	DMA4CH	31:24			PPEN	PCHAEN		RELOADC	RELOADD	RELOADS
		23:16								RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	INTOEN	MATEN	HALFEN	CHEN
0x23C4	DMA4SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x23C8	DMA4STAT	31:24								
		23:16								
		15:8							DMAFLT[3]	DMAFLT[2]
		7:0	DMAFLT[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
0x23CC	DMA4SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							
0x23D0	DMA4DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x23D4	DMA4CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x23D8	DMA4CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							
0x23DC	DMA4SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							
0x23E0	DMA4INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x23E4	DMA4MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							
0x23E8	DMA4PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x23EC	DMA5CH	31:24			PPEN	PCHAEN		RELOADC	RELOADD	RELOADS
		23:16								RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	INTOEN	MATEN	HALFEN	CHEN
0x23F0	DMA5SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x23F4	DMA5STAT	31:24								
		23:16								
		15:8							DMAFLT[3]	DMAFLT[2]
		7:0	DMAFLT[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
0x23F8	DMA5SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							
0x23FC	DMA5DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x2400	DMA5CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x2404	DMA5CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							
0x2408	DMA5SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							
0x240C	DMA5INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x2410	DMA5MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							
0x2414	DMA5PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							

13.3.1 DMA Module Control Register

Name: DMACON
Offset: 0x2300

Legend: r = Reserved bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON		SIDL					
Access	R/W		R/W					
Reset	0		0					

Bit	7	6	5	4	3	2	1	0
								PRIORITY
Access								R/W
Reset								0

Bit 15 – ON DMA Module Enable bit

Value	Description
1	Enables DMA module
0	Disables DMA module. When set low, all state machines are reset, resulting in immediate termination of all active DMA operation(s); however, the contents of the control registers are NOT reset to their default settings.

Bit 13 – SIDL DMA Stop in Idle bit

Value	Description
1	When system enters Idle mode, module stops operation
0	When system enters Idle mode, module continues operation

Bit 0 – PRIORITY Channel Priority Scheme Selection bit

Value	Description
1	Round robin scheme
0	Fixed priority scheme

13.3.2 DMA Data Buffer Register

Name: DMABUF
Offset: 0x2304

Bit	31	30	29	28	27	26	25	24
	DMABUF[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DMABUF[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DMABUF[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DMABUF[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DMABUF[31:0] Internal Data Buffer bits
These bits reflect the content of the internal data buffer that the DMA uses to hold the data being moved from the Source Address to the Destination Address.

13.3.3 DMA Low Address Limit Register

Name: DMALOW
Offset: 0x2308

Note:

1. The LADDR[23:0] bits apply to data space only. SFR space outside the range from LADDR[23:0] to HADDR[23:0] is accessible by the DMA and will not result in an interrupt. Setting LADDR[23:0] to an address in SFR will, therefore, be ignored.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	LADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – LADDR[23:0] Low Limit Address bits(1)
These bits indicate the lower address location below which the DMA module initiates a transaction, which results in the associated channel interrupt, DMAFLT[1:0] = `01` and CHEN is cleared.

13.3.4 DMA High Address Limit Register

Name: DMAHIGH
Offset: 0x230C

Note:

1. The HADDR[23:0] bits apply to data space only. SFR space outside the range from LADDR[23:0] to HADDR[23:0] is accessible by the DMA and will not result in an interrupt. Setting HADDR[23:0] to an address in SFR will, therefore, be ignored.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	HADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – HADDR[23:0] High Limit Address bits(1)

These bits indicate the upper address location beyond which the DMA module initiates a transaction, which results in the associated channel interrupt, DMAFLT[1:0] = `10` and CHEN is cleared.

13.3.5 DMA Channel x Control Register

Name: DMAxCH
Offset: 0x2310, 0x233C, 0x2368, 0x2394, 0x23C0, 0x23EC

Legend: HS = Hardware Settable bit, HC = Hardware Clearable bit

Notes:

1. The number of transfers per CHREQ bit setting depends on TRMODE[1:0].
2. The channel will only be able to transfer if PCHAEN is high, provided that PPEN is also set high for ping-pong operation support.
3. CNT[31:0] are reloaded in every repeated operation regardless of RELOADC.

Bit	31	30	29	28	27	26	25	24
			PPEN	PCHAEN		RELOADC	RELOADD	RELOADS
Access			R/W	R/W/HS/HC		R/W	R/W	R/W
Reset			0	0		0	0	0

Bit	23	22	21	20	19	18	17	16
								RETEN
Access								R/W
Reset								0

Bit	15	14	13	12	11	10	9	8
	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SIZE[1:0]			CHREQ	INTOEN	MATEN	HALFEN	CHEN
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W/HC
Reset	0	0		0	0	0	0	0

Bit 29 – PPEN Ping-Pong Operation Support Enable bit (refer to 13.4.11. Ping-Pong)

Value	Description
1	Ping-pong operation support is enabled
0	Ping-pong operation support is disabled

Bit 28 – PCHAEN Ping-Pong Channel Enable bit (refer to 13.4.11. Ping-Pong)
Intended to support the ping-pong operation; the bit takes effect only when PPEN is set high as follows.

Value	Description
1	The DMA channel is enabled if CHEN is set high; this bit is hardware settable via PCHAEN of the other DMA channel pair used in the Ping-Pong mode
0	The DMA channel is disabled

Bit 26 – RELOADC CNT[31:0] Reload bit(3)

Value	Description
1	CNT[31:0] are reloaded to their previously written value (buffered original content) upon the start of the next operation
0	CNT[31:0] are not reloaded

Bit 25 – RELOADD DADDR[23:0] Reload bit

Value	Description
1	DADDR[23:0] are reloaded to their previously written value upon the start of the next operation
0	DADDR[23:0] are not reloaded

Bit 24 – RELOADS SADDR[23:0] Reload bit

Value	Description
1	SADDR[23:0] are reloaded to their previously written value upon the start of the next operation
0	SADDR[23:0] are not reloaded

Bit 16 – RETEN Read Error Trap Enable bit

Value	Description
1	DMA channel suspends transfer operation on bus read error and asserts DMA trap signal
0	DMA channel continues transfer operation when bus read error is encountered

Bits 15:14 – SAMODE[1:0] Source Address Mode Selection bits

Value	Description
11	Reserved
10	SADDR[23:0] are decremented based on SIZE[1:0] after a transfer completion
01	SADDR[23:0] are incremented based on SIZE[1:0] after a transfer completion
00	SADDR[23:0] remain unchanged after a transfer completion

Bits 13:12 – DAMODE[1:0] Destination Address Mode Selection bits

Value	Description
11	Reserved
10	DADDR[23:0] are decremented based on SIZE[1:0] after a transfer completion
01	DADDR[23:0] are incremented based on SIZE[1:0] after a transfer completion
00	DADDR[23:0] remain unchanged after a transfer completion

Bits 11:10 – TRMODE[1:0] Transfer Mode Selection bits

Value	Description
11	Repeated Continuous
10	Continuous
01	Repeated One-Shot
00	One-Shot

Bits 9:8 – FLWCON[1:0] Data Flow Control bits

Value	Description
11	Reserved
10	Read from SADDR[23:0] only
01	Read from SADDR[23:0] followed by write to DADDR[23:0] and SADDR[23:0] (Null Write Mode)
00	Read from SADDR[23:0] followed by write to DADDR[23:0]

Bits 7:6 – SIZE[1:0] Data Size Selection bits

Value	Description
11	Reserved
10	One 32-bit word is transferred at a time
01	One 16-bit word is transferred at a time
00	One byte word is transferred at a time

Bit 4 – CHREQ DMA Channel Software Request bit(1)
This bit is automatically reset to '0' upon completion of a DMA transfer.

Value	Description
1	One DMA request is initiated by software
0	No DMA request is initiated by software

Bit 3 – INTOEN Interrupt Output Enable bit

Value	Description
1	An interrupt is invoked upon DONE flag being set
0	An interrupt is not invoked by DONE condition

Bit 2 – MATEN Pattern Match Enable bit

Value	Description
1	Pattern match is enabled
0	Pattern match is disabled

Bit 1 – HALFEN 50% Completion Watermark bit

Value	Description
1	An interrupt is invoked when the CNT[31:0] counter has reached its halfway point
0	An interrupt is not invoked by a Half condition

Bit 0 – CHEN DMA Channel Enable bit **(2)**

Value	Description
1	The corresponding DMA channel is enabled
0	The corresponding DMA channel is disabled

13.3.6 DMA Channel x Selection Register

Name: DMAxSEL
Offset: 0x2314, 0x2340, 0x236C, 0x2398, 0x23C4, 0x23F0

Note:

1. Unused CHSEL bits should be unimplemented (U-0).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	CHSEL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CHSEL[7:0] DMA Channel Trigger Selection bits(1)
These bits select one of the possible DMA triggers connected to the corresponding channel's input. See 13.4.3. DMA Trigger Sources for more DMA trigger source information.

13.3.7 DMA Channel x Interrupt Register

Name: DMAxSTAT
Offset: 0x2318, 0x2344, 0x2370, 0x239C, 0x23C8, 0x23F4

Legend: C = Clearable bit, HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
							DMAFLT[3]	DMAFLT[2]
Access							R/C/HS	R/C/HS
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	DMAFLT[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
Access	R/C/HS	R/C/HS	R/C/HS	R/C/HS	R/C/HS		R/C/HS	R
Reset	0	0	0	0	0		0	0

Bit 9 – DMAFLT[3] DMA Fault Status bit 3

Value	Description
1	A bus write error has occurred; a write transaction could not be completed by the DMA
0	No bus write error has occurred

Bit 8 – DMAFLT[2] DMA Fault Status bit 2

Value	Description
1	A bus read error has occurred; invalid data were received by the DMA
0	No bus read error has occurred

Bits 7:6 – DMAFLT[1:0] DMA Fault Status bit 1 and bit 0

Value	Description
11	Address Fault condition(s)
10	Access attempted to address higher than HADDR[23:0], not detected until actual access
01	Access attempted to address lower than LADDR[23:0], but above the SFR range, not detected
00	No DMA Fault condition

Bit 5 – DONE DMA Complete Operation Interrupt Flag bit

Value	Description
1	The DMA channel's CNT[31:0] counter has reached 0
0	The DMA channel's CNT[31:0] counter has not reached 0

Bit 4 – HALF DMA 50% Watermark Level Interrupt Flag bit

Value	Description
1	The DMA channel's CNT[31:0] counter has reached the halfway point towards 0

Value	Description
0	The DMA channel's CNT[31:0] counter has not reached the halfway point towards 0

Bit 3 – OVERRUN DMA Channel Overrun Flag bit

Value	Description
1	The DMA channel is triggered while it is still completing the operation based on the previous trigger
0	The overrun condition has not occurred

Bit 1 – MATCH Pattern Match Status bit (see 13.4.13. Pattern Match)

Value	Description
1	Pattern match has been detected
0	Pattern match has not been detected

Bit 0 – DBUFWF Buffered Data Write Flag bit

Value	Description
1	The content of the DBUF[31:0] (DMABUF[31:0]) bits has not been stored into the location specified in DADDR[23:0] or SADDR[23:0] in Null Write mode
0	The content of the DBUF[31:0] (DMABUF[31:0]) bits has been stored into the location specified in DADDR[23:0] or SADDR[23:0] in Null Write mode

13.3.8 DMA Channel x Source Address Register(1,2)

Name: DMAxSRC
Offset: 0x231C, 0x2348, 0x2374, 0x23A0, 0x23CC, 0x23F8

Notes:

1. For SFR, the DMA module operates over the entire address range.
2. For data space, the DMA module operates over the entire range of available memory.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	SADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – SADDR[23:0] Source Address bits
These bits indicate the address location from which the DMA module initiates the read operation. SADDR[23:0] are dynamically updated based on SAMODE[1:0] and RELOADS.

13.3.9 DMA Channel x Destination Address Register(1,2)

Name: DMAxDST
Offset: 0x2320, 0x234C, 0x2378, 0x23A4, 0x23D0, 0x23FC

Notes:

1. For SFR, the DMA module operates over the entire address range.
2. For data space, the DMA module operates over the entire range of available memory.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	DADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – DADDR[23:0] Destination Address bits
These bits indicate the address location to which the DMA module initiates the write operation; DADDR[23:0] are dynamically updated based on DAMODE[1:0] and RELOADD.

13.3.10 DMA Channel x Count Register

Name: DMAxCNT
Offset: 0x2324, 0x2350, 0x237C, 0x23A8, 0x23D4, 0x2400

Bit	31	30	29	28	27	26	25	24
	CNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CNT[31:0] Count bits

The CNT[31:0] bits indicate the number of bytes to be transferred, based on the specified size. During each transaction, the DMA module decrements this register value according to the number of completed transfers. CNT[31:0] are dynamically updated and automatically reloaded in Repeated One-Shot and Repeated Continuous operations. In One-Shot and Continuous operations, CNT[31:0] are reloaded based on RELOADC. Note that this register occupies all 32 bits in order to accommodate repeated transfers.

13.3.11 DMA Channel x Clear Register

Name: DMAxCLR
Offset: 0x2328, 0x2354, 0x2380, 0x23AC, 0x23D8, 0x2404

Bit	31	30	29	28	27	26	25	24
	CLR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CLR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CLR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CLR[31:0] Clear Register bits
When set high, the corresponding data bit(s) from the DMABUF register is (are) cleared (after set and before inverted, if applicable; see 13.4.12.1. Priority and 13.4.12. Bit Manipulation) before reaching the destination.

13.3.12 DMA Channel x Set Register

Name: DMAxSET
Offset: 0x232C, 0x2358, 0x2384, 0x23B0, 0x23DC, 0x2408

Bit	31	30	29	28	27	26	25	24
	SET[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SET[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SET[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SET[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – SET[31:0] Set Register bits
When set high, the corresponding data bit(s) from the DMABUF register is (are) set (before cleared, then inverted, if applicable; see 13.4.12.1. Priority and 13.4.12. Bit Manipulation) before reaching the destination.

13.3.13 DMA Channel x Invert Register

Name: DMAxINV
Offset: 0x2330, 0x235C, 0x2388, 0x23B4, 0x23E0, 0x240C

Bit	31	30	29	28	27	26	25	24
	INV[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INV[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INV[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – INV[31:0] Inverter Register bits
When set high, the corresponding data bit(s) from the DMABUF register is (are) inverted (after set, then cleared, if applicable; see 13.4.12.1. Priority and 13.4.12. Bit Manipulation) before reaching the destination.

13.3.14 DMA Channel x Mask Register

Name: DMAxMSK
Offset: 0x2334, 0x2360, 0x238C, 0x23B8, 0x23E4, 0x2410

Bit	31	30	29	28	27	26	25	24
	MSK[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSK[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSK[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSK[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSK[31:0] Mask Register bits
When set high, only the corresponding data bit(s) from the DMABUF and DMAPATn registers are compared in the pattern match operation.

13.3.15 DMA Channel x Pattern Register

Name: DMAxPAT
Offset: 0x2338, 0x2364, 0x2390, 0x23BC, 0x23E8, 0x2414

Bit	31	30	29	28	27	26	25	24
	PAT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PAT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PAT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PAT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PAT[31:0] Pattern Register bits
These bits contain a user provided pattern for the pattern match operation.

13.4 Operation

13.4.1 Data Transfer Options

The DMA Controller transfers data from a source address (DMAxSRC) to a destination address (DMAxDST) upon the receipt of a hardware or software trigger. The transfer occurs as a two-step process: a read from the source address and transfer to the DMA buffer, DMABUF, followed immediately by a write from DMABUF to the destination address. The controller then determines if the operation on this channel has been completed. The active DMA channel may assert an interrupt to indicate the end of a transfer and/or the status of a transfer in progress.

Each channel of the DMA controller can be independently programmed to move data between the data RAM and peripherals (i.e., the SFR area), between peripherals or between areas of data RAM. Transactions can be single occurrences, repeated single occurrences, or continuous, based on an event trigger and/or the channel transaction counter. The source and destination address registers can be independently programmed to increment, decrement or remain unchanged during transactions.

13.4.2 Data Size

The DMA Controller can handle 8-bit (one byte), 16-bit and 32-bit transactions. Each DMA channel is individually configurable for the data size to be used with the SIZE[1:0] bits (DMAxCH[7:6]). These bits allow the user to specify whether one byte, one 16-bit word or one 32-bit word is transferred per one load or store transaction.

In this Byte mode, where SIZE[1:0] is '00', the counter (CNT[31:0]) represents the number of bytes remaining to be transferred. Similarly, the CNT[31:0] bits represent the number of 16-bit words and

32-bit words remaining to be transferred in 16-bit Word (SIZE[1:0] = 01) and 32-bit Word (SIZE[1:0]) modes, respectively.

In Byte mode, byte transfers are accommodated through bit 0 of the address. When bit 0 is '0', the lower byte is addressed, while the upper byte is addressed when bit 0 is '1'. For 16-bit word operation, the Address Pointers are 16-bit word-aligned. That is, bit 0 is always '0'. For 32-bit word operation, the Address Pointers are 32-bit word-aligned and bits[1:0] are always '00'.

By default (SIZE[1:0] = 00), the channel is configured for byte-size transactions.

13.4.3 DMA Trigger Sources

Each DMA channel can select from the hardware triggers to initiate a DMA transfer. The trigger sources are generally device-level interrupts from peripheral modules, as well as the external interrupts and interrupt-on-change sources. The CHSEL[7:0] bits (DMAxSEL[7:0]) select which interrupt (and thus module) is used as a trigger for a particular DMA channel (see Table 13-2). The CHSEL bits may change at any time to select another module to service. However, it is not recommended to make a change while the associated DMA channel is in operation. A DMA channel can be configured to service any memory-mapped peripheral, regardless of the trigger's origin. This is because the trigger (interrupt) source is independent of the DMA source and destination addresses. For example, a DMA channel configured to respond to the INT0 interrupt could be used to move data into or out of a UART FIFO. In most cases, it makes more sense to use a peripheral's own interrupt for performing a data transfer. However, there are also many cases where it is desirable to use one peripheral interrupt to perform a DMA operation on another peripheral – perhaps even another DMA channel. Examples of such operations are provided in 13.5. Application Examples. In addition, a DMA channel may be triggered in software by setting the CHREQ bit (DMAxCH[4]). This allows for an application to use the DMA Controller to move data directly, without having to wait for a hardware interrupt. CHREQ is also set when a hardware trigger occurs.

13.4.4 Channel Priority and Priority Schemes

While DMA channels can function independently to service different peripherals at the same time, they are still limited by the presence of a single DMA data bus and a single data channel to data space. When two or more channels request the DMA controller to handle a data transfer at the same time, the controller arbitrates the requests and decides which channel receives priority and bus access. The controller uses two defined arbitration schemes to assign channel priority: Fixed and Round Robin. The PRIORITY bit (DMACON[0]) determines the scheme to be used. In the Round Robin scheme (DMACON[0] = 1), the controller assigns priority and bus grant to the lowest numbered channel for the first time when there is channel contention. Grant determination is evaluated after every iteration of data transfers. For each successive transfer conflict, the next higher channel receives preference, continuing as a cycle through all the channels. If the channel that has priority does not make a request at that time, it is skipped for the next channel in the cycle. As an example, if Channels 0, 1 and 2 all simultaneously request a data transfer, Channel 0 is serviced; Channels 1 and 2 are then serviced in that order. During the next service request, any request from Channel 1 will receive preference; Channel 2 will receive preference in the following round. Any subsequent transfer requests from Channel 0 will be ignored until all the other channels have received priority once. Typical examples of Round Robin arbitration are shown in Table 13-3.

Table 13-3. Examples of Channel Access Using Round Robin Priority Scheme

Requesting DMA Channel(s)				Channel Granted Priority
0	1	2	3	
				None
	X			CH1
X	X	X		CH2
X	X			CH0
X	X			CH1

.....continued

Requesting DMA Channel(s)				Channel Granted Priority
0	1	2	3	
X	X		X	CH3
X	X			CH0

In contrast, the Fixed scheme (DMACON[0] = 0) always gives priority to the lowest requesting channel number. Using the previous example, if there are several sequential transfer requests involving Channel 0, Channel 0 will always receive preference over other channels. The Fixed priority scheme is the default. Typical examples are shown in Table 13-4.

Table 13-4. Examples of Channel Access Using Fixed Priority Scheme

Requesting DMA Channel(s)				Channel Granted Priority
0	1	2	3	
				None
	X			CH1
X	X	X		CH0
X	X			CH0
	X			CH1
	X		X	CH1
			X	CH3

13.4.5 Memory Boundary

While the 24-bit DMAxSRC and DMAxDST registers allow access to the entire data space, there may be circumstances where it is desirable to limit DMA operations to a much narrower range. This may be required for many reasons; for example, to protect program variables or a software stack.

The DMAHIGH and DMALOW registers allow the user to set the upper and lower address limits for DMA operations in the data space. All DMA channels are restricted to the address range set by DMAHIGH and DMALOW. When an active DMA channel initiates a memory transaction outside of the boundary defined by this register pair, an interrupt will be invoked on a channel basis. Fault status bits, DMAFLT[1:0] = 10, indicate DMA operations that attempt to access above DMAHIGH, and DMAFLT[1:0] = 01 indicates operations that attempt to access below DMALOW. Boundaries are applicable only to the SRAM region. The memory-mapped SFR range is always accessible by DMA.

13.4.6 Buffer Data Write Bit

The DBUFWF bit (DMAxSTAT[0]) indicates whether buffered data in DMABUF have been stored into the specified destination location. It serves as a protection against data loss due to unexpected termination of the active DMA operation. For example, if the user decides to stop the DMA operation that happens to be between load and store, the buffered data in DMABUF will not reach their destination. The user can examine this bit to see if the buffered data still need to be stored at the specified destination location.

Table 13-5 summarizes the behavior of DBUFWF in various modes of operation.

Table 13-5. Interpretation of the DBUFWF Bit Status

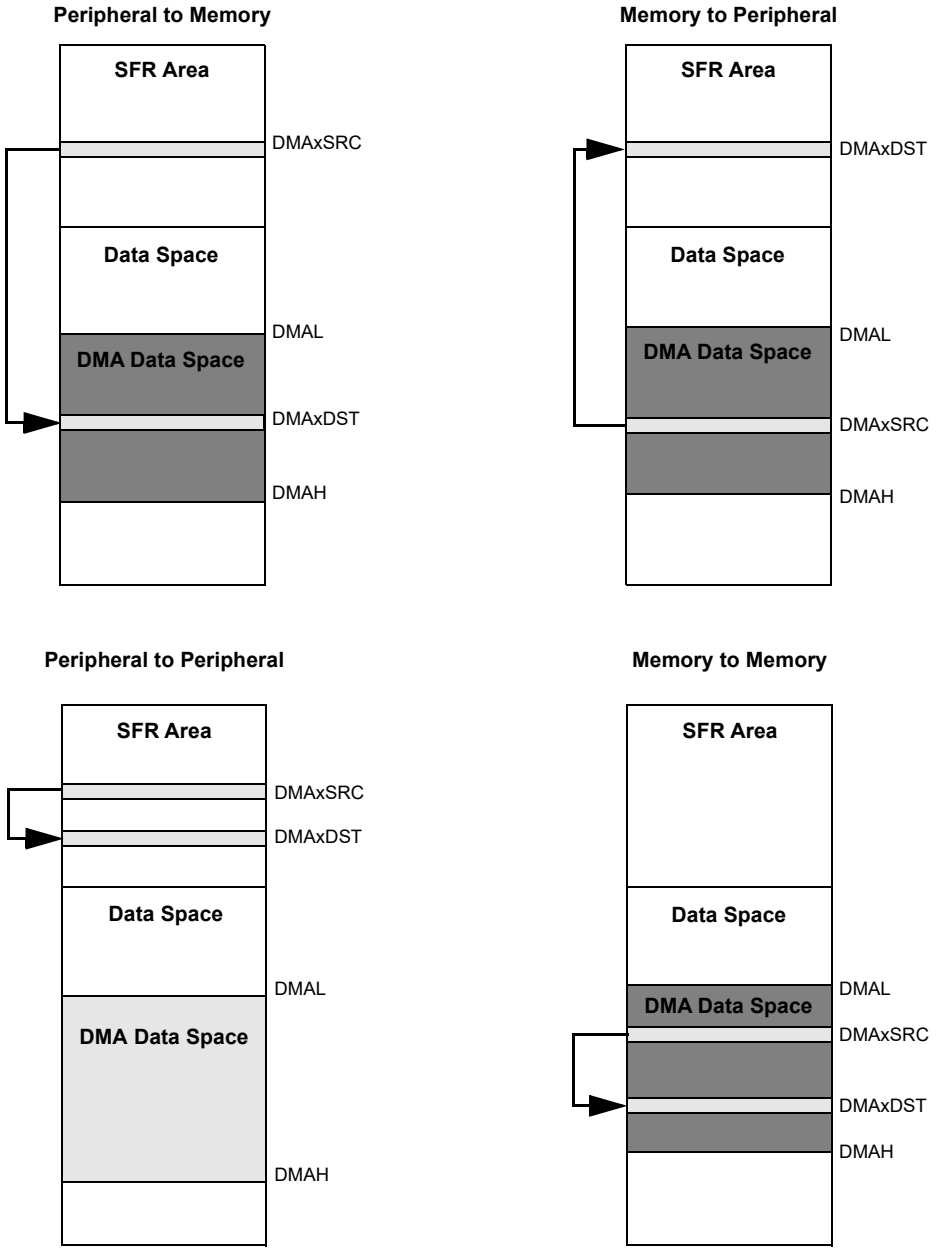
DBUFWF Status	Operation Status		
	Repeated One-Shot	Repeated Continuous	Null Write
1	After Loading DMABUF		
0	After writing to [DMAxDST]		After Writing to [DMAxSRC]

13.4.7 Types of Data Transfers

All DMA transactions occur solely within the data space address space. In the least restricted case, all Data Space addresses are available to the DMA Controller; this includes the entire SFR space and (by extension) all peripherals. As defined by the source and destination, there are four types of DMA data transfers (Figure 13-3):

- Peripheral to Memory (Receive)
- Memory to Peripheral (Transmit)
- Memory to Memory
- Peripheral to Peripheral

Figure 13-3. Types of DMA Data Transfers



Note: Relative sizes of memory areas are not shown to scale.

13.4.7.1 Peripheral to Memory (Receive)

If a source address register is programmed with an SFR address while the destination register contains a data space address, the controller will read from the peripheral module being serviced and write the retrieved data content to the specified location in data space. This is most suitable for peripherals configured to receive data, such as a UART or SPI module.

13.4.7.2 Memory to Peripheral (Transmit)

If the source address register is programmed with a data space address and the destination address register contains a peripheral (SFR) address, the controller is forced to read from the data space and write to the SFR when triggered. This makes this type of data flow most suitable to support the peripheral modules configured to transmit data, such as a serial communication module.

13.4.7.3 Memory to Memory

If data relocation within data space is required, the source and destination address registers for any channel can simply be programmed with the desired data space memory locations. Obviously, this type of data transfer does not require access to any peripheral; however, the trigger from any of the peripherals can be used to initiate the transfer.

13.4.8 Data Transfers Modes

Data transfers are also defined by how the transaction is structured: the number of data transfers that can occur per trigger event, how events are counted and if the event repeats. The DMA Controller defines four transfer modes, which encompass all these features:

- One-Shot
- Repeated One-Shot
- Continuous
- Repeated Continuous

The transfer mode is defined by the TRMODE[1:0] bits (DMAxCH[11:10]). In addition, the RELOADS, RELOADD and RELOADC bits (DMAxCH[26:24]) can modify the behavior of some modes.

13.4.8.1 Common Transfer Mode Sequence

Regardless of the transfer mode, all DMA transfers follow the same basic sequence:

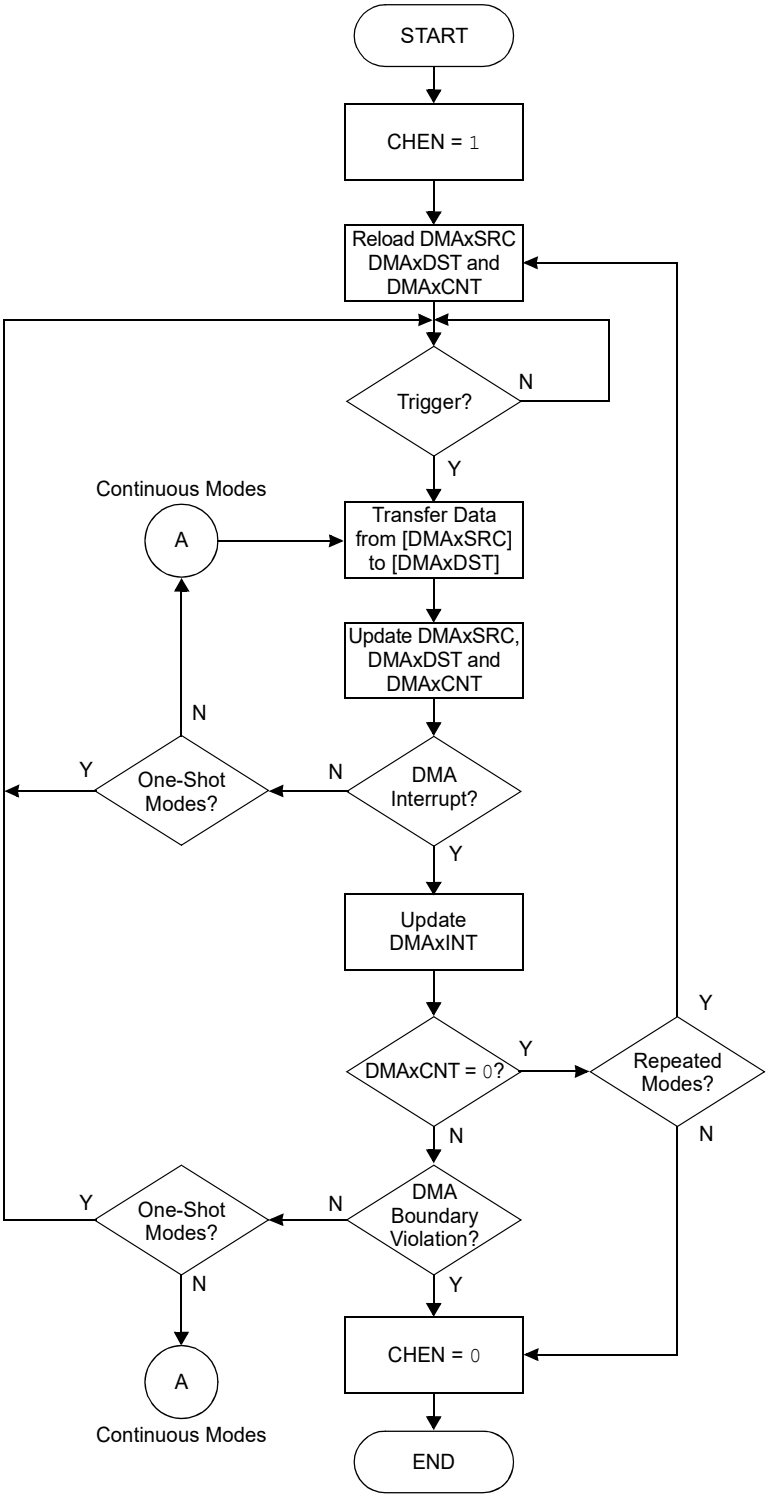
1. Upon the receipt of a DMA trigger or the setting of the CHREQ bit (DMAxCH[4]), data are loaded into DMABUF from the location addressed by DMAxSRC, then stored in the location addressed by DMAxDST.
2. Following the transaction, DMAxSRC and DMAxDST are updated appropriately; one or both may be incremented or decremented, depending on the channel's configuration (see 13.4.9. Addressing Modes for additional information). At the same time, DMAxCNT is decremented by one.
3. The module tests for any DMA interrupt conditions. If an interrupt condition has occurred, the DMAxSTAT register flags are updated accordingly:
   - If a DMA interrupt has occurred, all modes continue to step 4.
   - If an interrupt has not occurred, all modes return to step 1. In One-Shot mode, the controller waits for the next trigger. In Continuous mode, the controller repeats the cycle continuously until DMAxCNT value becomes 0 or a DMA interrupt occurs.
4. If DMAxCNT has decremented to zero:
   - The values of DMAxSRC, DMAxDST and DMAxCNT are reloaded and the sequence repeats from step 1 (all Repeated modes).
   - The CHEN bit (DMAxCH[0]) is cleared and the channel is disabled (One-Shot and Continuous modes).
5. If DMAxCNT has not decremented to zero, the controller checks for a memory address boundary violation of DMALOW or DMAHIGH:
   - If one of the boundaries has been crossed, the CHEN bit is cleared, and the channel is disabled.
   - If there is no boundary violation, the controller returns to step 1. For both One-Shot modes, the controller waits for the next trigger. For both Continuous modes, the controller proceeds to performing the next data transfer.

The four data transfer modes differ in the number of data transfers that can take place with a single trigger and how the DMAxCNT register behaves. The common logic flow for all data transfer modes is illustrated in the flowchart in Figure 13-4. The differences between the modes are summarized in Table 13-6.

Table 13-6. Comparison of DMA Data Transfer Modes

Transfer Mode	Transfers per Trigger	DMAxCNT Behavior	
		Decrements on	at 0000h
One-Shot	Single	Trigger	Disable Channel
Repeated One-Shot		Transfer	Reload and Repeat
Continuous	Multiple	Trigger	Disable Channel
Repeated Continuous		Transfer	Reload and Repeat

Figure 13-4. Common Logic Flow for Data Transfer Modes



Legend:
One-Shot Modes: One-Shot and Repeated One-Shot
Continuous Modes: Continuous and Repeated Continuous
Repeated Modes: Repeated One-Shot and Repeated Continuous

13.4.8.2 One-Shot Mode

In One-Shot mode (TRMODE[1:0] = 00), a single transfer (from DMAxSRC to DMAxDST) is performed for each trigger event. By default, the Reset value of DMAxCNT is 0001h. When a single One-Shot transfer occurs, DMAxCNT is decremented to 0000h; this disables the channel and requires the channel to be re-enabled to perform the next transaction. Of course, it is also possible to store a larger value in DMAxCNT and then conduct a defined number of One-Shot transfers.

Example 13-1 shows a typical code sequence for a One-Shot transfer.

Example 13-1. Typical Code Sequence for a One-Shot Transfer

```
unsigned int SRC[4];
unsigned int DEST[4];
int main()
{
    for (int i=0;i<4;i++)
    {
        SRC[i]=i+1;                     //fill with i+1
        DEST[i]=0;                      //fill with 0
    }
    DMACONbits.ON=1;
    DMACONbits.PRIORITY=1;
    IFS2bits.DMA0IF=0;
    DMALOW=0x4000;
    DMAHIGH=0x7FFF;                      //set lower and upper address limit
    DMA0SRC=(unsigned int)& SRC;              //load the source address
    DMA0DST=(unsigned int)& DEST;             //load destination address
    DMA0CNT=4; //Four Transfer to be done
    DMA0CH=0;
    DMA0CHbits.SAMODE=1;                    //Source address increment mode
    DMA0CHbits.DAMODE=1;                    //Destination address increment mode
    DMA0CHbits.TRMODE=0;                    //One-Shot Transfer mode
    DMA0CHbits.SIZE = 2;                    //32-bit transfer per count
    DMA0CHbits.CHEN=1;                      //Enable channel
    DMA0CHbits.CHREQ=1;                     //First trigger
    while(DMA0CHbits.CHREQ);
    DMA0CHbits.CHREQ=1;                     //Second trigger
    while(DMA0CHbits.CHREQ);
    DMA0CHbits.CHREQ =1;                    //Third trigger
    while(DMA0CHbits.CHREQ);                  //HALF=1 since DMACNT is
                                  //at halfway point
    DMA0CHbits.CHREQ=1;                      //Fourth trigger DMACNT=0
                                  //and transfer complete
    while(DMA0CHbits.CHREQ);
    while(!DMA0STATbits.DONE);              //Transfer Complete
                                 //DMA0IF=1 ,DONE=1, CHEN=0;
    IFS2bits.DMA0IF=0;
    DMA0STATbits.DONE=0;
    DMA0STATbits.HALF=0;
    while(1);
}
```

13.4.8.3 Repeated One-Shot Mode

In Repeated One-Shot mode (TRMODE[1:0] = 01), single transfers occur repeatedly as long as triggers are being provided or CHREQ is set. Each time a trigger occurs or CHREQ is set, DMAxCNT is decremented. In this case, however, the channel is not disabled when DMAxCNT reaches 0000h. Instead, the original value of DMAxCNT is reloaded; if RELOADS = 1 and RELOADD = 1, the original values of DMAxSRC and DMAxDST are also reloaded. The entire cycle then starts again on the next trigger. To end the sequence, the channel must be disabled by clearing the CHEN bit in software.

Example 13-2 shows a typical code sequence for a Repeated One-Shot transfer.

Example 13-2. Typical Code Sequence for a Repeated One-Shot Transfer

```
unsigned int SRC[4];
unsigned int DEST[4];
int main()
{
    for (int i=0;i<4;i++)
    {
        SRC[i]=i+1;                          //fill with i+1
        DEST[i]=0;                           //fill with 0
    }
    DMACONbits.ON=1;
    DMACONbits.PRIORITY=1;
    IFS2bits.DMA0IF=0;
    DMALOW=0x4000;
    DMAHIGH=0x7FFF;                          //set lower and upper address limit
    DMA0SRC=(unsigned int)& SRC;             //load the source address
    DMA0DST=(unsigned int)& DEST;            //load destination address
    DMA0CNT=4;
    DMA0CH=0;
    DMA0CHbits.SAMODE=1;                     //Source address increment mode
    DMA0CHbits.DAMODE=1;                     //Destination address increment
mode
    DMA0CHbits.TRMODE=1;                     //Transfer mode Repeat One-Shot
    DMA0CHbits.RELOADS=1;                    //Reload Source Address
    DMA0CHbits.RELOADD=1;                    //Reload Destination Address
    DMA0CHbits.SIZE=2;                       //32-Bit transfer per count
    DMA0CHbits.CHEN=1;                       //Channel enable
    while(1)
    {
        DMA0CHbits.CHREQ=1;                 //First trigger
        while(DMA0CHbits.CHREQ);
        DMA0CHbits.CHREQ=1;                 //Second trigger
        while(DMA0CHbits.CHREQ);
        DMA0CHbits.CHREQ=1;                 //Third trigger
        while(DMA0CHbits.CHREQ);            //HALF=1 since DMACNT is in half
way point
        DMA0CHbits.CHREQ=1;                //Fourth trigger DMACNT=0 and
transfer complete
        while(DMA0CHbits.CHREQ);            //DMACNT reloaded to 4, DMA0IF=1
                                            //Since RELOADS=1 and RELOADD=1,
                                            //DMASRC0/DMADST0 are reloaded

        while(!DMA0STATbits.DONE);
        DMA0STATbits.DONE=0;                //Clear DONE and HALIF flag
        DMA0STATbits.HALF=0;
        IFS2bits.DMA0IF=0;
    }
}
```

13.4.8.4 Continuous Mode

In Continuous mode (TRMODE[1:0] = 10), a single trigger starts a sequence of back-to-back transfers; these continue with each transfer decrementing DMAxCNT until it reaches 0000h. At this point, like One-Shot mode, the channel is disabled.

One-Shot and Continuous modes are similar in that each mode performs a certain number of transfers for one time. The difference is that One-Shot mode requires a trigger for each transfer, while Continuous mode allows many transfers for each trigger. In addition, DMAxCNT is controlled by the number of individual transactions, not the number of triggers.

Example 13-3 shows a typical code sequence for a Continuous transfer.

Example 13-3. Typical Code Sequence for a Continuous Transfer

```
unsigned int SRC[100];
unsigned int DEST[100];
int main()
{
    for (int i=0;i<100;i++)
    {
        SRC[i]=i+1;                          //fill with i+1
        DEST[i]=0;                           //fill with 0
    }
    DMACONbits.ON=1;
    DMACONbits.PRIORITY=1;
    IFS2bits.DMA0IF=0;
    DMAHIGH=0x5000;                          //set lower and upper address limit
    DMALOW=0x850;
    DMA0SRC=(unsigned int)& SRC;            // load the source address
    DMA0DST=(unsigned int)& DEST;           // load destination address
    DMA0CNT=0x64; //CNT = 100
    DMA0CHbits.SAMODE=1;                     //Source address increment mode
    DMA0CHbits.DAMODE=1;                     //Destination address increment mode
    DMA0CHbits.TRMODE=2;                     //Transfer mode Continuous
    DMA0CHbits.SIZE = 2;                     //32-bit transfer per count
    DMA0CHbits.CHEN=1;                      //Channel enable
    DMA0CHbits.CHREQ=1;                     //Enable the transfer by software
trigger
    while(!DMA0STATbits.DONE);              //DONE=1;CHEN=0,DMA0IF=1
    //100 (DMAXCNT=100)transfer complete with one trigger
    IFS2bits.DMA0IF=0;
    while(1);
}
```

13.4.8.5 Repeated Continuous Mode

Repeated Continuous mode (TRMODE[1:0] = 11) can be thought of as a combination of Continuous and Repeated One-Shot modes; data transfers keep occurring as long as triggers are provided, and multiple transfers can occur with each trigger. Like Continuous mode, each transfer decrements DMAxCNT. When it reaches 0000h, the address and count registers are reloaded, and the process is repeated.

Like Repeated One-Shot mode, ending the sequence requires disabling the channel, either by disabling the trigger source or clearing the CHEN bit in software.

Example 13-4 shows a typical code sequence for a Repeated Continuous mode transfer.

Example 13-4. Typical Code Sequence for a Repeated Continuous Transfer (Memory to Memory, RELOADS = 1, RELOADD = 1)

```
unsigned int Array1[100];
unsigned int Array2[100];
int i;
int main()
{
   ANSA=0;
   TRISA=0;
   LATA=0x70;
   for (i=0;i<100;i++)
   {
      Array1[i]=i+1;                  //fill with i+1
      Array2[i]=0;                    //fill with 0
   }
   DMACONbits.ON=1;
   DMACONbits.PRIORITY=1;
   DMALOW=0x4000;
   DMAHIGH=0x7FFF;                    //set lower and upper address limit
   DMA0SRC=(unsigned int)& Array1;    // load the source address
   DMA0DST=(unsigned int)& Array2;    // load destination address
   DMA0CNT=100;                       // 100 Transaction per trigger
   DMA0CH=0;
   DMA0CHbits.SAMODE=1;               //Source address increment mode
   DMA0CHbits.DAMODE=1;               //Destination address increment mode
   DMA0CHbits.TRMODE=3;               //Transfer mode Repeat continous
   DMA0CHbits.RELOADS=1;              //Reload Source Address
   DMA0CHbits.RELOADD=1;              //Reload Destination Address
   DMA0CHbits.CHEN=1;                 //Channel enable
   IFS3bits.DMA0IF=0;
   while(1)
   {
      DMA0CHbits.CHREQ=1;             //TRIGGER
      while(!IFS3bits.DMA0IF);
      for (i=0;i<100;i++)             //Clear the Destination Memory
      {
         Array2[i]=0;                 //fill with 0
      }
      IFS3bits.DMA0IF=0;
      PORTAbits.RA0=0;
      DMA0STATbits.DONE=0;
      DMA0STATbits.HALF=0;
   }
}
```

13.4.8.6 Address and Count Reload

Although the Repeated modes explicitly include it, all the transfer modes allow the automatic re-use of the initial source and destination addresses and transaction counts for multiple operations. Setting the RELOADS (DMAxCH[24]), RELOADD (DMAxCH[25]) and RELOADC (DMAxCH[26]) bits allows the values of DMAxSRC, DMAxDST and DMAxCNT to be restored for the next DMA operation. This causes the registers to be reloaded in One-Shot and Continuous modes after a transfer operation is complete and the channel is re-enabled. Address and transaction count reloading is automatic in Repeated One-Shot and Repeated Continuous modes. DMAxCNT also has its value

reloaded after it has been decremented to 0000h, regardless of the setting of the RELOADC or TRMODEx bits. The only exception is if the channel is stopped in mid-operation and restarted later.

Table 13-7 shows the effect of RELOADS/RELOADD/RELOADC on DMAxSRC, DMAxDST and DMAxCNT for the data transfer modes.

Table 13-7. RELOADS/RELOADD/RELOADC Bits and Data Transfer Modes

RELOADS/RELOADD/ RELOADC Bits	Transfer Mode	DMAxSRC	DMAxDST	DMAxCNT
1	—	Reloaded	Reloaded	Reloaded
0	Repeated One-Shot/ Continuous	Not Reloaded	Not Reloaded	Reloaded(1)
0	One-Shot/Continuous	Not Reloaded	Not Reloaded	Not Reloaded

Note:
1. The reload only happens after DMAxCNT has decremented to '0'. No reload occurs if the channel is stopped and later resumed.

13.4.9 Addressing Modes

Following each transfer, the DMAxSRC and DMAxDST registers may be automatically updated by the channel. This potentially allows the channel to move data between multiple locations without the need for user intervention. Automatic address updating is controlled by the SAMODEx and DAMODEx bits (DMAxCH[15:14] and [13:12]).

The combination of the different address update options (fixed, increment or decrement) provides four supported addressing modes:

- Fixed to Fixed
- Fixed to Block
- Block to Fixed
- Block to Block

Table 13-8 shows the address modes and the various SAMODEx and DAMODEx combinations.

Table 13-8. Configurations for DMA Addressing Modes

Mode(1)	SAMODE[1:0]	DAMODE[1:0]
Fixed to Fixed	00	00
Fixed to Block (Address Increment)	00	01
Fixed to Block (Address Decrement)	00	10
Block to Fixed (Address Increment)	01	00
Block to Fixed (Address Decrement)	10	00
Block to Block (Address Increment)	01	01
Block to Block (Address Decrement)	10	10

Note:
1. The increment and decrement of the address will be based on the SIZEx bits value.

13.4.9.1 Fixed to Fixed

Fixed to Fixed mode is set by configuring SAMODE[1:0] and DAMODE[1:0] to '00'. In this mode, the source and destination addresses remain the same after each transaction. This mode is suited for One-Shot transfers of a single byte, or word of data, between two fixed addresses.

13.4.9.2 Fixed to Block

In Fixed to Block mode, the source address remains unchanged throughout the transfer, but the destination address is incremented or decremented (depending on the DAMODEx setting). This

works well for receiving data from the single-word buffer of a serial communication peripheral and filling a block of addresses designated as a buffer.

Example 13-5 shows sample code for Fixed to Block Addressing. The source address (UART receive) generates an interrupt when the UART buffer has received four bytes; the UART Receive Interrupt Flag (U2RIF) will trigger the transfer.

Example 13-5. Code for Fixed to Block Continuous Transfer (Peripheral to Memory)

```
void UartInit(void);
unsigned char Array[1000];
int i;

int main()
{
    ANSELA = 0;
    UartInit();

    for (i = 0; i < 100; i++)
    {
        Array[i] = 0;                       //fill array with 0
    }

    DMA0CH = 0;
    IFS2bits.DMA0IF = 0;

    DMACONbits.ON = 1;                    //Enables DMA module
    DMAHIGH = 0x5000;                     //sets lower and upper address limit
    DMALOW = 0x4000;
    DMA0SRC = (unsigned int) &U2RXREG;  //load source address
    DMA0DST = (unsigned int) & Array;  // load destination address
    DMA0CNT = 4;                          //4 bytes are transferred per trigger

    DMA0CHbits.SIZE = 0;                //Per count 1 byte is transferred
    DMA0CHbits.SAMODE = 0;              //Source address increment mode,
    DMA0CHbits.DAMODE = 1;              //Destination address increment mode,
    //increment 1
    DMA0CHbits.TRMODE = 3;              //Transfer mode Continuous
    DMA0SELbits.CHSEL = 17;             //Trigger on UART2 Receive
    DMA0CHbits.CHEN = 1;                //Channel enable

    while (!DMA0STATbits.DONE);         //DONE=1,DMA0IF=1 and
                                        //transfer complete with one trigger
    IFS2bits.DMA0IF = 0;                //Disable interrupt flag

    while(1);
}
void UartInit(void)
{
    RPINR9bits.U2RXR = 10;             //U2RX
    U2BRG = 100;                       //BAUDRATEREG2;
    U2CON = 0;
    U2STAT = 0;
    U2CONbits.BRGS = 1;                //High Baud Rate Select
    U2CONbits.MODE = 0;                //Asynchronous 8-bit UART
    U2STATbits.RXWM = 3;               //Interrupt after 4 transfers
    U2CONbits.UARTEN = 1;
    U2CONbits.URXEN = 1;
    U2CONbits.TXEN = 1;
    IFS2bits.U2RXIF = 0;
}
```

13.4.9.3 Block to Fixed

In Block to Fixed mode, the source address is incremented or decremented throughout the transfer (depending on the SAMODEx setting) while the destination address remains unchanged. This is well-suited for moving a packet of data to be transmitted into the 'single-word' transmit buffer of a serial communication peripheral.

13.4.9.4 Block to Block

In Block to Block mode, both the source and destination addresses increment or decrement (depending on the SAMODEx and DAMODEx bits setting) throughout the transfer. This mode is useful for copying a block of data from one part of the data RAM to another.

13.4.10 Flow Control

The Flow Control bits, FLWCONx (DMAxCH[9:8]), can be used to configure three different types of data transfer:

- Source to Destination (default)
- Null Write mode
- Read-Only Mode

13.4.10.1 Source to Destination Transfer

By default (FLWCON[1:0] = 00), a DMA transfer occurs from the source address to the destination address. The behavior of the source and destination addresses will depend on the configuration of SAMODE[1:0] and DAMODE[1:0] (as described in 13.4.9.1. Fixed to Fixed, 13.4.9.2. Fixed to Block, 13.4.9.3. Block to Fixed and 13.4.9.4. Block to Block).

13.4.10.2 Null Write Mode

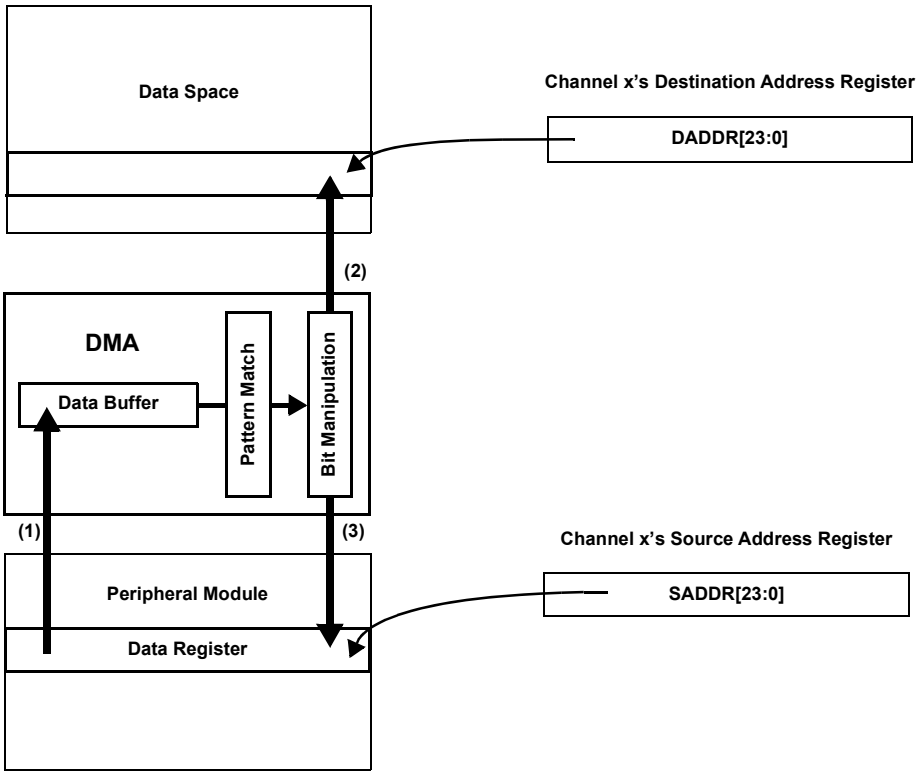
Some communication protocols require symmetrical buffer accesses; that is, for every read operation performed on a buffer, there must be an accompanying write operation. An example of this requirement occurs with the SPI module operating in Master mode.

The Null Write mode is designed to satisfy this requirement. This mode works by transferring data from the address in DMAxSRC to the address in DMAxDST, like any other DMA operation. Once this is done, however, the transferred data that are still stored in DMABUF are written back to the address specified by DMAxSRC. The write-back occurs before the DMA proceeds to its next transfer. A typical example of this is shown in Figure 13-5. Null Write mode is enabled by setting the flow control bits to '01' (FLWCON[1:0] = 01).

Note: In Null Write mode, the DBUFWF bit is clear only upon the completion of the write to the location specified in SADDR[23:0].

Figure 13-5 illustrates the basic DMA data flow in Null Write mode. In Null Write mode, both load and store transactions are initiated to the address location specified in SADDR[23:0].

Figure 13-5. Null Write Mode



Notes:

1. Read from the address location specified in SADDR[23:0] and set the DBUFWF bit.
2. Write to the address location specified in DADDR[23:0].
3. Write to the address location specified in SADDR[23:0] and clear the DBUFWF bit.
4. Repeat until completion.

13.4.10.3 Read-Only Mode

Read-Only mode for ECC support.

13.4.11 Ping-Pong

With its capability to seamlessly transfer data flow between its internal DMA channels, the 32-bit DMA Controller supports ping-pong operation to assist the CPU in effective and uninterrupted data transfer. By alternating the incoming data flow between two active DMA channels, the CPU can process the available data from one DMA channel while the other active DMA channel is making its own data available to be processed when the CPU has completed its current task.

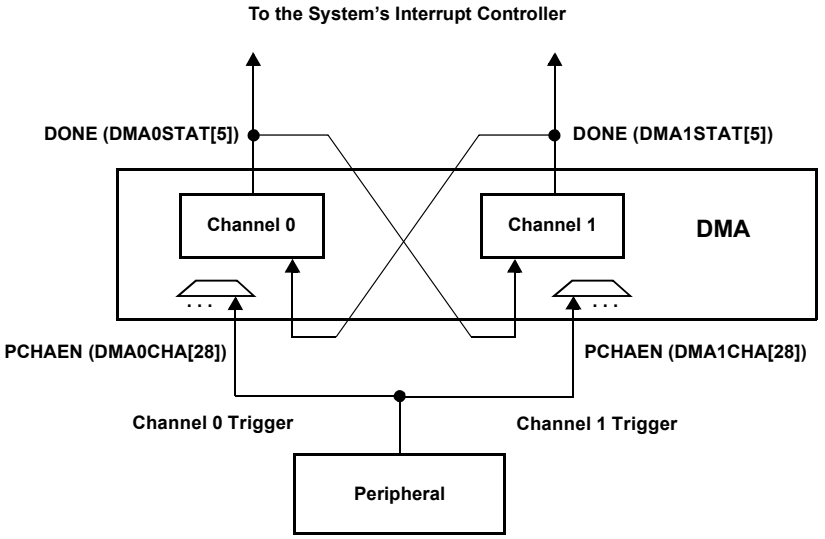
In DMA controllers with ping-pong mode, channels are paired in fixed combinations such as (0,1), (2,3) and (4,5). Either channel in each pair serves as the initiator channel enabling continuous data transfer. The initiator channel is determined by setting the PCHAEN to 1. Each channel alternates between transferring data and preparing for the next transfer, allowing for an uninterrupted data stream with minimal CPU intervention.

The capability relies on proper hardware/software interactions between the CPU and the two DMA Channels 1 and 2, see Figure 13-6. This is configured by setting CHEN (DMAxCH[0]) = 1 and PPEN (DMAxCH[29]) = 1 for both channels. Initially, before the ping-pong operation begins, PCHAEN should be set by user software explicitly for the channel that initiates the ping-pong operation.

Each channel of the pair requires its own individual trigger to facilitate transfer. Each channel must be individually set up with its own control settings, source and destination addresses, transfer count and trigger. The channels operate independently in terms of setup, but they are linked in operation through the ping-pong mechanism.

When one DMA channel completes its operation, it triggers the other active channel's hardware enable input, as shown here. This action sets the other channel's PCHAEN bit (DMAxCH[28]) high. Also, when the CPU finishes its task corresponding to the current DMA channel buffer, it sets the other channel's CHEN bit (DMAxCH[0]) high. Whenever both PCHAEN and CHEN are high (and PPEN = 1), the DMA channel is enabled and ready for data transfer when triggered. This event effectively transfers the data flow from one channel to the other.

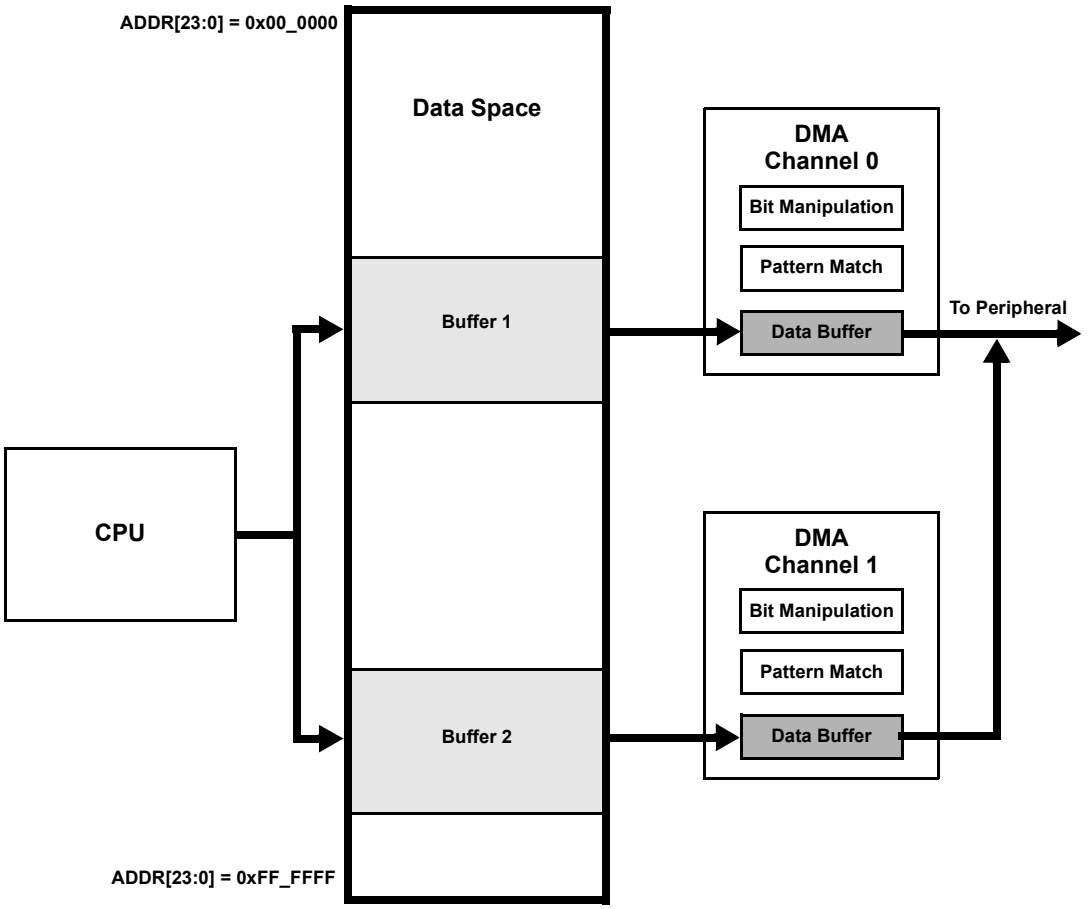
Figure 13-6. Ping-Pong Support Connection



13.4.11.1 Ping-Pong Data Transmit

When the DMA Controller is set up to transmit outgoing data to a peripheral, the setup for the ping-pong operation can be seen in Figure 13-7.

Figure 13-7. Ping-Pong – Transmit

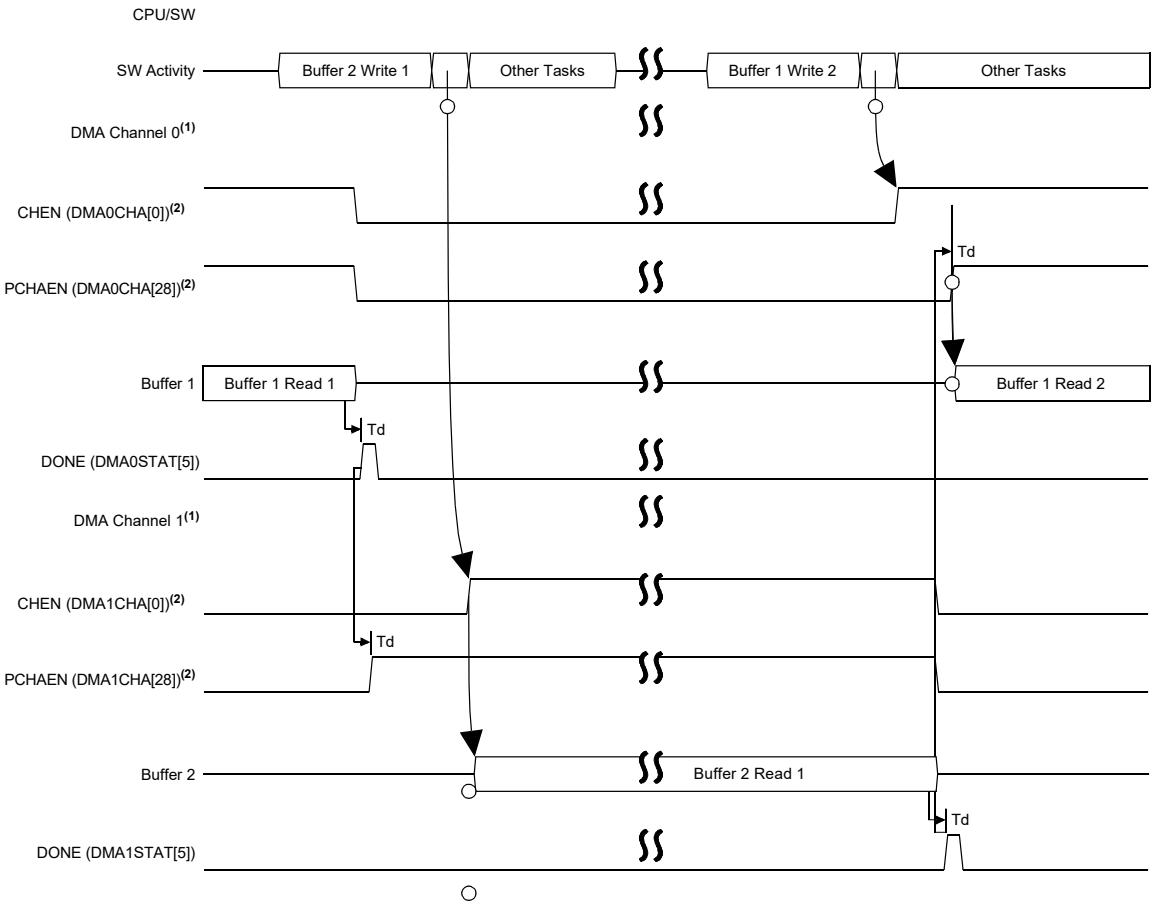


13.4.11.2 Ping-Pong Transmit Steady State

Figure 13-8 illustrates a ping-pong operation involving two connected DMA channels, as shown in Figure 13-7, alternately reading the CPU prepared data from the ping-pong buffers to a common peripheral responsible for transmitting the data. The CPU remains in full control of the operation without having to manually keep track of the DMA channel's transactions. The CPU is free to perform other tasks until completion while the DMA is operating independently. When the CPU has completed its tasks, it simply enables the appropriate DMA channel via the CHEN bit. Given that the corresponding DMA channel has also completed its transfer and subsequently triggers the other DMA channel, the data flow exchange happens autonomously. When the DMA channel has completed its transfer, it asserts its trigger output signal, which enables the other DMA channel via PCHAEN (with PPEN set high). Note that when operating in Ping-Pong mode (PPEN = 1), both the CHEN and PCHAEN bits must be set high to enable the DMA channel.

In Figure 13-8, it is initially Channel 0 that completes its transaction while the CPU is still working on the buffer for Channel 1. Therefore, the data flow exchange happens after the CPU finishes and subsequently sets Channel 1's PCHAEN bit. Later on in the diagram, one can confirm that the data flow exchange does happen correctly if the CPU finishes while Channel 1 is still performing its transaction (reverse order).

Figure 13-8. Ping-Pong Transmit in Steady State



Notes:

1. DMA Channel 0 finishes before the CPU's completion of its task, whereas DMA Channel 1 finishes after.
2. Both the CHEN (Non-Repeated modes) and PCHAEN bits are hardware cleared once the transmission is completed.

13.4.11.3 Ping-Pong Transmit Initialization

The following description serves only as a guideline for the firmware implementation.

Transmit data as an initiator:

1. User's SW sets up peripheral for transmit.
2. User's SW sets up DMA Channel 0 and Channel 1.
3. User's SW writes Buffer 1.
4. User's SW enables DMA Channel 0.
5. User's SW enables peripheral which later triggers for transmit data.
6. DMA Channel 0 reads from Buffer 1 while user's SW writes to Buffer 2.
7. See steady-state description in 13.4.11.2. Ping-Pong Transmit Steady State.

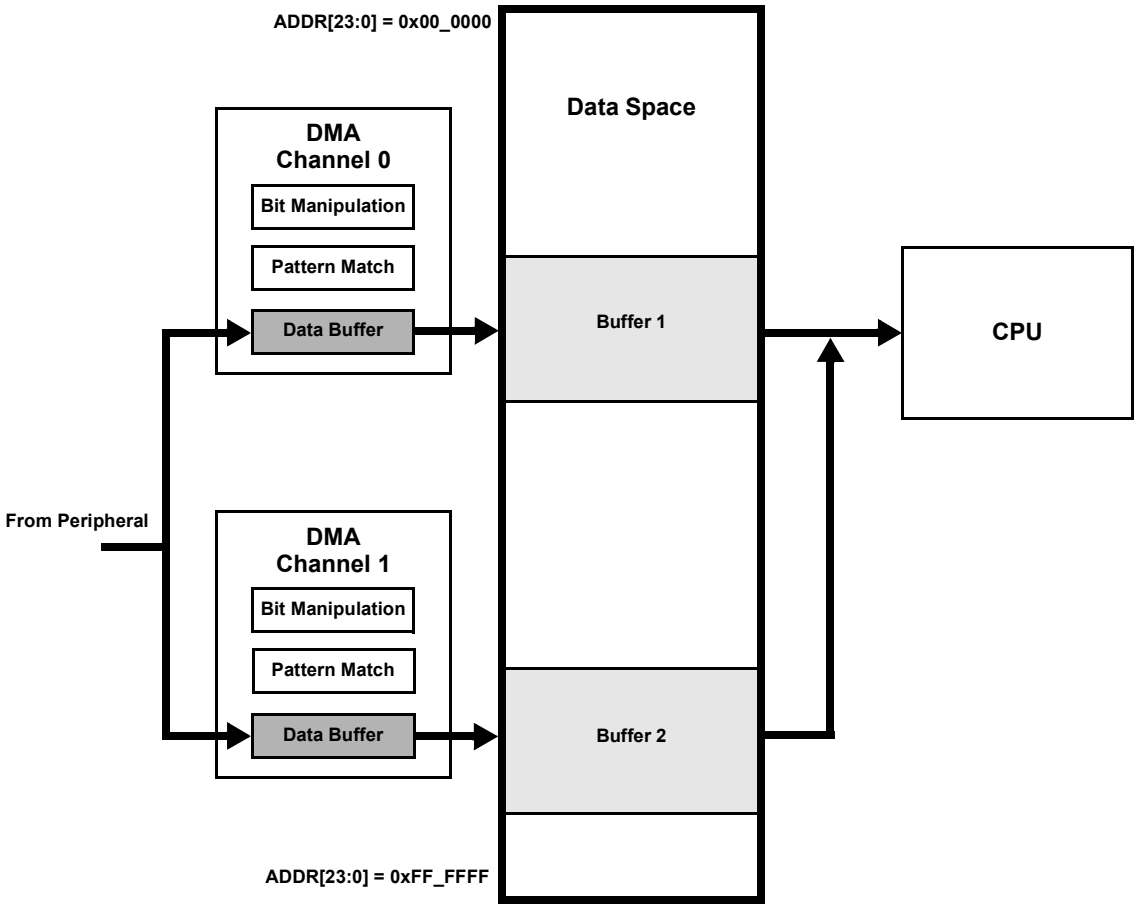
Transmit data as a responder:

1. User's SW sets up DMA Channel 0 and Channel 1, part of boot-up or initialization routine.
2. User's SW sets up peripheral for transmit, part of boot-up or initialization routine.
3. User's SW writes Buffer 1, part of boot-up or initialization routine.
4. User's SW gets INT when peripheral triggers DMA for transmit data (in response to the host requesting data).
5. DMA Channel 0 reads from Buffer 1 while user's SW writes to Buffer 2.
6. See steady-state description in 13.4.11.2. Ping-Pong Transmit Steady State.

13.4.11.4 Ping-Pong Data Receive

When the DMA Controller is set up to receive incoming data from a peripheral, the setup for the ping-pong operation can be seen in Figure 13-9.

Figure 13-9. Ping-Pong – Receive

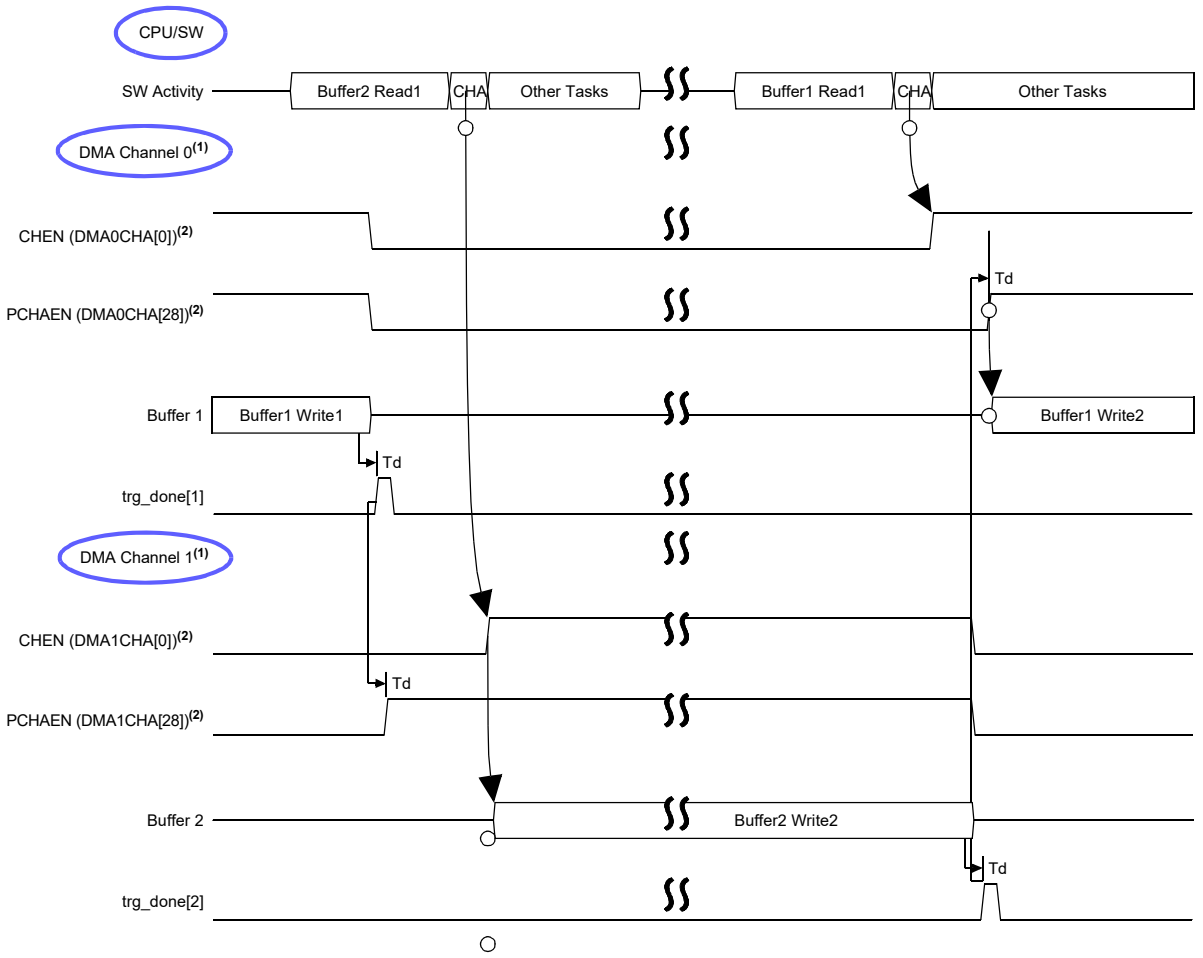


13.4.11.5 Ping-Pong Receive Steady State

Figure 13-10 illustrates a ping-pong operation involving two DMA channels, connected as shown in Figure 13-9, alternately writing the data to the ping-pong buffers (FIFO) from a common peripheral responsible for receiving the data. The CPU remains in full control of the operation without having to manually keep track of the DMA channel's transactions. The CPU is free to perform other tasks until completion while the DMA is operating independently. When the CPU has completed its tasks, it simply enables the appropriate DMA channel via the PCHAEN bit. When the DMA channel has

completed its transfer, it asserts its trigger output signal, which enables the other DMA channel via PCHAEN (with PPEN set high). Note that when operating in Ping-Pong mode (PPEN = 1), both the CHEN and PCHAEN bits must be set high to enable the DMA channel. This allows the data flow exchange, which happens upon the next trigger by the receiving peripheral to be independent of the completion order.

Figure 13-10. Ping-Pong Receive Operation in Steady State



Notes:

1. DMA Channel 0 finishes before the CPU's completion of its task, whereas DMA Channel 1 finishes after.
2. Both the CHEN (Non-Repeated Modes) and PCHAEN bits are hardware cleared once the transaction is completed.

13.4.11.6 Ping-Pong Receive Initialization

The following description serves only as a guideline for the firmware implementation.

Receive data as an initiator:

- User's SW sets up peripheral for receive
- User's SW sets up DMA Channel 0 and Channel 1
- User's SW enables DMA Channel 0 and sets INTOEN (DMA0CH)

- User's SW gets INT1 when DMA Channel 0 fills up Buffer 1
- User's SW reads from Buffer 1 while DMA Channel 1 writes to Buffer 2
- See steady-state description in 13.4.11.5. Ping-Pong Receive Steady State above

Receive data as a responder:

- User's SW sets up DMA Channel 0 and Channel 1, part of boot-up or initialization routine
- User's SW sets up peripheral for receive, part of boot-up or initialization routine
- User's SW enables DMA Channel 0 and sets INTOEN (DMA0CH)
- User's SW gets INT1 when DMA Channel 0 fills up Buffer 1
- User's SW reads from FIFO1 while DMA Channel 1 writes to Buffer 2
- See steady-state description in 13.4.11.5. Ping-Pong Receive Steady State above

13.4.12 Bit Manipulation

The 32-bit DMA Controller can perform real-time bit manipulation on the 32-bit data word being moved from the memory-mapped source to the destination location. The bit manipulation is made up of invert, clear and set functions, connected with their associated mask register. Note that each DMA channel is equipped with a one-bit manipulation logic block, and bit manipulation is applied to the DMABUF[31:0] bits during the write cycle, when the data are being stored to the destination location, once the bandwidth has been allocated to the requesting DMA channel.

Each mask register bit (see 13.3.13. DMAxINV, 13.3.11. DMAxCLR and 13.3.12. DMAxSET) is responsible for enabling its bit manipulation function onto the corresponding data bit, where logic '0' maintains the original bit value of the input data. The data coming out of the bit manipulation logic block can be expressed as:

```
{[DMABUF[n] XOR DMAINVx[n]]
AND ~DMACLRx[n]}
OR DMASETx[n]
```

13.4.12.1 Priority

Based on the connection shown, the set function is given the highest priority, followed by clear and invert, respectively. This means if the same bit position is enabled for more than one bit manipulation function, the higher priority function becomes dominant. As a result, the output data's logic value only reflects the highest bit manipulation function.

13.4.12.1.1 Channel Chaining

Channel Chaining is used to chain two or more channels together so that one channel triggers the other one. When the first channel in the sequence, often referred to as the initiating channel, completes its designated task, it sends a signal indicating completion. This signal acts as a trigger for the subsequent channel, known as the chained channel.

13.4.13 Pattern Match

When the content of the incoming data is required in making decisions in real time, the 32-bit DMA Controller can recognize a data pattern in its internal buffer being transferred from the source to the destination locations. The pattern match capability, when enabled, allows a user-programmable data pattern to be compared against a (partial) content of DMABUF[31:0]. Upon match detection, the DMA Controller invokes its interrupt to inform the CPU to take further action. This feature is implemented with control, status and pattern with its corresponding mask registers, along with digital control and comparator units, as shown in Figure 13-11.

13.4.13.1 Real-Time Matching

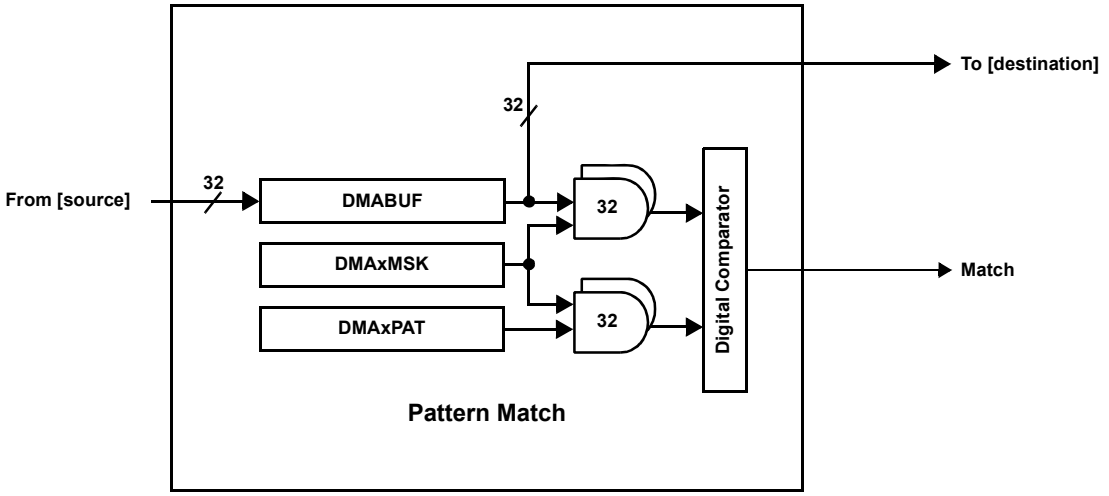
The MATEN bit (DMAxCH[2]) is used to enable the feature per selected channel; see 13.3.5. DMAxCH. Once enabled, the contents of DMABUF[31:0] and DMAxPAT[31:0] are subjected to

the corresponding DMAxMSK[31:0] bits value and compared against each other in an array of two input digital comparators, as shown below. When a match is detected, the DMA Controller proceeds to invoke its channel interrupt output while setting the MATCH bit (DMAxSTAT[1]) accordingly. Note that the DMA operation continues until completion, unaffected by the real-time matching. Upon interrupt detection by the CPU, the user's software is expected to start processing the incoming data until it comes across the pattern. It may also be necessary to update the DMAxPAT[31:0]/ DMAxMSK[31:0] bits for proper pattern matching associated with the current incoming data stream.

Notes:

1. When a match is first detected, the data still reside in the Data Buffer register. Depending on the bus infrastructure, it takes a varying number of clock cycles for it to reach the destination. It is up to the user's software to manage this latency period to properly recover the correct data.
2. It is up to the user's software to timely clear the MATCH bit (DMAxSTAT[1]) in order to prevent an overrun-like condition, where multiple matches occur with no CPU response. Currently, there is no capability to flag this condition.
3. The pattern match feature does not take SIZE[1:0] into account, so it is left to the user's software to set the mask accordingly.

Figure 13-11. Pattern Match



Note: The diagram shown above is intended for conceptual illustration only.

13.5 Application Examples

13.5.1 Basic Setup

To set up a DMA channel for any data transfer:

1. Enable the DMA Controller (ON = 1) and select an appropriate channel priority scheme by setting or clearing the PRIORITY bit.
2. Program DMAHIGH and DMALOW with the appropriate upper and lower address boundaries for data RAM operations.
3. Select the DMA channel to be used and disable its operation (CHEN = 0).
4. Program the appropriate source and destination addresses for the transaction into the channel's DMAxSRC and DMAxDST registers. For PIA Mode Addressing, use the base address value.
5. Program the RELOAD registers for source address, destination address and count depending on the transfer modes used.

6. Program the DMAxCNT register for the number of triggers per transfer (One-Shot or Continuous modes) or the number of words (bytes) to be transferred (Repeated modes).
7. Configure the SIZE[1:0] bits to select the data size.
8. Program the TRMODE[1:0] bits to select the Data Transfer mode.
9. Program the SAMODE[1:0] and DAMODE[1:0] bits to select the Addressing mode.
10. Program the CHSEL[7:0] bits to select the trigger source
11. Enable the DMA channel by setting the CHEN bit.
12. Enable the trigger source interrupt.

13.5.2 Standard Operation (Data Transfer)

A basic example of using the DMA Controller is moving a constant stream of data from a serial communication channel, such as a UART, and buffering it in a location in data RAM until the CPU can process it. In this example, DMA0 is used to service the UART for 16 data transfers in one iteration. It is configured as follows:

- DMA0 is configured to use the UART's receive interrupt as a trigger.
- DMA0 is programmed to use a single source address; to auto-increment the destination address and to use Repeated One-Shot Data Transfer mode.
- DMA0SRC is programmed with the address of the UART's receive buffer; DMA0DST is programmed with an address in data RAM.
- To support 16 transfers in one iteration, DMA0CNT is programmed with 0015h.

In this configuration, the sequence of events is as follows:

1. When the UART triggers a receive interrupt, DMA0 transfers the data from the buffer to a data RAM location.
2. After the transfer, the destination address is incremented.
3. After 16 interrupts, DMA0CNT is decremented to 0000h. Because this is a Repeated mode transfer, the original values of DMA0DST and DMA0CNT are reloaded and the cycle repeats.

This process allows the CPU to perform different tasks other than buffering incoming serial data and processes the data when it has the time. Because the DMA is overwriting the same 16 locations in memory, it is assumed that the CPU will be able to retrieve the fresh data first.

13.5.3 Nested Operation (Wait State Generation)

DMA channels may be nested, using one channel to trigger another in performing a data transfer. When one of the microcontroller's general purpose timers is included, it becomes possible to generate a fixed delay between a service request and the data transfer. In this case, DMA0 and DMA1 are used to service a UART after a forced Wait state. DMA Channels 0 and 1 are preconfigured as follows:

- DMA Channel 0 is configured to use the UART's receive interrupt as a trigger.
- DMA0SRC is programmed with an address in data RAM; DMA0DST is programmed with the address of the T0CON register.
- DMA Channel 1 is configured to use Timer0's interrupt as a trigger.
- DMA1SRC is programmed with the address of the UART's receive buffer; DMA0DST is programmed for the address of a destination in data RAM.

The sequence of events is as follows:

1. When the UART sends an interrupt, DMA0 transfers data into Timer0's Control register.
2. This causes Timer0 to count down once for a fixed interval (the Wait state), then generates an interrupt.

3. When Timer0 sends its interrupt, DMA1 is triggered and transfers data from the UART to data RAM.
4. Note that in this case, neither DMA channel was servicing the module from which it received its trigger.

13.5.4 Cascaded Operation (SPI Duplex Servicing)

Another method is to cascade two DMA channels together, allowing one to perform part of a function and then trigger a second channel to perform the other part. A good example is an SPI module operating in Client mode. Using two cascaded DMA channels allows automatic duplex operation, alternately receiving and sending data without the CPU's intervention. DMA0 (the read channel) and DMA1 (the write channel) are configured as follows:

- DMA0 is configured to use the SPI's transfer interrupt as the trigger; for Repeated One-Shot transfers, for a fixed source address and a fixed destination address
- DMA0SRC is programmed with the address of SPIBUF, while DMA0DST is programmed with a destination address in data RAM
- DMA0CNT is programmed with 0001h (its default)
- DMA1 is configured to use the DMA0 interrupt as the trigger; for Repeated One-Shot transfers, and for fixed source and destination addresses
- DMA1SRC is programmed with a data source address in data RAM, while DMA1DST is programmed with the address of SPIBUF
- DMA1CNT is programmed with 0001h

The sequence of events is as follows:

1. When the SPI receives data, it causes an SPI transfer interrupt.
2. This triggers DMA0 to transfer the data from the SPI buffer into RAM; at the completion of the transfer, the DMA0 interrupt is triggered.
3. The DMA0 interrupt triggers DMA1 to move data out of the data RAM location into SPIBUF to be transmitted. The process ends at this point.

For simplicity, this example moves one word of data in and out of the SPI. By changing the SAMODEx and DAMODEx bits for DMA0 and DMA1, respectively, and using different values for DMAxCNT, it is also possible to create mutliword buffers for larger duplex transactions.

13.6 DMA Interrupts

Each DMA channel has its own set of four interrupt flags, used to indicate a range of conditions during and following data transfers. Setting any of these flags with an interrupt event causes the device-level DMA Channel x Interrupt Flag (DMAxIF) to be set. With one exception, these flags are always enabled and not configurable. The DMAxIE bits, located in the IECx interrupt registers, will determine if a device-level interrupt is actually generated.

Since any of the DMA channel's individual event flags can trigger a device-level interrupt for the channel, the user must include a method within the ISR to determine which flag triggered the interrupt.

The four DMA channel interrupts are:

- DMA Completion Interrupt
- DMA Halfway Point Interrupt
- Overrun Interrupt
- Pattern Match Interrupt

Apart from these, the DMA trap is generated when there are bus error conditions, such as:

- Address Fault

- Bus Write Error
- Bus Read Error

In all of these conditions, the DMA will suspend its operation and the CHEN bit will be cleared.

13.6.1 DMA Channel Interrupts

13.6.1.1 DMA Completion Interrupt

The DONE bit (DMAxSTAT[5]) indicates the completion status of the last DMA operation. It is automatically set when DMAxCNT decrements to 0000h during a One-Shot or Continuous DMA transaction. When the DONE bit gets set, an interrupt is generated, indicating the DMA transfer completion.

By also examining the corresponding CHEN bit (DMAxCH[0]), it is possible to gain additional information on the status of the previous and current transactions. The possible interpretations are shown in Table 13-9.

Note that DONE remains cleared (= 0) when any Repeated Transfer modes are being used. This is because the address registers and transaction counters automatically reload, and the transaction automatically repeats when DMAxCNT decrements to 0000h. Repeated mode transfers must be terminated in software by clearing the CHEN bit.

The DONEEN bit (DMAxCH[3]) enables the DMA completion interrupt. When DONEEN is not set, the interrupt will not be generated.

Table 13-9. DMA Transaction Status

Bit Status		DMA Transaction Status
DONEIF	CHEN	
0	0	Previous transaction ended without completion
0	1	Current transaction is not yet complete
1	0	Previous transaction ended with completion
1	1	Previous transaction ended with completion

13.6.1.2 DMA Halfway Point Interrupt

The HALF interrupt flag (DMAxSTAT[4]) is an optional interrupt that indicates that the DMAxCNT register is at the halfway point between its original programmed value and 0000h. This can be used with the DONE interrupt to monitor the progress of the DMA transfer.

When enabled, HALF is set only when DMAxCNT reaches the halfway mark, but not thereafter. This results in a non-persistent interrupt. In Repeated modes, the DMA Controller attempts to set HALF every time DMAxCNT reaches the halfway point, whether or not the bit has been cleared. It is the user's responsibility to clear the bit after it has been set.

The HALFEN bit (DMAxCH[1]) enables the halfway point interrupt. If HALFEN is not set, then the interrupt will not be generated.

13.6.1.3 Overrun Interrupt

When a DMA channel receives a trigger while its CHREQ bit is already set (either by software or another hardware trigger), an Overrun condition occurs. This condition indicates that the channel is being requested before its current transaction is finished. This implies that the active channel may not be able to keep up with the demands from the peripheral module being serviced, which may result in data loss. An Overrun condition causes the OVERRUN flag (DMAxSTAT[3]) to be set.

Note that the OVERRUN flag being set does not cause the current DMA operation to terminate. Therefore, the channel for which OVERRUN is set does not need to be the active channel.

Setting the priority scheme correctly also helps to avoid overrun errors. For example, if one of the channels operates more frequently, a Fixed Priority scheme with that as the channel will help to reduce overrun interrupts.

13.6.1.4 Pattern Match Interrupt

When the MATEN bit is enabled, the contents of DMABUF[31:0] and DMAxPAT[31:0] are subjected to the corresponding DMAxMSK[31:0] bits' value and compared against each other . When a match is detected, the DMA Controller proceeds to invoke its channel interrupt output while setting the MATCH bit (DMAxSTAT[1]) accordingly. For more information, refer to 13.4.13.1. Real-Time Matching.

13.6.2 DMA Bus Error Trap

13.6.2.1 Address Fault

DMA requires that all its Address Pointers be naturally aligned and within their designated ranges. When DMA tries to access an out of range address or a misaligned address, a DMA trap will be generated on a channel basis.

Address Faults are detected as follows:

- DMA operation has crossed the data RAM address boundaries set by the DMAHIGH and DMALOW registers.
- For 16-bit word operation (SIZE[1:0] = `01`) when the pointers are not 16-bit word-aligned.
- For 32-bit word operation (SIZE[1:0] = `10`) when the pointers are not 32-bit word-aligned.

The user can determine the Fault condition by reading DMAxSTAT for the value of DMAFLT[1:0]. These flag bits are set on any operation that attempts to read data from, or write data to, an address outside of the DMA boundaries or a misaligned address as mentioned above. An address Fault interrupt immediately terminates any DMA transaction in progress.

13.6.2.2 Bus Read Error

The 32-bit DMA module responds to read and write errors on the bus. A bus read error Fault occurs when the data read by the DMA are invalid. This might occur, for example, under the following conditions:

- The read was not allowed because of the device security settings
- The read was attempted at an unimplemented address

When a bus read error occurs, the DMAFLT2 (DMAxSTAT[8]) bit is set so that user software can detect the read error. When a bus read error occurs, the DMA channel can optionally suspend operation based on the setting of the Read Error Trap Enable bit, RETEN (DMAxCH[16]). When the RETEN bit is cleared (default), the DMA channel will continue operation when a bus read error is encountered. The data obtained from the bus read will be transferred to the write destination by the DMA channel. When the RETEN bit is set, the DMA channel will suspend operation and the DMA trap will be asserted.

The RETEN control bit allows the user to make a trade-off between time-sensitive data streaming applications and applications where the data must be accurate. For example, an audio streaming application may be able to tolerate an occasional data error even if it produces audible artifacts. In contrast, a data error in a closed-loop control system could have more severe consequences, and it would be safer to suspend DMA transfers until the cause of the data error has been resolved.

13.6.2.3 Bus Write Error

A bus write error occurs when the data write by the DMA could not be completed. This might occur, for example, under the following conditions:

- The write was not allowed because of the device security settings.
- The write was attempted at an unimplemented address.

When a bus write error occurs, the DMAFLT3 (DMAxSTAT[9]) bit is set so that user software can detect the write error. When a bus write error is detected, the DMA halts its operation and a DMA trap is asserted.

13.7 Operation During Sleep and Idle Modes

Although the DMA Controller can be thought of as an extension of the CPU, it is treated as a peripheral when it comes to power-saving operations. Like other peripherals, the DMA Controller also uses Peripheral Module Disable (PMD) bits to further tailor its operation in Low-Power states.

13.7.1 Idle Mode

Since the system clock persists through Idle mode, the DMA Controller supports the operation in Idle mode. However, the DMA Controller provides an option to suspend operation upon Idle mode entry using the Stop in Idle Mode bit, DMASIDL (DMACON[13]). In this mode, there will not be any active DMA operations. The controller resumes any partially completed transactions on exiting from Idle mode.

13.7.2 Sleep Mode

When the device enters Sleep mode, all clock sources to the module are shut down and stay at logic '0'. Any transfers in progress are aborted. The controller will not resume any partially completed transactions on exiting from Sleep mode.

Register contents are not affected by the device entering or leaving Sleep mode. It is recommended that DMA transactions be allowed to finish before entering Sleep mode.

13.7.3 Peripheral Module Disable (PMD) Register

The Peripheral Module Disable (PMD) registers provide a method to disable DMA channels by stopping all clock sources supplied to that channel. For efficient usage, each DMA PMD bit controls a block of up to four channels; for example, DMA0MD disables channels 0 through 3, DMA1MD disables channels 4 through 7 and so on. Setting a DMA PMD bit disables all channels in that block.

When all channels are disabled via their corresponding PMD control bits, the DMA Controller is in a Minimum Power Consumption state. The module-level registers (DMACON, DMABUF, DMAHIGH and DMALOW) remain active. However, the control and status registers associated with any disabled channels will be disabled, so writes to those registers will have no effect and read values will be invalid.

13.7.4 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the DMA Controller and all channels to be turned off, and any transfers in progress to be aborted. All buffer and address registers are initialized to 0 (except the CNT register that holds a value of 1 on Reset).

14. PWM

This section describes the features and use of the High-Resolution Pulse-Width Modulation (PWM).

This flexible module provides features to support many types of Motor Control (MC) and Power Control (PC) applications, including:

- AC-to-DC Converters
- DC-to-DC Converters
- AC and DC Motor Control: Brushed DC, BLDC, PMSM, ACIM, SRM, Stepper, etc.
- Inverters
- Battery Chargers
- Digital Lighting
- Power Factor Correction (PFC)
- Four Independent PWM Generators, each with Dual Outputs
- Operating modes:
  - Independent Edge PWM mode
  - Variable Phase PWM mode
  - Independent Edge PWM mode, Dual Output
  - Center-Aligned PWM mode
  - Double Update Center-Aligned PWM mode
  - Dual Edge Center-Aligned PWM mode
- Output modes:
  - Complementary
  - Independent
  - Push-Pull
- Dead-Time Generator
- Dead-Time Compensation
- Leading-Edge Blanking (LEB)
- Output Override for Fault Handling
- Flexible Period/Duty Cycle Updating Options
- PWM Control Inputs (PCI) for PWM Pin Overrides and External PWM Synchronization
- Advanced Triggering Options
- Combinatorial Logic Output
- PWM Event Outputs

14.1 Device-Specific Information

Table 14-1. PWM Summary Table

PWM Module Instances	Number of PWM Events	Combinatorial PWM Logic	Fine Edge Placement	Outputs per Instance	Clock Source	Peripheral Bus Speed	CLKGEN Instance
4	6	6	No	2	Table 14-2	Standard	CLKGEN5

Table 14-2. MCLKSEL PWM Master Clock Selection

Value	Description
1	CKLGEN5
0	Standard Speed Peripheral Clock

Table 14-3. PCI Source Selection (PSS)

Value	Description
11111	CLC1
11110	Reserved
11101	Comparator 3 output
11100	Comparator 2 output
11011	Comparator 1 output
11010	PWM Event D
11001	PWM Event C
11000	PWM Event B
10111	PWM Event A
10110	Device pin, PCI[22]
10101	Device pin, PCI[21]
10100	Device pin, PCI[20]
10011	Device pin, PCI[19]
10010	RPn input, PCI18R
10001	RPn input, PCI17R
10000	RPn input, PCI16R
01111	RPn input, PCI15R
01110	RPn input, PCI14R
01101	RPn input, PCI13R
01100	RPn input, PCI12R
01011	RPn input, PCI11R
01010	RPn input, PCI10R
01001	RPn input, PCI9R
01000	RPn input, PCI8R
00111-00100	Reserved
00011	Internally connected to Combo Trigger B
00010	Internally connected to Combo Trigger A
00001	Internally connected to the output of PWMPCI[2:0] MUX
00000	Tied to '0'

Table 14-4. Combinatorial PWM Logic Source Selection

Value	Description
1111-1000	Reserved
0111	PWM4L
0110	PWM4H
0101	PWM3L
0100	PWM3H
0011	PWM2L
0010	PWM2H
0001	PWM1L
0000	PWM1H

Table 14-5. Combinatorial PWM Logic Destination Selection

Value	Description
111-100	Reserved
011	Logic Function is assigned to PWM4
010	Logic function is assigned to PWM3
001	Logic function is assigned to PWM2
000	No assignment, combinatorial PWM logic function is disabled

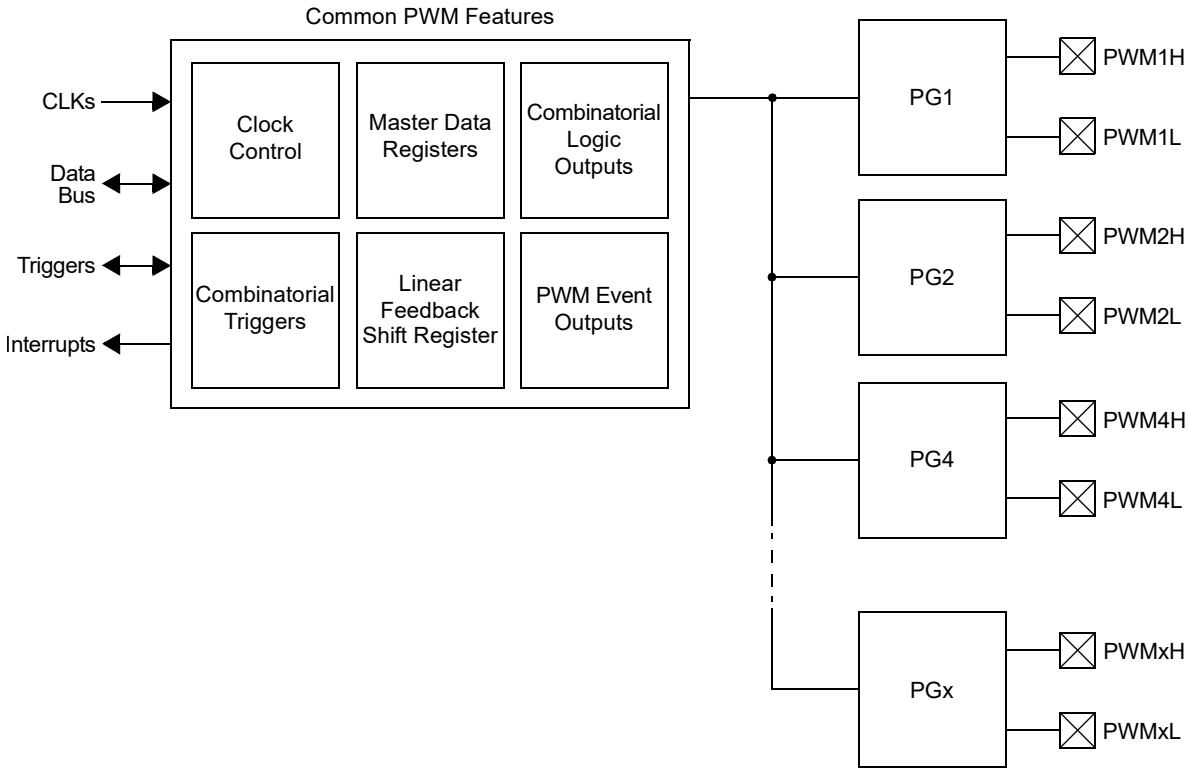
Table 14-6. PWM Source Selection

Value	Description
111-100	Reserved
011	PWM Generator 4
010	PWM Generator 3
001	PWM Generator 2
000	PWM Generator 1

14.2 Architectural Overview

The PWM module consists of a common set of controls and features and multiple instantiations of PWM Generators (PGx). Each PWM Generator can be independently configured or multiple PWM Generators can be used to achieve complex multiphase systems. PWM Generators can also be used to implement sophisticated triggering, protection and logic functions. A high-level block diagram is shown in Figure 14-1.

Figure 14-1. PWM High-Level Block Diagram



Each PWM Generator behaves as a separate peripheral that can be independently enabled from the other PWM Generators. Each PWM Generator consists of a signal generator and an Output Control block.

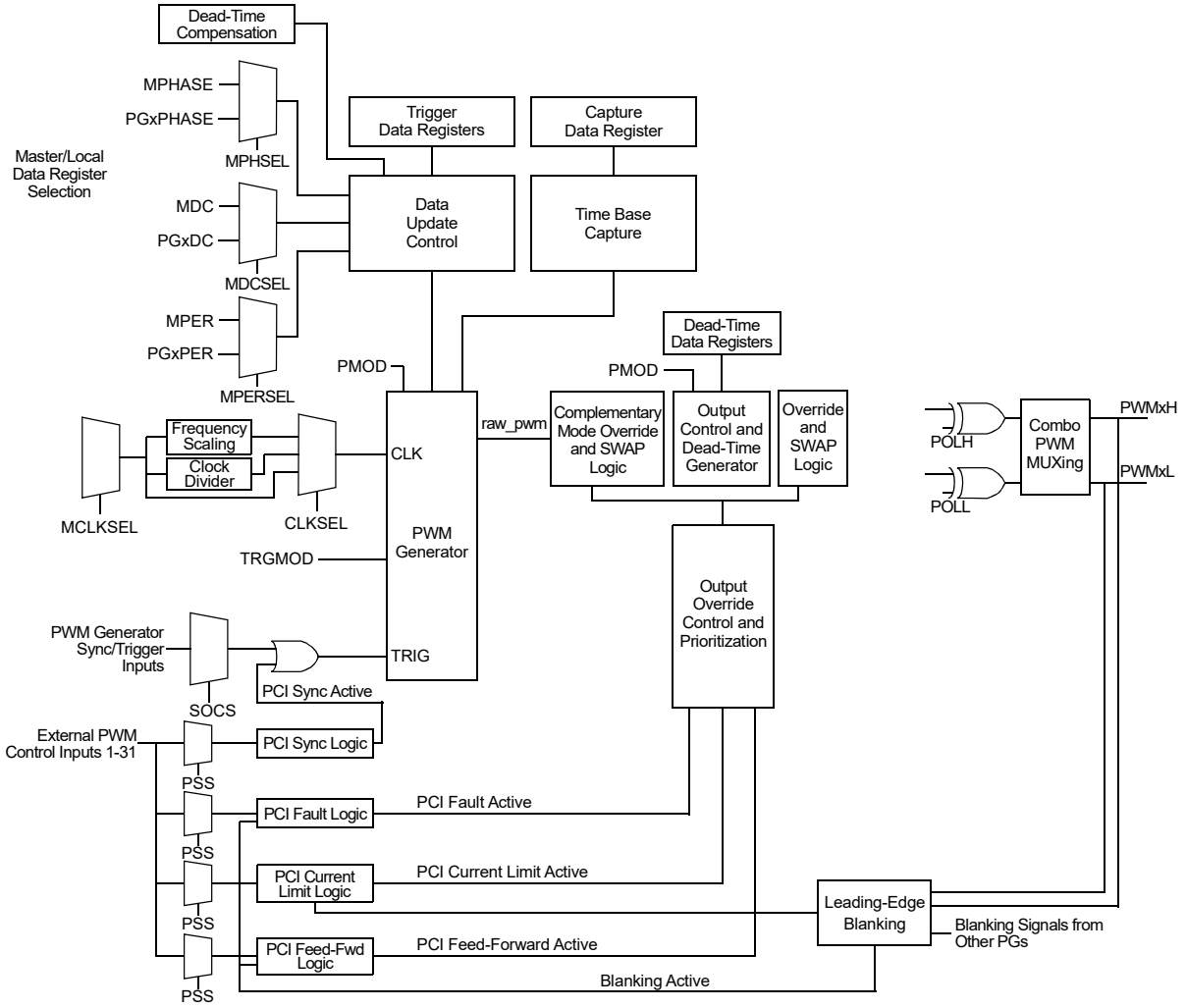
The PWM Generators use ‘events’ to trigger other PWM Generators, ADC conversions and external operations. Each PWM Generator accepts a trigger input and produces a trigger output. The trigger input signals the PWM Generator when to start a new PWM period. The trigger output is generated when the trigger time value is equal to the PWM Generator timer value.

Output Control blocks provide the capability to alter the base PWM signal sent to the output pins and incorporate several functions, including:

- Output Mode Selection (Complementary, Push-Pull, Independent)
- Dead-Time Generator
- PWM Control Input (PCI) block
- Leading-Edge Blanking (LEB)
- Override

Each PWM Generator Output block is associated with the control of two PWM output pins. Output blocks contain a PWM Control Input (PCI) that can be used for many purposes, including Fault detection, external triggering and interfacing with other peripherals. The LEB block works in conjunction with the PCI block and allows PCI inputs to be ignored during certain periods of the PWM cycle. The Override block determines the PWM output pin states during various types of events, including Faults, current limit and feed-forward control. A block diagram of a single PWM Generator is shown in Figure 14-2.

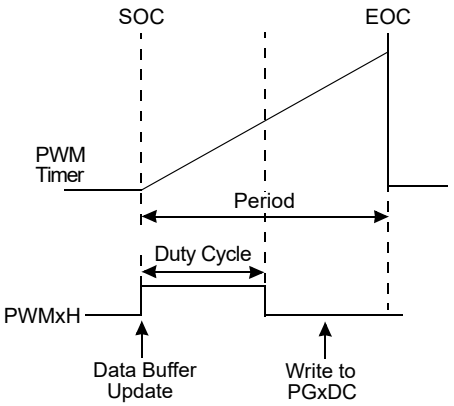
Figure 14-2. Single PWM Generator



PWM Generator operation is based on triggers. To generate a PWM cycle, a SOC (Start-of-Cycle) trigger must be received; the trigger can either be self-triggered or from an external source. Figure 14-3 illustrates a basic PWM waveform, showing SOC and EOC (End-of-Cycle) events. The PWMxH output starts the cycle 'active' (logic high), and when the internal counter reaches the duty cycle value, it transitions to 'inactive' (logic low). EOC is reached when the counter value reaches the period value.

Some operating modes and output modes use multiple counter cycles to produce a single PWM cycle. Refer to 14.4.2.2. PWM Modes and 14.4.2.3. Output Modes for more information.

Figure 14-3. Basic PWM Waveform



14.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1004	PCLKCON	31:24								
		23:16								
		15:8								LOCK
		7:0			DIVSEL[1:0]					MCLKSEL
0x1004	FSCL	31:24								
		23:16					FSCL[19:16]			
		15:8	FSCL[15:8]							
		7:0	FSCL[7:4]							
0x1008	FSMINPER	31:24								
		23:16					FSMINPER[19:16]			
		15:8	FSMINPER[15:8]							
		7:0	FSMINPER[7:4]				Reserved[3:0]			
0x100C	MPHASE	31:24								
		23:16					MPHASE[19:16]			
		15:8	MPHASE[15:8]							
		7:0	MPHASE[7:4]				Reserved[3:0]			
0x1010	MDC	31:24								
		23:16					MDC[19:16]			
		15:8	MDC[15:8]							
		7:0	MDC[7:4]				Reserved[3:0]			
0x1014	MPER	31:24								
		23:16					MPER[19:16]			
		15:8	MPER[15:8]							
		7:0	MPER[7:4]				Reserved[3:0]			
0x1018	LFSR	31:24								
		23:16								
		15:8		LFSR[14:8]						
		7:0	LFSR[7:0]							
0x101C	CMBTRIG	31:24								
		23:16	CTB8EN	CTB7EN	CTB6EN	CTB5EN	CTB4EN	CTB3EN	CTB2EN	CTB1EN
		15:8								
		7:0	CTA8EN	CTA7EN	CTA6EN	CTA5EN	CTA4EN	CTA3EN	CTA2EN	CTA1EN
0x1020	LOGCONA	31:24								
		23:16								
		15:8	PWMS1A[3:0]				PWMS2A[3:0]			
		7:0	S1APOL	S2APOL	PWMLFA[1:0]			PWMLFAD[2:0]		
0x1024	LOGCONB	31:24								
		23:16								
		15:8	PWMS1B[3:0]				PWMS2B[3:0]			
		7:0	S1BPOL	S2BPOL	PWMLFB[1:0]			PWMLFBD[2:0]		
0x1028	LOGCONC	31:24								
		23:16								
		15:8	PWMS1C[3:0]				PWMS2C[3:0]			
		7:0	S1CPOL	S2CPOL	PWMLFC[1:0]			PWMLFCD[2:0]		
0x102C	LOGCOND	31:24								
		23:16								
		15:8	PWMS1D[3:0]				PWMS2D[3:0]			
		7:0	S1DPOL	S2DPOL	PWMLFD[1:0]			PWMLFDD[2:0]		
0x1030	LOGCONE	31:24								
		23:16								
		15:8	PWMS1E[3:0]				PWMS2E[3:0]			
		7:0	S1EPOL	S2EPOL	PWMLFE[1:0]			PWMLFED[2:0]		
0x1034	LOGCONF	31:24								
		23:16								
		15:8	PWMS1F[3:0]				PWMS2F[3:0]			
		7:0	S1FPOL	S2FPOL	PWMLFF[1:0]			PWMLFFD[2:0]		

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1038	PWMEVTA	31:24								
		23:16								
		15:8	EVTAOEN	EVTAPOL	EVTASTRD	EVTASYNC				
		7:0	EVTASEL[3:0]					EVTAPGS[2:0]		
0x103C	PWMEVTB	31:24								
		23:16								
		15:8	EVTBOEN	EVTBPOL	EVTBSTRD	EVTBSYNC				
		7:0	EVTBSEL[3:0]					EVTBPGS[2:0]		
0x1040	PWMEVTC	31:24								
		23:16								
		15:8	EVTCOEN	EVTCPOL	EVTCSTRD	EVTCSYNC				
		7:0	EVTCSEL[3:0]					EVTCPGS[2:0]		
0x1044	PWMEVTD	31:24								
		23:16								
		15:8	EVTDOEN	EVTDPOL	EVTDSTRD	EVTDSYNC				
		7:0	EVTDSEL[3:0]					EVTDPGS[2:0]		
0x1048	PWMEVTE	31:24								
		23:16								
		15:8	EVTEOEN	EVTEPOL	EVTESTRD	EVTESYNC				
		7:0	EVTESEL[3:0]					EVTEPGS[2:0]		
0x104C	PWMEVTF	31:24								
		23:16								
		15:8	EVTFOEN	EVTFPOL	EVTFSTRD	EVTFSYNC				
		7:0	EVTFSEL[3:0]					EVTFPGS[2:0]		
0x1050	PG1CON	31:24	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
		23:16	TRGMOD[1:0]				SOCS[3:0]			
		15:8	ON					TRGCNT[2:0]		
		7:0	Reserved			CLKSEL[1:0]		MODSEL[2:0]		
0x1054	PG1STAT	31:24								
		23:16								
		15:8	SEVT	FLTEVT	CLEVT	FFEVT	SACT	FLTACT	CLACT	FFACT
		7:0	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
0x1058	PG1IOCON	31:24		CAPSRC[2:0]					PPSEN	DTCMPSEL
		23:16			PMOD[1:0]		PENH	PENL	POLH	POLL
		15:8	CLMOD	SWAP	OVRENH	OVRENL	OVRDAT[1:0]		OSYNC[1:0]	
		7:0	FLTDAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBDAT[1:0]	
0x105C	PG1EVT	31:24	FLTIEN	CLIEN	FFIEN	SIEN			IEVTSEL[1:0]	
		23:16	ADTR2EN3	ADTR2EN2	ADTR2EN1	ADTR1OFS[4:0]				
		15:8	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
		7:0	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
0x1060	PG1FPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS			PSS[4:0]		
0x1064	PG1CLPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS			PSS[4:0]		
0x1068	PG1FFPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS			PSS[4:0]		
0x106C	PG1SPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS			PSS[4:0]		
0x1070	PG1LEB	31:24								
		23:16					PHR	PHF	PLR	PLF
		15:8	LEB[15:8]							
		7:0	LEB[7:4]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1074	PG1PHASE	31:24								
		23:16						PHASE[19:16]		
		15:8	PHASE[15:8]							
		7:0	PHASE[7:4]					Reserved[3:0]		
0x1078	PG1DC	31:24								
		23:16						DC[19:16]		
		15:8	DC[15:8]							
		7:0	DC[7:4]					Reserved[3:0]		
0x107C	PG1DCA	31:24								
		23:16								
		15:8						DCA[11:8]		
		7:0	DCA[7:4]					Reserved[3:0]		
0x1080	PG1PER	31:24								
		23:16						PER[19:16]		
		15:8	PER[15:8]							
		7:0	PER[7:4]					Reserved[3:0]		
0x1084	PG1TRIGA	31:24	CAHALF							
		23:16						TRIGA[19:16]		
		15:8	TRIGA[15:8]							
		7:0	TRIGA[7:4]					Reserved[3:0]		
0x1088	PG1TRIGB	31:24	CAHALF							
		23:16						TRIGB[19:16]		
		15:8	TRIGB[15:8]							
		7:0	TRIGB[7:4]					Reserved[3:0]		
0x108C	PG1TRIGC	31:24	CAHALF							
		23:16						TRIGC[19:16]		
		15:8	TRIGC[15:8]							
		7:0	TRIGC[7:4]					Reserved[3:0]		
0x1090	PG1DT	31:24		DTH[14:8]						
		23:16	DTH[7:0]							
		15:8		DTL[14:8]						
		7:0	DTL[7:0]							
0x1094	PG1CAP	31:24								
		23:16						CAP[19:16]		
		15:8	CAP[15:8]							
		7:0	CAP[7:4]							
0x1098	PG2CON	31:24	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
		23:16	TRGMOD[1:0]				SOCS[3:0]			
		15:8	ON					TRGCNT[2:0]		
		7:0	Reserved			CLKSEL[1:0]		MODSEL[2:0]		
0x109C	PG2STAT	31:24								
		23:16								
		15:8	SEVT	FLTEVT	CLEVT	FFEVT	SACT	FLTACT	CLACT	FFACT
		7:0	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
0x10A0	PG2IOCON	31:24		CAPSRC[2:0]					PPSEN	DTCMPSEL
		23:16			PMOD[1:0]		PENH	PENL	POLH	POLL
		15:8	CLMOD	SWAP	OVRENH	OVRENL	OVRDAT[1:0]		OSYNC[1:0]	
		7:0	FLTDAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBDAT[1:0]	
0x10A4	PG2EVT	31:24	FLTIEN	CLIEN	FFIEN	SIEN			IEVTSEL[1:0]	
		23:16	ADTR2EN3	ADTR2EN2	ADTR2EN1	ADTR1OFS[4:0]				
		15:8	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
		7:0	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
0x10A8	PG2FPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x10AC	PG2CLPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x10B0	PG2FFPCI	31:24	BPEN		BPSEL[2:0]		TERMPS		ACP[2:0]	
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS		TQSS[2:0]	
		15:8	TSYNCDIS		TERM[2:0]		AQPS		AQSS[2:0]	
		7:0	SWTERM	PSYNC	PPS		PSS[4:0]			
0x10B4	PG2SPCI	31:24	BPEN		BPSEL[2:0]		TERMPS		ACP[2:0]	
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS		TQSS[2:0]	
		15:8	TSYNCDIS		TERM[2:0]		AQPS		AQSS[2:0]	
		7:0	SWTERM	PSYNC	PPS		PSS[4:0]			
0x10B8	PG2LEB	31:24								
		23:16					PHR	PHF	PLR	PLF
		15:8	LEB[15:8]							
		7:0	LEB[7:4]							
0x10BC	PG2PHASE	31:24								
		23:16					PHASE[19:16]			
		15:8	PHASE[15:8]							
		7:0	PHASE[7:4]				Reserved[3:0]			
0x10C0	PG2DC	31:24								
		23:16					DC[19:16]			
		15:8	DC[15:8]							
		7:0	DC[7:4]				Reserved[3:0]			
0x10C4	PG2DCA	31:24								
		23:16								
		15:8					DCA[11:8]			
		7:0	DCA[7:4]				Reserved[3:0]			
0x10C8	PG2PER	31:24								
		23:16					PER[19:16]			
		15:8	PER[15:8]							
		7:0	PER[7:4]				Reserved[3:0]			
0x10CC	PG2TRIGA	31:24	CAHALF							
		23:16					TRIGA[19:16]			
		15:8	TRIGA[15:8]							
		7:0	TRIGA[7:4]				Reserved[3:0]			
0x10D0	PG2TRIGB	31:24	CAHALF							
		23:16					TRIGB[19:16]			
		15:8	TRIGB[15:8]							
		7:0	TRIGB[7:4]				Reserved[3:0]			
0x10D4	PG2TRIGC	31:24	CAHALF							
		23:16					TRIGC[19:16]			
		15:8	TRIGC[15:8]							
		7:0	TRIGC[7:4]				Reserved[3:0]			
0x10D8	PG2DT	31:24		DTH[14:8]						
		23:16	DTH[7:0]							
		15:8		DTL[14:8]						
		7:0	DTL[7:0]							
0x10DC	PG2CAP	31:24								
		23:16					CAP[19:16]			
		15:8	CAP[15:8]							
		7:0	CAP[7:4]							
0x10E0	PG3CON	31:24	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
		23:16	TRGMOD[1:0]				SOCS[3:0]			
		15:8	ON					TRGCNT[2:0]		
		7:0	Reserved			CLKSEL[1:0]		MODSEL[2:0]		
0x10E4	PG3STAT	31:24								
		23:16								
		15:8	SEVT	FLTEVT	CLEVT	FFEVT	SACT	FLTACT	CLACT	FFACT
		7:0	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
0x10E8	PG3IOCON	31:24		CAPSRC[2:0]					PPSEN	DTCMPSEL
		23:16			PMOD[1:0]		PENH	PENL	POLH	POLL
		15:8	CLMOD	SWAP	OVRENH	OVRENL	OVRDAT[1:0]		OSYNC[1:0]	
		7:0	FLTDAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBDAT[1:0]	

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x10EC	PG3EVT	31:24	FLTIEN	CLIEN	FFIEN	SIEN			IEVTSEL[1:0]	
		23:16	ADTR2EN3	ADTR2EN2	ADTR2EN1	ADTR1OFS[4:0]				
		15:8	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
		7:0	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
0x10F0	PG3FPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x10F4	PG3CLPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x10F8	PG3FFPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x10FC	PG3SPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x1100	PG3LEB	31:24								
		23:16					PHR	PHF	PLR	PLF
		15:8	LEB[15:8]							
		7:0	LEB[7:4]							
0x1104	PG3PHASE	31:24								
		23:16					PHASE[19:16]			
		15:8	PHASE[15:8]							
		7:0	PHASE[7:4]				Reserved[3:0]			
0x1108	PG3DC	31:24								
		23:16					DC[19:16]			
		15:8	DC[15:8]							
		7:0	DC[7:4]				Reserved[3:0]			
0x110C	PG3DCA	31:24								
		23:16								
		15:8					DCA[11:8]			
		7:0	DCA[7:4]				Reserved[3:0]			
0x1110	PG3PER	31:24								
		23:16					PER[19:16]			
		15:8	PER[15:8]							
		7:0	PER[7:4]				Reserved[3:0]			
0x1114	PG3TRIGA	31:24	CAHALF							
		23:16					TRIGA[19:16]			
		15:8	TRIGA[15:8]							
		7:0	TRIGA[7:4]				Reserved[3:0]			
0x1118	PG3TRIGB	31:24	CAHALF							
		23:16					TRIGB[19:16]			
		15:8	TRIGB[15:8]							
		7:0	TRIGB[7:4]				Reserved[3:0]			
0x111C	PG3TRIGC	31:24	CAHALF							
		23:16					TRIGC[19:16]			
		15:8	TRIGC[15:8]							
		7:0	TRIGC[7:4]				Reserved[3:0]			
0x1120	PG3DT	31:24		DTH[14:8]						
		23:16	DTH[7:0]							
		15:8		DTL[14:8]						
		7:0	DTL[7:0]							
0x1124	PG3CAP	31:24								
		23:16					CAP[19:16]			
		15:8	CAP[15:8]							
		7:0	CAP[7:4]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1128	PG4CON	31:24	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
		23:16	TRGMOD[1:0]				SOCS[3:0]			
		15:8	ON					TRGCNT[2:0]		
		7:0	Reserved			CLKSEL[1:0]		MODSEL[2:0]		
0x112C	PG4STAT	31:24								
		23:16								
		15:8	SEVT	FLTEVT	CLEVT	FFEVT	SACT	FLTACT	CLACT	FFACT
		7:0	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
0x1130	PG4IOCON	31:24		CAPSRC[2:0]					PPSEN	DTCMPSEL
		23:16			PMOD[1:0]		PENH	PENL	POLH	POLL
		15:8	CLMOD	SWAP	OVRENH	OVRENL	OVRDAT[1:0]		OSYNC[1:0]	
		7:0	FLTDAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBDAT[1:0]	
0x1134	PG4EVT	31:24	FLTIEN	CLIEN	FFIEN	SIEN			IEVTSEL[1:0]	
		23:16	ADTR2EN3	ADTR2EN2	ADTR2EN1	ADTR1OFS[4:0]				
		15:8	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
		7:0	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
0x1138	PG4FPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x113C	PG4CLPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x1140	PG4FFPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x1144	PG4SPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x1148	PG4LEB	31:24								
		23:16					PHR	PHF	PLR	PLF
		15:8	LEB[15:8]							
		7:0	LEB[7:4]							
0x114C	PG4PHASE	31:24								
		23:16					PHASE[19:16]			
		15:8	PHASE[15:8]							
		7:0	PHASE[7:4]				Reserved[3:0]			
0x1150	PG4DC	31:24								
		23:16					DC[19:16]			
		15:8	DC[15:8]							
		7:0	DC[7:4]				Reserved[3:0]			
0x1154	PG4DCA	31:24								
		23:16								
		15:8					DCA[11:8]			
		7:0	DCA[7:4]				Reserved[3:0]			
0x1158	PG4PER	31:24								
		23:16					PER[19:16]			
		15:8	PER[15:8]							
		7:0	PER[7:4]				Reserved[3:0]			
0x115C	PG4TRIGA	31:24	CAHALF							
		23:16					TRIGA[19:16]			
		15:8	TRIGA[15:8]							
		7:0	TRIGA[7:4]				Reserved[3:0]			
0x1160	PG4TRIGB	31:24	CAHALF							
		23:16					TRIGB[19:16]			
		15:8	TRIGB[15:8]							
		7:0	TRIGB[7:4]				Reserved[3:0]			

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1164	PG4TRIGC	31:24	CAHALF							
		23:16					TRIGC[19:16]			
		15:8	TRIGC[15:8]							
		7:0	TRIGC[7:4]				Reserved[3:0]			
0x1168	PG4DT	31:24		DTH[14:8]						
		23:16	DTH[7:0]							
		15:8		DTL[14:8]						
		7:0	DTL[7:0]							
0x116C	PG4CAP	31:24								
		23:16					CAP[19:16]			
		15:8	CAP[15:8]							
		7:0	CAP[7:4]							

14.3.1 Registers

There are two categories of Special Function Registers (SFRs) used to control the operation of the PWM module:

- Common, shared by all PWM Generators
- PWM Generator-specific. Auxiliary PWM generators that operate identically to primary PWM generators, but provide a total increase in number of PWM generators.

An ‘x’ in the register name denotes an instance of a PWM Generator.

A ‘y’ in the register name denotes an instance of a common function. An optional ‘w’ in the register name denotes an instance of an Auxiliary PWM Generator.

The LOCK bit in the PCLKCON register may be set in software to block writes to certain registers and bits (see 14.4.2. PWM Generator (PG) Features for more information). Writes to certain data and control registers are not safe at certain times of a PWM cycle or when the module is enabled.

14.3.2 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1004	PCLKCON	31:24								
		23:16								
		15:8								LOCK
		7:0			DIVSEL[1:0]					MCLKSEL
0x1004	FSCL	31:24								
		23:16					FSCL[19:16]			
		15:8	FSCL[15:8]							
		7:0	FSCL[7:4]							
0x1008	FSMINPER	31:24								
		23:16					FSMINPER[19:16]			
		15:8	FSMINPER[15:8]							
		7:0	FSMINPER[7:4]				Reserved[3:0]			
0x100C	MPHASE	31:24								
		23:16					MPHASE[19:16]			
		15:8	MPHASE[15:8]							
		7:0	MPHASE[7:4]				Reserved[3:0]			
0x1010	MDC	31:24								
		23:16					MDC[19:16]			
		15:8	MDC[15:8]							
		7:0	MDC[7:4]				Reserved[3:0]			
0x1014	MPER	31:24								
		23:16					MPER[19:16]			
		15:8	MPER[15:8]							
		7:0	MPER[7:4]				Reserved[3:0]			
0x1018	LFSR	31:24								
		23:16								
		15:8		LFSR[14:8]						
		7:0	LFSR[7:0]							
0x101C	CMBTRIG	31:24								
		23:16	CTB8EN	CTB7EN	CTB6EN	CTB5EN	CTB4EN	CTB3EN	CTB2EN	CTB1EN
		15:8								
		7:0	CTA8EN	CTA7EN	CTA6EN	CTA5EN	CTA4EN	CTA3EN	CTA2EN	CTA1EN
0x1020	LOGCONA	31:24								
		23:16								
		15:8	PWMS1A[3:0]				PWMS2A[3:0]			
		7:0	S1APOL	S2APOL	PWMLFA[1:0]			PWMLFAD[2:0]		
0x1024	LOGCONB	31:24								
		23:16								
		15:8	PWMS1B[3:0]				PWMS2B[3:0]			
		7:0	S1BPOL	S2BPOL	PWMLFB[1:0]			PWMLFBD[2:0]		
0x1028	LOGCONC	31:24								
		23:16								
		15:8	PWMS1C[3:0]				PWMS2C[3:0]			
		7:0	S1CPOL	S2CPOL	PWMLFC[1:0]			PWMLFCD[2:0]		
0x102C	LOGCOND	31:24								
		23:16								
		15:8	PWMS1D[3:0]				PWMS2D[3:0]			
		7:0	S1DPOL	S2DPOL	PWMLFD[1:0]			PWMLFDD[2:0]		
0x1030	LOGCONE	31:24								
		23:16								
		15:8	PWMS1E[3:0]				PWMS2E[3:0]			
		7:0	S1EPOL	S2EPOL	PWMLFE[1:0]			PWMLFED[2:0]		
0x1034	LOGCONF	31:24								
		23:16								
		15:8	PWMS1F[3:0]				PWMS2F[3:0]			
		7:0	S1FPOL	S2FPOL	PWMLFF[1:0]			PWMLFFD[2:0]		

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1038	PWMEVTA	31:24								
		23:16								
		15:8	EVTAOEN	EVTAPOL	EVTASTRD	EVTASYNC				
		7:0	EVTASEL[3:0]					EVTAPGS[2:0]		
0x103C	PWMEVTB	31:24								
		23:16								
		15:8	EVTBOEN	EVTBPOL	EVTBSTRD	EVTBSYNC				
		7:0	EVTBSEL[3:0]					EVTBPGS[2:0]		
0x1040	PWMEVTC	31:24								
		23:16								
		15:8	EVTCOEN	EVTCPOL	EVTCSTRD	EVTCSYNC				
		7:0	EVTCSEL[3:0]					EVTCPGS[2:0]		
0x1044	PWMEVTD	31:24								
		23:16								
		15:8	EVTDOEN	EVTDPOL	EVTDSTRD	EVTDSYNC				
		7:0	EVTDSEL[3:0]					EVTDPGS[2:0]		
0x1048	PWMEVTE	31:24								
		23:16								
		15:8	EVTEOEN	EVTEPOL	EVTESTRD	EVTESYNC				
		7:0	EVTESEL[3:0]					EVTEPGS[2:0]		
0x104C	PWMEVTF	31:24								
		23:16								
		15:8	EVTFOEN	EVTFPOL	EVTFSTRD	EVTFSYNC				
		7:0	EVTFSEL[3:0]					EVTFPGS[2:0]		

14.3.2.1 PWM Clock Control Register

Name: PCLKCON
Offset: 0x1004

Legend: C = Clearable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
								LOCK
Access								R/W
Reset								0

Bit	7	6	5	4	3	2	1	0
			DIVSEL[1:0]					MCLKSEL
Access			R/W	R/W				R/W
Reset			0	0				0

Bit 8 – LOCK Lock

Value	Description
1	Write-protected registers and bits are locked
0	Write-protected registers and bits are unlocked

Bits 5:4 – DIVSEL[1:0] PWM Clock Divider Selection

Value	Description
11	Divide ratio is 1:16
10	Divide ratio is 1:8
01	Divide ratio is 1:4
00	Divide ratio is 1:2

Bit 0 – MCLKSEL PWM Master Clock Selection

Note: Do not change the MCLKSEL bit while ON (PGxCON[15]) = 1.

See Table 14-2 for MCLKSEL PWM Master Clock Selection.

14.3.2.2 Frequency Scale Register

Name: FSCL
Offset: 0x1004

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					FSCL[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FSCL[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FSCL[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bits 19:16 – FSCL[19:16] Frequency Scale Register
The value in this register is added to the frequency scaling accumulator at each pwm_master_clk. When the accumulated value exceeds the value of FSMINPER, a clock pulse is produced.

Bits 15:8 – FSCL[15:8] Frequency Scale Register
The value in this register is added to the frequency scaling accumulator at each pwm_master_clk. When the accumulated value exceeds the value of FSMINPER, a clock pulse is produced.

Bits 7:4 – FSCL[7:4] Frequency Scale Register
The value in this register is added to the frequency scaling accumulator at each pwm_master_clk. When the accumulated value exceeds the value of FSMINPER, a clock pulse is produced.

14.3.2.3 Frequency Scaling Minimum Period Register

Name: FSMINPER
Offset: 0x1008

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					FSMINPER[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FSMINPER[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FSMINPER[7:4]				Reserved[3:0]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 19:16 – FSMINPER[19:16] Frequency Scaling Minimum Period Register
This register holds the minimum clock period (maximum clock frequency) that can be produced by the frequency scaling circuit.

Bits 15:8 – FSMINPER[15:8] Frequency Scaling Minimum Period Register
This register holds the minimum clock period (maximum clock frequency) that can be produced by the frequency scaling circuit.

Bits 7:4 – FSMINPER[7:4] Frequency Scaling Minimum Period Register
This register holds the minimum clock period (maximum clock frequency) that can be produced by the frequency scaling circuit.

Bits 3:0 – Reserved[3:0]

14.3.2.4 Master Phase Register

Name: MPHASE
Offset: 0x100C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					MPHASE[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MPHASE[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MPHASE[7:4]				Reserved[3:0]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 19:16 – MPHASE[19:16] Master Phase Register
This register holds the phase offset value that can be shared by multiple PWM Generators.

Bits 15:8 – MPHASE[15:8] Master Phase Register
This register holds the phase offset value that can be shared by multiple PWM Generators.

Bits 7:4 – MPHASE[7:4] Master Phase Register
This register holds the phase offset value that can be shared by multiple PWM Generators.

Bits 3:0 – Reserved[3:0]

14.3.2.5 Master Duty Cycle Register

Name: MDC
Offset: 0x1010

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					MDC[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MDC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MDC[7:4]				Reserved[3:0]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 19:16 – MDC[19:16] Master Duty Cycle Register
This register holds the duty cycle value that can be shared by multiple PWM Generators.
Note: Duty cycle values less than 0x0010 should not be used.

Bits 15:8 – MDC[15:8] Master Duty Cycle Register
This register holds the duty cycle value that can be shared by multiple PWM Generators.
Note: Duty cycle values less than 0x0010 should not be used.

Bits 7:4 – MDC[7:4] Master Duty Cycle Register
This register holds the duty cycle value that can be shared by multiple PWM Generators.
Note: Duty cycle values less than 0x0010 should not be used.

Bits 3:0 – Reserved[3:0]

14.3.2.6 Master Period Register

Name: MPER
Offset: 0x1014

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					MPER[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MPER[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MPER[7:4]				Reserved[3:0]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 19:16 – MPER[19:16] Master Period Register
This register holds the period value that can be shared by multiple PWM Generators.
Note: Period values less than 0x0100 should not be used.

Bits 15:8 – MPER[15:8] Master Period Register
This register holds the period value that can be shared by multiple PWM Generators.
Note: Period values less than 0x0100 should not be used.

Bits 7:4 – MPER[7:4] Master Period Register
This register holds the period value that can be shared by multiple PWM Generators.
Note: Period values less than 0x0100 should not be used.

Bits 3:0 – Reserved[3:0]

14.3.2.7 Linear Feedback Shift Register

Name: LFSR
Offset: 0x1018

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
		LFSR[14:8]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LFSR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 14:0 – LFSR[14:0] Linear Feedback Shift Register
A read of this register will provide a 15-bit pseudorandom value.

14.3.2.8 Combinational Trigger Register

Name: CMBTRIG
Offset: 0x101C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	CTB8EN	CTB7EN	CTB6EN	CTB5EN	CTB4EN	CTB3EN	CTB2EN	CTB1EN
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	CTA8EN	CTA7EN	CTA6EN	CTA5EN	CTA4EN	CTA3EN	CTA2EN	CTA1EN
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 23 – CTB8EN Enable Trigger Output from PWM Generator #8 as Source for Combinational Trigger B

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal
0	Disabled

Bit 22 – CTB7EN Enable Trigger Output from PWM Generator #7 as Source for Combinational Trigger B

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal
0	Disabled

Bit 21 – CTB6EN Enable Trigger Output from PWM Generator #6 as Source for Combinational Trigger B

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal
0	Disabled

Bit 20 – CTB5EN Enable Trigger Output from PWM Generator #5 as Source for Combinational Trigger B

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal
0	Disabled

Bit 19 – CTB4EN Enable Trigger Output from PWM Generator #4 as Source for Combinational Trigger B

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal
0	Disabled

Bit 18 – CTB3EN Enable Trigger Output from PWM Generator #3 as Source for Combinational Trigger B

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal
0	Disabled

Bit 17 – CTB2EN Enable Trigger Output from PWM Generator #2 as Source for Combinational Trigger B

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal
0	Disabled

Bit 16 – CTB1EN Enable Trigger Output from PWM Generator #1 as Source for Combinational Trigger B

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal
0	Disabled

Bit 7 – CTA8EN Enable Trigger Output from PWM Generator #8 as Source for Combinational Trigger A

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal
0	Disabled

Bit 6 – CTA7EN Enable Trigger Output from PWM Generator #7 as Source for Combinational Trigger A

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal
0	Disabled

Bit 5 – CTA6EN Enable Trigger Output from PWM Generator #6 as Source for Combinational Trigger A

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal
0	Disabled

Bit 4 – CTA5EN Enable Trigger Output from PWM Generator #5 as Source for Combinational Trigger A

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal
0	Disabled

Bit 3 – CTA4EN Enable Trigger Output from PWM Generator #4 as Source for Combinational Trigger A

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal
0	Disabled

Bit 2 – CTA3EN Enable Trigger Output from PWM Generator #3 as Source for Combinational Trigger A

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal
0	Disabled

Bit 1 – CTA2EN Enable Trigger Output from PWM Generator #2 as Source for Combinational Trigger A

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal
0	Disabled

Bit 0 – CTA1EN Enable Trigger Output from PWM Generator #1 as Source for Combinational Trigger A

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal
0	Disabled

14.3.2.9 Combinatorial PWM Logic Control Register y

Name: LOGCONy
Offset: 0x1020, 0x1024, 0x1028, 0x102C, 0x1030, 0x1034

Note: 'y' denotes a common instance (A-F); the number of the available combinatorial PWM logic is device-dependent.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PWMS1y[3:0]				PWMS2y[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	S1yPOL	S2yPOL	PWMLFy[1:0]			PWMLFyD[2:0]		
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bits 15:12 – PWMS1y[3:0] Combinatorial PWM Logic Source #1 Selection
See Table 14-4 for device-specific selections.
Note: Logic function input will be connected to '0' if the PWM channel is not present.

Bits 11:8 – PWMS2y[3:0] Combinatorial PWM Logic Source #2 Selection
See Table 14-4 for device-specific selections.
Note: Logic function input will be connected to '0' if the PWM channel is not present.

Bit 7 – S1yPOL Combinatorial PWM Logic Source #1 Polarity

Value	Description
1	Input is inverted
0	Input is positive logic

Bit 6 – S2yPOL Combinatorial PWM Logic Source #2 Polarity

Value	Description
1	Input is inverted
0	Input is positive logic

Bits 5:4 – PWMLFy[1:0] Combinatorial PWM Logic Function Selection

Value	Description
11	Reserved
10	PWMS1y ^ PWMS2y (XOR)
01	PWMS1y & PWMS2y (AND)
00	PWMS1y | PWMS2y (OR)

Bits 2:0 – PWMLFyD[2:0] Combinatorial PWM Logic Destination Selection
See Table 14-5 for device-specific selections.
Note: Instances of y = A, C, E of LOGCONy assign logic function output to the PWMxH pin. Instances of y = B, D, F of LOGCONy assign logic function to the PWMxL pin.

14.3.2.10 PWM Event Output Control Register y

Name: PWMEVTy
Offset: 0x1038, 0x103C, 0x1040, 0x1044, 0x1048, 0x104C

Note: 'y' denotes a common instance (A-F); the number of the available combinatorial PWM logic is device-dependent.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	EVTyOEN	EVTyPOL	EVTySTRD	EVTySYNC				
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bit	7	6	5	4	3	2	1	0
	EVTySEL[3:0]					EVTyPGS[2:0]		
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bit 15 – EVTyOEN PWM Event Output Enable

Value	Description
1	Event output signal is output on the PWMEy pin
0	Event output signal is internal only

Bit 14 – EVTyPOL PWM Event Output Polarity

Value	Description
1	Event output signal is active-low
0	Event output signal is active-high

Bit 13 – EVTySTRD PWM Event Output Stretch Disable

Note: The event signal is stretched using peripheral_clk because different PWM Generators may be operating from different clock sources.

Value	Description
1	Event output signal pulse width is not stretched
0	Event output signal is stretched to eight PWM clock cycles minimum

Bit 12 – EVTySYNC PWM Event Output Sync

Event output signal pulse will be synchronized to peripheral_clk.

Value	Description
1	Event output signal is synchronized to the system clock
0	Event output is not synchronized to the system clock

Bits 7:4 – EVTySEL[3:0] PWM Event Selection

Note: This is the PWM Generator output signal prior to Output mode logic and any output override logic.

Value	Description
1111-1010	Reserved
1001	ADC Trigger 2 signal
1000	ADC Trigger 1 signal
0111	STEER signal (available in Push-Pull Output modes only)
0110	CAHALF signal (available in Center-Aligned modes only)
0101	PCI Fault active output signal
0100	PCI current limit active output signal
0011	PCI feed-forward active output signal
0010	PCI Sync active output signal
0001	PWM Generator output signal(1)
0000	Source is selected by the PGTRGSEL[2:0] bits

Bits 2:0 – EVTyPGS[2:0] PWM Event Source Selection

See Table 14-6 for device-specific selections.

Note: No event will be produced if the selected PWM Generator is not present.

14.3.3 PWM Generator Register Map

Legend: x = PWM Generator #; y = F, CL, FF or S.

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1050	PG1CON	31:24	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
		23:16	TRGMOD[1:0]				SOCS[3:0]			
		15:8	ON					TRGCNT[2:0]		
		7:0	Reserved			CLKSEL[1:0]		MODSEL[2:0]		
0x1054	PG1STAT	31:24								
		23:16								
		15:8	SEVT	FLTEVT	CLEVT	FFEVT	SACT	FLTACT	CLACT	FFACT
		7:0	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
0x1058	PG1IOCON	31:24		CAPSRC[2:0]					PPSEN	DTCMPSEL
		23:16			PMOD[1:0]		PENH	PENL	POLH	POLL
		15:8	CLMOD	SWAP	OVRENH	OVRENL	OVRDAT[1:0]		OSYNC[1:0]	
		7:0	FLTDAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBDAT[1:0]	
0x105C	PG1EVT	31:24	FLTIEN	CLIEN	FFIEN	SIEN			IEVTSEL[1:0]	
		23:16	ADTR2EN3	ADTR2EN2	ADTR2EN1	ADTR1OFS[4:0]				
		15:8	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
		7:0	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
0x1060	PG1FPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x1064	PG1CLPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x1068	PG1FFPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x106C	PG1SPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x1070	PG1LEB	31:24								
		23:16					PHR	PHF	PLR	PLF
		15:8	LEB[15:8]							
		7:0	LEB[7:4]							
0x1074	PG1PHASE	31:24								
		23:16					PHASE[19:16]			
		15:8	PHASE[15:8]							
		7:0	PHASE[7:4]				Reserved[3:0]			
0x1078	PG1DC	31:24								
		23:16					DC[19:16]			
		15:8	DC[15:8]							
		7:0	DC[7:4]				Reserved[3:0]			
0x107C	PG1DCA	31:24								
		23:16								
		15:8					DCA[11:8]			
		7:0	DCA[7:4]				Reserved[3:0]			
0x1080	PG1PER	31:24								
		23:16					PER[19:16]			
		15:8	PER[15:8]							
		7:0	PER[7:4]				Reserved[3:0]			
0x1084	PG1TRIGA	31:24	CAHALF							
		23:16					TRIGA[19:16]			
		15:8	TRIGA[15:8]							
		7:0	TRIGA[7:4]				Reserved[3:0]			

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1088	PG1TRIGB	31:24	CAHALF							
		23:16					TRIGB[19:16]			
		15:8	TRIGB[15:8]							
		7:0	TRIGB[7:4]				Reserved[3:0]			
0x108C	PG1TRIGC	31:24	CAHALF							
		23:16					TRIGC[19:16]			
		15:8	TRIGC[15:8]							
		7:0	TRIGC[7:4]				Reserved[3:0]			
0x1090	PG1DT	31:24		DTH[14:8]						
		23:16	DTH[7:0]							
		15:8		DTL[14:8]						
		7:0	DTL[7:0]							
0x1094	PG1CAP	31:24								
		23:16					CAP[19:16]			
		15:8	CAP[15:8]							
		7:0	CAP[7:4]							
0x1098	PG2CON	31:24	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
		23:16	TRGMOD[1:0]				SOCS[3:0]			
		15:8	ON					TRGCNT[2:0]		
		7:0	Reserved			CLKSEL[1:0]		MODSEL[2:0]		
0x109C	PG2STAT	31:24								
		23:16								
		15:8	SEVT	FLTEVT	CLEVT	FFEVT	SACT	FLTACT	CLACT	FFACT
		7:0	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
0x10A0	PG2IOCON	31:24		CAPSRC[2:0]					PPSEN	DTCMPSEL
		23:16			PMOD[1:0]		PENH	PENL	POLH	POLL
		15:8	CLMOD	SWAP	OVRENH	OVRENL	OVRDAT[1:0]		OSYNC[1:0]	
		7:0	FLTDAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBDAT[1:0]	
0x10A4	PG2EVT	31:24	FLTIEN	CLIEN	FFIEN	SIEN			IEVTSEL[1:0]	
		23:16	ADTR2EN3	ADTR2EN2	ADTR2EN1	ADTR1OFS[4:0]				
		15:8	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
		7:0	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
0x10A8	PG2FPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x10AC	PG2CLPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x10B0	PG2FFPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x10B4	PG2SPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x10B8	PG2LEB	31:24								
		23:16					PHR	PHF	PLR	PLF
		15:8	LEB[15:8]							
		7:0	LEB[7:4]							
0x10BC	PG2PHASE	31:24								
		23:16					PHASE[19:16]			
		15:8	PHASE[15:8]							
		7:0	PHASE[7:4]				Reserved[3:0]			
0x10C0	PG2DC	31:24								
		23:16					DC[19:16]			
		15:8	DC[15:8]							
		7:0	DC[7:4]				Reserved[3:0]			

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x10C4	PG2DCA	31:24								
		23:16								
		15:8					DCA[11:8]			
		7:0	DCA[7:4]				Reserved[3:0]			
0x10C8	PG2PER	31:24								
		23:16					PER[19:16]			
		15:8	PER[15:8]							
		7:0	PER[7:4]				Reserved[3:0]			
0x10CC	PG2TRIGA	31:24	CAHALF							
		23:16					TRIGA[19:16]			
		15:8	TRIGA[15:8]							
		7:0	TRIGA[7:4]				Reserved[3:0]			
0x10D0	PG2TRIGB	31:24	CAHALF							
		23:16					TRIGB[19:16]			
		15:8	TRIGB[15:8]							
		7:0	TRIGB[7:4]				Reserved[3:0]			
0x10D4	PG2TRIGC	31:24	CAHALF							
		23:16					TRIGC[19:16]			
		15:8	TRIGC[15:8]							
		7:0	TRIGC[7:4]				Reserved[3:0]			
0x10D8	PG2DT	31:24		DTH[14:8]						
		23:16	DTH[7:0]							
		15:8		DTL[14:8]						
		7:0	DTL[7:0]							
0x10DC	PG2CAP	31:24								
		23:16					CAP[19:16]			
		15:8	CAP[15:8]							
		7:0	CAP[7:4]							
0x10E0	PG3CON	31:24	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
		23:16	TRGMOD[1:0]				SOCS[3:0]			
		15:8	ON					TRGCNT[2:0]		
		7:0	Reserved			CLKSEL[1:0]		MODSEL[2:0]		
0x10E4	PG3STAT	31:24								
		23:16								
		15:8	SEVT	FLTEVT	CLEVT	FFEVT	SACT	FLTACT	CLACT	FFACT
		7:0	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
0x10E8	PG3IOCON	31:24		CAPSRC[2:0]					PPSEN	DTCMPSEL
		23:16			PMOD[1:0]		PENH	PENL	POLH	POLL
		15:8	CLMOD	SWAP	OVRENH	OVRENL	OVRDAT[1:0]		OSYNC[1:0]	
		7:0	FLTDAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBDAT[1:0]	
0x10EC	PG3EVT	31:24	FLTIEN	CLIEN	FFIEN	SIEN			IEVTSEL[1:0]	
		23:16	ADTR2EN3	ADTR2EN2	ADTR2EN1	ADTR1OFS[4:0]				
		15:8	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
		7:0	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
0x10F0	PG3FPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS			PSS[4:0]		
0x10F4	PG3CLPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS			PSS[4:0]		
0x10F8	PG3FFPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS			PSS[4:0]		
0x10FC	PG3SPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS			PSS[4:0]		

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1100	PG3LEB	31:24								
		23:16					PHR	PHF	PLR	PLF
		15:8	LEB[15:8]							
		7:0	LEB[7:4]							
0x1104	PG3PHASE	31:24								
		23:16					PHASE[19:16]			
		15:8	PHASE[15:8]							
		7:0	PHASE[7:4]				Reserved[3:0]			
0x1108	PG3DC	31:24								
		23:16					DC[19:16]			
		15:8	DC[15:8]							
		7:0	DC[7:4]				Reserved[3:0]			
0x110C	PG3DCA	31:24								
		23:16								
		15:8					DCA[11:8]			
		7:0	DCA[7:4]				Reserved[3:0]			
0x1110	PG3PER	31:24								
		23:16					PER[19:16]			
		15:8	PER[15:8]							
		7:0	PER[7:4]				Reserved[3:0]			
0x1114	PG3TRIGA	31:24	CAHALF							
		23:16					TRIGA[19:16]			
		15:8	TRIGA[15:8]							
		7:0	TRIGA[7:4]				Reserved[3:0]			
0x1118	PG3TRIGB	31:24	CAHALF							
		23:16					TRIGB[19:16]			
		15:8	TRIGB[15:8]							
		7:0	TRIGB[7:4]				Reserved[3:0]			
0x111C	PG3TRIGC	31:24	CAHALF							
		23:16					TRIGC[19:16]			
		15:8	TRIGC[15:8]							
		7:0	TRIGC[7:4]				Reserved[3:0]			
0x1120	PG3DT	31:24		DTH[14:8]						
		23:16	DTH[7:0]							
		15:8		DTL[14:8]						
		7:0	DTL[7:0]							
0x1124	PG3CAP	31:24								
		23:16					CAP[19:16]			
		15:8	CAP[15:8]							
		7:0	CAP[7:4]							
0x1128	PG4CON	31:24	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
		23:16	TRGMOD[1:0]				SOCS[3:0]			
		15:8	ON					TRGCNT[2:0]		
		7:0	Reserved			CLKSEL[1:0]		MODSEL[2:0]		
0x112C	PG4STAT	31:24								
		23:16								
		15:8	SEVT	FLTEVT	CLEVT	FFEVT	SACT	FLTACT	CLACT	FFACT
		7:0	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
0x1130	PG4IOCON	31:24		CAPSRC[2:0]					PPSEN	DTCMPSEL
		23:16			PMOD[1:0]		PENH	PENL	POLH	POLL
		15:8	CLMOD	SWAP	OVRENH	OVRENL	OVRDAT[1:0]		OSYNC[1:0]	
		7:0	FLTDAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBDAT[1:0]	
0x1134	PG4EVT	31:24	FLTIEN	CLIEN	FFIEN	SIEN			IEVTSEL[1:0]	
		23:16	ADTR2EN3	ADTR2EN2	ADTR2EN1	ADTR1OFS[4:0]				
		15:8	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
		7:0	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
0x1138	PG4FPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x113C	PG4CLPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x1140	PG4FFPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x1144	PG4SPCI	31:24	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	SWTERM	PSYNC	PPS	PSS[4:0]				
0x1148	PG4LEB	31:24								
		23:16					PHR	PHF	PLR	PLF
		15:8	LEB[15:8]							
		7:0	LEB[7:4]							
0x114C	PG4PHASE	31:24								
		23:16					PHASE[19:16]			
		15:8	PHASE[15:8]							
		7:0	PHASE[7:4]				Reserved[3:0]			
0x1150	PG4DC	31:24								
		23:16					DC[19:16]			
		15:8	DC[15:8]							
		7:0	DC[7:4]				Reserved[3:0]			
0x1154	PG4DCA	31:24								
		23:16								
		15:8					DCA[11:8]			
		7:0	DCA[7:4]				Reserved[3:0]			
0x1158	PG4PER	31:24								
		23:16					PER[19:16]			
		15:8	PER[15:8]							
		7:0	PER[7:4]				Reserved[3:0]			
0x115C	PG4TRIGA	31:24	CAHALF							
		23:16					TRIGA[19:16]			
		15:8	TRIGA[15:8]							
		7:0	TRIGA[7:4]				Reserved[3:0]			
0x1160	PG4TRIGB	31:24	CAHALF							
		23:16					TRIGB[19:16]			
		15:8	TRIGB[15:8]							
		7:0	TRIGB[7:4]				Reserved[3:0]			
0x1164	PG4TRIGC	31:24	CAHALF							
		23:16					TRIGC[19:16]			
		15:8	TRIGC[15:8]							
		7:0	TRIGC[7:4]				Reserved[3:0]			
0x1168	PG4DT	31:24		DTH[14:8]						
		23:16	DTH[7:0]							
		15:8		DTL[14:8]						
		7:0	DTL[7:0]							
0x116C	PG4CAP	31:24								
		23:16					CAP[19:16]			
		15:8	CAP[15:8]							
		7:0	CAP[7:4]							

14.3.3.1 PWM Generator x Control Register

Name: PGxCON
Offset: 0x1050, 0x1098, 0x10E0, 0x1128

Notes:

1. This bit(s) cannot be modified while PGxCON.ON = 1.
2. This bit(s) cannot be modified while PCLKCON.LOCK = 1. Otherwise, caution should be exercised when modifying this bit when PGxCON.ON = 1; unexpected results may occur.
3. This bit(s) cannot be modified while UPDATE = 1.
4. Caution should be exercised when modifying this bit(s) while PGxCON.ON = 1; unexpected results may occur.

Bit	31	30	29	28	27	26	25	24
	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
Access	R/W	R/W	R/W		R/W	R/W	R/W	R/W
Reset	0	0	0		0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRGMOD[1:0]				SOCS[3:0]			
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON					TRGCNT[2:0]		
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0

Bit	7	6	5	4	3	2	1	0
	Reserved			CLKSEL[1:0]		MODSEL[2:0]		
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit 31 – MDCSEL Master Duty Cycle Register Select(4)

Value	Description
1	PWM Generator uses MDC register
0	PWM Generator uses PGxDC register

Bit 30 – MPERSEL Master Period Register Select(4)

Value	Description
1	PWM Generator uses MPER register
0	PWM Generator uses PGxPER register

Bit 29 – MPHSEL Master Phase Register Select(4)

Value	Description
1	PWM Generator uses MPHASE register
0	PWM Generator uses PGxPHASE register

Bit 27 – MSTEN Master Update Enable(4)

Value	Description
1	PWM Generator broadcasts software set of UPDREQ control bit and EOC signal to other PWM Generators

Value	Description
0	PWM Generator does not broadcast UPDREQ status bit state or EOC signal

Bits 26:24 – UPDMOD[2:0] PWM Buffer Update Mode Selection
See Table 14-10 for details.

Bits 23:22 – TRGMOD[1:0] PWM Generator x Trigger Mode Selection(4)

Value	Description
11	Reserved
10	Reserved
01	PWM Generator operates in Retriggerable mode
00	PWM Generator operates in Single Trigger mode

Bits 19:16 – SOCS[3:0] Start-of-Cycle Selection bits(4)

Notes:

1. The PCI selected Sync signal is always available to be OR'd with the selected SOC signal per the SOCS[3:0] bits if the PCI Sync function is enabled.
2. The source selected by the SOCS[3:0] bits MUST operate from the same clock source as the local PWM Generator. If not, the source must be routed through the PCI Sync logic so the trigger signal may be synchronized to the PWM Generator clock domain.
3. PWM Generators are grouped into groups of four: PG1-PG4 and PG5-PG8, if available. Any generator within a group of four may be used to trigger another generator within the same group.

Value	Description
1111	TRIG bit or PCI Sync function only (no hardware trigger source is selected)
1110 – 0101	Reserved
0100	Trigger output selected by PG4 or PG8 PGTRGSEL[2:0] bits (PGxEVT[2:0])
0011	Trigger output selected by PG3 or PG7 PGTRGSEL[2:0] bits (PGxEVT[2:0])
0010	Trigger output selected by PG2 or PG6 PGTRGSEL[2:0] bits (PGxEVT[2:0])
0001	Trigger output selected by PG1 or PG5 PGTRGSEL[2:0] bits (PGxEVT[2:0])
0000	Local EOC – PWM Generator is self-triggered

Bit 15 – ON PWM Generator x Enable(1)

Value	Description
1	PWM Generator is enabled
0	PWM Generator is not enabled

Bits 10:8 – TRGCNT[2:0] PWM Generator x Trigger Count Select(1)

Value	Description
111	PWM Generator produces 8 PWM cycles after triggered
110	PWM Generator produces 7 PWM cycles after triggered
101	PWM Generator produces 6 PWM cycles after triggered
100	PWM Generator produces 5 PWM cycles after triggered
011	PWM Generator produces 4 PWM cycles after triggered
010	PWM Generator produces 3 PWM cycles after triggered
001	PWM Generator produces 2 PWM cycles after triggered
000	PWM Generator produces 1 PWM cycle after triggered

Bit 7 – Reserved

Bits 4:3 – CLKSEL[1:0] Clock Selection(1)

Notes:

1. Do not change the CLKSEL[1:0] bits while ON (PGxCON[15]) = 1.
2. The PWM Generator time base operates from the frequency scaling circuit clock, effectively scaling the duty cycle and period of the PWM Generator output.

Value	Description
11	PWM Generator uses the master clock scaled by the frequency scaling circuit(2)
10	PWM Generator uses the master clock divided by the clock divider circuit(2)
01	PWM Generator uses the master clock selected by the MCLKSEL (PCLKCON[0]) control bits
00	No clock selected, PWM Generator is in the lowest power state (default)

Bits 2:0 – MODSEL[2:0] PWM Generator x Mode Selection(1)

Value	Description
111	Dual Edge Center-Aligned PWM mode (interrupt/register update twice per cycle)
110	Dual Edge Center-Aligned PWM mode (interrupt/register update once per cycle)
101	Double Update Center-Aligned PWM mode
100	Center-Aligned PWM mode
011	Reserved
010	Independent Edge PWM mode, dual output
001	Variable Phase PWM mode
000	Independent Edge PWM mode

14.3.3.2 PWM Generator x Status Register

Name: PGxSTAT
Offset: 0x1054, 0x109C, 0x10E4, 0x112C

Legend: C = Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	SEVT	FLTEVT	CLEVT	FFEVT	SACT	FLTACT	CLACT	FFACT
Access	HS/C	HS/C	HS/C	HS/C	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
Access	W	W	R/HS	R	W	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 15 – SEVT PCI Sync Event

Value	Description
1	A PCI Sync event has occurred (rising edge on PCI Sync output or PCI Sync output is high when module is enabled)
0	No PCI Sync event has occurred

Bit 14 – FLTEVT PCI Fault Active Status

Value	Description
1	A Fault event has occurred (rising edge on PCI Fault output or PCI Fault output is high when module is enabled)
0	No Fault event has occurred

Bit 13 – CLEVT PCI Current Limit Status

Value	Description
1	A PCI current limit event has occurred (rising edge on PCI current limit output or PCI current limit output is high when module is enabled)
0	No PCI current limit event has occurred

Bit 12 – FFEVT PCI Feed-Forward Active Status

Value	Description
1	A PCI feed-forward event has occurred (the rising edge on the PCI feed-forward output or PCI feed-forward output is high when module is enabled)
0	No PCI feed-forward event has occurred

Bit 11 – SACT PCI Sync Status

Value	Description
1	PCI Sync output is active

Value	Description
0	PCI Sync output is inactive

Bit 10 – FLTACT PCI Fault Active Status

Value	Description
1	PCI Fault output is active
0	PCI Fault output is inactive

Bit 9 – CLACT PCI Current Limit Status

Value	Description
1	PCI current limit output is active
0	PCI current limit output is inactive

Bit 8 – FFACT PCI Feed-Forward Active Status

Value	Description
1	PCI feed-forward output is active
0	PCI feed-forward output is inactive

Bit 7 – TRSET PWM Generator Software Trigger Set

User software writes a '1' to this bit location to trigger a PWM Generator cycle. The bit location always reads as '0'. The TRIG bit will indicate '1' when the PWM Generator is triggered.

Bit 6 – TRCLR PWM Generator Software Trigger Clear

User software writes a '1' to this bit location to stop a PWM Generator cycle. The bit location always reads as '0'. The TRIG bit will indicate '0' when the PWM Generator is not triggered.

Bit 5 – CAP Capture Status

Value	Description
1	PWM Generator time base value has been captured in PGxCAP
0	No capture has occurred

Bit 4 – UPDATE PWM Data Register Update Status/Control

Value	Description
1	PWM Data register update is pending – user Data registers are not writable
0	No PWM Data register update is pending

Bit 3 – UPDREQ PWM Data Register Update Request

User software writes a '1' to this bit location to request a PWM Data register update. The bit location always reads as '0'. The UPDATE status bit will indicate a '1' when an update is pending.

Bit 2 – STEER Output Steering Status (Push-Pull Output mode only)

Value	Description
1	PWM Generator is in 2nd cycle of Push-Pull mode
0	PWM Generator is in 1st cycle of Push-Pull mode

Bit 1 – CAHALF Half Cycle Status (Center-Aligned modes only)

Value	Description
1	PWM Generator is in 2nd half of time base cycle
0	PWM Generator is in 1st half of time base cycle

Bit 0 – TRIG Trigger Status

Value	Description
1	PWM Generator is triggered and PWM cycle is in progress
0	No PWM cycle is in progress

14.3.3.3 PWM Generator x I/O Control Register

Name: PGxIOCON
Offset: 0x1058, 0x10A0, 0x10E8, 0x1130

Notes:

1. This bit(s) cannot be modified while PGxCON.ON = 1.
2. This bit(s) cannot be modified while PCLKCON.LOCK = 1. Otherwise, caution should be exercised when modifying this bit when PGxCON.ON = 1; unexpected results may occur.
3. This bit(s) cannot be modified while UPDATE = 1.
4. Caution should be exercised when modifying this bit(s) while PGxCON.ON = 1; unexpected results may occur.

Bit	31	30	29	28	27	26	25	24
		CAPSRC[2:0]					PPSEN	DTCMPSEL
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit	23	22	21	20	19	18	17	16
			PMOD[1:0]		PENH	PENL	POLH	POLL
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CLMOD	SWAP	OVRENH	OVRENL	OVRDAT[1:0]		OSYNC[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FLTDAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBDAT[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 30:28 – CAPSRC[2:0] Time Base Capture Source Selection

Note: A capture may be initiated in software at any time by writing a '1' to PGxCAP[0].

Value	Description
111	Reserved
110	Reserved
101	Reserved
100	Capture time base value at assertion of selected PCI Fault signal
011	Capture time base value at assertion of selected PCI Current Limit signal
010	Capture time base value at assertion of selected PCI Feed-Forward signal
001	Capture time base value at assertion of selected PCI Sync signal
000	No hardware source selected for time base capture – software only

Bit 25 – PPSEN Peripheral Pin Select Enable bit(4)

Value	Description
1	Peripheral pin select enabled
0	Peripheral pin select disabled, as a result, PWM outputs are hard-mapped to pins

Bit 24 – DTCMPSEL Dead-Time Compensation Select(4)

Value	Description
1	Dead-time compensation is controlled by PCI feed-forward limit logic
0	Dead-time compensation is controlled by PCI Sync logic

Bits 21:20 – PMOD[1:0] PWM Generator Output Mode Selection(2)

Value	Description
11	Reserved
10	PWM Generator outputs operate in Push-Pull mode
01	PWM Generator outputs operate in Independent mode
00	PWM Generator outputs operate in Complementary mode

Bit 19 – PENH PWMxH Output Port Enable(2)

Value	Description
1	PWM Generator controls the PWMxH output pin
0	PWM Generator does not control the PWMxH output pin

Bit 18 – PENL PWMxL Output Port Enable(2)

Value	Description
1	PWM Generator controls the PWMxL output pin
0	PWM Generator does not control the PWMxL output pin

Bit 17 – POLH PWMxH Output Polarity(2)

Value	Description
1	Output pin is active-low
0	Output pin is active-high

Bit 16 – POLL PWMxL Output Polarity(2)

Value	Description
1	Output pin is active-low
0	Output pin is active-high

Bit 15 – CLMOD Current Limit Mode Select

Value	Description
1	If PCI current limit is active, then the PWMxH and PWMxL output signals are inverted (bit flipping), and the CLDAT[1:0] bits are not used
0	If PCI current limit is active, then the CLDAT[1:0] bits define the PWM output levels

Bit 14 – SWAP Swap PWM Signals to PWMxH and PWMxL Device Pins

Value	Description
1	The PWMxH signal is connected to the PWMxL pin and the PWMxL signal is connected to the PWMxH pin
0	PWMxH/L signals are mapped to their respective pins

Bit 13 – OVRENH User Override Enable for PWMxH Pin

Value	Description
1	OVRDAT[1] provides data for output on the PWMxH pin
0	PWM Generator provides data for the PWMxH pin

Bit 12 – OVRENL User Override Enable for PWMxL Pin

Value	Description
1	OVRDAT[0] provides data for output on the PWMxL pin
0	PWM Generator provides data for the PWMxL pin

Bits 11:10 – OVRDAT[1:0] Data for PWMxH/PWMxL Pins if Override is Enabled

Description
If OVRENH = 1, then OVRDAT[1] provides data for PWMxH.
If OVRENL = 1, then OVRDAT[0] provides data for PWMxL.

Bits 9:8 – OSYNC[1:0] User Output Override Synchronization Control

Value	Description
11	User output overrides via the SWAP, OVRENL/H, and OVRDAT[1:0] bits are synchronized to the data buffer update of the selected PWM mode. This makes this setting equivalent to setting 10 when UPDMOD[2:0] = 000 with UPDREQ properly set manually
10	User output overrides via the SWAP, OVRENL/H and OVRDAT[1:0] bits occur when specified by the UPMOD[2:0] bits in the PGxCON register
01	User output overrides via the SWAP, OVRENL/H and OVRDAT[1:0] bits occur immediately (as soon as possible)
00	User output overrides via the SWAP, OVRENL/H and OVRDAT[1:0] bits are synchronized to the local PWM time base (next start of cycle)

Bits 7:6 – FLTDAT[1:0] Data for PWMxH/PWMxL Pins if FLT Event is Active

Description
If Fault is active, then FLTDAT[1] provides data for PWMxH.
If Fault is active, then FLTDAT[0] provides data for PWMxL.

Bits 5:4 – CLDAT[1:0] Data for PWMxH/PWMxL Pins if CLMT Event is Active

Description
If current limit is active, then CLDAT[1] provides data for PWMxH.
If current limit is active, then CLDAT[0] provides data for PWMxL.

Bits 3:2 – FFDAT[1:0] Data for PWMxH/PWMxL Pins if Feed-Forward Event is Active

Description
If feed-forward is active, then FFDAT[1] provides data for PWMxH.
If feed-forward is active, then FFDAT[0] provides data for PWMxL.

Bits 1:0 – DBDAT[1:0] Data for PWMxH/PWMxL Pins if Debug Mode is Active

Description
If Debug mode is active and PTFRZ = 1, then DBDAT[1] provides data for PWMxH.
If Debug mode is active and PTFRZ = 1, then DBDAT[0] provides data for PWMxL.

14.3.3.4 PWM Generator x Event Register

Name: PGxEVT
Offset: 0x105C, 0x10A4, 0x10EC, 0x1134

Notes:

1. Caution should be exercised when modifying this bit(s) while PGxCON.ON = 1; unexpected results may occur.
2. This source can optionally be used as a PCI input, PCI qualifier, PCI terminator or PCI terminator qualifier.

Bit	31	30	29	28	27	26	25	24
	FLTIEN	CLIEN	FFIEN	SIEN			IEVTSEL[1:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	23	22	21	20	19	18	17	16
	ADTR2EN3	ADTR2EN2	ADTR2EN1	ADTR1OFS[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – FLTIEN PCI Fault Interrupt Enable
Note: An interrupt is only generated on the rising edge of the PCI Fault active signal.

Value	Description
1	Fault interrupt is enabled
0	Fault interrupt is disabled

Bit 30 – CLIEN PCI Current Limit Interrupt Enable
Note: An interrupt is only generated on the rising edge of the PCI current limit active signal.

Value	Description
1	Current limit interrupt is enabled
0	Current limit interrupt is disabled

Bit 29 – FFIEN PCI Feed-Forward Interrupt Enable
Note: An interrupt is only generated on the rising edge of the PCI feed-forward active signal.

Value	Description
1	Feed-forward interrupt is enabled
0	Feed-forward interrupt is disabled

Bit 28 – SIEN PCI Sync Interrupt Enable
Note: An interrupt is only generated on the rising edge of the PCI Sync active signal.

Value	Description
1	Sync interrupt is enabled
0	Sync interrupt is disabled

Bits 25:24 – IEVTSEL[1:0] Interrupt Event Selection(1)

Value	Description
11	Time base interrupts are disabled (Sync, Fault, current limit and feed-forward events can be independently enabled)
10	Interrupts CPU at ADC Trigger 1 event
01	Interrupts CPU at TRIGA compare event
00	Interrupts CPU at EOC

Bit 23 – ADTR2EN3 ADC Trigger 2 Source is PGxTRIGC Compare Event Enable(1)

Value	Description
1	PGxTRIGC register compare event is enabled as trigger source for ADC Trigger 2
0	PGxTRIGC register compare event is disabled as trigger source for ADC Trigger 2

Bit 22 – ADTR2EN2 ADC Trigger 2 Source is PGxTRIGB Compare Event Enable(1)

Value	Description
1	PGxTRIGB register compare event is enabled as trigger source for ADC Trigger 2
0	PGxTRIGB register compare event is disabled as trigger source for ADC Trigger 2

Bit 21 – ADTR2EN1 ADC Trigger 2 Source is PGxTRIGA Compare Event Enable(1)

Value	Description
1	PGxTRIGA register compare event is enabled as trigger source for ADC Trigger 2
0	PGxTRIGA register compare event is disabled as trigger source for ADC Trigger 2

Bits 20:16 – ADTR1OFS[4:0] ADC Trigger 1 Offset Selection(1)

Value	Description
11111	Offset by 31 trigger events
. . .	. . .
00010	Offset by 2 trigger events
00001	Offset by 1 trigger event
00000	No offset

Bits 15:11 – ADTR1PS[4:0] ADC Trigger 1 Postscaler Selection(1)

Value	Description
11111	1:32
. . .	. . .
00010	1:3
00001	1:2
00000	1:1

Bit 10 – ADTR1EN3 ADC Trigger 1 Source is PGxTRIGC Compare Event Enable(1)

Value	Description
1	PGxTRIGC register compare event is enabled as trigger source for ADC Trigger 1
0	PGxTRIGC register compare event is disabled as trigger source for ADC Trigger 1

Bit 9 – ADTR1EN2 ADC Trigger 1 Source is PGxTRIGB Compare Event Enable(1)

Value	Description
1	PGxTRIGB register compare event is enabled as trigger source for ADC Trigger 1

Value	Description
0	PGxTRIGB register compare event is disabled as trigger source for ADC Trigger 1

Bit 8 – ADTR1EN1 ADC Trigger 1 Source is PGxTRIGA Compare Event Enable(1)

Value	Description
1	PGxTRIGA register compare event is enabled as trigger source for ADC Trigger 1
0	PGxTRIGA register compare event is disabled as trigger source for ADC Trigger 1

Bits 7:5 – PWMPCI[2:0] PWM PCI Source Selection(2)
See Table 14-6 for device-specific PWM output selection for PCI signal.

Value	Description
111	PWM Generator #8 output used as PCI signal
110	PWM Generator #7 output used as PCI signal
101	PWM Generator #6 output used as PCI signal
100	PWM Generator #5 output used as PCI signal
011	PWM Generator #4 output used as PCI signal
010	PWM Generator #3 output used as PCI signal
001	PWM Generator #2 output used as PCI signal
000	PWM Generator #1 output used as PCI signal

Bits 4:3 – UPDTRG[1:0] Update Trigger Select(1)

Value	Description
11	A write of the PGxTRIGA register automatically sets the UPDREQ bit
10	A write of the PGxPHASE register automatically sets the UPDREQ bit
01	A write of the PGxDC register automatically sets the UPDREQ bit
00	User must set the UPDREQ bit (PGxSTAT[3]) manually

Bits 2:0 – PGTRGSEL[2:0] PWM Generator Trigger Output Selection(1)
Note: These events are derived from the internal PWM Generator time base comparison events.

Value	Description
111	Reserved
110	Reserved
101	Reserved
100	Reserved
011	PGxTRIGC compare event is the PWM Generator trigger
010	PGxTRIGB compare event is the PWM Generator trigger
001	PGxTRIGA compare event is the PWM Generator trigger
000	EOC event is the PWM Generator trigger

14.3.3.5 PWM Generator Fault PCI(1)

Name: PGxFPCI
Offset: 0x1060, 0x10A8, 0x10F0, 0x1138

Notes:

1. Caution should be exercised when modifying this register while PGxCON.ON = 1; unexpected results may occur.
2. This bit has no effect when the SWTERM control bit is used as the PCI Termination Event or if TERM[2:0] < '101'.

Bit	31	30	29	28	27	26	25	24
	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SWTERM	PSYNC	PPS	PSS[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – BPEN PCI Bypass Enable

Value	Description
1	PCI function is enabled and local PCI logic is bypassed; PWM Generator will be controlled by PCI function in the PWM Generator selected by the BPSEL[2:0] bits
0	PCI function is not bypassed

Bits 30:28 – BPSEL[2:0] PCI Bypass Source Selection
See Table 14-6 for device-specific PCI Bypass source options.
Note: Selects '0' if the selected PWM Generator is not present.

Bit 27 – TERMPS PCI Termination Polarity Select bit
Note: This bit has no effect when the SWTERM control bit is used as the PCI Termination Event or if TERM[2:0] < '101'.

Value	Description
1	Inverted
0	Not inverted

Bits 26:24 – ACP[2:0] PCI Acceptance Criteria Selection
Note: Do not use this selection when the TERM[2:0] bits (PGxyPCI[14:12]) are set to auto-termination.

Value	Description
111	Reserved
110	Reserved
101	Latched any edge(1)
100	Latched rising edge
011	Latched
010	Any edge
001	Rising edge
000	Level-sensitive

Bit 23 – SWPCI Software PCI Control

Value	Description
1	Drives a '1' to PCI logic assigned to by the SWPCIM[1:0] control bits
0	Drives a '0' to PCI logic assigned to by the SWPCIM[1:0] control bits

Bits 22:21 – SWPCIM[1:0] Software PCI Control Mode

Value	Description
11	Reserved
10	SWPCI bit is assigned to termination qualifier logic
01	SWPCI bit is assigned to acceptance qualifier logic
00	SWPCI bit is assigned to PCI acceptance logic

Bit 20 – LATMOD PCI SR Latch Mode

Value	Description
1	SR latch is Reset-dominant in Latched Acceptance modes
0	SR latch is set-dominant in Latched Acceptance modes

Bit 19 – TQPS Termination Qualifier Polarity Select

Value	Description
1	Inverted
0	Not inverted

Bits 18:16 – TQSS[2:0] Termination Qualifier Source Selection

Note: Polarity control bit, TQPS, has no effect on these selections.

Value	Description
111	SWPCI control bit only (qualifier forced to '1')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered
010	LEB is active
001	Duty cycle is active (base PWM Generator signal)
000	No termination qualifier used (qualifier forced to '1')

Bit 15 – TSYNCDIS Termination Synchronization Disable

Value	Description
1	Termination of latched PCI occurs immediately
0	Termination of latched PCI occurs at PWM EOC

Bits 14:12 – TERM[2:0] Termination Event Selection

Note: Do not use this selection when the ACP[2:0] bits (PGxyPCI[26:24]) are set for latched on any edge.

Value	Description
111	Selects PCI Source #9
110	Selects PCI Source #8
101	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
100	PGxTRIGC trigger event
011	PGxTRIGB trigger event
010	PGxTRIGA trigger event
001	Auto-Terminate: Terminate when PCI source transitions from active to inactive(2)
000	Manual Terminate: Terminate on a write of '1' to the SWTERM bit location

Bit 11 – AQPS Acceptance Qualifier Polarity Select

Value	Description
1	Inverted
0	Not inverted

Bits 10:8 – AQSS[2:0] Acceptance Qualifier Source Selection

Value	Description
111	SWPCI control bit only (qualifier forced to '0')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered
010	LEB is active
001	Duty cycle is active (base PWM Generator signal)
000	No acceptance qualifier is used (qualifier forced to '1')

Bit 7 – SWTERM PCI Software Termination
A write of '1' to this location will produce a termination event. This bit location always reads as '0'.

Bit 6 – PSYNC PCI Synchronization Control

Value	Description
1	PCI source is synchronized to PWM EOC
0	PCI source is not synchronized to PWM EOC

Bit 5 – PPS PCI Polarity Select

Value	Description
1	Inverted
0	Not inverted

Bits 4:0 – PSS[4:0] PCI Source Selection
Refer to Table 14-3 for device-specific PSS bit information.

14.3.3.6 PWM Generator Current Limit PCI Register(1)

Name: PGxCLPCI
Offset: 0x1064, 0x10AC, 0x10F4, 0x113C

Notes:

1. Caution should be exercised when modifying this register while PGxCON.ON = 1; unexpected results may occur.
2. This bit has no effect when the SWTERM control bit is used as the PCI Termination Event or if TERM[2:0] < '101'.

Bit	31	30	29	28	27	26	25	24
	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SWTERM	PSYNC	PPS	PSS[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – BPEN PCI Bypass Enable

Value	Description
1	PCI function is enabled and local PCI logic is bypassed; PWM Generator will be controlled by PCI function in the PWM Generator selected by the BPSEL[2:0] bits
0	PCI function is not bypassed

Bits 30:28 – BPSEL[2:0] PCI Bypass Source Selection
See Table 14-6 for device-specific PCI Bypass Source selection options.
Note: Selects '0' if the selected PWM Generator is not present.

Bit 27 – TERMPS PCI Termination Polarity Select bit
Note: This bit has no effect when the SWTERM control bit is used as the PCI Termination Event or if TERM[2:0] < '101'.

Value	Description
1	Inverted
0	Not inverted

Bits 26:24 – ACP[2:0] PCI Acceptance Criteria Selection
Note:

1. Do not use this selection when the TERM[2:0] bits (PGxyPCI[14:12]) are set to auto-termination.

Value	Description
111	Reserved
110	Reserved
101	Latched any edge(1)
100	Latched rising edge
011	Latched
010	Any edge
001	Rising edge
000	Level-sensitive

Bit 23 – SWPCI Software PCI Control

Value	Description
1	Drives a '1' to PCI logic assigned to by the SWPCIM[1:0] control bits
0	Drives a '0' to PCI logic assigned to by the SWPCIM[1:0] control bits

Bits 22:21 – SWPCIM[1:0] Software PCI Control Mode

Value	Description
11	Reserved
10	SWPCI bit is assigned to termination qualifier logic
01	SWPCI bit is assigned to acceptance qualifier logic
00	SWPCI bit is assigned to PCI acceptance logic

Bit 20 – LATMOD PCI SR Latch Mode

Value	Description
1	SR latch is Reset-dominant in Latched Acceptance modes
0	SR latch is set-dominant in Latched Acceptance modes

Bit 19 – TQPS Termination Qualifier Polarity Select

Value	Description
1	Inverted
0	Not inverted

Bits 18:16 – TQSS[2:0] Termination Qualifier Source Selection

Note: Polarity control bit, TQPS, has no effect on these selections.

Value	Description
111	SWPCI control bit only (qualifier forced to '1')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered
010	LEB is active
001	Duty cycle is active (base PWM Generator signal)
000	No termination qualifier used (qualifier forced to '1')

Bit 15 – TSYNCDIS Termination Synchronization Disable

Value	Description
1	Termination of latched PCI occurs immediately
0	Termination of latched PCI occurs at PWM EOC

Bits 14:12 – TERM[2:0] Termination Event Selection

Note: Do not use this selection when the ACP[2:0] bits (PGxyPCI[26:24]) are set for latched on any edge.

Value	Description
111	Selects PCI Source #9
110	Selects PCI Source #8
101	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
100	PGxTRIGC trigger event
011	PGxTRIGB trigger event
010	PGxTRIGA trigger event
001	Auto-Terminate: Terminate when PCI source transitions from active to inactive
000	Manual Terminate: Terminate on a write of '1' to the SWTERM bit location

Bit 11 – AQPS Acceptance Qualifier Polarity Select

Value	Description
1	Inverted
0	Not inverted

Bits 10:8 – AQSS[2:0] Acceptance Qualifier Source Selection

Value	Description
111	SWPCI control bit only (qualifier forced to '0')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered
010	LEB is active
001	Duty cycle is active (base PWM Generator signal)
000	No acceptance qualifier is used (qualifier forced to '1')

Bit 7 – SWTERM PCI Software Termination
A write of '1' to this location will produce a termination event. This bit location always reads as '0'.

Bit 6 – PSYNC PCI Synchronization Control

Value	Description
1	PCI source is synchronized to PWM EOC
0	PCI source is not synchronized to PWM EOC

Bit 5 – PPS PCI Polarity Select

Value	Description
1	Inverted
0	Not inverted

Bits 4:0 – PSS[4:0] PCI Source Selection
Refer to Table 14-3 for device-specific PSS bit information.

14.3.3.7 PWM Generator Feed Forward PCI(1)

Name: PGxFFPCI
Offset: 0x1068, 0x10B0, 0x10F8, 0x1140

Notes:

1. Caution should be exercised when modifying this register while PGxCON.ON = 1; unexpected results may occur.
2. This bit has no effect when the SWTERM control bit is used as the PCI Termination Event or if TERM[2:0] < '101'.

Bit	31	30	29	28	27	26	25	24
	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SWTERM	PSYNC	PPS	PSS[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – BPEN PCI Bypass Enable

Value	Description
1	PCI function is enabled and local PCI logic is bypassed; PWM Generator will be controlled by PCI function in the PWM Generator selected by the BPSEL[2:0] bits
0	PCI function is not bypassed

Bits 30:28 – BPSEL[2:0] PCI Bypass Source Selection
See Table 14-6 for device-specific PCI Bypass Source selection options.
Note: Selects `0' if the selected PWM Generator is not present.

Bit 27 – TERMPS PCI Termination Polarity Select bit
Note: This bit has no effect when the SWTERM control bit is used as the PCI Termination Event or if TERM[2:0] < '101'.

Value	Description
1	Inverted
0	Not inverted

Bits 26:24 – ACP[2:0] PCI Acceptance Criteria Selection(1)
Note:

1. Do not use this selection when the TERM[2:0] bits (PGxyPCI[14:12]) are set to auto-termination.

Value	Description
111	Reserved
110	Reserved
101	Latched any edge(1)
100	Latched rising edge
011	Latched
010	Any edge
001	Rising edge
000	Level-sensitive

Bit 23 – SWPCI Software PCI Control

Value	Description
1	Drives a '1' to PCI logic assigned to by the SWPCIM[1:0] control bits
0	Drives a '0' to PCI logic assigned to by the SWPCIM[1:0] control bits

Bits 22:21 – SWPCIM[1:0] Software PCI Control Mode

Value	Description
11	Reserved
10	SWPCI bit is assigned to termination qualifier logic
01	SWPCI bit is assigned to acceptance qualifier logic
00	SWPCI bit is assigned to PCI acceptance logic

Bit 20 – LATMOD PCI SR Latch Mode

Value	Description
1	SR latch is Reset-dominant in Latched Acceptance modes
0	SR latch is set-dominant in Latched Acceptance modes

Bit 19 – TQPS Termination Qualifier Polarity Select

Value	Description
1	Inverted
0	Not inverted

Bits 18:16 – TQSS[2:0] Termination Qualifier Source Selection

Value	Description
111	SWPCI control bit only (qualifier forced to '1')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered
010	LEB is active
001	Duty cycle is active (base PWM Generator signal)
000	No termination qualifier used (qualifier forced to '1')

Bit 15 – TSYNCDIS Termination Synchronization Disable

Value	Description
1	Termination of latched PCI occurs immediately
0	Termination of latched PCI occurs at PWM EOC

Bits 14:12 – TERM[2:0] Termination Event Selection

Note:

1. PCI sources are device-dependent.

2. Do not use this selection when the ACP[2:0] bits (PGxyPCI[26:24]) are set for latched on any edge.

Value	Description
111	Selects PCI Source #9(2)
110	Selects PCI Source #8(2)
101	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
100	PGxTRIGC trigger event
011	PGxTRIGB trigger event
010	PGxTRIGA trigger event
001	Auto-Terminate: Terminate when PCI source transitions from active to inactive(3)
000	Manual Terminate: Terminate on a write of '1' to the SWTERM bit location

Bit 11 – AQPS Acceptance Qualifier Polarity Select

Value	Description
1	Inverted
0	Not inverted

Bits 10:8 – AQSS[2:0] Acceptance Qualifier Source Selection

Value	Description
111	SWPCI control bit only (qualifier forced to '0')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered
010	LEB is active
001	Duty cycle is active (base PWM Generator signal)
000	No acceptance qualifier is used (qualifier forced to '1')

Bit 7 – SWTERM PCI Software Termination
A write of '1' to this location will produce a termination event. This bit location always reads as '0'.

Bit 6 – PSYNC PCI Synchronization Control

Value	Description
1	PCI source is synchronized to PWM EOC
0	PCI source is not synchronized to PWM EOC

Bit 5 – PPS PCI Polarity Select

Value	Description
1	Inverted
0	Not inverted

Bits 4:0 – PSS[4:0] PCI Source Selection
Refer to Table 14-3 for device-specific PSS bit information.

14.3.3.8 PWM Generator Sync PCI(1)

Name: PGxSPCI
Offset: 0x106C, 0x10B4, 0x10FC, 0x1144

Notes:

1. Caution should be exercised when modifying this register while PGxCON.ON = 1; unexpected results may occur.
2. This bit has no effect when the SWTERM control bit is used as the PCI Termination Event or if TERM[2:0] < '101'.

Bit	31	30	29	28	27	26	25	24
	BPEN	BPSEL[2:0]			TERMPS	ACP[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SWTERM	PSYNC	PPS	PSS[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – BPEN PCI Bypass Enable

Value	Description
1	PCI function is enabled and local PCI logic is bypassed; PWM Generator will be controlled by PCI function in the PWM Generator selected by the BPSEL[2:0] bits
0	PCI function is not bypassed

Bits 30:28 – BPSEL[2:0] PCI Bypass Source Selection
See Table 14-6 for device-specific PCI Bypass Source selection options.
Note: Selects '0' if the selected PWM Generator is not present.

Bit 27 – TERMPS PCI Termination Polarity Select bit
Note: This bit has no effect when the SWTERM control bit is used as the PCI Termination Event or if TERM[2:0] < '101'.

Value	Description
1	Inverted
0	Not inverted

Bits 26:24 – ACP[2:0] PCI Acceptance Criteria Selection
Note:

1. Do not use this selection when the TERM[2:0] bits (PGxyPCI[14:12]) are set to auto-termination.

Value	Description
111	Reserved
110	Reserved
101	Latched any edge(1)
100	Latched rising edge
011	Latched
010	Any edge
001	Rising edge
000	Level-sensitive

Bit 23 – SWPCI Software PCI Control

Value	Description
1	Drives a '1' to PCI logic assigned to by the SWPCIM[1:0] control bits
0	Drives a '0' to PCI logic assigned to by the SWPCIM[1:0] control bits

Bits 22:21 – SWPCIM[1:0] Software PCI Control Mode

Value	Description
11	Reserved
10	SWPCI bit is assigned to termination qualifier logic
01	SWPCI bit is assigned to acceptance qualifier logic
00	SWPCI bit is assigned to PCI acceptance logic

Bit 20 – LATMOD PCI SR Latch Mode

Value	Description
1	SR latch is Reset-dominant in Latched Acceptance modes
0	SR latch is set-dominant in Latched Acceptance modes

Bit 19 – TQPS Termination Qualifier Polarity Select

Value	Description
1	Inverted
0	Not inverted

Bits 18:16 – TQSS[2:0] Termination Qualifier Source Selection

Note:

1. Polarity control bit, TQPS, has no effect on these selections.

Value	Description
111	SWPCI control bit only (qualifier forced to '1')(1)
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered
010	LEB is active
001	Duty cycle is active (base PWM Generator signal)
000	No termination qualifier used (qualifier forced to '1')(1)

Bit 15 – TSYNCDIS Termination Synchronization Disable

Value	Description
1	Termination of latched PCI occurs immediately
0	Termination of latched PCI occurs at PWM EOC

Bits 14:12 – TERM[2:0] Termination Event Selection

Notes:

1. PCI sources are device-dependent.
2. Do not use this selection when the ACP[2:0] bits (PGxyPCI[26:24]) are set for latched on any edge.

Value	Description
111	Selects PCI Source #9(1)
110	Selects PCI Source #8(1)
101	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
100	PGxTRIGC trigger event
011	PGxTRIGB trigger event
010	PGxTRIGA trigger event
001	Auto-Terminate: Terminate when PCI source transitions from active to inactive(2)
000	Manual Terminate: Terminate on a write of '1' to the SWTERM bit location

Bit 11 – AQPS Acceptance Qualifier Polarity Select

Value	Description
1	Inverted
0	Not inverted

Bits 10:8 – AQSS[2:0] Acceptance Qualifier Source Selection

Value	Description
111	SWPCI control bit only (qualifier forced to '0')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered
010	LEB is active
001	Duty cycle is active (base PWM Generator signal)
000	No acceptance qualifier is used (qualifier forced to '1')

Bit 7 – SWTERM PCI Software Termination

A write of '1' to this location will produce a termination event. This bit location always reads as '0'.

Bit 6 – PSYNC PCI Synchronization Control

Value	Description
1	PCI source is synchronized to PWM EOC
0	PCI source is not synchronized to PWM EOC

Bit 5 – PPS PCI Polarity Select

Value	Description
1	Inverted
0	Not inverted

Bits 4:0 – PSS[4:0] PCI Source Selection

Refer to Table 14-1 for device-specific PSS bit information.

14.3.3.9 PWM Generator x Leading-Edge Blanking Register

Name: PGxLEB
Offset: 0x1070, 0x10B8, 0x1100, 0x1148

Caution should be exercised when modifying this register while PGxCON.ON = 1; unexpected results may occur.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					PHR	PHF	PLR	PLF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LEB[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LEB[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bit 19 – PHR PWMx Rising Edge Trigger Enable bit

Value	Description
1	Rising edge of PWMx will trigger the LEB duration counter
0	LEB ignores the rising edge of PWMx

Bit 18 – PHF PWMxH Falling Edge Trigger Enable bit

Value	Description
1	Falling edge of PWMx will trigger the LEB duration counter
0	LEB ignores the falling edge of PWMx

Bit 17 – PLR PWMx Rising Edge Trigger Enable bit

Value	Description
1	Rising edge of PWMx will trigger the LEB duration counter
0	LEB ignores the rising edge of PWMx

Bit 16 – PLF PWMx Falling Edge Trigger Enable bit

Value	Description
1	Falling edge of PWMx will trigger the LEB duration counter
0	LEB ignores the falling edge of PWMx

Bits 15:8 – LEB[15:8] Leading-Edge Blanking Period bits
These bits select the leading edge blanking period. The 4 LSbs of the blanking time are not implemented, providing a blanking resolution of TPWMCLK. The LEB period is (LEB[15:4]+1)*TPWMCLK. The minimum blanking period is 5, values 0,1,2,3,4 all get LEB period of 5*TPWMCLK.

Bits 7:4 – LEB[7:4] Leading-Edge Blanking Period bits
These bits select the leading edge blanking period. The 4 LSbs of the blanking time are not implemented, providing a blanking resolution of TPWMCLK. The LEB period is (LEB[15:4]+1)*TPWMCLK. The minimum blanking period is 5, values 0,1,2,3,4 all get LEB period of 5*TPWMCLK.

14.3.3.10 PWM Generator x Phase Register

Name: PGxPHASE
Offset: 0x1074, 0x10BC, 0x1104, 0x114C

Note: This register cannot be modified while PGxSTAT.UPDATE = 1.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					PHASE[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PHASE[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PHASE[7:4]				Reserved[3:0]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 19:16 – PHASE[19:16] PWM Generator x Phase Register

Bits 15:8 – PHASE[15:8] PWM Generator x Phase Register

Bits 7:4 – PHASE[7:4] PWM Generator x Phase Register

Bits 3:0 – Reserved[3:0]

14.3.3.11 PWM Generator x Duty Cycle Register

Name: PGxDC
Offset: 0x1078, 0x10C0, 0x1108, 0x1150

Note: This register cannot be modified while PGxSTAT.UPDATE = 1.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					DC[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DC[7:4]				Reserved[3:0]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 19:16 – DC[19:16] PWM Generator x Duty Cycle Register
Note: Duty cycle values less than 0x0010 should not be used.

Bits 15:8 – DC[15:8] PWM Generator x Duty Cycle Register
Note: Duty cycle values less than 0x0010 should not be used.

Bits 7:4 – DC[7:4] PWM Generator x Duty Cycle Register
Note: Duty cycle values less than 0x0010 should not be used.

Bits 3:0 – Reserved[3:0]

14.3.3.12 PWM Generator x Duty Cycle Adjustment Register

Name: PGxDCA
Offset: 0x107C, 0x10C4, 0x110C, 0x1154

Note: This register cannot be modified while PGxSTAT.UPDATE = 1.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					DCA[11:8]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCA[7:4]				Reserved[3:0]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 11:8 – DCA[11:8] PWM Generator x Duty Cycle Adjustment Value
Depending on the state of the selected PCI source, the PGxDCA value will be added to the value in the PGxDC register to create the effective duty cycle. When the PCI source is active, PGxDCA is added.

Bits 7:4 – DCA[7:4] PWM Generator x Duty Cycle Adjustment Value
Depending on the state of the selected PCI source, the PGxDCA value will be added to the value in the PGxDC register to create the effective duty cycle. When the PCI source is active, PGxDCA is added.

Bits 3:0 – Reserved[3:0]

14.3.3.13 PWM Generator x Period Register

Name: PGxPER
Offset: 0x1080, 0x10C8, 0x1110, 0x1158

Note: This register cannot be modified while PGxSTAT.UPDATE = 1.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					PER[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PER[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PER[7:4]				Reserved[3:0]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 19:16 – PER[19:16] PWM Generator x Period Register
Note: Period values less than 0x0100 should not be used.

Bits 15:8 – PER[15:8] PWM Generator x Period Register
Note: Period values less than 0x0100 should not be used.

Bits 7:4 – PER[7:4] PWM Generator x Period Register
Note: Period values less than 0x0100 should not be used.

Bits 3:0 – Reserved[3:0]

14.3.3.14 PWM Generator x Trigger A Register

Name: PGxTRIGA
Offset: 0x1084, 0x10CC, 0x1114, 0x115C

Note: This register cannot be modified while PGxSTAT.UPDATE = 1.

Bit	31	30	29	28	27	26	25	24
	CAHALF							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
					TRIGA[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TRIGA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TRIGA[7:4]				Reserved[3:0]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 31 – CAHALF Specifies where the trigger compare time occurs

Value	Description
1	The second phase of the center aligned period
0	The first phase of the center aligned period

Bits 19:16 – TRIGA[19:16] PWM Generator x Trigger A Register

Bits 15:8 – TRIGA[15:8] PWM Generator x Trigger A Register

Bits 7:4 – TRIGA[7:4] PWM Generator x Trigger A Register

Bits 3:0 – Reserved[3:0]

14.3.3.15 PWM Generator x Trigger B Register

Name: PGxTRIGB
Offset: 0x1088, 0x10D0, 0x1118, 0x1160

Note: This register cannot be modified while PGxSTAT.UPDATE = 1.

Bit	31	30	29	28	27	26	25	24
	CAHALF							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
					TRIGB[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TRIGB[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TRIGB[7:4]				Reserved[3:0]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 31 – CAHALF Specifies where the trigger compare time occurs

Value	Description
1	The second phase of the center-aligned period
0	The first phase of the center-aligned period

Bits 19:16 – TRIGB[19:16] PWM Generator x Trigger B Register

Bits 15:8 – TRIGB[15:8] PWM Generator x Trigger B Register

Bits 7:4 – TRIGB[7:4] PWM Generator x Trigger B Register

Bits 3:0 – Reserved[3:0]

14.3.3.16 PWM Generator x Trigger C Register

Name: PGxTRIGC
Offset: 0x108C, 0x10D4, 0x111C, 0x1164

Note: This register cannot be modified while PGxSTAT.UPDATE = 1.

Bit	31	30	29	28	27	26	25	24
	CAHALF							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
					TRIGC[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TRIGC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TRIGC[7:4]				Reserved[3:0]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 31 – CAHALF Specifies where the trigger compare time occurs

Value	Description
1	The second phase of the center-aligned period
0	The first phase of the center-aligned period

Bits 19:16 – TRIGC[19:16] PWM Generator x Trigger C Register

Bits 15:8 – TRIGC[15:8] PWM Generator x Trigger C Register

Bits 7:4 – TRIGC[7:4] PWM Generator x Trigger C Register

Bits 3:0 – Reserved[3:0]

14.3.3.17 PWM Generator x Dead-Time Register

Name: PGxDT
Offset: 0x1090, 0x10D8, 0x1120, 0x1168

Note: This register cannot be modified while PGxSTAT.UPDATE = 1.

Bit	31	30	29	28	27	26	25	24
		DTH[14:8]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DTH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		DTL[14:8]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DTL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 30:16 – DTH[14:0] PWMx Dead-Time Delay

Bits 14:0 – DTL[14:0] PWMxL Dead-Time Delay

14.3.3.18 PWM Generator x Capture Register

Name: PGxCAP
Offset: 0x1094, 0x10DC, 0x1124, 0x116C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					CAP[19:16]			
Access					R	R	R	R
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CAP[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CAP[7:4]							
Access	R	R	R	R				
Reset	0	0	0	0				

Bits 19:16 – CAP[19:16] PGx Time Base Capture
PGx Time Base Capture bits.
Note: A capture event can be manually initiated in software by writing a '1' to PGxCAP[0].
The CAP bit (PGxSTAT[5]) will indicate when a new capture value is available. A read of PGxCAP will automatically clear the CAP bit and allow a new capture event to occur. PGxCAP[3:0] will always read as '0'.

Bits 15:8 – CAP[15:8] PGx Time Base Capture
PGx Time Base Capture bits.
Note: A capture event can be manually initiated in software by writing a '1' to PGxCAP[0].
The CAP bit (PGxSTAT[5]) will indicate when a new capture value is available. A read of PGxCAP will automatically clear the CAP bit and allow a new capture event to occur. PGxCAP[3:0] will always read as '0'.

Bits 7:4 – CAP[7:4] PGx Time Base Capture
PGx Time Base Capture bits.
Note: A capture event can be manually initiated in software by writing a '1' to PGxCAP[0].
The CAP bit (PGxSTAT[5]) will indicate when a new capture value is available. A read of PGxCAP will automatically clear the CAP bit and allow a new capture event to occur. PGxCAP[3:0] will always read as '0'.

14.4 Operation

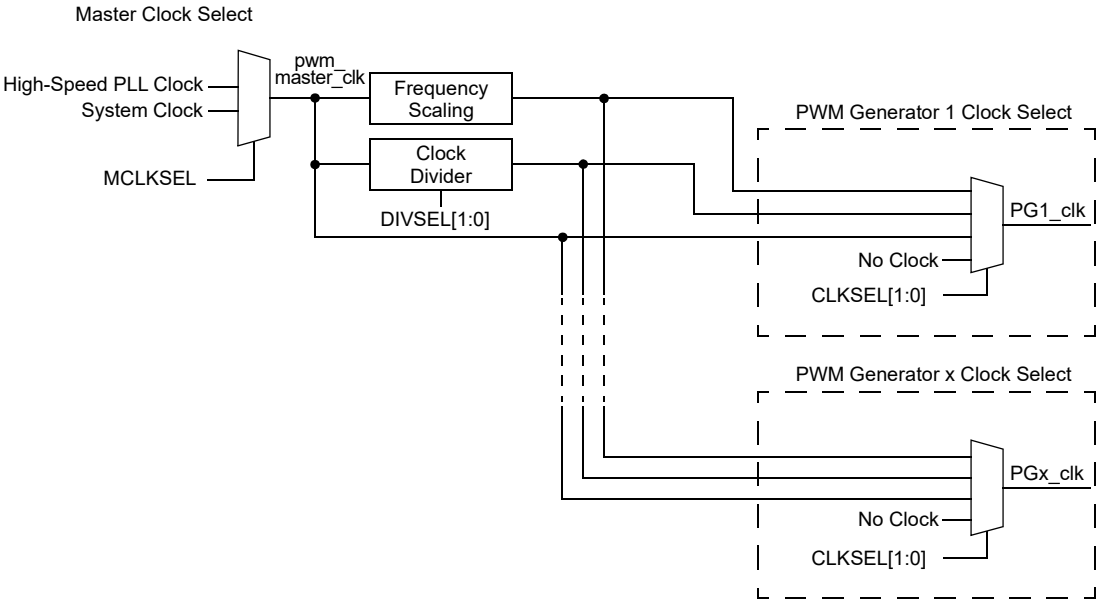
14.4.1 PWM Clocking

14.4.1.1 Master Clocking

The PWM module provides several clocking features at the top level of the module. Each PWM Generator can then independently select one of the clock sources, as shown in Figure 14-4.
The clock input into the PWM module is selected with the MCLKSEL control bit (PCLKCON[0]).

The CLKSEL[1:0] control bits (PGxCON[4:3]) are used to select the clock for each PWM Generator instance; see 14.4.2.1. PWM Generator Clocking for details. Frequency scaling and the clock divider are discussed in 14.4.3.3. Shared Clocking. The CLKSELx bits need to be changed from the default selection to allow the PWM Generator to function.

Figure 14-4. PWM Generator Clocking



Note: Writing MCLKSEL to a non-zero value will request and enable the chosen clock source, whether any PWM Generators are enabled or not. This allows a PLL, for example, to be requested and warmed up before using it as a PWM clock source. For the lowest device power consumption, the MCLKSEL bits should be set to the value, '00', if all PWM Generators have been disabled. Changing the MCLKSEL or CLKSEL[1:0] bits while ON (PGxCON[15]) = 1) is not recommended.

Note: The CPU and PWM typically run at different clock speeds depending on the application requirements. If the PWM clock speed is equal or slower than the CPU, writes to registers may have delayed behavior. For example, if SWTERM is used to clear a Fault, the instruction may need to be stretched with a REPEAT instruction to ensure the PWM can detect the edge within its clock cycle.

14.4.1.2 Clocking Equations in Standard Resolution

Some modes of operation utilize multiple period matches to complete one PWM 'cycle'. The following equation provides timing equations for the various operating modes.

Equation: PWM Period Calculation, Standard Resolution

Edge-Aligned, Variable Phase Operating Modes

$$F_{PWM} = \frac{16 \bullet F_{PGx\_clk}}{PGxPER + 16}$$

$$PGxPER = \left(\frac{16 \bullet F_{PGx\_clk}}{F_{PWM}}\right) - 16$$

Where:
F_{PWM} = Switching Frequency
PWM Period = $1/F_{PWM}$

Center-Aligned Modes, Edge-Aligned and Variable Phase Modes with Push-Pull Output Mode

$$F_{PWM} = \frac{8 \bullet F_{PGx\_clk}}{(PGxPER + 16)}$$

$$PGxPER = \left(\frac{8 \bullet F_{PGx\_clk}}{F_{PWM}}\right) - 16$$

Center-Aligned Modes with Push-Pull Output Mode

$$F_{PWM} = \frac{4 \bullet F_{PGx\_clk}}{(PGxPER + 16)}$$

$$PGxPER = \left(\frac{4 \bullet F_{PGx\_clk}}{F_{PWM}}\right) - 1$$

Equation: PWM Duty Cycle, Phase, Trigger and Dead-Time Calculations, Standard

$$MDC \text{ or } PGxDC(A) = (PGxPER + 16) \bullet Duty\ Cycle$$

Where:
Duty Cycle is % between 0 and 100

$$MPHASE \text{ or } PGxPHASE = 16 \bullet F_{PWM} \bullet Phase$$

$$PGxTRIGy = 16 \bullet F_{PWM} \bullet Trigger\ Offset$$
($y = A, B$ or C)

$$PGxDTy = 16 \bullet F_{PWM} \bullet Dead\ Time$$
($y = H$ or L)

Where:
Phase, Trigger Offset and *Dead Time* are specified in time units (ms, µs or ns)

Resolution

14.4.1.3 Clocking Synchronization

Due to the separate clocking domains of the PWM module and the CPU's system clock, there are inherent synchronization delays associated with SFR reads. This delay is dependent on the relative speeds of the CPU (sys_clk) and the PWM Generator clock (PGx_clk). Typically, the CPU clock will be slower and SFR data can be delayed up to one sys_clk cycle. It is also important to note that each PWM Generator can run at a different speed and this can have an effect on interactions between PWM Generators.

14.4.1.4 Minimum PWM Period and Pulse Width

The PWM module has some restrictions regarding minimum PWM periods and pulse widths. Depending on the operating mode, the pulse width can also be dependent on PHASE and TRIGy, in addition to duty cycle. The minimum pulse width applies to both active and inactive states; 0% and 100% duty cycles are supported.

Table 14-7 below lists restrictions to period and pulse width.

Table 14-7. Minimum Period and Pulse Width

Mode	Minimum Period (PGxPER or MPER)	Minimum Active Pulse Width in Counts	Minimum Inactive Pulse Width in Counts
Standard Resolution	0x0100	0x0010	Period – 0x0010

14.4.2 PWM Generator (PG) Features

Most of the features and controls of the PWM module are at the PWM Generator level and are controlled using each PWM Generator's SFRs. PWM Generator operation is based on triggers. The PWM Generator must receive a Start-of-Cycle (SOC) trigger signal to generate each PWM cycle. The trigger signal may be generated outside of the PWM Generator or the PWM Generator may be self-triggered. When a PWM Generator reaches the end of a PWM cycle, it produces an End-of-Cycle (EOC) trigger that can be used by other PWM Generators.

If multiple PWM Generators run at different frequencies, the triggers can be synchronized using the PCI Sync block.

14.4.2.1 PWM Generator Clocking

Each PWM Generator can be clocked independently of one another for maximum flexibility. There are four clock options available selected by the CLKSEL[1:0] bits (PGxCON[4:3]):

1. No clock (lowest power state)
2. Output of MCLKSEL
3. Output of clock divider
4. Output frequency scaler

This configuration flexibility allows, for example, one group of PWM Generators to operate at a higher frequency, while another group of PWM Generators operates at a lower frequency.

Note: Do not change the MCLKSEL or CLKSEL[1:0] bits while the PWM Generator is in operation (ON (PGxCON[15]) = 1).

14.4.2.2 PWM Modes

The PWM module supports a wide range of PWM modes for both motor control and power supply designs. The following PWM modes are supported:

- Independent Edge PWM mode (default)
- Variable Phase PWM mode
- Independent Edge PWM mode, Dual Output
- Center-Aligned PWM mode

- Double Update Center-Aligned PWM mode
- Dual Edge Center-Aligned PWM mode

The PWM modes are selected by setting the MODSEL[2:0] bits (PGxCON[2:0]). Some modes utilize multiple time base cycles to complete a single PWM cycle. Refer to the previous equation for specifics on timing.

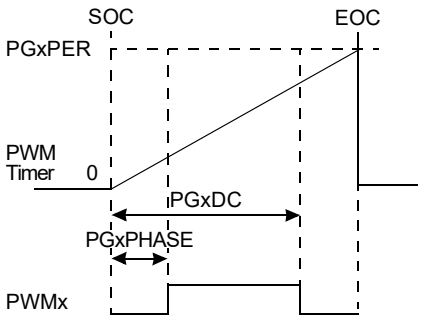
14.4.2.2.1 Independent Edge PWM Mode

Independent Edge PWM mode is used for many applications and can be used to create edge-aligned PWM signals, as well as PWM signals with arbitrary phase offsets. This mode is the default operating mode of the PWM Generator and is selected when MODSEL[2:0] = 000 (PGxCON[2:0]). Two Data registers must be written to define the rising and falling edges. The characteristics of the PWM signal are determined by these three SFRs:

- PGxPHASE: Determines the position of the PWM signal rising edge from the start of the timer count cycle
- PGxDC: Determines the position of the PWM signal falling edge from the start of the timer count cycle
- PGxPER: Determines the end of the PWM timer count cycle

A basic Edge-Aligned PWM mode signal is created by setting PGxPHASE = 0. Alternatively, multiple PWM Generators can be synchronized to one another by using the same PGxPHASE value. A constant value equivalent to the PGxPHASE value of other PWM Generators can be used to synchronize multiple PGs. The duty cycle is varied by writing to the PGxDC register. Arbitrary phase PWM signals may be generated by writing to PGxPHASE and PGxDC with the appropriate values. If PGxPHASE = PGxDC, no PWM pulse will be produced. If PGxDC ≥ PGxPER, a 100% duty cycle pulse is produced. Figure 14-5 shows the relationship between the control SFRs and the output waveform.

Figure 14-5. Independent Edge PWM Mode



Independent Edge

Multiple PWM generators can be synchronized to make a multiphase system. There are multiple methods that can be used in different applications and PWM modes. SOC triggers can be shared across groups of four PWM generators (PG1-PG4 or PG5-PG8).

In this example, three PWM generators are set to start and run synchronized:

- Configure all PWM generators for application (ckl, PER, DC, etc)
- PG1 is self-triggered SOCS = 0b0000
- PG2 SOCS trigger is PG1 , SOCS = 0b0001
- PG3 SOCS trigger is PG2, SOCS = 0b0010
- PG2 and PG3 ON bits are set; cycles will not start until trigger from PWM1
- PG1 ON bit is set; all PWM generators start simultaneously

In this example, a daisy-chain triggering scheme is used to synchronize and create an offset between each PWM generator's SOC:

1. Configure all PWM generators for application (ckl, PER, DC, etc)
2. PG1 is self-triggered SOCS = 0b0000
3. PG1TRIGA is set to provide phase offset for PG1 to PG2
4. PG1 PTGRGSEL = 0001 to select TRIGA as trigger
5. PG2 SOCS trigger is PG1 , SOCS = 0b0001
6. PG2 TRIGA is set to provide phase offset for PG2 to PG3
7. PG2 PTGRGSEL = 0001 to select TRIGA as trigger
8. PG3 SOCS trigger is PG2, SOCS = 0b0010
9. PG2 and PG3 ON bits are set; cycles will not start until trigger from PWM1
10. PG1 ON bit is set

14.4.2.2.2 Variable Phase PWM Mode

The Variable Phase PWM mode differs from Independent Edge mode in that one register is used to select the phase offset from the Start-of-Cycle, and a second register is used to select the width of the pulse. The Variable Phase PWM mode is useful as the PGxDC register is programmed to a constant value, while the PGxPHASE value is modulated. The PWM logic will automatically calculate rising edge and falling edge times to maintain a constant pulse width. Similarly, the user can leave the PGxPHASE register programmed to a constant value to create signals with a constant phase offset and variable duty cycle. The Variable Phase PWM mode is selected when MODSEL[2:0] (PGxCON[2:0]) = `001`. The characteristics of the PWM signal are determined by these three SFRs:

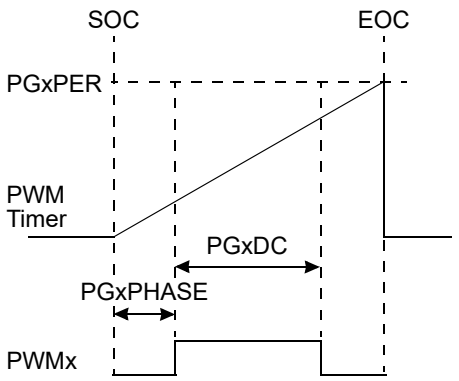
- PGxPHASE: Determines the offset of the PWM signal rising edge from the start of the timer cycle
- PGxDC: Determines the width of the PWM pulse and location of the PWM signal falling edge
- PGxPER: Determines the end of the PWM timer count cycle

When updating PER in Variable Phase mode, ensure that the client PWM generator PER > host PER. The recommended method is client PER = host PER + max PHASE to ensure that the client PWM generator does not generate its own SOC.

The host will always provide an SOC to trigger the client PWM generators. If the host PER is lengthened such that client PER > host PER, a client PWM cycle may be missed as the host SOC trigger arrives when the client cycle is in progress and the trigger is ignored.

Figure 14-6 shows the relationship between the control SFRs and the output waveform.

Figure 14-6. Variable Phase PWM Mode

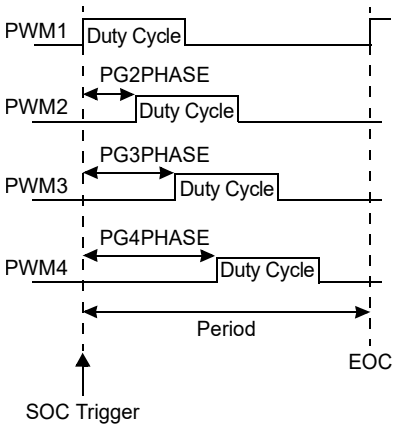


The Master Duty Cycle SFR (MDC) can also be used to change the duty cycle of all phases with a single register write. An example of a multiphase system is shown in Figure 14-7. Variable Phase mode cannot support active duty cycles across EOC boundaries. Ensure Phase + DC ≤ Period to allow for completion of the duty cycle for EOC.

In this scenario, PG1 provides SOC triggers to PG2-PG4. The output wave form is shown in Figure 14-7.

1. Set all PGs to same clock source.
2. Set all PGs in Variable Phase mode, MODSEL = 0b001.
3. Set PG1 for self trigger, SOCS = 0b0000.
4. Set PG2-PG4 to use PG1's SOC trigger, SOCS = 0b0001.
5. Write initial PER, DC and PHASE to all PGs.
6. Enable outputs, PENx = '1'.
7. Set ON = '1' of PG2-PG4. No cycles will start until PG1 sends trigger.
8. Set ON = '1' of PG1. This will start all PG synchronously.

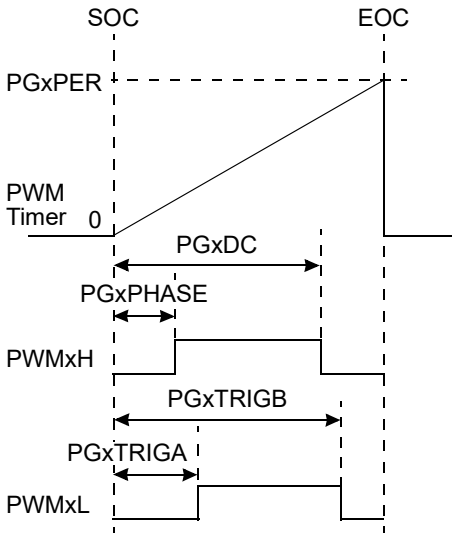
Figure 14-7. Multiphase PWM Example



14.4.2.2.3 Dual PWM Mode

The Dual PWM mode allows a single PWM Generator to produce two independent pulse widths on the PWMx output pins. This mode is the equivalent of Independent Edge mode, except that it allows a second PWM pulse to be produced if the Independent Output mode is used. The Dual PWM modes are selected when MODSEL[2:0] (PGxCON[2:0]) = 010. The PGxTRIGA and PGxTRIGB registers function as a second set of PGxPHASE and PGxDC registers to allow control of a second duty cycle generator. Figure 14-8 shows the relationship between the control SFRs and the output waveform.

Figure 14-8. Dual PWM Mode



The PGxTRIGA and PGxTRIGB event output signals continue to operate normally in this mode, and can still be used as phase offset triggers for other PWM Generators, ADC triggers, etc. If an independent trigger is needed, the PGxTRIGC register can be used. For additional information on ADC triggers, see 14.4.2.9.1. ADC Triggers.

Since the PWM signals produced on the PWMx pins are produced from the same PWM generator, they will be equally affected by any PWM Control Input (PCI) signals that are enabled. The PWMx pins will be driven to the states defined in the PGxIOCON register. Therefore, it is important that the two PWM outputs be used for related application functions if the PCI signals are to be used.

14.4.2.2.4 Center-Aligned PWM Mode

Center-Aligned PWM mode signals avoid coincident rising or falling edges between PWM Generators when the duty cycles are different, reducing excessive current ripple and filtering requirements in power inverter applications.

The PWM pulse maintains symmetry around the end of the first timer cycle and the beginning of the second cycle. If the duty cycle of the PWM signal is increased, then the position of the rising edge and the falling edge will change to maintain this symmetry. Center-Aligned PWM mode is selected when MODSEL[2:0] (PGxCON[2:0]) = 100. Center-Aligned PWM operating mode uses two timer cycles to produce a single pulse. The characteristics of the PWM signal are defined by two SFRs:

- PGxDC: Determines the width of the PWM pulse from the center of the two timer cycles
- PGxPER: Determines the end of the PWM timer count cycle

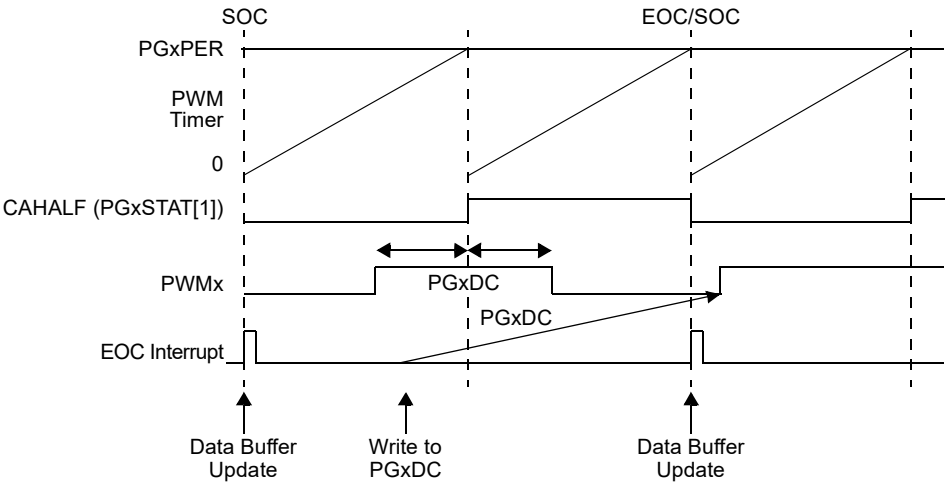
The falling edge occurs when the PWM Generator Timer = PGxDC, and the rising edge occurs when the PG Timer = PGxPER – PGxDC + 1. An offset of 1 is added to the rising edge calculation to produce symmetry between the two timer count cycles. A PGxDC value of '1', for example, will produce a pulse that is two cycles in duration. When PGxDC =0, there are restrictions on the maximum PER value. In standard resolution, the max value is 0xFFFE. If the PER value is above this value, a short pulse at EOC may occur.

The timer cycle is tracked using the CAHALF status bit (PGxSTAT[1]), and is read as '0' on the first half of cycle and '1' on the second half. Buffer updates to the duty cycle or period are allowed only at the beginning of the first timer cycle. The End-of-Cycle (EOC) interrupt is generated only after the completion of both period cycles. Figure 14-9 shows the relationship between the control SFRs and the output waveform. See 14.4.2.10. Data Buffering for additional information on data buffering.

Center-Aligned mode data restrictions and behavior:

- PHASE must be > 0; if not rising edge will not occur
- PHASE must be < PER; if not falling edge will not occur
- DC must be > 0; if not falling edge will not occur
- DC must be < PER; if not falling edge will not occur, 0% DC
- IF PER=DC, DC is 100%

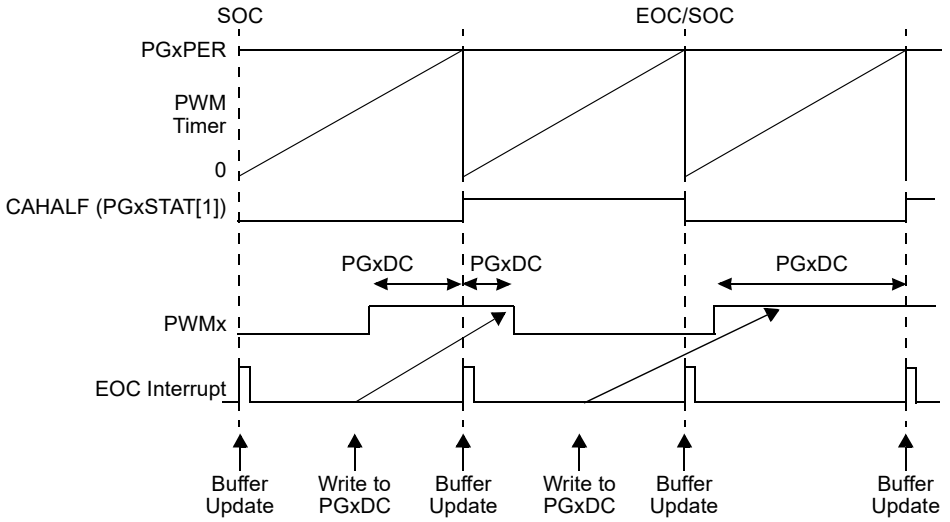
Figure 14-9. Center-Aligned PWM Mode



14.4.2.2.5 Double Update Center-Aligned PWM Mode

Double Update Center-Aligned PWM mode works identically to Center-Aligned PWM mode, except that two interrupts and two data buffer updates occur per PWM cycle. This mode is useful when the user wants to decrease the latency of a control loop response. Note that this will eliminate the symmetrical nature of the Center-Aligned PWM mode pulse, since the rising edge and falling edge of the pulse can be controlled independently. Double Update Center-Aligned PWM mode is selected when MODSEL[2:0] (PGxCON[2:0]) = `101`. Figure 14-10 shows the relationship between the control SFRs and the output waveform.

Figure 14-10. Double Update Center-Aligned Mode



14.4.2.2.6 Dual Edge Center-Aligned PWM Mode

Dual Edge Center-Aligned PWM mode works identically to Double Update Center-Aligned PWM mode, but allows the rising edge time and the falling edge time to be controlled via separate Data registers. This mode gives the user the most flexibility in the adjustment of the center-aligned pulse, yet offers a lower frequency of interrupt events. Note that this will eliminate the symmetrical nature of the center-aligned PWM pulse unless the PGxPHASE = PGxDC.

- PGxPHASE: Determines the rising edge time pulse from the center of the two timer cycles
- PGxDC: Determines the falling edge time pulse from the center of the two timer cycles

Note: PGxPHASE must be used for PHASE data; MPHASE is not supported.

Both Single and Double Data Buffer Update modes are available within the Dual Edge Center-Aligned PWM mode. Single Update mode is selected when MODSEL[2:0] = `110` and Double Update mode is selected when MODSEL[2:0] = `111`. In Single Update mode, the user may write a new PGxPHASE and PGxDC value at any time during the cycle to be used on the next center-aligned cycle. In Double Update mode, an interrupt event and a Data register update occur every timer cycle. This provides user software the opportunity to modify the PGxDC value for the falling edge event and PGxPHASE for the rising edge event. User software must check the state of the CAHALF bit (PGxSTAT[1]) to determine the appropriate register to update. If CAHALF = `0` (first half of the center-aligned cycle), the user software should write to the PGxDC register. If CAHALF = `1` (second half of cycle), the user software should write to the PGxPHASE register. Figure 14-11 and Figure 14-12 show the relationship between the control SFRs and the output waveform.

Figure 14-11. Dual Edge Center-Aligned PWM Mode (MODSEL[2:0] = **110**)

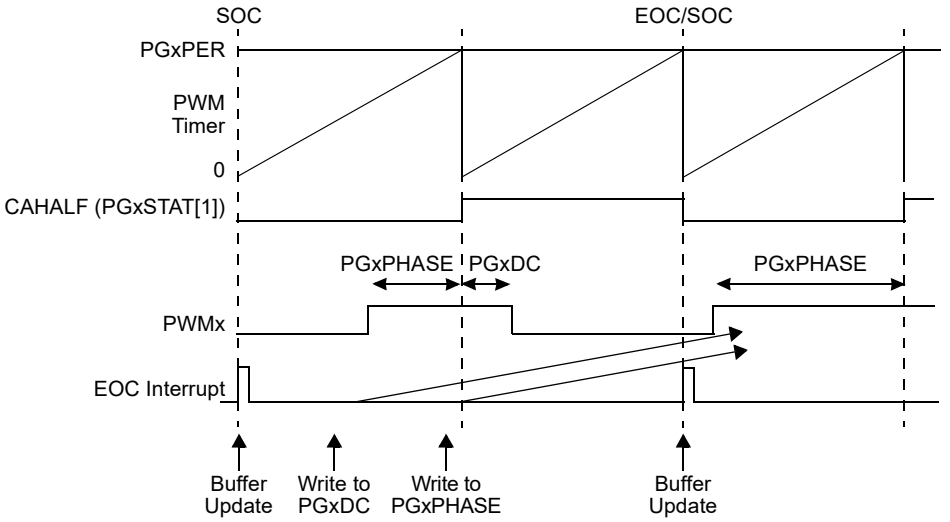
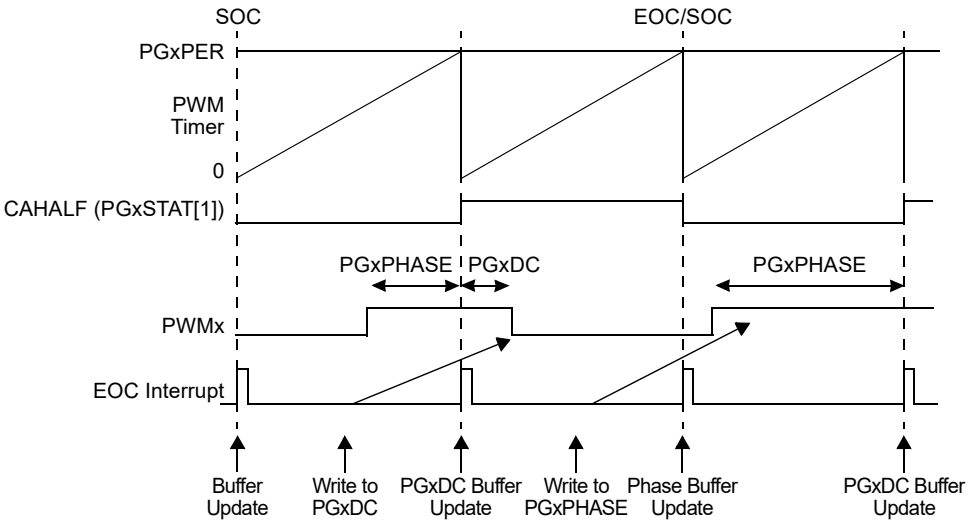


Figure 14-12. Dual Edge Center-Aligned PWM Mode (MODSEL[2:0] = **111**)



14.4.2.3 Output Modes

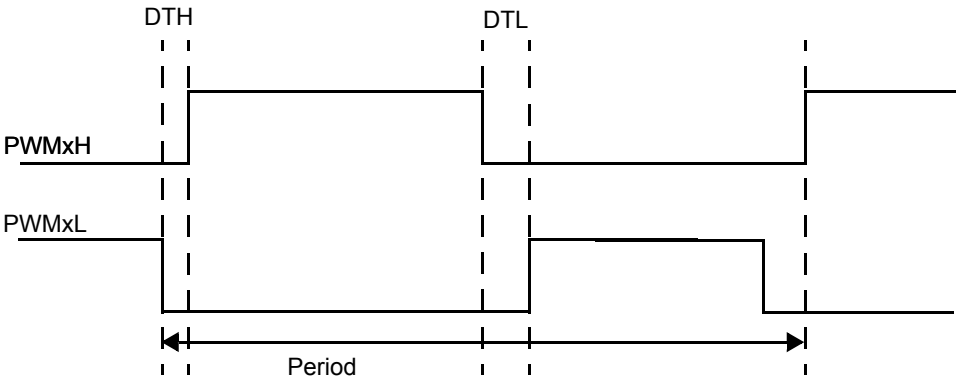
Each PWM Generator can be programmed to one of three output modes to control the behavior of the PWMxH and PWMxL pins. The output mode selection is independent of the PWM mode. The output modes are:

- Complementary Output mode (default)
- Independent Output mode
- Push-Pull Output mode

14.4.2.3.1 Complementary Output Mode

In Complementary Output mode, both the PWMxH and PWMxL signals are never active at the same time. A dead-time switching delay may be inserted between the two signals and is controlled by the PGxDT register. Complementary Output mode is selected when PMOD[1:0] (PGxIOCON[21:20]) = 00. For more information on dead time, see 14.4.2.6. Dead Time.

Figure 14-13. PWMxH/PWMxL Rising and Falling Edges Due to Dead Time



Output Override Behavior in Complementary Output Mode

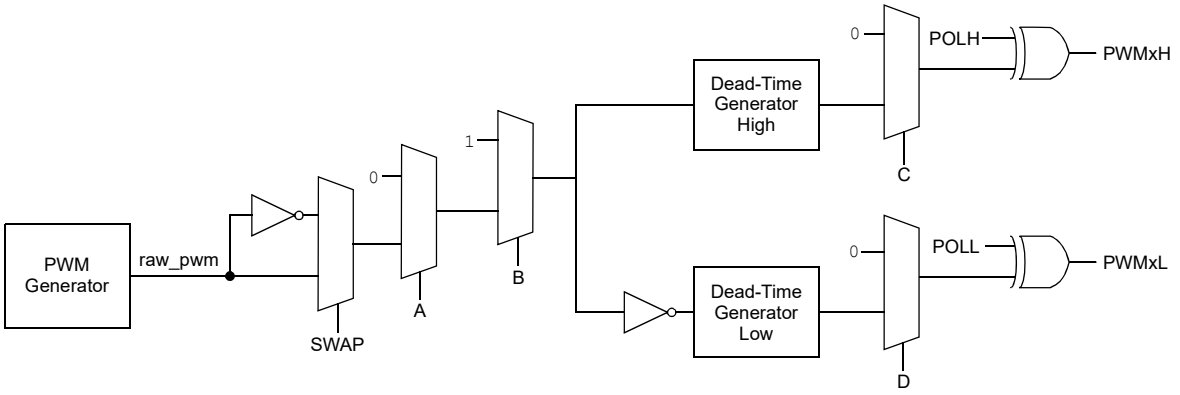
The PWMxH and PWMxL outputs may be controlled by external hardware signals or by software overrides. The output pins are restricted from being placed in a state which violates the complementary output relationship or in a state which violates dead-time insertion delays. An output pin may be driven inactive immediately as a result of a hardware event. However, a pin will not be driven active until the programmed dead-time delay has expired. The following hardware and software override states are programmed using the following:

- PCI Fault event, FLTDAT[1:0] (PGxIOCON[7:6])
- PCI current limit event, CLDAT[1:0] (PGxIOCON[5:4])
- PCI feed-forward event, FFDAT[1:0] (PGxIOCON[3:2])
- Debugger Halt, DBDAT[1:0] (PGxIOCON[1:0])
- Software override, OVRENH (PGxIOCON[13]) and OVRENL (PGxIOCON[12])
- Swap of PWMxH and PWMxL pins, SWAP (PGxIOCON[14])

Figure 14-14 shows the signal chain for override behavior in Complementary mode. The SWAP control is applied first and is therefore overridden by all other controls. Next, the request to drive a pin active is applied before dead time, so dead time is still applied to the output; after which, the dead-time generator is requested to drive a pin inactive. This arrangement allows the inactive state to take precedence over SWAP and an active request. Finally, the polarity control is applied to the pin.

The PCI overrides operate on a priority scheme; see 14.4.2.5.2. Output Control PCI Blocks for more information.

Figure 14-14. Override and SWAP Signal Flow, Complementary Mode



A = Request to drive PWMxL active (OVRDAT[0] = 1)
B = Request to drive PWMxH active (OVRDAT[1] = 1)
C = Request to drive PWMxH inactive (OVRDAT[1] = 0)
D = Request to drive PWMxL inactive (OVRDAT[0] = 0)

Table 14-8 shows the rules for pin override conditions. The active state is a '1' on the output pin and the inactive state is a '0'. An 'x' denotes a 'don't care' input; ~PWM indicates the complementary output of the PWM Generator's output.

Table 14-8. Override Behavior in Complementary Output Mode

Source	SWAP	OVRENH	OVRENL	OVRDAT[1:0]	FFDAT[1:0]	CLDAT[1:0]	FLTDAT[1:0]	DBGDAT[1:0]	PWMxH Signal	PWMxL Signal
Debug Override										
DEBUG	x	x	x	xx	xx	xx	xx	00	Inactive	Inactive
								01	Inactive	Active
								1x	Active	Inactive
Fault Override – Debug Override must be Inactive										
PCI FLT	x	x	x	xx	xx	xx	00	xx	Inactive	Inactive
							01		Inactive	Active
							1x		Active	Inactive
Current Limit Override – Fault and Debug Overrides must be Inactive										
PCI CL	x	x	x	xx	xx	00	xx	xx	Inactive	Inactive
						01			Inactive	Active
						1x			Active	Inactive
Feed-Forward Override – Software, Current Limit, Fault and Debug Overrides must be Inactive										
PCI FF	x	0	0	xx	00	xx	xx	xx	Inactive	Inactive
					01				Inactive	Active
					1x				Active	Inactive
Software Override – Current Limit, Fault and Debug Overrides must be Inactive										
Software Override	0	0	1	x0	xx	xx	xx	xx	PWM	Inactive
		1	0	0x					Inactive	~PWM
	1	0	0	00					~PWM	PWM
		0	1	x0					~PWM	Inactive
		0	1	x1					Inactive	Active
	x	1	0	1x					Active	Inactive
		1	1	00					Inactive	Inactive
		1	1	01					Inactive	Active
		1	1	1x					Active	Inactive

Output Behavior At Start-Up in Complementary Mode

When the PWM is initialized and the ON bit is set, the outputs immediately go to a Complementary state. There is an output delay as the signals propagate through the PWM logic. This causes the start of the active duty cycle to appear delayed, with the PWMxL output transitioning to an Inactive state (pin high) for four master_pwm_clk cycles. Once active duty cycle starts, the PWMx pins will behave as stated in Table 14-8.

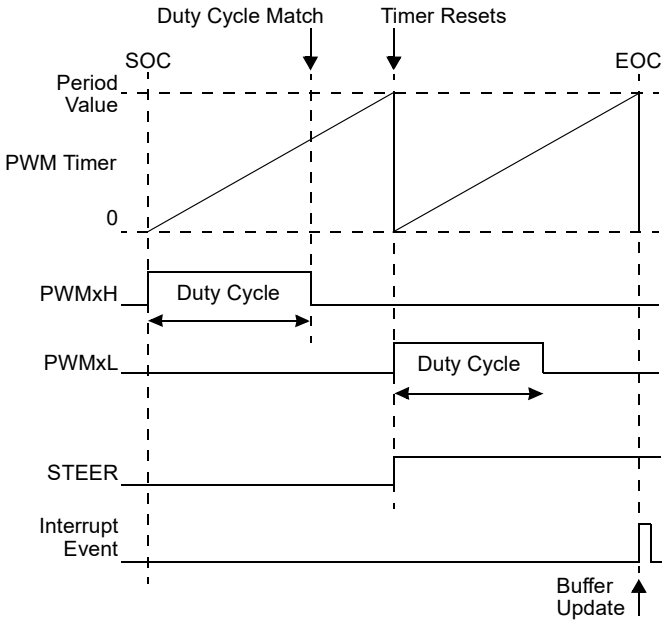
14.4.2.3.2 Independent Output Mode

In Independent Output mode, the output of the PWM Generator is connected to both the PWMxH and PWMxL pins. In most application scenarios, only the PWMxH or PWMxL pin would be enabled. The other pin remains available for GPIO or other peripheral functions. If the Dual PWM mode is selected, the PWM Generator will produce independent pulse widths on PWMxH and PWMxL, as described in 14.4.2.2.3. Dual PWM Mode. No dead-time switching delay is used in Independent Output mode. No restrictions exist for the states of the PWMxH and PWMxL pins; they can be controlled by external hardware signals or by software overrides. Independent Output mode is selected when PMOD[1:0] (PGxIOCON[21:20]) = 01.

14.4.2.3.3 Push-Pull Output Mode

The Push-Pull Output mode is similar to Independent Edge mode, however, the PWM cycle, as defined by the MODSEL[2:0] bits, is repeated twice each time a SOC trigger is received. The EOC trigger event and updates from Data registers are held off until the end of the second PWM cycle. Figure 14-15 shows the 2nd cycle that is invoked when using Push Pull Output mode.

Figure 14-15. Push-Pull PWM



Note: Operating the PWM in Push-Pull mode will double the period for a complete cycle, as there are two timer matches per cycle. If PGxTRIGy timers are used for event timing, the STEER signal can be used to gate application timing.

Push-Pull PWM mode is typically used in transformer coupled circuits to ensure that no net DC currents flow through the transformer. Push-Pull mode ensures that the same duty cycle PWM pulse is applied to the transformer windings in alternate directions. The phase of the push-pull count period can be determined by reading the STEER status bit (PGxSTAT[2]). If STEER = 0, the PWM Generator is generating the first PWM pulse. If STEER = 1, the PWM Generator is generating the second PWM pulse.

Since dead time is not available in Push-Pull mode, delays can be emulated in the Push-Pull Output mode by introducing a small phase offset with the PGxPHASE register. Similarly, the maximum duty cycle may be limited in software to avoid a pulse that ends too close to the start of the next PWM cycle.

Push-Pull Operation with Center-aligned Modes

When the PWM Generator is operated in one of the two Center-Aligned modes, and the Push-Pull mode is selected, a complete PWM cycle will comprise four time base cycles.

Figure 14-16 shows the operation of the module with Push-Pull Output mode and Center-Aligned PWM mode. This combination of modes limits PWM buffer updates and interrupt events to every fourth time base cycle. Therefore, the same pulse is produced on the PWMxH and PWMxL pins before any changes to the duty cycle are allowed. Similar interrupt behavior also occurs when Dual Edge Center-Aligned mode (one update per cycle) is selected (MODSEL[2:0] = `110`).

Figure 14-16. Push-Pull PWM: Center-Aligned Mode, Dual Edge Center-Aligned Mode with One Update per Cycle (MODSEL[2:0] = **110**)

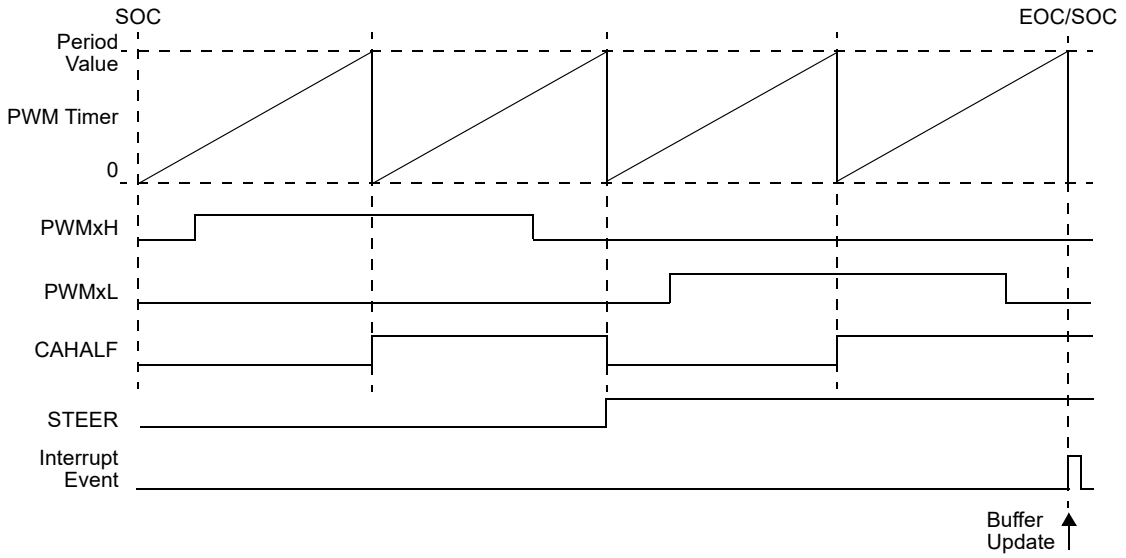
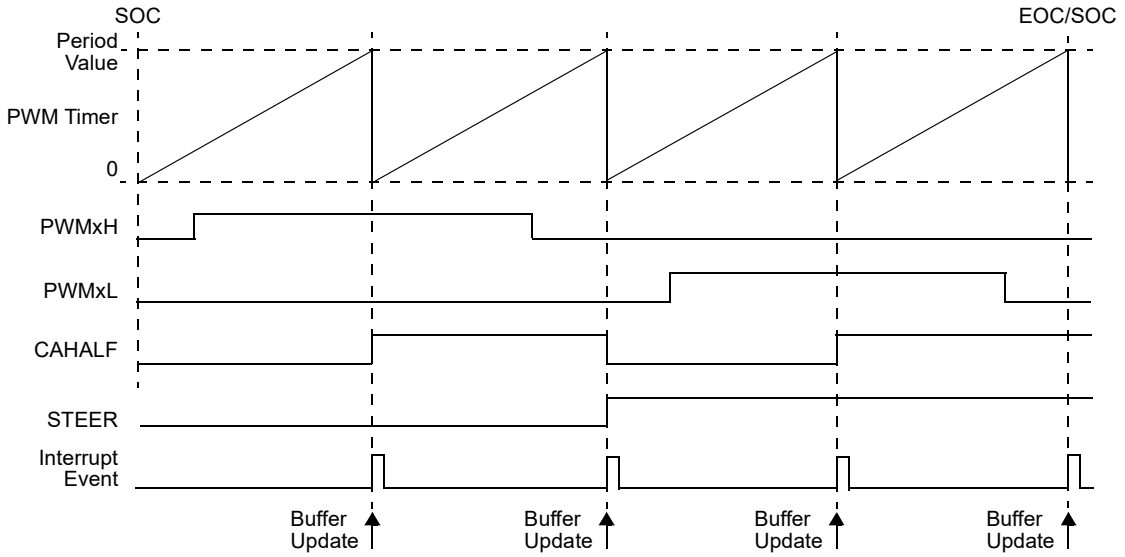


Figure 14-17 shows the operation of the module with Push-Pull Output mode and Dual Edge Center-Aligned PWM mode (two updates per cycle, MODSEL[2:0] = `111`) or Double Update Center-Aligned mode. This combination of modes allows a buffer update and interrupt event on every time base cycle. This operating configuration does not attempt to maintain symmetrical pulses on the PWMxH and PWMxL outputs, which is a requirement for many push-pull applications. User software can change the edge times of the center-aligned pulses after every edge event, which minimizes control loop latency.

Figure 14-17. Push-Pull PWM: Double Update Center-Aligned Mode, Dual Edge Center-Aligned Mode with Four Updates per Cycle (MODSEL[2:0] = **111**)



14.4.2.3.4 Output Override in Push-Pull and Independent Modes

When operating in Push-Pull or Independent Output modes, there is no logic that enforces a complementary relationship between the PWMxH and PWMxL signals. It is possible to drive both pins to an Active state with a software or hardware (PCI) override. This output state may or may not be desirable, depending on the external circuit that is controlled by the PWM Generator. Therefore, care must be taken when selecting the pin override values. Many push-pull applications require an equal pulse on both the PWMxH and PWMxL outputs to avoid a DC component. If the application is sensitive to this, perform software overrides after two complete timer cycles have taken place. Hardware PCI overrides should be configured to take effect after both timer cycles in the push-pull sequence have occurred. This can be accomplished by using the STEER signal, routed through the event logic to a pin, which can then be selected as an input to the PCI block.

Figure 14-18. Override and SWAP Signal Flow, Push-Pull Output Mode

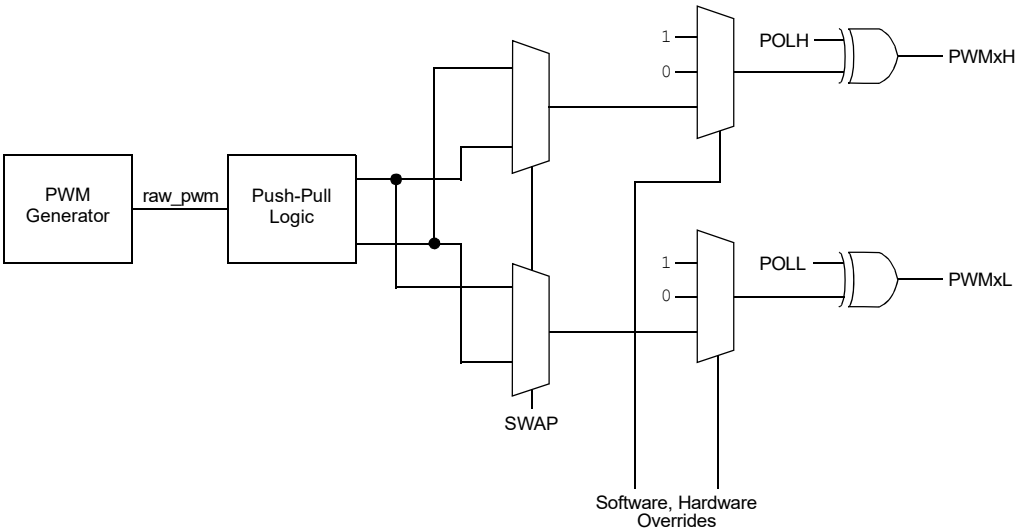


Table 14-9 shows the rules for pin override conditions. The Active state is a '1' on the output pin and the Inactive state is a '0'. An 'x' denotes a 'don't care' input; ~PWM indicates the complementary output of the PWM Generator's output.

Table 14-9. Override and SWAP Behavior in Push-Pull, Independent Output Modes

Source	SWAP	OVRENH	OVRENL	OVRDAT[1:0]	FFDAT[1:0]	CLDAT[1:0]	FLTDAT[1:0]	DBGDAT[1:0]	PWMxH Pin State	PWMxL Pin State
Debug Override										
DEBUG	x	x	x	xx	xx	xx	xx	00	Inactive	Inactive
								01	Inactive	Active
								10	Active	Inactive
								11	Active	Active
Fault Override – Debug Override Must be Inactive										
PCI FLT	x	x	x	xx	xx	xx	00	xx	Inactive	Inactive
							01		Inactive	Active
							10		Active	Inactive
							11		Active	Active
Current Limit Override – Fault and Debug Overrides Must be Inactive										
PCI CL	x	x	x	xx	xx	00	xx	xx	Inactive	Inactive
						01			Inactive	Active
						10			Active	Inactive
						11			Active	Active
Feed-Forward Override – Software, Current Limit, Fault and Debug Overrides Must be Inactive										
PCI FF	x	0	0	xx	00	xx	xx	xx	Inactive	Inactive
					01				Inactive	Active
					10				Active	Inactive
					11				Active	Active
Software Override – Current Limit, Fault and Debug Overrides Must be Inactive										

.....continued

Source	SWAP	OVRENH	OVRENL	OVRDAT[1:0]	FFDAT[1:0]	CLDAT[1:0]	FLTDAT[1:0]	DBGDAT[1:0]	PWMxH Pin State	PWMxL Pin State
Software Override	0	0	1	x0	xx	xx	xx	xx	PWMH	Inactive
		0	1	x1					PWMH	Active
		1	0	0x					Inactive	PWML
		1	0	1x					Active	PWML
	1	0	1	x0					PWML	Inactive
		0	1	x1					PWML	Active
		1	0	0x					Inactive	PWMH
		1	0	1x					Active	PWMH
	x	1	1	00					Inactive	Inactive
				01					Inactive	Active
				10					Active	Inactive
				11					Active	Active

14.4.2.4 PWM Generator Triggers

Each PWM Generator must receive a Start-of-Cycle (SOC) trigger to begin a PWM cycle. The trigger signal can be supplied by the PWM Generator itself (self-triggered) or another external trigger source. The SOC trigger can be generated from three sources:

- An internal source operating from the same clock source selected by the SOCS[3:0] bits (PGxCON[19:16])
- An external source selected by the PWM Control Input (PCI) Sync block
- A software trigger request, write to TRSET (PGxSTAT[7])

Any of the PWM Generators may act as a 'host' by providing the trigger for other PWM Generators. Many trigger configurations may be achieved, including:

- Multiple PWM outputs with independent periods (no synchronization between PWM Generators)
- Multiple PWM outputs with synchronized periods (synchronized operation)
- Multiple PWM outputs with offset phase relationships (triggered operation)

Synchronized operation is achieved by setting a (client) PWM Generator's SOCSx bits to that of another (host) PWM Generator, with the host's PGTRGSEL[2:0] bits (PGxEVT[2:0]) set to '000'. This selects the host's EOC to be used as the client's SOC trigger. When using PGxTRIGy to phase offset PWM generators from one another, a synchronization delay of up to five pwm_master_clk are present. If the TRIG value is '0', there will still be an offset. The TRIGy value may need to be compensated for the application.

Triggered operation is achieved in a similar way, but with the host's PGTRGSEL[2:0] bits (PGxEVT[2:0]) set to select one of the PGxTRIGy counters (y = A, B or C). The value specified in the host's TRIGy register defines the client's trigger offset from that of the host's SOC.

The SOCS[3:0] control bits have two special selections. When SOCS[3:0] = 0000, the PWM Generator is internally triggered. When SOSC[3:0] = 1111, no trigger source is selected. This selection is useful when the PWM Generator will be triggered only by software, using the TRSET bit, or from a source connected to the PCI Sync block. In this mode, the next PWM cycle will not start until another trigger is received. The sources available to the PCI Sync block include external signals, such as comparator events and device I/O pins, etc. One of the important functions of the PCI Sync block is to synchronize external input signals into the clock domain of the PWM Generator. See 14.4.2.5. PWM Control Input (PCI) Logic Blocks for more information on the PCI block. The PCI Sync can be OR'd into any of the other Start-of-Cycle inputs (SOCS[3:0]) as long as the block is enabled. The trigger output of another PWM Generator can also be used as a SOC event. See 14.4.2.9. Event Selection Block for configuration options.

14.4.2.4.1 Trigger Operation

A PWM cycle starts only when it receives a SOC trigger. When the time base reaches its end, the PWM cycle completes and the PWM Generator exits a triggered state. The PWM Generator must be retriggered to continue operation, which can be done in several ways.

- The PWM Generator is self-triggered (SOCS[3:0] = 0000, default)
- A new trigger pulse is received which is coincident with the end of the PWM cycle event. This can be achieved by multiple PWM Generators having matching PGxPER values, PWM modes and PWM Output modes

The TRIG status bit (PGxSTAT[0]) indicates whether the PWM Generator is in a triggered state.

The EOC signal is the default input to the SOC trigger selection multiplexer, which allows self-triggering. The EOC trigger is generated when the PWM Generator has finished a PWM cycle. It is also generated when the ON bit (PGxCON[15]) associated with the PWM Generator has been set. This allows all PWM Generators receiving the EOC signal to start in unison when the ON bit of the

host PWM Generator has been set. The ON bit of the other client PWM Generators needs to be set previously to achieve a synchronous start.

14.4.2.4.2 Triggering Modes

The PWM Generator provides two types of Triggering modes that determine how the SOC trigger is used. The Triggering modes are:

- Single Trigger mode (default)
- Retriggerable mode

The Trigger mode is selected using the TRGMOD[1:0] control bits (PGxCON[23:22]).

Single Trigger Mode

Single Trigger (Single Shot) mode is used when a PWM Generator timer is started at the same time as (or a time that is offset from) another PWM Generator timer. This mode is also useful for creating a single PWM pulse or creating a single delay based on an external event. If a timer cycle is currently in progress, any incoming SOC trigger pulses will be ignored. The entire timer cycle must complete before another SOC trigger can restart the timer. Single Trigger mode is selected when TRGMOD[1:0] = 00.

Retriggerable Mode

Retriggerable mode is different from Single Trigger mode in that a PWM cycle may be restarted before the end of a cycle that is already in progress. If this is done, the count will be reset when a new trigger is received and the current PWM cycle stopped. This mode can be especially useful when a PWM Generator is synchronized to an external, off-chip source that operates from a different clock source. The Retriggerable mode is selected when TRGMOD[1:0] = 01. The TRGCNT[2:0] bits (PGxCON[10:8]) can be written to produce a multiple cycle PWM event.

If using a software trigger, TRSET, subsequent writes of TRSET should wait for the TRIG bit to clear, indicating that the request number of cycle is complete.

14.4.2.4.3 Burst Mode

In some applications, it is desirable to have the PWM cycle repeat a certain number of times after the PWM Generator is triggered. The TRGCNT[2:0] control bits (PGxCON[10:8]) select the number of times the PWM cycle will be repeated after a trigger event. Any incoming triggers will be ignored until all PWM cycles are completed. Use of Retriggerable mode in conjunction with Burst mode is not recommended.

14.4.2.4.4 Trigger Registers in Center-Aligned Mode

When using any of the Center-Aligned modes, bit 15 of the PGxTRIGA, PGxTRIGB and PGxTRIGC Trigger registers specifies whether the trigger compare time occurs in the first phase (CAHALF = 0) or the second phase (CAHALF = 1). User software should limit the maximum time base count period to 0x7FFF (15 bits) in Center-Aligned PWM mode to ensure proper operation of the Trigger registers in all Center-Aligned modes. Otherwise, two trigger events may be generated in the center-aligned count phase depending on the values of the programmed period and Trigger register value. In some situations, it is desirable to have a trigger event occur in both count phases; this can be accomplished by programming two Trigger registers.

14.4.2.4.5 Behavior of PWM Generator Output Signal Across PWM Cycle Boundaries

During normal operation, the PWM Data registers will be programmed to create a PWM pulse which begins and terminates within a single PWM cycle. It is possible to write values to the PWM Data registers that will result in a 100% duty cycle output or produce an active output that spans across PWM cycles. The PWM Generator must remain in a continuously triggered state in order for the PWM output to remain active across PWM cycles. To remain triggered, the PWM Generator trigger input signal must be coincident with the EOC output signal. This will happen automatically when:

- The PWM Generator is self-triggered (SOCS[3:0] = 0000)

- The local PGxPER value is set to the same value as the external PWM Generator that is providing the trigger signal

If the PWM Generator trigger input signal does not occur at or before the EOC output signal, then the PWM Generator will exit the triggered state and the PWM Generator output will be driven inactive.

14.4.2.5 PWM Control Input (PCI) Logic Blocks

The PWM Control Input (PCI) Logic blocks are flexible state machines that can be used for a wide variety of purposes. The PCI blocks condition input signals and provide output signals used to trigger, gate and override the PWM outputs. The PCI also allows PWM Generators to interface with one another and external input signals. The PCI blocks can be used to implement output control and trigger algorithms in hardware instead of using software resources. There are four identical PCI blocks available for each PWM Generator. The PCI blocks are:

- Fault
- Current Limit
- Feed-Forward
- Sync

The names of the PCI block do not limit their usage; they are given unique names to designate the priority levels. The Sync PCI block is intended for triggering, specifically from external events, including other PWM Generators. The Fault, Current Limit and Feed-Forward PCI blocks are used to control the PWM output from external signals and other peripherals. The output state of the PWM pins can be independently configured to a predefined state for each PCI block, and it operates in a priority scheme if more than one PCI block requests control over the PWM outputs. Each PCI block has its own control register, PGxyPCI (with y = F, CL, FF or S), that contains the control bit associated with its operation. The PCI logic has three major components used to create logic functions:

- Inputs:
  - PCI source
  - PCI source qualifier, used to gate the PCI source signal
  - Terminator event, used to stop the 'PCI_active' output signal
  - Terminator qualifier event, used to gate the terminator event
- Acceptance logic
- Output and bypass function

Typical PCI source signals may include:

- Outputs to other PWM Generators
- Combo triggers (see 14.4.3.5. Combinatorial Triggers)
- Analog-to-Digital Converter (ADC)
- Analog comparator
- Input capture
- Configurable Logic Cell (CLC)
- External input (device pin)

The output of a PCI block (PCI_active signal) is made available to the PWM output logic and other PWM Generators. The status of the output signal of each PCI block is made available in the PGxSTAT register in both current and latched states. The PCI blocks can also generate interrupts; see 14.4.2.9.3. Event Interrupts for more information. The block diagrams of the PCI function are shown in Figure 14-19 and Figure 14-20.

Figure 14-19. PCI Function Block Diagram

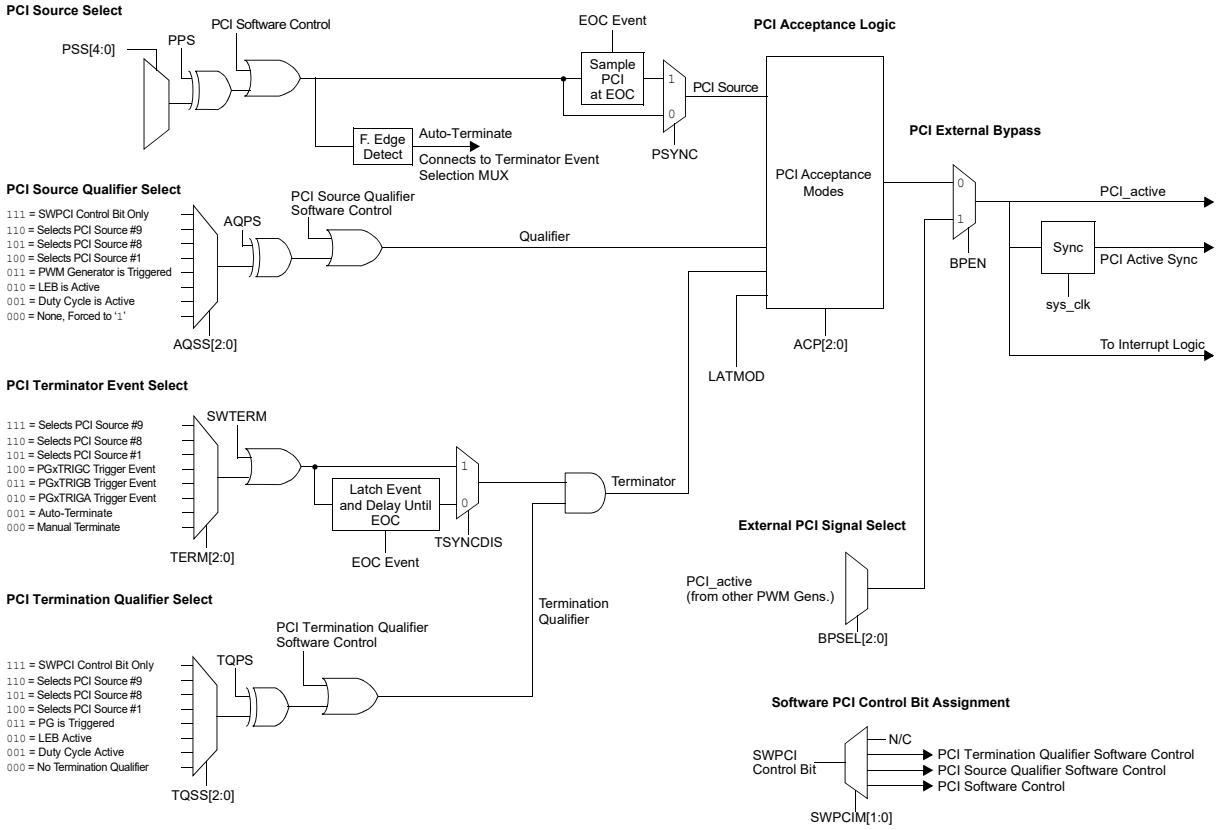
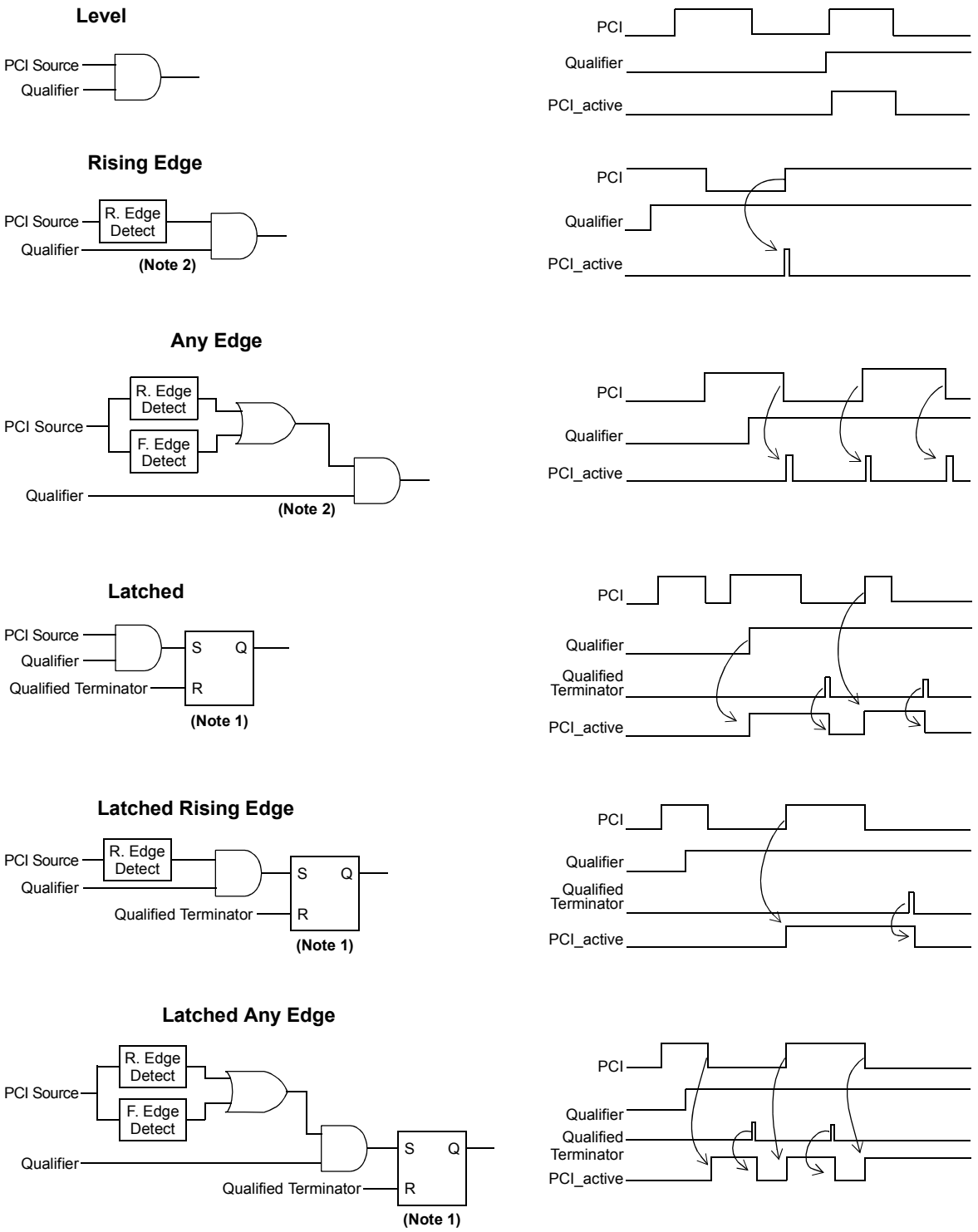


Figure 14-20. PCI Acceptance Modes



Note 1: SR latch is Set-dominant when LATMOD = 0 and Reset-dominant when LATMOD = 1.

2: Qualifier signal and edge detection is synchronized to PGx_clk.

14.4.2.5.1 Sync PCI

The main purpose of the Sync PCI block is to trigger and synchronize external events to the PWM clock domain. The synchronization can induce up to a one PWM clock delay. The Sync block is the only PCI block that can initiate a Start-of-Cycle and is available as an input to the SOCS[3:0] (PGxCON[19:16]) MUX. The Sync and Feed-Forward are the only PCI blocks that can trigger alternate dead time (duty cycle adjustment); see 14.4.2.6.1. Dead-Time Compensation for additional details.

14.4.2.5.2 Output Control PCI Blocks

The three output control type PCI blocks are provided to place the PWM outputs in a predetermined state. The blocks are prioritized in the following order, along with further details shown in 14.4.2.3.1. Complementary Output Mode and 14.4.2.3.4. Output Override in Push-Pull and Independent Modes:

1. Fault
2. Current Limit
3. Feed-Forward

The Fault PCI block has the highest priority of the PCI blocks and will dictate the state of the outputs when the block is used. A Fault condition is generally considered catastrophic and is typically cleared with software.

The Current Limit PCI block was intended for use with current limit sensing circuitry, either for protection or use as a control loop. Leading-Edge Blanking (LEB) is typically used to ignore switching transients in current sensing applications. See 14.4.2.7. Leading-Edge Blanking for details on LEB.

The Feed-Forward PCI block is intended for use as a control loop for power supply applications. If the sensing circuitry detects a rapid change in load conditions, the system can be configured to take immediate action without having to wait until the next PWM cycle to react.

Once a 'PCI active' signal is asserted, the value stored in the xDAT[1:0] bits (x = FLT, CL or FF) will be applied immediately to the pins. xDAT bits are located in the PGxIOCON register.

14.4.2.5.3 PCI Logic Description

The PCI block contains three major blocks to support a wide range of applications. First are the inputs, with logic for selecting and conditioning the input signals. Second, the PCI acceptance logic, which is the selectable logic functions applied to the inputs, and finally, output logic, including bypass.

PCI Source

The PCI source input is the main input into the PCI block and has the following features:

- Input Selection Multiplexer
- Polarity Control
- Software Control (SW override)
- Edge Detect Circuit for Auto-Terminate
- End-of-Cycle (EOC) Synchronization

The PCI source is selected using the PSS[4:0] control bits (PGxyPCI[4:0]). The polarity of the PCI input source can be selected using the PPS control bit. The chosen PCI input source may be optionally synchronized to the end of a PWM cycle using the PSYNC control bit. This synchronization is useful when a PCI signal is used to gate PWM pulses, as the PCI signal can be delayed to the next PWM boundary, ensuring that a partial pulse is not produced at the output. A falling edge detect circuit is present and can be used to automatically terminate the PCI active signal when selected by the terminator event selection multiplexer.

PCI Source Qualifier

The PCI source qualifier is a second input signal used to 'qualify' the PCI source. The PCI source qualifier is ANDed with the PCI source within the PCI acceptance logic. Inputs into the PCI source qualifier multiplexer include:

- Duty Cycle Active
- LEB Active
- PWM Generator Triggered
- PWMx Output Selected by PWMPCI[2:0] (PGxEVT[7:5])
- External Input (Another Peripheral or Device Pin)

Like the PCI source input, the PCI source qualifier input has polarity and software control. The PCI source qualifier is used in all PCU acceptance logic types; however, if it is unneeded, AQSS[2:0] (PGxyPCI[10:8]) can be set to '000' to effectively disable the qualifier.

PCI Terminator Event and Qualifier

The PCI termination event sources are used only in the Latched modes of the PCI acceptance logic functions and are used to reset the latch. Inputs to the terminator event inputs include:

- SWTERM Bit
- Trigger Events (Trigger A, B, C)
- Auto-Termination (Falling Edge Detect on PCI Source)
- PWMx Output Selected by PWMPCI[2:0] (PGxEVT[7:5])
- External Input (Another Peripheral or Device Pin)

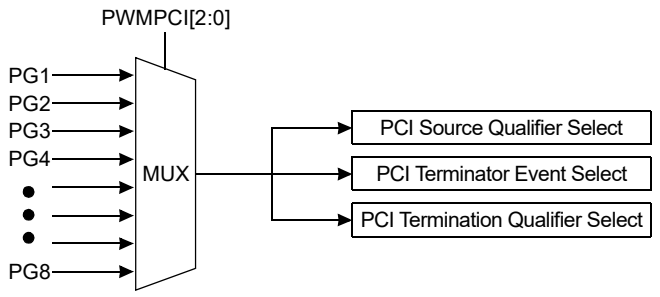
The default option for the PCI terminator is SWTERM. The SWTERM bit must be written at least two PGx_clk cycles prior to the EOC. Otherwise, the PCI termination event will be delayed until the following EOC. The PCI trigger option (Trigger A, B or C) allows the PCI logic to be reset at a particular time in the PWM cycle. User software must select the appropriate PGxTRIG to be used as the PCI trigger source. When using Automatic Termination mode, it is recommended to select 'none' as the PCI termination qualifier. An EOC synchronization is provided by default and can be disabled by setting the TSYNCDIS bit (PGxyPCI[15]).

The inputs and features of the termination qualifier are similar to the PCI source qualifier. The termination qualifier is used to create more advanced termination events.

Using a PWMx Output for PCI Function Input

The PWMPCI[2:0] control bits (PGxEVT[7:5]) are used to select which one of the eight PWM outputs can be used by the PCI block. In some control loops, it is desirable to use the output of one PWM Generator to control another generator. The selected PWMx output is made available as a selection on the PCI source qualifier select, PCI terminator event select and PCI termination qualifier select MUXes, as shown in Figure 14-21.

Figure 14-21. PWM Source Selection for PCI



14.4.2.5.4 PCI Acceptance Logic

PCI acceptance logic is the selectable logic function that is applied to the PCI inputs. The six types of available logic functions shown in Figure 14-20 are:

- Level mode: The PCI signal is passed directly through for use by the PWM Generator. The PCI signal may be optionally qualified (ANDed) with an acceptance qualifier signal.
- Rising Edge modes: The PCI signal is passed through a rising edge detection circuit that generates a pulse event. The PCI signal may be optionally qualified (ANDed) with an acceptance qualifier signal.
- Any Edge mode: The PCI signal is passed through both rising and falling edge detection circuits that generate a pulse event on either edge transition. The PCI signal may be optionally qualified (ANDed) with an acceptance qualifier signal.
- Latched mode: The PCI signal is used to set an SR latch. In this mode, a terminator signal and optional terminator qualifier are used to reset the latch. The entry into the PCI active state is asynchronously latched and possibly gated by a qualifier signal. The exit from the PCI active state is determined by a terminator signal and possibly a terminator qualifier signal. The exit from the PCI active state can also be qualified by the absence of the PCI signal itself. (This is particularly important when the Latched mode is used for Fault control applications.)
- Latched Rising Edge mode: The PCI signal is passed through a rising edge detect circuit and optionally qualified to create a pulse event. This pulse event is used to set an SR latch. The SR latch can be reset in a similar fashion to the Latched mode. The Latched Edge Detect mode allows the PCI to become active on a PCI edge event after a qualifier signal is present.
- Latched Any Edge mode: This mode is similar to Latched Rising Edge mode except that either a rising or falling edge is used to create the pulse event to set the latch.

Each mode of the PCI logic is intended to target a particular kind of power control function, although the functions can be applied to a wide variety of applications. The Level mode is useful when the PCI signals are used to affect the state of the PWM outputs asynchronously. For example, the Level mode could be used to allow an external blanking signal to force the PWM output pins to a specific state for a period of time.

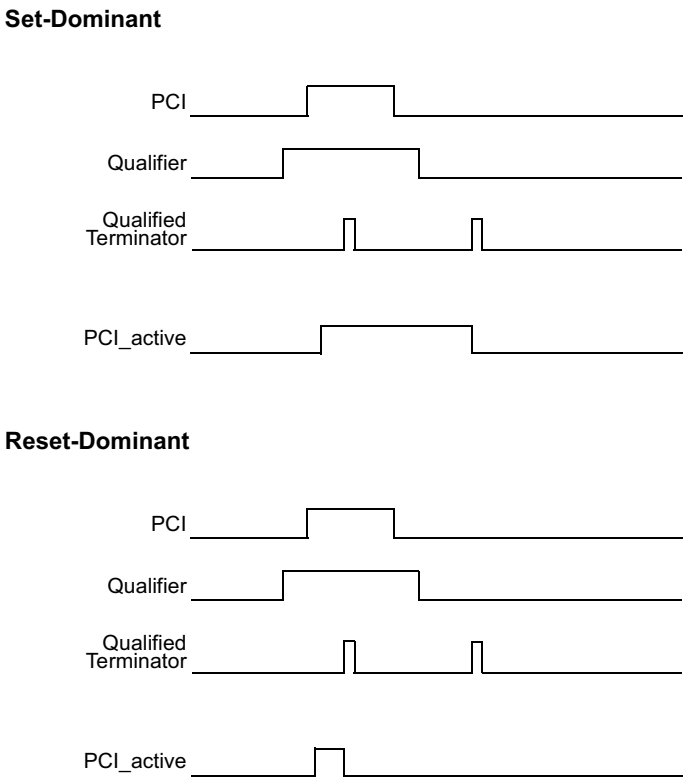
The Edge Event mode is useful when a PCI signal is used to synchronize a PWM Generator time base to an external source. When the PCI logic is used as a synchronization function, the rising edge event of the PCI signal is of primary interest. The edge event causes the PCI logic to generate an internal pulse which triggers a PWM Generator.

The Latched mode is useful for Fault and current-limiting applications. In these applications, it is important for the PCI logic to enter the active state asynchronously when qualified. The PCI logic will remain active until a selected terminating event occurs. Usually, the terminating event is a software action (manual) or the end of a PWM cycle (automatic). The Latched Edge Detect mode is useful for some types of current control applications. The PCI output cannot become active until a transition of the PCI input occurs after a qualifying condition.

Latching Mode Control

By default, the SR latch used in Latched Acceptance modes is Set-dominant. This prevents a reset of the SR latch if the PCI signal is active when the termination event signal is asserted. The LATMOD control bit (PGxyPCI[20]) can be used to create a Reset-dominant SR latch for certain PWM control functions. It is not recommended to use a Reset-dominant SR latch when the PCI logic is used to handle Fault conditions as this could allow the active state of the PCI logic to be reset while the PCI input signal is still active. Examples of Latched modes are shown in Figure 14-22.

Figure 14-22. Latch Mode Control



14.4.2.5.5 PCI External Bypass

An option to use the PCI output of another PWM Generator is possible using the PCI external bypass feature. The PCI bypass function is useful when auxiliary, client or combinatorial PWM Generators require PCI functions based on the host PWM Generator's timing. The local PCI logic can be bypassed, using instead the output of the PCI block from another PWM Generator. Only the same type of PCI (Fault, Overcurrent, Sync or Feed-Forward) block can be utilized from another PWM Generator. When BPEN = 1, the PWM Generator specified by the BPSEL[2:0] bits (PGxyPCI[30:28]) provides the PCI control. The override states of the FLTDAT[1:0], CLDAT[1:0] and FFDAT[1:0] control bits are not affected when BPEN = 1. PWM pin override states are always determined by local control bits.

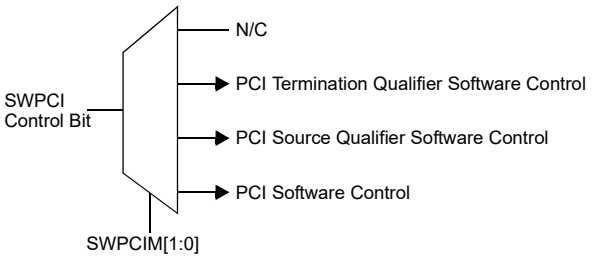
14.4.2.5.6 Software PCI Control

All PCI blocks have provisions for software control to force and clear events or for development debugging. There are three controls that can be used to manually control the PCI inputs:

- SWPCI Control Bit
- SWPCIM Demultiplexer (for SWPCI Control Bit)
- SWTERM to Generate a Termination Event

The SWPCI control bit (PGxyPCI[23]) can have its programmed state of '0' or '1' routed to one of three destinations, specified by the SWPCIM[1:0] bits (PGxyPCI[22:21]), as shown in Figure 14-23.

Figure 14-23. Software PCI Control Bit Assignment

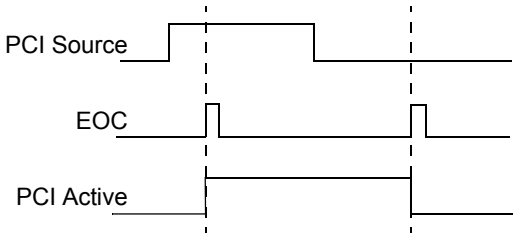


The SWTERM bit is tied directly to the terminator event input logic, and it can be used to manually terminate PCI events by writing a '1' to SWTERM and having the TERM[2:0] bits selection set to '000'. Additionally, the acceptance and terminator qualifier input multiplexers have an option to output a fixed state of '1', or, when used with their respective polarity control, a fixed state of '0'. These fixed states can be used for debugging or when the acceptance function is not needed.

PCI Source EOC, Level Mode

When the PCI acceptance logic is operated in Level mode and the PCI source is synchronized to the EOC event, there is no logic that retains the state of the prior PCI source signal. Therefore, the resultant PCI output is simply the PCI source signal synchronized to the EOC event. This configuration is useful for PWM chopping applications where the PCI source signal is used as a gating signal. The gating signal is automatically aligned to the PWM cycle boundaries, as shown in Figure 14-24.

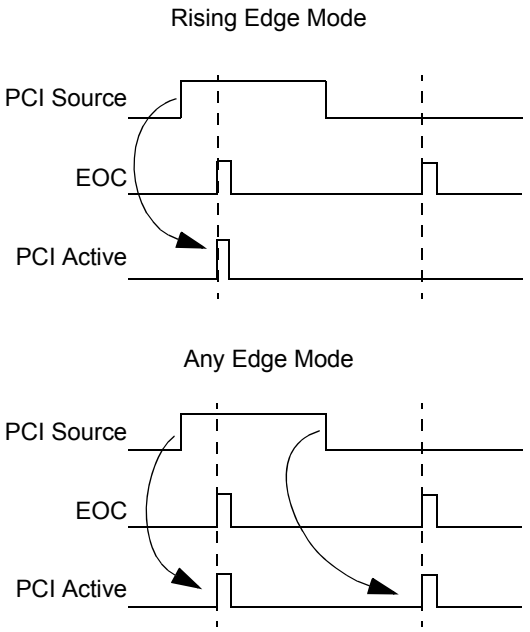
Figure 14-24. PCI Source EOC Sync, Level Acceptance Mode



PCI Source EOC, Edge Modes

When the PCI acceptance logic is operated in the Rising Edge or Any Edge modes and PSYNC = 1, the PCI source is synchronized to the EOC event, as shown in Figure 14-25. If an edge event is detected, the pulse is delayed until the next EOC event. In the case that a PCI source signal becomes active and then inactive within a single PWM cycle, the PCI active signal will not assert.

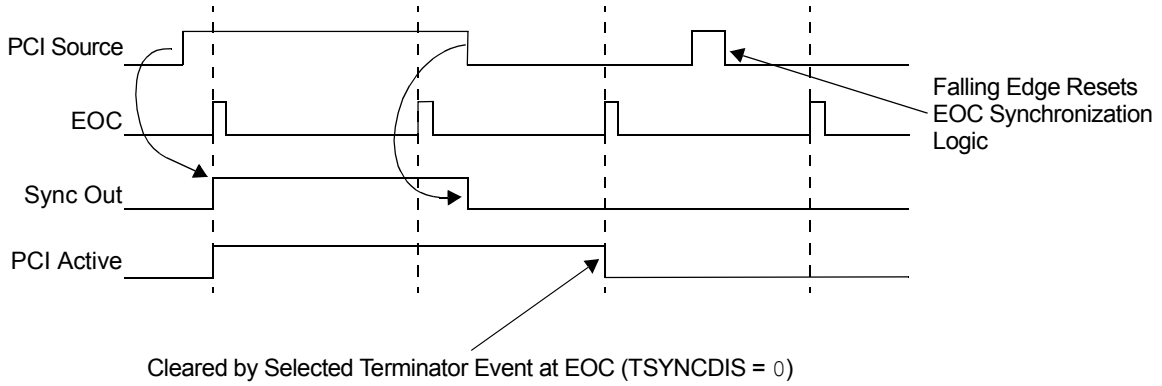
Figure 14-25. PCI Source EOC Sync, Edge Acceptance Modes



PCI Source EOC, Latched Modes

When the PCI acceptance logic is operated in the Latched mode and PSYNC = 1, the PCI source is synchronized to the EOC event, as shown in Figure 14-26. The synchronization logic delays the rising edge of the PCI source signal until the next occurrence of the EOC signal. The output of the synchronization logic is deasserted on the falling edge of the PCI source signal. The output of the synchronization logic is then used to set the SR latch. A PCI input pulse that operates entirely within one EOC period will not assert the PCI active signal. This is because the falling edge of the PCI input signal resets the EOC synchronization logic before an event can be produced.

Figure 14-26. PCI Source EOC Sync, Latched Acceptance Mode

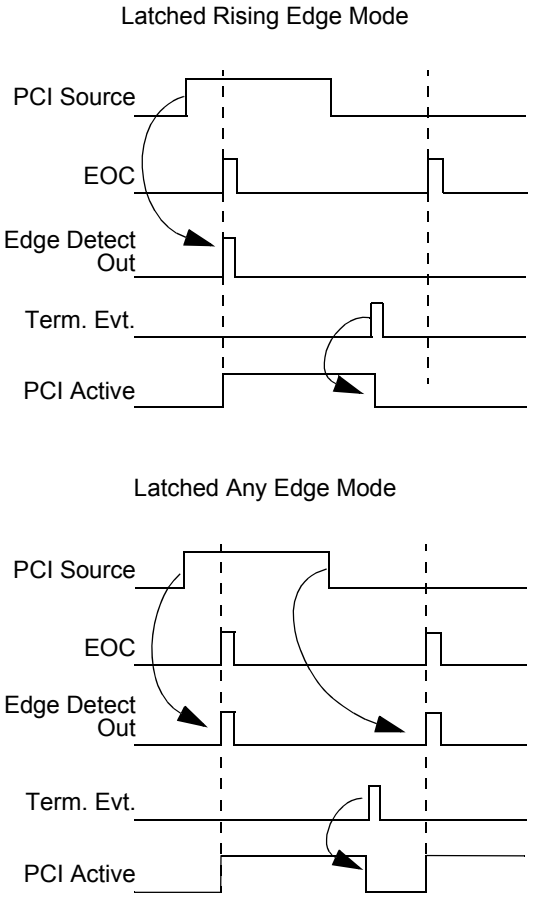


PCI Source EOC, Latched Edge Modes

When the PCI acceptance logic is operated in the Latched Edge modes and PSYNC = 1, the PCI source is synchronized to the EOC event, as shown in Figure 14-27. This configuration operates similar to the Rising Edge and Any Edge modes, except that the event output of the synchronization

logic is latched. A PCI source input pulse that operates entirely within a PWM cycle will assert the PCI active signal.

Figure 14-27. PCI Source EOC Sync, Latched Edge Acceptance Modes

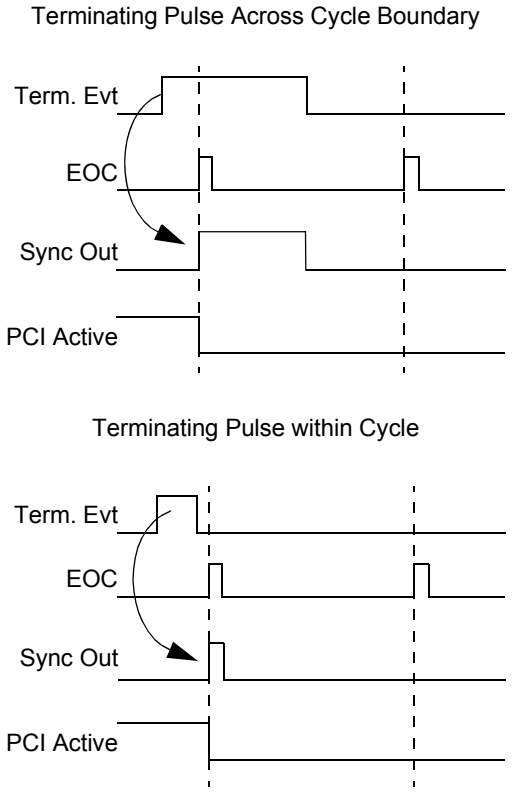


Note: These timing diagrams assume TSYNCDIS = `1`; therefore, the termination event takes effect immediately.

PCI Terminator EOC

By default, the PCI logic synchronizes a terminator event to the PWM EOC. This allows the PWM to resume cleanly at the start of a new cycle. The rising edge of the terminating signal is held off until an occurrence of the EOC event. The terminator signal is usually a pulse event used to reset the latched state of the PCI logic. If a short pulse is received prior to the occurrence of an EOC event, a Reset pulse is produced at the next EOC event. If the terminator signal is a longer pulse, the synchronized output is held active for as long as the terminator signal is present. This behavior can be used to force the PCI logic to a Reset state, if desired. Terminator event synchronization timing is shown in Figure 14-28.

Figure 14-28. PCI Terminator EOC Synchronization

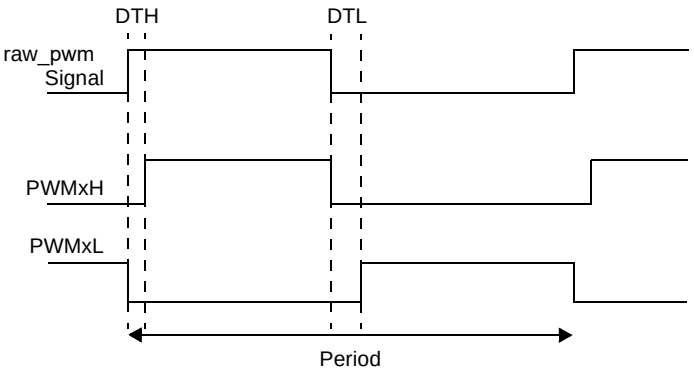


14.4.2.6 Dead Time

The dead-time feature is used to provide a time period where neither complementary outputs are active at the same time. Dead time is used to prevent both output driver devices (switches) in a bridge from conducting at the same time, causing excessive current flow. Since output switch turn-on and turn-off times are non-instantaneous, dead time is set to ensure that only one device is active. Dead time is implemented by holding off the assertion of the Active state. For the PWMxH and PWMxL outputs, this will delay the rising edge, as shown in Figure 14-29.

Dead-time duration is configured using the PGxDT register. The PGxDT register holds a pair of up to 15-bit dead-time values, DTH and DTL, that are applied independently to the PWMxH and PWMxL outputs, respectively. Dead time is typically only used in Complementary Output mode.

Figure 14-29. PWMxH/PWMxL Rising and Falling Edges Due to Dead Time

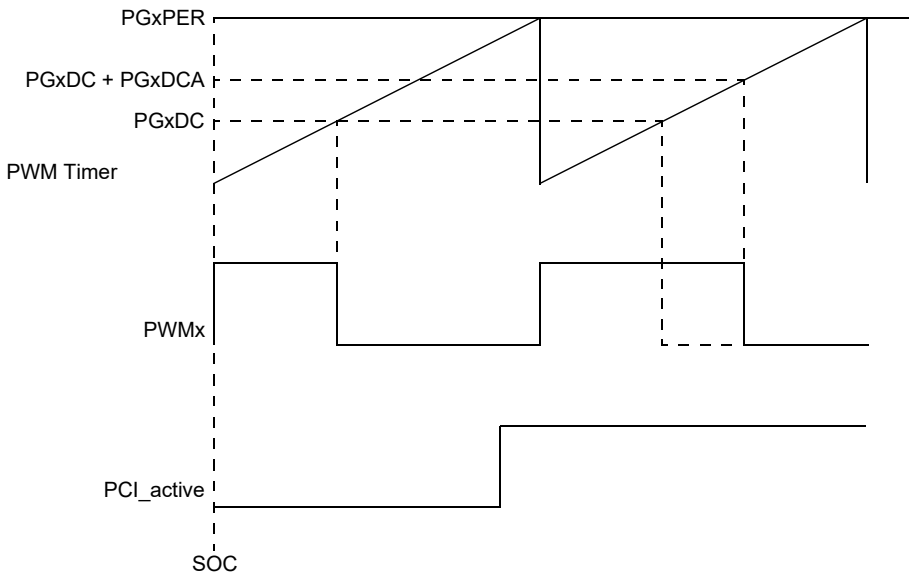


14.4.2.6.1 Dead-Time Compensation

The dead-time compensation feature allows the duty cycle to be selectively controlled by a PCI input. Dead-time compensation is enabled by writing a non-zero value to the PGxDCA (PWM Generator x Duty Cycle Adjustment) register and setting up PCI logic to control the compensation adjustment. When active, the PGxDCA value will be added to the value in the PGxDC register to create the effective duty cycle, as shown in Figure 14-30.

The DTCMPSEL control bit (PGxIOCON[24]) selects the PCI Logic block to be used for dead-time compensation, which can either be the Feed-Forward or Sync PCI blocks. If the PGxDCA register is '0', the dead-time compensation function is disabled regardless of the DTCMPSEL value. The dead-time compensation input signal from the PCI logic is sampled at the end of a PWM cycle for use in the next PWM cycle. The modification of the duty cycle duration via the PGxDCA registers occurs during the end (trailing edge) of the duty cycle.

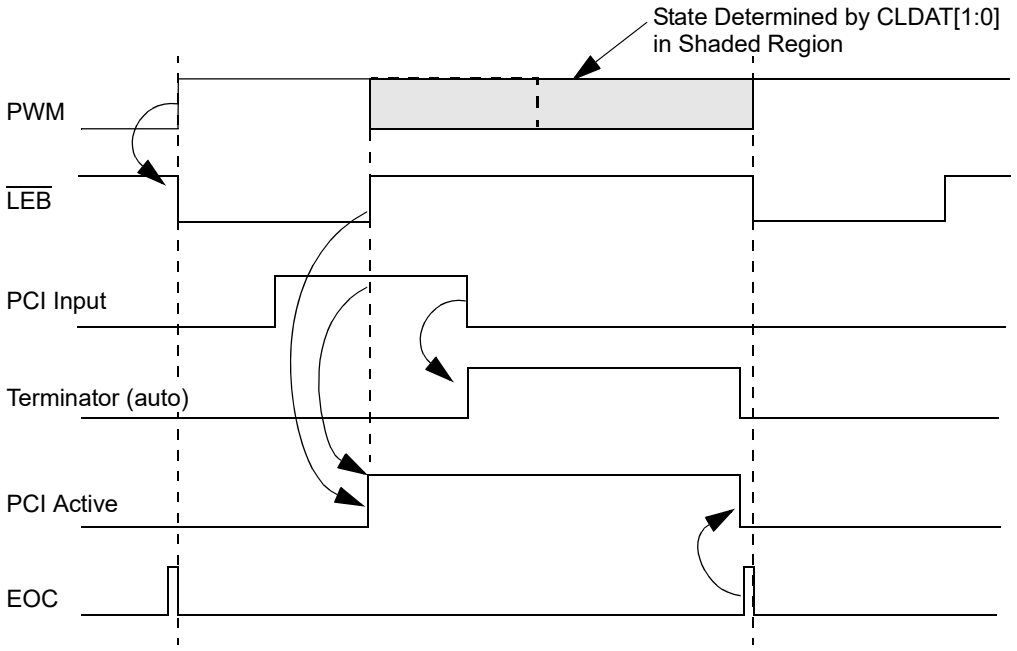
Figure 14-30. Adding PGxDCA Value to the PGxDC Register Value



14.4.2.7 Leading-Edge Blanking

The Leading-Edge Blanking (LEB) feature is used to mask transients that could otherwise cause an erroneous Fault condition. Leading-Edge Blanking can be implemented using any of the PCI blocks and basically 'ignores' an input signal for a specified time following a PWM edge event.

Figure 14-31. Leading-Edge Blanking (LEB)



Both the rising and falling edges of both PWMxH and PWMxL signals can be selected to start the LEB timer. The LEB time duration is set by writing a value to the LEB bits (PGxLEBL[15:3]). More than one edge (PHR, PHF, PLR or PLF; refer to the 14.3.3.9. PGxLEB register) may be used; however, if timing overlaps, the counter will be reset on each valid edge. In most applications, only one edge of the PWM signal needs to be selected to trigger the LEB timer.

The LEB counter is commonly used to avoid a false trip when the PCI logic is used for current limiting. In this scenario, the LEB counter can be triggered on both edges of the PWM signal. The PCI logic is operated in Latched Acceptance mode with the LEB active signal used as a disqualifier to the PCI input signal. Figure 14-31 shows a PWM cycle where a PCI input goes active during the LEB timer. There is no PCI active event or output override until the LEB timer has expired.

14.4.2.7.1 Leading-Edge Blanking Counter Period Calculation

The LEB counter value is stored in the PGxLEB[15:4] bits and defines the period of the counter (T_{LEB}). The lower four bits are read-only and always read as '0', yielding a minimum LEB resolution of 16 PGx_clks. The minimum blanking period is five PGx_clks, which occurs when LEB[15:0] ≤ 4.

Equation 14-1. Leading-Edge Blanking Period

$$T_{LEB} = \frac{16 * (LEB[15:4] + 1)}{F_{PWM}}$$

$$LEB[15:4] = \left(\frac{F_{PWM} * T_{LEB}}{16}\right) - 1$$

Where:
T_{LEB} is specified in time units (ms, µs or ns)

14.4.2.7.2 Leading-Edge Blanking PCI Configuration

The LEB counter produces an “LEB active” signal that is supplied to the acceptance qualifier and/or termination qualifier selection multiplexers of the PCI blocks. This allows the LEB timer to be used as a gating signal for the selected PCI or PCI terminator signal. The polarity of the acceptance qualifier and termination qualifier signals can be inverted using the PCI control bits, so that the “LEB active” signal is changed to “LEB inactive”. It is recommended that the Latched PCI Acceptance mode be used when using LEB, so that the LEB active signal can only affect entry or exit to/from the PCI active state. Auto-termination can also be used to ‘reset’ the system after the Fault condition has cleared.

Example LEB initialization sequence:

1. Select the type PCI to use for LEB (y = CL, FF, Fault).
2. Select the PCI input using the PSS[4:0] (PGxyPCI[4:0]) bits. This is typically connected to a comparator to sense overcurrent.
3. Select LEB as the acceptance qualifier using the AQSS[2:0] (PGxyPCI[10:8]) bits.
4. Invert the acceptance qualifier using the AQPS (PGxyPCI[11]) bit.
5. Configure logic for Latched mode using the ACP[2:0] (PGxyPCI[26:24]) bits.
6. Select auto-terminate using the TREM[2:0] (PGxyPCI[14:12]) bits.

14.4.2.8 Override

The override feature can be used to take control of the PWM outputs and force certain conditions onto the pins. User software can override the output states of the pins by writing a ‘1’ to the OVRENH and OVRENL control bits located in the PGxIOCON register. The state of the pins, when overridden, will be that of the value written to OVRDAT[1:0], unless it conflicts with restrictions imposed by the given output mode. Most constraints are in Complementary mode and are discussed in 14.4.2.3.1. Complementary Output Mode.

The OVRDAT[1:0] and OVRENH/L control bits are double-buffered for flexibility. The OSYNC[1:0] control bits in the PGxIOCON register specify when the user override values are applied to the PWM outputs. Manual software overrides can be applied at the following times:

- At the start of a new PWM cycle
- Immediately (or as soon as possible)
- As configured by the UPDMOD[2:0] control bits (see 14.4.2.10. Data Buffering).

Synchronizing overrides on multiple PG is done by enabling one of the Host/Client modes as defined by UPDMOD bits.

Like data updates, overrides are applied by a write of the UPDREQ. When UPDATE = 0, the user software may write new values to OVRENx or OVRDAT to set the UPDREQ bit. Setting the UPDREQ bit ‘commits’ the new values to the PWM Generator. The UPDATE bit will set and writes are prohibited. When the process is complete the UPDATE bit is cleared by hardware.

In general, SOC update modes are recommend as they allow scheduling of override updates with an interrupt, which provides the maximum time to complete the next write. Otherwise the UPDATE bit should be polled to write at a safe time.

Override update process:

1. Configure OSYNC and UPDMOD as needed for application
2. Configure PWM interrupt for SOC (default)
3. When interrupt event occurs:
   a. Read UPDATE bit and verify it is '0'
   b. Write new values to OVRENx or OVRDAT
   c. Set UPDREQ bit to a '1' to commit data to PWM buffer

Synchronizing Multiple PWM Generator override updates:

1. Configure host PG for SOC updates by writing OSYNC = '0' and MSTEN = '1'
2. Configure client(s) PG for 'Client SOC' mode by writing OSYNC = 0b10 and UPDMOD = 0b0101
3. Configure host PWM interrupt for SOC (default)
4. When interrupt event occurs:
   a. Read host UPDATE bit and verify it is '0'
   b. Host write new data values to OVRENx or OVRDAT for all PWM generators (PGx)
   c. Set host UPDREQ bit to a '1' to commit data to PWM buffer

On the next SOC, all PWM generators configured as clients will operate with new data values.

14.4.2.9 Event Selection Block

Each PWM Generator has a logic block for events, triggers and interrupts. These signals are then used by the Event Output block (see 14.4.3.6. PWM Event Outputs) or as the trigger source to start a PWM cycle (SOCS[3:0]). The Event Selection block has three main functions:

- ADC Trigger Configuration
- PWM Generator Trigger
- Interrupts

14.4.2.9.1 ADC Triggers

Each PWM Generator has the capability to trigger multiple ADCs, either internal or external to the device. ADC triggers are based on the TRIGA, TRIGB and TRIGC compare events for timing within the PWM cycle. The ADC triggers are also used as triggers for other peripherals and functions, such as the PTG, DAC and interrupts. The ADC triggers are also made available externally through the Event Output block (see 14.4.3.6. PWM Event Outputs) or internally in conjunction with the CPU interrupt controller.

Multiple TRIGx sources may be enabled to create the ADC trigger output, and when enabled, they are logically OR'd together. If the multiple TRIGx registers are enabled to produce ADC trigger events, they must be configured to allow unique trigger events to the ADC.

Each PWM Generator can generate two ADC triggers: ADC Trigger 1 and ADC Trigger 2. The two trigger outputs are useful for SMPS applications, where it is often desirable to measure two quantities in a single cycle. Each trigger is connected to a separate ADC, or possibly, a separate ADC trigger input. The ADC Trigger 1 output has an additional offset and postscaler function to allow these functions:

- Postscaler, to reduce the frequency of ADC trigger events.
- Offset, a one-time offset may be applied to ADC trigger events. This allows postscaled trigger events to be interleaved with trigger events from other PWM Generators.

Trigger events from ADC Trigger 2 will be produced every PWM cycle. ADC Trigger 1 output may be postscaled using the ADTR1PS[4:0] control bits (PGxEVT[15:11]) to reduce the frequency of ADC conversions. In addition, the ADC Trigger 1 output can be offset by a certain number of trigger events using the ADTR1OFS[4:0] control bits (PGxEVT[20:16]). Together, these two sets of control bits allow the user to establish an interleaved set of ADC triggers from multiple PWM Generators. In addition, ADC trigger events may be simply postscaled to reduce the frequency of ADC measurements. If the ADTR1PS[4:0] control bits are set to '00000', an ADC trigger event will be produced on every PWM cycle. When these control bits are set to a non-zero value, an ADC trigger will be produced during the PWM cycle after the ON bit is set and every N cycle thereafter. The ADTR1OFS[4:0] bits value establishes a one-time offset of 0 to 15 trigger events after the ON bit has been set. After this offset has been established, the trigger postscaler will begin to count the number of trigger events determined by the ADTR1PS[4:0] bits value. When interleaving ADC triggers from multiple PWM Generators, all PWM Generators should be programmed to have the same period to ensure consistent spacing between the trigger events.

14.4.2.9.2 PWM Generator Trigger Output

One of the PWM Generator internal events may be selected to drive the PWM Generator trigger output. The PWM Generator trigger output signal is selected using the PGTRGSEL[2:0] bits (PGxEVTL[2:0]) with the selection being either EOC or one of the three TRIGx compare events. Using one of the TRIGx events as an SOC trigger for another PWM Generator is useful for implementing a variable phase PWM. The phase relationship between two different PWM Generators can be controlled by the value written to the TRIGx registers.

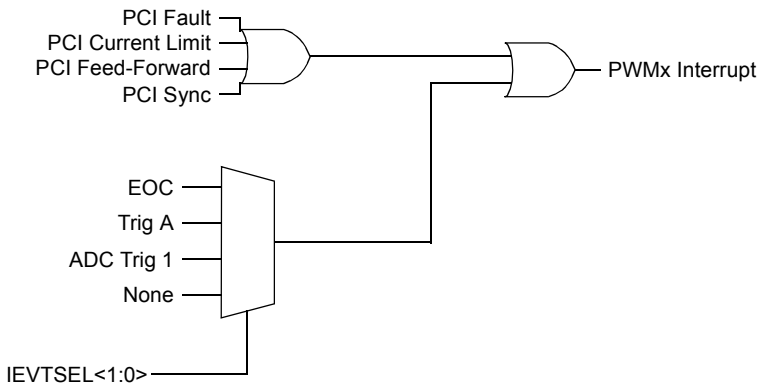
14.4.2.9.3 Event Interrupts

The PWM event that causes a CPU interrupt is programmable for flexibility. The IEVTSEL[1:0] control bits (PGxEVT[25:24]) allow the user to select one of the following:

- EOC (default)
- TRIGA Compare Event
- ADC Trigger 1 Event
- None (disabled)

The Event Selection block also contains interrupt enables for each of the four PCI blocks. The SIEN, FFIEN, CLIEN and FLTIEN bits in the PGxEVT register are used to independently enable interrupts for their respective PCI block. When IEVTSEL[1:0] are set to disabled, the PCI interrupts can still be used independently.

Figure 14-32. Event Selection Block



14.4.2.10 Data Buffering

The PWM module allows for certain SFR data values to be buffered and applied to the PWM output at later events. The following user registers and/or bits are buffered, allowing the user to modify data while the PWM Generator operates on the previous set of data values:

- PGxPER
- PGxPHASE
- PGxDC
- PGxTRIGA
- PGxTRIGB
- PGxTRIGC
- PGxDT
- SWAP
- OVRDAT[1:0] (software output override values)

- OVREN (software output override enables)

Data are transferred from the SFR registers to the internal PWM registers at the start of a PWM cycle. This can be every one, two or four timer cycles, depending on the PWM Generator mode and the Output mode. It may be required that a register be updated immediately to produce an immediate change in a power converter operation. In other cases, it may be desirable to hold off the buffer update until some external event occurs where data coherency between multiple PWM Generators is of concern. The module supports the user to specify when the contents of the SFRs associated with a PWM Generator are transferred into the “active” internal registers. Available options are:

- Immediately
- At the beginning of the next PWM cycle
- As part of a larger group

The UPMOD[2:0] control bits in the PGxCON register determine the operating mode for register updates. The UPDATE status bit in the PGxSTAT register allows visibility to when register updates are complete and changes may be applied. When UPDATE = 0, the user software may write new values to the PWM Data registers and set the UPDREQ bit when done. Setting the UPDREQ bit ‘commits’ the new values to the PWM Generator, and user software can not modify PWM data values until the bit is cleared by hardware.

In general, SOC update modes are recommended as they allow scheduling of override updates with an interrupt, which provides the maximum time to complete the next write. Otherwise the UPDATE bit should be polled to write at a safe time.

Data update process:

1. Configure UPDMOD as needed for application
2. Configure PWM interrupt for SOC (default)
3. When interrupt event occurs:
   a. Read UPDATE bit and verify it is '0'
   b. Write new data value(s)
   c. Set UPDREQ bit to a '1' to commit data to PWM buffer

On the next SOC, the PWM generator will operate with new data values.

In order to avoid extra CPU cycles, the data updates can be configured to be automatically performed on a write to one of the PWM Data registers. The register is selected using the UPDTRG[1:0] bits in the PGxEVT control register. The default selection is that the UPDREQ bit must be manually set in software. A write to the PGxDC register can trigger an update, since many applications frequently change the duty cycle of the PWM. The PGxPHASE and PGxTRIGA registers may also be chosen as update triggers. These registers may be modified on a frequent basis in variable phase applications. The register that is selected as the update trigger must be the last one to be written if several PWM Data registers are to be updated. The PWM Data registers should not be modified once the UPDATE bit becomes set. User software must wait for the PWM hardware to clear the UPDATE bit before the Data registers can be modified again.

14.4.2.10.1 Synchronizing Multiple PWM Generator Buffer Updates

The MSTEN control bit (PGxCON[27]) allows the PWM Generator to control register updates in other PWM Generators. The UPDREQ control and UPDATE status bits can be effectively broadcast to other PWM Generators to allow coherent register updates among a set of PWM Generators that control a common function. When MSTEN is set and the user software (or the PWM Generator hardware) sets the UPDREQ control bit, this event will be broadcast to all other PWM Generators. If UPDMOD[2:0] = 01x in a PWM Generator that receives the request, the receiving module will set its local UPDREQ bit. The local UPDATE status bit will then be cleared when the local registers have been updated. The user software may set a local UPDREQ bit manually.

Table 14-10. PWM Data Register Update Modes

UPDMOD[2:0]	Mode	Description
000	SOC	Update Data registers at start of next PWM cycle if UPDREQ = 1. The UPDATE status bit will be cleared automatically after the update occurs.(1)
001	Immediate	Update Data registers immediately, or as soon as possible, if UPDREQ = 1. The UPDATE status bit will be cleared automatically after the update occurs.
010	Client SOC	Update Data registers at start of next cycle if a host update request is received. A host update request will be transmitted if MSTEN = 1 and UPDREQ = 1 for the requesting PWM Generator.
011	Client Immediate	Update Data registers immediately, or as soon as possible, when a host update request is received. A host update request will be transmitted if MSTEN = 1 and UPDREQ = 1 for the requesting PWM Generator.

Note:

1. The UPDREQ bit must be set at least three sys_clk cycles, followed by three PGx_clk cycles, followed by another three sys_clk cycles, before the next PWM cycle boundary in order to take effect. Otherwise, the data update will be delayed until the following PWM cycle.

Synchronizing Multiple PWM Generator data updates:

1. Configure host PG for SOC updates and write MSTEN = '1'
2. Configure client(s) PG for 'Client SOC' mode by writing UPDMOD = 0b0101
3. Configure host PWM interrupt for SOC (default)
4. When interrupt event occurs:
   a. Read host UPDATE bit and verify it is '0'
   b. Write new data values to all PWM generators (PGx)
   c. Set host UPDREQ bit to a '1' to commit data to PWM buffer

On the next SOC, all PWM generators configured as clients will operate with new data values.

For the purpose of Data register updates, a PWM cycle length is variable. A PWM cycle may comprise one, two or four timer cycles, depending on the PWM operating mode and the Output mode that is selected. The PWM Data registers may be updated on the next, second or fourth timer cycle when a SOC update has been requested. Table 14-11 summarizes the number of timer cycles between each SOC update vs. the PWM Generator operating mode and the Output mode. For additional information on the timing of update events, refer to the chapter pertaining to the selected PWM mode.

Table 14-11. Timer Cycles per Data Register Update

PWM Mode	Output Mode	Timer Cycles per PWM Cycle	Timer Cycles per Interrupt and Data Register Update
Independent Edge, Dual PWM or Variable Phase	Independent Output, Complementary	1	1
Independent Edge, Dual PWM or Variable Phase	Push-Pull	2	2
Center-Aligned	Independent Output, Complementary	2	2
Center-Aligned	Push-Pull	4	4
Double Update Center-Aligned or Dual Edge Center-Aligned	Independent Output, Complementary	2	1
Double Update Center-Aligned or Dual Edge Center-Aligned	Push-Pull	4	1

14.4.2.10.2 Immediate Updates

When using Immediate Update mode, there may be latency from the time of commanding a change, to it getting applied. This mode applies changes as soon as possible to prevent unexpected results.

Note: Avoid immediate updates near new or existing edge values to prevent data corruption. When using immediate updates, write timing should be tested in the software to avoid PWM edges.

Immediate update of period value updates to the PGxPER value become effective instantaneously. Care should be taken when the PWM period is shortened. If the PWM time base has already counted beyond the new (shorter) PWM period value, a long period will result as the counter must now count to 0xFFFF and then roll over. If immediate updates are required, the best practice is to capture the time base value prior to the period update so a safe minimum period value may be calculated and written.

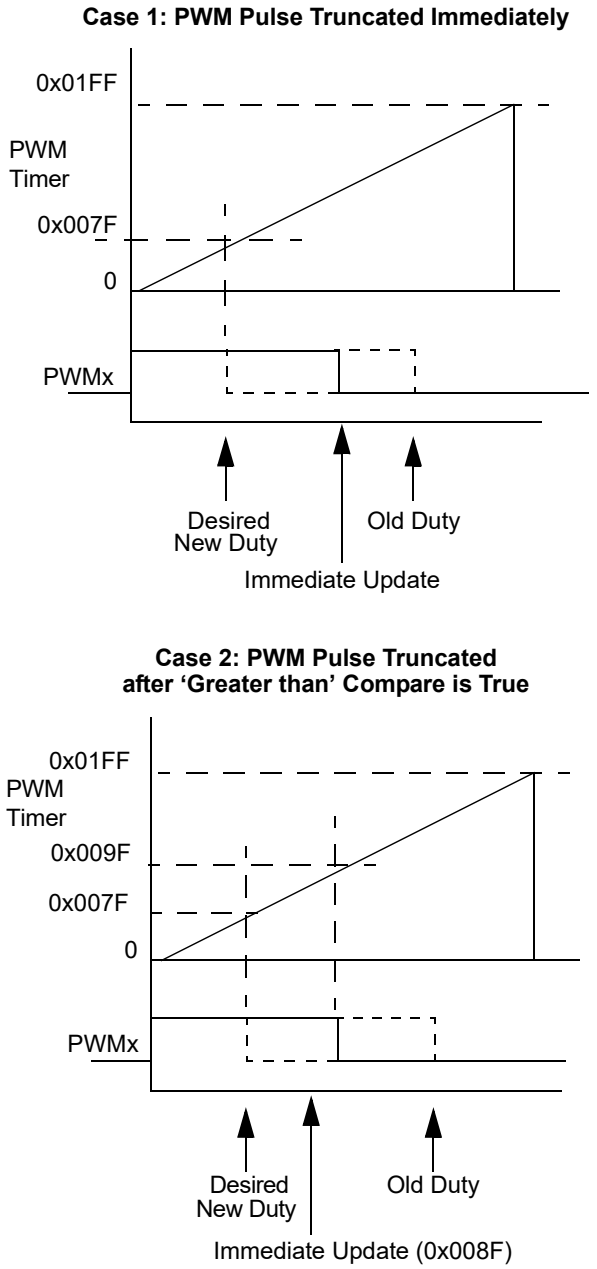
Immediate Updates to Duty Cycle, Phase Offset

The immediate update correction logic keeps track of whether a PWM pulse has already been created in the current PWM cycle and will not restart a second pulse within the same cycle. For example, if an immediate update changes the PGxPHASE value to a later time in the cycle, but a rising edge and falling edge have already been generated, then the new value of PGxPHASE will not take effect until the next cycle. An immediate update of PGxPHASE can only be used to delay a PWM pulse which has not occurred yet or cause the pulse to start sooner.

Figure 14-33 shows two examples of a correction during immediate updates. In this update mode, the PWM module implements a compare function that compares the upper 12-bits of the data to the time base. This logic in hardware takes action as needed for the given conditions. The PWM period is relatively short in these examples and a large duty cycle adjustment is made to emphasize how the correction works. In both examples, the duty cycle is decreased from 75% to 25% (0x7F) at approximately the mid-point of the PWM cycle. In both cases, the time base has already elapsed beyond the compare time for 25% duty.

In the first case, the immediate update write occurs at approximately 55% duty cycle. The PWM pulse is truncated immediately because the PWM time base is at least 0x8F. The programmed duty cycle is 0x7F, so the value of 0x80 provides a true ‘greater than’ comparison when compared to 0x7F. In the second case, the immediate update write occurs just beyond the time of the newly programmed duty cycle. The PWM pulse is not truncated until the time base reaches a value of 0x0080 and the ‘greater than’ comparison becomes true.

Figure 14-33. Immediate Update Correction Examples



Immediate Updates to PER

Using Immediate Update mode with PER data has some inherent hazards and should be avoided unless required by application. If used, user software must manage write time in relation to the PWM counter and known or planned event timing. The time base capture feature can be used for this purpose.

For example, it is possible to write a smaller period after both rising and falling edges have occurred. The PER compare will not occur in the cycle and PWM counter will roll over at 0xFFFF. Due to no EOC event, the subsequent cycle will have no rising or falling edges. This can be avoided by reading time base capture and gating update if new PGxPER < PGxCAP.

Immediate Update to Dead Time

If a DT blanking is in progress and an immediate update to the DT occurs, the actual dead time after the update will be extended. This extension is due to the DT counter being reloaded before it has expired. Future dead-time delays after the immediate update will be the new time, as expected.

14.4.2.10.3 Time Base Capture

A time base capture feature is provided as the PWM timer itself is not directly readable. When the timer value is needed, it may be captured and read via the PGxCAP register. There are two methods to capture a value: either manually with software or with hardware on a PCI event. The CAPSRC[2:0] control bits (PGxIOCON[30:28]) are used to select either a manual capture or one of the four PCI blocks as the trigger for a time base capture.

To manually capture the timer value, write a '1' to PGxCAP[0]. The CAP status bit (PGxSTAT[5]) will set to indicate the capture is complete and then the user may read the PGxCAP register to determine the time base value at the time of the hardware event. A read operation of PGxCAP will clear the CAP status bit. No further captures are allowed until user software reads the PGxCAP register. Similarly, when a PCI block is used to capture a time base event, a read operation is needed to reset the logic to allow a subsequent capture event. It is recommended to read the CAP status bit to verify it is set before reading PGxCAP. This is to avoid a read of the PGxCAP register at the same time as the PWM hardware is writing it. An alternative method is to schedule reads with an interrupt to avoid concurrent access.

There will be up to four time base clock cycles of latency between the time of the actual event that caused the capture and the actual time base value that is captured. This delay is due to synchronization and sampling delays.

Time base capture example:

1. Read the CAP status bit and verify CAP is '0' (no pending capture).
2. Initiate capture event (SW or PCI).
3. Poll the CAP status bit and wait for it to set to indicate data are ready.

14.4.2.10.4 Operation in Debug Mode

When halting program flow using the debugger, the PWMx output pins can be left in a state that may be harmful to the hardware. To avoid this, logic is included to force the pins to a predetermined state, defined by the DBDAT[1:0] bits (PGxIOCON[1:0]). The pin states are still subject to the priority of overrides.

14.4.3 Common Features

14.4.3.1 Master Data Registers

The PWM module has a set of common Data registers that can be optionally assigned to multiple PWM Generators:

- MDC: Master Duty Cycle register
- MPER: Master Period register
- MPHASE: Master Phase register

These master registers allow user software to affect the operation of multiple PWM Generators by writing one Data register. The MDCSEL, MPERSEL and MPHSEL control bits in each PGxCON register determine whether the PWM Generator will use the local Data registers or the Master Data registers.

14.4.3.2 LFSR – Linear Feedback Shift Register

The Linear Feedback Shift register (LFSR) is a pseudorandom number generator that provides 15-bit values that can be used in applications to modify either the duty cycle and/or period by a small amount to dither the corresponding switching edges of the application circuit's power transistors. This dithering can be useful in reducing peak EMI (Electromagnetic Interference) emissions.

Each read of the LFSR register will result in a new update of the LFSR value. The LFSR initializes at a Power-on Reset to 0x0000, and for successive reads it follows the deterministic sequence shown in Table 14-12. It has the equivalent circuit, as shown in Figure 14-34, which implements a Fibonacci form LFSR based on the primitive polynomial: $x^{15} + x^{14} + 1$ over GF(2). The circuit is modified by a Zero-Detect circuit that causes the 0x0000 value to be followed by 0x0001. Subsequent reads of the LFSR will cycle through all 15-bit values, other than 0x0000, before repeating. The high bit of the LFSR output is always '0'.

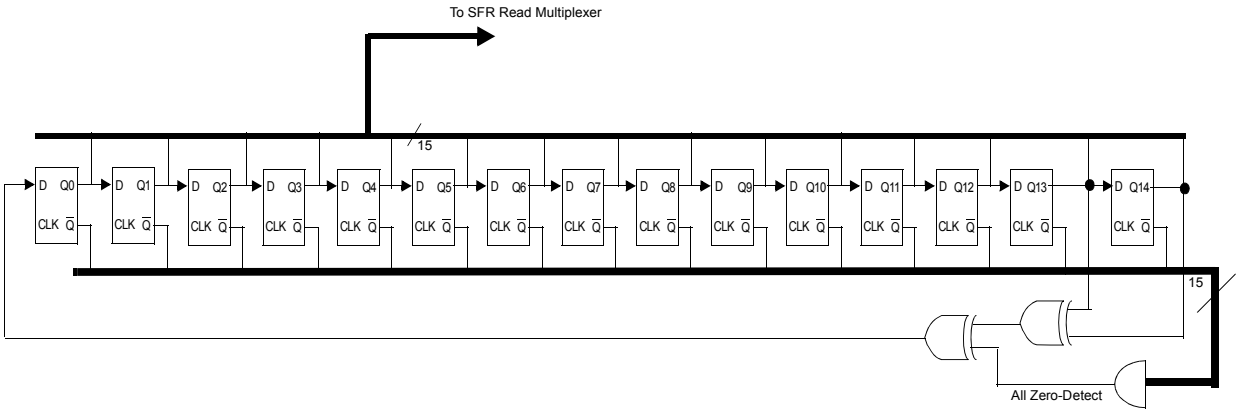
If the same LFSR value is to be used for multiple calculations, then the value read from the LFSR register should be saved in a temporary location for this purpose.

Successive readings of one particular bit of the LFSR forms a Pseudonoise (PN) sequence with an impulse auto-correlation function (shifted versions of the PN sequence are uncorrelated) and may be useful for dithering applications. The entire 15-bit LFSR value has auto-correlation properties that may be undesirable in some applications as a source of pseudorandom noise; its use should be validated in the end application.

Table 14-12. LFSR Successive Read Sequence

0x0000, 0x0001, 0x0002, 0x0004, 0x0008, 0x0010, 0x0020, 0x0040, 0x0080, 0x0100, 0x0200, 0x0400, 0x0800, 0x1000, 0x2000, 0x4001, 0x0003, 0x0006, 0x000c, 0x0018, 0x0030, 0x0060, 0x00c0, 0x0180, 0x0300, 0x0600, 0x0c00, 0x1800, 0x3000, 0x6001, 0x4002, 0x0005, 0x000a, 0x0014, 0x0028, 0x0050, 0x00a0, 0x0140, 0x0280, ...,
0x5557, 0x2AAF, 0x555F, 0x2ABF, 0x557F, 0x2AFF, 0x55FF, 0x2BFF, 0x57FF, 0x2FFF, 0x5FFF, 0x3FFF, 0x7FFF, 0x7FFE, 0x7FFC, 0x7FF8, 0x7FF0, 0x7FE0, 0x7FC0, 0x7F80, 0x7F00, 0x7E00, 0x7C00, 0x7800, 0x7000, 0x6000, 0x4000, 0x0001, 0x0002, 0x0004, 0x0008, ...

Figure 14-34. LFSR Block Diagram



14.4.3.3 Shared Clocking

The PWM clocking system has two additional clock features to support a wide variety of applications. Each PWM generator's clock selection (CLKSEL[1:0]) can independently select the pwm_master_clk, or a divided or scaled version of it.

14.4.3.3.1 Clock Divider

A common clock divider circuit is available for use by all PWM Generators and allows a PWM Generator to be operated at a low frequency. Four different divider ratios may be selected using the DIVSEL[1:0] control bits (PCLKCON[5:4]). The clock divider circuit remains in a low-power state if none of the PWM Generators have requested it.

14.4.3.3.2 Frequency Scaling

Frequency scaling provides the ability to drop clocks to effectively stretch the period or duty cycle and is useful for resonant power control applications that require a variable frequency control input.

The clock input for the frequency scaling circuit is chosen using the MCLKSEL bits (PCLKCON[1:0]). The frequency scaling clock output is available to each PWM Generator and can be selected using the CLKSEL[1:0] control bits (PGxCON[4:3]).

The FSCL (Frequency Scale) and FSMINPER (Frequency Scaling Minimum Period) registers specify the amount of frequency scaling and are read/writable at all times. The frequency scaling circuit performs modulo arithmetic where the FSCL value is constantly accumulated until the sum is larger than the FSMINPER register value. When the sum becomes larger than the FSMINPER register value, a clock pulse is produced and the accumulated value is reduced by the value in the FSMINPER register, as shown in Figure 14-35.

Note that the frequency scaling signal is applied only to the PWM time base counter, and does not affect the operation of the dead time or the LEB counters. The frequency scaling circuit remains in a low-power state if not selected by any of the PWM Generators.

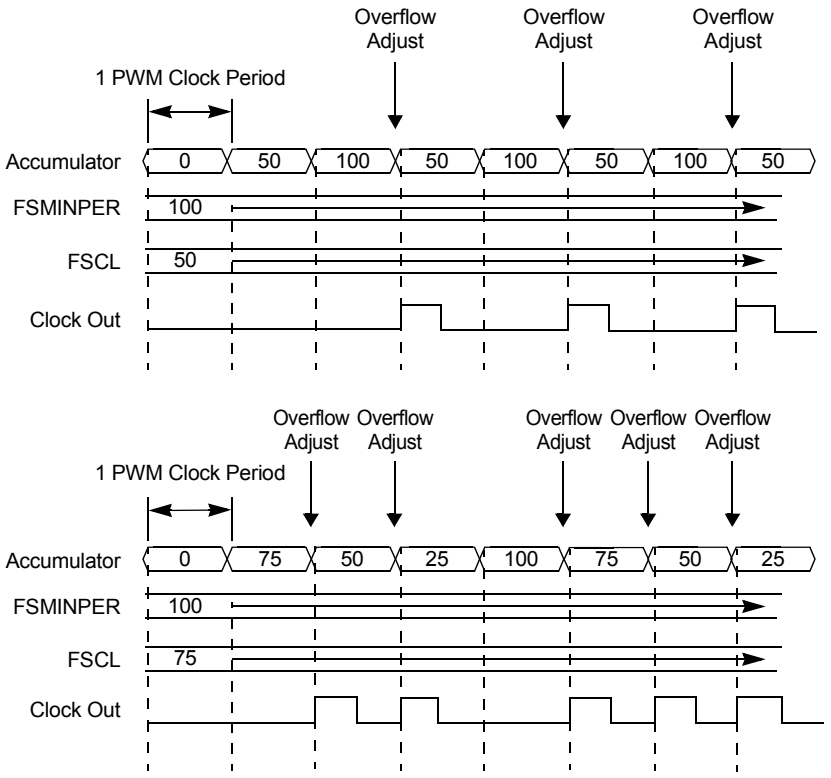
Example: Frequency Scaling Calculation

$F_{FSCL} = (FSCL/FSMINPER) * F_{PWM}$

Where:

FSCL ≤ FSMINPER

Figure 14-35. Frequency Scaling Examples



Note: When the frequency scaling circuit is selected as a PWM Generator clock source, the PWM Generator receives two clocks. One clock is the raw clock used to operate the frequency scaling circuit itself. This clock is also used to operate the dead-time counter and LEB counter within the PWM Generator. The second clock is the output of the frequency scaling circuit. This clock is used to operate the PWM time base counter.

14.4.3.4 Combinatorial Logic Output

The combinatorial logic output feature can be used to generate control signals for synchronous rectification or other applications. One or more PWM Generators can be used to output a logic function with programmable input selections and logic functions. When assigned to a PWM output, the combinatorial logic function replaces the PWM signal that would normally be connected to that pin. The controls include:

- Input sources (PWMSxy)
- Input polarity (SxyPOL)
- Logic AND, OR, XOR function (PWMLFy)
- Output destination (PWMLFyD)
  Note: An 'x' in a bit name denotes Input Source '1' or '2'. A 'y' denotes a function instance (A-F).

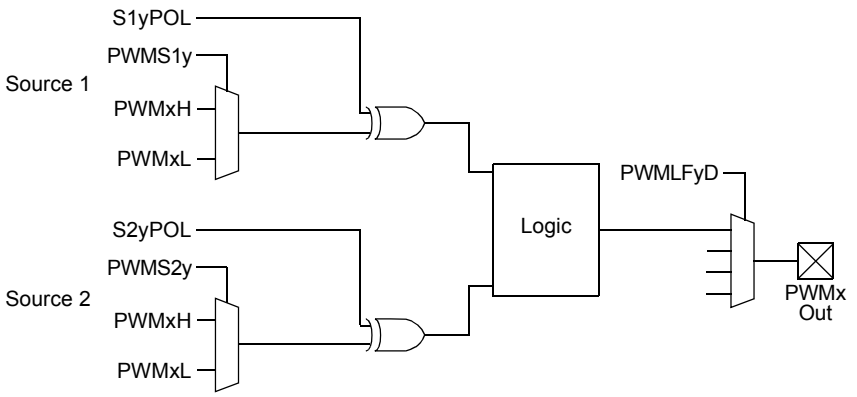
An example of a device with six 14.3.2.9. LOGCONy registers and combinatorial logic output functions, A-F, is shown in Table 14-13.

Table 14-13. Combinatorial Logic Instance Mapping

Register	Combinatorial Logic Instance	Available Output Pin Selection
LOGCONA	A	PWM2H-PWM4H
LOGCONB	B	PWM2L-PWM4L
LOGCONC	C	PWM2H-PWM4H
LOGCOND	D	PWM2L-PWM4L
LOGCONE	E	PWM2H-PWM4H
LOGCONF	F	PWM2L-PWM4L

Figure 14-36 shows the combinatorial logic function block diagram.

Figure 14-36. Combinatorial Logic Function Block Diagram



Note: When using combinatorial logic, the two inputs from the PWM Generators must operate from the same clock source; otherwise, the outputs may not be valid.
The minimum pulse width of the combinatorial output is device-specific and may be limited by the device pins.

The PWM Generator outputs selected as the source inputs are taken before the PWMx output polarity control, POLH/POLL (PGxIOCON[17:16]). If no destination is selected, the combinatorial logic is disabled. The output destination is grouped into pairs where the odd LOGCONy registers (Instances A, C and E) can only be assigned to the PWMxH output pins and the even LOGCONy registers (Instances B, D and F) can only be assigned to the PWMxL pins. Only PWM2-PWM8 can use the combinatorial logic output; PWM1 is not available. More than one instance (A-F) of a

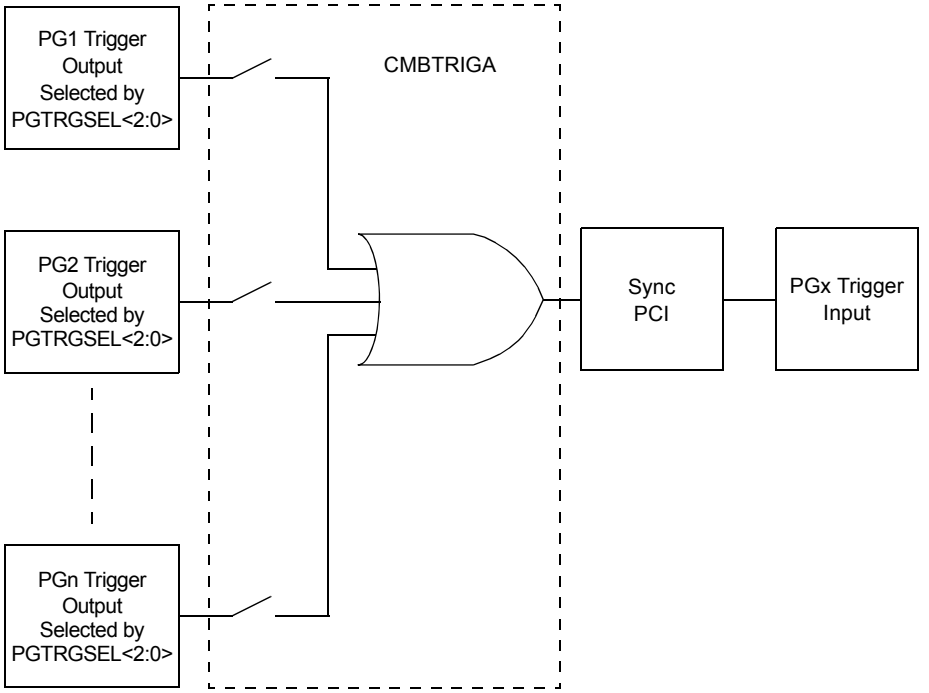
combinatorial logic output can be assigned to a single PWM output if desired. In the case that multiple combinatorial logic functions have been enabled and assigned to the same PWM output, the function with the lowest letter value will take priority.

14.4.3.5 Combinatorial Triggers

Complex triggering algorithms can be created using the combinatorial trigger feature. There are two independent combinatorial trigger circuits: A and B. This feature allows trigger outputs from multiple PWM Generators to be combined into a single trigger signal, to be used as the trigger source for another PWM Generator.

The input signals used as sources for the combinatorial trigger logic are the trigger outputs selected by the PGTRGSEL[2:0] control bits in each PGxEVTL control register. This trigger output can either be End-of-Cycle (EOC) or one of the three PGxTRIGy (y = A, B or C) compare events. These trigger output signals can be enabled and logically OR'd together by setting the appropriate bits in the CMBTRIGH/CMBTRIGL registers. The Combinatorial Trigger A and Combinatorial Trigger B outputs are then made available on the PWM Control Input (PCI) logic input multiplexers and routed through the PCI logic for synchronization. Finally, the signal can then be selected as a PWM Generator's input trigger. A block diagram showing an example is shown in Figure 14-37. See 14.4.2.4. PWM Generator Triggers for details on triggering.

Figure 14-37. Combinatorial Triggers Block Diagram, Example of Instance A



14.4.3.6 PWM Event Outputs

The PWM event output feature provides a mechanism to interface various PWM signals and events to other peripherals and external devices. The PWM event output logic provides a way to select and condition an event from any of the PWM Generators. Each PWM Event Output block has the following configuration options:

- PWM Generator Instance (PG1...PG8)
- Choice of Signal from PWM Generator
- Pulse Stretching

- Output Signal Polarity
- System Clock Synchronization
- Output Enable for the Signal

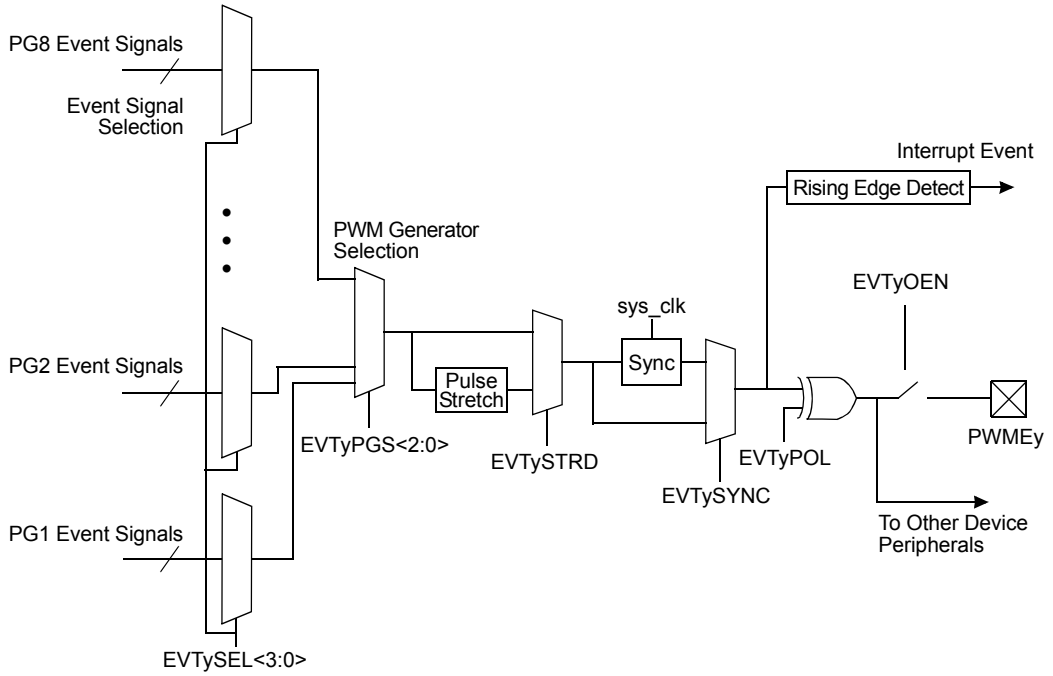
Each 14.3.2.10. PWMEVTy register contains controls for a PWM event output. A device may have multiple instances (A-F) of the PWMEVTy registers, resulting in four or more total PWM event outputs.

The EVTySEL[3:0] (PWMEVTy[7:4]) bits select the signal to be used by the Output block. The default source is the selection determined by PGTRGSEL[2:0] (PGxEVTL[2:0]). For additional information on these signals and configuring the ADC triggers, see 14.4.2.9. Event Selection Block. The EVTyPGS[2:0] bits (PWMEVTy[2:0]) are then used to select which of the eight PWM Generator's event signals is to be used.

Some of the event signals running at high speed have short pulses that may not be detected by other circuits and would make it impossible, for example, to connect a PWM event signal to an off-chip destination. A pulse stretching circuit can be used to extend the duration of the pulse by setting the EVTySTRD bit (PWMEVTy[13]). If synchronization to the main PWM clock domain is desired, the EVTySYNC bit (PWMEVTy[12]) can be set. The EVTyPOL (PWMEVTy[14]) control is provided to invert the polarity of the event signal. Finally, an output enable bit, EVTyOEN (PWMEVTy[15]), is provided for control over the output pin PWMEy.

The PWM event output can also generate a system interrupt. An interrupt can be generated from any of the various triggers and events that are input into the Event Output block. A block diagram of the event output function is shown in Figure 14-38.

Figure 14-38. PWM Event Output Function



14.5 Application Examples

14.5.1 Six-Step Commutation of Three-Phase BLDC Motor

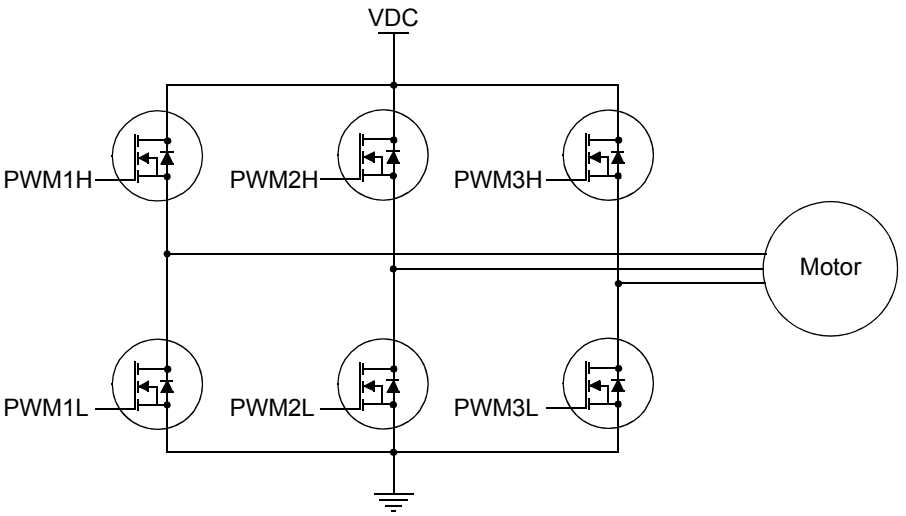
One method for controlling a three-phase Brushless DC (BLDC) is six-step commutation. Six-step commutation is also called 120° commutation, or trapezoidal control, and uses six steps or 'sectors'

over one electrical cycle to energize a BLDC motor. Each sector is equivalent to 60 electrical degrees, with the six sectors resulting in 360 electrical degrees or one electrical cycle.

Sequencing through these steps moves the motor through one electrical cycle, which in mechanical terms, corresponds to one pair of rotor magnet poles moving past stator windings. A given BLDC motor has a certain number of pole pairs, defined by Np as a positive integer. If the motor is rotated one mechanical revolution, this corresponds to Np electrical cycles. This yields one electrical cycle for a 2-pole motor, two electrical cycles for a 4-pole motor, three electrical cycles for a 6-pole motor and so on.

A six-step commutation drive is typically implemented using a three-phase bridge circuit, as shown in Figure 14-39. Each phase of the motor is connected to a half-bridge driver and controlled with a complementary PWM pair output. At any given time in the six-step commutation scheme, only two of the three motor windings are energized. Current in the motor winding flows from one phase to another, in either direction.

Figure 14-39. Three-Phase Bridge and Motor



Various PWM switching techniques can be employed for six-step commutation. The following Table 14-14 summarizes the three schemes presented in this manual.

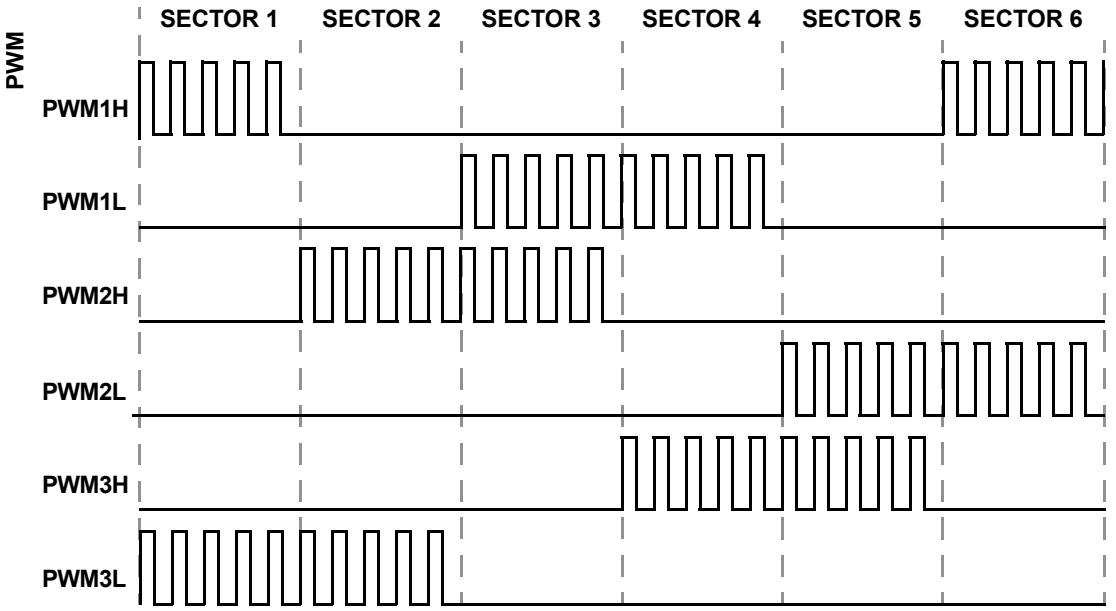
Table 14-14. PWM Switching Schemes for 6-Step Commutation

Scheme	Technique Overview	Advantage	Disadvantage
Scheme1	High side of one active phase and low side of the other active phase driven at any given time	Simplest scheme; no dead time needed; low switching loss	High-current ripple
Scheme2	One active phase driven complementary and the other active phase's low side is driven to 100% duty cycle	Low switching loss	Requires dead time
Scheme 3	Both active phases are driven complementary	Lowest current ripple	Higher switching loss and requires dead time

14.5.1.1 Six-Step Commutation – PWM Scheme 1

In this PWM scheme, only two switches are active at any given time. Of the two active phases, one high-side and one low-side switch are controlled with their phase's corresponding PWM waveform, as shown in Figure 14-40.

Figure 14-40. Six-Step PWM Scheme 1 Waveform



Since only one switch needs to be driven at a time on a given phase, Independent PWM Output mode is used. The output override feature is then used to suppress the unused output. A three-phase scheme is implemented using PWM Generator 1 (PG1) configured as host and the other two PWM Generators (PG2 and PG3) configured as clients. PG1 is self-triggered, whereas PG2 and PG3 are triggered from PG1's Start-of-Cycle (SOC). Enabling PG1 will start the system in a synchronized fashion.

Configuration Summary:

- Independent Edge PWM mode
- Independent Output mode
- Master Period and Duty Cycle Used
- Override State is drive low

Example 14-1. Six-Step PWM Scheme 1 Code

```
//#include <stdint.h>
//For delay function
#define FCY 8000000                                 //CPU frequency in Hz
#include "libpic30.h"

uint16_t state = 0;

#define H_ACTIVE_L_LOW  0x001C1000
   //PGxIOCONbits.PMOD = 0b01 -- Independent output mode
   //PGxIOCONbits.PENH = 1 -- PWM Generator x controls PWMxH output pin
   //PGxIOCONbits.PENL = 1 -- PWM Generator x controls PWMxL output pin
   //PGxIOCONbits.OVRENH = 0 -- PWMxH output not overridden
   //PGxIOCONbits.OVRENL = 1 -- PWMxL output overridden
   //PGxIOCONbits.OVRDAT = 0b00 -- Override data for PWMxH/L (only L uses it)
is 0
   //PGxIOCONbits.OSYNC = 0b00 -- User output overrides are synchronized to
next start of cycle
#define H_LOW_L_LOW     0x001C3000
   //PGxIOCONbits.PMOD = 0b01 -- Independent output mode
   //PGxIOCONbits.PENH = 1 -- PWM Generator x controls PWMxH output pin
   //PGxIOCONbits.PENL = 1 -- PWM Generator x controls PWMxL output pin
   //PGxIOCONbits.OVRENH = 1 -- PWMxH output overridden
   //PGxIOCONbits.OVRENL = 1 -- PWMxL output overridden
```

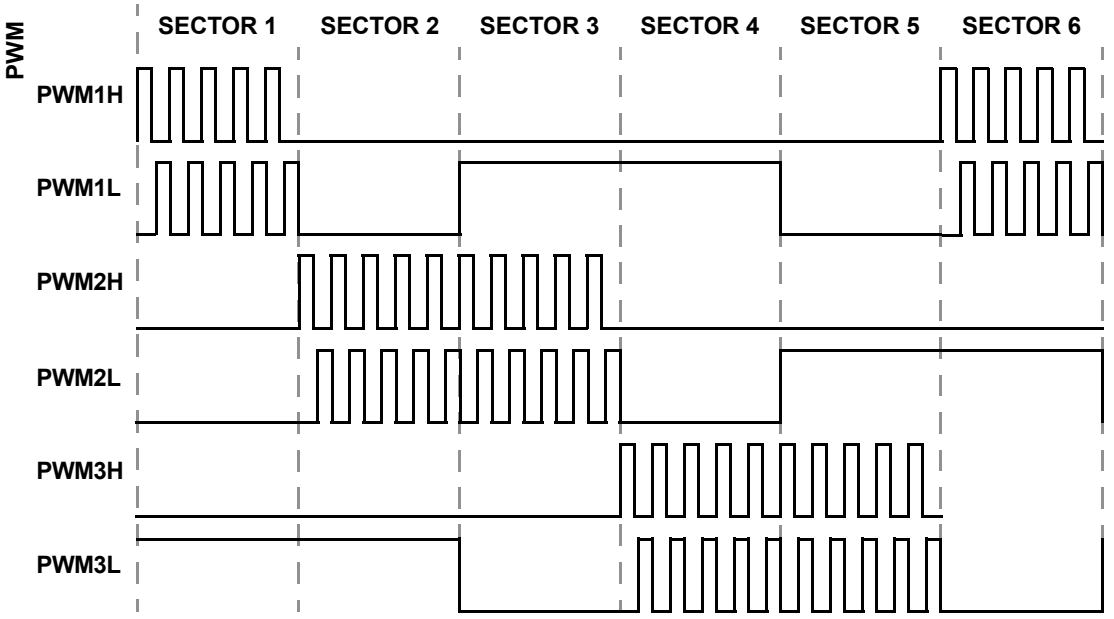
```
   //PGxIOCONbits.OVRDAT = 0b00 -- Override data for PWMxH/L is 0
   //PGxIOCONbits.OSYNC = 0b00 -- User output overrides are synchronized to
next start of cycle
#define H_LOW_L_ACTIVE  0x001C2000
   //PGxIOCONbits.PMOD = 0b01 -- Independent output mode
   //PGxIOCONbits.PENH = 1 -- PWM Generator x controls PWMxH output pin
   //PGxIOCONbits.PENL = 1 -- PWM Generator x controls PWMxL output pin
   //PGxIOCONbits.OVRENH = 1 -- PWMxH output overridden
   //PGxIOCONbits.OVRENL = 0 -- PWMxL output not overridden
   //PGxIOCONbits.OVRDAT = 0b00 -- Override data for PWMxH/L (only H uses it)
is 0
   //PGxIOCONbits.OSYNC = 0b00 -- User output overrides are synchronized to
next start of cycle

uint32_t PWM1State[6] = {H_ACTIVE_L_LOW, H_LOW_L_LOW, H_LOW_L_ACTIVE,
H_LOW_L_ACTIVE, H_LOW_L_LOW, H_ACTIVE_L_LOW};
uint32_t PWM2State[6] = {H_LOW_L_LOW, H_ACTIVE_L_LOW, H_ACTIVE_L_LOW,
H_LOW_L_LOW, H_LOW_L_ACTIVE, H_LOW_L_ACTIVE};
uint32_t PWM3State[6] = {H_LOW_L_ACTIVE, H_LOW_L_ACTIVE, H_LOW_L_LOW,
H_ACTIVE_L_LOW, H_ACTIVE_L_LOW, H_LOW_L_LOW};

//Sector 1:
//PWM1H is active, PWM1L is overridden low.
//PWM2H is overridden low, PWM2L is overridden low.
//PWM3H is overridden low, PWM3L is active.

//Sector 2:
//PWM1H is overridden low, PWM1L is overridden low.
//PWM2H is active, PWM2L is overridden low.
//PWM3H is overridden low, PWM3L is active.

//Sector 3:
//PWM1H is overridden low, PWM1L is active.
//PWM2H is active, PWM2L is overridden low.
//PWM3H is overridden low, PWM3L is overridden low.

//Sector 4:
//PWM1H is overridden low, PWM1L is active.
//PWM2H is overridden low, PWM2L is overridden low.
//PWM3H is active, PWM3L is overridden low.

//Sector 5:
//PWM1H is overridden low, PWM1L is overridden low.
//PWM2H is overridden low, PWM2L is active.
//PWM3H is active, PWM3L is overridden low.

//Sector 6:
//PWM1H is active, PWM1L is overridden low.
//PWM2H is overridden low, PWM2L is active.
//PWM3H is overridden low, PWM3L is overridden low.


void PWMInitialization(void)
{
    //Set PWM master clock to 400MHz from PLL2 through CLKGEN5
    configure_PLL2_Fout_400MHz();
    clock_PWM_from_PLL2_Fout();

    //Ensure PWM generators are disabled before initializing
    PG1CONbits.ON = 0;
    PG2CONbits.ON = 0;
    PG3CONbits.ON = 0;

    //Set PWM Master Period (frequency 100kHz given a 400MHz master clock)
    MPER = (4000 << 4); //4000 master clocks; time scale units are 1/16 of a
clock

    //Set Master Duty Cycle - 25%
    MDC = (2000 << 4);

    //Set Phase shift - No phase shift
    MPHASE = 0;

    //Configure PWM Generator 1

    PG1CONbits.MDCSEL = 1;   //Select MDC as PWM Generator 1's duty cycle
register
    PG1CONbits.MPERSEL = 1;  //Select MPER as PWM Generator 1's period register
```

```
    PG1CONbits.MPHSEL = 1;   //Select MPHASE as PWM Generator 1's phase
register
    PG1CONbits.MSTEN = 1;    //PWM Generator 1 broadcasts software set of
UPDREQ control bit and EOC signal to other PWM Generators


    PG1CONbits.UPDMOD = 0b000;   //PWM buffer update mode is at start of next
PWM cycle if UPDREQ = 1
    PG1CONbits.TRGMOD = 0b00;    //PWM generator 1 operates in single trigger
mode
    PG1CONbits.SOCS = 0b0000;    //Start of cycle is local EOC


    PG1CONbits.ON = 0;          //PWM Generator 1 is disabled (do not start
yet)
    PG1CONbits.TRGCNT = 0;      //PWM Generator 1 produces 1 PWM cycle when
triggered
    PG1CONbits.CLKSEL = 0b01;   //PWM Generator 1 uses PWM Master Clock,
undivided and unscaled
    PG1CONbits.MODSEL = 0b000;  //PWM Generator 1 operates in Independent Edge
PWM mode

    PG1IOCONbits.PMOD = 0b01;   //PWM Generator 1 Output Mode is Independent
Mode
    PG1IOCONbits.PENH = 1;      //PWM Generator 1 controls the PWM1H output pin
    PG1IOCONbits.PENL = 1;      //PWM Generator 1 controls the PWM1L output pin

    //Override is enabled on PWMxH/L with OVRDAT = 0b00, turning OFF PWM
outputs
    PG1IOCONbits.OVRENH = 1;
    PG1IOCONbits.OVRENL = 1;
    PG1IOCONbits.OVRDAT = 0b00;
    PG1IOCONbits.OSYNC = 0b0
```

14.5.1.2 Six-Step Commutation – PWM Scheme 2

In this PWM scheme, three switches are used to control the two active phases. In a given sector, one active phase is driven with a complementary PWM waveform and the other active phase has only its low side driven low at 100% duty cycle, as shown in Figure 14-41. Like Scheme 1, overrides are used to control the outputs in each sector.

Figure 14-41. Six-Step PWM Scheme 2 Waveform



In this scheme, Complementary Output mode is used and overridden as needed in each sector. The same three-phase host/client synchronization technique is used as in Scheme 1.

Configuration Summary:

- Independent Edge PWM mode
- Complementary Output mode
- Master Period and Duty Cycle Used
- Override State is Dependent on Sector State
- Dead time is applied to the Complementary PWM Signal

Example 14-2. Six-Step PWM Scheme 2 Code

```
#include<stdint.h>
//For delay function
#define FCY 8000000                 //CPU frequency in Hz
#include "libpic30.h"


void PWMInitialization(void);

#define H_ACTIVE_L_ACTIVE  0x000C0000
   //PGxIOCONbits.PMOD = 0b00 -- Complementary output mode
   //PGxIOCONbits.PENH = 1 -- PWM generator x controls PWMxH pin
   //PGxIOCONbits.PENL = 1 -- PWM generator x controls PWMxL pin
   //PGxIOCONbits.OVRENH = 0 -- Override not enabled on PWMxH pin
   //PGxIOCONbits.OVRENL = 0 -- Override not enabled on PWMxL pin
#define H_LOW_L_HIGH         0x000C3400
   //PGxIOCONbits.PMOD = 0b00 -- Complementary output mode
   //PGxIOCONbits.PENH = 1 -- PWM generator x controls PWMxH pin
   //PGxIOCONbits.PENL = 1 -- PWM generator x controls PWMxL pin
   //PGxIOCONbits.OVRENH = 1 -- Override enabled on PWMxH pin
   //PGxIOCONbits.OVRENL = 1 -- Override enabled on PWMxL pin
   //PGxIOCONbits.OVRDAT = 0b01 -- PWMxH pin overridden high, PWMxL pin
overridden low
#define H_LOW_L_LOW             0x000C3000
   //PGxIOCONbits.PMOD = 0b00 -- Complementary output mode
   //PGxIOCONbits.PENH = 1 -- PWM generator x controls PWMxH pin
   //PGxIOCONbits.PENL = 1 -- PWM generator x controls PWMxL pin
   //PGxIOCONbits.OVRENH = 1 -- Override enabled on PWMxH pin
   //PGxIOCONbits.OVRENL = 1 -- Override enabled on PWMxL pin
   //PGxIOCONbits.OVRDAT = 0b00 -- PWMxH pin overridden low, PWMxL pin
overridden low

uint16_t state = 0;

uint32_t PWM1State[6] = {H_ACTIVE_L_ACTIVE, H_LOW_L_LOW, H_LOW_L_HIGH,
H_LOW_L_HIGH, H_LOW_L_LOW, H_ACTIVE_L_ACTIVE};
uint32_t PWM2State[6] = {H_LOW_L_LOW, H_ACTIVE_L_ACTIVE, H_ACTIVE_L_ACTIVE,
H_LOW_L_LOW, H_LOW_L_HIGH, H_LOW_L_HIGH};
uint32_t PWM3State[6] = {H_LOW_L_HIGH, H_LOW_L_HIGH, H_LOW_L_LOW,
H_ACTIVE_L_ACTIVE, H_ACTIVE_L_ACTIVE, H_LOW_L_LOW};

//Sector 1:
//PWM1 has complementary outputs, both controlled by the PWM generator.
//PWM2 has both outputs overridden to low.
//PWM3 has High output overridden to low, Low output overridden to high.
//Sector 2:
//PWM1 has both outputs overridden to low.
//PWM2 has complementary outputs, both controlled by the PWM generator.
//PWM3 has High output overridden to low, Low output overridden to high.
//Sector 3:
//PWM1 has High output overridden to low, Low output overridden to high.
//PWM2 has complementary outputs, both controlled by the PWM generator.
//PWM3 has both outputs overridden to low.
//Sector 4:
//PWM1 has High output overridden to low, Low output overridden to high.
//PWM2 has both outputs overridden to low.
//PWM3 has complementary outputs, both controlled by the PWM generator.
//Sector 5:
//PWM1 has both outputs overridden to low.
//PWM2 has High output overridden to low, Low output overridden to high.
```

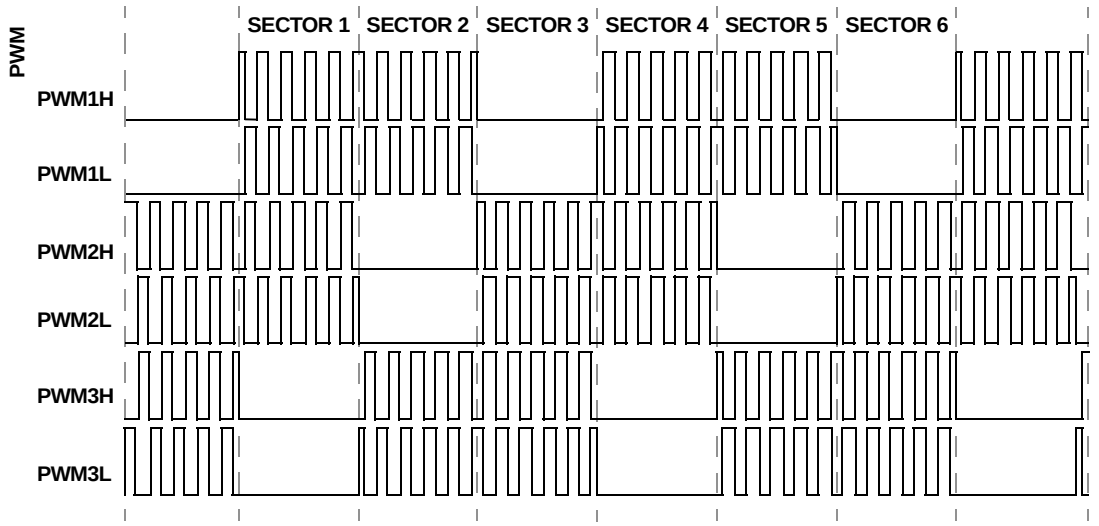
```
//PWM2 has complementary outputs, both controlled by the PWM generator.
//Sector 6:
//PWM1 has complementary outputs, both controlled by the PWM generator.
//PWM2 has High output overridden to low, Low output overridden to high.
//PWM3 has both outputs overridden to low.


void Delay() {
    __delay_us(50); //Delay 5 PWM cycles
}

int main()
{


   //Initialize PWM module
    PWMInitialization();

    while(1)
    {
        for(state = 0; state < 6; state++)
        {
            //Delay is used to simulate BLDC commutation.
            //In practical application, commutation state transition will be
based on feedback from Motor.
            Delay();

            PG1IOCON = PWM1State[state];
            PG2IOCON = PWM2State[state];
            PG3IOCON = PWM3State[state];
        }
    }
}
void PWMInitialization(void)
{
    //Ensure PWM Generators 1-3 are disabled before configuring
    PG1CONbits.ON = 0;
    PG2CONbits.ON = 0;
    PG3CONbits.ON = 0;

    //Set PWM master clock to 400MHz from PLL2 through CLKGEN5
    configure_PLL2_Fout_400MHz();
    clock_PWM_from_PLL2_Fout();

    //Set PWM Period -- 100kHz given a 400MHz PWM master clock
    MPERbits.MPER = (4000 << 4); //Time base units are 1/16 of a PWM clock
    //Set Duty Cycle-- 25%
    MDCbits.MDC = (2000 << 4);
    //Set Phase shift - No phase shift
    MPHASEbits.MPHASE = 0;

    PG1CONbits.MDCSEL = 1;        //Select MDC as PWM Generator 1's Duty Cycle
Register
    PG1CONbits.MPERSEL = 1;       //Select MPER as PWM Generator 1's Period
Register
    PG1CONbits.MPHSEL = 1;        //Select MPHASE as PWM Generator 1's Phase
Register

    //PWM Generator broadcasts software set of UPDREQ control bit and EOC
signal to other PWM Generators
    PG1CONbits.MSTEN = 1;

    PG1CONbits.UPDMOD = 0b000;    //PWM Buffer Update Mode is at start of next
PWM cycle if UPDREQ = 1
    PG1CONbits.TRGMOD = 0b00;     //PWM generator operates in Single Trigger
Mode
    PG1CONbits.SOCS = 0b0000;     //Start of Cycle is local EOC


    PG1CONbits.CLKSEL = 0b01;     //PWM Generator uses PWM Master Clock,
undivided and unscaled
    PG1CONbits.MODSEL = 0b000;    //PWM Generator operates in Independent Edge
PWM mode


    PG1IOCONbits.PMOD = 0b00;    //PWM Generator 1 Output Mode is
```

```
Complementary Mode
    PG1IOCONbits.PENH = 1;       //PWM Generator 1 controls the PWM1H output
```

14.5.1.3 Six-Step Commutation – PWM Scheme 3

In this PWM scheme, four switches are driven in a given sector. Two pairs of complementary PWM outputs are applied to the two active phases. The inactive phase is overridden low as needed, as shown in Figure 14-42.

Figure 14-42. Six-Step PWM Scheme 3 Waveform



In this scheme, Center-Aligned PWM mode is used with dead time to prevent high current during switching transitions. The two active phases are driven 180 degrees out of phase to one another using the SWAP feature.

Configuration Summary:

- Center-Aligned PWM mode
- Complementary Output mode
- Master Period and Duty Cycle Used
- Override and SWAP State are Dependent on Sector State
- Dead time is applied to the Complementary PWM Signal

Example 14-3. Six-Step PWM Scheme 3 Code

```
//For delay function
#define FCY 8000000                         //CPU frequency in Hz
#include "libpic30.h"

void PWMInitialization(void);

unsigned int state = 0;
unsigned int cycleCount = 0;
unsigned int cyclesPerSector = 5;

#define OUTPUTS_ACTIVE 0x000C0000
    //PGxIOCONbits.PMOD = 0b00 -- Complementary output mode
    //PGxIOCONbits.PENH = 1 -- PWM generator x controls PWMxH pin
    //PGxIOCONbits.PENL = 1 -- PWM generator x controls PWMxL pin
    //PGxIOCONbits.OVRENH = 0 -- PWMxH output is not overridden
    //PGxIOCONbits.OVRENL = 0 -- PWMxL output is not overridden
```

```
    //PGxIOCONbits.SWAP = 0 -- PWMxH/PWMxL signals are not swapped

#define OUTPUTS_OFF 0x000C3000
    //PGxIOCONbits.PMOD = 0b00 -- Complementary output mode
    //PGxIOCONbits.PENH = 1 -- PWM generator x controls PWMxH pin
    //PGxIOCONbits.PENL = 1 -- PWM generator x controls PWMxL pin
    //PGxIOCONbits.OVRENH = 1 -- PWMxH output is overridden
    //PGxIOCONbits.OVRENL = 1 -- PWMxL output is overridden
    //PGxIOCONbits.OVRDAT = 0b00 --  Override data for PWMxH/L outputs is 0
    //PGxIOCONbits.SWAP = 0 -- PWMxH/PWMxL signals are not swapped

#define OUTPUTS_ACTIVE_SWAPPED 0x000C4000
    //PGxIOCONbits.PMOD = 0b00 -- Complementary output mode
    //PGxIOCONbits.PENH = 1 -- PWM generator x controls PWMxH pin
    //PGxIOCONbits.PENL = 1 -- PWM generator x controls PWMxL pin
    //PGxIOCONbits.OVRENH = 0 -- PWMxH output is not overridden
    //PGxIOCONbits.OVRENL = 0 -- PWMxL output is not overridden
    //PGxIOCONbits.SWAP = 1 -- PWMxH/PWMxL signals are swapped

#define OUTPUTS_OFF_SWAPPED 0x000C7000
    //PGxIOCONbits.PMOD = 0b00 -- Complementary output mode
    //PGxIOCONbits.PENH = 1 -- PWM generator x controls PWMxH pin
    //PGxIOCONbits.PENL = 1 -- PWM generator x controls PWMxL pin
    //PGxIOCONbits.OVRENH = 1 -- PWMxH output is overridden
    //PGxIOCONbits.OVRENL = 1 -- PWMxL output is overridden
    //PGxIOCONbits.OVRDAT = 0b00 --  Override data for PWMxH/L outputs is 0
    //PGxIOCONbits.SWAP = 1 -- PWMxH/PWMxL signals are swapped

unsigned int PWM1State[6] = {OUTPUTS_ACTIVE, OUTPUTS_ACTIVE, OUTPUTS_OFF,
OUTPUTS_ACTIVE_SWAPPED, OUTPUTS_ACTIVE_SWAPPED, OUTPUTS_OFF_SWAPPED};
unsigned int PWM2State[6] = {OUTPUTS_ACTIVE_SWAPPED, OUTPUTS_OFF_SWAPPED,
OUTPUTS_ACTIVE, OUTPUTS_ACTIVE, OUTPUTS_OFF, OUTPUTS_ACTIVE_SWAPPED};
unsigned int PWM3State[6] = {OUTPUTS_OFF, OUTPUTS_ACTIVE_SWAPPED,
OUTPUTS_ACTIVE_SWAPPED, OUTPUTS_OFF_SWAPPED, OUTPUTS_ACTIVE, OUTPUTS_ACTIVE};

//Sector 1:
//PWM1 is operating with outputs active, non-swapped.
//PWM2 is operating with outputs active, swapped.
//PWM3 has outputs overridden off.

//Sector 2:
//PWM1 is operating with outputs active, non-swapped.
//PWM2 has outputs overridden off, swapped.
//PWM3 is operating with outputs active, swapped.

//Sector 3:
//PWM1 has outputs overridden off.
//PWM2 is operating with outputs active, non-swapped.
//PWM3 is operating with outputs active, swapped.

//Sector 4:
//PWM1 is operating with outputs active, swapped.
//PWM2 is operating with outputs active, non-swapped.
//PWM3 has outputs overridden off, swapped.

//Sector 5:
//PWM1 is operating with outputs active, swapped.
//PWM2 has outputs overridden off.
//PWM3 is operating with outputs active, non-swapped.

//Sector 6:
//PWM1 has outputs overridden off, swapped.
//PWM2 is operating with outputs active, swapped.
//PWM3 is operating with outputs active, non-swapped.


int main(void)
{

    //Initialize PWM Generators 1 - 3
    PWMInitialization();

    //Between setting PGxCONbits.ON = 1 and assigning to MDC, a delay is
needed.
    //Otherwise the write will not be successful (duty cycle will remain 0)
    __delay_us(10);

    //To Update Duty cycle values to PG1-PG3:
```

```
    //1) Write MDC register
    //2) Set update request bit PG1STATbits.UPDREQ.
    //This will transfer MDC value to all the PWM generators PG1-PG3.
    //Note that Update Mode(UPDMOD) of PG2,PG3 is client EOC and
PG1CONbits.MSTEN == 1.

    //Set master duty cycle to 50%
    MDC = (1000 << 4);
    //Update duty cycle, etc.
    PG1STATbits.UPDREQ = 1;

    //Clear variables used in the _PWM1Interrupt()
    state = 0;
    cycleCount = 0;

    //Enable PWM Generator 1 Interrupt
    _PWM1IE = 1;

    while (1)
    {

    }

}
void PWMInitialization(void)
{
    //Ensuring PWM Generators 1-3 are disabled prior to configuring module
    PG1CONbits.ON = 0;
    PG2CONbits.ON = 0;
    PG3CONbits.ON = 0;

    //Set PWM master clock to 400MHz from PLL2 through CLKGEN5
    configure_PLL2_Fout_400MHz();
    clock_PWM_from_PLL2_Fout();

    //Set PWM frequency to 100kHz given a PWM master clock of 400MHz
    //Note that center-aligned mode uses 2 timer periods per PWM cycle.
    MPER = (2000 << 4); //Time base units are 1/16 of a clock period
    //Master duty cycle initialized as 0
    MDC = 0;
    //No master phase offset
    MPHASE = 0;

    PG1CONbits.CLKSEL = 0b01;   //PWM Generator 1 uses Master PWM clock,
undivided and unscaled
    PG1CONbits.MODSEL = 0b100;  //PWM Generator 1 uses Center-Aligned PWM mode
(interrupt/register update o
```

14.5.2 Three-Phase Sinusoidal Control of PMSM/ACIM Motors

Three-phase sinusoidal control applies voltages to the three-phase motor windings, which are Pulse-Width Modulated to produce sinusoidal currents as the motor spins. This eliminates the torque ripple and commutation spikes associated with trapezoidal commutation. Typically, Center-Aligned Complementary PWMs are used for sinusoidal control of a Permanent Magnet Synchronous Motor (PMSM) or three-phase AC Induction Motor (ACIM). Center-aligned PWM signals reduce the level of harmonics in output voltages and currents as compared to edge-aligned PWMs. Three PWM Generators are connected to the three-phase power bridge driving the motor, as shown in Figure 14-39.

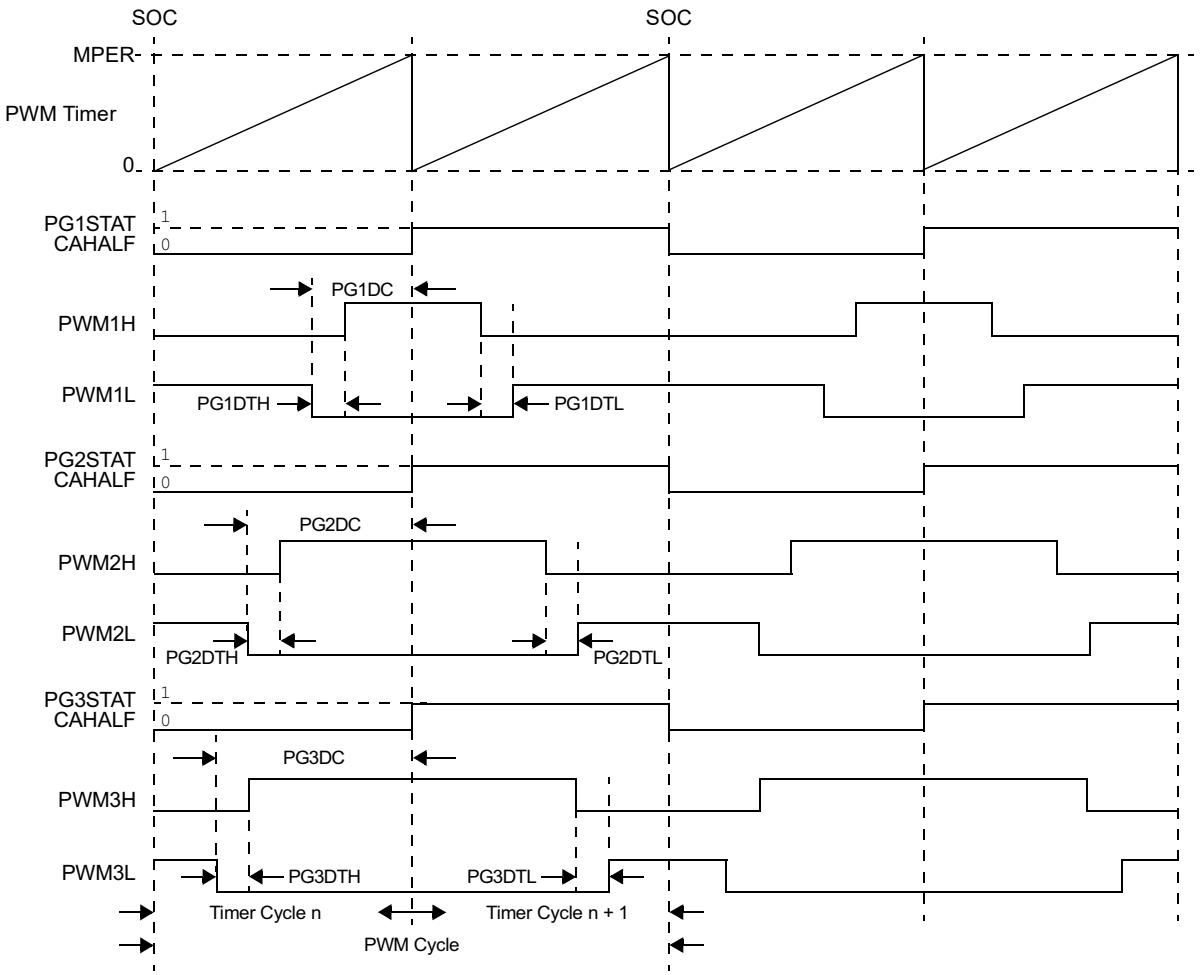
PWM Generator 1 is configured as host, and PWM Generator 2 and PWM Generator 3 are configured as client PWMs. PWM configuration used in three-phase sinusoidal control is summarized below:

- PG1-PG3: Uses master period and independent duty cycles
- PG1-PG3: PWM Operating mode is selected as Center-Aligned mode
- PG1-PG3: Configured to operate in Single Trigger mode
- PG1-PG3: PWM Output mode is configured as Complementary Output mode
- PG2-PG3: Uses PG1 trigger output as Start-of-Cycle, whereas PG1 is self-triggered

- PG2-PG3: Enabled during initialization
- PG1 is enabled only after configuring all the control registers; whenever PG1 is enabled, all the generators will start in unison

Figure 14-43 shows the PWM waveforms for a given point in time. Center-Aligned mode uses two timer cycles to produce a PWM cycle and maintains symmetry at the center of each PWM cycle. Each timer cycle can be tracked using the status bit, CAHALF (PGxSTAT[1]), of the respective PWM Generator. The leading edge is produced when CAHALF = 0 and the falling edge is produced when CAHALF = 1. Note that with Center-Aligned mode, as long as the duty cycles are different for each phase, the switching instants occur at different times. (In Edge-Aligned mode, the turn-on times are coincident.) This generally reduces electromagnetic interference.

Figure 14-43. Three-Phase Center-Aligned Waveforms



Example 14-4. Three-Phase Sinusoidal PMSM/ACIM Motor Control Code

```
void PWMInitialization(void);

int main() {

    PWMInitialization();

    while(1) {
        //Anything to do here?
    }
```

```
    return 0;
}

void PWMInitialization(void) {

    //Set PWM master clock to 400MHz from PLL2 through CLKGEN5
    configure_PLL2_Fout_400MHz();
    clock_PWM_from_PLL2_Fout();

    //Set PWM Master Phase Register - No phase shift
    MPHASE = 0;

    //Set PWM Period (frequency 100kHz, given a 400MHz PWM master clock)
    //Note that center-aligned mode takes 2 timer cycles to produce 1 PWM
cycle.
    MPER = (2000 << 4); //Time units are 1/16 of PWM period

    //Set PWM Duty Cycles
    PG1DC = (500 << 4); //25%
    PG2DC = (1000 << 4); //50%
    PG3DC = (1500 << 4); //75%

    //Ensure PWM Generators 1-3 are disabled before initializing
    PG1CONbits.ON = 0;
    PG2CONbits.ON = 0;
    PG3CONbits.ON = 0;

    //Set Dead Time Registers
    PG1DTbits.DTL = (200 << 4);    //200 PWM clocks of dead time on PWM1L
    PG1DTbits.DTH = (200 << 4);    //200 PWM clocks of dead time on PWM1H
    PG2DTbits.DTL = (200 << 4);    //200 PWM clocks of dead time on PWM2L
    PG2DTbits.DTH = (200 << 4);    //200 PWM clocks of dead time on PWM2H
    PG3DTbits.DTL = (200 << 4);    //200 PWM clocks of dead time on PWM3L
    PG3DTbits.DTH = (200 << 4);    //200 PWM clocks of dead time on PWM3H


    PG1CONbits.MDCSEL = 0;         //Select PG1DC as PWM Generator 1's duty
cycle register
    PG1CONbits.MPERSEL = 1;        //Select MPER as PWM Generator 1's period
register
    PG1CONbits.MPHSEL = 1;         //Select MPHASE as PWM Generator 1's phase
register
    PG1CONbits.MSTEN = 1;          //PWM Generator 1 broadcasts software set
of UPDREQ control bit and EOC signal to other PWM generators.
    PG1CONbits.UPDMOD = 0b000;     //PWM Buffer Update Mode is at start of
next PWM cycle if UPDREQ = 1

    PG1CONbits.TRGMOD = 0b00;      //PWM generator 1 operates in Single
Trigger Mode

    PG1CONbits.CLKSEL = 1;         //PWM Generator 1 uses PWM Master Clock,
undivided and unscaled
    PG1CONbits.MODSEL = 0b100;     //PWM Generator 1 operates in Center-
Aligned mode
    PG1CONbits.SOCS = 0b0000;      //Start of Cycle is Local EOC

    PG1IOCONbits.PMOD = 0b00;      //PWM Generator 1 Output operates in
Complementary Mode
    PG1IOCONbits.PENH = 1;         //PWM Generator 1 controls the PWM1H output
pin
    PG1IOCONbits.PENL = 1;         //PWM Generator 1 controls the PWM1L output
pin


    PG1EVTbits.ADTR1EN1 = 1;       //PGxTRIGA register compare event is
enabled as trigger source for ADC Trigger 1
    PG1EVTbits.UPDTRG = 0b01;      //A write of the PGxDC register
automatically sets the UPDREQ bit
    PG1EVTbits.PGTRGSEL = 0b000;   //PWM generator trigger output is EOC


    PG2CONbits.MDCSEL = 0;         //Select PG2DC as PWM Generator 2's duty
cycle register
    PG2CONbits.MPERSEL = 1;        //Select MPER as PWM Generator 2's period
register
    PG2CONbits.MPHSEL = 1;         //Select MPHASE as PWM Generator 2's phase
register
```

```
    PG2CONbits.MSTEN = 0;          //PWM generator 2 does not broadcast UPDATE
status bit or EOC signal to other PWM generators
    PG2CONbits.UPDMOD = 0b011;     //PWM Buffer Update Mode is client immediate
    PG2CONbits.TRGMOD = 0b00;      //PWM generator 2 operates in Single
Trigger Mode
    PG2CONbits.SOCS = 0b0001;      //Start of Cycle is PG1 trigger output
selected by PG1EVTbits.PGTRGSEL<2:0> bits


    PG2CONbits.CLKSEL = 1;         //PWM Generator 2 uses PWM Master Clock,
undivided and unscaled
    PG2CONbits.MODSEL = 0b100;     //PWM Generator 2 operates in Center-
Aligned mode


    PG2IOCONbits.PMOD = 0b00;      //PWM Generator 2 output operates in
Complementary Mode
    PG2IOCONbits.PENH = 1;         //PWM Generator 2 controls the PWM2H output
pin
    PG2IOCONbits.PENL = 1;         //PWM Generator 2 controls the PWM2L output
pin


    PG3CONbits.MDCSEL = 0;         //Select PG3DC as PWM Generator 3's duty
cycle register
    PG3CONbits.MPERSEL = 1;        //Select MPER as PWM Generator 3's period
register
    PG3CONbits.MPHSEL = 1;         //Select MPHASE as PWM Generator 3's phase
register
    PG3CONbits.MSTEN = 0;          //PWM generator 3 does not broadcast UPDATE
status bit or EOC signal to other PWM generators

    PG3CONbits.UPDMOD = 0b011;     //PWM Buffer Update Mode is client immediate
    PG3CONbits.TRGMOD = 0b00;      //PWM generator 3 operates in Single
Trigger Mode
    PG3CONbits.SOCS = 0b0001;      //Start of Cycle is PG1 trigger output
selected by PG1EVTbits.PGTRGSEL<2:0> bits


    PG3CONbits.CLKSEL = 1;         //PWM Generator 3 uses PWM Master Clock,
undivided and unscaled
    PG3CONbits.MODSEL = 0b100;     //PWM Generator operates in Center-Aligned
mode

    PG3IOCONbits.PMOD = 0b00;      //PWM Generator 3 output operates in
Complementary Mode
    PG3IOCONbits.PENH = 1;         //PWM Generator 3 controls the PWM3H output
pin
    PG3IOCONbits.PENL = 1;         //PWM Generator 3 controls the PWM3L output
pin


    //Enable all PWM generators; PWM generator 1 will start the sequence
```

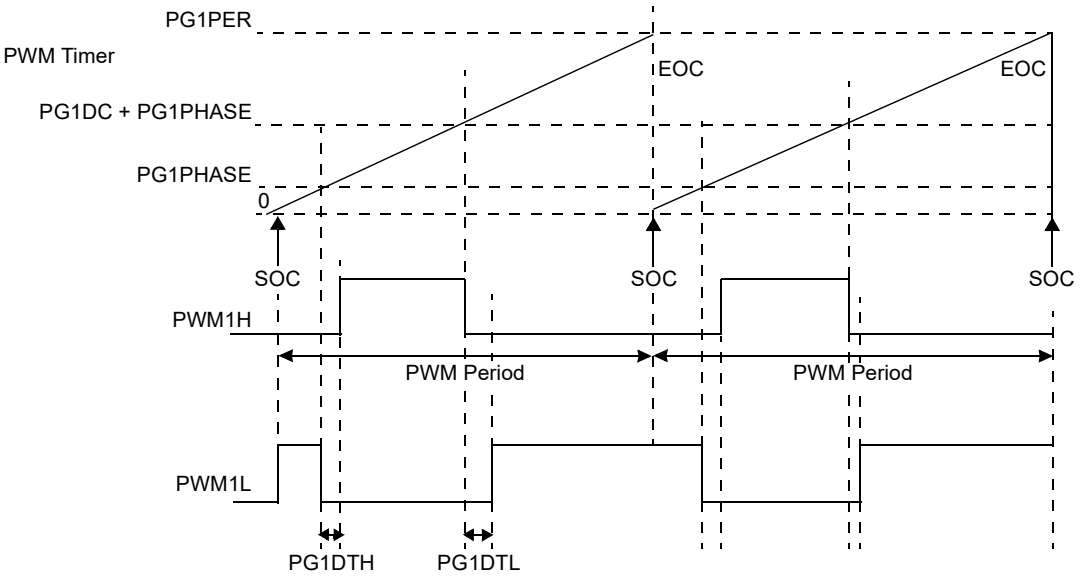
14.5.3 Simple Complementary PWM Output

This complementary PWM example uses a single PWM Generator and can be used for half-bridge applications. The PWM is configured as follows:

- Independent Edge PWM mode
- Complementary Output mode
- Self-Triggered mode

Figure 14-44 shows the timing relations of the PWM signals. In this example, continuous triggering (local EOC) is used in addition to a phase offset (PG1PHASE). Dead time is implemented to prevent simultaneous switch conduction.

Figure 14-44. Timing Diagram for Complementary and Local EOC Triggered PWM Output



Example 14-5. Complementary PWM Output Mode

```
void PWMInitialization(void);

int main() {

    PWMInitialization();

    while(1);

    return 0;
}

void PWMInitialization(void) {

    //clock_PWM_at_400MHz_from_PLL2_Fout();
    clock_PWM_from_UPB_clock();

    //PWM Generator 1 uses PG1DC, PG1PER, PG1PHASE registers
    PG1CONbits.MDCSEL = 0;
    PG1CONbits.MPERSEL = 0;
    PG1CONbits.MPHSEL = 0;

    PG1CONbits.CLKSEL = 1;      //PWM Generator 1 uses PWM Master Clock,
undivided and unscaled
    PG1CONbits.MODSEL = 0b000;  //Independent edge triggered mode
    PG1CONbits.TRGMOD = 0b00;   //PWM Generator 1 operates in Single Trigger
mode
    PG1CONbits.SOCS = 0b0000;   //Start of cycle (SOC) = local EOC

    PG1IOCONbits.PMOD = 0b00;   //PWM Generator 1 outputs operate in
Complementary mode

    //PWM Generator controls the PWM1H and PWM1L output pins
    PG1IOCONbits.PENH = 1;
    PG1IOCONbits.PENL = 1;

    //PWM1H and PWM1L output pins are active high
    PG1IOCONbits.POLH = 0;
    PG1IOCONbits.POLL = 0;

    //Given the 400MHz input clock from CLKGEN5, this period will result in
100kHz PWM frequency
    PG1PER = (4000 << 4);       //Time base units are 1/16 of a PWM clock
    PG1DC = (1000 << 4);        //25% duty cycle
```

```
    PG1PHASE = (400 << 4);      //Rising edge has 400 PWM clocks of phase time
    PG1DTbits.DTH = (80 << 4);  //80 PWM clocks of dead time on PWM1H
    PG1DTbits.DTL = (80 << 4);  //80 PWM clocks of dead time on PWM1L

    //Enable PWM Generator 1
    PG1CONbits.ON = 1;

}
```

14.5.4 Cycle-by-Cycle Current Limit Mode

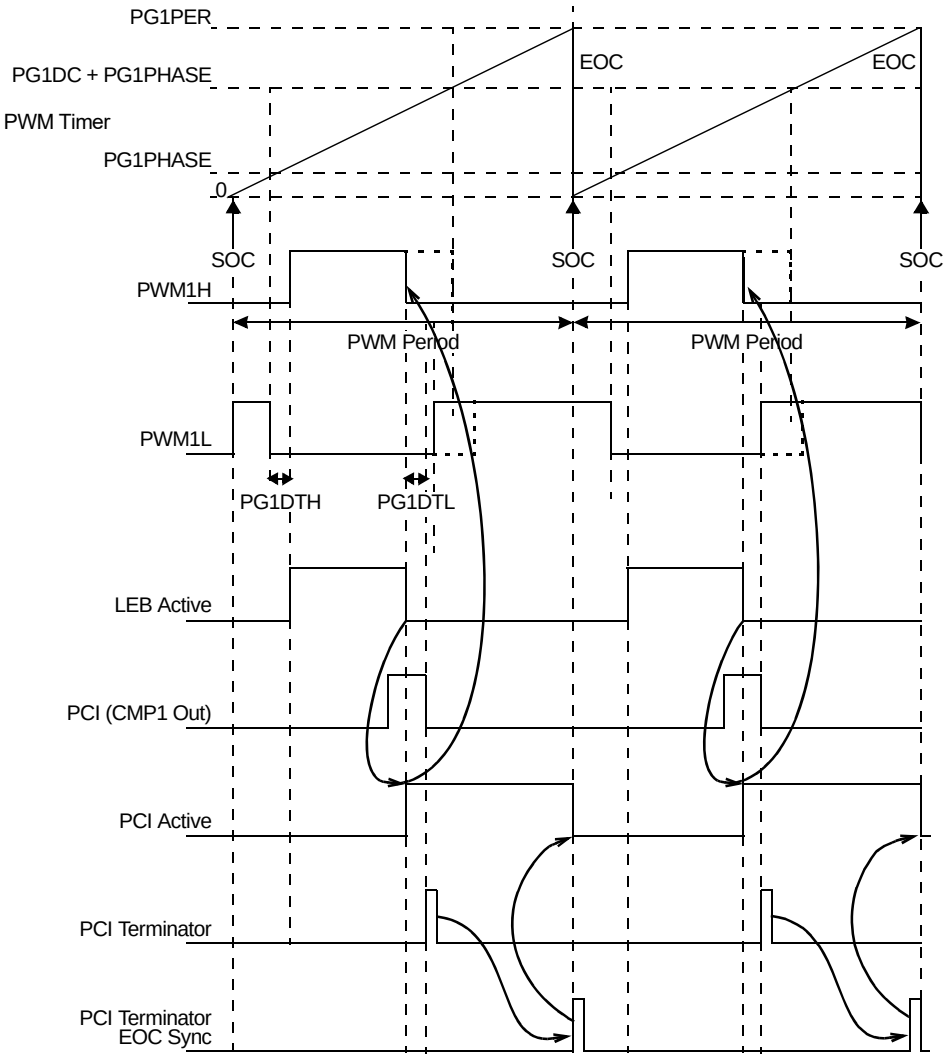
Cycle-by-Cycle Current Limit mode is a widely adopted control strategy for power applications and motor control. Current is measured and limited to a predetermined level using the internal comparator module. Cycle-by-Cycle Current Limit mode control for BLDC motors can automatically limit motor phase currents to a predetermined maximum value, using a comparator to provide an input to the PCI block. This has the advantage of allowing operation to continue when the limit is reached, rather than triggering a Fault. The PWM is configured as follows:

- Independent Edge PWM mode
- Complementary Output mode
- Self-Triggered mode

The current limit PCI block logic is used to control the cycle truncation. The Leading-Edge Blanking feature is used to filter out switching transients and slightly delay cycle truncation, as shown with the upper arrows in Figure 14-45. The duty cycle is truncated when the PCI active signal goes high.

To reset the PCI block for the next cycle, the PCI terminator is configured to detect the falling edge of the comparator (CMP1 Out). The terminator signal is then synchronized to the End-of-Cycle (EOC) and the PCI active signal is reset, as shown with the lower arrows in Figure 14-45.

Figure 14-45. Timing Diagram for Self-Triggered, Complementary Output and Current Limit Cycle-by-Cycle PWM Modes



Example 14-6. Cycle-by-Cycle Current Limit Mode

```
void PWMInitialization(void);
void enable_CMP1();

int main() {

    PWMInitialization();

    //The CMP1A input will be compared against the DAC1 output to create the
CMP1 out signal.
    //If CMP1A > DAC output, PWM1 output will be overridden
    //If CMP1A < DAC output, PWM1 ourput will be active

    while(1);

    return 0;
}

void PWMInitialization(void) {

    configure_PLL2_Fout_200MHz_and_VCODIV_500_MHz();
    clock_PWM_from_PLL2_Fout();
```

```
    initialize_CMP1_and_clock_from_PLL2_VCODIV();

    PG1CONbits.CLKSEL = 1;      //PWM generator 1 uses the PWM master clock,
undivided and unscaled
    PG1CONbits.MODSEL = 0b000;  //PWM generator 1 uses independent edge PWM
mode
    PG1CONbits.TRGMOD = 0b00;   //PWM generator 1 uses single trigger mode
    PG1CONbits.UPDMOD = 0b000;  //Update data registers at SOC

    //PWM Generator 1 uses PG1DC, PG1PER, PG1PHASE registers
    PG1CONbits.MDCSEL = 0;
    PG1CONbits.MPERSEL = 0;
    PG1CONbits.MPHSEL = 0;

    PG1CONbits.MSTEN = 0;       //PWM Generator does not broadcast UPDATE
status bit state or EOC signal
    PG1CONbits.SOCS = 0b0000;   //Start of cycle (SOC) = local EOC

    PG1IOCONbits.PMOD = 0b00;   //PWM Generator 1 outputs operate in
Complementary mode

    //PWM Generator 1 controls the PWM1H and PWM1L output pins
    PG1IOCONbits.PENH = 1;
    PG1IOCONbits.PENL = 1;
    //PWM1H and PWM1L output pins are active high
    PG1IOCONbits.POLH = 0;
    PG1IOCONbits.POLL = 0;
    //Current limit data: 1 on PWM1L and 0 on PWM1H
    PG1IOCONbits.CLDAT = 0b01;


    //Given the 200MHz PWM clock, this period will result in a PWM frequency
of 100kHz
    PG1PER = (2000 << 4);           //Time units are 1/16 of a PWM clock
    PG1DC = (1000 << 4);            //50% duty cycle
    PG1PHASE = (200 << 4);          //200 PWM clocks of phase offset in rising
edge of PWM
    PG1DTbits.DTH = (40 << 4);      //40 PWM clocks of dead time on PWM1H
    PG1DTbits.DTL = (40 << 4);      //40 PWM clocks of dead time on PWM1L
    PG1LEBbits.PHR = 1;             //Rising edge of PWM1H will trigger the
LEB counter
    PG1LEBbits.LEB = (100 << 4);    //100 PWM clocks of LEB

    //PCI logic configuration for current limit cycle by cycle mode,
comparator 1 output as PCI source
    PG1CLPCIbits.TERM = 0b001;      //Terminate when PCI source transitions
from active to inactive
    PG1CLPCIbits.TSYNCDIS = 0;      //Termination of latched PCI delays till
PWM EOC (for Cycle by cycle mode)
    PG1CLPCIbits.AQSS = 0b010;      //LEB active is selected as acceptance
qualifier
    PG1CLPCIbits.AQPS = 1;          //LEB active is inverted to accept PCI
signal when LEB duration is over
    PG1CLPCIbits.PSYNC = 0;         //PCI source is not synchronized to PWM
EOC so that current limit resets PWM immediately
    PG1CLPCIbits.PSS = 0b11011;     //Comparator 1 output is selected as PCI
source signal
    PG1CLPCIbits.PPS = 0;           //PCI source signal is not inverted
    PG1CLPCIbits.ACP = 0b011;       //latched PCI is selected as acceptance
criteria to work when CMP1 out is active
    PG1CLPCIbits.TQSS = 0b0000;     //No termination qualifier used so
terminator will work straight away without any qualifier

    //Enable PWM generator 1
    PG1CONbits.ON = 1;
}
```

14.5.5 External Period Reset Mode

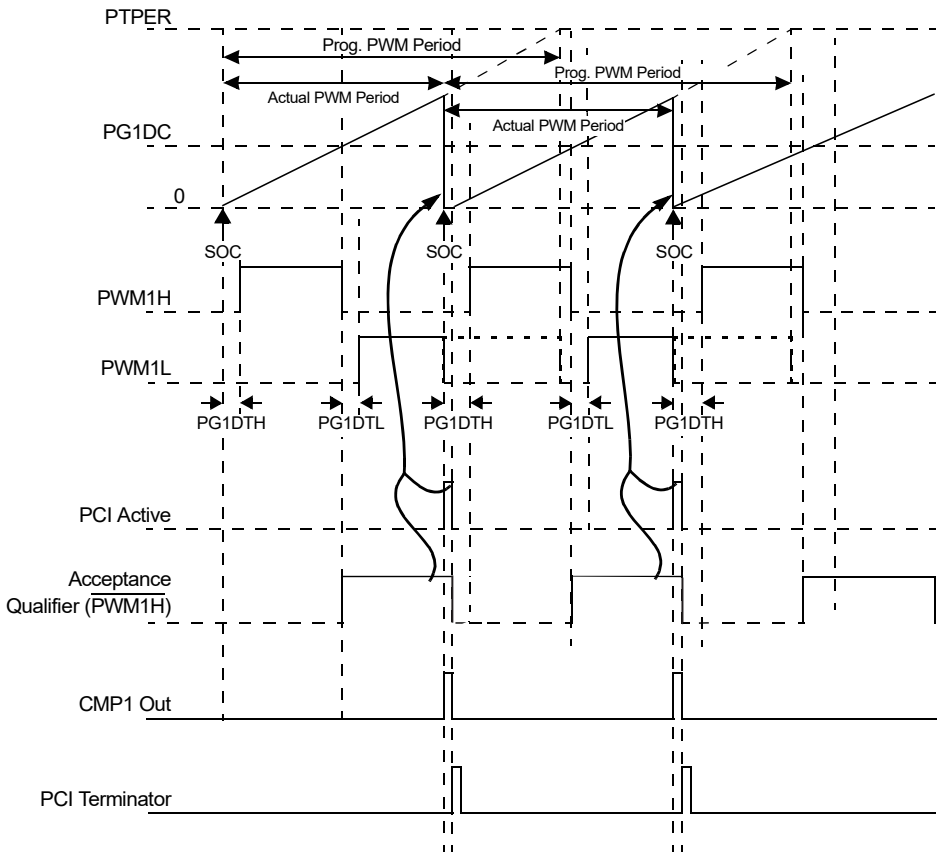
External Period Reset mode for power control monitors the inductor current and varies the PWM period to control power delivery. When the inductor current returns to '0', the PWM cycle will restart. The PWM is configured as follows:

- Independent Edge PWM mode

- Complementary Output mode
- Self-Triggered mode

The initial programmed PWM period may be shortened depending on the inductor current compared to a predetermined trip point. The Sync PCI block is used to control cycle truncation. The comparator (CMP1 Out) output is used as the input to the Sync PCI block, and an inverted PWM1H signal is used as a qualifier to allow truncation only on the duty cycle inactive time of a cycle, as shown by the arrows in Figure 14-46. The PCI block is reset for the next cycle using the auto-terminate function.

Figure 14-46. Timing Diagram for Self-Triggered, Complementary Output and Current Reset PWM Modes



Example 14-7. External Period Reset Mode

```
void PWMInitialization(void);

int main() {

    PWMInitialization();

    while(1);

    return 0;
}

void PWMInitialization(void) {

    configure_PLL2_Fout_200MHz_and_VCODIV_500_MHz();
    clock_PWM_from_PLL2_Fout();
    initialize_CMP1_and_clock_from_PLL2_VCODIV();
```

```
    //PWM generator 1 uses PG1DC, PG1PER, PG1PHASE registers
    PG1CONbits.MDCSEL = 0;
    PG1CONbits.MPERSEL = 0;
    PG1CONbits.MPHSEL = 0;

    PG1CONbits.MSTEN = 0;             //PWM Generator does not broadcast
UPDATE status bit state or EOC signal
    PG1CONbits.TRGMOD = 0b01;         //PWM Generator operates in Re-
Triggerable mode
    PG1CONbits.SOCS = 0b0000;         //Start of cycle (SOC) = local EOC is
OR'd with PCI sync
    PG1CONbits.UPDMOD = 0b000;        //Update the data registers at start of
next PWM cycle (SOC)

    PG1CONbits.MODSEL = 0b000;        //Independent edge triggered mode
    PG1CONbits.CLKSEL = 1;            //PWM Generator 1 uses PWM Master Clock,
undivided and unscaled

    PG1IOCONbits.PMOD = 0b00;         //PWM Generator outputs operate in
Complementary mode

    //PWM Generator controls the PWM1H and PWM1L output pins
    PG1IOCONbits.PENH = 1;
    PG1IOCONbits.PENL = 1;

    //PWM1H and PWM1L output pins are active high
    PG1IOCONbits.POLH = 0;
    PG1IOCONbits.POLL = 0;

    //For a 200MHz PWM clock, this will result in 100kHz PWM frequency
    PG1PER = (2000 << 4);           //Time units are 1/16 of a PWM clock
    PG1DC = (500 << 4);             //25% duty cycle
    PG1PHASE = 0;                   //No Phase offset in rising edge of PWM

    PG1DTbits.DTH = (40 << 4);      //40 PWM clocks of dead time on PWM1H
    PG1DTbits.DTL = (40 << 4);      //40 PWM clocks of dead time on PWM1L
    PG1LEBbits.PHR = 1;             //Rising edge of PWM1H will trigger the
LEB counter
    PG1LEBbits.LEB = (80 << 4);     //80 PWM clocks of LEB

    //PCI logic configuration for current reset (PCI sync mode),
    //comparator 1 output (current reset signal) as PCI source,
    //and PWM1H falling edge as acceptance qualifier

    PG1EVTbits.PWMPCI = 0b000; //PWM Generator #1 output used as PCI signal

    PG1SPCIbits.TERM = 0b001;   //Terminate when PCI source transitions from
active to inactive
    PG1SPCIbits.TSYNCDIS = 1;   //Termination of latched PCI occurs immediately
    PG1SPCIbits.AQSS = 0b100;   //Inverted PWM1H is selected as acceptance
qualifier because PWM should be reset in OFF time
    PG1SPCIbits.AQPS = 1;       //Acceptance qualifier inverted to accept PCI
signal when PWM1H on time is over
    PG1SPCIbits.PSYNC = 0;      //PCI source is not synchronized to PWM EOC so
that current limit resets PWM immediately
    PG1SPCIbits.PSS = 0b11011;  //CMP1 out is selected as PCI source signal
    PG1SPCIbits.PPS = 1;        //PCI source signal is inverted
    PG1SPCIbits.ACP = 0b011;    //Latched PCI is selected as acceptance
criteria to work when CMP1 out is active
    PG1SPCIbits.TQSS = 0b000;   //No termination qualifier used so terminator
will work straight away without any qualifier


    //Enable PWM generator 1
    PG1CONbits.ON = 1;

}
```

14.6 Interrupts

The interrupt sources within the PWM system are routed to the CPU in one of two ways:

1. Through a shared signal for each PWM Generator (see 14.4.2.9.3. Event Interrupts). The signals include:

a. EOC event
b. TRIGA compare event
c. ADC Trigger 1 event
d. PCI_active event

2. Through the PWM Event Output block (see 14.4.3.6. PWM Event Outputs) are various sources of interrupts that can be generated by the PWM module.

For example, a device which has eight PWM Generators and six PWM event outputs would have a total of fourteen independent interrupt sources.

Since the PWM module can operate at a much higher frequency than the CPU system clock, care should be taken with the interrupt event configuration. Successive interrupt events on the same interrupt vector that occur at a rate greater than the CPU can detect and service them may result in missed processing and unexpected results. This limitation is dependent on the relationship of the system clock frequency, PWM operating frequency and the execution time of the Interrupt Service Routine (the number of instructions is irrelevant, what matters is how long the ISR takes to execute before it yields control back to the interrupted thread).

Interrupts from different sources that occur in close proximity to each other will also not be detected by the CPU as separate interrupt events. Therefore, it is good software practice to check the PWM status flags to differentiate a PCI interrupt event from a PWM time base interrupt. In some cases, it is desirable to separate different types of interrupt events that could be produced by the PWM Generator. When multiple PWM Generators have been synchronized together, it is possible to enable the time base interrupt on a separate PWM Generator that is synchronized to the host PWM Generator. Using this method, the time base interrupt can be serviced by a separate interrupt vector. It is also possible to bring various PWM event signals outside of the PWM module via the PWM Event blocks to an external off-chip destination (see 14.4.3.6. PWM Event Outputs).

14.7 Operation in Power-Saving Modes

This section discusses the operation of the High-Speed PWM module in Sleep mode and Idle mode.

14.7.1 Operation in Sleep Mode

When the device enters Sleep mode, the clocks available to the PWM are disabled. All enabled PWM output pins that were in operation prior to entering Sleep mode will be frozen in their current output states. If the application circuit can be damaged by this condition, the outputs must be placed into a safe state before executing the `PWRSAV` instruction to enter Sleep mode.

14.7.2 Operation in Idle Mode

When the device enters Idle mode, the CPU is no longer clocked; however, the PWM remains clocked and operational. If the PWM module is controlling a power conversion/motor control application, the action of putting the device into Idle mode will cause any control loops to be disabled, and most applications will likely experience issues unless they are explicitly designed to operate in an Open-Loop mode. It is recommended that the outputs be placed into a safe state before executing the `PWRSAV` instruction to enter Idle mode.

15. High-Speed, 12-Bit Low Latency ADC

dsPIC33AK128MC106 devices have two 12-bit High-Speed Successive Approximation Register (SAR) Analog-to-Digital Converter (ADC) cores that feature low conversion latency, high resolution and oversampling capabilities to improve performance in AC/DC and DC/DC power converters.

Each 12-bit ADC includes the following features:

- 12-bit Resolution
- Up to 40 Msps Conversion
- Up to 22 Analog Input Pins
- 20 Settings Channels. Each Channel:
  - Supports Discrete Configuration
  - Can be assigned to any analog input (I/O pin or internal signal)
  - Can be set to a different sampling time
  - Can be configured as single-ended or differential
  - Conversion result can be formatted as unsigned or signed
  - Conversion result can be left-aligned (fraction format)
  - Has a separate 32-bit conversion result register
- All Channels Support Four Sampling Modes:
  - Oversampling of multiple samples
  - Integration of multiple samples
  - Window (multiple samples are accumulated when the gate signal is active)
  - Single Conversion
- All channels have a digital comparator to detect when the conversion result is less than, greater than, or in bounds or out of bounds for the configurable thresholds
- Channels 17, 18 and 19 have a second result accumulator which can be used for a filter implementation
- Operational during CPU Sleep and Idle modes
- Band Gap Reference and Temperature Sensor Diode Inputs

15.1 Device-Specific Information

Table 15-1. ADC Summary

Number of Cores	Number of Channels	Max Input Clock	Clock Source	Peripheral Bus Speed
2	20	See Electrical Characteristics (ADC)	CLKGEN6	Fast

The number of available positive and negative analog inputs is dependent on package size, as shown in the table below.

Table 15-2. ADC External Input Availability

ADC Input	28-Pin	36-Pin	48-Pin	64-Pin	Comments
AD1ANN0	AV_{SS}				ADC 1 ground negative input 0 supporting differential mode
AD1ANN1	X	X	X	X	ADC 1 negative input 1 supporting differential mode

.....continued

ADC Input	28-Pin	36-Pin	48-Pin	64-Pin	Comments
AD1ANN2			X	X	ADC 1 negative input 2 supporting differential mode
AD1ANN3			X	X	ADC 1 negative input 3 supporting differential mode
AD1AN0	X	X	X	X	ADC 1 positive input 0
AD1AN1	X	X	X	X	ADC 1 positive input 1
AD1AN2		X	X	X	ADC 1 positive input 2
AD1AN3		X	X	X	ADC 1 positive input 3
AD1AN4	X	X	X	X	ADC 1 positive input 4
AD1AN5	X	X	X	X	ADC 1 positive input 5
AD1AN6			X	X	ADC 1 positive input 6
AD1AN7	X	X	X	X	ADC 1 positive input 7
AD1AN8			X	X	ADC 1 positive input 8
AD1AN9			X	X	ADC 1 positive input 9
AD1AN10				X	ADC 1 positive input 10
AD1AN11				X	ADC 1 positive input 11
AD1AN13	Internal				ADC 1 die temperature diode
AD1AN14	Internal				ADC 1 15/16*V_{DD} reference input
AD1AN15	Internal				ADC 1 Band Gap 0.8V reference input
AD2ANN0	AV_{SS}				ADC 2 ground negative input 0 supporting differential mode
AD2ANN1	X	X	X	X	ADC 2 negative input 1 supporting differential mode
AD2ANN2			X	X	ADC 2 negative input 2 supporting differential mode
AD2ANN3				X	ADC 2 negative input 3 supporting differential mode
AD2AN0	X	X	X	X	ADC 2 positive input 0
AD2AN1	X	X	X	X	ADC 2 positive input 1
AD2AN2		X	X	X	ADC 2 positive input 2
AD2AN3	X	X	X	X	ADC 2 positive input 3
AD2AN4	X	X	X	X	ADC 2 positive input 4
AD2AN5	X	X	X	X	ADC 2 positive input 5
AD2AN6	X	X	X	X	ADC 2 positive input 6
AD2AN7				X	ADC 2 positive input 7
AD2AN8			X	X	ADC 2 positive input 8
AD2AN9			X	X	ADC 2 positive input 9
AD2AN10				X	ADC 2 positive input 10
AD2AN11	Internal				ADC 2 V_{DDCORE} input
AD2AN12	Internal				ADC 2 V_{REG} input
AD2AN13	Internal				ADC 2 PLL V_{REG} input
AD2AN14	Internal				ADC 2 15/16*V_{DD} reference input
AD2AN15	Internal				ADC 2 Band Gap 0.8V reference input

Table 15-3. TRG1SRC Trigger Source Selection Bits

Value	Description
11111	ADTRG31 (PPS)
11110	PTG trigger 12
11101	CLC2 out
11100	CLC1 out
11011-11000	Reserved
10111	SCCP4 OCMP/ICAP out
10110	SCCP3 OCMP/ICAP out
10101	SCCP2 OCMP/ICAP out
10100	SCCP1 OCMP/ICAP out
10011	Reserved
10010	CLC4 out
10001	CLC3 out
10000	Reserved
01111	SCCP4 trigger out
01110	SCCP3 trigger out
01101	SCCP2 trigger out
01100	SCCP1 trigger out
01011	PWM4 ADC trigger 2
01010	PWM4 ADC trigger 1
01001	PWM3 ADC trigger 2
01000	PWM3 ADC trigger 1
00111	PWM2 ADC trigger 2
00110	PWM2 ADC trigger 1
00101	PWM1 ADC trigger 2
00100	PWM1 ADC trigger 1
00011-00010	Reserved
00001	Software trigger initiated using ADnSWTRG register
00000	Triggers are disabled

Table 15-4. TRG2SRC Trigger Source Selection Bits

Value	Description
11111	ADTRG31 (PPS) falling edge
11110	PTG trigger 12
11101	CLC2 out
11100	CLC1 out
11011-11000	Reserved
10111	SCCP4 OCMP/ICAP out
10110	SCCP3 OCMP/ICAP out
10101	SCCP2 OCMP/ICAP out
10100	SCCP1 OCMP/ICAP out
10011	Reserved
10010	CLC4 out
10001	CLC3 out
10000	Reserved
01111	SCCP4 trigger out

.....continued

Value	Description
01110	SCCP3 trigger out
01101	SCCP2 trigger out
01100	SCCP1 trigger out
01011	PWM4 ADC trigger 2
01010	PWM4 ADC trigger 1
01001	PWM3 ADC trigger 2
01000	PWM3 ADC trigger 1
00111	PWM2 ADC trigger 2
00110	PWM2 ADC trigger 1
00101	PWM1 ADC trigger 2
00100	PWM1 ADC trigger 1
00011	Conversion repeat timer trigger defined by PTCNT[5:0] (ADnCON[23:18]) bits
00010	Immediate re-trigger request
00001	Software trigger initiated using ADnSWTRG register
00000	Triggers are disabled

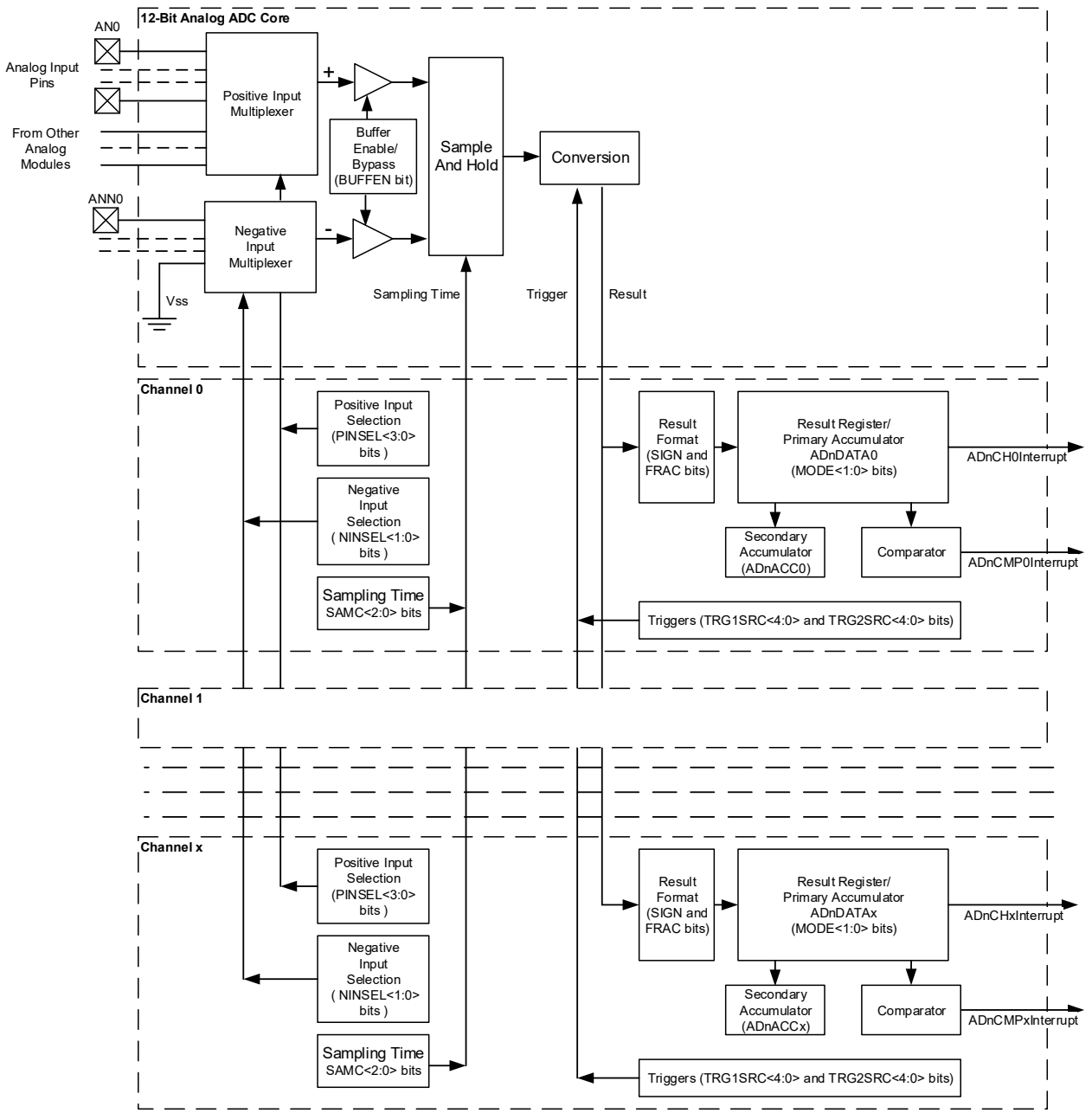
15.2 Architectural Overview

The analog inputs are connected through multiplexers and switches to the Sample-and-Hold (S&H) circuit of the ADC core. The core uses the settings channel information (the output format, the measurement mode and the input number) to process the analog sample. When conversion is complete, the result is stored in the result buffer for the specific channel and passed to the digital filter and digital comparator if they were configured.

The ADC provides each settings channel the ability to specify its own trigger source. This capability allows the ADC to sample and convert analog inputs that are associated with PWM Generators operating on independent time bases.

A simplified block diagram of the 12-bit ADC is illustrated in Figure 15-1.

Figure 15-1. ADC Module Block Diagram



Note: Band Gap Reference (VBG) is an internal analog input and is not available on device pins.

15.2.1 Temperature Sensor

The ADC channel that is connected to a forward biased diode can be used to measure die temperature. This diode provides a voltage output that can be monitored by the ADC.

The temperature coefficient and sampling timing requirements are listed in Table 37-41. To get the exact gain and offset numbers, the two temperature points calibration is recommended.

15.2.2 Band Gap Reference

The voltage reference options vary with devices. The Band Gap Reference (VBG) is an internal analog input and is not available on device pins. Additionally, the ADC module depends on the internal band gap circuit voltage. The voltage reference source cannot be changed when the module is enabled.

15.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0800	AD1CON	31:24	ADRDY	CALRDY	CALREQ	ACALEN	CALRATE[1:0]		OMODE[1:0]	
		23:16	RPTCNT[5:0]						RESERVED	STNDBY
		15:8	ON					TSTLOCK		TSTEN
		7:0		CALCNT[1:0]						
0x0804	AD1DATAOVR	31:24								
		23:16								
		15:8					DATAOVR[11:8]			
		7:0	DATAOVR[7:0]							
0x0808	AD1STAT	31:24	CH[31:24]RDY							
		23:16	CH[23:16]RDY							
		15:8	CH[15:8]RDY							
		7:0	CH[7:0]RDY							
0x080C	AD1CMPSTAT	31:24								
		23:16					CMP[19:16]FLG			
		15:8	CH[15:8]CMP							
		7:0	CH[7:0]CMP							
0x0810	AD1SWTRG	31:24	CH[31:24]TRG							
		23:16	CH[23:16]TRG							
		15:8	CH[15:8]TRG							
		7:0	CH[7:0]TRG							
0x0814	AD1CH0CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0818	AD1CH0DATA	31:24	DATA0[31:24]							
		23:16	DATA0[23:16]							
		15:8	DATA0[15:8]							
		7:0	DATA0[7:0]							
0x081C	AD1CH0CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0820	AD1CH0CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0824	AD1CH0CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0828 ... 0x082B	Reserved									
0x082C	AD1CH1CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0830	AD1CH1DATA	31:24	DATA1[31:24]							
		23:16	DATA1[23:16]							
		15:8	DATA1[15:8]							
		7:0	DATA1[7:0]							
0x0834	AD1CH1CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0838	AD1CH1CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x083C	AD1CH1CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0840 ... 0x0843	Reserved									
0x0844	AD1CH2CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0848	AD1CH2DATA	31:24	DATA2[31:24]							
		23:16	DATA2[23:16]							
		15:8	DATA2[15:8]							
		7:0	DATA2[7:0]							
0x084C	AD1CH2CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0850	AD1CH2CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0854	AD1CH2CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0858 ... 0x085B	Reserved									
0x085C	AD1CH3CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0860	AD1CH3DATA	31:24	DATA3[31:24]							
		23:16	DATA3[23:16]							
		15:8	DATA3[15:8]							
		7:0	DATA3[7:0]							
0x0864	AD1CH3CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0868	AD1CH3CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x086C	AD1CH3CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0870 ... 0x0873	Reserved									
0x0874	AD1CH4CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0878	AD1CH4DATA	31:24	DATA4[31:24]							
		23:16	DATA4[23:16]							
		15:8	DATA4[15:8]							
		7:0	DATA4[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x087C	AD1CH4CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0880	AD1CH4CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0884	AD1CH4CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0888 ... 0x088B	Reserved									
0x088C	AD1CH5CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0890	AD1CH5DATA	31:24	DATA5[31:24]							
		23:16	DATA5[23:16]							
		15:8	DATA5[15:8]							
		7:0	DATA5[7:0]							
0x0894	AD1CH5CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0898	AD1CH5CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x089C	AD1CH5CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x08A0 ... 0x08A3	Reserved									
0x08A4	AD1CH6CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x08A8	AD1CH6DATA	31:24	DATA6[31:24]							
		23:16	DATA6[23:16]							
		15:8	DATA6[15:8]							
		7:0	DATA6[7:0]							
0x08AC	AD1CH6CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x08B0	AD1CH6CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x08B4	AD1CH6CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x08B8 ... 0x08BB	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x08BC	AD1CH7CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x08C0	AD1CH7DATA	31:24	DATA7[31:24]							
		23:16	DATA7[23:16]							
		15:8	DATA7[15:8]							
		7:0	DATA7[7:0]							
0x08C4	AD1CH7CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x08C8	AD1CH7CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x08CC	AD1CH7CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x08D0 ... 0x08D3	Reserved									
0x08D4	AD1CH8CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x08D8	AD1CH8DATA	31:24	DATA8[31:24]							
		23:16	DATA8[23:16]							
		15:8	DATA8[15:8]							
		7:0	DATA8[7:0]							
0x08DC	AD1CH8CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x08E0	AD1CH8CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x08E4	AD1CH8CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x08E8 ... 0x08EB	Reserved									
0x08EC	AD1CH9CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x08F0	AD1CH9DATA	31:24	DATA9[31:24]							
		23:16	DATA9[23:16]							
		15:8	DATA9[15:8]							
		7:0	DATA9[7:0]							
0x08F4	AD1CH9CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x08F8	AD1CH9CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x08FC	AD1CH9CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0900 ... 0x0903	Reserved									
0x0904	AD1CH10CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0908	AD1CH10DATA	31:24	DATA0[31:24]							
		23:16	DATA0[23:16]							
		15:8	DATA0[15:8]							
		7:0	DATA0[7:0]							
0x090C	AD1CH10CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0910	AD1CH10CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0914	AD1CH10CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0918 ... 0x091B	Reserved									
0x091C	AD1CH11CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0920	AD1CH11DATA	31:24	DATA11[31:24]							
		23:16	DATA11[23:16]							
		15:8	DATA11[15:8]							
		7:0	DATA11[7:0]							
0x0924	AD1CH11CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0928	AD1CH11CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x092C	AD1CH11CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0930 ... 0x0933	Reserved									
0x0934	AD1CH12CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0938	AD1CH12DATA	31:24	DATA2[31:24]							
		23:16	DATA2[23:16]							
		15:8	DATA2[15:8]							
		7:0	DATA2[7:0]							
0x093C	AD1CH12CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0940	AD1CH12CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0944	AD1CH12CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0948 ... 0x094B	Reserved									
0x094C	AD1CH13CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0950	AD1CH13DATA	31:24	DATA3[31:24]							
		23:16	DATA3[23:16]							
		15:8	DATA3[15:8]							
		7:0	DATA3[7:0]							
0x0954	AD1CH13CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0958	AD1CH13CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x095C	AD1CH13CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0960 ... 0x0963	Reserved									
0x0964	AD1CH14CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0968	AD1CH14DATA	31:24	DATA4[31:24]							
		23:16	DATA4[23:16]							
		15:8	DATA4[15:8]							
		7:0	DATA4[7:0]							
0x096C	AD1CH14CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0970	AD1CH14CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0974	AD1CH14CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0978 ... 0x097B	Reserved									
0x097C	AD1CH15CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0980	AD1CH15DATA	31:24	DATA5[31:24]							
		23:16	DATA5[23:16]							
		15:8	DATA5[15:8]							
		7:0	DATA5[7:0]							
0x0984	AD1CH15CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0988	AD1CH15CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x098C	AD1CH15CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0990 ... 0x0993	Reserved									
0x0994	AD1CH16CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0998	AD1CH16DATA	31:24	DATA6[31:24]							
		23:16	DATA6[23:16]							
		15:8	DATA6[15:8]							
		7:0	DATA6[7:0]							
0x099C	AD1CH16CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x09A0	AD1CH16CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x09A4	AD1CH16CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x09A8 ... 0x09AB	Reserved									
0x09AC	AD1CH17CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x09B0	AD1CH17DATA	31:24	DATA7[31:24]							
		23:16	DATA7[23:16]							
		15:8	DATA7[15:8]							
		7:0	DATA7[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x09B4	AD1CH17CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x09B8	AD1CH17CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x09BC	AD1CH17CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x09C0	AD1CH17ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x09C4	AD1CH18CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x09C8	AD1CH18DATA	31:24	DATA8[31:24]							
		23:16	DATA8[23:16]							
		15:8	DATA8[15:8]							
		7:0	DATA8[7:0]							
0x09CC	AD1CH18CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x09D0	AD1CH18CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x09D4	AD1CH18CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x09D8	AD1CH18ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x09DC	AD1CH19CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x09E0	AD1CH19DATA	31:24	DATA9[31:24]							
		23:16	DATA9[23:16]							
		15:8	DATA9[15:8]							
		7:0	DATA9[7:0]							
0x09E4	AD1CH19CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x09E8	AD1CH19CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x09EC	AD1CH19CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x09F0	AD1CH19ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x09F4 ... 0x09FF	Reserved									
0x0A00	AD2CON	31:24	ADRDY	CALRDY	CALREQ	ACALEN	CALRATE[1:0]		OMODE[1:0]	
		23:16	RPTCNT[5:0]						RESERVED	STNDBY
		15:8	ON					TSTLOCK		TSTEN
		7:0		CALCNT[1:0]						
0x0A04	AD2DATAOVR	31:24								
		23:16								
		15:8					DATAOVR[11:8]			
		7:0	DATAOVR[7:0]							
0x0A08	AD2STAT	31:24	CH[31:24]RDY							
		23:16	CH[23:16]RDY							
		15:8	CH[15:8]RDY							
		7:0	CH[7:0]RDY							
0x0A0C	AD2CMPSTAT	31:24								
		23:16					CMP[19:16]FLG			
		15:8	CH[15:8]CMP							
		7:0	CH[7:0]CMP							
0x0A10	AD2SWTRG	31:24	CH[31:24]TRG							
		23:16	CH[23:16]TRG							
		15:8	CH[15:8]TRG							
		7:0	CH[7:0]TRG							
0x0A14	AD2CH0CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0A18	AD2CH0DATA	31:24	DATA0[31:24]							
		23:16	DATA0[23:16]							
		15:8	DATA0[15:8]							
		7:0	DATA0[7:0]							
0x0A1C	AD2CH0CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0A20	AD2CH0CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0A24	AD2CH0CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0A28 ... 0x0A2B	Reserved									
0x0A2C	AD2CH1CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0A30	AD2CH1DATA	31:24	DATA1[31:24]							
		23:16	DATA1[23:16]							
		15:8	DATA1[15:8]							
		7:0	DATA1[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0A34	AD2CH1CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0A38	AD2CH1CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0A3C	AD2CH1CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0A40 ... 0x0A43	Reserved									
0x0A44	AD2CH2CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0A48	AD2CH2DATA	31:24	DATA2[31:24]							
		23:16	DATA2[23:16]							
		15:8	DATA2[15:8]							
		7:0	DATA2[7:0]							
0x0A4C	AD2CH2CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0A50	AD2CH2CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0A54	AD2CH2CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0A58 ... 0x0A5B	Reserved									
0x0A5C	AD2CH3CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0A60	AD2CH3DATA	31:24	DATA3[31:24]							
		23:16	DATA3[23:16]							
		15:8	DATA3[15:8]							
		7:0	DATA3[7:0]							
0x0A64	AD2CH3CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0A68	AD2CH3CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0A6C	AD2CH3CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0A70 ... 0x0A73	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0A74	AD2CH4CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0A78	AD2CH4DATA	31:24	DATA4[31:24]							
		23:16	DATA4[23:16]							
		15:8	DATA4[15:8]							
		7:0	DATA4[7:0]							
0x0A7C	AD2CH4CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0A80	AD2CH4CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0A84	AD2CH4CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0A88 ... 0x0A8B	Reserved									
0x0A8C	AD2CH5CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0A90	AD2CH5DATA	31:24	DATA5[31:24]							
		23:16	DATA5[23:16]							
		15:8	DATA5[15:8]							
		7:0	DATA5[7:0]							
0x0A94	AD2CH5CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0A98	AD2CH5CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0A9C	AD2CH5CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0AA0 ... 0x0AA3	Reserved									
0x0AA4	AD2CH6CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0AA8	AD2CH6DATA	31:24	DATA6[31:24]							
		23:16	DATA6[23:16]							
		15:8	DATA6[15:8]							
		7:0	DATA6[7:0]							
0x0AAC	AD2CH6CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0AB0	AD2CH6CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0AB4	AD2CH6CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0AB8 ... 0x0ABB	Reserved									
0x0ABC	AD2CH7CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0AC0	AD2CH7DATA	31:24	DATA7[31:24]							
		23:16	DATA7[23:16]							
		15:8	DATA7[15:8]							
		7:0	DATA7[7:0]							
0x0AC4	AD2CH7CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0AC8	AD2CH7CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0ACC	AD2CH7CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0AD0 ... 0x0AD3	Reserved									
0x0AD4	AD2CH8CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0AD8	AD2CH8DATA	31:24	DATA8[31:24]							
		23:16	DATA8[23:16]							
		15:8	DATA8[15:8]							
		7:0	DATA8[7:0]							
0x0ADC	AD2CH8CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0AE0	AD2CH8CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0AE4	AD2CH8CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0AE8 ... 0x0AEB	Reserved									
0x0AEC	AD2CH9CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0AF0	AD2CH9DATA	31:24	DATA9[31:24]							
		23:16	DATA9[23:16]							
		15:8	DATA9[15:8]							
		7:0	DATA9[7:0]							
0x0AF4	AD2CH9CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0AF8	AD2CH9CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0AFC	AD2CH9CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0B00 ... 0x0B03	Reserved									
0x0B04	AD2CH10CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0B08	AD2CH10DATA	31:24	DATA10[31:24]							
		23:16	DATA10[23:16]							
		15:8	DATA10[15:8]							
		7:0	DATA10[7:0]							
0x0B0C	AD2CH10CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0B10	AD2CH10CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0B14	AD2CH10CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0B18 ... 0x0B1B	Reserved									
0x0B1C	AD2CH11CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0B20	AD2CH11DATA	31:24	DATA11[31:24]							
		23:16	DATA11[23:16]							
		15:8	DATA11[15:8]							
		7:0	DATA11[7:0]							
0x0B24	AD2CH11CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0B28	AD2CH11CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0B2C	AD2CH11CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0B30 ... 0x0B33	Reserved									
0x0B34	AD2CH12CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0B38	AD2CH12DATA	31:24	DATA1[31:24]							
		23:16	DATA1[23:16]							
		15:8	DATA1[15:8]							
		7:0	DATA1[7:0]							
0x0B3C	AD2CH12CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0B40	AD2CH12CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0B44	AD2CH12CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0B48 ... 0x0B4B	Reserved									
0x0B4C	AD2CH13CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0B50	AD2CH13DATA	31:24	DATA13[31:24]							
		23:16	DATA13[23:16]							
		15:8	DATA13[15:8]							
		7:0	DATA13[7:0]							
0x0B54	AD2CH13CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0B58	AD2CH13CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0B5C	AD2CH13CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0B60 ... 0x0B63	Reserved									
0x0B64	AD2CH14CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0B68	AD2CH14DATA	31:24	DATA14[31:24]							
		23:16	DATA14[23:16]							
		15:8	DATA14[15:8]							
		7:0	DATA14[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0B6C	AD2CH14CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0B70	AD2CH14CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0B74	AD2CH14CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0B78 ... 0x0B7B	Reserved									
0x0B7C	AD2CH15CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0B80	AD2CH15DATA	31:24	DATA15[31:24]							
		23:16	DATA15[23:16]							
		15:8	DATA15[15:8]							
		7:0	DATA15[7:0]							
0x0B84	AD2CH15CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0B88	AD2CH15CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0B8C	AD2CH15CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0B90 ... 0x0B93	Reserved									
0x0B94	AD2CH16CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0B98	AD2CH16DATA	31:24	DATA16[31:24]							
		23:16	DATA16[23:16]							
		15:8	DATA16[15:8]							
		7:0	DATA16[7:0]							
0x0B9C	AD2CH16CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0BA0	AD2CH16CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0BA4	AD2CH16CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0BA8 ... 0x0BAB	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0BAC	AD2CH17CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0BB0	AD2CH17DATA	31:24	DATA17[31:24]							
		23:16	DATA17[23:16]							
		15:8	DATA17[15:8]							
		7:0	DATA17[7:0]							
0x0BB4	AD2CH17CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0BB8	AD2CH17CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0BBC	AD2CH17CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0BC0	AD2CH17ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0BC4	AD2CH18CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0BC8	AD2CH18DATA	31:24	DATA18[31:24]							
		23:16	DATA18[23:16]							
		15:8	DATA18[15:8]							
		7:0	DATA18[7:0]							
0x0BCC	AD2CH18CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0BD0	AD2CH18CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0BD4	AD2CH18CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0BD8	AD2CH18ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0BDC	AD2CH19CON	31:24	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
		23:16	TRG2SRC[4:0]					CMPMOD[2:0]		
		15:8	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
		7:0	SAMC[2:0]			TRG1SRC[4:0]				
0x0BE0	AD2CH19DATA	31:24	DATA19[31:24]							
		23:16	DATA19[23:16]							
		15:8	DATA19[15:8]							
		7:0	DATA19[7:0]							
0x0BE4	AD2CH19CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0BE8	AD2CH19CMPLO	31:24	LO[31:24]							
		23:16	LO[23:16]							
		15:8	LO[15:8]							
		7:0	LO[7:0]							
0x0BEC	AD2CH19CMPHI	31:24	HI[31:24]							
		23:16	HI[23:16]							
		15:8	HI[15:8]							
		7:0	HI[7:0]							
0x0BF0	AD2CH19ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							

15.3.1 ADC n Control Register

Name: ADnCON
Offset: 0x800, 0xA00

Notes:

1. Timing is approximate and dependent on the 32K oscillator accuracy. Changing this value during ADC operation may cause erratic re-calibration timing.
2. Recovery from Standby mode requires 230 ADC clock cycles.
3. Set the ADON bit only after the ADC module has been configured. Changing ADC configuration bits when ADON = 1 will result in unpredictable behavior.

Legend: n = ADC number, HS = Hardware Settable bit, HC = Hardware Clearable bit, R = Readable bit, W = Writable bit, S = Set Only bit, C = Clear Only bit

Bit	31	30	29	28	27	26	25	24
	ADRDY	CALRDY	CALREQ	ACALEN	CALRATE[1:0]		OMODE[1:0]	
Access	HS/HC/R	HS/HC/R	R/W/HC	R/W	R/W	R/W	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RPTCNT[5:0]						RESERVED	STNDBY
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8
	ON					TSTLOCK		TSTEN
Access	R/W					R/S		R/W/C
Reset	0					0		0

Bit	7	6	5	4	3	2	1	0
		CALCNT[1:0]						
Access		R/W	R/W					
Reset		0	0					

Bit 31 – ADRDY ADC Ready bit

The bit indicates that the ADC has been enabled and has completed its power-up and self-calibration process.

Value	Description
1	ADC is ready
0	ADC is off

Bit 30 – CALRDY Calibration Done bit

Value	Description
1	Calibration cycle has finished
0	Calibration was not started or is in progress

Bit 29 – CALREQ Software Calibration Cycle Request bit

Value	Description
1	Setting this bit executes the calibration cycle
0	Calibration cycle is not requested

Bit 28 – ACALEN This bit enables periodic ADC re-calibration. The calibration cycles period is defined by CALRATE (ADnCON[27:26]) bits.

Value	Description
1	Periodic recalibration is enabled
0	Periodic recalibration is off

Bits 27:26 – CALRATE[1:0] Auto Re-Calibration Period bits (1)

Value	Description
11	Recalibration every 4096 seconds
10	Recalibrate every 1024 seconds
01	Recalibrate every 64 seconds
00	Recalibrate every second

Bits 25:24 – OMODE[1:0] ADC Operation Mode Status bits

Value	Description
1x	ADC is on
01	ADC is in standby mode
00	ADC is powered down

Bits 23:18 – RPTCNT[5:0] Conversion Repeat Timer Period bits
This timer can be used to generate ADC triggers periodically by selecting the RPTCNT timer as a trigger source in TRG1SRC (AdnCHCONx[4:0]) or TRG2SRC (ADnCHCONx[23:19]) bits. This timer counts ADC clock cycles.

Value	Description
111111	64 ADC clock cycles between triggers
...	
0000010	3 ADC clock cycles between triggers
000001	2 ADC clock cycles between triggers
000000	1 ADC clock cycle between triggers

Bit 17 – RESERVED

Bit 16 – STNDBY ADC Standby Enable bit(2)

Value	Description
1	ADC module is in a power reduced mode
0	ADC is in normal active mode

Bit 15 – ON ADC Enable bit(3)

Value	Description
1	ADC module is enabled
0	ADC module is disabled

Bit 10 – TSTLOCK TSTEN (ADnCON[8]) Lock bit

Value	Description
1	TSTEN bit cannot be set to 1 but can be cleared to 0
0	TSTEN bit can be set to 1

Bit 8 – TSTEN Test Mode Enable bit
In the test mode the result of a conversion for all channels is overwritten with a value from ADnDATAOVR register.

Value	Description
1	The test mode is enabled
0	The test mode is disabled

Bits 6:5 – CALCNT[1:0] ADC Idle Cycles Prior to Calibration bits

Value	Description
11	Wait for 16 activity free ADC clock cycles before initiating requested calibration
10	Wait for 8 activity free ADC clock cycles before initiating requested calibration
01	Wait for 4 activity free ADC clock cycles before initiating requested calibration
00	Wait for 2 activity free ADC clock cycles before initiating requested calibration

15.3.2 ADC n Test Mode Data Register

Name: ADnDATAOVR
Offset: 0x804, 0xA04

Legend: n = ADC number, R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					DATAOVR[11:8]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATAOVR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 11:0 – DATAOVR[11:0] Conversion Result Value in Test Mode bits

15.3.3 ADC n Result Ready Flags Register

Name: ADnSTAT
Offset: 0x808, 0xA08

Note: Each bit in this register is set by hardware when the corresponding channel x conversion result is written into ADnDATAx register. The bit is cleared by hardware when ADnDATAx register is read by software.

Legend: n = ADC number, HS = Hardware Settable bit, HC = Hardware Clearable bit

Bit	31	30	29	28	27	26	25	24
	CH[31:24]RDY							
Access	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CH[23:16]RDY							
Access	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CH[15:8]RDY							
Access	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CH[7:0]RDY							
Access	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – CH[31:24]RDY Channel x Conversion Result Ready bit

Bits 23:16 – CH[23:16]RDY Channel x Conversion Result Ready bit

Bits 15:8 – CH[15:8]RDY Channel x Conversion Result Ready bit

Bits 7:0 – CH[7:0]RDY Channel x Conversion Result Ready bit

15.3.4 ADC n Comparators Status Register

Name: ADnCMPSTAT
Offset: 0x80C, 0xA0C

These bits are set when ADC channel data meets comparison criteria and cleared when a "0" is written to this bit by software.

Legend: n = ADC number, HS = Hardware Settable bit, C = Clear Only bit, R = Readable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					CMP[19:16]FLG			
Access					HS/C/R	HS/C/R	HS/C/R	HS/C/R
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CH[15:8]CMP							
Access	HS/C/R	HS/C/R	HS/C/R	HS/C/R	HS/C/R	HS/C/R	HS/C/R	HS/C/R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CH[7:0]CMP							
Access	HS/C/R	HS/C/R	HS/C/R	HS/C/R	HS/C/R	HS/C/R	HS/C/R	HS/C/R
Reset	0	0	0	0	0	0	0	0

Bits 19:16 – CMP[19:16]FLG Channel x Comparator Event Detection bit

Bits 15:8 – CH[15:8]CMP Channel x Comparator Event Detection bit

Bits 7:0 – CH[7:0]CMP Channel x Comparator Event Detection bit

15.3.5 ADC n Software Triggers Request Register

Name: ADnSWTRG
Offset: 0x810, 0xA10

Note: A software trigger for channel x is generated when the corresponding bit is set in this register. The software trigger must be selected for the channel in TRG1SRC (ADnCHCONx[4:0]) or TRG2SRC (ADnCHCONx[23:19]) bits.

Legend: n = ADC number, W = Writable bit, HC = Hardware Clearable bit, R = Readable bit

Bit	31	30	29	28	27	26	25	24
	CH[31:24]TRG							
Access	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CH[23:16]TRG							
Access	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CH[15:8]TRG							
Access	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CH[7:0]TRG							
Access	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – CH[31:24]TRG Channel x Software Trigger Request bit

Bits 23:16 – CH[23:16]TRG Channel x Software Trigger Request bit

Bits 15:8 – CH[15:8]TRG Channel x Software Trigger Request bit

Bits 7:0 – CH[7:0]TRG Channel x Software Trigger Request bit

15.3.6 ADC 1 Channel x Control Register

Name: AD1CHxCON
Offset: 0x814, 0x82C, 0x844, 0x85C, 0x874, 0x88C, 0x8A4, 0x8BC, 0x8D4, 0x8EC, 0x904, 0x91C, 0x934, 0x94C, 0x964, 0x97C, 0x994, 0x9AC, 0x9C4, 0x9DC

Legend: n = ADC number, x = Channel number, R = Readable bit, W = Writable bit

Note:

1. These bits are used with Oversampling mode only when MODE[1:0] bits = '11' and are ignored for all other sampling modes.
2. For the correct operation this bit must be used only for the triggers TRG1SRC[4:0] = '00100' – '11111'.
3. Early interrupts are only available for single sample operations (MODE[1:0] bits = '00'). Multi-sample operations ignore the EIEN bit and use normal interrupt timing. Do not use early interrupt if using DMA transfers; the data might not be ready. The early interrupt is asserted at the start of sampling.

Bit	31	30	29	28	27	26	25	24
	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRG2SRC[4:0]					CMPMOD[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SAMC[2:0]			TRG1SRC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:30 – MODE[1:0] Sampling Mode Selection bits

Value	Description
11	Oversampling of multiple samples defined by ACCNUM[1:0] bits. The first conversion is initiated by TRG1SRC[4:0] trigger and all other conversions are executed by TRG2SRC[4:0] trigger.
10	Integration of multiple samples defined by: CNTx[15:0] bits (ADnCNTx[15:0]). The first conversion is initiated by TRG1SRC[4:0] trigger and all other conversions are executed by TRG2SRC[4:0] trigger.
01	Window gated by TRG1SRC[4:0] source. In this mode the samples are accumulated when a signal selected by TRG1SRC[4:0] bits has an active level. All conversions are initiated by TRG2SRC[4:0] trigger. The number of conversions is limited by CNTx[15:0] bits (ADnCNTx[15:0]).
00	Single sample initiated by TRG1SRC[4:0] trigger

Bits 29:28 – ACCNUM[1:0] Oversampling Mode Number of Samples Selection bits (1)

Value	Description
11	256 samples, 16 bits result in ADnDATAx register
10	64 samples, 15 bits result in ADnDATAx register

Value	Description
01	16 samples, 14 bits result in ADnDATAx register
00	4 samples, 13 bits result in ADnDATAx register

Bit 27 – ACCBRST Oversampling Burst Mode Enable bit (1)

Value	Description
1	The oversampling is performed as a continuous non-interruptible burst, during which all other conversion requests are blocked out until the process is completed.
0	Oversampling can be interrupted by a high priority conversion request

Bit 26 – ACCRO Accumulator Roll-Over Enable bit
The Roll-Over must be enabled when the ADnACCx register is used

Value	Description
1	ADnDATAx accumulator is not cleared; it is allowed to roll-over
0	ADnDATAx accumulator is cleared at the end of a multi-sample sequence

Bit 25 – TRG1POL Starting Trigger Polarity Selection bit (2)

Value	Description
1	Active level of the signal selected by TRG1SRC[4:0] bits is low; a falling edge generates a conversion request
0	Active level of the signal selected by TRG1SRC[4:0] bits is high; a rising edge generates a conversion request

Bit 24 – EIEN Early Interrupt Enable bit (3)

Value	Description
1	Early interrupt is enabled
0	Normal interrupt timing

Bits 23:19 – TRG2SRC[4:0] Multi-Sample Conversions Re-Trigger Source Selection bits
See Table 15-4 for device-specific selections.

Bits 18:16 – CMPMOD[2:0] Comparison Criteria Selection bits

Value	Description
111-101	Comparison disabled
100	Conversion result is less or equal to (≤) ADnCMPLOx register
011	Conversion result is greater than (>) ADnCMPLOx register
010	Conversion result is in bounds of (≥) ADnCMPLOx and (≤) ADnCMPHIx
001	Conversion result is out of bounds of (<) ADnCMPLOx or (>) ADnCMPHIx
000	Comparison disabled

Bit 15 – DIFF Signed Result Format Enable bit

Value	Description
1	Differential Input mode; data is output as signed (two's complement)
0	Single-Ended Input mode; data is output as unsigned

Bits 14:11 – PINSEL[3:0] Analog Positive Input Selection for ADC bits

Value	Description
1111	ADxAN15
...	
0001	ADxAN1
0000	ADxAN0

Bit 10 – LEFT Fractional Data Output Format Enable bit

Value	Description
1	Result in ADnDATAx register is aligned to the left (in the fractional format)
0	Result in ADnDATAx register is aligned to the right

Bits 9:8 – NINSEL[1:0] Negative Analog Input Selection bit

Value	Description
11	ADxANN3
10	ADxANN2
01	ADxANN1
00	Ground (V_{SS}); single-ended mode

Bits 7:5 – SAMC[2:0] Sampling Time Selection bits (times listed for 80 MHz TAD clock)

Value	Description
111	14.5 T_{AD} (181.25 nS)
110	12.5 T_{AD} (156.25 nS)
101	10.5 T_{AD} (131.25 nS)
100	8.5 T_{AD} (106.25 nS)
011	6.5 T_{AD} (81.25 nS)
010	4.5 T_{AD} (56.25 nS)
001	2.5 T_{AD} (31.25 nS)
000	0.5 T_{AD} (6.25 nS)

Bits 4:0 – TRG1SRC[4:0] Trigger Source Selection bits
See Table 15-3 for device-specific selections.

15.3.7 ADC 1 Channel x Result Register

Name: AD1CHxDATA
Offset: 0x818, 0x830, 0x848, 0x860, 0x878, 0x890, 0x8A8, 0x8C0, 0x8D8, 0x8F0, 0x908, 0x920, 0x938, 0x950, 0x968, 0x980, 0x998, 0x9B0, 0x9C8, 0x9E0

Legend: n = ADC number, x = Channel number, R = Readable bit

Bit	31	30	29	28	27	26	25	24
	DATAx[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATAx[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATAx[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATAx[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATAx[31:0] Conversion Result/Primary Accumulator

15.3.8 ADC 1 Channel x Counter Register

Name: AD1CHxCNT
Offset: 0x81C, 0x834, 0x84C, 0x864, 0x87C, 0x894, 0x8AC, 0x8C4, 0x8DC, 0x8F4, 0x90C, 0x924, 0x93C, 0x954, 0x96C, 0x984, 0x99C, 0x9B4, 0x9CC, 0x9E4

Legend: n = ADC number, x = Channel number, HS = Hardware Settable bit, HC = Hardware Clearable bit, R = Readable bit

Bit	31	30	29	28	27	26	25	24
	CNTSTAT[15:8]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CNTSTAT[7:0]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CNTSTAT[15:0]

Number of conversions done in Integration (MODE[1:0] bits = '10') and Window (MODE[1:0] bits = '01') Sampling modes

Bits 15:0 – CNT[15:0]

Number of samples for an Integration Sampling mode (MODE[1:0] bits = '10') and maximum number of samples for a Window Sampling mode (MODE[1:0] bits = '01')

15.3.9 ADC 1 Channel x Low Compare Register

Name: AD1CHxCMPLO
Offset: 0x820, 0x838, 0x850, 0x868, 0x880, 0x898, 0x8B0, 0x8C8, 0x8E0, 0x8F8, 0x910, 0x928, 0x940, 0x958, 0x970, 0x988, 0x9A0, 0x9B8, 0x9D0, 0x9E8

Legend: n = ADC number, x = Channel number, R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	LO[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LO[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LO[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LO[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – LO[31:0] Low Limit Comparator Value

15.3.10 ADC 1 Channel x High Compare Register

Name: AD1CHxCMPHI
Offset: 0x824, 0x83C, 0x854, 0x86C, 0x884, 0x89C, 0x8B4, 0x8CC, 0x8E4, 0x8FC, 0x914, 0x92C, 0x944, 0x95C, 0x974, 0x98C, 0x9A4, 0x9BC, 0x9D4, 0x9EC

Legend: n = ADC number, x = Channel number, R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	HI[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HI[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HI[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HI[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HI[31:0] High Limit Comparator Value

15.3.11 ADC 1 Channel x Secondary Accumulator Register

Name: AD1CHxACC
Offset: 0x9C0, 0x9D8, 0x9F0

Legend: n = ADC number, x = Channel number, R = Readable bit

Bit	31	30	29	28	27	26	25	24
	ACC[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ACC[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACC[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ACC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ACC[31:0] Secondary Accumulator

15.3.12 ADC 2 Channel x Control Register

Name: AD2CHxCON
Offset: 0xA14, 0xA2C, 0xA44, 0xA5C, 0xA74, 0xA8C, 0xAA4, 0xABC, 0xAD4, 0xAEC, 0xB04, 0xB1C, 0xB34, 0xB4C, 0xB64, 0xB7C, 0xB94, 0xBAC, 0xBC4, 0xBDC

Legend: n = ADC number, x = Channel number, R = Readable bit, W = Writable bit

Note:

1. These bits are used with Oversampling mode only when MODE[1:0] bits = '11' and are ignored for all other sampling modes.
2. For the correct operation this bit must be used only for the triggers TRG1SRC[4:0] = '00100' – '11111'.
3. Early interrupts are only available for single sample operations (MODE[1:0] bits = '00'). Multi-sample operations ignore the EIEN bit and use normal interrupt timing. Do not use early interrupt if using DMA transfers; the data might not be ready. The early interrupt is asserted at the start of sampling.

Bit	31	30	29	28	27	26	25	24
	MODE[1:0]		ACCNUM[1:0]		ACCBRST	ACCRO	TRG1POL	EIEN
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRG2SRC[4:0]					CMPMOD[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DIFF	PINSEL[3:0]				LEFT	NINSEL[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SAMC[2:0]			TRG1SRC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:30 – MODE[1:0] Sampling Mode Selection bits

Value	Description
11	Oversampling of multiple samples defined by ACCNUM[1:0] bits. The first conversion is initiated by TRG1SRC[4:0] trigger and all other conversions are executed by TRG2SRC[4:0] trigger.
10	Integration of multiple samples defined by: CNTx[15:0] bits (ADnCNTx[15:0]). The first conversion is initiated by TRG1SRC[4:0] trigger and all other conversions are executed by TRG2SRC[4:0] trigger.
01	Window gated by TRG1SRC[4:0] source. In this mode the samples are accumulated when a signal selected by TRG1SRC[4:0] bits has an active level. All conversions are initiated by TRG2SRC[4:0] trigger. The number of conversions is limited by CNTx[15:0] bits (ADnCNTx[15:0]).
00	Single sample initiated by TRG1SRC[4:0] trigger

Bits 29:28 – ACCNUM[1:0] Oversampling Mode Number of Samples Selection bits (1)

Value	Description
11	256 samples, 16 bits result in ADnDATAx register
10	64 samples, 15 bits result in ADnDATAx register

Value	Description
01	16 samples, 14 bits result in ADnDATAx register
00	4 samples, 13 bits result in ADnDATAx register

Bit 27 – ACCBRST Oversampling Burst Mode Enable bit (1)

Value	Description
1	The oversampling is performed as a continuous non-interruptible burst, during which all other conversion requests are blocked out until the process is completed.
0	Oversampling can be interrupted by a high priority conversion request

Bit 26 – ACCRO Accumulator Roll-Over Enable bit
The Roll-Over must be enabled when the ADnACCx register is used

Value	Description
1	ADnDATAx accumulator is not cleared; it is allowed to roll-over
0	ADnDATAx accumulator is cleared at the end of a multi-sample sequence

Bit 25 – TRG1POL Starting Trigger Polarity Selection bit (2)

Value	Description
1	Active level of the signal selected by TRG1SRC[4:0] bits is low; a falling edge generates a conversion request
0	Active level of the signal selected by TRG1SRC[4:0] bits is high; a rising edge generates a conversion request

Bit 24 – EIEN Early Interrupt Enable bit (3)

Value	Description
1	Early interrupt is enabled
0	Normal interrupt timing

Bits 23:19 – TRG2SRC[4:0] Multi-Sample Conversions Re-Trigger Source Selection bits
See Table 15-4 for device-specific selections.

Bits 18:16 – CMPMOD[2:0] Comparison Criteria Selection bits

Value	Description
111-101	Comparison disabled
100	Conversion result is less or equal to (≤) ADnCMPLOx register
011	Conversion result is greater than (>) ADnCMPLOx register
010	Conversion result is in bounds of (≥) ADnCMPLOx and (≤) ADnCMPHIx
001	Conversion result is out of bounds of (<) ADnCMPLOx or (>) ADnCMPHIx
000	Comparison disabled

Bit 15 – DIFF Signed Result Format Enable bit

Value	Description
1	Differential Input mode; data is output as signed (two's complement)
0	Single-Ended Input mode; data is output as unsigned

Bits 14:11 – PINSEL[3:0] Analog Positive Input Selection for ADC bits

Value	Description
1111	ADxAN15
...	
0001	ADxAN1
0000	ADxAN0

Bit 10 – LEFT Fractional Data Output Format Enable bit

Value	Description
1	Result in ADnDATAx register is aligned to the left (in the fractional format)
0	Result in ADnDATAx register is aligned to the right

Bits 9:8 – NINSEL[1:0] Negative Analog Input Selection bit

Value	Description
11	ADxANN3
10	ADxANN2
01	ADxANN1
00	Ground (V_{SS}); single-ended mode

Bits 7:5 – SAMC[2:0] Sampling Time Selection bits

Value	Description
111	14.5 T_{AD} (181.25 nS)
110	12.5 T_{AD} (156.25 nS)
101	10.5 T_{AD} (131.25 nS)
100	8.5 T_{AD} (106.25 nS)
011	6.5 T_{AD} (81.25 nS)
010	4.5 T_{AD} (56.25 nS)
001	2.5 T_{AD} (31.25 nS)
000	0.5 T_{AD} (6.25 nS)

Bits 4:0 – TRG1SRC[4:0] Trigger Source Selection bits
See Table 15-3 for device-specific selections.

15.3.13 ADC 2 Channel x Result Register

Name: AD2CHxDATA
Offset: 0xA18, 0xA30, 0xA48, 0xA60, 0xA78, 0xA90, 0xAA8, 0xAC0, 0xAD8, 0xAF0, 0xB08, 0xB20, 0xB38, 0xB50, 0xB68, 0xB80, 0xB98, 0xBB0, 0xBC8, 0xBE0

Legend: n = ADC number, x = Channel number, R = Readable bit

Bit	31	30	29	28	27	26	25	24
	DATAx[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATAx[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATAx[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATAx[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATAx[31:0] Conversion Result/Primary Accumulator

15.3.14 ADC 2 Channel x Counter Register

Name: AD2CHxCNT
Offset: 0xA1C, 0xA34, 0xA4C, 0xA64, 0xA7C, 0xA94, 0xAAC, 0xAC4, 0xADC, 0xAF4, 0xB0C, 0xB24, 0xB3C, 0xB54, 0xB6C, 0xB84, 0xB9C, 0xBB4, 0xBCC, 0xBE4

Legend: n = ADC number, x = Channel number, HS = Hardware Settable bit, HC = Hardware Clearable bit, R = Readable bit

Bit	31	30	29	28	27	26	25	24
	CNTSTAT[15:8]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CNTSTAT[7:0]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CNTSTAT[15:0]

Number of conversions done in Integration (MODE[1:0] bits = '10') and Window (MODE[1:0] bits = '01') Sampling modes

Bits 15:0 – CNT[15:0]

Number of samples for an Integration Sampling mode (MODE[1:0] bits = '10') and maximum number of samples for a Window Sampling mode (MODE[1:0] bits = '01')

15.3.15 ADC 2 Channel x Low Compare Register

Name: AD2CHxCMPLO
Offset: 0xA20, 0xA38, 0xA50, 0xA68, 0xA80, 0xA98, 0xAB0, 0xAC8, 0xAE0, 0xAF8, 0xB10, 0xB28, 0xB40, 0xB58, 0xB70, 0xB88, 0xBA0, 0xBB8, 0xBD0, 0xBE8

Legend: n = ADC number, x = Channel number, R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	LO[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LO[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LO[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LO[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – LO[31:0] Low Limit Comparator Value

15.3.16 ADC 2 Channel x High Compare Register

Name: AD2CHxCMPHI
Offset: 0xA24, 0xA3C, 0xA54, 0xA6C, 0xA84, 0xA9C, 0xAB4, 0xACC, 0xAE4, 0xAFC, 0xB14, 0xB2C, 0xB44, 0xB5C, 0xB74, 0xB8C, 0xBA4, 0xBBC, 0xBD4, 0xBEC

Legend: n = ADC number, x = Channel number, R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	HI[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HI[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HI[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HI[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HI[31:0] High Limit Comparator Value

15.3.17 ADC 2 Channel x Secondary Accumulator Register

Name: AD2CHxACC
Offset: 0xBC0, 0xBD8, 0xBF0

Legend: n = ADC number, x = Channel number, R = Readable bit

Bit	31	30	29	28	27	26	25	24
	ACC[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ACC[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACC[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ACC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ACC[31:0] Secondary Accumulator

15.4 ADC Operation

The Analog-to-Digital conversion is performed in the following three steps:

1. Sampling of the input signal.
2. Capturing of the input signal (holding) and transferring to the converter.
3. Conversion of the analog signal to its digital representation.

During the sampling of the input signal, the Sample-and-Hold (S/H) circuit capacitor is charged. The sampling time must be adequate so that the capacitor charges to a value equal to the input voltage. At the appropriate time, the input is disconnected from the capacitor, and subsequently, the analog voltage is transferred to the converter. The converter then digitizes the analog signal and provides the result.

15.4.1 Channels

The channel is a group of settings controlling the conversion process. The ADC has up to 20 channels. All channel settings are placed in the ADnCHCONx register (where x is a channel number). To resolve simultaneous requests for channel conversions between different channels, the fixed order priority scheme is used. The priority scheme is defined by the channel number, with the channel with lowest number receiving the highest priority. In other words, channel conversions occur in ascending order of the channel number, with the lowest channel number being converted first.

15.4.2 Analog Inputs

Any analog input (external pin or internal analog signal) can be mapped to any channel by PINSEL[3:0] bits (ADnCHCONx[14:11]) for a positive input and by NINSEL[1:0] bits (ADnCHCONx[9:8])

for a negative input. The input can be configured as a single-ended or differential using DIFF bit (ADnCHCONx[15]).

The ANSELx registers for the I/O ports, associated with the analog input, are used to configure the corresponding pins as analog pins. A pin is configured as an analog input when the corresponding ANSELx bit = 1. When the ANSELx bit = 0, the pin is set to a digital control. The ANSELx bits are set when the device comes out of Reset, causing the ADC input pins to be configured as analog inputs by default. The TRISx registers control the digital function of the port pins. The port pin that is required as an analog input must have its corresponding bit set in the specific TRISx register, configuring the pin as an input. If the I/O pin associated with an ADC input is configured as an output by clearing the TRISx bit, the port's digital output level will be converted. After a device Reset, all TRISx bits are set. The PORT register bit reads as '0' if its corresponding pin is configured as an analog input. For more information on port pin configuration, refer to the **"I/O Ports"** chapter.

15.4.3 Sampling and Conversion Timing

Each channel can be configured for a different sampling time using SAMC[2:0] bits (ADnCHCONx[7:5]) from 6.25 nS to 181.25 nS.

The input ADC module clock is divided by four to get the analog ADC core clock (TAD). 1.5 ADC clock cycles are needed to complete conversion.

15.4.3.1 Sampling Time Requirements

The analog input model of the ADC is illustrated in Figure 15-2.

It takes time to charge the Holding Capacitor (C_{HOLD}) to the input signal level through the source and internal device resistance.

The total acquisition time for the Analog-to-Digital conversion is a function of the Holding Capacitor (C_{HOLD}) charge time. For the ADC module to meet its specified accuracy, the Holding Capacitor (C_{HOLD}) must be allowed to fully charge to the voltage level on the analog input pin. The Signal Source Impedance (R_S) and the Interconnect Impedance (R_{IC}) combine to directly affect the time required to charge the C_{HOLD}. The combined impedance ($R_{TOTAL} = R_S + R_{IC}$) must therefore be small enough to fully charge the Holding Capacitor within the selected sample time. To charge the C_{HOLD} with 0.5 LSB error, the sampling time should be more than the time defined by the Minimum Sampling Time equation below.

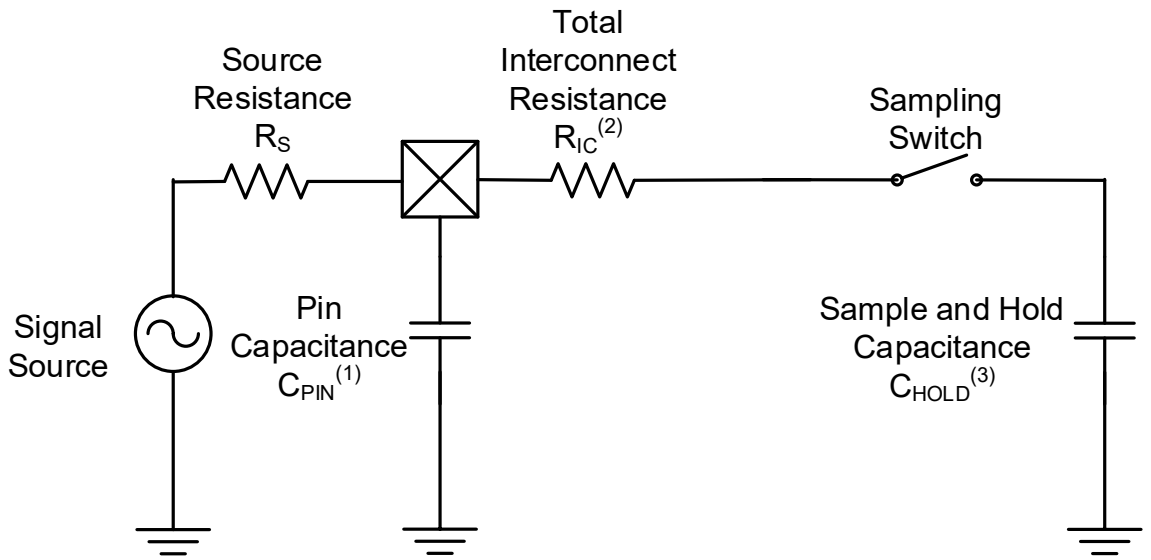
Equation 15-1. Minimum Sampling Time

$$T_{SAMPLING} = R_{TOTAL} \times C_{HOLD} \times \ln(2^{(RESOLUTION + 1)}) \text{ or}$$

For 12-Bit Resolution:

$$T_{SAMPLING} = 9 \times R_{TOTAL} \times C_{HOLD}$$

Figure 15-2. ADC Input Model



Notes:

1. The C_{PIN} value depends on the device package and is not tested.
2. See R_{IC} value in the Electrical Specifications.
3. See C_{HOLD} value in the Electrical Specifications.

15.4.4 Conversions

Each channel can be triggered to do a single conversion or accumulate results of the multiple conversions. The conversion mode is selected by MODE[1:0] bits (ADnCHCONx[31:30]). Depending on the conversion mode, the channel conversions are initiated by triggers defined in TRG1SRC[4:0] and TRG2SRC[4:0] bits (ADnCHCONx[4:0] and ADnCHCONx[23:19]).

The following conversion modes are available:

- **Single conversion (MODE[1:0] bits = '00').** The conversion is initiated by TRG1SRC[4:0] trigger. When the conversion is finished the conversion result is stored in the ADnDATAx register, the CHxRDY bit in the ADnSTAT register is set and the ADnCHxIF interrupt flag in the corresponding IFCx register is set.
- **Window conversions (MODE[1:0] bits = '01').** In this mode, the conversion results are accumulated in the ADnDATAx register (primary accumulator) when a signal selected by the TRG1SRC[4:0] bits has an active level. The active level can be changed by the TRG1POL bit (ADnCHCONx[25]). All conversions are initiated by a trigger selected in the TRG2SRC[4:0] bits (ADnCHCONx[23:19]). The number of conversions is limited by the CNTx[15:0] bits (ADnCNTx[15:0]). The number of accumulated conversion results is available in the CNTSTATx[15:0] bits (ADnCNT[31:16]). CHxRDY bit in the ADnSTAT register and the ADnCHxIF interrupt flag in the corresponding IFCx register are set when the gating signal selected by the TRG1SRC[4:0] bits is deactivated or when the number of accumulated conversion results reaches a limit defined by the CNTx[15:0] bits.
- **Integration (MODE[1:0] bits = '10').** In this mode, the number of conversion results defined by the CNTx[15:0] bits (ADnCNTx[15:0]) are accumulated in the ADnDATAx register (primary accumulator). The first conversion is initiated by a trigger selected by the TRG1SRC[4:0] bits and all other conversions are executed by a trigger selected by TRG2SRC[4:0] bits. When the number of accumulated conversion results reaches a value in the CNTx[15:0] bits (ADnCNTx[15:0]), the CHxRDY bit in ADnSTAT register and the ADnCHxIF interrupt flag in the corresponding IFCx register are set.

- **Oversampling (MODE[1:0] bits = '11').** In this mode, the number of conversion results defined by the ACCNUM[1:0] bits (ADnCHCONx[21:20]) are accumulated in the ADnDATAx register (primary accumulator). The first conversion is initiated by a trigger selected by the TRG1SRC[4:0] bits and all other conversions are executed by a trigger selected by TRG2SRC[4:0] bits. When the number of the accumulated conversion results reaches a value specified in the ACCNUM[21:20] bits, the CHxRDY bit in the ADnSTAT register and the ADnCHxIF interrupt flag in the corresponding IFCx register are set. The oversampling process can be delayed by high priority channel conversions. The ACCBRST bit (ADnCHCONx[27]) can disable the interruption by other high priority channels. When the ACCBURST bit is set ('1'), all other conversions will be suspended until all oversampling conversions for this channel are finished.

15.4.5 Triggers

The channel trigger source is defined in the ADnCHCONx register. The trigger source is selected by the TRG1SRC[4:0] and TRG2SRC[4:0] bits.

TRG1SRC[4:0] bits:

- Define a trigger source for a single conversion mode (MODE[1:0] bits = '00')
- Select a signal to enable/gate the TRG2SRC[4:0] triggers in the window mode (MODE[1:0] bits = '01'). The polarity of the TRG1SRC[4:0] signal can be changed by TRG1POL bit.
- Define the start/first trigger source for integration and oversampling modes (MODE[1:0] bits = '10' and MODE[1:0] bits = '11')

TRG2SRC[4:0] bits:

- Are not used for a single conversion mode (MODE[1:0] bits = '00')
- Select a trigger source for all conversions in the window mode (MODE[1:0] bits = '01')
- Re-trigger ADC in integration and oversampling modes after the first trigger specified by TRG1SRC[4:0] bits (MODE[1:0] bits = '10' and MODE[1:0] bits = '11')

The following types of triggers are available:

- Software
- Back-to-back
- Repeat timer
- From other peripheral modules

Software Trigger

The software trigger is used when TRG1SRC[4:0] or TRG2SRC[4:0] bits are set to '00001'. Trigger is generated when the corresponding bit in the ADnSWTRG register is set.

Back-to-Back Trigger

The channel is re-triggered immediately after the previous conversion is finished when TRG1SRC[4:0] or TRG2SRC[4:0] bits are set to '00010'. The channel conversions are executed back-to-back. The timing is affected (can be delayed) by priorities of other channels.

Repeat Timer Trigger

The channel is triggered from an internal ADC repeat timer when TRG1SRC[4:0] or TRG2SRC[4:0] bits are set to '00011'. This timer is clocked from the ADC clock (80 MHz frequency/12.5 nS period) and its period is set by RPTCNT[5:0] bits.

Peripheral Modules Triggers

All other TRG1SRC[4:0] and TRG2SRC[4:0] bit settings starting from '00100' select trigger sources from other modules. These trigger options are device specific.

15.4.6 Offset Calibration

This ADC requires an offset calibration. The hardware calibration procedure is executed:

- Each time when the ADC is enabled (ON bit = '1' in ADnCON[15])
- By a software request when the CALREQ bit (ADnCON[29]) is set.
- Periodically. The periodic recalibration is enabled when the ACALEN bit (ADnCON[28]) is set. The time between calibration cycles is selected by the CALRATE[1:0] bits, from a one second to a one hour period.

The CALRDY bit (ADnCON[30]) indicates the calibration status. This bit is set by hardware when the calibration cycle is finished and the hardware clears it when the calibration is in progress.

The calibration has lowest priority and is delayed when a conversion is in progress. The ADC must be idle for a few ADC clock cycles to start the calibration. This idle time is set by the CALCNT[1:0] bits.

The initial power-on calibration requires about 5000 ADC clock cycles, but recalibration requested by the software using the CALREQ bit, or recalibration done periodically when the ACALEN bit is set, takes about 10 ADC clock cycles.

15.4.7 Comparator

Each ADC channel has a dedicated digital comparator that compares the channel conversion result (ADnDATAx) with thresholds stored in the ADnCMPLOx and ADnCMPHIx registers. The following comparison criteria is selected by the CMPMOD[2:0] bits (ADnCHCONx[18:16]):

- Out of bounds (CMPMOD[2:0] bits = '001') when ADnCMPLOx<ADnDATAx or ADnDATAx>ADnCMPHIx.
- In bounds (CMPMOD[2:0] bits = '010') when ADnCMPLOx≤ADnDATAx≤ADnCMPHIx.
- Greater than (CMPMOD[2:0] bits = '011') when ADnDATAx>ADnCMPLOx.
- Less or equal (CMPMOD[2:0] bits = '100') when ADnDATAx≤ADnCMPLOx.

For all other CMPMOD[2:0] bit options, the digital comparator is disabled.

When the comparison match event is detected, the corresponding channel CMPxFLG bit in the ADnCMPSTAT register and the ADnCMPxIF interrupt flag are set.

15.4.8 Interrupts

Each channel has an individual result ready interrupt with an ADnCHxIF flag in the corresponding IFS register. The channel interrupt can be enabled by setting the ADnCHxIE bit in the IEC register.

The channel interrupt is generated:

- After each conversion for the single conversion mode (MODE[1:0] bits = '00').
- For the window mode (MODE[1:0] bits = `01') when the number of conversions reaches a value in the CNTx[15:0] bits (ADnCNTx[15:0]) or when the gate signal defined by the TRG1SRC[4:0] bits is deasserted.
- When all conversions are finished for the oversampling or integration modes (MODE[1:0] bits = '10' or '11').

The result ready interrupt can be generated before the result is available in the ADnDATAx register. This feature is called "Early Interrupt" and can reduce the ADC channel interrupt latency. This early interrupt for the channel is enabled by setting of the EIEN bit (ADnCHCONx[24]). Early interrupts can only be used in single conversion mode (MODE[1:0] bits = '00'). When the early interrupt is enabled (EIEN bit = '1'), the channel individual interrupt is generated and the CHxRDY bit in the ADnSTAT register is set at the start of the sampling time. The software must guarantee that the channel data are ready when the ADnDATAx register is read in the Interrupt Service Routine.

Each channel also has a comparator interrupt with the ADnCMPxIF flag in the corresponding IFS register. The channel comparator interrupt is generated on the comparator match event. The comparator interrupt can be enabled by setting the ADnCMPxIE bit in the IEC register.

15.4.9 Test Mode

The test mode allows the ADC controller to be tested. This mode is enabled when the TSTEN bit (ADnCON[8]) is set. When enabled, the result of any conversion is overwritten with a value from the ADnDATAOVR register. The TSTEN bit can be protected from an unintentional write by setting of the TSTLOCK bit (ADnCON[10]) .

15.4.10 Result Formatting

The result of a single conversion (MODE[1:0] bits = '00') is stored in the ADnDATAx register. For the multiple conversion modes (MODE[1:0] != '00'), the results are added to an internal primary accumulator. The result sum is copied from the primary accumulator to the ADnDATAx register when the corresponding CHxRDY bits are set.

The value in the ADnDATAx register is formatted using DIFF and LEFT bit settings (ADnCHCONx[15] and ADnCHCONx[10]). The format options are explained in Table 15-5.

Table 15-5. Output Format

Differential Format Option DIFF bit	Input Voltage	Result in ADnDATAx register			
	(V_{INP} = Voltage on non-inverting input, V_{INN} = Voltage on inverting input)	LEFT bit = 0		LEFT bit = 1	
		Decimal	Hex	Decimal	Hex
0	$V_{INP} = 0$	0	0000 0000	0	0000 0000
	$V_{INP} = V_{DD/2}$	+2047	0000 07FF	+2,146,435,072	7FF0 0000
	$V_{INP} \geq V_{DD}$	+4095	0000 0FFF	+4,293,918,720	FFF0 0000
1	$V_{INP} - V_{INN} \leqq - V_{DD}$	-2048	FFFF F800	-2,147,483,648	8000 0000
	$V_{INP} - V_{INN} = -V_{DD/2}$	-1024	FFFF FC00	-1,073,741,824	C000 0000
	$V_{INP} - V_{INN} = 0$	0	0000 0000	0	0000 0000
	$V_{INP} - V_{INN} = V_{DD}/2$	+1023	0000 003FF	+1,072,693,248	3FF0 0000
	$V_{INP} - V_{INN} \geq V_{DD}$	+2047	0000 007FF	+2,146,435,072	7FF0 0000

15.4.11 Enabling the ADC

The ADC module is enabled when the ON bit (ADnCON[15]) is set. The ON bit should be set only after the module has been configured. When the software sets the ON bit, the hardware requires approximately five ADC clock cycles to begin operation. When the ADC module is enabled, the module will perform an offset calibration cycle (~5,000 ADC clocks). The ADRDY bit (ADnCON[31]) is set by hardware when the ADC is ready for operation.

If the ON bit is cleared (after having been set), all status and ready bits are automatically cleared. The control bits and the result register's contents remain unaffected since it was last programmed/ updated.

The ADC can be in one of three operation states indicated by the OMODE[1:0] bits (ADnCON[25:24]):

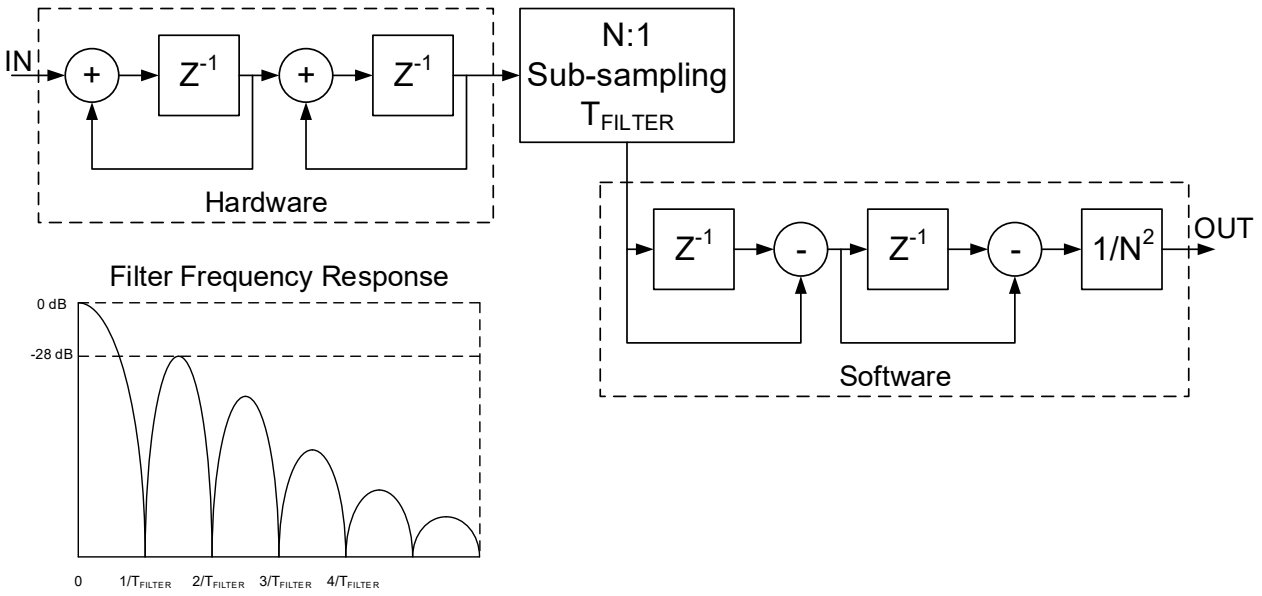
- **Off** (OMODE[1:0] = '00'): When power of the converter is switched off.
- **Standby** (OMODE[1:0] = '01'): When the module is in a power-saving mode.
- **Run** (OMODE[1:0] = '1x'): When the module is active and ready to convert.

15.4.12 Filter (Secondary Accumulator)

The setting channels 17, 18 and 19 with the implemented secondary accumulator ADnACCx sums the output of the primary accumulator ADnDATAx. The secondary accumulator is enabled when the

ACCRO bit (ADnCHCONx[26]) is set. If the ACCRO bit = '1', the ADnDATAx and ADnACCx accumulator are not cleared; instead, they will roll-over as the data is accumulated over many multi-sample operations. The accumulators function as a Second Order Cascaded-Integrator-Comb filter (CIC). Some of the CIC operations (differentiation functions) need to be performed by the application software as shown in Figure 15-3.

Figure 15-3. Second Order CIC Filter



15.4.13 Operation During Power-Saving Modes

The power-saving modes, Sleep and Idle, are useful for reducing the conversion noise by minimizing the digital activity of the CPU, buses and other peripherals.

To reduce the current consumption when the ADC is idle, the converter can be configured to a stand-by mode using the STNDBY bit (ADnCON[16]). When the STNDBY bit is set, the ADC enters the power-saving mode. The status OMODE[1:0] bits (ADnCON[25:24]) are switched to '01' when the ADC is in stand-by state.

15.4.13.1 Sleep and Idle Modes

When a device enters Sleep mode, the system oscillator (F_{OSC}) and all components that operate from it are halted. This includes the ADC when F_{OSC} is selected for the clock source. When Sleep mode is invoked during a conversion with F_{OSC} as the clock source, the conversion is aborted. The converter will not resume a partially completed conversion upon exiting from Sleep mode. The ADC register contents are not affected by the device entering or leaving Sleep mode.

The ADC module can operate during Sleep mode if the ADC clock source is active during Sleep mode. The FRC oscillator is a logical choice for operation in Sleep mode. ADC operation during Sleep mode reduces the digital switching noise from the rest of the microcontroller during the conversion process.

If any of the ADC interrupts are enabled, the device will wake from Sleep mode when the ADC interrupt occurs. The program execution will resume at the ADC ISR if the ADC interrupt is greater than the current CPU priority. Otherwise, execution will continue from the instruction after the PWRSAV instruction that placed the device in Sleep mode.

For operation during Sleep mode, the application must use a conversion trigger source that ensures that the A/D conversion will take place in Sleep mode. For example, the external trigger pin option (TRGSRCx[4:0] = 11111) can be used for performing sampling and conversion while the device is in Sleep mode.

Stopping the ADC module in Sleep/Idle is not recommended due to the need to perform a calibration cycle afterward (5000 cycle). The recommended method to reduce power is to put the module in Standby mode which only requires 200 clocks to re-enter operation. Another alternative method is to disable the module (ADCON=0) and then re-enable the module when SLEEP is exited (which requires a recalibration).

15.4.14 Calibration

The startup hardware calibration procedure is executed each time the ADC is enabled via setting the ON bit (ADnCON[15]). The startup calibration procedure includes both gain and offset adjustment. The CALRDY bit (ADnCON[30]) status bit provides indication that the process is complete. CALRDY is set by hardware when the calibration cycle is finished, and the hardware clears this bit when the calibration is in progress. The startup calibration takes about 5000 ADC clock cycles.

The ADC offset may drift slightly across temperature. The ADC has hardware to run the offset calibration. The offset calibration procedure takes 14 T_{AD} cycles and can be executed using one of the following methods:

- A software request by setting the CALREQ bit (ADnCON[29]).
- Automatic, periodically. The periodic recalibration is enabled by the ACALEN bit (ADnCON[28]). The time between calibration cycles is set with the CALRATE[1:0] bits from times ranging from 1 second to 1 hour.

The offset calibration has the lowest priority, and it is delayed when a conversion is in progress. The ADC must idle a few ADC clock cycles to start the calibration. This idle time is set by CALCNT[1:0] bits.

15.4.14.1 Software Gain Error Compensation

The start-up hardware calibration precision is limited, and after every ADC enable cycle the gain and offset errors may be different. When more precise and repeatable gain is required, the application can use reference voltages to measure the gain error and apply the corrective coefficient to the results of the conversions. The compensation coefficients can be calculated just once after the start-up calibration. A code example of the procedure is provided in Example 15-3.

15.4.14.1.1 Error Compensation Coefficient Calculation

To calculate the ADC gain error compensation coefficient, two reference voltages are required. The ADC has one input connected to 15/16 of AV_{DD} voltage. When it is assumed that the ADC offset after calibration is zero Equation 15-2 can be used to calculate the compensation coefficient.

Equation 15-2. One Reference Voltage Gain Error Compensation Coefficient Calculation

$$Gain\ Error\ Compensation\ Coefficient = \frac{\frac{15}{16} \times 4095}{\frac{15}{16} \times AVDD\ Result}$$

The offset may not always be calibrated to exact zero. Refer to the ADC electrical specifications for more information.

If the application requires more precise gain error compensation the second data point is needed. The external resistor voltage divider can be added to provide the second reference voltage on one of the ADC analog inputs. The second reference voltage can be 2%-10% of AVDD. If the external resistor divider voltage is AVDD/16, then the two reference voltages' gain error compensation coefficient and offset error can be calculated as shown in Equation 15-3.

Equation 15-3. Two Reference Voltages Gain Error Compensation Coefficient and Offset Error Calculation

$$Gain\ Error\ Compensation\ Coefficient = \frac{\frac{14}{16} \times 4095}{(\frac{15}{16} \times AVDD\ Result - \frac{1}{16} \times AVDD\ Result)}$$

$$Offset\ Error = \frac{1}{16} \times AVDD\ \ Result - \frac{\frac{1}{16} \times 4095}{Gain\ Error\ Compensation\ Coefficient}$$

15.4.14.1.2 Application of Error Compensation Coefficient

To compensate for the gain error, each ADC result should be multiplied by the calculated coefficient. The compensation can be done as either a fixed-point or floating-point multiplication. The floating-point calculation is shown in Example 15-1.

This operation may take up to 105 nS if the CPU clock is 200 MHz. The multiplication can be accelerated if the fix-point calculation is used as shown in Example 15-2

Example 15-1. ADC Conversion Result Correction Using Floating-Point Calculations

```
float coefficient = 3840.0/adc15div16result; // needed to be done once
……
// takes 21 instruction cycles or 105 nS @ 200 MHz CPU clock
long corrected_result_channel_0 = (long)(coefficient*((float)AD1CH0DATA));
```

Example 15-2. 12-bit Unsigned ADV Conversion Result Correction Using Fixed-Point Calculations

```
unsigned long coefficient = (unsigned long) ((1<<19)*3840.0/
adc15div16result)); // needed to be done once
……
// takes 6 instruction cycles or 30 nS @ 200 MHz CPU clock
unsigned long corrected_result_channel_0 = (coefficient*AD1CH0DATA)>>19;
```

15.5 Application Examples

15.5.1 Gain Error Calibration

Example 15-3. Gain Error Calibration Example

```
#include <xc.h>
// The channel output.
long result = 0;
// Gain compensation coefficient.
long  coefficient;
// Oscillator initialization procedure.
void OscillatorInitialization();
int main(){
    // Initialize the oscillator.
    // Clock generator 6 should provide 320MHZ to the ADCs.
    OscillatorInitialization();
    // Enable ADC.
    AD1CONbits.ON = 1;
    // Wait when ADC will be ready/calibrated.
    while(AD1CONbits.ADRDY == 0);
    ////////////////////////////////////////////////////////////////
    // GET A COEFFICIENT FOR THE GAIN ERROR COMPENSATION
    ////////////////////////////////////////////////////////////////
    // Select oversampling mode.
```

```
    AD1CH1CONbits.MODE = 3;
    // 256 conversions
    AD1CH1CONbits.ACCNUM = 3;
    // Software trigger will start a conversion.
    AD1CH1CONbits.TRG1SRC = 1;
    // Back-to-back conversions
    AD1CH1CONbits.TRG2SRC = 2;
    // Select the AN14 input which is connected to 15/16 of AVDD
    AD1CH1CONbits.PINSEL = 14;
    // Select signal sampling time (6.5 TADs = 81nS @ 40MHZ ADC clock).
    AD1CH1CONbits.SAMC = 3;
    // Average 256 results of the reference voltage
    AD1SWTRGbits.CH1TRG = 1;
    // Wait when the result is ready
    while(AD1STATbits.CH1RDY == 0);
    // Oversampling result is 16 Bit (has additional 4 bits).
    // Calculate the gain compensation coefficient.
    // The coefficient is in fixed-point format (18 bits before point).
    coefficient = (long)(3840.0*16.0*(1<<18)/AD1CH1DATA);
    ////////////////////////////////////////////////////////////////////
    // CONVERT AND COMPENSATE THE GAIN ERROR
    ////////////////////////////////////////////////////////////////////
    // Clean channel register for new settings.
    AD1CH1CON = 0;
    // Select single conversion mode.
    AD1CH1CONbits.MODE = 0;
    // Software trigger will start a conversion.
    AD1CH1CONbits.TRG1SRC = 1;
    // Select the AN7 input for conversions
    AD1CH1CONbits.PINSEL = 7;
    // Select signal sampling time (6.5 TADs = 81nS @ 40MHZ ADC clock).
    AD1CH1CONbits.SAMC = 3;
    // Trigger channel #1 in software and wait for the result.
    while(1){
        // Trigger channel # 1.
        AD1SWTRGbits.CH1TRG = 1;
        // Wait for a conversion ready flag.
        while(AD1STATbits.CH1RDY == 0);
        // Read result. It will clear the conversion ready flag.
        // The gain error correction coefficient is in fixed-point format (18
bits before point).
        result = (coefficient*AD1CH1DATA)>>18;
    }
    return 1;
}
void OscillatorInitialization(){
    PLL1CONbits.ON = 1;
    OSCCTRLbits.PLL1EN = 1;
    while(OSCCTRLbits.PLL1RDY == 0);
    PLL1CONbits.FSCMEN = 1; // disable clock fail monitor
    VCO1DIVbits.INTDIV = 1; // 1:2 = 320MHz
    PLL1DIVbits.PLLFBDIV = 80; // VCO = 640 MHz
    PLL1DIVbits.PLLPRE = 1;
    PLL1DIVbits.POSTDIV1 = 4;
    PLL1DIVbits.POSTDIV2 = 1;
    PLL1CONbits.DIVSWEN = 1;
    while(PLL1CONbits.DIVSWEN == 1);
    PLL1CONbits.NOSC = 1; // FRC
    PLL1CONbits.OSWEN = 1;
    while(PLL1CONbits.OSWEN == 1);
    PLL1CONbits.FOUTSWEN = 1;
    while(PLL1CONbits.FOUTSWEN == 1);
    PLL1CONbits.PLLSWEN = 1;
    while(PLL1CONbits.PLLSWEN == 1);
    while(PLL1CONbits.CLKRDY == 0);
    CLK1CONbits.NOSC = 5; // PLL1
    CLK1CONbits.OSWEN = 1;
    while(CLK1CONbits.OSWEN == 1);
    while(CLK1CONbits.CLKRDY == 0);
    // ADC high speed clock (Generator 6), should be 320 MHz for 80MHz
operation
    CLK6CONbits.ON = 1;
    CLK6CONbits.NOSC = 7; // PLL1 VCO divider
    CLK6CONbits.OSWEN = 1;
    while(CLK6CONbits.OSWEN == 1);
```

```
    while(CLK6CONbits.CLKRDY == 0);
}
```

15.5.2 Single Conversion

In Example 15-4, a software trigger is used to start a single conversion.

Example 15-4. Single Conversion Example

```
#include <xc.h>
// The channel output.
long result = 0;
int main(){
 // In this example channel 1 will be used.
 // Select single conversion mode.
 AD1CHCON1bits.MODE = 0;
 // Software trigger will start a conversion.
 AD1CHCON1bits.TRG1SRC = 1;
 // Use a differential input.
 AD1CHCON1bits.DIFF = 1;
 // Select the AN9 analog positive input/pin for the signal.
 AD1CHCON1bits.PINSEL = 9;
 // Select the ANN1 analog negative input/pin for the signal.
 AD1CHCON1bits.NINSEL = 1;
 // Select signal sampling time (6.5 TADs = 81nS).
 AD1CHCON1bits.SAMC = 3;
 // Set ADC to RUN mode.
 AD1CONbits.MODE = 2;
 // Enable ADC.
 AD1CONbits.ON = 1;
 // Wait when ADC will be ready/calibrated.
 while(AD1CONbits.ADRDY == 0);
 // Trigger channel #1 in software and wait for the result.
 while(1){
 // Trigger channel # 1.
 AD1SWTRGbits.CH1TRG = 1;
 // Wait for a conversion ready flag.
 while(AD1STATbits.CH1RDY == 0);
 // Read result. It will clear the conversion ready flag.
 result = AD1DATA1;
 }
 return 1;
}
```

15.5.3 Windowed Multiple Conversions

In Example 15-5, the conversion results are accumulated until the RA4 pin is at the high level. The conversions are triggered from the internal ADC timer.

Example 15-5. Windowed Conversions Example

```
#include <xc.h>
// The channel output from primary accumulator.
volatile long result = 0;
// The number of accumulated samples.
volatile long number_of_accumulated_samples;
int main(){
 // RA4/RP5 pin is a trigger input.
 // Switch to a digital input.
 ANSELAbits.ANSA4 = 0;
 // Make an input.
 TRISAbits.TRISA4 = 0;
 // Map external pin trigger to RP5/RA4.
 _ADTRIG31R = 5;
 // In this example channel 3 will be used.
 // Set limit for the accumulated samples number.
 AD1CNT3bits.CNT3 = 0xffff;
 // Window conversion mode.
 AD1CHCON3bits.MODE = 1;
 // Accumulation will be started/stopped from an external pin.
 AD1CHCON3bits.TRG1SRC = 31;
```

```
 // Trigger all conversions from the ADC repeat timer.
 AD1CHCON3bits.TRG2SRC = 3;
 // Select the AN5 analog positive input/pin.
 AD1CHCON3bits.PINSEL = 5;
 // Select signal sampling time (6.5 TADs = 81nS).
 AD1CHCON3bits.SAMC = 3;
 // Set period of the triggers timer (63 is maximum).
 AD1CONbits.RPTCNT = 60;
 // Set ADC to RUN mode.
 AD1CONbits.MODE = 2;
 // Enable ADC.
 AD1CONbits.ON = 1;
 // Wait when ADC will be ready/calibrated.
 while(AD1CONbits.ADRDY == 0);
 // Enable interrupt;
 _AD1CH3IE = 1;
 // Channel 3 is converted and results are accumulated until the RA4 pin is
high.
 // On transition from high to low an interrupt is generated.
 while(1);
 return 1;
}
void __attribute__((interrupt)) _AD1CH3Interrupt(){
 // Read result in accumulator and clear CH3RDY flag.
 result = AD1DATA3;
 number_of_accumulated_samples = AD1CNT3bits.CNTSTAT3;
 result /= number_of_accumulated_samples;
 // Clear interrupt flag.
 _AD1CH3IF = 0;
}
```

15.5.4 Integration of the Multiple Samples

In Example 15-6, the conversions are started by a software trigger and continued by back-to-back triggers until the number of conversions is less than set in the AD1CNT15 register.

Example 15-6. Integration of the Multiple Samples Example

```
#include <xc.h>
// The channel output.
long result = 0;
int main(){
 // In this example channel 15 will be used.
 // Integration conversion mode.
 AD1CHCON15bits.MODE = 2;
 // Set number of conversions accumulated to 123.
 AD1CNT15 = 123;
 // Software trigger will start a conversions.
 AD1CHCON15bits.TRG1SRC = 1;
 // Re-trigger back to back.
 AD1CHCON15bits.TRG2SRC = 2;
 // Select the AN5 analog positive input/pin.
 AD1CHCON15bits.PINSEL = 5;
 // Select signal sampling time (6.5 TADs = 81nS).
 AD1CHCON15bits.SAMC = 3;
 // Set ADC to RUN mode.
 AD1CONbits.MODE = 2;
 // Enable ADC.
 AD1CONbits.ON = 1;
 // Wait when ADC will be ready/calibrated.
 while(AD1CONbits.ADRDY == 0);
 // Trigger channel #15 in software and wait for the 123 samples
 // accumulated result.
 while(1){
 // Trigger channel # 4.
 AD1SWTRGbits.CH15TRG = 1;
 // Wait for a conversion ready flag.
 while(AD1STATbits.CH15RDY == 0);
 // Read oversampling result. It will clear the conversion ready flag.
 result = AD1DATA15;
 }
```

```
 return 1;
}
```

15.5.5 Oversampling

In Example 15-7, the conversions are started by a software trigger and continued by back-to-back triggers. The number of accumulated conversion results is set to 16. The oversampling process cannot be interrupted by other channel conversions because the ACCBRST bit is set.

Example 15-7. Oversampling Example

```
#include <xc.h>
// The channel output.
long result = 0;
int main(){
 // In this example channel 4 will be used.
 // Oversampling conversion mode.
 AD1CHCON4bits.MODE = 3;
 // Set number of conversions accumulated to 16.
 AD1CHCON4bits.ACCNUM = 1;
 // The oversampling if started cannot be interrupted
 // by a high priority channels conversion requests.
 AD1CHCON4bits.ACCBRST = 1;
 // Software trigger will start a conversions.
 AD1CHCON4bits.TRG1SRC = 1;
 // Re-trigger back to back.
 AD1CHCON4bits.TRG2SRC = 2;
 // Select the AN5 analog positive input/pin.
 AD1CHCON4bits.PINSEL = 5;
 // Select signal sampling time (6.5 TADs = 81nS).
 AD1CHCON4bits.SAMC = 3;
 // Set ADC to RUN mode.
 AD1CONbits.MODE = 2;
 // Enable ADC.
 AD1CONbits.ON = 1;
 // Wait when ADC will be ready/calibrated.
 while(AD1CONbits.ADRDY == 0);
 // Trigger channel #4 in software and wait for the 16 samples
 // oversampling result.
 while(1){
 // Trigger channel # 4.
 AD1SWTRGbits.CH4TRG = 1;
 // Wait for a conversion ready flag.
 while(AD1STATbits.CH4RDY == 0);
 // Read oversampling result. It will clear the conversion ready flag.
 result = AD1DATA4;
 }
 return 1;
}
```

15.5.6 Channel Comparator

In Example 15-8, the comparator interrupt is generated each time when the conversion result is outside of a window.

Example 15-8. Comparator Example

```
#include <xc.h>
// The channel output.
long result = 0;
int main(){
 // In this example channel 1 will be used.
 // Select single conversion mode.
 AD1CHCON1bits.MODE = 0;
 // Software trigger will start a conversion.
 AD1CHCON1bits.TRG1SRC = 1;
 // Select the AN9 analog positive input/pin for the signal.
 AD1CHCON1bits.PINSEL = 9;
 // Select signal sampling time (6.5 TADs = 81nS).
 AD1CHCON1bits.SAMC = 3;
```

```
 // Enable the comparator for this channel.
 // Select out of bounds mode.
 AD1CHCON1bits.CMPMOD = 1;
 // Select low limit. To generate comparator event when AD1DATA1 < 1024.
 AD1CMPLO1 = 1024;
 // Select high limit. To generate comparator event when AD1DATA1 > 3072.
 AD1CMPHI1 = 3072;
 // Enable timer interrupt.
 _AD1CMP1IE = 1;
 // Set ADC to RUN mode.
 AD1CONbits.MODE = 2;
 // Enable ADC.
 AD1CONbits.ON = 1;
 // Wait when ADC will be ready/calibrated.
 while(AD1CONbits.ADRDY == 0);
 // Trigger channel #1 in software and wait for the result.
 while(1){
 // Trigger channel # 1.
 AD1SWTRGbits.CH1TRG = 1;
 // Wait for a conversion ready flag.
 while(AD1STATbits.CH1RDY == 0);
 // Read result. It will clear the conversion ready flag.
 result = AD1DATA1;
 }
 return 1;
}
void __attribute__((interrupt)) _AD1CMP1Interrupt(){
 // Process the comparator event here.
 // Clear the comparator event flag.
 AD1CMPSTATbits.CH1FLG = 0;
 // Clear the comparator flag.
 _AD1CMP1IF = 0;
}
```

15.5.7 Multiple Channels Scan

In Example 15-9, three channels are scanned. To scan these channels, they are triggered from the same trigger source.

Example 15-9. Multiple Channels Scan Example

```
#include <xc.h>
volatile long channel_2_an1;
volatile long channel_4_an2;
volatile long channel_6_an3;
// Define peripheral clock frequency.
#define FCY (150000000UL) // 150MHz
// Define the CCP1 timer frequency.
#define TIMER_FREQUENCY (100UL) // 1kHz
int main(){
 // In this example channels ## 2, 4 and 6 will be scanned.
 // To scan channels they must be triggered from one source.
 // The channel with lowest number (#2) will be converted first.
 // The channel with highest number (#6) will be converted last.
 // CHANNEL 2
 // Single conversion mode.
 AD1CHCON2bits.MODE = 0;
 // CCP1 Timer starts conversion (same for all scanned channels).
 AD1CHCON2bits.TRG1SRC = 12;
 // Select the AN1 analog input/pin for the channel #2.
 AD1CHCON2bits.PINSEL = 1;
 // Select signal sampling time (6.5 TADs = 81nS).
 AD1CHCON2bits.SAMC = 3;
 // CHANNEL 4
 // Single conversion mode.
 AD1CHCON4bits.MODE = 0;
 // CCP1 Timer starts conversion (same for all scanned channels).
 AD1CHCON4bits.TRG1SRC = 12;
 // Select the AN2 analog input/pin for the channel #4.
 AD1CHCON4bits.PINSEL = 2;
 // Select signal sampling time (8.5 TADs = 106nS).
 AD1CHCON4bits.SAMC = 4;
 // CHANNEL 6
 // Single conversion mode.
```

```
 AD1CHCON6bits.MODE = 0;
 // CCP1 Timer starts conversion (same for all scanned channels).
 AD1CHCON6bits.TRG1SRC = 12;
 // Select the AN3 analog input/pin for the channel #6.
 AD1CHCON6bits.PINSEL = 3;
 // Select signal sampling time (10.5 TADs = 131nS).
 AD1CHCON6bits.SAMC = 5;
 // Set ADC to RUN mode.
 AD1CONbits.MODE = 2;
 // Enable ADC.
 AD1CONbits.ON = 1;
 // Wait when ADC will be ready/calibrated.
 while(AD1CONbits.ADRDY == 0);
 // Configure CCP1 Timer to trigger all channels (to scan).
 CCP1CON1bits.MOD = 0;
 // Set 32-bit timer.
 CCP1CON1bits.T32 = 1;
 // Set period.
 CCP1PR = FCY/TIMER_FREQUENCY;
 // Run timer.
 CCP1CON1bits.ON = 1;
 // Enable channel # 6 interrupt.
 // This channel is processed last and all other channels results
 // will be ready in the channel #6 ISR.
 _AD1CH6IE = 1;
 while(1);
 return 1;
}
// Channel # 6 interrupt (processed last). All channels
// results in the scan are available here.
void __attribute__((interrupt)) _AD1CH6Interrupt(){
 channel_2_an1 = AD1DATA2;
 channel_4_an2 = AD1DATA4;
 channel_6_an3 = AD1DATA6;
 _AD1CH6IF = 0;
}
```

15.5.8 Channel Filter (Secondary Accumulator)

In Example 15-10, the second order low pass filter is implemented using the second accumulator.

Example 15-10. Second Order Low Pass Filter Example

```
#include <xc.h>
// VARIABLES OF THE SOFTWARE PART OF THE FILTER.
// These global variables are used in an interrupt.
// That's why they must be declared as "volatile".
// Input for the software part of the filter's first stage.
volatile long ch19_current_1 = 0;
// Input delayed for the first stage.
volatile long ch19_previous_1 = 0;
// Input for the software part of the filter's second stage.
volatile long ch19_current_2 = 0;
// Input delayed for the second stage.
volatile long ch19_previous_2 = 0;
// The filter output.
volatile long filtered_result = 0;
// Define peripheral clock frequency.
#define FCY (150000000UL) // 150MHz
// Define the CCP1 timer frequency.
#define TIMER_FREQUENCY (100UL) // 1kHz
int main(){
 // This example assumes that dsPIC33AK128MC106 device is used.
 // dsPIC33AK128MC106 device has 3 channels with the secondary
 // accumulator implemented: ## 17, 18 and 19.
 // This example will use channel #19.
 // Enable accumulators roll-over to enable the secondary accumulator.
 AD1CHCON19bits.ACCRO = 1;
 // Select integration sampling mode.
 AD1CHCON19bits.MODE = 2;
 // CCP1 Timer starts conversions (1kHz frequency).
 AD1CHCON19bits.TRG1SRC = 12;
 // CCP1 Timer re-triggers (1kHz frequency).
 AD1CHCON19bits.TRG2SRC = 12;
```

```
 // Select the AN1 analog input/pin for the signal to b filtered.
 AD1CHCON19bits.PINSEL = 1;
 // Select signal sampling time (6.5 TADs = 81nS).
 AD1CHCON19bits.SAMC = 3;
 // Set number of conversions = 8 for the filter (sub-sampler).
 // The CH19RDY bit will be set after 8 conversions.
 // The conversions frequency is 1kHz defined by CCP1 Timer period.
 // The signal maximum frequency is in twice less = 500 Hz.
 // The filter cut-off frequency is 500kHz/8 = 62.5 Hz.
 AD1CNT19 = 8;
 // Set ADC to RUN mode.
 AD1CONbits.MODE = 2;
 // Enable ADC.
 AD1CONbits.ON = 1;
 // Wait when ADC will be ready/calibrated.
 while(AD1CONbits.ADRDY == 0);
 // Configure CCP1 Timer to trigger the channel # 19.
 CCP1CON1bits.MOD = 0;
 // Set 32-bit timer.
 CCP1CON1bits.T32 = 1;
 // Set period.
 CCP1PR = FCY/TIMER_FREQUENCY;
 // Run timer.
 CCP1CON1bits.ON = 1;
 // Enable channel # 19 interrupt.
 _AD1CH19IE = 1;
 // The AN1 pin filtered result is available in the channel # 19 interrupt.
 while(1);
 return 1;
}
// Channel # 19 interrupt.
// Called when integration is finished (every AD1CNT19 = 8 conversions).
void __attribute__((interrupt)) _AD1CH19Interrupt(){
long primary_accumulator;
 // Clear interrupt flag. If the interrupt is persistent then
 // to clear the flag it is required to read the ADC channel
 // result register first.
 primary_accumulator = AD1DATA19;
 _AD1CH19IF = 0;
 // Process software part of the filter.
 ch19_current_1 = AD1ACC19;
 ch19_current_2 = ch19_previous_1 - ch19_current_1;
 ch19_previous_1 = ch19_current_1;
 filtered_result = ch19_previous_2 - ch19_current_2;
 ch19_previous_2 = ch19_current_2;
 // Divide by 1:(8*8) or 1:64 or shift right by 6
 filtered_result >>= 6;
}
```

15.6 Effects of Reset

Following any Reset event, all the ADC control and status registers are reset to their default values with the control bits in a non-active state. This disables the ADC module and sets the analog input pins to Analog Input mode. Any conversion that was in progress will be terminated and the result will not be written to the result buffer. The values in the ADCBUFx registers are initialized to 0000h during a device Reset.

16. High-Speed Analog Comparator with Slope Compensation DAC

The high-speed analog comparator module provides a method to monitor voltage, current and other critical signals in a power conversion application that may be too fast for the CPU and ADC to capture. The analog comparator module can be used to implement Peak Current mode control, Critical Conduction mode (variable frequency) and Hysteretic Control mode.

The High-Speed Analog Comparator with Slope Compensation DAC consists of the following key features:

- Rail-to-Rail Analog Comparator
- Programmable Comparator Hysteresis
- Programmable Output Polarity
- Interrupt Generation Capability
- Dedicated 12-bit Resolution Pulse Density Modulation (PDM) Digital-to-Analog Converter (DAC) for each Analog Comparator
- Multimode Multipole RC Output Filter:
  - Transition mode: provides the fastest response
  - Fast mode: for tracking DAC slopes
  - Steady-State mode: provides 12-bit resolution
- Dedicated Support for the Following Modes:
  - Slope Generation
  - Hysteretic Control
  - Triangular Wave
- Functional Support for the High-Speed PWM Module Which Includes:
  - PWM duty cycle control
  - PWM period control
  - PWM Fault detect

16.1 Device-Specific Information

Table 16-1. DAC Summary

DAC Module Instances	Inputs per Instance	DAC Outputs	Clock Source	Peripheral Bus Speed
3	4	1	CLKGEN7	Standard

Table 16-2. High-Speed Analog Comparator Module Availability

DAC Input	28-Pin	36-Pin	48-Pin	64-Pin	PPS
CMP1A	x	x	x	x	No
CMP1B	x	x	x	x	No
CMP1C	x	x	x	x	No
CMP1D	x	x	x	x	No
CMP2A	x	x	x	x	No
CMP2B	x	x	x	x	No
CMP2C	x	x	x	x	No
CMP2D	x	x	x	x	No
CMP3A		x	x	x	No
CMP3B		x	x	x	No
CMP3C	x	x	x	x	No

.....continued					
DAC Input	28-Pin	36-Pin	48-Pin	64-Pin	PPS
CMP3D	x	x	x	x	No

Table 16-3. Slope Start Signal Selection (SLPSTRT)

Slope Start Signal Selection	Source
5-15	1
4	PWM4 Trigger 1
3	PWM3 Trigger 1
2	PWM2 Trigger 1
1	PWM1 Trigger 1
0	0

Table 16-4. Slope Stop A Signal Select bits (SLPSTOPA)

Slope Stop A Signal Selection	Source
5 - 15	1
4	PWM4 Trigger 2
3	PWM3 Trigger 2
2	PWM2 Trigger 2
1	PWM1 Trigger 2
0	0

Table 16-5. Slope Stop B Signal Select bits (SLPSTOPB)

Slope Stop B Signal Selection	Source
2 - 15	1
1	CMP1 Out
0	0

Table 16-6. Hysteretic Comparator Function Input Select Bits (HCFSEL)

Hysterectic Comparator Function Input Selection	Description
15	1
5 - 14	0
4	PWM4
3	PWM3
2	PWM2
1	PWM1
0	0

The calibration register FPDMDAC is located in Flash at 0x7F20B0 with the POSINLADJ, NEGINLADJ and DNLADJ bit fields. The location should be copied and written to the corresponding bit fields in the DACCTRL1 SFR at start-up.

Table 16-7. FPDMDAC Calibration Register

Name	Address Offset	Bit Field	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
FPDMDAC	0x0B0	31:24	TOVRE	—	—	—	CFG_DAC_FILTER[3:0]			
		23:16	—	—	POSINLADJ[5:0]					
		15:8	—	NEGINLADJ[6:0]						
		7:0	—	—	—	DNLADJ[4:0]				

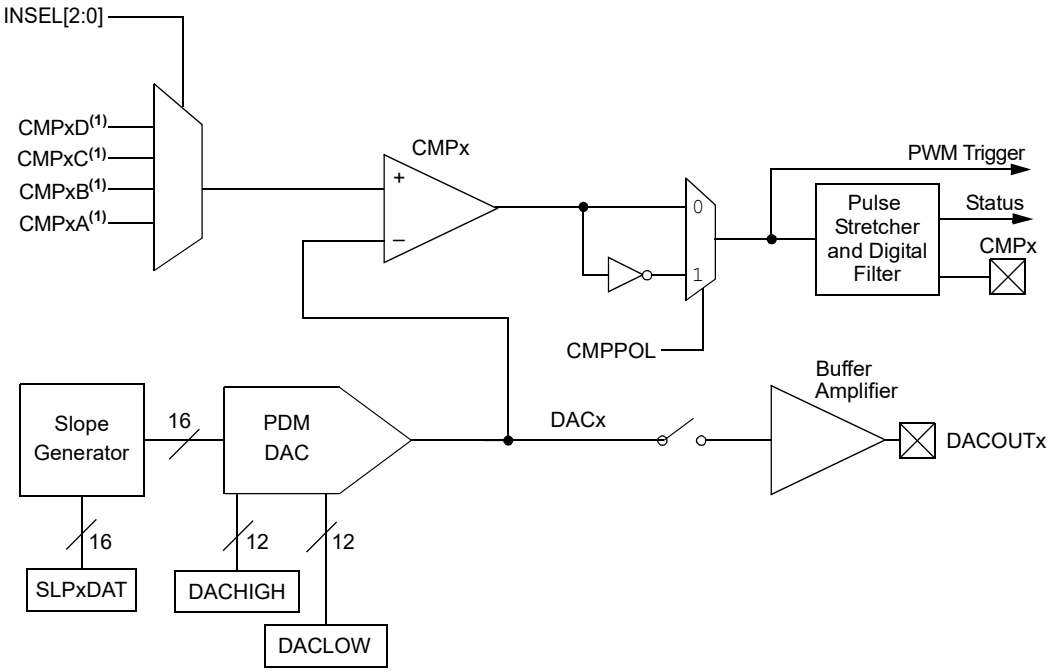
16.2 Architectural Overview

The high-speed analog comparator module is comprised of a high-speed comparator, Pulse Density Modulation (PDM) DAC and a slope compensation unit. The slope compensation unit provides a user-defined slope which can be used to alter the DAC output. This feature is useful in applications such as Peak Current mode control, where slope compensation is required to maintain the stability of the power supply. The user simply specifies the direction and rate of change for the slope compensation, and the output of the DAC is modified accordingly.

The DAC consists of a PDM unit followed by a digitally controlled multiphase RC filter. The PDM unit uses a phase accumulator circuit to generate an output stream of pulses. The density of the pulse stream is proportional to the input data value, relative to the maximum value supported by the bit width of the accumulator. The output pulse density is representative of the desired output voltage. The pulse stream is filtered with an RC filter to yield an analog voltage. The output of the DAC is connected to the negative input of the comparator. The positive input of the comparator can be selected using an MUX from the input pins. The comparator provides a high-speed operation with a typical delay of 15 ns.

The output of the comparator can be processed by the pulse stretcher and the digital filter blocks, which prevent a comparator response to unintended fast transients in the inputs. Figure 16-1 shows a block diagram of the high-speed analog comparator module. The DAC module can be operated in one of four modes: Slope Generation, Triangular Wave, Hysteretic or as a normal 12-bit DAC. Each of these modes can be used in a variety of power supply applications.

Figure 16-1. High-Speed Analog Comparator Module Block Diagram



Note:

1. Refer to specific device pinout for available inputs.

16.3 DAC Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1D40	DACCTRL1	31:24	RREN		POSINLADJ[5:0]					
		23:16		NEGINLADJ[6:0]						
		15:8	DACON		DACSIDL	DNLADJ[4:0]				
		7:0		Reserved				FCLKDIV[2:0]		
0x1D44	DACCTRL2	31:24							SSTIME[9:8]	
		23:16	SSTIME[7:0]							
		15:8							TMODTIME[9:8]	
		7:0	TMODTIME[7:0]							
0x1D48	DAC1CON	31:24							TMCB[9:0]	
		23:16	TMCB[9:0]							
		15:8	DACEN	IRQM[1:0]		EXTUPD	UPDTMDIS	CBE	DACOE	FLTREN
		7:0	CMPSTAT	CMPPOL	INSEL[2:0]			HYSPOL	HYSSEL[1:0]	
0x1D4C	DAC1DAT	31:24					DACDAT[11:0]			
		23:16	DACDAT[11:0]							
		15:8					DACLOW[11:0]			
		7:0	DACLOW[11:0]							
0x1D50	SLP1CON	31:24	SLOPEN				HME	TWME	PSE	
		23:16								FFSEN
		15:8					SLPSTOPA[3:0]			
		7:0	SLPSTOPB[3:0]				SLPSTRT[3:0]			
0x1D54	SLP1DAT	31:24								
		23:16								
		15:8	SLPDAT[15:8]							
		7:0	SLPDAT[7:0]							
0x1D58	DAC2CON	31:24							TMCB[9:0]	
		23:16	TMCB[9:0]							
		15:8	DACEN	IRQM[1:0]		EXTUPD	UPDTMDIS	CBE	DACOE	FLTREN
		7:0	CMPSTAT	CMPPOL	INSEL[2:0]			HYSPOL	HYSSEL[1:0]	
0x1D5C	DAC2DAT	31:24					DACDAT[11:0]			
		23:16	DACDAT[11:0]							
		15:8					DACLOW[11:0]			
		7:0	DACLOW[11:0]							
0x1D60	SLP2CON	31:24	SLOPEN				HME	TWME	PSE	
		23:16								FFSEN
		15:8					SLPSTOPA[3:0]			
		7:0	SLPSTOPB[3:0]				SLPSTRT[3:0]			
0x1D64	SLP2DAT	31:24								
		23:16								
		15:8	SLPDAT[15:8]							
		7:0	SLPDAT[7:0]							
0x1D68	DAC3CON	31:24							TMCB[9:0]	
		23:16	TMCB[9:0]							
		15:8	DACEN	IRQM[1:0]		EXTUPD	UPDTMDIS	CBE	DACOE	FLTREN
		7:0	CMPSTAT	CMPPOL	INSEL[2:0]			HYSPOL	HYSSEL[1:0]	
0x1D6C	DAC3DAT	31:24					DACDAT[11:0]			
		23:16	DACDAT[11:0]							
		15:8					DACLOW[11:0]			
		7:0	DACLOW[11:0]							
0x1D70	SLP3CON	31:24	SLOPEN				HME	TWME	PSE	
		23:16								FFSEN
		15:8					SLPSTOPA[3:0]			
		7:0	SLPSTOPB[3:0]				SLPSTRT[3:0]			
0x1D74	SLP3DAT	31:24								
		23:16								
		15:8	SLPDAT[15:8]							
		7:0	SLPDAT[7:0]							

16.3.1 DAC Control 1 Register

Name: DACCTRL1
Offset: 0x1D40

Note:

1. These bits should only be changed when DACON = 0 to avoid unpredictable behavior.

Bit	31	30	29	28	27	26	25	24
	RREN		POSINLADJ[5:0]					
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
		NEGINLADJ[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	DACON		DACSIDL	DNLADJ[4:0]				
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		Reserved				FCLKDIV[2:0]		
Access						R/W	R/W	R/W
Reset		0				0	0	0

Bit 31 – RREN Ripple Reduction Enable

Value	Description
1	Ripple Reduction mode is enabled
0	Ripple Reduction mode is disabled

Bits 29:24 – POSINLADJ[5:0] Positive INL Correction value

The number of ones in this register controls the rise time of the PDM drivers. Reducing the number of ones in this register will increase the rise time. This bit field only exists if a TYPE#2 filter is instantiated. This value is shared by all of the PDM DACs.

Bits 22:16 – NEGINLADJ[6:0] The number of ones in this register controls the fall time of the PDM drivers. Reducing the number of ones in this register will increase driver fall time. This bit field only exists if a type #2 filter is instantiated. This value is shared by all of the PDM DACs.

The number of ones in this register control the rise time of the PDM drivers. Reducing the number of ones in this register increase the rise time. This bit field only exists if a TYPE#2 filter is instantiated. This value is shared by all of the PDM DACs.

Bit 15 – DACON Common DAC Module Enable bit

Value	Description
1	Enables DAC modules
0	Disables DAC modules and disables FSCM clocks to reduce power consumption; any pending Slope mode and/or underflow condition is cleared

Bit 13 – DACSIDL DAC Stop in Idle Mode bit

Value	Description
1	Discontinues module operation when device enters Idle mode
0	Continues module operation in Idle mode

Bits 12:8 – DNLADJ[4:0] DNL Adjustment Override

Each bit assert to a "1"overrides the DNL pulse stretcher finger control. Each "1" reduces the amount of DNL pulse stretching, thus reducing the amount of DNL correction. This bit field only exists if a type #2 filter is instantiated. This value is shared by all of the PDM DACs

Bit 6 – Reserved

Bits 2:0 – FCLKDIV[2:0] Comparator Filter Clock Divider bits

Value	Description
111	Divide-by-8
110	Divide-by-7
101	Divide-by-6
100	Divide-by-5
011	Divide-by-4
010	Divide-by-3
001	Divide-by-2
000	1x

16.3.2 DAC Control 2 Register

Name: DACCTRL2
Offset: 0x1D44

Bit	31	30	29	28	27	26	25	24
							SSTIME[9:8]	
Access							R/W	R/W
Reset							0	0

Bit	23	22	21	20	19	18	17	16
	SSTIME[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8
							TMODTIME[9:8]	
Access							R/W	R/W
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	TMODTIME[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 25:16 – SSTIME[9:0] Time from Start of Transition Mode until Steady-State Filter is Enabled bits

Bits 9:0 – TMODTIME[9:0] Transition Mode Duration bits

16.3.3 DACx Control Low Register

Name: DACxCON
Offset: 0x1D48, 0x1D58, 0x1D68

Bit	31	30	29	28	27	26	25	24
							TMCB[9:0]	
Access							R/W	R/W
Reset							0	0

Bit	23	22	21	20	19	18	17	16
	TMCB[9:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DACEN	IRQM[1:0]		EXTUPD	UPDTMDIS	CBE	DACOE	FLTREN
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CMPSTAT	CMPPOL	INSEL[2:0]			HYSPOL	HYSSEL[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 25:16 – TMCB[9:0] DACx Leading-Edge Blanking bits
These register bits specify the blanking period for the comparator, following changes to the DAC output during Change-of-State (COS), for the input signal selected by the HCFSEL[3:0] bits in SLPxCONL.

Bit 15 – DACEN Individual DACx Module Enable bit

Value	Description
1	Enables DACx module
0	Disables DACx module to reduce power consumption; any pending Slope mode and/or underflow condition is cleared

Bits 14:13 – IRQM[1:0] Interrupt Mode select bits

Value	Description
11	Generates an interrupt on either a rising or falling edge detect
10	Generates an interrupt on a falling edge detect
01	Generates an interrupt on a rising edge detect
00	Interrupts are disabled

Bit 12 – EXTUPD External Triggered Data Updates

Value	Description
1	New DACDAT is transferred to the active DAC data register when the selecteddac_slope_start[14:1] signal specified by the SLPCON[x] SLPSTRT[3:0] is asserted.
0	New DACDAT is immediately transferred to the active DAC data register. This is known as an immediate update.

Bit 11 – UPDTMDIS Update Transition Mode Disable

Value	Description
1	The transition mode is disabled for DACDATA updates, the output voltage will be smoother, but the DAC may be slower in reaching the target voltage.
0	Transition mode is applied following DACDATA updates to reduce time to reach new specified DAC output voltage. DAC output voltage transient ripple may be introduced.

Bit 10 – CBE Comparator Blank Enable bit

Value	Description
1	Enables the analog comparator output to be blanked (gated off) during the recovery transition following the completion of a slope operation
0	Disables the blanking signal to the analog comparator; therefore, the analog comparator output is always active

Bit 9 – DACOE DAC Output Buffer Enable bit

Value	Description
1	DACx analog voltage is connected to the DACOUTx pin
0	DACx analog voltage is not connected to the DACOUTx pin

Bit 8 – FLTREN Comparator Digital Filter Enable bit

Value	Description
1	Digital filter is enabled
0	Digital filter is disabled

Bit 7 – CMPSTAT Comparator Status bits
Current state of comparator output including CMPPOL selection

Bit 6 – CMPPOL Comparator Output Polarity Control bit

Value	Description
1	Output is inverted
0	Output is noninverted

Bits 5:3 – INSEL[2:0] Comparator Input Source Select bits

Value	Description
111	Reserved
110	Reserved
101	Reserved
100	Reserved
011	CMPxD input pin
010	CMPxC input pin
001	CMPxB input pin
000	CMPxA input pin

Bit 2 – HYSPOL Comparator Hysteresis Polarity Select bit

Value	Description
1	Hysteresis is applied to falling edge of comparator output
0	Hysteresis is applied to rising edge of comparator output

Bits 1:0 – HYSSEL[1:0] Comparator Hysteresis Select bits

Value	Description
11	45 mV hysteresis
10	30 mV hysteresis
01	15 mV hysteresis

Value	Description
00	No hysteresis selected

16.3.4 DACx Data Register

Name: DACxDAT
Offset: 0x1D4C, 0x1D5C, 0x1D6C

Bit	31	30	29	28	27	26	25	24
					DACDAT[11:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DACDAT[11:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
					DACLOW[11:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DACLOW[11:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 27:16 – DACDAT[11:0] DACx High Data bits
In Hysteretic mode, Slope Generator mode and Triangle mode, this register specifies the high data value and/or limit for the DACx module. Valid values are from 205 to 3890.

Bits 11:0 – DACLOW[11:0] DACx Low Data bits
See DAC output level. In Hysteretic mode, Slope Generator mode and Triangle mode, this register specifies the low data value and/or limit for the DACx module. Valid values are from 205 to 3890.

16.3.5 DAC Slope x Control Register

Name: SLPxCON
Offset: 0x1D50, 0x1D60, 0x1D70

Bit	31	30	29	28	27	26	25	24
	SLOPEN				HME	TWME	PSE	
Access	R/W				R/W	R/W	R/W	
Reset	0				0	0	0	

Bit	23	22	21	20	19	18	17	16
								FFSEN
Access								R/W
Reset								0

Bit	15	14	13	12	11	10	9	8
					SLPSTOPA[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SLPSTOPB[3:0]				SLPSTRT[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – SLOPEN Slope Function Enable/On bit

Value	Description
1	Enables slope function
0	Disables slope function; slope accumulator is disabled to reduce power consumption

Bit 27 – HME Hysteretic Mode Enable bit

Value	Description
1	Enables Hysteretic mode for DACx
0	Disables Hysteretic mode for DACx

Bit 26 – TWME Triangle Wave Mode Enable bit

Value	Description
1	Enables Triangle Wave mode for DACx
0	Disables Triangle Wave mode for DACx

Bit 25 – PSE Positive Slope Mode Enable bit

Value	Description
1	Slope mode is positive (increasing)
0	Slope mode is negative (decreasing)

Bit 16 – FFSEN Fast First Step Mode Enable bit

Value	Description
1	Fast First Step Mode enabled
0	Fast First Step Mode disabled

Bit 16 – HCFSEL[3:0] Hysteretic Comparator Function Input Select bits
Refer to Table 16-6 for device-specific HCFSEL bit information.

Bits 11:8 – SLPSTOPA[3:0] Slope Stop A Signal Select bits
The selected Slope Stop A signal is logically OR'd with the selected Slope Stop B signal to terminate the slope function. Refer to Table 16-4 for device-specific SLPSTOPA bit information.

Bits 7:4 – SLPSTOPB[3:0] Slope Stop B Signal Select bits
The selected Slope Stop B signal is logically OR'd with the selected Slope Stop A signal to terminate the slope function. Refer to Table 16-5 for device-specific SLPSTOPB bit information.

Bits 3:0 – SLPSTRT[3:0] Slope Start Signal Select bits
Refer to Table 16-3 for device specific SLPSTRT bit information

16.3.6 DAC Slope x Data Register

Name: SLPxDAT
Offset: 0x1D54, 0x1D64, 0x1D74

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	SLPDAT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SLPDAT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – SLPDAT[15:0] Slope Ramp Rate Value bits
The SLPDATx value is in 12.4 format.

16.4 Operation

The High-Speed Analog Comparator with Slope Compensation DAC module is comprised of various blocks, such as a comparator, DAC, etc. The functionality and configuration of different blocks are discussed in this section.

16.4.1 Comparator Stage

16.4.1.1 Comparator Inputs

The inputs to the comparator module are configured using the DACxCON register. The INSEL[2:0] bits (DACxCON[5:3]) are used to select the comparator input source. The positive input of the comparator is connected to one of the selected input sources, while the negative input is always internally connected to the DAC output.

16.4.1.2 Comparator Outputs

The comparator output can be used to trigger PWM modules or interrupts for actions based on the comparator event. When the digital filter is disabled, the comparator signal is made directly available to the PWM module as a current limit and/or Fault signal. This ensures minimal latency for Current-mode applications and for time-critical (safety) applications. The status signal and the interrupt request signal will be processed by the pulse stretcher circuit. When the digital filter is enabled, the PWM trigger signal, status signal and interrupt request signal are all processed by the pulse stretcher and the digital filter logic. This will cause a delay in the current limit/Fault limit event. The polarity of the comparator output is selected by configuring the CMPPOL bit (DACxCON[6]). The comparator output can be monitored on the I/O pin by configuring the Peripheral Pin Select (PPS) register.

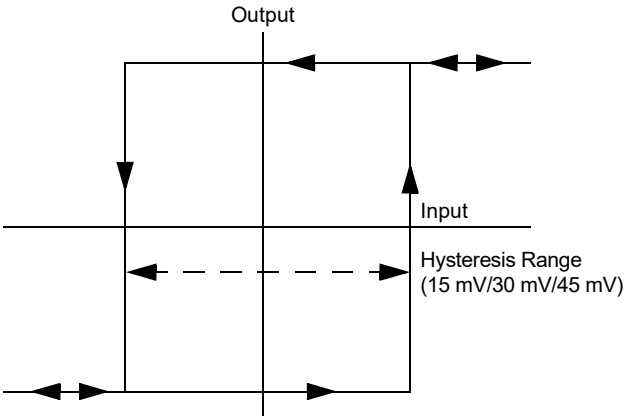
16.4.1.3 Comparator Interrupt

The analog comparator interrupt can be used to service the comparator switching event and can be enabled or disabled from the interrupt controller. The analog comparator interrupt can be configured to interrupt on the rising edge, falling edge or both by setting the IRQM[1:0] bits (DACxCON[14:13]). The comparator interrupt signal is generated on the selected edge of the comparator output, following the polarity processing through the CMPPOL bit (DACxCON[6]) and the subsequent processing by the pulse stretcher and the digital filter logic. If the CMPPOL bit is changed during operation, the bit change will not cause an interrupt. Only the selected edge of an actual change of the comparator output status will initiate an interrupt.

16.4.1.4 Comparator Hysteresis Control

The HYSSEL[1:0] bits in the DACxCON register specify the amount of hysteresis for the analog comparator. The HYSPOL bit specifies whether hysteresis is applied to the rising edge or falling edge of the signal. Configuration of hysteresis helps the comparator to avoid oscillation (i.e., toggling of the comparator output), which could be caused by noise on the positive input.

Figure 16-2. Hysteresis Control



16.4.1.5 Pulse Stretcher

The High-Speed Analog Comparator can respond to very fast transient signals. To avoid a comparator malfunction, after choosing the comparator output polarity using the CMPPOL bit (DACxCON[6]), the signal is passed to a pulse stretching circuit. The pulse stretching circuit waits for the comparator output to transition to a high state or a low state and then will stretch the signal for three clock cycles. For example, a comparator output signal of '01000101000' will be modified by the pulse stretcher circuit to '01110111110'. The pulse stretcher clock operates at a frequency of $F_{DAC}/2$ and uses the DAC clock setting bits, CLKSELx and CLKDIVx. A configuration example to set the pulse stretcher is shown in Example 16-1.

Example 16-1. Configuration for Pulse Stretcher

```
/* Pulse Stretcher Configuration */
DAC1CONbits.CMPPOL = 0;                 /* Non inverted comparator output*/
DACCTRL1bits.CLKSEL = 2;                /* F_DAC = AFPLL Auxillary PLL out */
DACCTRL1bits.CLKDIV = 1;                /* Divide by 2 */
```

16.4.1.6 Digital Filter

In many motor and power control applications, the analog comparator input signals can be corrupted by the large electromagnetic fields generated by the external switching power transistors. Corruption of the analog input signals to the comparator can cause unwanted comparator output transitions. A digital output filter can minimize the effects of the input signal corruption. The digital

filter processes the comparator signal from the pulse stretcher circuit. The digital filter is enabled by the FLTREN bit (DACxCON[8]). The digital filter operates with the clock selected by the CLKSEL[1:0] bits (DACCTRL1[7:6]) and FCLKDIV[2:0] bits (DACCTRL1[2:0]). The pulse stretcher output signal must be stable, either in a high state or a low state, for at least three times the selected filter clock frequency for it to pass through the digital filter. Assuming the current state is '0', a comparator output string of '0011110000000000' gets modified by the pulse stretcher to '0011111100000000' and to '0000000001111110' by the digital filter if the filter clock frequency is divided by two. Because of the requirement of three similar consecutive states for the filter, the selected digital filter clock period must be one third or less than the maximum desired comparator response time. In Sleep mode or Idle mode, the digital filter is bypassed to enable an asynchronous signal from the comparator to the interrupt controller. This asynchronous signal can be used to wake-up the processor from Sleep mode or Idle mode. A configuration example to enable the digital filter is provided in Example 16-2.

Example 16-2. Configuration for Digital Filter

```
DACCTRL1bits.FCLKDIV = 1;                    /* Filter Clk Divide by 2 */
DAC1CONbits.FLTREN = 1;                      /* Filter enabled */
```

16.4.2 Pulse Density Modulation (PDM) DAC

Each instance of the High-Speed Analog Comparator with Slope Compensation DAC has a dedicated DAC that is used to program the comparator threshold voltage via the DACxDAT register. The DAC comprises a digital Pulse Density Modulation (PDM) module, followed by a multistage RC filter. The PDM module generates a high-frequency output signal whose density is proportional to the DACxDAT register value. The PDM module clock is selected by the CLKSEL[1:0] and CLKDIV[1:0] bits of the DACCTRL1 register. The clock selection plays an important role in the dynamic performance of the DAC module.

The DACxDAT register values have limits of 0x0CD and 0xF32 and will provide a DAC output of 5% to 95% of AV_{DD}. For any intermediate value in the register, between 0xCD and 0xF32, the output voltage of the DAC will be proportional. The equation to calculate the DAC output voltage based on the AV_{DD} voltage source is provided in 16.3.4. DACxDAT. The DAC voltage can be varied in steps of $AV_{DD}/(2^N - 1)$, where N is the number of DAC bits (N = 12). The DAC modules are controlled by the DACON bit (DACCTRL1[15]). The DACON bit enables or disables all of the comparator modules instantiated on a given device or device core. The DACEN bit (DACxCON[15]) provides individual control of the DAC module. The individual DAC registers have an output enable bit, DACOEN (DACxCON[9]), which enables the DAC output voltage to be routed to an external output pin, DACOUT1. The DACOUT1 pin can only be associated with a single DAC or PGA output (if available on the device) at any given time. If more than one DACOEN bit is set, or the PGA Output Enable bit (PGAOEN) and the DACOEN bit are set, the DACOUT1 pin will be a combination of the signals. A configuration example to set the DAC output voltage is shown in Example 16-3.

Equation 16-1.

$$V_{DAC} = DACOUT1 \bullet (AV_{DD})/4095$$

Where: 0x0CD ≤ DACDATx ≤ 0xF32

Example 16-3. Configuration of DAC Register

```
/* DAC Register Settings */
DAC1DATbits.DACDAT = 0x4D9;          /* DAC Output set to 1V (AVDD = 3.3V)*/
DAC1CONbits.DACOEN = 1;              /* Enable DAC 1 output on pin DACOUT1 */
DAC1CONbits.DACEN = 1;               /* Enable DAC 1 */
DACCTRL1bits.DACON = 1;              /* Turn ON all DACs */
```

16.4.2.1 Slope Generator

The function of the slope generator is to vary the DAC data value at a user-defined rate to reach a desired endpoint value. The slope generator, along with the DAC, has three modes of operation: Slope Generation mode, Hysteretic mode and Triangle Wave mode.

16.4.2.2 Slope Generation Mode

The slope generator function can be utilized in Peak Current-mode control-based power supply applications, where slope compensation is required. The slope function modifies the non-slope PDM DAC value repeatedly, at a user-defined rate, until the DAC data value reaches its endpoint. The slope generation function can be enabled or disabled by the SLOPEN bit (SLPxCON[31]). The slope rate is controlled by the data in the SLPxDAT register. The direction of slope being positive or negative is controlled by the PSE bit (SLPxCON[25]). For negative slopes (default, PSE = 0), DACDAT holds the nominal non-slope count, while DACLOW holds the count corresponding to the end of the slope. For positive slopes (PSE = 1), DACLOW holds the nominal non-slope count, while DACDAT holds the count corresponding to the end of the slope.

The slope generation start is controlled by the bits, SLPSTRT[3:0] (SLPxCON[3:0]). Depending on the value of SLPSTRT[3:0], the selected PWM trigger will be used to start the slope generation. The DAC output voltage changes to the value in DACDAT first by going to the Transition mode and then to the Steady-State mode. In Transition mode, the filter responds to new data values as fast as possible. The Transition mode duration is specified by the TMODTIME[9:0] bits (DACCTRL2[9:0]). The source of the clock for the DAC operation is selected by bits, CLKSEL[1:0], and the frequency of operation (F_{DAC}) is set by the divider bits, CLKDIV[1:0] of the DACCTRL1 register. The Transition mode duration, T_{TR}, is given by the equation:

$$T_{TR} = TMODTIME[9:0] * 2/F_{DAC} \text{ in Seconds}$$

Where: F_{DAC} = DAC Frequency in Hz

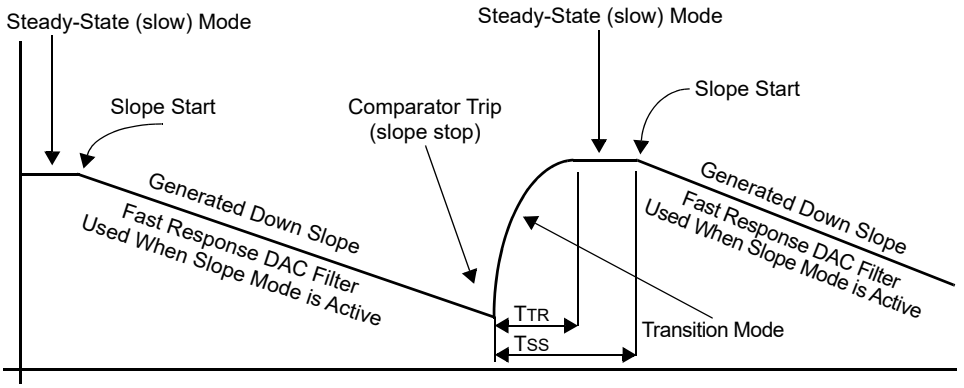
The steady-state timer, specified by the SSTIME[9:0] bits, starts at the same time as the Transition mode timer. Once the Transition mode ends, the Steady-State mode starts, wherein the DAC output voltage settles to the new value. The Steady-State Time, T_{SS}, is calculated by the equation:

T_{SS} = SSTIME[9:0] * $2/F_{DAC}$ in Seconds

Note that the SSTIME[9:0] count should always be greater than the TMODTIME[9:0] count. At the end of the Steady-State mode, the DAC value settles at the new value and is ready for slope generation. The SLPSTRT[3:0] signal triggers the slope generation process.

Refer to the DAC electrical specifications for additional information on T_{SS} and T_{TR} values. These timing parameters can be additionally adjusted as needed for the application.

Figure 16-3. Slope Generation Mode DAC Output Waveform



The slope generation is terminated when one of the two stop signals is asserted. The eight control register bits, SLPSTOPA[3:0] (SLPxCON[11:8]) and SLPSTOPB[3:0] (SLPxCON[7:4]), select the control

signal to terminate the slope generation. The stop signals are logically ORed so that the slope is terminated when one of the trigger events materializes. In most power supply applications, SLPSTOPA[3:0] can be configured to terminate the slope at the end of the PWM cycle, while SLPSTOPB[3:0] can be configured to trigger when the current reaches a limit under a normal or Fault condition. It should be noted that the stop signal must terminate the slope at least T_{SS} (Steady-State Time) prior to the next PWM cycle start. This is necessary to allow the DAC value to reach and settle at the steady-state value, specified by DACDAT, before the next cycle begins.

The slope rate value to be specified in the SLPxDAT register depends on the start and end values of the slope specified by DACDAT and DACLOW, PWM time period, DAC clock frequency and the SSTIME[9:0] bits value. The SLPxDAT value can be determined by using the below equation.

Equation 16-2. Determining the SLPxDAT Value

$$SLPxDAT = \frac{(DACDAT - DACLOW) \bullet 16}{(T_{SLOPE\_DURATION})/T_{DAC}}$$

Where:

DACDAT = DAC value at the start of slope

DACLOW = DAC value and the end of slope

$T_{SLOPE\_DURATION}$ = Slope duration time in seconds

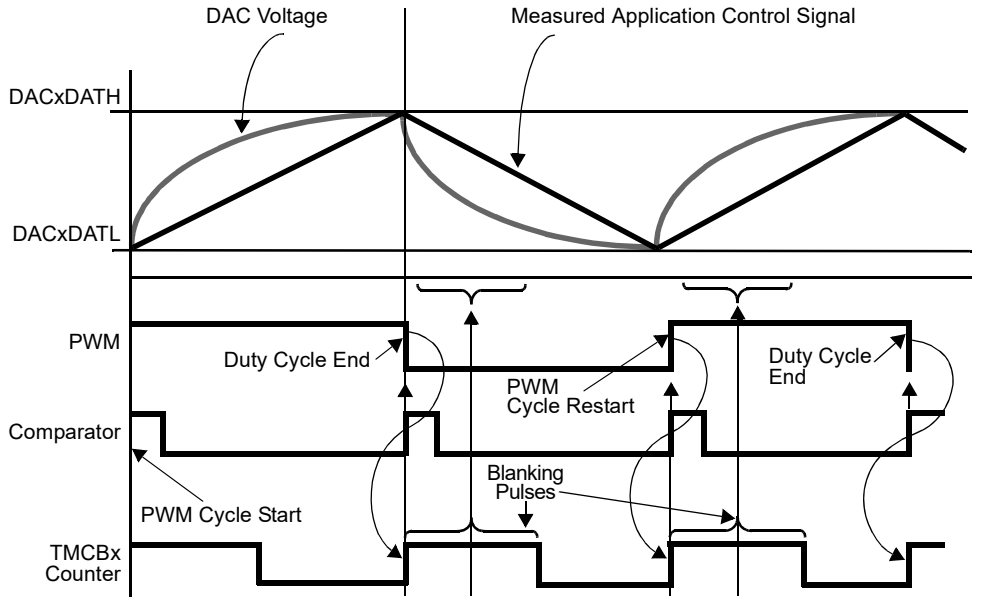
T_{DAC} = $2/F_{DAC}$ in seconds

Note: Multiplication by 16 sets results in the SLPxDAT value in 12.4 format.

16.4.2.3 Hysteretic Mode

Hysteretic mode control is sometimes called “Bang-Bang” control, where a signal within a power converter is controlled within an upper cutoff and a lower cutoff limit. The Hysteretic mode is used in power supply applications utilizing hysteretic control, such as LED drivers. The Hysteretic mode is enabled by the HME bit (SLPxCON[27]) and requires the SLOPEN bit to be cleared. Hysteretic Control mode enables a single DAC and comparator to monitor both the high and low limits for a signal. DACDAT provides the higher value, while DACLOW provides the lower value. When the DAC changes direction, it uses Transition mode to respond and reach the new value as fast as possible. In Hysteretic mode, the comparator effectively functions as a window comparator. The DAC output races ahead of the monitored voltage in the application circuit. While the DAC is transitioning to the new value, the comparator output is “blanked” via the TMCB[9:0] bits (DACxCON[23:16]) to prevent spurious responses. The state of the PWM output is monitored via the input multiplexer controlled by the HCFSEL[3:0] bits of the SLPxCON register. This module monitors the actual state of the PWM output rather than make assumptions that could damage the application circuit. A configuration example to use Hysteretic mode is shown in Figure 16-4.

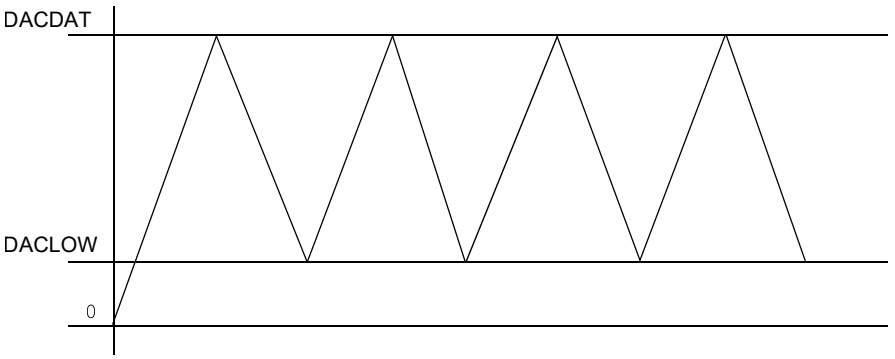
Figure 16-4. Hysteretic Mode DAC Output Waveform



16.4.2.4 Triangle Wave Mode

Triangle Wave mode generates an output voltage that rises and falls with a triangle wave pattern. The Triangle Wave mode is enabled by the bit, TWME (SLPxCON[26]), and requires SLOPEN to be set to '1'. The high and low points of the waveform are specified via DACDAT and DACLOW. The rise and fall times and the frequency of the triangle wave are controlled via the SLPxDAT register. The very first clock cycle of the slope process selects a scaled SLPxDAT value, instead of the specified value, to provide prompt DAC response to the DAC trajectory. For all subsequent clock cycles of the slope process, the slope generator uses the specified SLPxDAT data value for incrementing/decrementing the DAC data value. The Fast DAC mode is exclusive to the Triangle Wave mode and is used to provide a fast response. The slope changes the direction automatically after reaching either the DACDAT or DACLOW value. The Triangle Wave mode is useful in digital audio applications, where an analog input signal is sampled via an analog comparator using a triangle wave reference signal (Figure 16-5).

Figure 16-5. Triangle Wave Mode



A configuration example to set the Triangle Wave mode is shown in Example 16-4.

Example 16-4. Triangle Wave Mode Configuration[1]

```
/* Triangle Wave Mode Settings */
DAC1DATbits.DACLOW = 0x100;                /* Lower data value */
DAC1DATbits.DACDAT = 0xF00;                /* Upper data value */
SLP1DATbits.SLPDAT = 0x1;                  /* Slope rate, counts per step */
SLP1CONbits.TWME = 1;                      /* Enable Triangle Mode */
SLP1CONbits.SLOPEN = 1;                    /* Enable Slope Mode */
```

Note:

1. The maximum value of DACDAT must be set at 0xF32 – SLPxDAT and the minimum value of DACLOW must be set at 0xCD + SLPxDAT.

16.4.3 Operation in Sleep and Idle Mode

During Sleep mode, the High-Speed Analog Comparator operates with reduced functionality, allowing the device to wake-up when an active signal is applied to the comparator input. To reduce power consumption when the device enters Idle mode, the comparator module can be disabled by setting the DACSIDL bit (DACCTRL1[13]). The DACSIDL bit controls all the comparators on a device or device core.

16.4.4 Calibration

The DAC supports offset calibration. This is accomplished by software copying calibration data from the FPDMDAC configuration register to the POSINLADJ[5:0] and NEGINLADJ[6:0] bitfields within the DACCTRL1 SFR. The address of FPDMDAC is device dependent and is located in 16.1. Device-Specific Information.

16.5 Application Examples

The High-Speed Analog Comparator with Slope Compensation DAC can be used in many power conversion applications. The outputs of the comparator module can be used to perform the following functions:

- Generate an Interrupt
- Trigger an ADC Sample and Convert Process
- Truncate the PWM Signal (Current Limit)
- Truncate the PWM Period (Current Reset)
- Extend the PWM Period (Feed Forward)
- Disable the PWM Outputs (Fault Latch)

The output of the comparator module can be used in multiple modes at the same time. For example, the comparator output can be used to generate an interrupt, have the ADC take a sample and convert it, and truncate the PWM output, all in response to a voltage being detected beyond its expected value. The SMPS analog comparator module can also be used to wake-up the system from Sleep mode or Idle mode when the analog input voltage exceeds the programmed threshold voltage. The slope compensation module allows the user to utilize built-in hardware-based slope compensation in SMPS applications. The potential applications of the comparator module are numerous and varied.

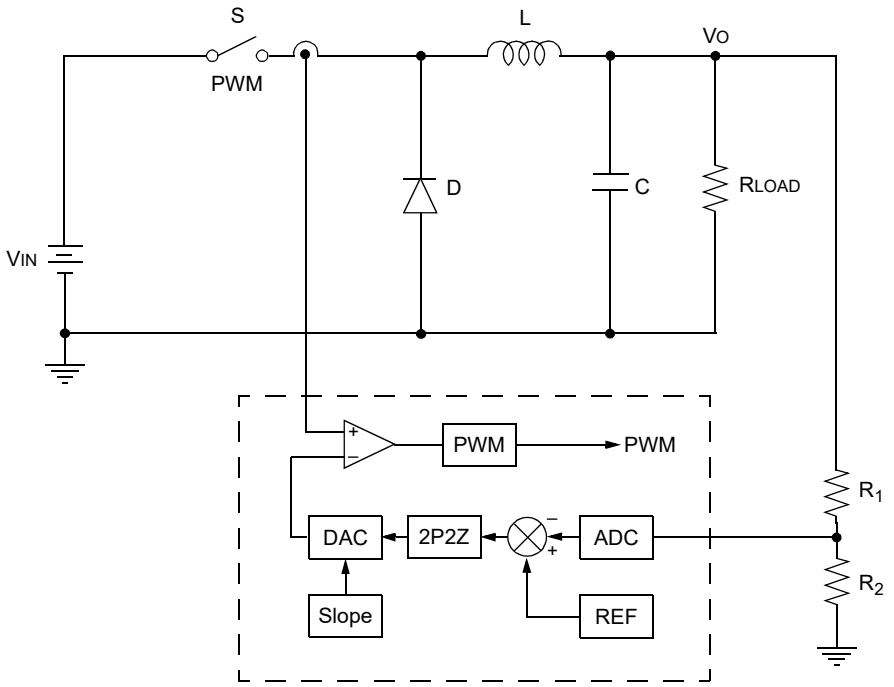
The following section describes typical applications of the comparator module in power conversion circuits.

16.5.1 Peak Current-Mode Control

The SMPS topologies, such as Buck, Boost and Buck-Boost, generate subharmonic oscillations when controlled with Peak Current-mode control. These oscillations occur under specific conditions, such

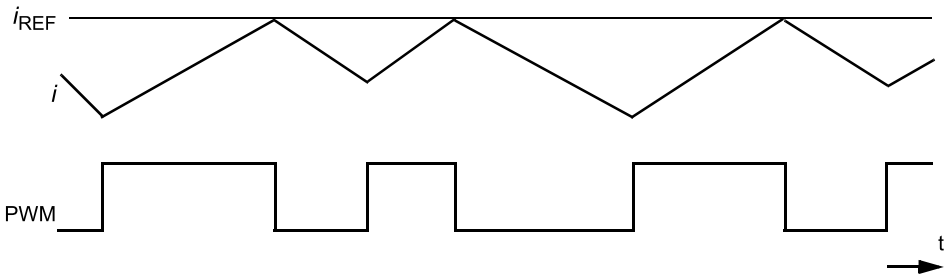
as Continuous Current-mode and a duty cycle greater than 50%. The subharmonic oscillations can be damped by using slope compensation. The analog comparator module can be utilized for such applications, eliminating the need for additional external analog circuitry to perform slope compensation. The comparator module is used in conjunction with the PWM module to generate the Current-mode PWM signal. A typical Peak Current Buck mode power supply is illustrated in Figure 16-6.

Figure 16-6. Buck Converter with Peak Current-Mode Control



The analog comparator module is configured to reset the PWM module when the measured inductor current peak reaches the current level determined by the outer control loop. The outer control loop consists of the output voltage, measured by the ADC, and compared with a desired voltage reference. The error counts generated are treated with a compensator gain to arrive at the peak current level for the current PWM cycle. The peak current level is applied to the DAC to generate an equivalent analog signal with which the actual inductor current is compared by the comparator. The waveforms of the Peak Current-mode control are as shown in Figure 16-7. Note that the pulse width is different in the consecutive cycles, even though the current reference, i_{REF}, is constant.

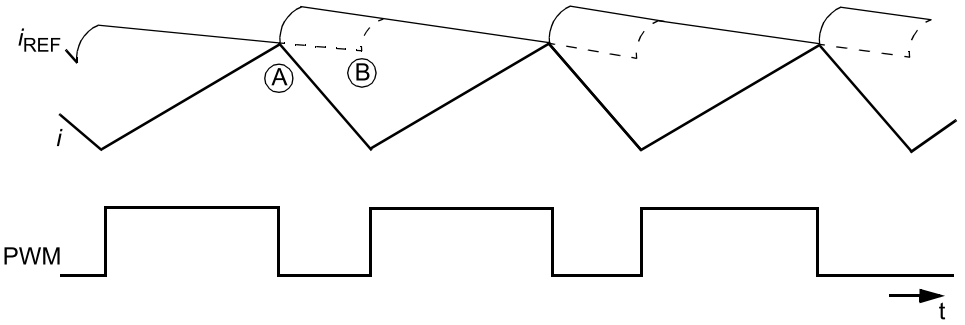
Figure 16-7. Peak Current-Mode Waveform without Slope Compensation



The slope compensation module alters the DAC output voltage slope, hence the reference to the inner current loop. In the absence of the slope compensation module, the output of the DAC is held

constant for a given PWM cycle (as shown in Figure 16-8). The slope generation module causes the reference current to slope based on the values set in the register. The slope direction can be set to positive or negative depending on the applications, although negative slope is generally used. The rate of the slope is determined by the SLPxDAT register. DACDAT holds the DAC value at the start of the PWM cycle, and the DAC value at the end of the PWM cycle is held by DACLOW. The waveforms of the Peak Current-mode control with slope compensation are as shown in Figure 16-8. Note that the pulse width is the same in all cycles for a constant current reference, i_{REF}.

Figure 16-8. Peak Current-Mode Waveform with Slope Compensation



Example 16-5 shows the settings of the analog comparator module for generating the slope compensated waveforms. In this design, the buck converter operates at an input of 5V and an output of 3.3V, 1A. The operating frequency of the converter is 200 kHz. The period timer for the PWM is set at 200 kHz, and the duty cycle is set at 95%. The clock frequency for the DAC module is set at 500 MHz. The current measurement is connected to the positive input of the comparator. The DAC provides the reference current for the peak current trip and is connected internally to the negative input of the comparator. The reference current is generally the output of the compensator (digital filter), which operates on the outer voltage loop error signal. The PWM cycle is terminated when the measured input current exceeds the DAC reference current, i_{REF}, as shown by Point A in Figure 16-8. This is due to configuration of the SLPSTOPBx trigger in Figure 16-6.The dashed curve shows the reference current, i_{REF}, if SLPSTOPBx is not triggered. In this case, i_{REF} continues until the SLPSTOPAx signal is triggered and is indicated by Point B in Figure 16-8.

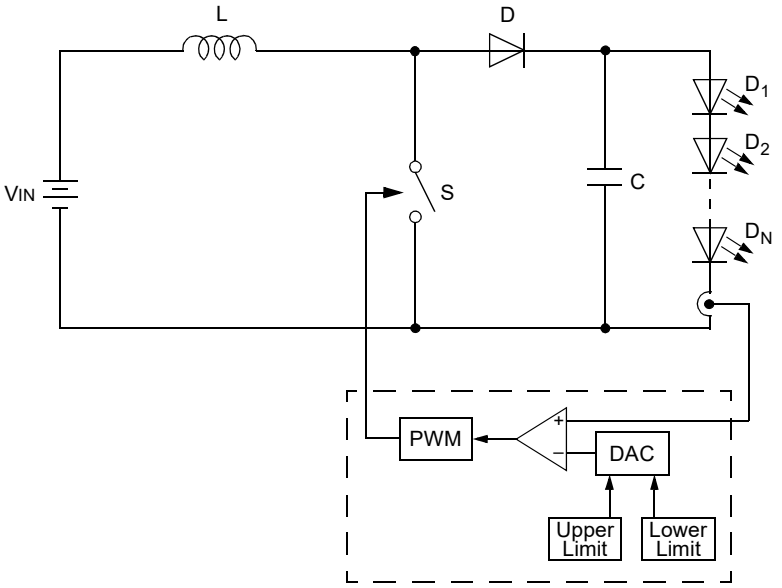
Example 16-5. Initialize DAC with Slope Compensation

```
    // PWM Configuration
    PCLKCONbits.MCLKSEL = 1;    // AFVCO/2 as clock source (500 Mhz)
    PG1CONLbits.CLKSEL = 1;     // Clock selected by MCLKSEL
    PG1PER = 2499;              // PWM frequency is 200 kHz, 5 uS period
    PG1DC = 2375;               // 95% duty cycle, 4.75 uS on time
    PG1IOCONHbits.PENH = 1;     // PWM Generator controls the PWMxH output pin
    PG1IOCONHbits.PENL = 1;     // PWM Generator controls the PWMxL output
pin
    // PWM PCI setup, use CLDAT when comparator 1 trips
    PG1CLPCILbits.PSS = 27;     // PCI source is Comparator 1
    PG1CLPCILbits.AQSS = 2;     // LEB active as Acceptance Qualifier
    PG1CLPCILbits.AQPS = 1;     // Invert Acceptance Qualifier (LEB not active)
    PG1CLPCILbits.TERM = 1;     // Auto terminate as Termination Event
    PG1CLPCIHbits.ACP = 3;      // Latched PCI Acceptance Criteria
    PG1IOCONLbits.CLDAT = 0b01; // PWM1L = 1 and PWM1H = 0 if CL event is
active
    PG1LEBHbits.PHR = 1;        // Rising edge of PWMxH triggers the LEB
counter
    PG1LEBL = 30;               // 500 nS LEB timer
    // PWM to DAC Trigger setup
    PG1TRIGA = 750;             // ADC Trigger 1 at 1.5 uS, used as SLPSTRT
    PG1EVTLbits.ADTR1EN1 = 1;   // PGxTRIGA as trigger source for ADC Trigger 1
    PG1TRIGB = 2000;            // ADC Trigger 2 at 4 uS, used as SLPSTOPA
    PG1EVTHbits.ADTR2EN2 = 1;   // PGxTRIGB as trigger source for ADC Trigger 2
    PG1CONLbits.ON = 1;         // Enable PWM
    // DAC Configuration
    DACCTRL1bits.CLKSEL = 0;    // AFVCO/2 as clock source (500 Mhz)
    DACCTRL2bits.SSTIME = 0x8A; // Default value 552 nS @ 500MHz
    DACCTRL2bits.TMODTIME = 0x55;// Default value 340 nS @ 500MHz
    DAC1DATbits.DACDAT = 2703;  // 2.17v steady state value
    DAC1DATbits.DACLOW = 1113;  // 0.89v, value at the end slope
    SLP1DATbits.SLPDAT = 41;    // Slope = (2703-1113)*16/((4u-1.5u)/4n)
    SLP1CONbits.SLOPEN = 1;     // Enable Slope compensation
    SLP1CONbits.SLPSTRT = 1;    // PWM1 ADC Trigger 1
    SLP1CONbits.SLPSTOPA = 1;   // PWM1 ADC Trigger 2
    SLP1CONbits.SLPSTOPB = 1;   // Comparator 1
    DAC1CONbits.CBE = 1;        // Enable comparator blanking
    DAC1CONbits.TMCB = 125;     // 125 * 4 nS = 500 nS blanking time
    DAC1CONbits.DACOEN = 1;     // Enable DAC output
    DAC1CONbits.DACEN = 1;      // Enable DAC 1
    DACCTRL1bits.DACON = 1;     // Enable DAC system
```

16.5.2 Hysteretic Control for LED Drivers

The hysteretic control offers the fastest response to the changing parameters, such as voltage or current. The hysteretic control finds wide usage in LED applications, where the current is required to be in a limited range around the average value. The disadvantage of the hysteretic topology is the variable frequency of operation. Figure 16-9 shows an example circuit where the LED current is controlled to an average value with a tolerance decided by the register values.

Figure 16-9. Hysteretic Control for LED Drivers



The comparator uses Hysteretic mode for such applications. The Hysteretic mode is controlled by the HME bit in the SLPxCON register. In order to enable the Hysteretic mode, the HME bit must be set to '1' and the SLOPEN bit is cleared. The upper limit of the hysteretic control is defined by DACDAT, while the lower limit is defined by DACLOW. In Hysteretic mode, the comparator module has a pair of output signals that are available as Peripheral Pin Select (PPS) inputs: PWM_Req_on and PWM_Req_off. These signals can then be mapped to the PWM PCI input for controlling the PWM outputs. The DAC settings for Hysteretic mode are shown in Example 16-6.

Example 16-6. DAC Settings for Hysteretic Mode

```
/* Clock Selection */
PCLKCONbits.MCLKSEL = 3;        // Master Clock Source is APLL
PG1CONLbits.CLKSEL = 1;         // Clock selected by MCLKSEL
while(!_APLLCK)                 // Wait for PLL lock
PG1IOCONHbits.PENH = 1;         // Enable H output
PG1IOCONHbits.PENL = 1;         // Enable L output
PG1IOCONLbits.FFDAT = 0b11;     // FF PCI data is 0b11
// PPS setup
RPINR12bits.PCI8R = 168;        // 'PWM_Req_On' from DAC to PCI8
RPINR12bits.PCI9R = 169;        // 'PWM_Req_Off' from DAC to PCI9
// FF PCI setup
PG1FFPCILbits.TSYNCDIS = 1;     // Terminate immediately
PG1FFPCILbits.TERM = 0b111;     // PCI 9 (PWM_Req_Off)
PG1FFPCILbits.PSS = 8;          // PCI 8 (PWM_Req_On)
PG1FFPCIHbits.ACP = 0b100;      // Latched rising edge
PG1CONLbits.ON = 1;             // Enable PG1
// DAC initialization
DACCTRL1bits.CLKSEL = 2;        // APLL
DAC1DATbits.DACLOW = 0x400;     // Lower cmp limit, 0.825 V
DAC1DATbits.DACDAT = 0xC00;     // Upper cmp limit, 2.475 V
DAC1CONbits.CBE = 1;            // Enable comparator blanking
DAC1CONbits.TMCB = 100;         // 2/500 MHz * 100 = 400 nS
SLP1CONbits.HCFSEL = 1;         // 1 = PWM1H
SLP1CONbits.HME = 1;            // Hysteretic Mode
DAC1CONbits.INSEL = 1;          // CMP1B input
DAC1CONbits.DACOEN = 1;         // Output DAC voltage to DACOUT1 pin
DAC1CONbits.DACEN = 1;          // Enable DAC module
DACCTRL1bits.DACON = 1;         // Enable DAC1
```

16.5.3 Using DAC as Voltage Reference for External Comparator

If the DAC output is used as a reference voltage for a comparator external to the dsPIC33 device, the Transition mode's transient response can cause unwanted comparator trips when changing DACDAT values. When the DAC is in Transition mode, the blanking feature (CBE = 1) can be set to mask the internal comparator's output. However, this blanking feature is not available on an external comparator.

A work around for this use case is to set the TMODTIME[9:0] bits in the DACCTRL2 register to zero to disable Transition mode. The DAC will slew slower to the new target voltage.

17. Quadrature Encoder Interface (QEI)

This section describes the Quadrature Encoder Interface (QEI) implemented in dsPIC33A devices. The QEI is typically used in motor control applications to detect the mechanical position, direction or rotation and speed of rotation of quadrature encoders. The following high-level features are covered in this section:

- Four Input Pins: Two Phase Signals, an Index Pulse and a Home Pulse
- Programmable Digital Noise Filters on Inputs
- Quadrature Decoder Providing Counter Pulses and Count Direction
- x4 Count Resolution
- Index Pulse to Reset the Position Counter
- General Purpose 32-bit Timer/Counter Mode
- Interrupts Generated by QEI or Counter Events
- 32-bit Velocity Counter
- 32-bit Position Counter
- 32-bit Index Pulse Counter
- 32-bit Interval Timer
- 32-bit Position Initialization/Capture/Compare High Word Register
- 32-bit Position Initialization/Capture/Compare Low Word Register
- 4X Quadrature Count Mode
- External Up/Down Count Mode
- External Gated Count Mode
- External Gated Timer Mode
- Interval Timer Mode

17.1 Device-Specific Information

Table 17-1. QEI Summary

QEI Module Instances	Peripheral Bus Speed	Clock Source
1	Standard	Standard Speed Peripheral Clock

17.2 Architectural Overview

The Quadrature Encoder Interface (QEI) module provides the interface to incremental encoders for obtaining mechanical position data. Quadrature encoders, also known as incremental encoders or optical encoders, detect position and speed of rotating motion systems. Quadrature encoders enable closed-loop control of motor control applications, such as Switched Reluctance (SR) and AC Induction Motors (ACIM).

A typical quadrature encoder includes a slotted wheel attached to the shaft of the motor and an emitter/detector module that senses the slots in the wheel. Typically, three output channels, Phase A (QEAx), Phase B (QEBx) and Index (INDXx) provide information on the movement of the motor shaft, including distance and direction.

The two channels, Phase A (QEA) and Phase B (QEB), are typically 90° out of phase with respect to each other. The Phase A and Phase B channels have a unique relationship. If Phase A leads Phase B, the direction of the motor is deemed positive or forward. If Phase A lags Phase B, the direction of the motor is deemed negative or reverse. The index pulse occurs once per mechanical revolution

and is used as a reference to indicate an absolute position. Figure 17-1 illustrates the Quadrature Encoder Interface signals.

The quadrature signals from the encoder can have four unique states ('01', '00', '10' and '11') that reflect the relationship between QEA and QEB. Figure 17-1 illustrates these states for one count cycle. The order of the states get reversed when the direction of travel changes.

The quadrature decoder increments or decrements the 32-bit up/down Position Counter (POSxCNT) for each Change-of-State (COS). The counter increments when QEA leads QEB and decrements when QEB leads QEA.

Figure 17-1. Quadrature Encoder Interface Signals

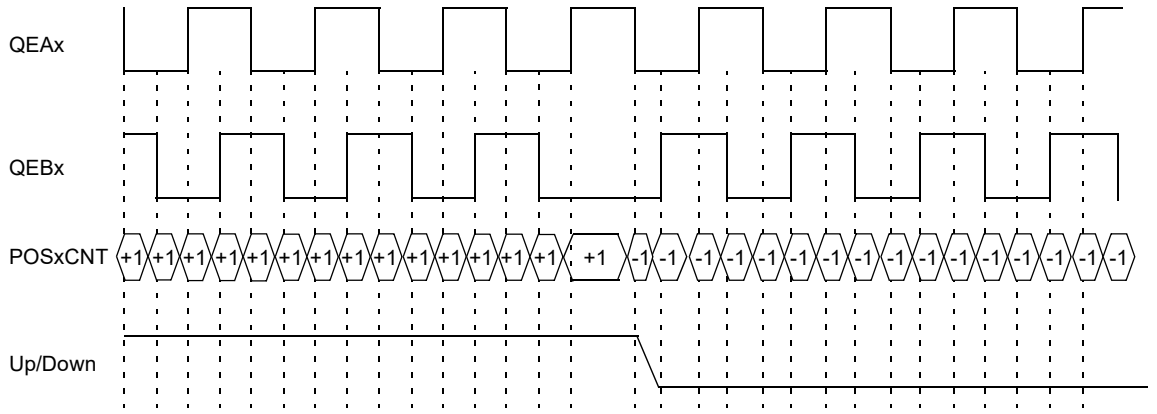


Table 2-1 shows the truth table that describes how the quadrature signals are decoded.

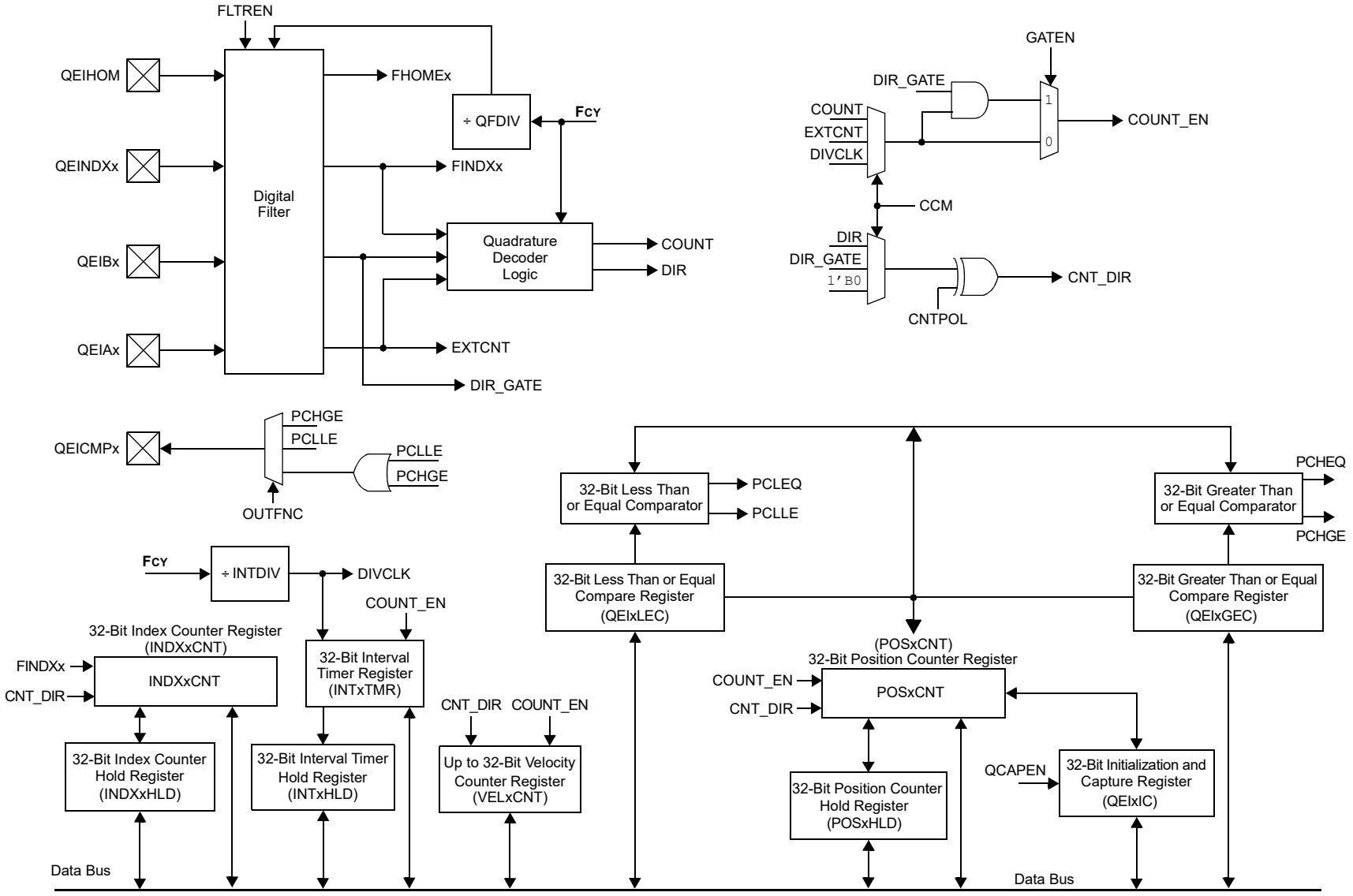
Table 17-2. Truth Table for Quadrature Encoder

Current Quadrature State		Previous Quadrature State		Action
QEA	QEB	QEA	QEB	
1	1	1	1	No count or direction change
1	1	1	0	Count up
1	1	0	1	Count down
1	1	0	0	Invalid state change, ignore
1	0	1	1	Count down
1	0	1	0	No count or direction change
1	0	0	1	Invalid state change, ignore
1	0	0	0	Count up
0	1	1	1	Count up
0	1	1	0	Invalid state change, ignore
0	1	0	1	No count or direction change
0	1	0	0	Count down
0	0	1	1	Invalid state change, ignore
0	0	1	0	Count down
0	0	0	1	Count up
0	0	0	0	No count or direction change

Figure 17-2 illustrates the simplified block diagram of the QEI module. The QEI module consists of decoder logic to interpret the Phase A (QEA) and Phase B (QEB) signals, and an up/down counter to accumulate the count. The counter pulses are generated when the quadrature state changes. The

count direction information must be maintained in a register until a direction change is detected. The module also includes digital noise filters that condition the input signal.

Figure 17-2. Quadrature Encoder Interface (QEI) Module Block Diagram



17.3 QEI Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1A00	QEI1CON	31:24								
		23:16								
		15:8	QEIEN		QEISIDL	PIMOD[2:0]			IMV[1:0]	
		7:0		INTDIV[2:0]			CNTPOL	GATEN	CCM[1:0]	
0x1A04	QEI1IOC	31:24								
		23:16								HCAPEN
		15:8	QCAPEN	FLTREN	QFDIV[2:0]			OUTFNC[1:0]		SWPAB
		7:0	HOMPOL	IDXPOL	QEBPOL	QEAPOL	HOME	INDEX	QEB	QEA
0x1A08	QEI1STAT	31:24								
		23:16								
		15:8			PCHEQIRQ	PCHEQIEN	PCLEQIRQ	PCLEQIEN	POSOVIRQ	POSOVIEN
		7:0	PCIIRQ	PCIIEN	VELOVIRQ	VELOVIEN	HOMIRQ	HOMIEN	IDXIRQ	IDXIEN
0x1A0C	POS1CNT	31:24	POSCNT[31:24]							
		23:16	POSCNT[23:16]							
		15:8	POSCNT[15:8]							
		7:0	POSCNT[7:0]							
0x1A10	POS1HLD	31:24	POSHLD[31:24]							
		23:16	POSHLD[23:16]							
		15:8	POSHLD[15:8]							
		7:0	POSHLD[7:0]							
0x1A14	VEL1CNT	31:24	VELCNT[31:24]							
		23:16	VELCNT[23:16]							
		15:8	VELCNT[15:8]							
		7:0	VELCNT[7:0]							
0x1A18	VEL1HLD	31:24	VELHLD[31:24]							
		23:16	VELHLD[23:16]							
		15:8	VELHLD[15:8]							
		7:0	VELHLD[7:0]							
0x1A1C	INT1TMR	31:24	INTTMR[31:24]							
		23:16	INTTMR[23:16]							
		15:8	INTTMR[15:8]							
		7:0	INTTMR[7:0]							
0x1A20	INT1HLD	31:24	INTHLD[31:24]							
		23:16	INTHLD[23:16]							
		15:8	INTHLD[15:8]							
		7:0	INTHLD[7:0]							
0x1A24	INDX1CNT	31:24	INDXCNT[31:24]							
		23:16	INDXCNT[23:16]							
		15:8	INDXCNT[15:8]							
		7:0	INDXCNT[7:0]							
0x1A28	INDX1HLD	31:24	INDXHLD[31:24]							
		23:16	INDXHLD[23:16]							
		15:8	INDXHLD[15:8]							
		7:0	INDXHLD[7:0]							
0x1A2C	QEI1IC	31:24	QEIIC[31:24]							
		23:16	QEIIC[23:16]							
		15:8	QEIIC[15:8]							
		7:0	QEIIC[7:0]							
0x1A2C	QEI1GEC	31:24	QEIGEC[31:24]							
		23:16	QEIGEC[23:16]							
		15:8	QEIGEC[15:8]							
		7:0	QEIGEC[7:0]							
0x1A30	QEI1LEC	31:24	QEILEC[31:24]							
		23:16	QEILEC[23:16]							
		15:8	QEILEC[15:8]							
		7:0	QEILEC[7:0]							

17.3.1 QEI Control Register

Name: QEI1CON
Offset: 0x1A00

Notes:

1. When CCMx = 10 or CCMx = 11, all of the QEI counters operate as timers and the PIMOD[2:0] bits are ignored.
2. When CCMx = 00, and QEAx and QEBx values match the Index Match Value (IMV), the POSxCNT registers are reset.
3. The selected clock rate should be at least twice the expected maximum quadrature count rate.
4. The index match value applies to the A&B inputs after the SWAP, and polarity bits have been applied.
5. The QCAPEN and HCAPEN bits must be cleared during PIMODx Modes two through seven to ensure proper functionality.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	QEIEN		QEISIDL	PIMOD[2:0]			IMV[1:0]	
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		INTDIV[2:0]			CNTPOL	GATEN	CCM[1:0]	
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 15 – QEIEN Quadrature Encoder Interface Module Counter Enable bit

Value	Description
1	Module counters are enabled
0	Module counters are disabled, but SFRs can be read or written

Bit 13 – QEISIDL QEI Stop in Idle Mode bit

Value	Description
1	Discontinues module operation when device enters Idle mode
0	Continues module operation in Idle mode

Bits 12:10 – PIMOD[2:0] Position Counter Initialization Mode Select bits(1,5)

Value	Description
111	Modulo Count mode for position counter and every index event resets the position counter
110	Modulo Count mode for position counter
101	Resets the position counter when the position counter equals the QEIxGEC register

Value	Description
100	Second index event after home event initializes the position counter with the contents of the QEIxIC register
011	First index event after home event initializes the position counter with the contents of the QEIxIC register
010	Next index input event initializes the position counter with the contents of the QEIxIC register
001	Every index input event resets the position counter
000	Index input event does not affect the position counter

Bits 9:8 – IMV[1:0] Index Match Value bits(2,4)

Value	Description
11	Index match occurs when QEBx = 1 and QEAx = 1
10	Index match occurs when QEBx = 1 and QEAx = 0
01	Index match occurs when QEBx = 0 and QEAx = 1
00	Index match occurs when QEBx = 0 and QEAx = 0

Bits 6:4 – INTDIV[2:0] Timer Input Clock Prescale Select bits(3)
(Interval timer, main timer (position counter), velocity counter and index counter internal clock divider select)

Value	Description
111	1:128 prescale value
110	1:64 prescale value
101	1:32 prescale value
100	1:16 prescale value
011	1:8 prescale value
010	1:4 prescale value
001	1:2 prescale value
000	1:1 prescale value

Bit 3 – CNTPOL Position and Index Counter/Timer Direction Select bit

Value	Description
1	Counter direction is negative unless modified by an external up/down signal
0	Counter direction is positive unless modified by an external up/down signal

Bit 2 – GATEN External Count Gate Enable bit

Value	Description
1	External gate signal controls position counter operation
0	External gate signal does not affect position counter operation

Bits 1:0 – CCM[1:0] Counter Control Mode Selection bits

Value	Description
11	Internal Timer mode
10	External Clock Count with External Gate mode
01	External Clock Count with External Up/Down mode
00	Quadrature Encoder mode

17.3.2 QEI I/O Control Register

Name: QEI1IOC
Offset: 0x1A04

Legend: x = Bit is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
								HCAPEN
Access								R/W
Reset								0

Bit	15	14	13	12	11	10	9	8
	QCAPEN	FLTREN	QFDIV[2:0]			OUTFNC[1:0]		SWPAB
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HOMPOL	IDXPOL	QEBPOL	QEAPOL	HOME	INDEX	QEB	QEA
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	x	x	x	x

Bit 16 – HCAPEN Position Counter Input Capture by Home Event Enable bit

Value	Description
1	HOMEx input event (positive edge) triggers a position capture event (QCAPEN must be cleared)
0	HOMEx input event (positive edge) does not trigger a position capture event

Bit 15 – QCAPEN QEI Position Counter Input Capture by Index Match Event Enable bit

Value	Description
1	Index match event (positive edge) triggers a position capture event (HCAPEN must be cleared)
0	Index match event (positive edge) does not trigger a position capture event

Bit 14 – FLTREN QEAx/QEBx/INDXx/HOMEx Digital Filter Enable bit

Value	Description
1	Input pin digital filter is enabled
0	Input pin digital filter is disabled (bypassed)

Bits 13:11 – QFDIV[2:0] QEAx/QEBx/INDXx/HOMEx Digital Input Filter Clock Divide Select bits

Value	Description
111	1:128 clock divide
110	1:64 clock divide
101	1:32 clock divide
100	1:16 clock divide
011	1:8 clock divide
010	1:4 clock divide
001	1:2 clock divide
000	1:1 clock divide

Bits 10:9 – OUTFNC[1:0] QEI Module Output Function Mode Select bits

Value	Description
11	The CNTCMPx pin goes high when POSxCNT ≤ QEIxLEC or POSxCNT ≥ QEIxGEC
10	The CNTCMPx pin goes high when POSxCNT ≤ QEIxLEC
01	The CNTCMPx pin goes high when POSxCNT ≥ QEIxGEC
00	Output is disabled

Bit 8 – SWPAB Swap QEAx and QEBx Inputs bit

Value	Description
1	QEAx and QEBx are swapped prior to Quadrature Decoder logic
0	QEAx and QEBx are not swapped

Bit 7 – HOMPOL HOMEx Input Polarity Select bit

Value	Description
1	Input is inverted
0	Input is not inverted

Bit 6 – IDXPOL INDXx Input Polarity Select bit

Value	Description
1	Input is inverted
0	Input is not inverted

Bit 5 – QEBPOL QEBx Input Polarity Select bit

Value	Description
1	Input is inverted
0	Input is not inverted

Bit 4 – QEAPOL QEAx Input Polarity Select bit

Value	Description
1	Input is inverted
0	Input is not inverted

Bit 3 – HOME Status of HOMEx Input Pin After Polarity Control bit (read-only)

Value	Description
1	Pin is at logic '1' if HOMPOL bit is set to '0'; Pin is at logic '0' if HOMPOL bit is set to '1'
0	Pin is at logic '0' if HOMPOL bit is set to '0'; Pin is at logic '1' if HOMPOL bit is set to '1'

Bit 2 – INDEX Status of INDXx Input Pin After Polarity Control bit (read-only)

Value	Description
1	Pin is at logic '1' if the IDXPOL bit is set to '0'; Pin is at logic '0' if the IDXPOL bit is set to '1'
0	Pin is at logic '0' if the IDXPOL bit is set to '0'; Pin is at logic '1' if the IDXPOL bit is set to '1'

Bit 1 – QEB Status of QEBx Input Pin After Polarity Control and SWPAB Pin Swapping bit (read-only)

Value	Description
1	Physical pin, QEBx, is at logic '1' if QEBPOL bit is set to '0' and SWPAB bit is set to '0'; physical pin, QEBx, is at logic '0' if QEBPOL bit is set to '1' and SWPAB bit is set to '0'; physical pin, QEAx, is at logic '1' if QEBPOL bit is set to '0' and SWPAB bit is set to '1'; physical pin, QEAx, is at logic '0' if QEBPOL bit is set to '1' and SWPAB bit is set to '1'
0	Physical pin, QEBx, is at logic '0' if QEBPOL bit is set to '0' and SWPAB bit is set to '0'; physical pin, QEBx, is at logic '1' if QEBPOL bit is set to '1' and SWPAB bit is set to '0'; physical pin, QEAx, is at logic '0' if QEBPOL bit is set to '0' and SWPAB bit is set to '1'; physical pin, QEAx, is at logic '1' if QEBPOL bit is set to '1' and SWPAB bit is set to '1'

Bit 0 – QEA Status of QEAx Input Pin After Polarity Control and SWPAB Pin Swapping bit (read-only)

Value	Description
1	Physical pin, QEAx, is at logic '1' if QEAPOL bit is set to '0' and SWPAB bit is set to '0'; physical pin, QEAx, is at logic '0' if QEAPOL bit is set to '1' and SWPAB bit is set to '0'; physical pin, QEBx, is at logic '1' if QEAPOL bit is set to '0' and SWPAB bit is set to '1'; physical pin, QEBx, is at logic '0' if QEAPOL bit is set to '1' and SWPAB bit is set to '1'
0	Physical pin, QEAx, is at logic '0' if QEAPOL bit is set to '0' and SWPAB bit is set to '0'; physical pin, QEAx, is at logic '1' if QEAPOL bit is set to '1' and SWPAB bit is set to '0'; physical pin, QEBx, is at logic '0' if QEAPOL bit is set to '0' and SWPAB bit is set to '1'; physical pin, QEBx, is at logic '1' if QEAPOL bit is set to '1' and SWPAB bit is set to '1'

17.3.3 QEI Status Register

Name: QEI1STAT
Offset: 0x1A08

Note:
1. This status bit is only applicable to PIMOD[2:0] modes, '011' and '100'.

Legend: C = Clearable bit, HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			PCHEQIRQ	PCHEQIEN	PCLEQIRQ	PCLEQIEN	POSOVIRQ	POSOVIEN
Access			R/W/HS	R/W	R/W/HS	R/W	R/W/HS	R/W
Reset			0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PCIIRQ	PCIIEN	VELOVIRQ	VELOVIEN	HOMIRQ	HOMIEN	IDXIRQ	IDXIEN
Access	R/W/HS	R/W	R/W/HS	R/W	R/W/HS	R/W	R/W/HS	R/W
Reset	0	0	0	0	0	0	0	0

Bit 13 – PCHEQIRQ Position Counter Greater Than Compare Status bit

Value	Description
1	POSxCNT > QEIxGEC
0	POSxCNT < QEIxGEC

Bit 12 – PCHEQIEN Position Counter Greater Than Compare Interrupt Enable bit

Value	Description
1	Position Counter Greater Than Compare Interrupt is enabled
0	Interrupt is disabled

Bit 11 – PCLEQIRQ Position Counter Less Than Compare Status bit

Value	Description
1	POSxCNT < QEIxLEC
0	POSxCNT > QEIxLEC

Bit 10 – PCLEQIEN Position Counter Less Than Compare Interrupt Enable bit

Value	Description
1	Position Counter Less Than Compare Interrupt is enabled
0	Interrupt is disabled

Bit 9 – POSOVIRQ Position Counter Overflow Status bit

Value	Description
1	Position Counter Overflow has occurred
0	Position Counter Overflow has not occurred

Bit 8 – POSOVIEN Position Counter Overflow Interrupt Enable bit

Value	Description
1	Position Counter Overflow Interrupt is enabled
0	Position Counter Overflow Interrupt is disabled

Bit 7 – PCIIRQ Position Counter (Homing) Initialization Process Complete Status bit(1)

Value	Description
1	POSxCNT was reinitialized
0	POSxCNT was not reinitialized

Bit 6 – PCIIEN Position Counter (Homing) Initialization Process Complete Interrupt Enable bit

Value	Description
1	Position Counter (Homing) Initialization Process Complete Interrupt is enabled
0	Position Counter (Homing) Initialization Process Complete Interrupt is disabled

Bit 5 – VELOVIRQ Velocity Counter Overflow Status bit

Value	Description
1	Velocity Counter Overflow has occurred
0	Velocity Counter Overflow has not occurred

Bit 4 – VELOVIEN Velocity Counter Overflow Interrupt Enable bit

Value	Description
1	Velocity Counter Overflow Interrupt is enabled
0	Velocity Counter Overflow Interrupt is disabled

Bit 3 – HOMIRQ Home Event Status bit

Value	Description
1	Home event has occurred
0	No home event has occurred

Bit 2 – HOMIEN Home Input Event Interrupt Enable bit

Value	Description
1	Home Input Event Interrupt is enabled
0	Home Input Event Interrupt is disabled

Bit 1 – IDXIRQ Index Event Status bit

Value	Description
1	Index event has occurred
0	No index event has occurred

Bit 0 – IDXIEN Index Input Event Interrupt Enable bit

Value	Description
1	Index Input Event Interrupt is enabled
0	Index Input Event Interrupt is disabled

17.3.4 Position Counter Register

Name: POS1CNT
Offset: 0x1A0C

Bit	31	30	29	28	27	26	25	24
	POSCNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	POSCNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	POSCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	POSCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – POSCNT[31:0] Position Counter Register bits

17.3.5 Position Counter Hold Register

Name: POS1HLD
Offset: 0x1A10

Bit	31	30	29	28	27	26	25	24
	POSHLD[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	POSHLD[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	POSHLD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	POSHLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – POSHLD[31:0] Position Counter hold register bits

17.3.6 Velocity Counter Register

Name: VEL1CNT
Offset: 0x1A14

Bit	31	30	29	28	27	26	25	24
	VELCNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	VELCNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	VELCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VELCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – VELCNT[31:0] Velocity Counter bits

17.3.7 Position Counter Hold Register

Name: VEL1HLD
Offset: 0x1A18

Bit	31	30	29	28	27	26	25	24
	VELHLD[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	23	22	21	20	19	18	17	16
	VELHLD[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	15	14	13	12	11	10	9	8
	VELHLD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	7	6	5	4	3	2	1	0
	VELHLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 31:0 – VELHLD[31:0] Velocity Counter Hold Register bits

17.3.8 Interval Timer Register

Name: INT1TMR
Offset: 0x1A1C

Bit	31	30	29	28	27	26	25	24
	INTTMR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INTTMR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INTTMR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INTTMR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – INTTMR[31:0] Interval Timer Register bits

17.3.9 Interval Timer Hold Register

Name: INT1HLD
Offset: 0x1A20

Bit	31	30	29	28	27	26	25	24
	INTHLD[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INTHLD[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INTHLD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INTHLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – INTHLD[31:0] Interval Timer Hold Register (INTxHLD) bits

17.3.10 Index Counter Register

Name: INDX1CNT
Offset: 0x1A24

Bit	31	30	29	28	27	26	25	24
	INDXCNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INDXCNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INDXCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INDXCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – INDXCNT[31:0] Index Counter Value bits

17.3.11 Index Counter Hold Register

Name: INDX1HLD
Offset: 0x1A28

Bit	31	30	29	28	27	26	25	24
	INDXHLD[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INDXHLD[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INDXHLD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INDXHLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – INDXHLD[31:0] Index Counter Hold Register bits

17.3.12 Initialization/Capture Register

Name: QEI1IC
Offset: 0x1A2C

Bit	31	30	29	28	27	26	25	24
	QEIIC[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	23	22	21	20	19	18	17	16
	QEIIC[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	15	14	13	12	11	10	9	8
	QEIIC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	7	6	5	4	3	2	1	0
	QEIIC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 31:0 – QEIIC[31:0] Initialize/Capture bits

17.3.13 QEI Greater Than or Equal Compare Register

Name: QEI1GEC
Offset: 0x1A2C

Bit	31	30	29	28	27	26	25	24
	QEIGEC[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	QEIGEC[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	QEIGEC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	QEIGEC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – QEIGEC[31:0] Greater Than or Equal Compare bits

17.3.14 Less Than or Equal Compare Register

Name: QEI1LEC
Offset: 0x1A30

Bit	31	30	29	28	27	26	25	24
	QEILEC[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	QEILEC[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	QEILEC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	QEILEC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – QEILEC[31:0] Lesser Than or Equal Compare bits

17.4 Operation

17.4.1 Position Counter

The Position Counter is 32-bits wide and counts the number of pulses generated by an encoder.

If the POSOVIEN bit in the QEIx Status register (QEIxSTAT[8]) is set and the Position Counter rolls over from 0x7FFFFFFF to 0x80000000, or from 0x80000000 to 0x7FFFFFFF, an interrupt will be generated.

The operating mode of the Position Counter is controlled by the CCM[1:0] bits in the QEIx Control register (QEIxCON[1:0]). The Position Counter supports the following operating modes:

- Quadrature Count Mode
- External Count with External Up/Down Mode
- External Count with External Gate Mode
- Internal Timer Mode

17.4.2 Quadrature Count Mode

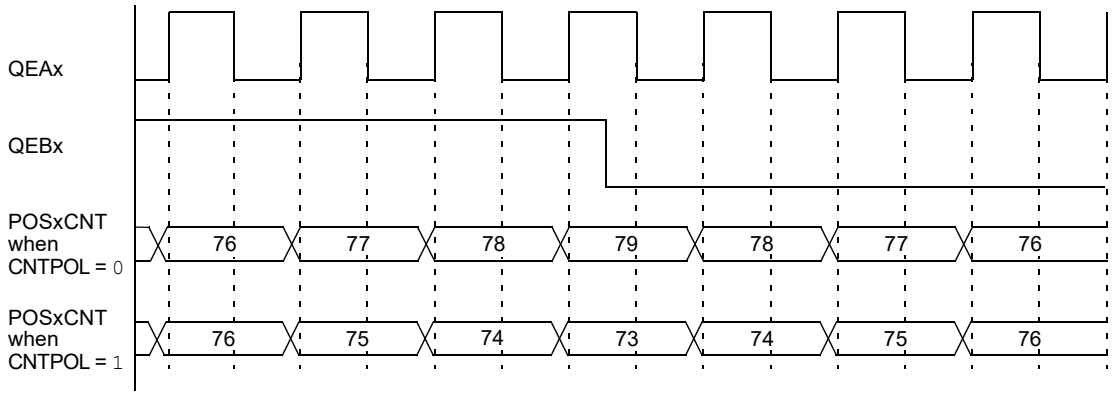
In this mode, the QEA/EXTCNT and QEB/DIR/GATE inputs are decoded to generate count pulses and direction information to control the POSxCNT and VELxCNT registers. The INDXxCNT register counts when a valid edge is detected on the INDXx input. Figure 17-1 illustrates the timing diagram of the Quadrature Count mode operation.

17.4.3 External Count with External Up/Down Mode

In this mode, the QEAx/EXTCNT input is considered as an external count signal and the QEBx/DIR/GATE input provides the count direction information. The count direction is positive unless

overridden by the CNTPOL bit in the QEIx Control register (QEIxCON[3]). Figure 17-3 illustrates the timing diagram of an external count with External Up/Down mode operation.

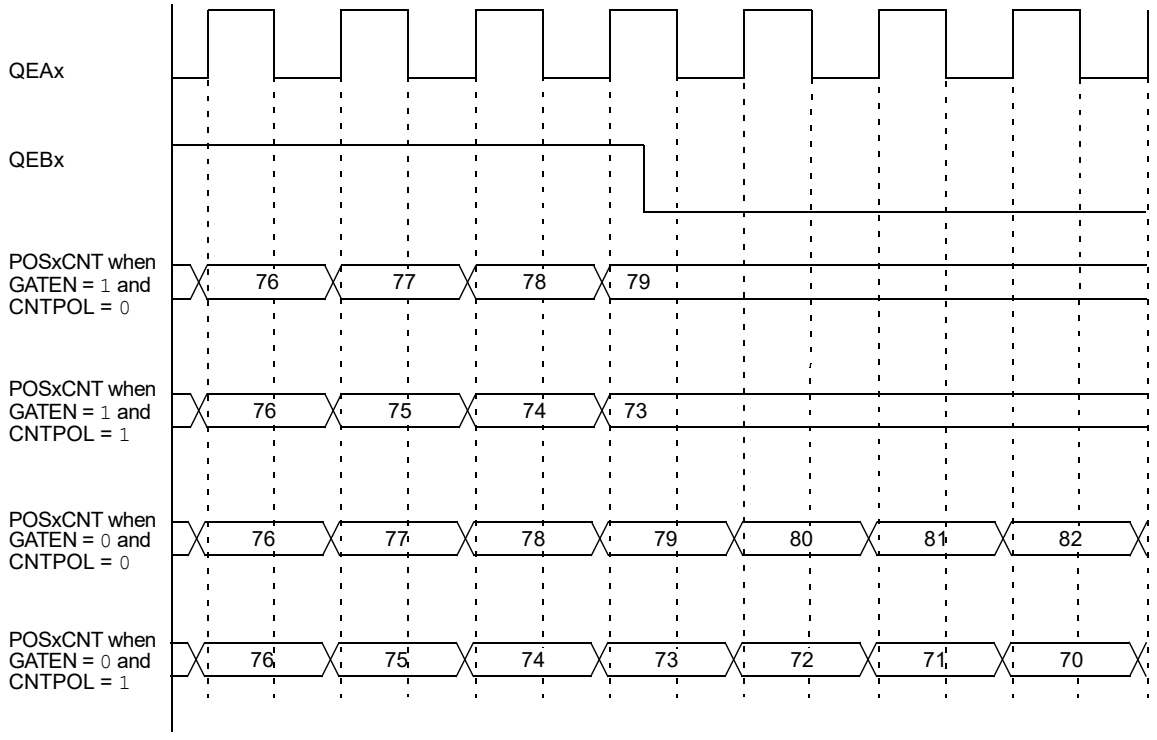
Figure 17-3. External Count with External Up/Down Mode



17.4.4 External Count with External Gate Mode

In this mode, the QEAx/EXTCNT input is considered as an external count signal. If the GATEN bit in the QEIx Control register (QEIxCON[2]) is set, and QEBx/DIR/GATE = 0, the QEBx/DIR/GATE input will inhibit the counter signal. If the GATEN bit is cleared, the gate signal does not affect the counter operation. The default count direction is positive. If the CNTPOL bit in the QEIx Control register (QEIxCON[3]) is set, the count direction is negative. Figure 17-4 illustrates the timing diagram of an external count with External Gate mode operation.

Figure 17-4. External Count with External Gate Mode

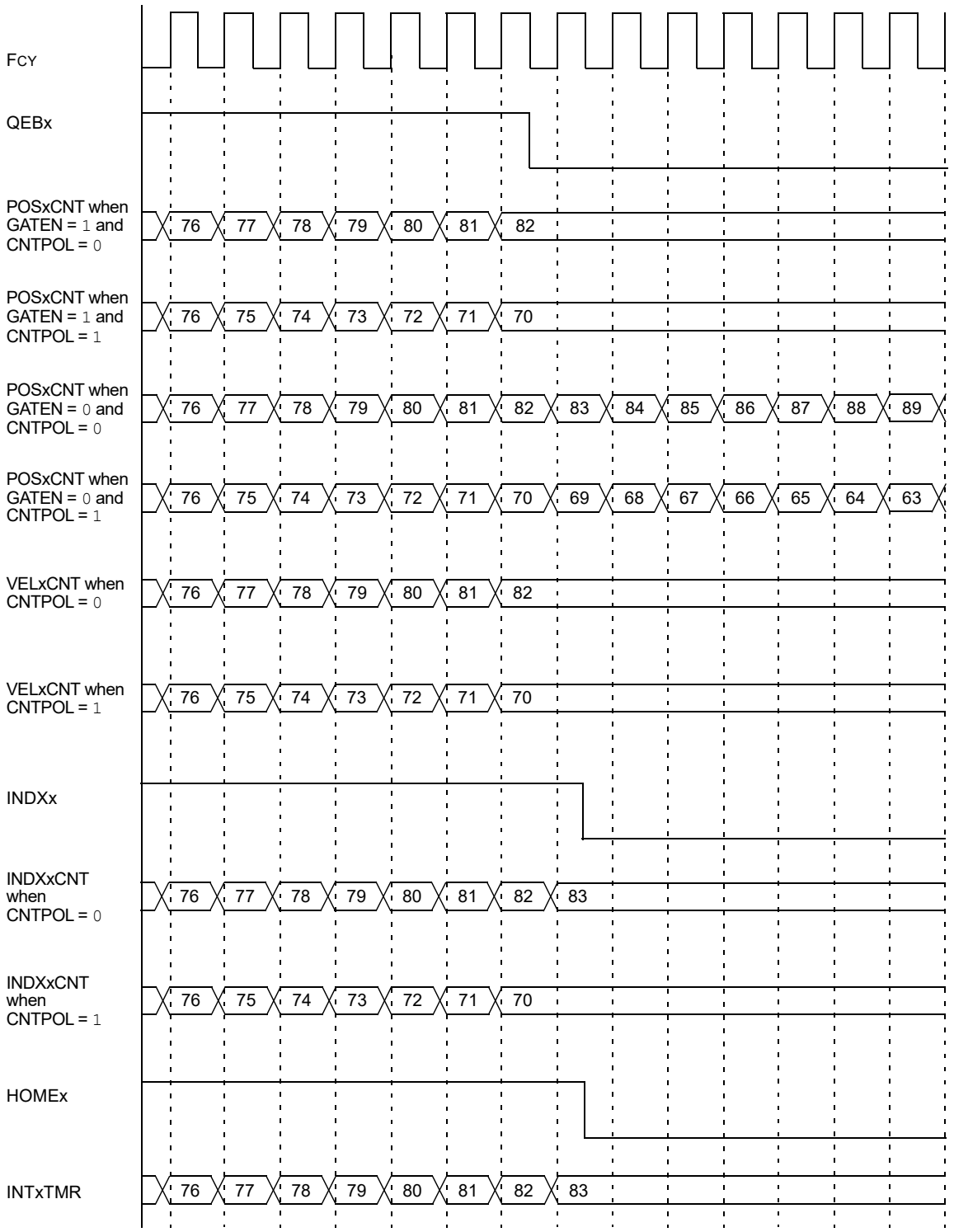


17.4.5 Internal Timer Mode

In this mode, the position counter, velocity, index and interval counters use an internal clock as the count source. The internal clock is divided by the clock divider using the INTDIV[2:0] bits in the QEIx Control register (QEIxCON[6:4]). If the GATEN bit in the QEIx Control register (QEIxCON[2]) is set and QEBx/DIR/GATE = 0, the QEBx/DIR/GATE input will inhibit the counter signal. If the GATEN bit is cleared, the gate signal does not affect the operation of the counter. The default count direction is positive. If the CNTPOL bit in the QEIx Control register (QEIxCON[3]) is set, the count direction is negative. Figure 17-5 illustrates the timing diagram of an Internal Timer mode operation.

The INDEX input enables and disables (gates) the counting of the index counter. The HOME input enables and disables (gates) the counting of the interval counter.

Figure 17-5. Internal Timer Mode



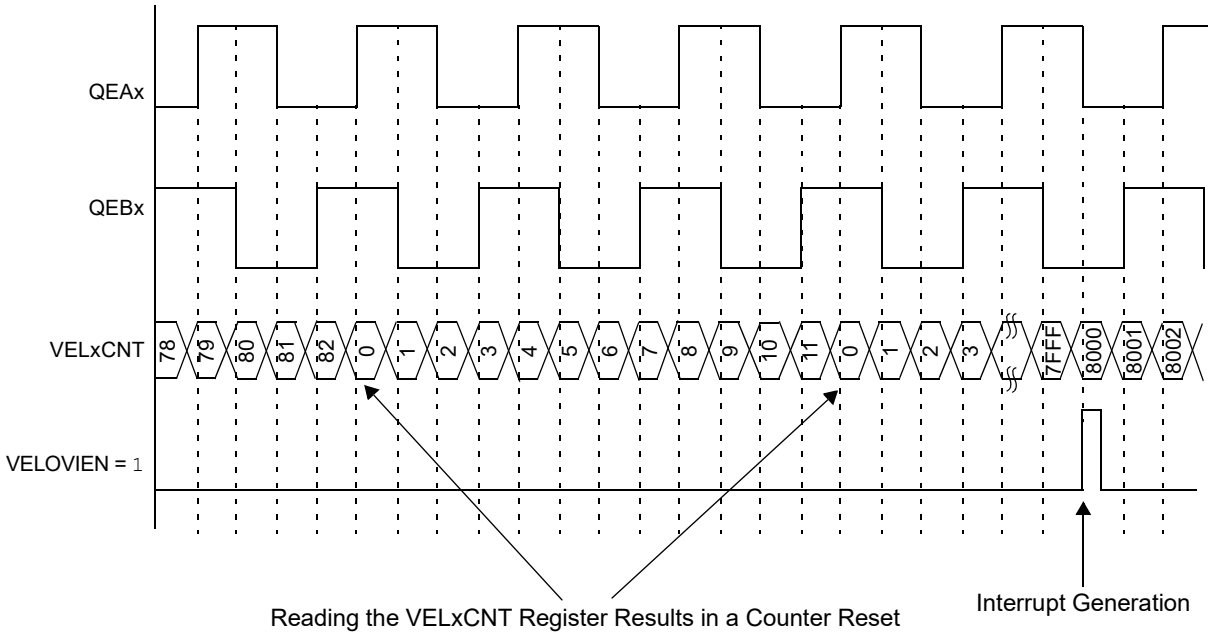
17.4.6 Velocity Counter

The Velocity Counter (VELxCNT) is a register that is up to 32-bits wide and increments or decrements based on the signal from the quadrature decoder logic. Reading this register results in a counter Reset. The Index input or any of the modes specified by the PIMOD[2:0] bits in the QEIx Control register (QEIxCON[12:10]) do not affect the operation of the Velocity Counter. If the Velocity Counter

rolls over from 0x7FFF to 0x8000, or from 0x8000 to 0x7FFF, and the VELOVIEN bit in the QEIx Status register (QEIxSTAT[4]) is set, an interrupt will be generated. Figure 17-6 illustrates the timing diagram of the Velocity Counter operation.

Note: The Velocity Counter specifies the distance traveled between the time interval of each sample. Reading the VELxCNT register results in a counter Reset. The user application should read the Velocity Counter at a rate of 1-4 kHz.

Figure 17-6. Velocity Counter



17.4.7 Index Counter

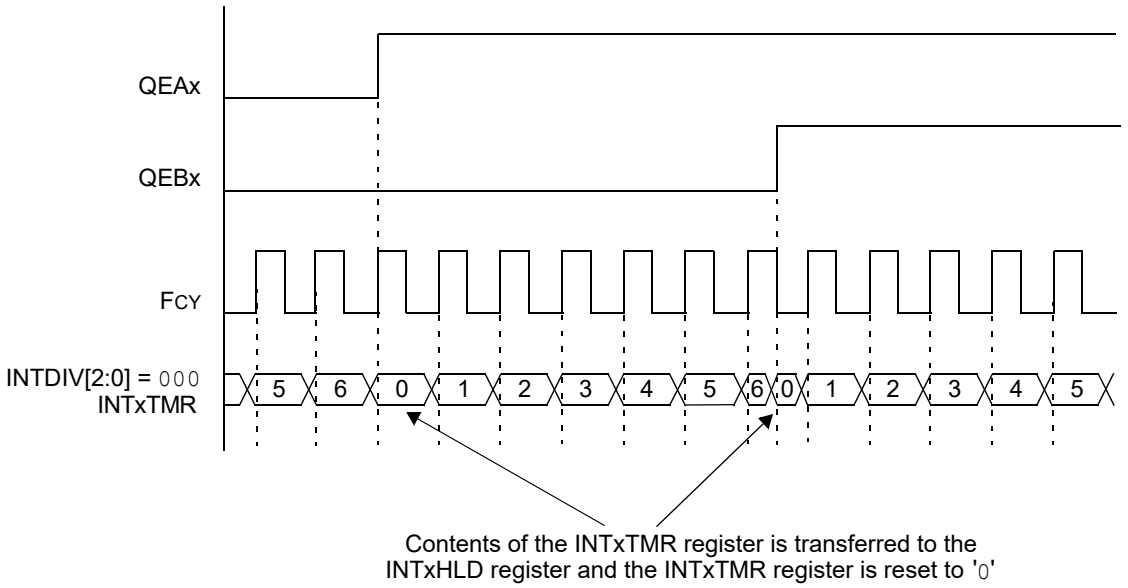
The 32-bit wide Index Counter (INDXxCNT) register counts index events and is incremented or decremented based on the direction output of the quadrature logic decoder (see Figure 17-2). For more information, refer to 17.4.11. Index Event.

17.4.8 Interval Timer

When a motor runs at a very low speed, the encoder does not generate enough pulses for accurate speed measurement. Therefore, instead of counting the number of pulses, the pulse duration can be measured. The 32-bit Interval Timer (INTxTMR) is used to measure the time interval between each decoded quadrature count pulse when the motor operates at a very low speed. The timer counts at a rate specified by the INTDIV[2:0] bits in the QEIx Control register (QEIxCON[6:4]). The Interval Timer is cleared when the first count pulse is detected. When the next count pulse is detected, the current contents of the Interval Timer are transferred to the Interval Hold register (INTxHLD), the Interval Timer is cleared and then the process repeats. The Interval Hold registers always contain the most recent completed timing measurements. The Interval Timer is automatically cleared when the module gets disabled. Figure 17-7 illustrates the timing diagram of the Interval Timer operation.

Note: If the INTxHLD register is read when a new position count pulse is detected, the contents of the INTxHLD register are not updated to avoid incoherent data reading.

Figure 17-7. Interval Timer



17.4.9 Initialization/Capture Register

The 32-bit QEIx Initialization/Capture register (QEIxIC) is a general purpose register that can be used to perform the following functions:

- Initialize the Position Counter
- Capture the contents of the Position Counter

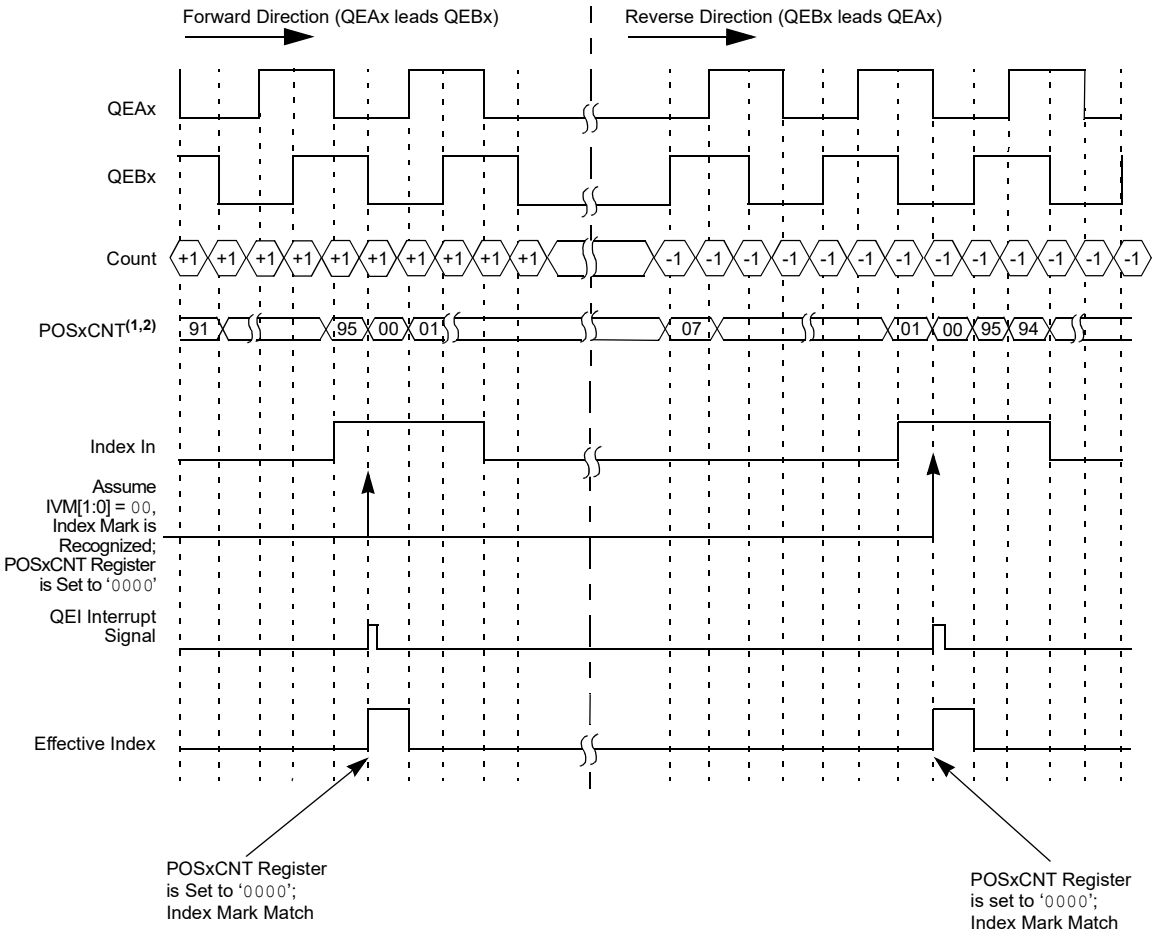
The QEIxIC register can perform only one of these tasks at a time, but the mode of operation can be changed during the operation. The selection is done by the PIMOD[2:0] bits of the QEIx Control register (QEIxCON[12:10]). To initialize the Position Counter mode, the contents of the QEIxIC register are loaded into the POSxCNT register based on the condition set by the PIMOD[2:0] bits.

In Capture mode, the input signal is used to capture the contents of the position register into the QEIxIC register. When used for position capture, an index match event (QCAPEN = 1) or a home event (HCAPEN = 1) can cause the QEIxIC register to take a copy of the current Position Counter contents.

17.4.10 Position Comparator

The 32-bit Compare registers (QEIxGEC and QEIxLEC) and associated comparator allow the user application to compare the contents of the Position Counter to a specified value. The comparator provides two outputs: greater than and less than. When a suitable condition is met, the comparator generates and sets the PCHEQIRQ or PCLEQIRQ bit in the QEIx Status register (QEIxSTAT[13] and QEIxSTAT[11], respectively). If the interrupt enable bit, PCHEQIEN or PCLEQIEN, is set, an interrupt is generated. The comparator output is available on the CNTCMPx pin. The selection of a condition is made by the OUTFNC[1:0] bits of the QEIx I/O Control register (QEIxIOC[10:9]). The comparator can also be used to reset the Position Counter when a match is detected. The selection is made by the PIMOD[2:0] bits of the QEIx Control register (QEIxCON[12:10]). Figure 17-8 illustrates the Index Reset Position Counter operation.

Figure 17-8. Index Reset Position Counter Operation



Notes:

1. Position count update shown is when CCM[1:0] = 00.
2. Position Counter (POSxCNT) contents are incremented and decremented at each new count state although it is not shown in this diagram.

17.4.11 Index Event

The IMV[1:0] bits in the QEI Control register (QEIxCON[9:8]) specify the state of the QEAx and QEBx input signals required to Acknowledge an index event. An index event is accepted when an index pulse occurs while the value of the QEAx and QEBx inputs match the condition set in the IMV[1:0] bits. This prevents further index events from being accepted until the index input signal is deasserted, and ensures that only one index event occurs for each index input pulse. Figure 17-8 illustrates the Index Reset Position Counter operation.

17.4.12 Position Counter Initialization Modes

By using the PIMOD[2:0] bits in the QEIx Control register (QEIxCON[12:10]), the user application can specify how the Position Counter is initialized during the module operation.

- **Mode 0** – The Position Counter is unaffected by the index input.
- **Mode 1** – The Position Counter is cleared whenever an index input event is detected.
- **Mode 2** – The Position Counter is initialized with the contents of the QEIxIC register on the next detected index input event. When the index event occurs, the PIMOD[2:0] bits are cleared and then the counter operates in Mode 0.

- **Mode 3** – The Position Counter is initialized with the contents of the QEIxIC register on the next detected index input event following the assertion of the home input. When an index event occurs following the home event, the PIMOD[2:0] bits are cleared and then the counter operates in Mode 0.
- **Mode 4** – The Position Counter is initialized with the contents of the QEIxIC register on the second detected index input event following the assertion of the home input. When the second index event occurs following the home event, the PIMOD[2:0] bits are cleared and then the counter operates in Mode 0.
- **Mode 5** – The Position Counter is cleared when the Position Counter value equals the QEIxGEC register value.
- **Mode 6** – The Position Counter is loaded with the contents of the QEIxLEC register when the Position Counter value equals the QEIxGEC register value and a count up pulse is detected. The counter is loaded with the contents of the QEIxGEC register when the Position Counter value equals the QEIxLEC register value and a count down pulse is detected.
- **Mode 7** – Modulo Count mode with index pulse clearing. The Position Counter is loaded with the contents of the QEIxLEC register when the Position Counter equals the QEIxGEC register's contents and a count up pulse is detected. The Position Counter is loaded with the contents of the QEIxGEC register when the Position Counter equals the QEIxLEC register contents and a count down pulse is detected. If an index pulse is detected, the Position Counter is cleared whenever an index input event is detected.

17.4.13 Digital Input Filter

The QEI module uses digital noise filters to reject noise on the incoming index and quadrature phase signals. These filters reject low-level noise and large, short duration noise spikes that typically occur in motor systems.

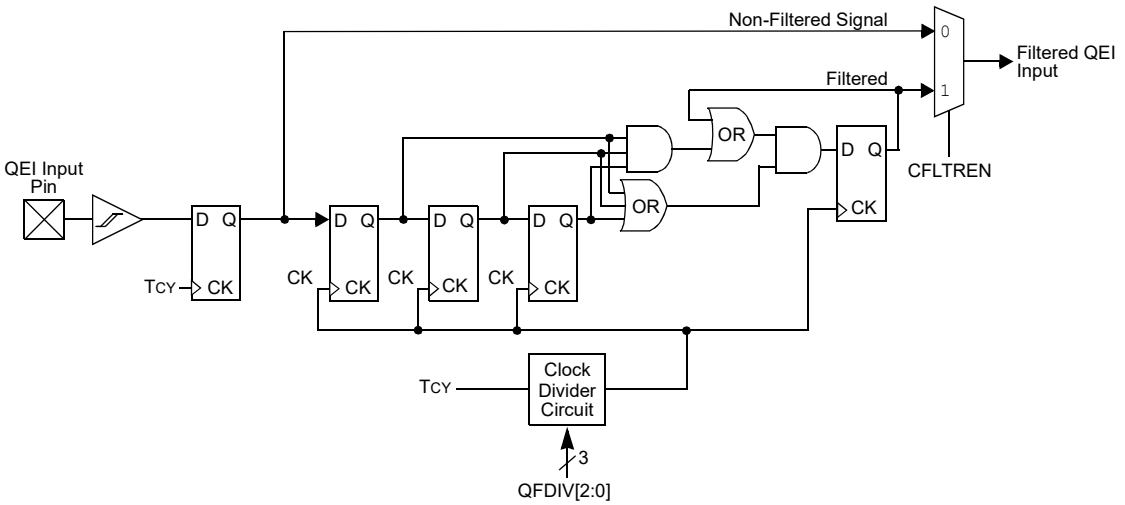
The filtered output signals can change only after an input level has the same value for three consecutive rising clock edges. The result is that short noise spikes between rising clock edges are ignored, and pulses shorter than two clock periods are rejected.

The filter clock's rate determines the low passband of the filter. A slower filter clock results in a passband rejecting lower frequencies.

The digital filter is enabled by setting the FLTREN bit in the QEIx I/O Control register (QEIxIOC[14]). The QFDIV[2:0] bits in the QEIx I/O Control register (QEIxIOC[13:11]) select the filter clock divider ratio for the clock signal.

Figure 17-9 illustrates the simplified block diagram of the digital noise filter.

Figure 17-9. Digital Noise Filter Block Diagram



17.5 Application Example

Often at system power-up, a machine or motor must be “homed” to a known position to establish a reference point. In this case, the motor and/or machine is moved in a specific direction at slow speed until a home sensor is detected. The home switch position is not precise. Therefore, the system will use an index pulse to determine the precise point of “home.”

Once the motor is homed, the position counter is required to be loaded with a known value.

In this application scenario, the position counter could be initialized to Mode 3 (PIMOD[2:0] = 3), and the QEIxIC register could be loaded with the initialization value. Optionally, an interrupt could be generated at the end of this position counter (homing) initialization process by setting the PCIIEN bit in the QEIxSTAT register.

```
int main()
{
/*Remap of QEA, QEB, Index pins goes here*/

/*Clear position counter*/
POS1CNT = 0;

/*Enable QEI interrupt*/
_QEI1IE = 1;

/*Enable QEI interrupt on initializing the position counter on completion of
homing process*/
QEI1STATbits.PCIIEN = 1;

/*Initialize QEIxIC register*/
QEI1IC = 0x50;

QEI1CONbits.CCM = 0;      // Position counter in quadrature mode
QEI1CONbits.PIMOD = 3;    // Initialize the position counter on completion of
homing process
QEI1CONbits.QEIEN = 1;    // Enable QEI module
}

void __attribute__((interrupt, no_auto_psv)) _QEI1Interrupt(void)
{
     _QEI1IF = 0;

    // Check if the source of interrupt is homing initialization
    if(QEI1STATbits.PCIIRQ)
    {
         // User application code goes here
```

```
        }
    }
```

17.6 Interrupts

The following are the sources of QEI interrupts:

- Position Counter Overflow or Underflow Event (POSOVIRQ)
- Velocity Counter Overflow or Underflow Event (VELOVIRQ)
- Position Counter Initialization Process Complete (PCIIRQ)
- Position Counter Greater Than or Equal Compare Interrupt (PCEHQIRQ)
- Position Counter Less Than or Equal Compare Interrupt (PCLEQIRQ)
- Index Event Interrupt (IDXIRQ) (When not in Internal Timer mode CCM = 0b11)
- Home Event Interrupt (HOMIRQ) (When not in Internal Timer mode CCM = 0b11)

The QEIx Status register (QEIxSTAT) contains the individual interrupt enable bits and the corresponding interrupt status bits for each interrupt source. A status bit indicates that an interrupt request has occurred. The module reduces all of the QEI interrupts to a single interrupt signal to the interrupt controller module.

17.7 QEI Operation in Power-Saving Modes

17.7.1 Sleep Mode

When the device enters Sleep mode, QEI operations cease. The POSxCNT register stops at the current value. The QEI does not respond to active signals on the QEAx, QEBx or INDXx pins. The QEIxCON register remains unchanged.

17.7.2 Idle Mode

When the device enters Idle mode, the QEISIDL bit in the QEIx Control register (QEIxCON[13]) determines whether the QEI module stops in Idle mode or continues to operate in Idle mode.

If QEISIDL = 1, the QEI module enters into a power-saving mode and performs the same functions as in Sleep mode. If QEISIDL = 0, the module does not enter into a power-saving mode and continues operation in Idle mode.

18. Universal Asynchronous Receiver Transmitter (UART)

The Universal Asynchronous Receiver Transmitter (UART) is a flexible serial communication peripheral used to interface dsPIC® microcontrollers with other equipment, including computers and peripherals. The UART is a full-duplex, asynchronous communication channel that can be used to implement protocols, such as RS-232 and RS-485.

The UART also supports the following hardware extensions:

- LIN 2.2/J2602
- IrDA®
- Digital Multiplex 512 (DMX)
- Smart Card (ISO 7816)

The primary features of the UART are:

- Full or Half-Duplex Operation
- Up to 8-Deep TX and RX First-In First-Out (FIFO) Buffers
- 8-Bit or 9-Bit Data Width
- Configurable Stop Bit Length
- Flow Control
- Auto-Baud Calibration
- Parity, Framing and Buffer Overrun Error Detection
- Address Detect
- Break Transmission
- Transmit and Receive Polarity Control
- Operation in Sleep mode
- Wake from Sleep on Sync Break Received Interrupt

18.1 Device-Specific Information

Table 18-1. UART Summary

UART Module Instances	PPS Availability	Peripheral Bus Speed	Input Clock Speed	Clock Source
3	All Instances	Standard	See Electrical Characteristics	See Table 18-2

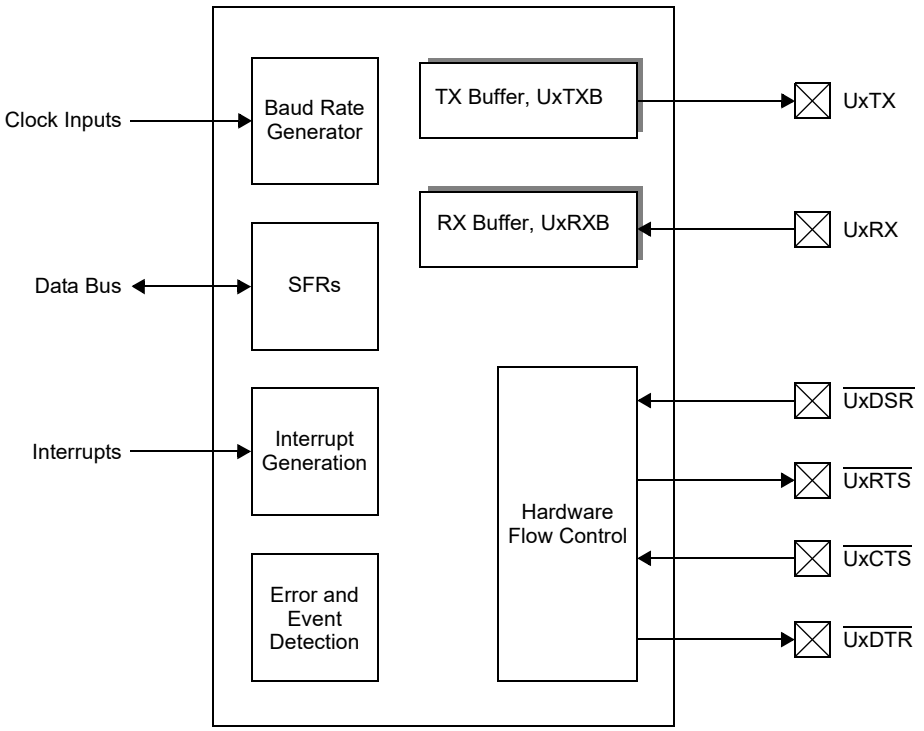
Table 18-2. Baud Clock Source Selection bits

Value	Description
1	CLKGEN8
0	Standard Speed Peripheral Clock

18.2 Architectural Overview

The UART transfers bytes of data to and from device pins using First-In First-Out (FIFO) buffers up to eight bytes deep. The status of the buffers and data is made available to user software through Special Function Registers (SFRs). The UART implements multiple interrupt channels for handling transmit, receive and error events. A simplified block diagram of the UART is shown in Figure 18-1.

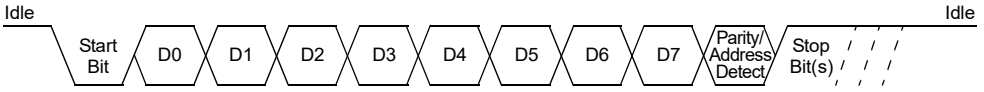
Figure 18-1. Simplified UARTx Block Diagram



18.2.1 Character Frame

A typical UART character frame is shown in Figure 18-2. The Idle state is high with a 'Start' condition indicated by a falling edge. The Start bit is followed by a number of data, parity/address detect and Stop bits defined by the MODE[3:0] (UxCON[3:0]) bits selected.

Figure 18-2. UART Character Frame



18.2.2 Data Buffers

Both transmit and receive functions use buffers to store data shifted to/from the pins. These buffers are FIFOs and are accessed by reading the SFRs, UxTXB and UxRXB, respectively. Each data buffer has multiple flags associated with its operation to allow the software to read the status. Interrupts can also be configured based on the space available in the buffers. The transmit and receive buffers can be cleared and their pointers reset using the associated TX/RX Buffer Empty Status bits, TXBE and RXBE (U1STAT).

18.2.3 Protocol Extensions

The UART provides hardware support for LIN/J2602, DMX and smart card protocol extensions to reduce software overhead. A protocol extension is enabled by writing a value to the MODE[3:0] (UxCON[3:0]) selection bits and further configured using the UARTx Timing Parameter registers, 18.3.6. UxPA and 18.3.7. UxPB. Details regarding operation and usage are discussed in their respective chapters. Not all protocols are available on all devices.

18.3 UART Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1700	U1CON	31:24	SLPEN	ACTIVE			CLKMOD	CLKSEL[1:0]		HALFDPLX
		23:16	RUNOVF	RXPOL	STP[1:0]		C0EN	TXPOL	FLO[1:0]	
		15:8	ON		SIDL	WUE	RXBIMD		BRKOVR	SENDB
		7:0	BRGS	ABDEN	TXEN	RXEN	MODE[3:0]			
0x1704	U1STAT	31:24		TXWM[2:0]				RXWM[2:0]		
		23:16	TXWRE	STPMD	TXBE	TXBF	RCIDL	XON	RXBE	RXBF
		15:8	TXMTIE	PERIE	ABDOVIE	CERIE	FERIE	RXBKIE	RXFOIE	TXCIE
		7:0	TXMTIF	PERIF	ABDOVF	CERIF	FERIF	RXBKIF	RXFOIF	TXCIF
0x1708	U1BRG	31:24								
		23:16					BRG[19:16]			
		15:8	BRG[15:8]							
		7:0	BRG[7:0]							
0x170C	U1RXB	31:24								
		23:16								
		15:8								RXB[8]
		7:0	RXB[7:0]							
0x1710	U1TXB	31:24								
		23:16								
		15:8	LAST							
		7:0	TXB[7:0]							
0x1714	U1PA	31:24	WIP							P2[8]
		23:16	P2[7:0]							
		15:8								P1[8]
		7:0	P1[7:0]							
0x1718	U1PB	31:24	WIP							
		23:16	P3[23:16]							
		15:8	P3[15:8]							
		7:0	P3[7:0]							
0x171C	U1CHK	31:24								
		23:16	RXCHK[7:0]							
		15:8								
		7:0	TXCHK[7:0]							
0x1720	U1SCCON	31:24			RXRPTIF	TXRPTIF		BTCIF	WTCIF	GTCIF
		23:16			RXRPTIE	TXRPTIE		BTCIE	WTCIE	GTCIE
		15:8								
		7:0			TXRPT[1:0]		CONV	T0PD	PRTCL	
0x1724	U1UIR	31:24								
		23:16								
		15:8								
		7:0	WUIF	ABDIF				ABDIE		
0x1728 ... 0x173F	Reserved									
0x1740	U2CON	31:24	SLPEN	ACTIVE			CLKMOD	CLKSEL[1:0]		HALFDPLX
		23:16	RUNOVF	RXPOL	STP[1:0]		C0EN	TXPOL	FLO[1:0]	
		15:8	ON		SIDL	WUE	RXBIMD		BRKOVR	SENDB
		7:0	BRGS	ABDEN	TXEN	RXEN	MODE[3:0]			
0x1744	U2STAT	31:24		TXWM[2:0]				RXWM[2:0]		
		23:16	TXWRE	STPMD	TXBE	TXBF	RCIDL	XON	RXBE	RXBF
		15:8	TXMTIE	PERIE	ABDOVIE	CERIE	FERIE	RXBKIE	RXFOIE	TXCIE
		7:0	TXMTIF	PERIF	ABDOVF	CERIF	FERIF	RXBKIF	RXFOIF	TXCIF
0x1748	U2BRG	31:24								
		23:16					BRG[19:16]			
		15:8	BRG[15:8]							
		7:0	BRG[7:0]							
0x174C	U2RXB	31:24								
		23:16								
		15:8								RXB[8]
		7:0	RXB[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1750	U2TXB	31:24								
		23:16								
		15:8	LAST							
		7:0	TXB[7:0]							
0x1754	U2PA	31:24	WIP							P2[8]
		23:16	P2[7:0]							
		15:8								P1[8]
		7:0	P1[7:0]							
0x1758	U2PB	31:24	WIP							
		23:16	P3[23:16]							
		15:8	P3[15:8]							
		7:0	P3[7:0]							
0x175C	U2CHK	31:24								
		23:16	RXCHK[7:0]							
		15:8								
		7:0	TXCHK[7:0]							
0x1760	U2SCCON	31:24			RXRPTIF	TXRPTIF		BTCIF	WTCIF	GTCIF
		23:16			RXRPTIE	TXRPTIE		BTCIE	WTCIE	GTCIE
		15:8								
		7:0			TXRPT[1:0]		CONV	T0PD	PRTCL	
0x1764	U2UIR	31:24								
		23:16								
		15:8								
		7:0	WUIF	ABDIF				ABDIE		
0x1768 ... 0x177F	Reserved									
0x1780	U3CON	31:24	SLPEN	ACTIVE			CLKMOD	CLKSEL[1:0]		HALFDPLX
		23:16	RUNOVF	RXPOL	STP[1:0]		C0EN	TXPOL	FLO[1:0]	
		15:8	ON		SIDL	WUE	RXBIMD		BRKOVR	SENDB
		7:0	BRGS	ABDEN	TXEN	RXEN	MODE[3:0]			
0x1784	U3STAT	31:24		TXWM[2:0]				RXWM[2:0]		
		23:16	TXWRE	STPMD	TXBE	TXBF	RCIDL	XON	RXBE	RXBF
		15:8	TXMTIE	PERIE	ABDOVIE	CERIE	FERIE	RXBKIE	RXFOIE	TXCIE
		7:0	TXMTIF	PERIF	ABDOVF	CERIF	FERIF	RXBKIF	RXFOIF	TXCIF
0x1788	U3BRG	31:24								
		23:16					BRG[19:16]			
		15:8	BRG[15:8]							
		7:0	BRG[7:0]							
0x178C	U3RXB	31:24								
		23:16								
		15:8								RXB[8]
		7:0	RXB[7:0]							
0x1790	U3TXB	31:24								
		23:16								
		15:8	LAST							
		7:0	TXB[7:0]							
0x1794	U3PA	31:24	WIP							P2[8]
		23:16	P2[7:0]							
		15:8								P1[8]
		7:0	P1[7:0]							
0x1798	U3PB	31:24	WIP							
		23:16	P3[23:16]							
		15:8	P3[15:8]							
		7:0	P3[7:0]							
0x179C	U3CHK	31:24								
		23:16	RXCHK[7:0]							
		15:8								
		7:0	TXCHK[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x17A0	U3SCCON	31:24			RXRPTIF	TXRPTIF		BTCIF	WTCIF	GTCIF
		23:16			RXRPTIE	TXRPTIE		BTCIE	WTCIE	GTCIE
		15:8								
		7:0			TXRPT[1:0]		CONV	T0PD	PRTCL	
0x17A4	U3UIR	31:24								
		23:16								
		15:8								
		7:0	WUIF	ABDIF				ABDIE		

18.3.1 UARTx Configuration Register

Name: UxCON
Offset: 0x1700, 0x1740, 0x1780

Note:
1. R/HS/HC in DMX and LIN mode.

Bit	31	30	29	28	27	26	25	24
	SLPEN	ACTIVE			CLKMOD	CLKSEL[1:0]		HALFDPLX
Access	R/W	R			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RUNOVF	RXPOL	STP[1:0]		C0EN	TXPOL	FLO[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	WUE	RXBIMD		BRKOVR	SENDB
Access	R/W		R/W	R/W	R/W		R/W	R/W/HC
Reset	0		0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	BRGS	ABDEN	TXEN	RXEN	MODE[3:0]			
Access	R/W	R/W/HC	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – SLPEN Run During Sleep Enable bit

Value	Description
1	UART BRG clock runs during Sleep
0	UART BRG clock is turned off during Sleep

Bit 30 – ACTIVE UART Running Status bit

Value	Description
1	UART clock request is active (user should not update the UxCON register)
0	UART clock request is not active (user can update the UxCON register)

Bit 27 – CLKMOD Baud Clock Generation Mode Select bit

Value	Description
1	Uses fractional Baud Rate Generation
0	Uses legacy divide-by-x counter for baud clock generation (x = 4 or 16 depending on the BRGS bit)

Bits 26:25 – CLKSEL[1:0] Baud Clock Source Selection bits
See Table 18-2.

Bit 24 – HALFDPLX UART Half-Duplex Selection Mode bit
(HALFDPLX is ignored in LIN and Smart Card modes)

Value	Description
1	Half-Duplex mode
0	Full-Duplex mode

Bit 23 – RUNOVF Run During Overflow Condition Mode bit

Value	Description
1	When an Overflow Error (RXFOIF) condition is detected, the RX shifter continues to run so as to remain synchronized with incoming RX data; data are not transferred to UxRXB when it is full (i.e., no UxRXB data are overwritten)
0	When an Overflow Error (RXFOIF) condition is detected, the RX shifter stops accepting new data; data are transferred to UxRXB when one empty slot becomes available in the buffer (Legacy mode)

Bit 22 – RXPOL UART Receive Polarity bit

Value	Description
1	Inverts RX polarity; Idle state is low
0	Input is not inverted; Idle state is high

Bits 21:20 – STP[1:0] Number of Stop Bits Selection bits

Value	Description
11	2 Stop bits sent, 1 checked at receive
10	2 Stop bits sent, 2 checked at receive
01	1.5 Stop bits sent, 1.5 checked at receive
00	1 Stop bit sent, 1 checked at receive

Bit 19 – C0EN Enable Legacy Checksum (C0) Transmit and Receive bit

Value	Description
1	Checksum Mode 1 (enhanced LIN checksum in LIN mode; add all TX/RX words in all other modes)
0	Checksum Mode 0 (legacy LIN checksum in LIN mode; not used in all other modes)

Bit 18 – TXPOL UART Transmit Polarity bit

Value	Description
1	Inverts TX polarity; TX is low in Idle state
0	Output data are not inverted; TX output is high in Idle state

Bits 17:16 – FLO[1:0] Flow Control Enable bits (only valid when MODE[3:0] = 0xxx)

Value	Description
11	Reserved
10	UxRTS-UxDSR (for TX side)/UxCTS-UxDTR (for RX side) hardware flow control
01	XON/XOFF software flow control
00	Flow control off

Bit 15 – ON UART Enable bit

Value	Description
1	UART is ready to transmit and receive
0	UART state machine, FIFO Buffer Pointers and counters are reset; registers are readable and writable

Bit 13 – SIDL UART Stop in Idle Mode bit

Value	Description
1	Discontinues module operation when device enters Idle mode
0	Continues module operation in Idle mode

Bit 12 – WUE Wake-up Enable bit

Value	Description
1	Module will continue to sample the UxRX pin – interrupt generated on falling edge, bit cleared in hardware on following rising edge; if ABDEN is set, Auto-Baud Detection (ABD) will begin immediately

Value	Description
0	UxRX pin is not monitored nor rising edge detected

Bit 11 – RXBIMD Receive Break Interrupt Mode bit

Value	Description
1	RXBKIF flag when a minimum of 23 (DMX)/11 (asynchronous or LIN/J2602) low bit periods are detected
0	RXBKIF flag when the Break makes a low-to-high transition after being low for at least 23/11 bit periods

Bit 9 – BRKOVR Send Break Software Override bit
Overrides the TX Data Line:

Value	Description
1	Makes the TX line active (Output 0 when TXPOL = 0, Output 1 when TXPOL = 1)
0	TX line is driven by the shifter

Bit 8 – SENDB UART Transmit Break bit(1)

Value	Description
1	Sends Sync Break on next transmission; cleared by hardware upon completion
0	Sync Break transmission is disabled or has completed

Bit 7 – BRGS High Baud Rate Select bit

Value	Description
1	High speed
0	Low speed

Bit 6 – ABDEN Auto-Baud Detect Enable bit (read-only when MODE[3:0] = 1xxx)

Value	Description
1	Enables baud rate measurement on the next character – requires reception of a Sync field (55h); cleared in hardware upon completion
0	Baud rate measurement is disabled or has completed

Bit 5 – TXEN UART Transmit Enable bit

Value	Description
1	Transmit enabled – except during Auto-Baud Detection
0	Transmit disabled – all transmit counters, pointers and state machines are reset; TX buffer is not flushed, status bits are not reset

Bit 4 – RXEN UART Receive Enable bit

Value	Description
1	Receive enabled – except during Auto-Baud Detection
0	Receive disabled – all receive counters, pointers and state machines are reset; RX buffer is not flushed, status bits are not reset

Bits 3:0 – MODE[3:0] UART Mode bits

Value	Description
Other	Reserved
1111	Smart card
1110	IrDA®
1101	Reserved
1100	LIN Commander/Responder
1011	LIN Responder only
1010	DMX

Value	Description
1001-0101	Reserved
0100	Asynchronous 9-bit UART with address detect, ninth bit = 1 signals address
0011	Asynchronous 8-bit UART without address detect, ninth bit is used as an even parity bit
0010	Asynchronous 8-bit UART without address detect, ninth bit is used as an odd parity bit
0001	Asynchronous 7-bit UART
0000	Asynchronous 8-bit UART

18.3.2 UARTx Status Register

Name: UxSTAT
Offset: 0x1704, 0x1744, 0x1784

Note:

1. The receive watermark interrupt is not set if PERIF, FERIF or TXCIF is set and the corresponding IE bit is set.

Legend: S = Settable bit, HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
		TXWM[2:0]				RXWM[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit	23	22	21	20	19	18	17	16
	TXWRE	STPMD	TXBE	TXBF	RCIDL	XON	RXBE	RXBF
Access	R/W/HS	R/W	R/S	R	R	R	R/S	R
Reset	0	0	1	0	1	1	1	0

Bit	15	14	13	12	11	10	9	8
	TXMTIE	PERIE	ABDOVIE	CERIE	FERIE	RXBKIE	RXFOIE	TXCIE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TXMTIF	PERIF	ABDOVF	CERIF	FERIF	RXBKIF	RXFOIF	TXCIF
Access	R	R	R/W/HS	R/W/HC	R	R/W/HC	R/W/HC	R/W/HC
Reset	1	0	0	0	0	0	0	0

Bits 30:28 – TXWM[2:0] UART Transmit Interrupt Select bits

Value	Description
111	Sets transmit interrupt when there is one empty slot left in the buffer
. . .	
010	Sets transmit interrupt when there are six empty slots or more in the buffer
001	Sets transmit interrupt when there are seven empty slots or more in the buffer
000	Sets transmit interrupt when there are eight empty slots in the buffer; TX buffer is empty

Bits 26:24 – RXWM[2:0] UART Receive Interrupt Select bits(1)

Value	Description
111	Triggers receive interrupt when there are eight words in the buffer; RX buffer is full
. . .	
001	Triggers receive interrupt when there are two words or more in the buffer
000	Triggers receive interrupt when there is one word or more in the buffer

Bit 23 – TXWRE TX Write Transmit Error Status bit

LIN and Parity Modes:
1 = A new byte was written when buffer was full or when P2[8:0] = 0 (must be cleared by software)
0 = No error
Address Detect Mode:
1 = A new byte was written when buffer was full or to P1[8:0] when P1x was full (must be cleared by software)

0 = No error
Other Modes:
1 = A new byte was written when buffer was full (must be cleared by software)
0 = No error

Bit 22 – STPMD Stop Bit Detection Mode bit

Value	Description
1	Triggers RXIF at the end of the last Stop bit
0	Triggers RXIF in the middle of the first (or second, depending on the STP[1:0] setting) Stop bit

Bit 21 – TXBE UART TX Buffer Empty Status bit

Value	Description
1	Transmit buffer is empty; writing '1' when TXEN = 0 will reset the TX FIFO pointers and counters
0	Transmit buffer is not empty

Bit 20 – TXBF UART TX Buffer Full Status bit

Value	Description
1	Transmit buffer is full
0	Transmit buffer is not full

Bit 19 – RCIDL Receive Idle bit

Value	Description
1	UART RX line is in the Idle state
0	UART RX line is receiving something

Bit 18 – XON UART in XON Mode bit
Only valid when FLO[1:0] control bits are set to XON/XOFF mode.

Value	Description
1	UART has received XON
0	UART has not received XON or XOFF was received

Bit 17 – RXBE UART RX Buffer Empty Status bit

Value	Description
1	Receive buffer is empty; writing '1' when RXEN = 0 will reset the RX FIFO pointers and counters
0	Receive buffer is not empty

Bit 16 – RXBF UART RX Buffer Full Status bit

Value	Description
1	Receive buffer is full
0	Receive buffer is not full

Bit 15 – TXMTIE Transmit Shifter Empty Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is disabled

Bit 14 – PERIE Parity Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is disabled

Bit 13 – ABDOVIE Auto-Baud Rate Acquisition Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is disabled

Bit 12 – CERIE Checksum Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is disabled

Bit 11 – FERIE Framing Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is disabled

Bit 10 – RXBKIE Receive Break Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is disabled

Bit 9 – RXFOIE Receive Buffer Overflow Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is disabled

Bit 8 – TXCIE Transmit Collision Interrupt Enable bit

Value	Description
1	Interrupt is enabled
0	Interrupt is disabled

Bit 7 – TXMTIF Transmit Shifter Empty Interrupt Flag bit

Value	Description
1	Transmit Shift Register (TSR) is empty (TXMTIF bit gets set at the end of the last Stop bit; TXMTIF bit behavior is independent of the STPMD bit)
0	Transmit Shift Register is not empty

Bit 6 – PERIF Parity Error/Address Received

LIN and Parity Modes:
1 = Parity error detected
0 = No parity error detected
Address Mode:
1 = Address received
0 = No address detected
All Other Modes:
Not used.

Bit 5 – ABDOVF Auto-Baud Rate Acquisition Interrupt Flag bit

Value	Description
1	BRG rolled over during the auto-baud rate acquisition sequence
0	BRG has not rolled over during the auto-baud rate acquisition sequence

Bit 4 – CERIF Checksum Error Interrupt Flag bit (must be cleared by software)

Value	Description
1	Checksum error
0	No checksum error

Bit 3 – FERIF Framing Error Interrupt Flag bit

Value	Description
1	Framing Error: Inverted level of the Stop bit corresponding to the topmost character in the buffer; propagates through the buffer with the received character
0	No framing error

Bit 2 – RXBKIF Receive Break Interrupt Flag bit (must be cleared by software)

Value	Description
1	A Break was received
0	No Break was detected

Bit 1 – RXFOIF Receive Buffer Overflow Interrupt Flag bit (must be cleared by software)

Value	Description
1	Receive buffer has overflowed
0	Receive buffer has not overflowed

Bit 0 – TXCIF Transmit Collision Interrupt Flag bit (must be cleared by software)

Value	Description
1	Transmitted word is not equal to the received word
0	Transmitted word is equal to the received word

18.3.3 UARTx Baud Rate Register

Name: UxBRG
Offset: 0x1708, 0x1748, 0x1788

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					BRG[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BRG[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BRG[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 19:0 – BRG[19:0] Baud Rate Divisor bits

18.3.4 UARTx Receive Buffer Register

Name: UxRXB
Offset: 0x170C, 0x174C, 0x178C

Note:
1. The RXB[8] bit is used only in Address Detect mode.

Legend: x = Bit is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								R
Reset								0
								RXB[8]

Bit	7	6	5	4	3	2	1	0
	RXB[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 8:0 – RXB[8:0] Received Character Data bits 8-0[(1)]

18.3.5 UARTx Transmit Buffer Register

Name: UxTXB
Offset: 0x1710, 0x1750, 0x1790

Legend: x = Bit is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LAST							
Access	W							
Reset	x							

Bit	7	6	5	4	3	2	1	0
	TXB[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	x	x	x	x	x	x	x	x

Bit 15 – LAST Last Byte Indicator for Smart Card Support bit

Bits 7:0 – TXB[7:0] Transmitted Character Data bits 7-0
If the buffer is full, further writes to the buffer are ignored.

18.3.6 UARTx Timing Parameter A Register

Name: UxPA
Offset: 0x1714, 0x1754, 0x1794

Bit	31	30	29	28	27	26	25	24
	WIP							P2[8]
Access	R/W							R/W
Reset	0							0

Bit	23	22	21	20	19	18	17	16
	P2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
								P1[8]
Access								R/W
Reset								0

Bit	7	6	5	4	3	2	1	0
	P1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – WIP UxPA Write in Progress bit

Value	Description
1	Write still in progress (user should not update the UxPA register)
0	No write in progress (user can update the UxPA register)

Bits 24:16 – P2[8:0] Parameter 2 bits

DMX RX:
The First Byte Number to Receive – 1, not including start code (bits[8:0]).
LIN Responder TX:
Number of bytes to transmit (bits[7:0]).
Asynchronous RX with Address Detect:
ADDR to match (bits[7:0]).
Smart Card Mode:
Block Time Counter (BTC) bits. This counter is operated on the bit clock whose period is always equal to one ETU (bits[8:0]).
Other Modes:
Not used

Bits 8:0 – P1[8:0] Parameter 1 bits

DMX TX:
Number of bytes to transmit – 1 (not including Start code).
LIN Commander TX:
PID to transmit (bits[5:0]).
Asynchronous TX with Address Detect:
Address to transmit. A '1' is automatically inserted into bit 8 (bits[7:0]).
Smart Card Mode:
Guard Time Counter bits. This counter is operated on the bit clock whose period is always equal to one ETU (bits[8:0]).

Other Modes:
Not used.

18.3.7 UARTx Timing Parameter B Register

Name: UxPB
Offset: 0x1718, 0x1758, 0x1798

Bit	31	30	29	28	27	26	25	24
	WIP							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
	P3[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	P3[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	P3[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – WIP UxPB Write in Progress bit

Value	Description
1	Write still in progress (user should not update the UxPB registers)
0	No write in progress (user can update the UxPB registers)

Bits 23:0 – P3[23:0] Parameter 3 bits

DMX RX:
The last byte number to receive – 1, not including start code (bits[8:0]).
LIN Responder RX:
Number of bytes to receive (bits[7:0]).
Asynchronous RX:
Used to mask the P2 address bits; 1 = P2 address bit is used, 0 = P2 address bit is masked off (bits[7:0]).
Smart Card Mode:
Waiting Time Counter bits (bits[23:0]).
Other Modes:
Not used.

18.3.8 UART Checksum Result Register

Name: UxCHK
Offset: 0x171C, 0x175C, 0x179C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	RXCHK[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	TXCHK[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – RXCHK[7:0] Receive Checksum bits (calculated from RX words)
LIN Modes:
C0EN = 1: Sum of all received data + addition carries, including PID.
C0EN = 0: Sum of all received data + addition carries, excluding PID.
LIN Responder:
Cleared when Break is detected.
LIN Commander/Responder:
Cleared when Break is detected.
Other Modes:
C0EN = 1: Sum of every byte received + addition carries.
C0EN = 0: Value remains unchanged.

Bits 7:0 – TXCHK[7:0] Transmit Checksum bits (calculated from TX words)
LIN Modes:
C0EN = 1: Sum of all transmitted data + addition carries, including PID.
C0EN = 0: Sum of all transmitted data + addition carries, excluding PID.
LIN Responder:
Cleared when Break is detected.
LIN Commander/Responder:
Cleared when Break is detected.
Other Modes:
C0EN = 1: Sum of every byte transmitted + addition carries.
C0EN = 0: Value remains unchanged

18.3.9 UARTx Smart Card Configuration Register

Name: UxSCCON
Offset: 0x1720, 0x1760, 0x17A0

Bit	31	30	29	28	27	26	25	24
			RXRPTIF	TXRPTIF		BTCIF	WTCIF	GTCIF
Access			R/W/HS	R/W/HS		R/W/HS	R/W/HS	R/W/HS
Reset			0	0		0	0	0

Bit	23	22	21	20	19	18	17	16
			RXRPTIE	TXRPTIE		BTCIE	WTCIE	GTCIE
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			TXRPT[1:0]		CONV	T0PD	PRTCL	
Access			R/W	R/W	R/W	R/W	R/W	
Reset			0	0	0	0	0	

Bit 29 – RXRPTIF Receive Repeat Interrupt Flag bit

Value	Description
1	Parity error has persisted after the same character has been received five times (four retransmits)
0	Flag is cleared

Bit 28 – TXRPTIF Transmit Repeat Interrupt Flag bit

Value	Description
1	Line error has been detected after the last retransmit per TXRPT<1:0>
0	Flag is cleared

Bit 26 – BTCIF Block Time Counter Interrupt Flag bit

Value	Description
1	Block time counter has reached 0
0	Block time counter has not reached 0

Bit 25 – WTCIF Waiting Time Counter Interrupt Flag bit

Value	Description
1	Waiting time counter has reached 0
0	Waiting time counter has not reached 0

Bit 24 – GTCIF Guard Time Counter Interrupt Flag bit

Value	Description
1	Guard time counter has reached 0
0	Guard time counter has not reached 0

Bit 21 – RXRPTIE Receive Repeat Interrupt Enable bit

Value	Description
1	An interrupt is invoked when a parity error has persisted after the same character has been received five times (four retransmits)
0	Interrupt is disabled

Bit 20 – TXRPTIE Transmit Repeat Interrupt Enable bit

Value	Description
1	An interrupt is invoked when a line error is detected after the last retransmit per TXRPT<1:0> has been completed)
0	Interrupt is disabled

Bit 18 – BTCIE Block Time Counter Interrupt Enable bit

Value	Description
1	Block time counter interrupt is enabled
0	Block time counter interrupt is disabled

Bit 17 – WTCIE Waiting Time Counter Interrupt Enable bit

Value	Description
1	Waiting time counter interrupt is enabled
0	Waiting time counter interrupt is disabled

Bit 16 – GTCIE Guard Time Counter Interrupt Enable bit

Value	Description
1	Guard time counter interrupt is enabled
0	Guard time counter interrupt is disabled

Bits 5:4 – TXRPT[1:0] Transmit Repeat Selection bits

Value	Description
11	Retransmit the error byte four times
10	Retransmit the error byte three times
01	Retransmit the error byte twice
00	Retransmit the error byte once

Bit 3 – CONV Logic Convention Selection bit

Value	Description
1	Inverse logic convention
0	Direct logic convention

Bit 2 – T0PD Pull-Down Duration for T = 0 Error Handling bit

Value	Description
1	2 ETU
0	1 ETU

Bit 1 – PRTCL Smart Card Protocol Selection bit

Value	Description
1	T = 1
0	T = 0

18.3.10 UARTx Interrupt Register

Name: UxUIR
Offset: 0x1724, 0x1764, 0x17A4

Legend: HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	WUIF	ABDIF				ABDIE		
Access	R/W/HS	R/W/HS				R/W		
Reset	0	0				0		

Bit 7 – WUIF Wake-up Interrupt Flag bit
After the occurrence of the WAKE event, the WUIF flag can only be cleared once the wake (WUE) bit is cleared by hardware following the rising edge.

Value	Description
1	Sets when WUE = 1 and RX makes a 1-to-0 transition; triggers event interrupt (must be cleared by software)
0	WUE is not enabled or WUE is enabled, but no wake-up event has occurred

Bit 6 – ABDIF Auto-Baud Completed Interrupt Flag bit

Value	Description
1	Sets when ABD sequence makes the final 1-to-0 transition; triggers event interrupt (must be cleared by software)
0	ABDEN is not enabled or ABDEN is enabled but auto-baud has not completed

Bit 2 – ABDIE Auto-Baud Completed Interrupt Enable Flag bit

Value	Description
1	Allows ABDIF to set an event interrupt
0	ABDIF does not set an event interrupt

18.4 Operation

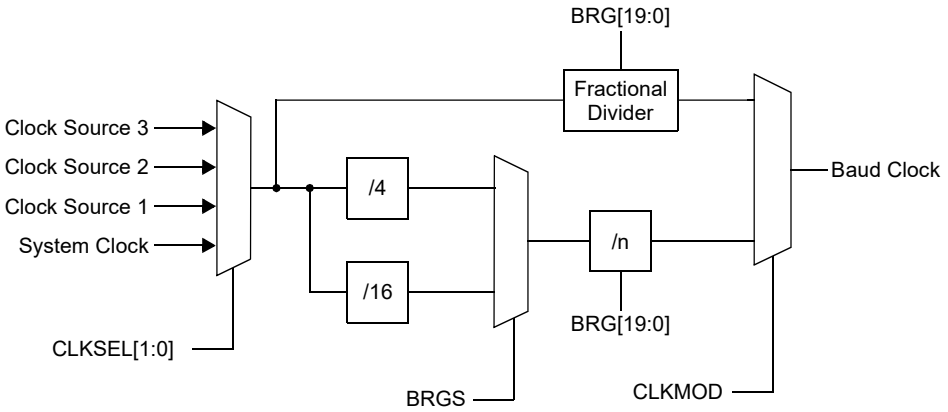
18.4.1 Clocking and Baud Rate Configuration

The UART supports multiple clock sources and two types of Baud Rate Generation (BRG). One of the clock sources provided is selected by the CLKSEL[1:0] bits (UxCON[26:25]). Clock source selection and prescaler can only be changed when the ON bit (UxCON[15]) is cleared. The baud clock can be generated with one of the following methods:

- Legacy mode, fixed division
- Fractional Division mode

To allow synchronized time of clock domains, do not make back-to-back writes to the UxBRG register. UxBRG should be written only when ON = 0 to avoid corruption of an ongoing transmission or reception. A block diagram of the UART clocking is shown in Figure 18-3.

Figure 18-3. UART Clocking Diagram



18.4.1.1 Legacy Mode

In Legacy mode, the clock source is divided down to the desired baud clock using integer division. Legacy mode is selected when CLKMOD = 0 (UxCON[27]). A selectable prescaler is present to support a wide range of baud rates and is controlled by the BRGS bit (UxCON[7]). Up to a 20-bit value of BRG (UxBRG[19:0]) is used to further divide down the input clock to the final baud rate.

Equation 18-1 and Equation 18-2 show formulas for baud rate and BRG value given by the BRGS for all protocol modes.

Equation 18-1. Baud Rate When BRG = 0

$$\text{Baud Rate} = \frac{F_P}{16 \times (\text{BRG} + 1)}$$

$$\text{BRG} = \frac{F_P}{16 \times \text{Baud Rate}} - 1$$

Note: F_P = Peripheral Clock.

Equation 18-2. Baud Rate When BRG = 1

$$\text{Baud Rate} = \frac{F_P}{4 \times (\text{BRG} + 1)}$$

$$\text{BRG} = \frac{F_P}{14 \times \text{Baud Rate}} - 1$$

Note: BRG values should be three or more for proper smart card communication.

UART fixed division baud rate setup procedure:

1. Select the clock input source with the CLKSEL[1:0] bits.
2. Clear the CLKMOD bit.
3. Select the clock prescaler by writing a value to BRGS.
4. Using Equation 18-1 or Equation 18-2, calculate the value for BRG and write to the UxBRG register.
5. Set the ON bit.

18.4.1.2 Fractional Division Mode

To reduce baud rate error, a fractional division scheme can be used by setting CLKMOD = 1. The fractional baud clock circuit works by occasionally extending clock pulses of the 16x baud clock to achieve a baud clock closer to the ideal baud rate. This mode allows for faster operation of the UART while maintaining the noise rejection benefits of 16x oversampling, where in Legacy mode, a small value of BRG results in an unacceptable error.

The fractional Baud Rate Generation logic performs modulo arithmetic, where the value 16 is constantly accumulated in a counter until the sum is larger than the UxBRG register value. When the sum becomes larger than the UxBRG register value, a clock pulse is produced and the accumulated value is reduced by the value in the UxBRG register.

A timing example for the fractional baud clock circuit is shown in Figure 18-4. In this example, a 50 MHz peripheral clock and a target baud rate of 921600 baud are used. This results in UxBRG = 54. The upper waveform shows one full bit time, while the lower waveform shows the accumulation process. In this example, the 16x baud clock pulses are generated every three to four peripheral clock (F_P) cycles. While this results in 16x baud clock sampling pulses with unequal periods, each full bit time (16 pulses of the 16x baud clock) will be equal. The resulting bit period generated will be closer to the ideal bit period than with the legacy divider-based BRG.

Equation 18-3 shows the baud rate formulas.

Equation 18-3. Baud Rate Formulas

$$\text{Baud Rate} = \frac{F_P}{\text{BRG}}$$

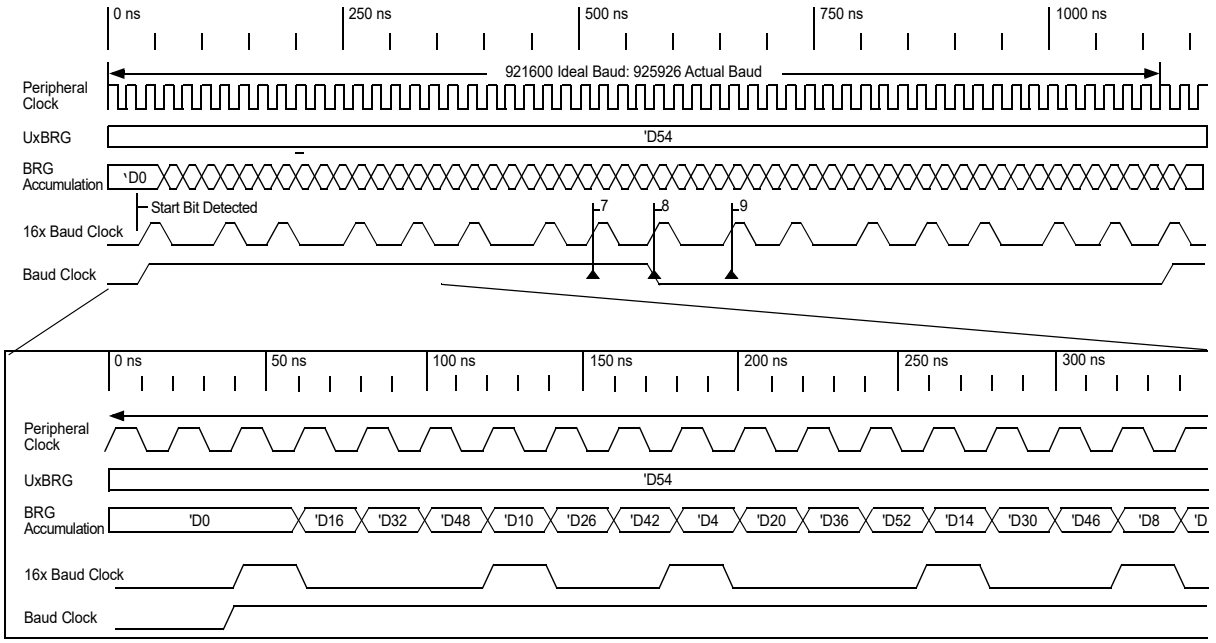
$$\text{BRG} = \frac{F_P}{\text{Baud Rate}}$$

Note: When CLKMOD (UxCON[27]) = 1, the minimum BRG value is 16.

UART fractional baud rate setup procedure:

1. Select the clock input source with the CLKSEL[1:0] bits.
2. Set the CLKMOD bit.
3. Using Equation 18-3, calculate the value for BRG and write to the UxBRG register.
4. Set the ON bit.

Figure 18-4. Fractional Division Mode



18.4.1.3 UART Baud Rate Tables

UART baud rates are provided in Table 18-3, Table 18-4 and Table 18-5 for different Peripheral Clock Frequencies (Fp). The minimum and maximum baud rates for each frequency are also shown.

Table 18-3. UART Baud Rates (CLKMOD = 0 and BRGS = 0)

BAUD RATE	FP = 100 MHz			FP = 80 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
9,600	9600.6	0.01	650	9615.4	0.16	519
19,200	19230.8	0.16	324	19230.8	0.16	259
38,400	38580.2	0.47	161	38461.5	0.16	129
56,000	56306.3	0.55	110	56179.8	0.32	88
115,000	115740.7	0.64	53	116279.1	1.11	42
250,000	250000.0	0.00	24	250000.0	0.00	19
300,000	312500.0	4.17	19	312500.0	4.17	15
500,000	520833.3	4.17	11	500000.0	0.00	9
1,000,000	1041666.7	4.17	5	1000000.0	0.00	4
Min.	5.96	0.00	1048575	4.77	0.00	1048575
Max.	6250000.0	0.00	0	5000000.0	0.00	0

BAUD RATE	FP = 50 MHz			FP = 32 MHz			FP = 4 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
9,600	9615.4	0.16	324	9615.4	0.16	207	9615.4	0.16	25
19,200	19290.1	0.47	161	19230.8	0.16	103	19230.8	0.16	12
38,400	38580.2	0.47	80	38461.5	0.16	51	41666.7	8.51	5
56,000	56818.2	1.46	54	57142.9	2.04	34	62500.0	11.61	3
115,000	115740.7	0.64	26	117647.1	2.30	16	125000.0	8.70	1
250,000	260416.7	4.17	11	250000.0	0.00	7	250000.0	0.00	0
300,000	312500.0	4.17	9	333333.3	11.11	5			
500,000	520833.3	4.17	5	500000.0	0.00	3			
1,000,000	1041666.7	4.17	2	1000000.0	0.00	1			
Min.	2.98	0.00	1048575	1.91	0.00	1048575	0.24	0.00	1048575
Max.	3125000.0	0.00	0	2000000.0	0.00	0	250000.0	0.00	0

Table 18-4. UART Baud Rates (CLKMOD = 0 and BRGS = 1)

BAUD RATE	FP = 100 MHz			FP = 80 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
9,600	9600.6	0.01	2603	9601.5	0.02	2082
19,200	19201.2	0.01	1301	19212.3	0.06	1040
38,400	38402.5	0.01	650	38461.5	0.16	519
56,000	56053.8	0.10	445	56022.4	0.04	356
115,000	115207.4	0.18	216	115606.9	0.53	172
250,000	250000.0	0.00	99	250000.0	0.00	79

.....continued

BAUD RATE	FP = 100 MHz			FP = 80 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
300,000	301204.8	0.40	82	303030.3	1.01	65
500,000	500000.0	0.00	49	500000.0	0.00	39
1,000,000	1000000.0	0.00	24	1000000.0	0.00	19
Min.	23.84	0.00	1048575	19.07	0.00	1048575
Max.	25000000.0	0.00	0	20000000.0	0.00	0

BAUD RATE	FP = 50 MHz			FP = 32 MHz			FP = 4 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
9,600	9600.6	0.01	1301	9603.8	0.04	832	9615.4	0.16	103
19,200	19201.2	0.01	650	19230.8	0.16	415	19230.8	0.16	51
38,400	38461.5	0.16	324	38461.5	0.16	207	38461.5	0.16	25
56,000	56053.8	0.10	222	56338.0	0.60	141	58823.5	5.04	16
115,000	115740.7	0.64	107	115942.0	0.82	68	125000.0	8.70	7
250,000	250000.0	0.00	49	250000.0	0.00	31	250000.0	0.00	3
300,000	304878.0	1.63	40	307692.3	2.56	25	333333.3	11.11	2
500,000	500000.0	0.00	24	500000.0	0.00	15	500000.0	0.00	1
1,000,000	1041666.7	4.17	11	1000000.0	0.00	7	1000000.0	0.00	0
Min.	11.92	0.00	1048575	7.63	0.00	1048575	0.95	0.00	1048575
Max.	12500000.0	0.00	0	8000000.0	0.00	0	1000000.0	0.00	0

Table 18-5. UART Baud Rates (CLKMOD = 1)

BAUD RATE	FP = 100 MHz			FP = 80 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
9,600	9600.6	0.01	10416	9600.4	0.00	8333
19,200	19201.2	0.01	5208	19203.1	0.02	4166
38,400	38402.5	0.01	2604	38406.1	0.02	2083
56,000	56022.4	0.04	1785	56022.4	0.04	1428
115,000	115074.8	0.07	869	115107.9	0.09	695
250,000	250000.0	0.00	400	250000.0	0.00	320
300,000	300300.3	0.10	333	300751.9	0.25	266
500,000	500000.0	0.00	200	500000.0	0.00	160
1,000,000	1000000.0	0.00	100	1000000.0	0.00	80
Min.	95.37	0.00	1048575	76.29	0.00	1048575
Max.	6250000.0	0.00	0	5000000.0	0.00	16

BAUD RATE	FP = 50 MHz			FP = 32 MHz			FP = 4 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
9,600	9600.6	0.01	5208	9601.0	0.01	3333	9615.4	0.16	416

.....continued

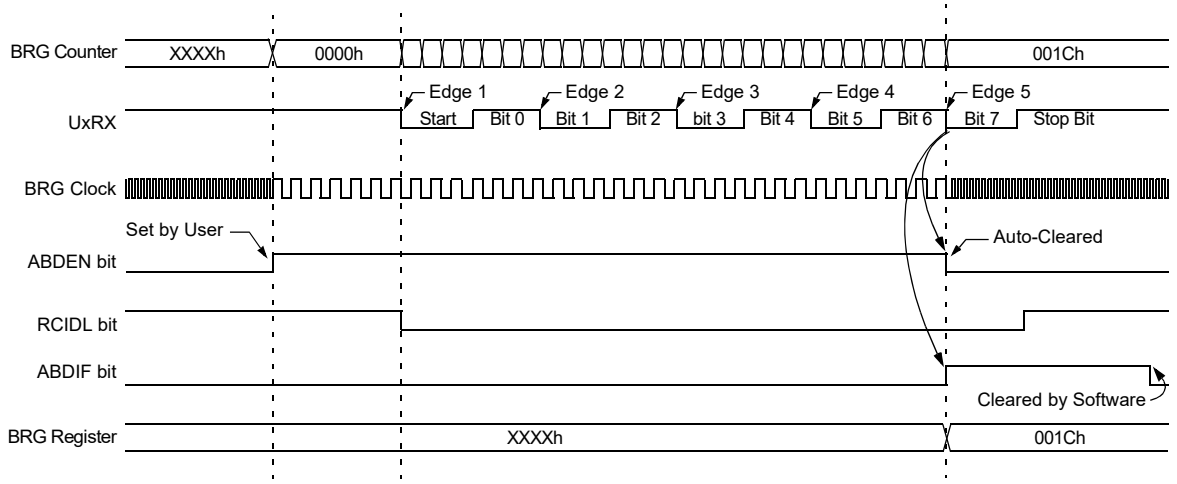
BAUD RATE	FP = 50 MHz			FP = 32 MHz			FP = 4 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
19,200	19201.2	0.01	2604	19207.7	0.04	1666	19230.8	0.16	208
38,400	38402.5	0.01	1302	38415.4	0.04	833	38461.5	0.16	104
56,000	56053.8	0.10	892	56042.0	0.08	571	56338.0	0.60	71
115,000	115207.4	0.18	434	115107.9	0.09	278	117647.1	2.30	34
250,000	1250000.0	0.00	200	250000.0	0.00	128	250000.0	0.00	16
300,000	301204.8	0.40	166	301886.8	0.63	106			
500,000	500000.0	0.00	100	500000.0	0.00	64			
1,000,000	1000000.0	0.00	50	1000000.0	0.00	32			
Min.	47.68	0.00	1048575	30.52	0.00	1048575	3.81	0.00	1048575
Max.	3125000.0	0.00	16	2000000.0	0.00	16	250000.0	0.00	16

18.4.1.4 Auto-Baud Feature

The auto-baud feature allows the receiver to determine the baud rate of the transmitter and synchronize to it. The transmitter sends a byte value of 0x55 (Sync byte) to the receiver, and the receiver calculates the average bit time from the falling edges. The UxBRG register is then written with the corresponding value. Sync byte (0x55) will not be stored in the Rx buffer. Auto-baud is supported in both Legacy and Fractional Baud Rate Generation modes (CLKMOD = `1` or `0`). The Sync byte may be preceded with a Break.

To enable auto-baud, the ABDEN bit (UxCON[6]) is set and the UART will begin to look for a falling edge (Start bit of Sync byte). While the auto-baud sequence is in progress, the UART state machine is held in Idle mode. On the fifth RX pin falling edge, an accumulated BRG counter value totaling the proper BRG period is transferred to the UxBRG register. Once the auto-baud process is complete, the ABDEN bit will be cleared by hardware and the ABDIF flag (UxUIR[6]) is set. If the ABDIE (UxUIR[2]) interrupt enable bit is set, an event interrupt will be generated. See Figure 18-5 for the Auto-Baud Detection sequence.

Figure 18-5. Auto-Baud Detection



If the fifth and final falling edge is not detected before the BRG counter rolls over, the ABDOVF flag (UxSTAT[5]) will set to indicate the condition. The flag cannot be cleared until ABDEN is cleared. If the ABDOVIE bit (UxSTAT[13]) is set, an error interrupt will be generated. For more information on interrupts, see 18.6. Interrupts.

Auto-baud setup procedure:

1. Configure the UART for receive operation as detailed in 18.4.2.1.2. Asynchronous Receive.
2. Set the ABDEN bit. If a Break precedes the Sync byte, also set the WUE (UxCON[12]) bit to configure the UART to perform the auto-baud procedure on the Sync and not the Break. The RXBKIF flag (UxSTAT[2]) will not be set.
3. Poll the ABDEN or ABDIF bit to determine when auto-baud has finished.

Alternatively, if a Break precedes the Sync and it is desired to detect the Break, use the following sequence:

1. Configure the UART for receive operation as detailed in 18.4.2.1.2. Asynchronous Receive.
2. Wait for the RXBKIF flag to set (see Break Character Reception for details).
3. Immediately set the ABDEN bit.
4. Poll the ABDEN or ABDIF bit to determine when auto-baud has finished.

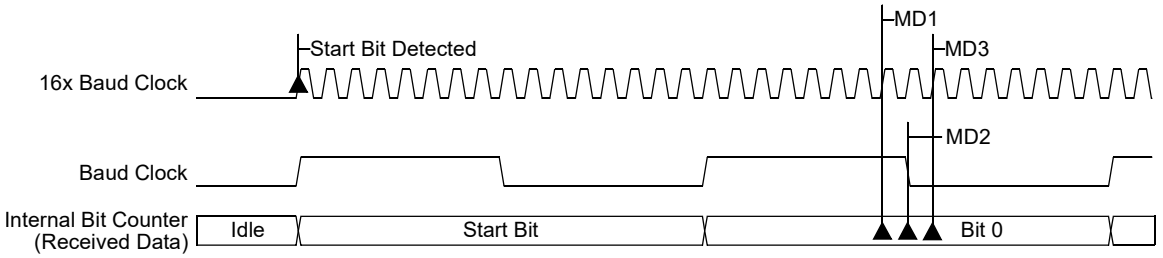
18.4.1.5 Data Bit Detection

18.4.1.5.1 Legacy Mode

Low-Speed Mode (CLKMOD = 0 AND BRGS = 0)

In Low-Speed mode, each bit of the received data is 16 clock pulses wide. To detect the value of an incoming data bit, the bit is sampled at the seventh, eighth and ninth rising edges of the clock. These rising edges are called Majority Detection (MD) edges. This mode is more robust than High-Speed mode.

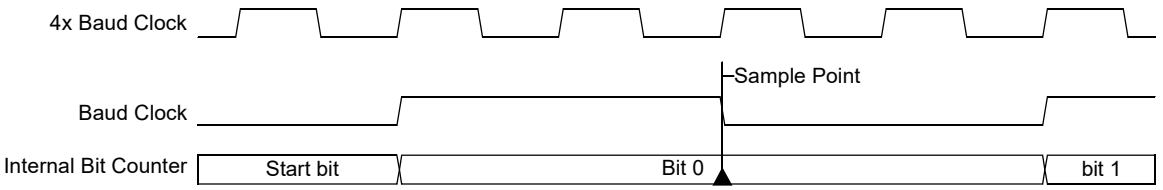
Figure 18-6. Low-Speed Mode with Majority Detection



High-Speed Mode (CLKMOD = 0 and BRGS = 1)

In High-Speed mode, each bit of the received data is four clock pulses wide. This mode does not provide enough edges to support the Majority Detection method. Therefore, the received data are sampled at the one-half bit width.

Figure 18-7. High-Speed Mode without Majority Detection



18.4.1.5.2 Fractional Division Mode (CLKMOD = 1)

In Fractional Division mode, each bit of the received data is 16 clock pulses wide. To detect the value of an incoming data bit, the bit is sampled at the seventh, eighth and ninth rising edges of the clock. Refer to Figure 18-4.

18.4.2 UART Modes

18.4.2.1 Asynchronous Mode

Asynchronous mode supports standard UART communication with the following configurable options:

- 7, 8-Bit Data Width
- 1, 1.5 or 2 Stop Bits
- Even, Odd or No Parity (Ninth data bit)
- Independently Selectable TX and RX Polarity
- Address Detect (9th data bit)
- Auto-Baud
- Break Transmission/Detection
- Flow Control (XON/XOFF and HW)
- Half/Full-Duplex TX Pin Control
- TX and RX Interrupt Configuration

The MODE[3:0] bits (UxCON[3:0]) are used to select the high-level operational mode of the UART for both transmit and receive. The five Asynchronous mode selections configure data width, parity and address detect, with the rest of the configuration options as spate controls.

18.4.2.1.1 Asynchronous Transmit

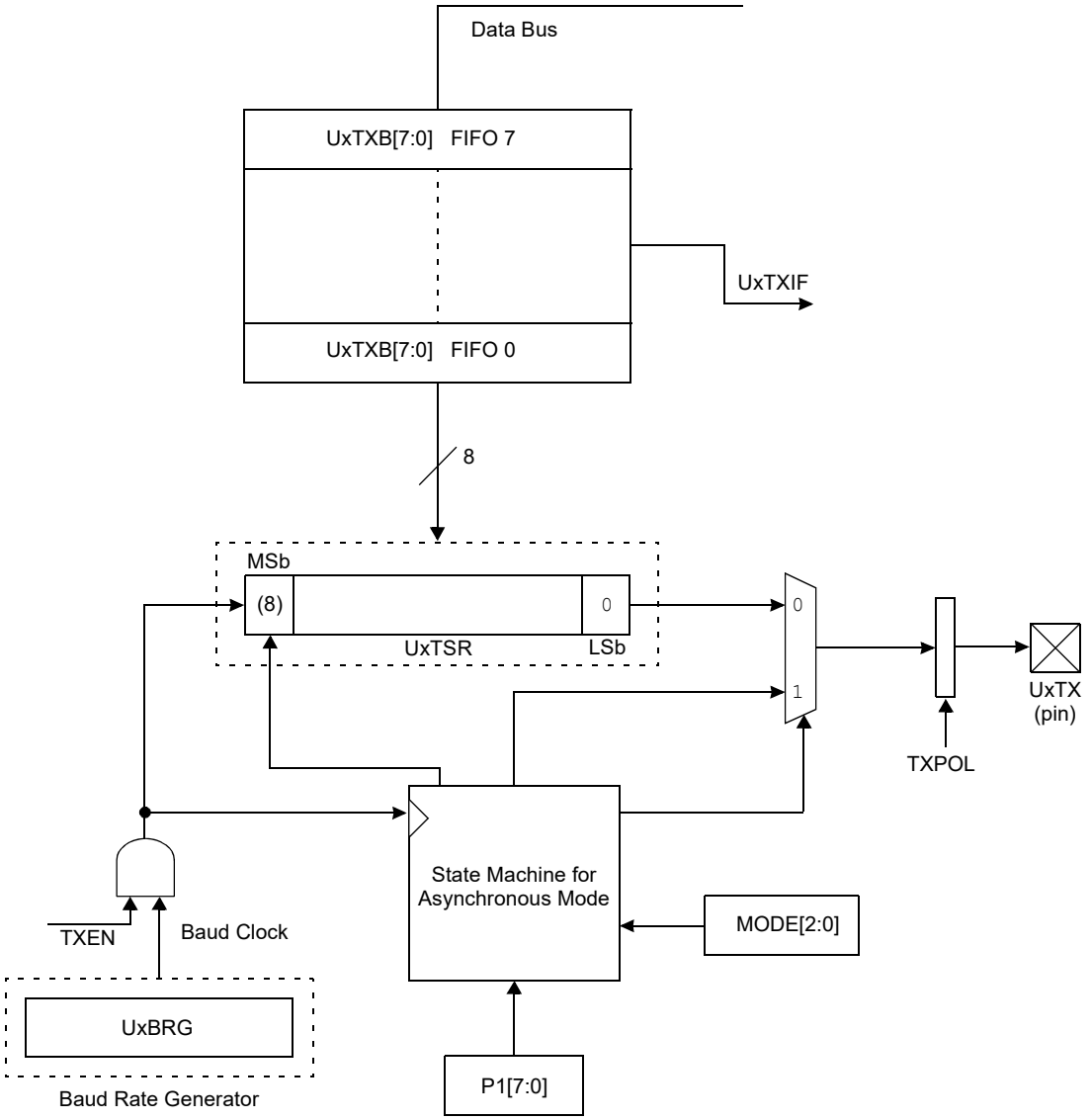
The transmitter block diagram of the UART module is illustrated in Figure 18-8. The important part of the transmitter is the UARTx Transmit Shift Register (UxTSR). The Shift register obtains its data from the transmit FIFO buffer, UxTXB. The UxTXB register is loaded with data in software.

The UxTSR register is not loaded until the Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the UxTSR is loaded with new data from the UxTXB register, if available.

Note: The UxTSR register is not mapped in data memory, so it is not available to the user application.

The transmission is enabled by setting the TXEN enable bit (UxCON[5]). The actual transmission will not occur until the UxTXB register has been loaded with data and the Baud Rate Generator (UxBRG) has produced a shift clock (Figure 18-8). Normally, when the first transmission is started, the UxTSR register is empty, so a transfer to the UxTXB register will result in an immediate transfer to UxTSR when the TXEN bit is set. When the TXEN bit is written to '0', the transmit process ends at the end of the current byte. As a result, the UxTX pin will revert to a High-Impedance state.

Figure 18-8. Asynchronous Transmitter Block Diagram



Note: 'x' denotes the UART number.

Setup for UART Transmit

The following procedure is used to transmit a byte of data:

1. Configure the clock input and baud rate as detailed in 18.4.1. Clocking and Baud Rate Configuration.
2. Configure the data width and parity by writing a selection to the MODE[3:0] bits.
3. Configure the polarity, Stop bit duration and flow control.
4. Configure the TX interrupt watermark using the TXWM[2:0] bits (UxSTAT[30:28]).
5. Configure the address detect if needed as detailed in 18.4.2.1.6. Address Detect.
6. Set the ON bit (UxCON[15]).
7. Set the TXEN bit (UxCON[5]).
8. Write the data byte value to the UxTXB register.

A TX interrupt will be generated according to the TXWM[2:0] bits' interrupt watermark setting. The TXWMx bits can be configured to generate a TX interrupt when the buffer has one to eight empty slots.

The UARTx Transmit Buffer (UxTXB) has two associated flags to indicate its contents. The TX Buffer Empty Status bit, TXBE (UxSTAT[21]), indicates that the buffer is empty, and the TX Buffer Full Status bit, TXBF (UxSTAT[20]), indicates that there are no empty slots in the buffer and it should not be written.

Example 18-1. UART1 Transmission with Interrupts

```
#include<xc.h>
#define FP 4000000
#define BAUDRATE 9600
#define BRGVAL (((FP/BAUDRATE)/16)-1)
uint8_t TxChar[10] = {'1','2','3','4','5','6','7','8','9','0'};
uint8_t i=0;
int main(void) {
    // Configure oscillator as needed

    // Configure oscillator as needed
    _RP23R = 9;         // Assign U1TX to pin RP23

    U1CONbits.MODE = 0;     // Asynchronous 8-bit UART
    U1CONbits.BRGS = 0;     // Low-Speed Mode
    U1CONbits.CLKSEL = 0;   // System_Clock/2 as Baud Clock source
    U1CONbits.STP = 0;      // 1 stop bit
    U1BRG = BRGVAL;         // BRG setting for 9600 baud rate
    U1STATbits.TXWM = 0;    // Sets transmit interrupt when there are eight
                            // empty slots in the buffer
    _U1TXIE = 1;            // Enable Transmit interrupt
    U1CONbits.ON = 1;       // Enable UART
    U1CONbits.TXEN = 1;     // Enable UART TX. This generates TX interrupt.
    while(1)
    {
    }
    return 0 ;
}
void __attribute__((interrupt, no_auto_psv)) _U1TXInterrupt(void)
{
    _U1TXIF = 0; // Clear TX interrupt flag
    while((U1STATbits.TXBF == 0) && (i<10) )
    {
        U1TXB = TxChar[i++]; // Transmit one character
    }
}
```

Transmit Errors and Events

The UART is capable of detecting bus collisions. The received byte is compared against the last byte transmitted to identify differences. The UxTX and UxRX pin functions need to be mapped to separate pins using Peripheral Pin Select (PPS). The UxRX pin has to be able to receive a byte in order for the comparison to happen. If the pin is stuck at Vdd or ground, such that a valid Start and Stop bit are not detected, the comparison cannot take place. If a bus collision is detected, it is flagged by the TXCIF bit (UxSTAT[0]). If the TXCIE bit (UxSTAT[8]) is set, an error interrupt will be generated.

If a write to UxTXB is done when the buffer is already full, a transmit write error is indicated by the TXWRE bit (UxSTAT[23]).

The Transmit Shift Register (TSR) has a status flag, TXMTIF (UxSTAT[7]), associated with it to indicate when a byte transmission is complete. An interrupt can be generated by setting the TXMTIE bit (UxSTAT[15]).

Half-Duplex Transmit

In a half-duplex application, the UxTX and UxRX lines are shorted together; this allows a reduction in wire count. However, control is needed to avoid both devices transmitting at the same time.

Setting the HALFDPLX bit (UxCON[24]) configures the UxTX pin to only drive the line during a byte transmission and is tri-stated at all other times. In UART systems, tri-state is not a permissible state, so it is important that the user enable a weak pull-up on the UART pad when such half-duplex systems are used. The RCIDL bit (UxSTAT[19]) can be read to determine if the line is Idle and a byte can be sent. However, a collision can still occur during the transmission. The Transmit Collision Interrupt Flag bit, TXCIF (UxSTAT[0]), can be read to determine if the byte was transmitted successfully. The receiver has to remain enabled for the transmission collision to be detected. If TXCIE (UxSTAT[8]) is set, the receive watermark interrupt will not be generated when TXCIF (UxSTAT[0]) is set.

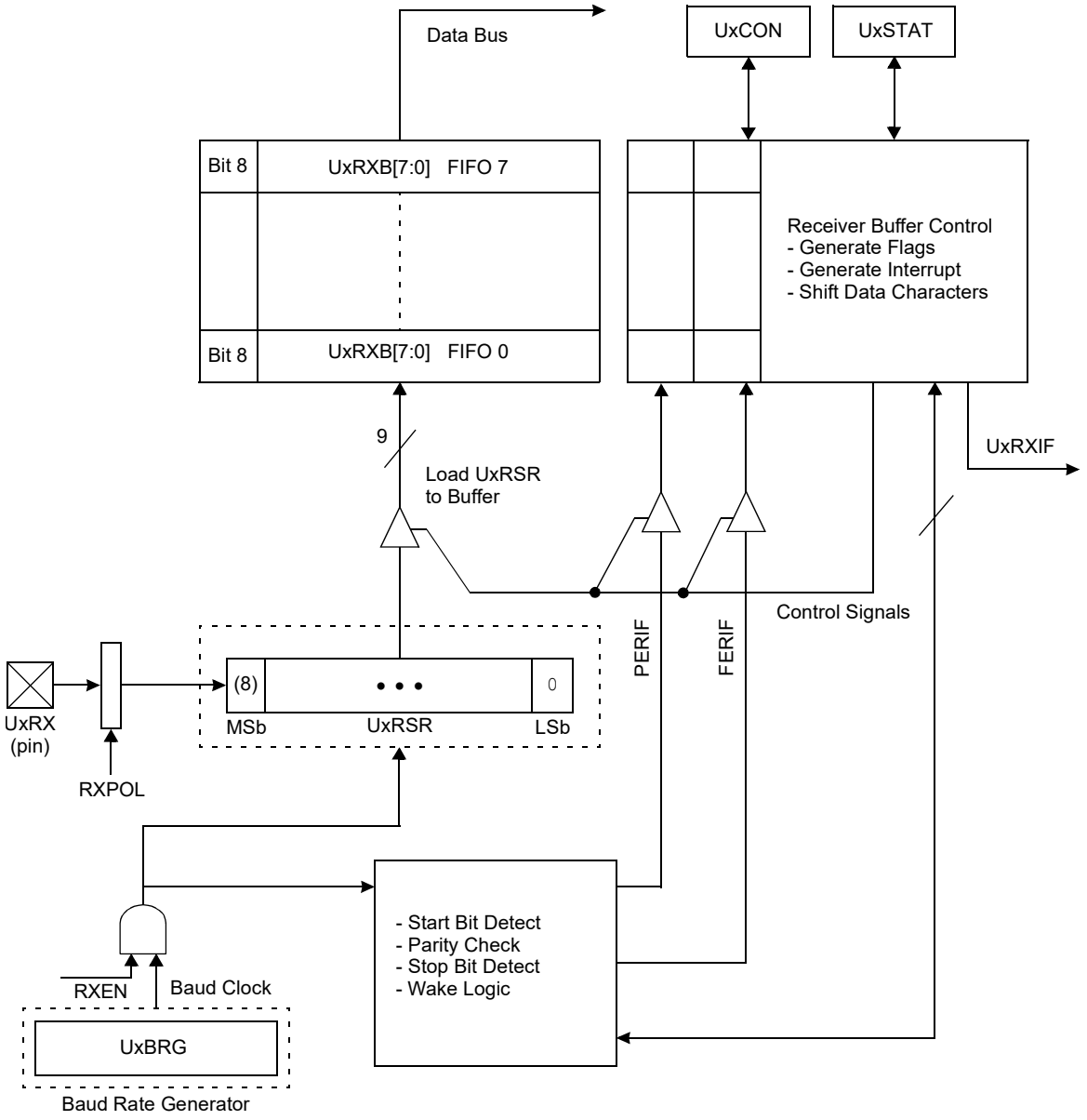
18.4.2.1.2 Asynchronous Receive

The receiver block diagram of the UART module is illustrated in Figure 18-9. The important part of the receiver is the UARTx Receive (Serial) Shift Register (UxRSR). The data is received on the UxRX pin. After sampling the UxRX pin for the Stop bit, the received data in the UxRSR register is transferred to the receive FIFO, if it is empty.

Note: The UxRSR register is not mapped in data memory, so it is not available to the user application.

The reception is enabled by setting the RXEN bit (UxCON[4]). Clearing the RXEN bit causes the receive shifter to stop. As a consequence, RXIDL (UxSTAT[19]) is set to '1'.

Figure 18-9. Asynchronous Receiver Block Diagram



Note: 'x' denotes the UART number.

Setup for UART Receive

The following procedure is used to receive a byte of data:

1. Configure the clock input and baud rate as detailed in 18.4.1. Clocking and Baud Rate Configuration.
2. Configure the data width and parity by writing a selection to the MODE[3:0] bits.
3. Configure the polarity, Stop bit duration and flow control.
4. Configure the RX interrupt watermark using the RXWM[2:0] bits (UxSTAT[26:24]).
5. Configure the address detect if needed as detailed in 18.4.2.1.6. Address Detect.
6. Set the ON bit (UxCON[15]).
7. Set the RXEN bit (UxCON[4]).

An RX interrupt will be generated when a byte is received according to the UART Receive Interrupt Select bits setting, RXWM[2:0] (UxSTAT[26:24]). The RXWMx bits can be configured to generate an RX interrupt when the RX buffer contains 1-8 bytes.

Software can then read the data from the UxRXB register. The time, relative to the Stop bit when the RX interrupt is generated, is configurable using the STPMD bit (UxSTAT[22]). By default, an RX interrupt is generated in the middle of the Stop bit. Writing a '1' will move the RX interrupt to the end of the Stop bit.

The RXBF status bit (UxSTAT[16]) can be read by software to determine if the receive buffer is full and a read operation of UxRXB is required to allow reception of additional bytes. Similarly, the RXBE status bit (UxSTAT[17]) can be read with software to determine if the receive buffer is empty.

Example 18-2. UART1 Reception with Interrupts

```
#include<xc.h>
#define FP 4000000
#define BAUDRATE 9600
#define BRGVAL (((FP/BAUDRATE)/16)-1)
uint8_t ReceivedChar[10];
uint8_t i=0;
int main(void) {
    // Configure oscillator as needed

    // Configure oscillator as needed
    ANSELB=0;
    _U1RXR = 24; // Assign U1RX to RP24

    U1CONbits.MODE = 0;         // Asynchronous 8-bit UART
    U1CONbits.BRGS = 0;         // Low-Speed Mode
    U1CONbits.CLKSEL = 0;       // System_Clock/2 as Baud Clock source
    U1CONbits.STP = 0;          // 1 stop bit
    U1BRG = BRGVAL;             // BRG setting for 9600 baud rate
    U1STATbits.RXWM = 0;        // Interrupt when there is one word or more in the buffer
    _U1RXIE = 1;                // Enable Receive interrupt
    U1CONbits.ON = 1;           // Enable UART
    U1CONbits.RXEN = 1;         // Enable UART RX

    while(1)
    {
    }
    return 0;
}
void __attribute__((interrupt, no_auto_psv)) _U1RXInterrupt(void)
{
    _U1RXIF = 0; // Clear RX interrupt flag
    while(U1STATbits.RXBE == 0)     // Check if RX buffer has data to read
    {
        ReceivedChar[i++] = U1RXB; // Read a character from RX buffer
    }
}
```

Receive Errors and Events

The receive framing and parity errors are associated with each byte received. Frame and parity error flags, indicated by the FERIF and PERIF bits, will indicate only the error status of the last data byte received. As UxRXB is read, the flags indicate the status of the current (top) byte in the buffer. This behavior differs from prior UART modules.

If a byte is received when the receive buffer is full, the RXFOIF bit (UxSTAT[1]) will set. Setting the corresponding error interrupt enable will generate an error interrupt. To clear the RXFOIF bit, the receive buffer needs to be read at least once. This behavior differs from prior UART modules. The receiver can handle overflow conditions in one of two options, defined by the RUNOVF (UxCON[23]) bit. By default, when RUNOVF = 0, the receiver will stop receiving data when the RX buffer is full. Alternatively, when RUNOVF = 1, the receiver will continue to receive data and overwrite the contents of the RX shifter.

A line Idle condition (line high) is indicated by the RCIDL bit (UxSTAT[19]). The flag will clear when a Start bit is detected and a reception is in progress.

18.4.2.1.3 Parity Support

Parity is a simple method of single bit error detection. The data bits are summed and compared to the parity value indicating a bit error. Parity selection can either be even or odd and is represented by the ninth data bit. To calculate parity, the number of data bits that are a '1' are counted.

- Even parity is defined as an odd number of data bits whose values are '1'
- Odd parity is defined as an even number of data bits whose values are '1'

The parity bit itself is then added to the count, hence the 'even' or 'odd' designation. Parity calculation and checking is enabled by selecting one of the two 8-Bit Asynchronous Parity modes (MODE[3:0] = 0b001x).

18.4.2.1.4 Break Character

A Break character is defined as several consecutive low bit times, usually longer than a whole byte. In Asynchronous mode, the UART will transmit a 13-bit long duration Break, and in receive, will flag 11 low bit times as a Break sequence.

Transmitting a Break Character

A Break character is transmitted by setting the SENDB bit (UxCON[8]) and then writing any value to UxTXB. The contents of UxTXB will follow the Break character.

Alternatively, the BRKOVR bit (UxCON[9]) can be controlled by software to override and drive the UxTX line for any duration. When TXPOL (UxCON[18]) = 0, the UxTX line will be driven low and when TXPOL = 1, the UxTX line will be driven high.

Break Character Reception

The receiver is always looking for a Break sequence and can detect one, even in the middle of a byte reception. A Break reception is indicated by the RXBKIF flag (UxSTAT[2]). An interrupt can be optionally generated by setting the RXBKIE bit (UxSTAT[10]). The Break detection criteria can be configured using the RXBIMD bit (UxCON[11]). By default, the RXBKIF flag will set when the line makes a low-to-high transition after 11 low bit times, signaling the end of the Break sequence. Alternatively, when RXBIMD = 1, the flag will set when the eleventh low bit time is detected.

18.4.2.1.5 Checksum

For LIN mode, two kinds of checksum are available: legacy and enhanced. In legacy checksum, only data bytes D0 through D7 are used to calculate the checksum. In enhanced checksum, data bytes D0 through D7 and PID[5:0], P0 and P1 are included. Which checksum is used in the calculation can be controlled by software using the C0EN bit. Refer to 18.4.2.2. LIN/J2602 for more information on checksum calculation.

For all other modes, the C0EN bit is ignored and UART calculates the checksum for every transmitted or received byte. Checksum register UxCHK is cleared on receiving a Break sequence in all protocol modes. These registers can also be cleared by the user.

18.4.2.1.6 Address Detect

Address Detect mode is used when multiple receivers are connected to a transmitter. It allows a receiver to determine if the message is intended for it and to ignore those that are not. If the ninth

data bit is a '1', the data are recognized as addresses to be processed by the receiver. An address mask is provided to allow multiple receivers to accept the same address.

If an address match is successful, the unmasked address is present in UxRXB and an RX interrupt is generated. If the address match is not successful, all of the following data are ignored until a byte with the ninth bit set is received.

In 8-Bit Address Detect mode, the transmitted address IDs are written to Parameter 1. For receivers, the expected address is written to Parameter 2 (P2[7:0]) and the mask value to Parameter 3 (P3[7:0]). Mask bit values of '1' will include the respective bit position in the compare, whereas a '0' indicates a 'don't care'. A mask value of 0x00 will accept all address values, effectively disabling the address detect feature. A mask value of 0xFF will allow only one matching value.

Address Detect Transmit

The following procedure is used to transmit in Address Detect mode:

1. Configure the UART for asynchronous transmit as detailed in 18.4.2.1.1. Asynchronous Transmit with the MODE[3:0] bits set to '0b0100'.
2. If a Break is desired, write a '1' to SENDB (UxCON[8]).
3. Write the address value to Parameter 1.
   a. If SENDB = 0, the contents of Parameter 1 will be transmitted with the ninth bit set.
   b. If SENDB = 1, a Break will be transmitted, followed by the contents of Parameter 1 with the ninth bit set.
4. Write data to be transmitted to the UxTXB register.

Address Detect Receive

The following procedure is used for receive in Address Detect mode. A framing error will not prevent an address match.

1. Configure the UART for asynchronous receive as detailed in 18.4.2.1.2. Asynchronous Receive with the MODE[3:0] bits set to '0b0100'.
2. Write the address match value to Parameter 2.
3. Write the optional address mask value to Parameter 3.
4. Upon the reception of a valid address, the PERIF bit will be set to indicate that the value stored in UxRXB is an address. The subsequent data can be read from UxRXB as they become available.

18.4.2.1.7 Flow Control

Flow control is used to prevent data loss between two devices. One device may be slower or have to process data. Flow control allows a device to tell the other to wait before sending additional bytes that may overrun its buffers. The UART supports two types of flow control:

- XON/OFF Messaging
- Hardware Flow Control ($\overline{\text{UxRTS}}$, $\overline{\text{UxCTS}}$, $\overline{\text{UxDTR}}$, $\overline{\text{UxDSR}}$)

XON/XOFF

XON/XOFF uses messages implemented as special byte values and does not require additional HW lines. An XON command is implemented by sending a byte value of 0x11 and an XOFF is implemented by a value of 0x13. There are two states of the control mechanism as indicated by the XON bit (UxSTAT[18]). By default, the UART is in the XON = 1 state and will transmit data as they become available in the TX buffer. If the device receives an XOFF command, the XON status bit is cleared and transmission stops until another XON command is received. XON/OFF commands are transmitted in the same manner as regular data.

The receiver keeps track of the UART buffer; if the RX buffer is left with two empty slots, an XOFF command is sent by the module automatically. After that, the module sends an XON command if more than two empty slots become available in the RX buffer.

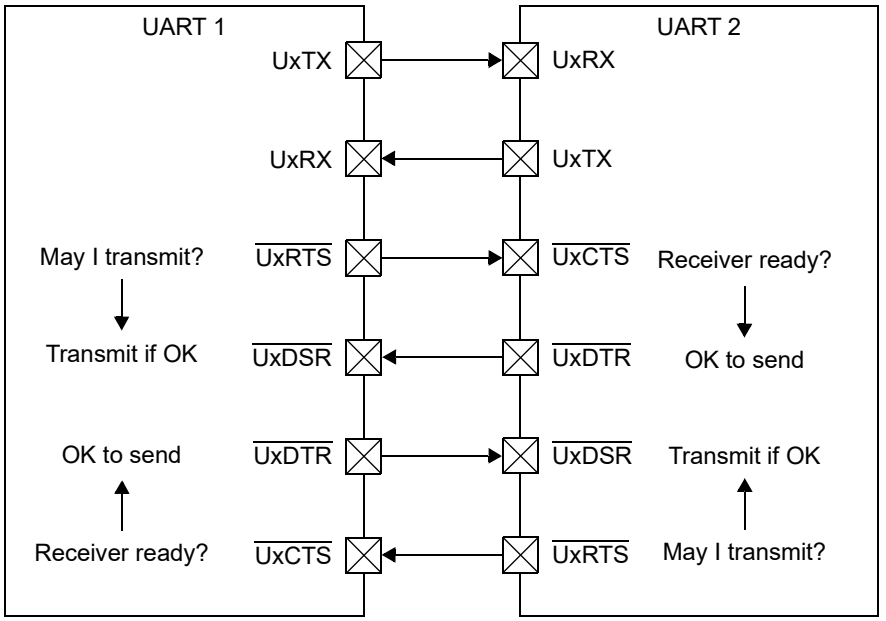
Hardware Flow Control

Hardware flow control uses up to four additional pins to indicate when a device is ready to receive additional data. The four active-low device pins are shown in Table 18-6. Figure 18-10 shows connections between two UARTs.

Table 18-6. Hardware Flow Control Pin Functions

Signal Name	Description	Used By	Direction
$\overline{\text{UxDSR}}$	Data-Set-Ready	Transmitter	Input
$\overline{\text{UxRTS}}$	Request-to-Send	Transmitter	Output
$\overline{\text{UxCTS}}$	Clear-to-Send	Receiver	Input
$\overline{\text{UxDTR}}$	Data-Terminal-Ready	Receiver	Output

Figure 18-10. Hardware Flow Control Pins and Connections



The transmitter asserts (drives low) the $\overline{\text{UxRTS}}$ output when it has one or more bytes in its TX buffer to indicate that it wants to send a byte. Then, the transmitter listens to the $\overline{\text{UxDSR}}$ to see if it is OK to do so. If $\overline{\text{UxDSR}}$ is active (low), the transmitter sends one byte. If $\overline{\text{UxDSR}}$ is inactive (high), it will wait.

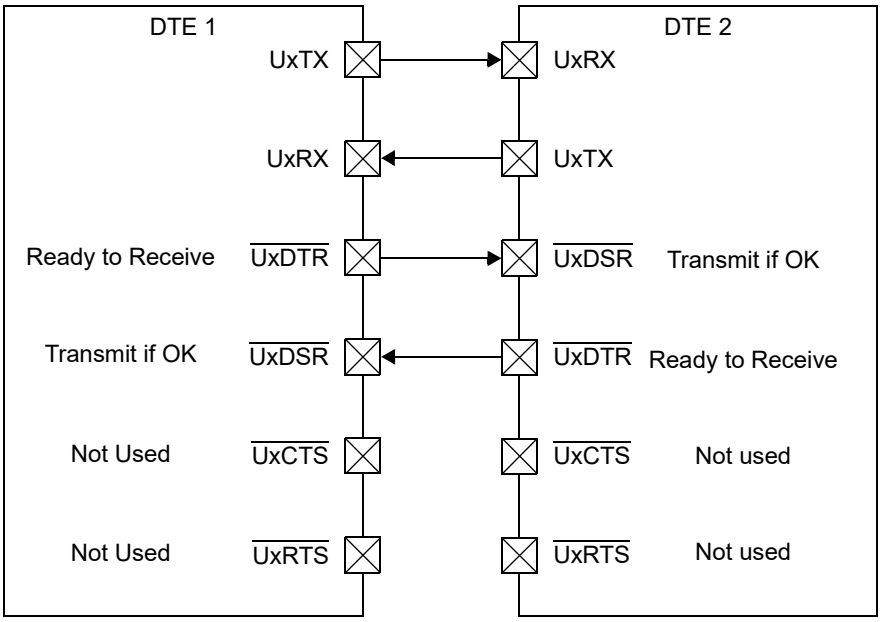
When the receiver detects the $\overline{\text{UxCTS}}$ signal going active (low), it checks to see if there are two empty slots in the receive buffer. If so, the receiver asserts (drives low) the $\overline{\text{UxDTR}}$ pin to indicate it is ready to receive data. No register setup is needed to enable the flow control pins. However, most devices will have the UART and associated flow control pins routed through the Peripheral Pin Select (PPS) feature.

When using flow control, devices can be categorized into two groups: DTE (Data Terminal Equipment) and DCE (Data Carrier Equipment). A typical DTE can be a computer or a microcontroller and a DCE is typically a modem. Not all hardware flow control pins are needed in all cases. The following sections show which flow control pins are used to interface different devices to one another.

DTE to DTE Configuration

When interfacing two DTE devices together, connect them as shown in Figure 18-11. The $\overline{\text{UxDTR}}$ output is connected to the $\overline{\text{UxDSR}}$ input terminal of the other. This allows the receiver to tell the transmitter that it is OK to transmit.

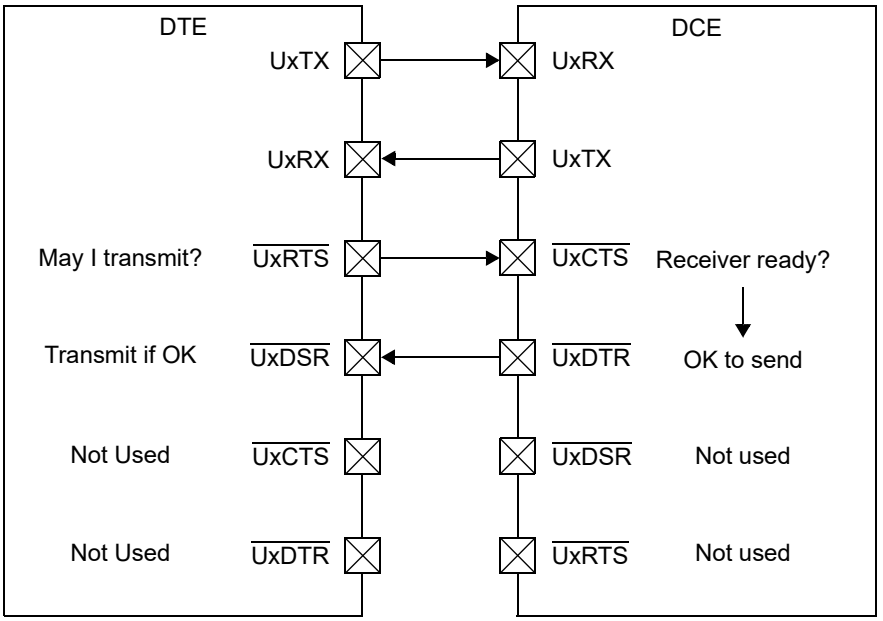
Figure 18-11. DTE to DTE Pin Connections



DTE to DCE Configuration

When interfacing a DTE device to a DCE device, connect them as shown in Figure 18-12. The $\overline{\text{UxRTS}}$ output of the DTE is connected to the $\overline{\text{UxCTS}}$ input terminal of the DCE, and the $\overline{\text{UxDTR}}$ output of the DCE is connected to the $\overline{\text{UxDSR}}$ input of the DTE. This allows the DCE to tell the DTE when it is ready to receive data.

Figure 18-12. DTE to DCE Pin Connections



18.4.2.2 LIN/J2602

The UART provides support for the Local Interconnect Network (LIN) protocol for both Commander and Responder processes to reduce software overhead. The LIN protocol is typically used in automotive applications and packages bytes into message frames. The LIN protocol has two types of processes: Commander and Responder. A network can have only one Commander and multiple Responders. The Commander process transmits a header containing a command that the Responder(s) can respond to. The Commander process part of a LIN message frame consists of the following:

1. Break character (11 bits minimum received, 13 transmitted).
2. Delimiter bit.
3. Sync byte (0x55).
4. Protected ID (PID) field.

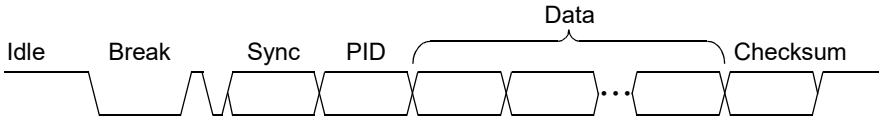
The Responder processes, then completes, the message frame by transmitting the requested data and checksum.

1. Data (up to eight bytes).
2. Checksum.

The UART has two LIN modes, Commander/Responder and Responder Only, selected by the MODE[3:0] bits (UxCON[3:0]). The Commander/Responder mode allows a single instance of a UART to handle both Commander and Responder software processes.

A LIN frame starts with the Commander process sending a Break, followed by a Sync to allow the receiver to synchronize the baud rate to the transmitter. The PID byte follows and is used by the Responder to determine if, or how, to respond. A Responder process then responds with up to eight bytes of data and a checksum. A LIN frame is shown in Figure 18-13.

Figure 18-13. LIN Frame



The PID byte consists of six bits of data and two parity bits, P0 followed by P1. The PID value is written to Parameter 1 (P1[5:0]) and the parity bits are automatically calculated. Parameter 1 can only be written when the transmitter is Idle. The parity bits are calculated as follows:

P0 = PID[0] XOR PID[1] XOR PID[2] XOR PID[4]
P1 = NOT (PID[1] XOR PID[3] XOR PID[4] XOR PID[5])

The UART automatically calculates the checksum. Two types of LIN checksums are supported and selected by the C0EN bit (UxCON[19]). When C0EN = 0 (default), the legacy LIN checksum method is used, which uses only data bytes. When C0EN = 1, the checksum also includes the PID. The checksum is calculated by adding the number of data bytes, defined by Parameter 2, adding the carry result and finally inverting the sum.

Table 18-7 provides an example checksum calculation for a LIN frame of four data bytes in length, with data values of 0x4A, 0x55, 0x93 and 0xE5.

Table 18-7. LIN Checksum Example (C0EN = 1 or 0)

Action	Hex	Carry	D7	D6	D5	D4	D3	D2	D1	D0
0x4A	0x4A		0	1	0	0	1	0	1	0
+0x55	0x9F	0	1	0	0	1	1	1	1	1
Add Carry	0x9F		1	0	0	1	1	1	1	1

.....continued

Action	Hex	Carry	D7	D6	D5	D4	D3	D2	D1	D0
+0x93 Add Carry	0x132 0x33	1	0 0	0 0	1 1	1 1	0 0	0 0	1 1	0 1
+0xE5 Add Carry	0x118 0x19	1	0 0	0 0	0 0	1 1	1 1	0 0	0 0	0 1
Invert	0xE6(1)		1	1	1	0	0	1	1	0
Receiver Verification										
Check Local + Received	0x19(2)+0xE6 (1)									

Notes:

1. This is the checksum value transmitted as the last byte.
2. This is the checksum value calculated by the receiver.

For a transmit and receive operation, the calculated checksum is stored in 18.3.8. UxCHK.

18.4.2.2.1 LIN Commander/Responder Transmit

The following procedure is used for Commander/Responder transmit:

1. Configure the clock input and baud rate as detailed in 18.4.1. Clocking and Baud Rate Configuration.
2. Configure LIN mode by writing '0b1100' to the MODE[3:0] bits.
3. Configure the checksum type by writing the C0EN bit.
4. Set the ON, RXEN and TXEN bits.
5. Write the 6-bit PID value to Parameter 1 (P1[5:0]).

Writing to Parameter 1 will cause the Break, Sync and PID to be transmitted. If it is desired to complete the message frame in Commander/Responder mode, the following steps are used:

1. Wait for the RXBKIF bit to set.
2. Write the number of bytes to transmit to Parameter 2 (P2[7:0]).
3. Write data to transmit to the UxTXB register.

18.4.2.2.2 LIN Responder Only Receive

The Responder is typically listening for a PID that it should respond to. Reception of a Break resets the checksum, parity calculation and contents of Parameter 3. The baud rate is automatically calculated and written to UxBRG. A Break after the auto-baud sequence starts will not be detected. Either not enough edges will be received and the BRG will overflow or the auto-baud sequence will complete on edges following the Break, resulting in a wrong baud rate value. The following procedure is used for Responder Only receive:

1. Configure LIN Responder Only mode by writing '0b1011' to the MODE[3:0] bits.
2. Set the ON and RXEN bits.

Upon reception of the Break, the RXBKIF flag is set. Upon reception of the PID, an RX interrupt is generated regardless of the RXWM[2:0] bits setting.

1. The PID can be read from UxRXB. If a parity mismatch (P0 and P1) has occurred, the PERIF flag will be set.
2. Write the number of bytes to receive to Parameter 3 (P3[7:0]).
3. Configure the RX watermark interrupt setting using the RXWM[2:0] bits.
4. Read data from UxRXB upon an interrupt event.

The UART automatically verifies the checksum. The checksum that the receiver calculates is stored in the RXCHK[7:0] bits (UxCHK[23:16]). If this value doesn't match that of the received checksum, the CERIF flag (UxSTAT[4]) will be set, and if the CERIE (UxSTAT[12]) bit was set, an interrupt will be generated.

18.4.2.2.3 LIN Responder Only Transmit

A LIN Responder Only transmit is a response to a header sent by the Commander and is preceded by a receive event. The baud rate is already determined by the reception of the Sync field. The following procedure is used for Responder Only transmit:

1. Configure LIN Responder Only mode by writing '`0b1011`' to the MODE[3:0] bits.
2. Configure the checksum type by writing C0EN.
3. Set the ON, RXEN and TXEN bits.

Upon reception of the Break, the RXBKIF flag is set. Upon reception of the PID, an RX interrupt is generated regardless of the RXWM[2:0] bits setting.

1. The PID can be read from UxRXB. If a parity mismatch (P0 and P1) has occurred, the PERIF flag will be set.
2. Write the number of bytes to transmit to Parameter 2 (P2[2:0]).
3. Load UxTXB with data to transmit.

After a Sync is received, writing to UxTXB will cause the data and checksum to be transmitted. A TX interrupt will indicate transmission according to the TXWM[2:0] bits setting.

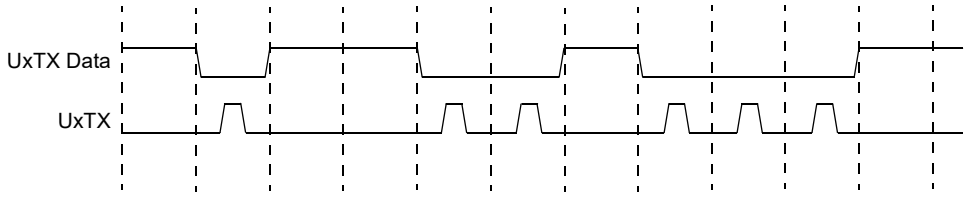
18.4.2.3 IRDA®

The UART supports the infrared IrDA protocol with a built-in encoder/decoder so an external codec is not required. Each bit time is divided into 16 clock cycles; therefore, the clock prescaler BRGS is forced to '0' and Equation 18-1 is used for the baud rate calculation. The IrDA encoding/decoding consists of the following translations:

- A value of '1' is translated to '0' for all of the 16 clocks
- A value of '0' is translated to '0' for the first seven clocks, '1' for the next three and '0' for the remaining six clocks

To enable IrDA, set MODE[3:0] = 0b1110 (UxCON[3:0]). Typical IrDA implementations require active-low Idle; therefore, it is recommended to operate the UART with both TXPOL and RXPOL set to '1'. This allows the UxTX and UxRX pins to connect directly to an infrared transceiver. Figure 18-14 shows an example IrDA waveform.

Figure 18-14. IrDA® Encoding/Decoding When TXPOL = 1



To configure the UART for IrDA mode, the following sequence is used:

1. Configure the clock input and baud rate as detailed in 18.4.1. Clocking and Baud Rate Configuration.
2. Configure IrDA mode by writing '0b1110' to the MODE[3:0] bits.
3. Set the TXPOL and RXPOL bits.
4. Configure the interrupt watermark using the TXWM[2:0] and RXWM[2:0] bits.
5. Set the ON bit.
6. Set the RXEN and TXEN bits.

In IrDA mode, the UART functions similar to Asynchronous mode regarding errors, events and interrupts.

18.4.2.4 Smart Card

The UART module supports communication with ISO 7816 smart cards. In a typical application, the UART module is intended to act as the host or terminal that always initiates communication transactions. The smart card acts as a client and always responds to commands and other stimuli from the terminal. Figure 18-15 shows a smart card subsystem using a microcontroller with a UART module for smart card data communication.

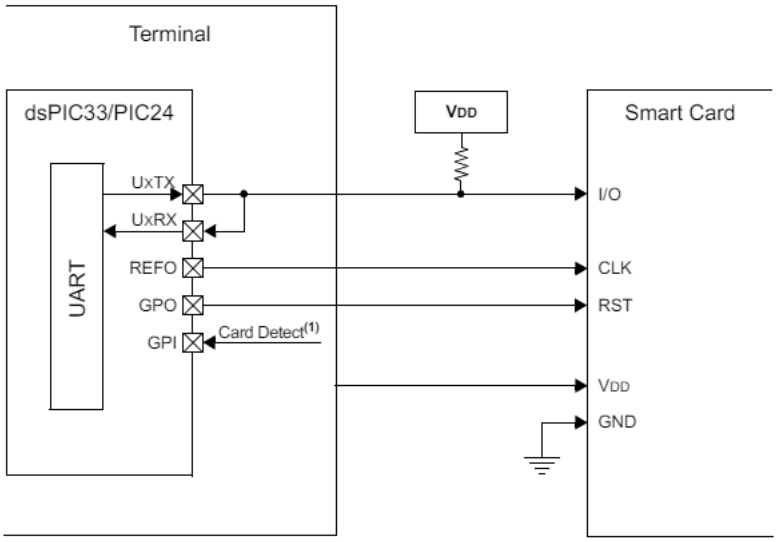
The terminal is also responsible for powering, clocking and resetting the smart card. The clock can be sourced by using the REFO output pin, and the Reset signal can be implemented with a general purpose output. The system is based on a half-duplex single wire, requiring the UART UxTX and UART UxRX pins to be shorted externally and pulled to V_{DD} with a weak pull-up.

The module can be configured to support either block (T = 1) or byte (T = 0) protocol. Block mode is set up for a predetermined message block size, whereas Byte mode transmits one byte at a time.

Upon detection of the card insertion, the terminal pulls the Reset line low to initiate a Reset sequence. The smart card responds with an Answer-to-Reset (ATR), which contains parameters used for communication details. The ATR baud rate is predetermined at the REFO clk/372. The terminal will need to be configured for this baud rate at the time the Reset pulse is sent to the smart card. Typical REFO clock rates are 1 MHz to 5 MHz. See ISO 7816 for additional details on ATR.

Note: Protocol characteristics, electrical characteristics of the smart card, Answer-To-Reset (ATR), PPS (Protocol Parameter Selection), calculation of guard time and wait times are out of the scope of this data sheet. Please refer to the licensed version of the ISO 7816-3 document for details about smart card communication.

Figure 18-15. Smart Card Subsystem



Note: Use a general purpose input to detect insertion of a smart card.

18.4.2.4.1 Protocol and Frame Details

The smart card communication scheme is based on an Elementary Time Unit (ETU) that is also the bit clock. The smart card will provide the ETU value in the ATR and the terminal is configured accordingly. A character frame consists of ten bits; a Start bit, eight data bits and a Parity bit. Depending on the mode, guard and wait times are used to separate bytes and message transitions.

The ISO 7816 specification defines two communication logic conventions: direct and inverse. Direct convention is defined as LSB first and a high state as logic one. Inverse mode is defined as MSB first

and a low on the line is interpreted as a logic high. The logic convention is set using the CONV bit (UxSCCON[3]).

Guard Time

Guard time is defined as the minimum delay between two consecutive character frames. The ISO 7816 specification defines both a Character Guard Time (CGT) and a Block Guard Time (BGT). In both T = 0 and T = 1 modes, CGT is defined as the minimum delay between the leading edges of the two consecutive characters in the same direction of transmission. Block Guard Time (BGT) for T = 1 mode only is defined as the minimum delay between the leading edges of the two consecutive characters in the opposite directions. The BGT has a standard fixed value of 22 ETUs.

Wait Time

Wait time is the maximum delay allowed between two consecutive characters transmitted by the card or an interfacing device. The Character Wait Time (CWT) is the maximum delay between the leading edges of the two consecutive characters in the block, as shown in Figure 18-16. The minimum delay is CGT. The Block Wait Time (BWT) is the maximum delay between the leading edge of the last character of the block received by the card and the leading edge of the first character of the next block transmitted by the card, as shown in Figure 18-17. BWT helps the interfacing device in detecting the unresponsive smart cards. The minimum delay is BGT.

Note: The LAST bit (UxTXB[15]) set by user software is used to automatically start the guard or wait timers, depending on the state of the module.

Figure 18-16. Character Guard and Wait Time

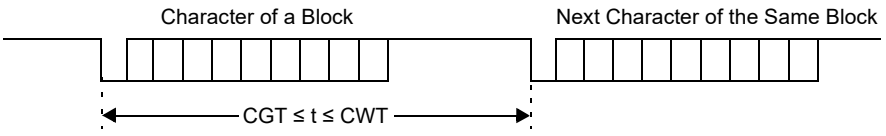
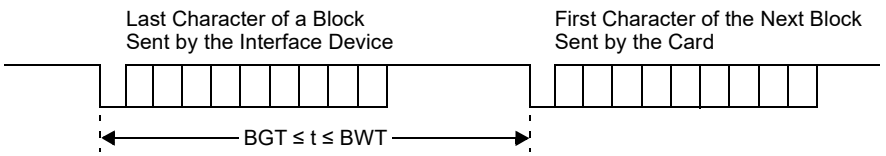


Figure 18-17. Block Guard and Wait Time

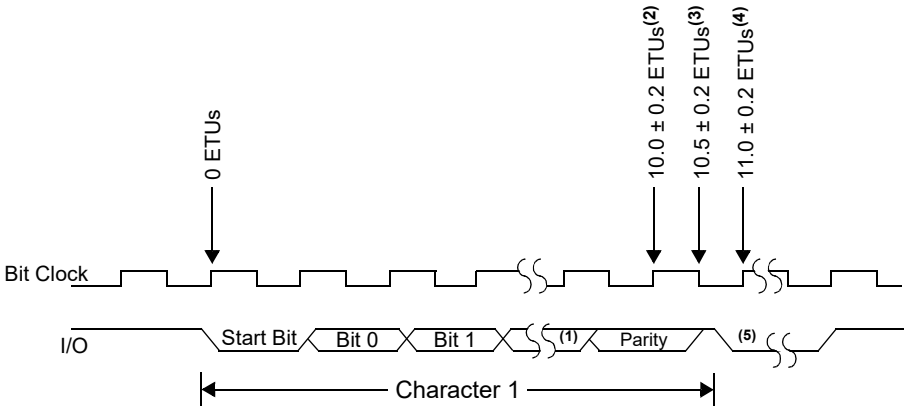


Error Detection

The transmitter is responsible for calculating the parity bit value. Parity is always even, defined as the number of logic ones and the parity bit always being an even count. The receiver also calculates the parity value and compares it to the received parity bit. If a discrepancy is found, the error is flagged by the receiver, pulling the line low for a duration defined by the T0PD bit (UxSCCON[2]).

The transmitter tests the I/O line at time 11 ± 0.2 ETUs after the leading edge of the Start bit of a character was sent. If the transmitter detects an error by detecting a low state on the I/O line, it will repeat the disputed character after a delay of at least two ETUs following the detection of the error. The number of repeats is configured with the TXRPT[1:0] bits (UxSCCON[5:4]). See Figure 18-18 for timing details in T = 0 mode.

Figure 18-18. T = 0 Character Repetition Timing



Notes:

1. 8-bit character.
2. At 10.0 ± 0.2 ETUs, the transmitter disables the driver.
3. At 10.5 ± 0.2 ETUs, the receiver sets the I/O line low if a parity error is detected.
4. At 11.0 ± 0.2 ETUs, the transmitter tests the I/O line.
5. See the T0PD bit (UxSCCON[2]) in 18.3.9. UxSCCON.

18.4.2.4.2 Smart Card Operation

Pre-ATR Initialization

The module should be configured in Receive mode prior to the Reset line being pulled low to initiate the smart card's response as follows:

1. Write the BRG register with a value corresponding to REFO/372.
2. Configure Smart Card mode by writing '0b1111' to the MODE[3:0] bits.
3. Set the ON, RXEN and TXEN bits.
4. Configure the RX interrupt watermark using the RXWM[2:0] bits (UxSTAT[26:24]).
5. Read data out of UxRXB as they become available and save for ATR processing.

Post-ATR Initialization

After the terminal has done a Reset of the Smart Card and received the setup parameters contained in the ATR, the user software can configure the module for communication as follows:

1. Disable the UART for configuration changes by clearing the ON bit.
2. Set the PRTCL (UxSCCON[1]), T0PD (UxSCCON[2]), CONV (UxSCCON[3]) and TXRPT[1:0] bits (UxSCCON[5:4]) according to ATR parameters.
3. Program the UxBRG register for the ETU defined in ATR.
4. Program guard time using Parameter 1 and set the GTCIE bit (UxSCCON[16]).
5. Program the wait time using Parameter 3 and set the WTCIE bit (UxSCCON[17]).
6. Configure the RX interrupt watermark using the RXWM[2:0] bits (UxSTAT[16:24]).
7. Set the ON, TXEN and RXEN bits.

T = 0 Protocol Communication

Transmission with T = 0:

1. Write data into UxTXB.

2. Take appropriate actions according to the ISO 7816 standard if the WTCIF, GTCIF and/or TXRPTIF bits are set.
3. Set LAST = 1 (UxTXB[15]) for the last byte of the data.
   Note: Due to the UxTX and UxRX pins being shorted, a receive interrupt will be generated on transmission of a character if enabled. It is recommended to disable receive interrupts when transmitting.

Reception with T = 0:

1. Read the UxRXB as the data become available upon a receive interrupt.
2. Take appropriate actions according to the ISO 7816 standard if the WTCIF, GTCIF and/or RXRPTIF bits are set.
   Note: For the last character, the user must ensure that the guard time is satisfied before transmitting the response. The GTC may be used for this purpose, whereas the WTC interrupt may be disabled or ignored.

T = 1 Protocol Communication

Transmission with T = 1:

1. Write data into UxTXB.
2. Program the value of the BWT to Parameter 2 and set the WTCIE bit (UxSCCON[17]).
3. Take appropriate actions according to the ISO 7816 standard if the WTCIF, GTCIF and/or TXRPTIF bits are set.
4. Set LAST = 1 (UxTXB[15]) for the last byte of the data.

Reception with T = 1:

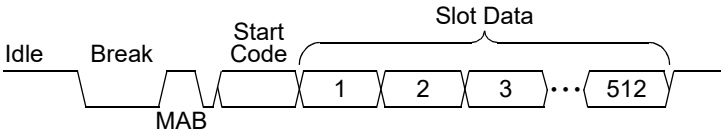
1. Read the UxRXB as the data become available upon a receive interrupt.
2. Program the value of the CWT to Parameter 3 and set the WTCIE bit (UxSCCON[17]) to '1'.
3. Take appropriate actions according to the ISO 7816 standard if the WTCIF, GTCIF and/or RXRPTIF bits are set.
   Note: For the last character, the user must ensure the guard time is satisfied before transmitting the response. The GTC may be used for this purpose, whereas the WTC interrupt may be disabled or ignored.

18.4.2.5 DMX

Digital Multiplex 512 (DMX) is a protocol used typically for stage lighting and effects. It supports a 'universe' of up to 512 channels and is typically implemented using EIA-485 at the physical layer. DMX communication is one way only, with the controller only sending messages and the client device only receiving. There is no error checking or confirmation that a command has been received. DMX operates at a baud rate of 250k with no parity and two Stop bits. A DMX message frame consists of a header and up to 512 'slots' (data bytes). A client device can be configured to accept more than one slot, given start and stop assignment values.

A DMX message frame consists of a Break, a Mark After Break (MAB), a start code and finally, the slot data bytes. The MAB is three bit times in length. The start code specifies the type of data and is typically at 0x00. Figure 18-19 shows a DMX frame.

Figure 18-19. DMX Frame



18.4.2.5.1 DMX Transmit

The following procedure is used for DMX transmit:

1. Configure the clock input and baud rate as detailed in 18.4.1. Clocking and Baud Rate Configuration for 250 kbaud.
2. Configure DMX mode by writing '0b1010' to the MODE[3:0] bits.
3. Set STP to '0b10'.
4. Configure the TX interrupt watermark using the TXWM[2:0] bits.
5. Write to Parameter 1 equal to the number of bytes – 1 (not including start code).
6. Set the ON and TXEN bits.
7. Write the start code value to UxTXB.

Writing the start code to UxTXB will transmit a 25-bit Break, MAB and start code.

1. Write slot data bytes to UxTXB.

If not all bytes defined by Parameter 1 are written, the line will return to the Idle state. Once all bytes are written, the frame is considered complete and the next write to UxTXB will send a Break for the next frame.

18.4.2.5.2 DMX Receive

The following procedure is used for DMX receive:

1. Configure the clock input and baud rate as detailed in 18.4.1. Clocking and Baud Rate Configuration for 250 kbaud.
2. Configure DMX mode by writing '0b1010' to the MODE[3:0] bits.
3. Configure the RX interrupt watermark using the RXWM[2:0] bits.
4. Set the ON bit
5. Set the RXEN bit.
6. Wait for the RXBKIF bit to set.
7. Write the byte start value – 1 to Parameter 2 (not including start code).
8. Write the byte end value – 1 to Parameter 3 (not including start code).

Once a Break is received, the UART will load the start code byte into UxRXB and generate an RX interrupt, regardless of the RX watermark setting (RXWM[2:0]). The range of bytes defined

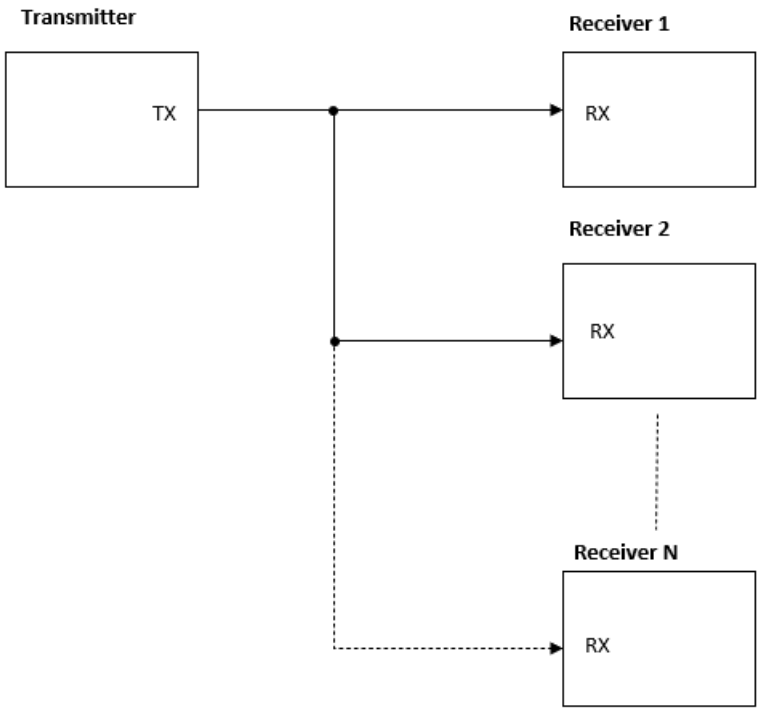
by Parameter 2 and Parameter 3 are then loaded into UxRXB, and an RX interrupt is generated according to the RXWM[2:0] bits.

18.5 Application Examples

18.5.1 UART Communication Between a Transmitter and Multiple Receiver using Address Detect Feature

A transmitter is communicating with multiple receivers as shown in Figure 18-20. When a transmitter wants to send data to any one of the receivers, it will first send the receiver's address followed by the data bytes. These data bytes will be ignored by the rest of the receivers. The mask value can also be configured so that more than one receiver can receive the same data bytes.

Figure 18-20. Address Detect Block Diagram



18.5.1.1 Transmission

Example 18-3. Address Detect Transmission

```
#include<xc.h>
#define FP 4000000
#define BAUDRATE 9600
#define BRGVAL (((FP/BAUDRATE)/16)-1)
int main(void) {
    // Configure oscillator as needed
    // Configure oscillator as needed
    _RP23R = 9;                 // Assign U1TX to RP23

    U1CONbits.MODE = 4;         // Asynchronous 9-bit UART with address detect
    U1CONbits.BRGS = 0;         // Low-Speed Mode
    U1CONbits.CLKSEL = 0;       // System_Clock/2 as Baud Clock source
    U1CONbits.STP = 0;          // 1 stop bit
    U1BRG = BRGVAL;             // BRG setting for 9600 baud rate
    U1CONbits.ON = 1;           // Enable UART
    U1CONbits.TXEN = 1;         // Enable UART TX.
    U1PAbits.P1 = 0x45;         // Write the address1 value to Parameter1
    /* Send data bytes for address1 */
    U1TXB = 'a';
    U1TXB = 'b';
    U1TXB = 'c';
    while(U1STATbits.TXMTIF == 0);
    U1PAbits.P1 = 0x55;         // Write the address2 value to Parameter1
    /* Send data bytes for address2 */
    U1TXB = 'd';
    U1TXB = 'e';
    U1TXB = 'f';
    while(U1STATbits.TXMTIF == 0);
    while(1)
    {
    }
    return 0;
}
```

18.5.1.2 Reception

Example 18-4. Address Detect Reception

```
#include<xc.h>
#define FP 4000000
#define BAUDRATE 9600
#define BRGVAL (((FP/BAUDRATE)/16)-1)
uint16_t ReceivedChar[10];
uint8_t i=0;

int main(void) {
                                // Configure oscillator as needed
                                // Configure oscillator as needed
    ANSELB = 0;
    _U1RXR = 24; // Assign U1RX to RP24

    U1CONbits.MODE = 4;         // Asynchronous 9-bit UART with address detect
    U1CONbits.CLKSEL = 0;       // System_Clock/2 as Baud Clock source
    U1CONbits.BRGS = 0;         // Low-Speed Mode
    U1CONbits.STP = 0;          // 1 stop bit
    U1BRG = BRGVAL;             // BRG setting for 9600 baud rate
    U1PAbits.P2 = 0x45;         // Write parameter 2 with Address to match
    U1PBbits.P3 = 0xFF;         // Write parameter 3 register with mask value
    U1STATbits.RXWM = 0;        // Interrupt when there is one word or more in the buffer
    _U1RXIE = 1;                // Enable Receive interrupt
    U1CONbits.ON = 1;           // Enable UART
    U1CONbits.RXEN = 1;         // Enable UART RX

    while(1)
    {
    }
    return 0;
}
void __attribute__((interrupt, no_auto_psv)) _U1RXInterrupt(void)
{
    _U1RXIF = 0; // Clear RX interrupt flag
    while(U1STATbits.RXBE == 0)    // Check if RX buffer has data to read
    {
        ReceivedChar[i++] = U1RXB; // Read a character from RX buffer
    }
}
```

18.6 Interrupts

The UART has four separate interrupts. To determine which event has caused the interrupt, the associated flag needs to be read and evaluated. All interrupt sources are listed in Table 18-8.

Table 18-8. Interrupts

Interrupt Type	Condition	Flag
TX	Number of Empty Slots in UxTXB Defined by TXWM[2:0] bits	TXIF(1)
RX	Number of Words in UxRXB Defined by RXWM[2:0] bits Address Match	RXIF(1) PERIF
Event	Auto-Baud Complete RX Break Received	ABDIF RXBKIF
	Wake Event (line is high-to-low)	WUIF
	Smart Card Guard Time Counter Match	GTCIF
	Smart Card Block Time Counter Match	BTCIF

.....continued

Interrupt Type	Condition	Flag
Error	Parity Error Framing Error	PERIF FERIF
	Transmit Collision	TXCIF
	Transmit Shift Register Empty	TXMTIF
	RX Buffer Overflow	RXFOIF
	Auto-Baud Rollover	ABDOVF
	Checksum Error (LIN mode only)	CERIF
	Smart Card Receive Repeat	RXRPTIF
	Smart Card Transmit Repeat	TXRPTIF
	Smart Card Waiting Time Counter Match	WTCIF

18.6.1 Interrupt Watermarks

Both TX and RX interrupt frequency can be configured using the watermark setting. For transmit, the TXWM[2:0] bits setting allows the TX interrupt frequency to be based on the number of empty slots left in the TX buffer (UxTXB). When the transmitter is initially enabled (TXEN = 1), the TX interrupt bit will be set on the condition that the module is enabled (ON = 1) as well. The user should clear the TX interrupt bit in the ISR. For receive, the RXWM[2:0] bits setting allows the RX interrupt frequency to be based on how many bytes are in the RX buffer (UxRXB). By default, an RX interrupt will be generated when the RX buffer has at least one byte in it. The receive watermark interrupt will not be set if PERIF, FERIF, or TXCIF are set and the corresponding PERIE, FERIE or TXCIE bits are set.

18.7 Power-Saving Modes

The UART provides support in power-saving modes, including the capability to run in Sleep and Idle modes. If a transmission or reception is in progress, and a power save command is executed, the operation will abort. SFR data, including UxCON, UxSTAT, UxBRG and the RX and TX buffers, will retain their values upon a wake condition and do not need to be reinitialized.

18.7.1 Sleep

When a device enters Sleep mode, the system clock used by the core processor and peripherals is halted. To use run in Sleep mode, a clock source other than the system clock must be selected and the SLPEN bit (UxCON[31]) is set. Clock sources are device-specific (see 18.4.1. Clocking and Baud Rate Configuration for details). This allows the UART to request the selected clock source and keep it active. The UART has the ability to continue transmitting the contents of UxTXB, receiving data and storing them in UxRXB.

The UART can also wake the processor from Sleep mode (when SPLEN = 0) on the detection of an incoming byte, including Break and Sync characters used for auto-baud. To enable the wake feature, set the WUE bit (UxCON[12]). When a wake from Sleep occurs, the WUIF (UxUIR[7]) bit is set and an event interrupt is generated, effectively waking the processor. If auto-baud is desired on wake, set the ABDEN bit before executing a `SLEEP` command.

18.7.2 Idle

In Idle mode, the core processor is halted. However, the peripherals, including the UART, continue to run. To also halt the UART in Idle, the UART Stop in Idle Mode bit, SIDL (UxCON[13]), can be set.

19. Serial Peripheral Interface (SPI)

The Serial Peripheral Interface (SPI) module is a synchronous serial interface useful for communicating with external peripherals and other microcontroller devices. These peripheral devices may be a serial EEPROM, shift register, display driver, Analog-to-Digital Converter (ADC) or an audio codec. The dsPIC33 family SPI module is compatible with Motorola® SPI and SIOP interfaces.

Some of the key features of this module are:

- Host and Client Modes Support
- Four Different Clock Formats
- Framed SPI Protocol Support
- Standard and Enhanced Buffering Modes
- User-Configurable 8-bit, 16-bit and 32-bit Data Width
- Two Separate Shift Registers for Transmission and Reception
- SPIx Receive and Transmit Buffers are FIFO Buffers in Enhanced Buffering Mode
- User-Configurable Variable Data Width, From 2 to 32-bit
- Programmable Interrupt Event on Every 8-bit, 16-bit and 32-bit Data Transfer
- Audio Protocol Interface Mode

dsPIC33 devices support audio codec serial protocols, such as Inter-IC Sound

(I^2S), Left Justified, Right Justified and PCM/DSP modes for 16, 24 and 32-bit audio data.

19.1 Device-Specific Information

Table 19-1. SPI Summary Table

SPI Module Instances	Max Clock Frequency	Peripheral Bus Speed	Clock Source
3	See Electrical Characteristics	Standard	See Table 19-2

Table 19-2. MCLKEN Host Clock Enable bit

Value	Description
1	CLKGEN9
0	Standard Speed Peripheral Clock

Table 19-3. Device FIFO Depth

Default	4-bytes
Note: 32-bit data equates to a buffer depth of X/4 = 1.	

19.2 Architectural Overview

The SPI module is a synchronous serial communication interface used for short distances in embedded systems.

SPI devices communicate in Full Duplex mode using a host/client architecture with a single host. The host device originates the frame for reading and writing. Multiple Client devices may be supported through selection with individual chip select (SSx) lines.

The SPI serial interface consists of four pins:

- SDIx: Serial Data Input
- SDOx: Serial Data Output

- SCKx: Shift Clock Input or Output
- SSx: Active-Low Client Select or Frame Synchronization I/O Pulse

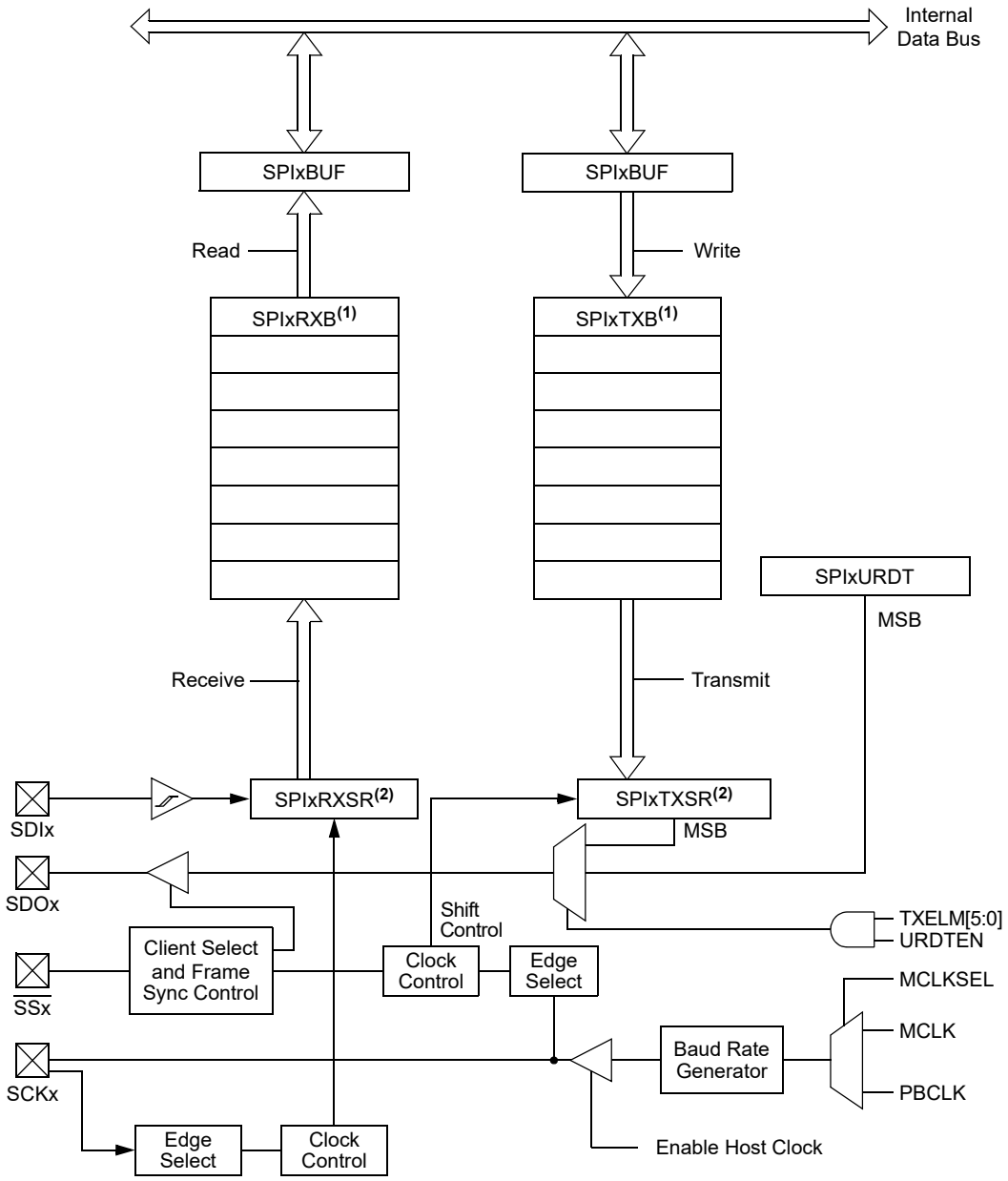
During each SPI clock cycle, a full-duplex data transmission occurs. The host sends a bit on the SDO line and the client reads the bit, while the client sends a bit on the SDI line and the host reads the bit. This sequence is the same for one-directional data transfer.

The SPI module offers the following operating modes:

- 8-bit, 16-bit and 32-bit Data Transmission modes
- 8-bit, 16-bit and 32-bit Data Reception modes
- Host and Client modes
- Framed SPI modes
- Audio Protocol Interface mode

Figure 19-1 shows the block diagram of the SPI module.

Figure 19-1. SPIx Module Block Diagram



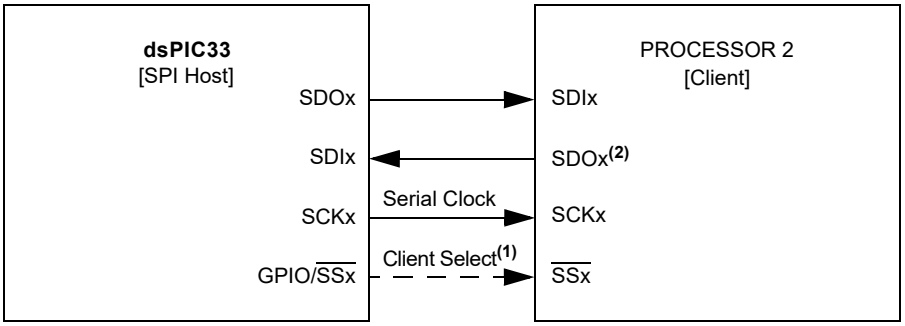
Note:

1. The SPIx Receive Buffer (SPIxRXB) and SPIx Transmit Buffer (SPIxTXB) registers are accessed through the SPIxBUF register and are multi-element FIFO buffers in Enhanced Buffer mode.
2. The SPIx Shift register is not directly accessible by application software.

19.2.1 Normal Mode SPI Operation

In Normal mode operation, the SPI host controls the generation of the serial clock. The number of output clock pulses corresponds to the transfer data width: 8, 16 or 32 bits, or depending upon variable data width configuration, from 2 to 32-bit. Figure 19-2 and Figure 19-3 illustrate SPI host-to-client and client-to-host device connections.

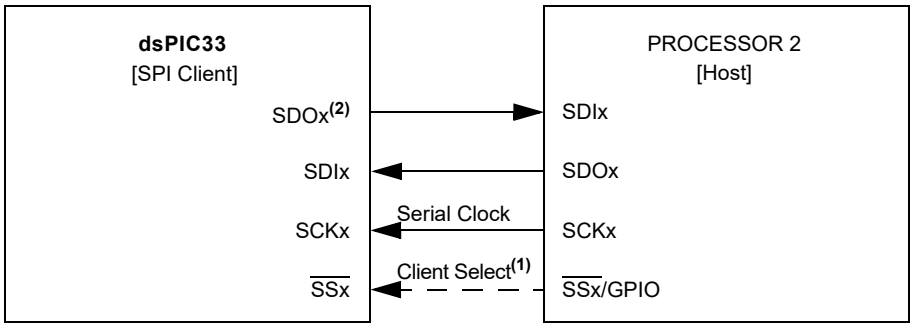
Figure 19-2. Typical SPIx Host-to-Client Device Connection Diagram



Notes:

1. In Normal mode, the usage of the Client Select pin ($\overline{SSx}$) is optional.
2. Control of the SDOx pin can be disabled for Receive Only modes.

Figure 19-3. Typical SPIx Client-to-Host Device Connection Diagram



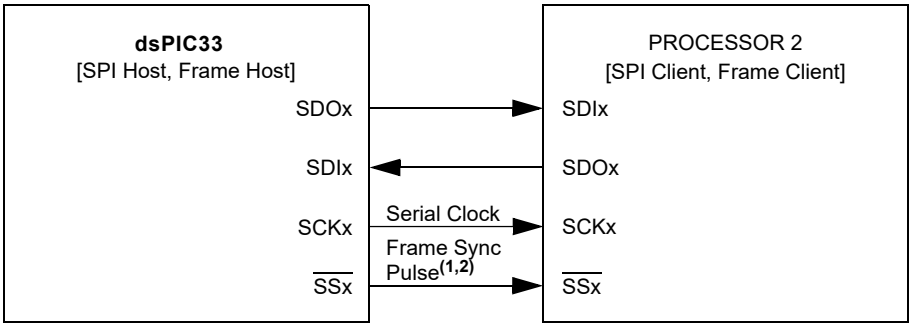
Notes:

1. In Normal mode, the usage of the Client Select pin ($\overline{SSx}$) is optional.
2. The control of the SDOx pin can be disabled for Receive Only modes.

19.2.2 Framed Mode SPI Operation

In Framed mode operation, the frame host controls the generation of the frame synchronization pulse. The SPI clock is still generated by the SPI host and is continuously running. Figure 19-4 and Figure 19-5 illustrate SPI frame host and frame client device connections.

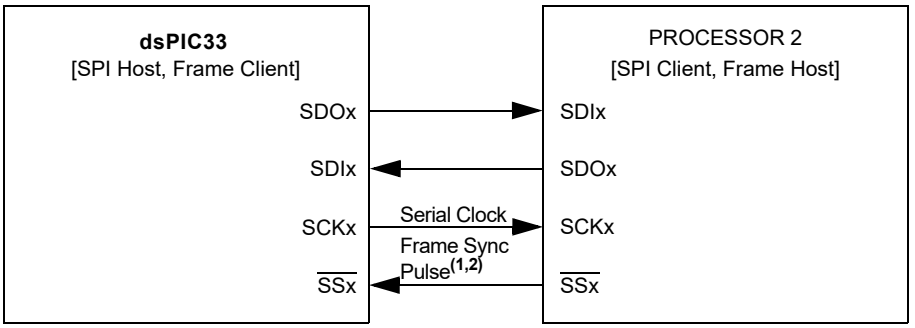
Figure 19-4. Typical SPIx Host, Frame Host Connection Diagram



Notes:

1. In Framed SPI mode, the $\overline{SSx}$ pin is used to transmit/receive the Frame Synchronization pulse.
2. Framed SPI mode requires the use of all four pins (i.e., using the $\overline{SSx}$ pin is not optional).

Figure 19-5. Typical SPIx Host, Frame Client Connection Diagram



Notes:

1. In Framed SPI mode, the $\overline{SSx}$ pin is used to transmit/receive the Frame Synchronization pulse.
2. Framed SPI mode requires the use of all four pins (i.e., using the $\overline{SSx}$ pin is not optional).

19.2.3 Audio Protocol Interface Mode

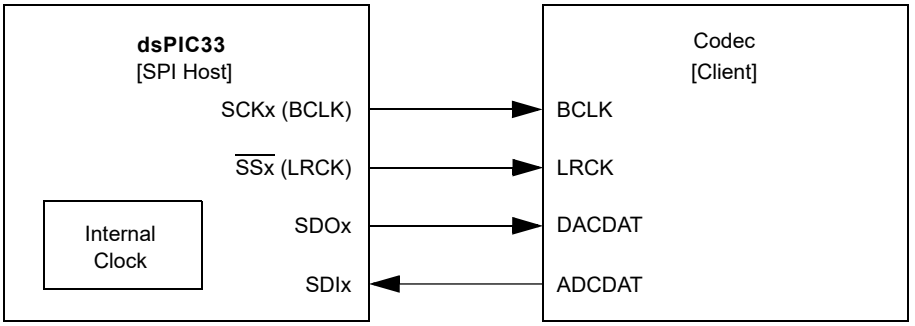
The SPI module supports audio interfaces and is configurable for a variety of protocols including:

- PCM/DSP mode
- Right Justified mode
- Left Justified mode
- I^2S mode

19.2.3.1 SPI in Audio Host Mode Connected to a Codec Client

Figure 19-6 shows the Bit Clock (BCLK) and Left/Right Channel Clock (LRCK) as generated by the dsPIC33 SPI module.

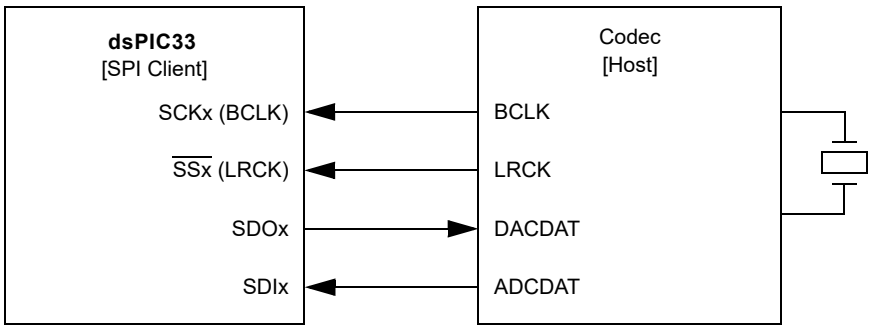
Figure 19-6. Host Generating its Own Clock – Output BCLK and LRCK



19.2.3.2 SPI in Audio Client Mode Connected to a Codec Host

Figure 19-7 shows the BCLK and LRCK as generated by the codec host.

Figure 19-7. Codec Device as Host Generates Required Clock via External Crystal



19.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1800	SPI1CON1	31:24	AUDEN	SPISGNEXT	IGNROV	IGNTUR	AUDMONO	URDTEN	AUDMOD[1:0]	
		23:16	FRMEN	FRMSYNC	FRMPOL	MSSEN	FRMSYPW	FRMCNT[2:0]		
		15:8	ON		SIDL	DISSDO	MODE[32,16]		SMP	CKE
		7:0	SSEN	CKP	MSTEN	DISSDI	DISSCK	MCLKEN	SPIFE	ENHBUF
0x1804	SPI1CON2	31:24								
		23:16								
		15:8								
		7:0				WLENGTH[4:0]				
0x1808	SPI1STAT	31:24						RXELM[2:0]		
		23:16						TXELM[2:0]		
		15:8				FRMERR	BUSY			SPITUR
		7:0	SRMT	SPIROV	SPIRBE		SPITBE		SPITBF	SPIRBF
0x180C	SPI1BUF	31:24	DATA[31:24]							
		23:16	DATA[23:16]							
		15:8	DATA[15:8]							
		7:0	DATA[7:0]							
0x1810	SPI1BRG	31:24								
		23:16								
		15:8				BRG[12:8]				
		7:0	BRG[7:0]							
0x1814	SPI1IMSK	31:24	RXWIEN					RXMSK[2:0]		
		23:16	TXWIEN					TXMSK[2:0]		
		15:8				FRMERREN	BUSYEN			SPITUREN
		7:0	SRMTEN	SPIROVEN	SPIRBEN		SPITBEN		SPITBFEN	SPIRBFEN
0x1818	SPI1URDT	31:24	SPI1URDT[31:24]							
		23:16	SPI1URDT[23:16]							
		15:8	SPI1URDT[15:8]							
		7:0	SPI1URDT[7:0]							
0x181C ... 0x181F	Reserved									
0x1820	SPI2CON1	31:24	AUDEN	SPISGNEXT	IGNROV	IGNTUR	AUDMONO	URDTEN	AUDMOD[1:0]	
		23:16	FRMEN	FRMSYNC	FRMPOL	MSSEN	FRMSYPW	FRMCNT[2:0]		
		15:8	ON		SIDL	DISSDO	MODE[32,16]		SMP	CKE
		7:0	SSEN	CKP	MSTEN	DISSDI	DISSCK	MCLKEN	SPIFE	ENHBUF
0x1824	SPI2CON2	31:24								
		23:16								
		15:8								
		7:0				WLENGTH[4:0]				
0x1828	SPI2STAT	31:24						RXELM[2:0]		
		23:16						TXELM[2:0]		
		15:8				FRMERR	BUSY			SPITUR
		7:0	SRMT	SPIROV	SPIRBE		SPITBE		SPITBF	SPIRBF
0x182C	SPI2BUF	31:24	DATA[31:24]							
		23:16	DATA[23:16]							
		15:8	DATA[15:8]							
		7:0	DATA[7:0]							
0x1830	SPI2BRG	31:24								
		23:16								
		15:8				BRG[12:8]				
		7:0	BRG[7:0]							
0x1834	SPI2IMSK	31:24	RXWIEN					RXMSK[2:0]		
		23:16	TXWIEN					TXMSK[2:0]		
		15:8				FRMERREN	BUSYEN			SPITUREN
		7:0	SRMTEN	SPIROVEN	SPIRBEN		SPITBEN		SPITBFEN	SPIRBFEN
0x1838	SPI2URDT	31:24	SPI2URDT[31:24]							
		23:16	SPI2URDT[23:16]							
		15:8	SPI2URDT[15:8]							
		7:0	SPI2URDT[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x183C ... 0x183F	Reserved									
0x1840	SPI3CON1	31:24	AUDEN	SPISGNEXT	IGNROV	IGNTUR	AUDMONO	URDTEN	AUDMOD[1:0]	
		23:16	FRMEN	FRMSYNC	FRMPOL	MSSEN	FRMSYPW	FRMCNT[2:0]		
		15:8	ON		SIDL	DISSDO	MODE[32,16]		SMP	CKE
		7:0	SSEN	CKP	MSTEN	DISSDI	DISSCK	MCLKEN	SPIFE	ENHBUF
0x1844	SPI3CON2	31:24								
		23:16								
		15:8								
		7:0				WLENGTH[4:0]				
0x1848	SPI3STAT	31:24						RXELM[2:0]		
		23:16						TXELM[2:0]		
		15:8				FRMERR	BUSY			SPITUR
		7:0	SRMT	SPIROV	SPIRBE		SPITBE		SPITBF	SPIRBF
0x184C	SPI3BUF	31:24	DATA[31:24]							
		23:16	DATA[23:16]							
		15:8	DATA[15:8]							
		7:0	DATA[7:0]							
0x1850	SPI3BRG	31:24								
		23:16								
		15:8				BRG[12:8]				
		7:0	BRG[7:0]							
0x1854	SPI3IMSK	31:24	RXWIEN					RXMSK[2:0]		
		23:16	TXWIEN					TXMSK[2:0]		
		15:8				FRMERREN	BUSYEN			SPITUREN
		7:0	SRMTEN	SPIROVEN	SPIRBEN		SPITBEN		SPITBFEN	SPIRBFEN
0x1858	SPI3URDT	31:24	SPI3URDT[31:24]							
		23:16	SPI3URDT[23:16]							
		15:8	SPI3URDT[15:8]							
		7:0	SPI3URDT[7:0]							

19.3.1 SPIx Control Register 1

Name: SPIxCON1
Offset: 0x1800, 0x1820, 0x1840

Notes:
1. When AUDEN = 1, this module functions as if CKE = 0, regardless of its actual value.
2. When FRMEN = 1, SSEN is not used.
3. MCLKEN can only be written when the ON bit = 0.
4. This channel is not meaningful for DSP/PCM mode as LRC follows FRMSYPW.
5. SPI operates with DMA in Standard Buffer mode only, ENHBUF = 0.
6. SPISGNEXT function is available for 8, 16 and 32-bit data length transfers only.

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	AUDEN	SPISGNEXT	IGNROV	IGNTUR	AUDMONO	URDTEN	AUDMOD[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FRMEN	FRMSYNC	FRMPOL	MSSEN	FRMSYPW	FRMCNT[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	DISSDO	MODE[32,16]		SMP	CKE
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SSEN	CKP	MSTEN	DISSDI	DISSCK	MCLKEN	SPIFE	ENHBUF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – AUDEN Audio Codec Support Enable bit(1)

Value	Description
1	Audio protocol is enabled; MSTEN controls the direction of both SCKx and frame (a.k.a. LRC), and this module functions as if FRMEN = 1, FRMSYNC = MSTEN, FRMCNT[2:0] = 001 and SMP = 0, regardless of their actual values
0	Audio protocol is disabled

Bit 30 – SPISGNEXT SPIx Sign-Extend RX FIFO Read Data Enable bit(6)

Value	Description
1	Data from RX FIFO is sign-extended (upper unused bits should replicate MSb of the received data)
0	Data from RX FIFO is not sign-extended (upper unused bits are always 1'b0)

Bit 29 – IGNROV Ignore Receive Overflow bit

Value	Description
1	A Receive Overflow (ROV) is NOT a critical error; during ROV, data in the FIFO is not overwritten by the receive data
0	A ROV is a critical error that stops SPI operation

Bit 28 – IGNTUR Ignore Transmit Underrun bit

Value	Description
1	A Transmit Underrun (TUR) is NOT a critical error and data indicated by URDTEN is transmitted until the SPIxTXB is not empty
0	A TUR is a critical error that stops SPI operation

Bit 27 – AUDMONO Audio Data Format Transmit bit(2)

Value	Description
1	Audio data is mono (i.e., each data word is transmitted on both left and right channels)
0	Audio data is stereo

Bit 26 – URDTEN Transmit Underrun Data Enable bit(3)

Value	Description
1	Transmits data out of SPIxURDT register during Transmit Underrun conditions
0	Transmits the last received data during Transmit Underrun conditions

Bits 25:24 – AUDMOD[1:0] Audio Protocol Mode Selection bits(4)

Value	Description
11	PCM/DSP mode
10	Right Justified mode: This module functions as if SPIFE = 1, regardless of its actual value
01	Left Justified mode: This module functions as if SPIFE = 1, regardless of its actual value
00	I^2S mode: This module functions as if SPIFE = 0, regardless of its actual value

Bit 23 – FRMEN Framed SPIx Support bit

Value	Description
1	Framed SPIx support is enabled ($\overline{SSx}$ pin is used as the FSYNC input/output)
0	Framed SPIx support is disabled

Bit 22 – FRMSYNC Frame Sync Pulse Direction Control bit

Value	Description
1	Frame Sync pulse input (client)
0	Frame Sync pulse output (host)

Bit 21 – FRMPOL Frame Sync/Client Select Polarity bit

Value	Description
1	Frame Sync pulse/Client Select is active-high
0	Frame Sync pulse/Client Select is active-low

Bit 20 – MSSEN Host Mode Client Select Enable bit

Value	Description
1	SPIx Client Select support is enabled with polarity determined by FRMPOL ($\overline{SSx}$ pin is automatically driven during transmission in Host mode)
0	Client Select SPIx support is disabled ($\overline{SSx}$ pin will be controlled by port I/O)

Bit 19 – FRMSYPW Frame Sync Pulse-Width bit

Value	Description
1	Frame Sync pulse is one serial word length wide (as defined by MODE[32,16]/WLENGTH[4:0])
0	Frame Sync pulse is one clock (SCKx) wide

Bits 18:16 – FRMCNT[2:0] Frame Sync Pulse Counter bits
Controls the number of serial words per Sync pulse.

Value	Description
111	Reserved
110	Reserved
101	Generate/Receive a Frame Sync pulse on every 32 serial words
100	Generate/Receive a Frame Sync pulse on every 16 serial words
011	Generate/Receive a Frame Sync pulse on every 8 serial words
010	Generate/Receive a Frame Sync pulse on every 4 serial words
001	Generate/Receive a Frame Sync pulse on every 2 serial words (value used by audio protocols)
000	Generate/Receive a Frame Sync pulse on each serial word

Bit 15 – ON SPIx On bit

Value	Description
1	Enables module
0	Turns off and resets module, disables clocks, disables interrupt event generation, allows SFR modifications

Bit 13 – SIDL SPIx Stop in Idle Mode bit

Value	Description
1	Halts in CPU Idle mode
0	Continues to operate in CPU Idle mode

Bit 12 – DISSDO Disable SDOx Output Port bit

Value	Description
1	SDOx pin is not used by the module; pin is controlled by port function
0	SDOx pin is controlled by the module

Bits 11:10 – MODE[32,16] Serial Word Length bits(1,4)

MODE32	MODE16	AUDEN	Communication
1	x	0	32-bit
0	1		16-bit
0	0		8-bit
1	1	1	24-bit Data, 32-Bit FIFO, 32-Bit Channel/64-Bit Frame
1	0		32-bit Data, 32-Bit FIFO, 32-Bit Channel/64-Bit Frame
0	1		16-bit Data, 16-Bit FIFO, 32-Bit Channel/64-Bit Frame
0	0		16-bit FIFO, 16-Bit Channel/32-Bit Frame

Bit 9 – SMP SPIx Data Input Sample Phase bit

Client Mode:

Input data is always sampled at the middle of data output time, regardless of the SMP setting.

Host Mode:

Value	Description
1	Input data is sampled at the end of data output time
0	Input data is sampled at the middle of data output time

Bit 8 – CKE SPIx Clock Edge Select bit(1)

Value	Description
1	Transmit happens on transition from active clock state to Idle clock state
0	Transmit happens on transition from Idle clock state to active clock state

Bit 7 – SSEN Client Select Enable bit (Client mode)(2)

Value	Description
1	$\overline{SSx}$ pin is used by the macro in Client mode; $\overline{SSx}$ pin is used as the Client Select input

Value	Description
0	$\overline{SSx}$ pin is not used by the macro ($\overline{SSx}$ pin will be controlled by the port I/O)

Bit 6 – CKP Clock Polarity Select bit

Value	Description
1	Idle state for clock is a high level; active state is a low level
0	Idle state for clock is a low level; active state is a high level

Bit 5 – MSTEN Host Mode Enable bit

Value	Description
1	Host mode
0	Client mode

Bit 4 – DISSDI Disable SDIx Input Port bit

Value	Description
1	SDIx pin is not used by the module; pin is controlled by port function
0	SDIx pin is controlled by the module

Bit 3 – DISSCK Disable SCKx Output Port bit

Value	Description
1	SCKx pin is not used by the module; pin is controlled by port function
0	SCKx pin is controlled by the module

Bit 2 – MCLKEN Host Clock Enable bit(3)
See Table 19-2.

Bit 1 – SPIFE Frame Sync Pulse Edge Select bit

Value	Description
1	Frame Sync pulse (Idle-to-active edge) coincides with the first bit clock
0	Frame Sync pulse (Idle-to-active edge) precedes the first bit clock

Bit 0 – ENHBUF Enhanced Buffer Enable bit(5)

Value	Description
1	Enhanced Buffer mode is enabled
0	Enhanced Buffer mode is disabled

19.3.2 SPIx Control Register 2

Name: SPIxCON2
Offset: 0x1804, 0x1824, 0x1844

Notes:

1. These bits are effective when AUDEN = 0 only.
2. Varying the length by changing these bits does not affect the depth of the TX/RX FIFO.

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				WLENGTH[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 4:0 – WLENGTH[4:0] Variable Word Length bits(1,2)

Value	Description
11111	32-bit data
11110	31-bit data
11101	30-bit data
11100	29-bit data
11011	28-bit data
11010	27-bit data
11001	26-bit data
11000	25-bit data
10111	24-bit data
10110	23-bit data
10101	22-bit data
10100	21-bit data
10011	20-bit data
10010	19-bit data
10001	18-bit data
10000	17-bit data
01111	16-bit data
01110	15-bit data
01101	14-bit data
01100	13-bit data
01011	12-bit data

Value	Description
01010	11-bit data
01001	10-bit data
01000	9-bit data
00111	8-bit data
00110	7-bit data
00101	6-bit data
00100	5-bit data
00011	4-bit data
00010	3-bit data
00001	2-bit data
00000	See MODE[32,16] bits in SPIxCON1[11:10]

19.3.3 SPIx Status Register

Name: SPIxSTAT
Offset: 0x1808, 0x1828, 0x1848

Note:

1. SPITUR is cleared when ON = 0. When IGNTUR = 1, SPITUR provides dynamic status of the Transmit Underrun condition, but does not stop RX/TX operation and does not need to be cleared by software.

Legend: C = Clearable bit, HS = Hardware Settable bit, HSC = Hardware Settable/Clearable bit

Bit	31	30	29	28	27	26	25	24
						RXELM[2:0]		
Access						R/HS	R/HS	R/HS
Reset						0	0	0

Bit	23	22	21	20	19	18	17	16
						TXELM[2:0]		
Access						R/HSC	R/HSC	R/HSC
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
				FRMERR	BUSY			SPITUR
Access				R/C/HS	R/HSC			R/HSC
Reset				0	0			0

Bit	7	6	5	4	3	2	1	0
	SRMT	SPIROV	SPIRBE		SPITBE		SPITBF	SPIRBF
Access	R/HSC	R/C/HS	R/HSC		R/HSC		R/HSC	R/HSC
Reset	0	0	1		1		0	0

Bits 26:24 – RXELM[2:0] Receive Buffer Element Count bits (valid in Enhanced Buffer mode)

Bits 18:16 – TXELM[2:0] Transmit Buffer Element Count bits (valid in Enhanced Buffer mode)

Bit 12 – FRMERR SPIx Frame Error Status bit

Value	Description
1	Frame error is detected
0	No frame error is detected

Bit 11 – BUSY SPIx Activity Status bit

Value	Description
1	Module is currently busy with some transactions
0	No ongoing transactions (at time of read)

Bit 8 – SPITUR SPIx Transmit Underrun Status bit(1)

Value	Description
1	Transmit buffer has encountered a Transmit Underrun condition
0	Transmit buffer does not have a Transmit Underrun condition

Bit 7 – SRMT Shift Register Empty Status bit

Value	Description
1	No current or pending transactions (i.e., neither SPIxTXB or SPIxTXSR contains data to transmit)
0	Current or pending transactions

Bit 6 – SPIROV SPIx Receive Overflow Status bit

Value	Description
1	A new byte/half-word/word has been completely received when the SPIxRXB was full
0	No overflow

Bit 5 – SPIRBE SPIx RX Buffer Empty Status bit

Standard Buffer Mode:
Automatically set in hardware when SPIxBUF is read from, reading SPIxRXB.
Automatically cleared in hardware when SPIx transfers data from SPIxRXSR to SPIxRXB.
Enhanced Buffer Mode:
Indicates RXELM[2:0] = 00.

Value	Description
1	RX buffer is empty
0	RX buffer is not empty

Bit 3 – SPITBE SPIx Transmit Buffer Empty Status bit

Standard Buffer Mode:
Automatically set in hardware when SPIx transfers data from SPIxTXB to SPIxTXSR.
Automatically cleared in hardware when SPIxBUF is written, loading SPIxTXB.
Enhanced Buffer Mode:
Indicates TXELM[2:0] = 00.

Value	Description
1	SPIxTXB is empty
0	SPIxTXB is not empty

Bit 1 – SPITBF SPIx Transmit Buffer Full Status bit

Standard Buffer Mode:
Automatically set in hardware when SPIxBUF is written, loading SPIxTXB.
Automatically cleared in hardware when SPIx transfers data from SPIxTXB to SPIxTXSR.
Enhanced Buffer Mode:
Indicates TXELM[2:0] = 11.

Value	Description
1	SPIxTXB is full
0	SPIxTXB not full

Bit 0 – SPIRBF SPIx Receive Buffer Full Status bit

Standard Buffer Mode:
Automatically set in hardware when SPIx transfers data from SPIxRXSR to SPIxRXB.
Automatically cleared in hardware when SPIxBUF is read from, reading SPIxRXB.
Enhanced Buffer Mode:
Indicates RXELM[2:0] = 11.

Value	Description
1	SPIxRXB is full
0	SPIxRXB is not full

19.3.4 SPI Buffer Register

Name: SPIxBUF
Offset: 0x180C, 0x182C, 0x184C

C
Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA[31:0] SPIx FIFO Data bits
Table 19-4 summarizes the valid data field for possible values of MODE32, MODE16 and WLENGTH[4:0] bits.

Table 19-4. MODE32, MODE16 and WLENGTH[4:0] Data Fields

MODE32	MODE16	WLENGTH[4:0]	COMMUNICATION	Valid Data Field
1	x	0	32-bit	DATA[31:0]
0	1	0	16-bit	DATA[15:0]
0	0	0	8-bit	DATA[07:0]
x	x	16 < N < 31	(N+1)-bit	DATA[31:(31-N)]
x	x	8 < N < 15	(N+1)-bit	DATA[15:(15-N)]
x	x	1 < N < 7	(N+1)-bit	DATA[07:(07-N)]

19.3.5 SPIx Baud Rate Generator Register

Name: SPIxBRG
Offset: 0x1810, 0x1830, 0x1850

Note:

1. Changing the BRG value when ON = 1 causes undefined behavior.

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				BRG[12:8]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BRG[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 12:0 – BRG[12:0] SPI Baud Rate Generator Divisor bits(1)

19.3.6 SPIx Interrupt Mask Register

Name: SPIxIMSK
Offset: 0x1814, 0x1834, 0x1854

Note:

1. Mask values higher than Value 4 are not valid. The module will not trigger a match for any value in this case.

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	RXWIEN					RXMSK[2:0]		
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0

Bit	23	22	21	20	19	18	17	16
	TXWIEN					TXMSK[2:0]		
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0

Bit	15	14	13	12	11	10	9	8
				FRMERREN	BUSYEN			SPITUREN
Access				R/W	R/W			R/W
Reset				0	0			0

Bit	7	6	5	4	3	2	1	0
	SRMTEN	SPIROVEN	SPIRBEN		SPITBEN		SPITBFEN	SPIRBFEN
Access	R/W	R/W	R/W		R/W		R/W	R/W
Reset	0	0	0		0		0	0

Bit 31 – RXWIEN Receive Watermark Interrupt Enable bit

Value	Description
1	Triggers receive buffer element watermark interrupt when RXMSK[2:0] ≤ RXELM[2:0]
0	Disables receive buffer element watermark interrupt

Bits 26:24 – RXMSK[2:0] RX Buffer Mask bits(1)
RX mask bits; used in conjunction with the RXWIEN bit.

Bit 23 – TXWIEN Transmit Watermark Interrupt Enable bit

Value	Description
1	Triggers transmit buffer element watermark interrupt when TXMSK[2:0] ≤ TXELM[2:0]
0	Disables transmit buffer element watermark interrupt

Bits 18:16 – TXMSK[2:0] TX Buffer Mask bits(1)
TX mask bits; used in conjunction with the TXWIEN bit.

Bit 12 – FRMERREN Enable Interrupt Events via FRMERR bit

Value	Description
1	Frame error generates an interrupt event
0	Frame error does not generate an interrupt event

Bit 11 – BUSYEN Enable Interrupt Events via SPIBUSY bit

Value	Description
1	BUSY generates an interrupt event
0	BUSY does not generate an interrupt event

Bit 8 – SPITUREN Enable Interrupt Events via SPITUR bit

Value	Description
1	Transmit Underrun (TUR) generates an interrupt event
0	Transmit Underrun does not generate an interrupt event

Bit 7 – SRMTEN Enable Interrupt Events via SRMT bit

Value	Description
1	Shift Register Empty (SRMT) generates interrupt events
0	Shift Register Empty does not generate interrupt events

Bit 6 – SPIROVEN Enable Interrupt Events via SPIROV bit

Value	Description
1	SPIx Receive Overflow (ROV) generates an interrupt event
0	SPIx Receive Overflow does not generate an interrupt event

Bit 5 – SPIRBEN Enable Interrupt Events via SPIRBE bit

Value	Description
1	SPIx RX buffer empty generates an interrupt event
0	SPIx RX buffer empty does not generate an interrupt event

Bit 3 – SPITBEN Enable Interrupt Events via SPITBE bit

Value	Description
1	SPIx transmit buffer empty generates an interrupt event
0	SPIx transmit buffer empty does not generate an interrupt event

Bit 1 – SPITBFEN Enable Interrupt Events via SPITBF bit

Value	Description
1	SPIx transmit buffer full generates an interrupt event
0	SPIx transmit buffer full does not generate an interrupt event

Bit 0 – SPIRBFEN Enable Interrupt Events via SPIRBF bit

Value	Description
1	SPIx receive buffer full generates an interrupt event
0	SPIx receive buffer full does not generate an interrupt event

19.3.7 SPIx Underrun Data Register

Name: SPIxURDT
Offset: 0x1818, 0x1838, 0x1858

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	SPIxURDT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SPIxURDT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SPIxURDT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SPIxURDT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – SPIxURDT[31:0] SPI Underrun Data bits
These bits are only used when URDTEN = 1. This register holds the data to transmit when a Transmit Underrun condition occurs.
Table 19-5 summarizes the valid data field for possible values of MODE32, MODE16 and WLENGTH[4:0] bits.

Table 19-5. MODE32, MODE16 and WLENGTH[4:0] Data Fields

MODE32	MODE16	WLENGTH[4:0]	COMMUNICATION	Valid Data Field
1	x	0	32-bit	DATA[31:0]
0	1	0	16-bit	DATA[15:0]
0	0	0	8-bit	DATA[07:0]
x	x	16 < N < 31	(N+1)-bit	DATA[31:(31-N)]
x	x	8 < N < 15	(N+1)-bit	DATA[15:(15-N)]
x	x	1 < N < 7	(N+1)-bit	DATA[07:(07-N)]

19.4 Operation

19.4.1 Modes of Operation

The SPI module offers the following operating modes:

- 8-bit, 16-bit and 32-bit Data Transmission modes
- 8-bit, 16-bit and 32-bit Data Reception modes
- Host and Client modes
- Framed SPI modes
- Audio Protocol Interface mode

Notes:

1. In Framed SPI mode, four pins are used: SDIx, SDOx, SCKx and $\overline{SSx}$.
2. If the Client Select feature is used, all four pins listed in note 1 are used.
3. If standard SPI is used, but CKE = 1, enabling/using the Client Select feature is mandatory, and therefore, all four pins listed in note 1 are used.
4. If standard SPI is used, but DISSDO = 1, only two pins are used: SDIx and SCKx; unless Client Select is also enabled.
5. In all other cases, three pins are used: SDIx, SDOx and SCKx.

19.4.1.1 SPI Host Mode Clock Frequency

The SPI module allows flexibility in baud rate generation through the 13-bit SPIxBRG register. SPIxBRG is readable and writable and determines the baud rate. The peripheral clock provided to the SPI module, PBCLK, is a divider function of the CPU core clock. This clock is divided based on the value loaded into SPIxBRG. The SCKx clock, obtained by dividing PBCLK, is 50% duty cycle and it is provided to the external devices through the SCKx pin.

Note: The SCKx clock is not free running for Non-Framed SPI modes. It will only run for 8, 16 or 32 pulses when SPIxBUF is loaded with data. It will, however, be continuous for Framed modes.

Equation 19-1 defines the SCKx clock frequency as a function of SPIxBRG settings.

Equation 19-1. SCKx Frequency

$$F_{SCK} = \frac{F_{PB}}{2 \cdot (SPIxBRG + 1)}$$

Therefore, the maximum baud rate possible is FPB/2 (SPIxBRG = 0) and the minimum baud rate possible is F_{PB}/16384.

Some sample SPI clock frequencies are shown in Table 19-6.

Table 19-6. Sample SCKx Frequencies(1)

SPIxBRG Setting	0	15	31	63	85	127	255	511
F_{PB} = 32 MHz	16.00 MHz	10.0 MHz	500 kHz	257 kHz	190.48 kHz	125 kHz	62.5 kHz	31.25 kHz
F_{PB} = 25 MHz	12.50 MHz	781.25 kHz	390.63 kHz	145.35 kHz	97.66 kHz	281.25 kHz	48.83 kHz	24.41 kHz
F_{PB} = 20 MHz	10.00 MHz	625 kHz	312.50 kHz	156.25 kHz	116.28 kHz	78.13 kHz	39.06 kHz	19.53 kHz
F_{PB} = 12 MHz	6.00 MHz	375 MHz	187.50 kHz	93.75 kHz	69.77 kHz	46.88 kHz	23.44 kHz	11.72 kHz
F_{PB} = 10 MHz	5.00 MHz	312.50 kHz	156.25 kHz	78.13 kHz	58.14 kHz	39.06 kHz	19.53 kHz	9.77kHz
F_{PB} = 8 MHz	4.00 MHz	250 kHz	125 kHz	62.50 kHz	46.51 kHz	31.25 kHz	15.63 kHz	7.81 kHz

Note:

1. Not all clock rates are supported. For further information, refer to the SPI timing specifications in the **37. Electrical Characteristics**.

19.4.1.2 8-Bit, 16-Bit and 32-Bit Operation

The SPI module allows three types of data widths when transmitting and receiving data over an SPI bus. The selection of data width determines the minimum length of SPI data. For example, when the selected data width is 32, all transmission and receptions are performed in 32-bit values. All reads and writes from the CPU are also performed in 32-bit values. Accordingly, the application software should select the appropriate data width to maximize its data throughput.

Two control bits, MODE32 and MODE16 (SPIxCON1[11:10]), which are referred to as MODE[32,16], define the mode of operation. To change the mode of operation on-the-fly, the SPI module must be Idle (i.e., not performing any transactions). If the SPI module is switched off (SPIxCON1[15] = 0), the new mode will be available when the module is again switched on.

Additionally, the following items should be noted in this context:

- The MODE[32,16] bits should not be changed when a transaction is in progress
- The first bit to be shifted out from SPIxTXSR varies with the selected mode of operation:
  - 8-bit mode, bit 7
  - 16-bit mode, bit 15
  - 32-bit mode, bit 31
- In each mode, data is shifted into bit 0 of the SPIxRXSR
- The number of clock pulses at the SCKx pin are also dependent on the selected mode of operation:
  - 8-bit mode, 8 clocks
  - 16-bit mode, 16 clocks
  - 32-bit mode, 32 clocks

19.4.1.3 Buffer Modes

There are two SPI Buffering modes: Standard and Enhanced.

19.4.1.3.1 Standard Buffer Mode

The SPIx Data Receive/Transmit Buffer (SPIxBUF) register is actually two separate internal registers: the Transmit Buffer (SPIxTXB) and the Receive Buffer (SPIxRXB). These two unidirectional registers share the SFR address of SPIxBUF.

When a complete byte/word is received, it is transferred from SPIxRXSR to SPIxRXB and the SPIRBF bit is set. If the software reads the SPIxBUF buffer, the SPIRBF bit is cleared.

As the software writes to SPIxBUF, the data is loaded into the SPIxTXB and the SPITBF bit is set by hardware. As the data is transmitted out of SPIxTXSR, the SPITBF bit is cleared.

The SPI module double-buffers transmit/receive operations and allows continuous data transfers in the background. Transmission and reception occur simultaneously in SPIxTXSR and SPIxRXSR, respectively.

19.4.1.3.2 Enhanced Buffer Mode

The Enhanced Buffer Enable (ENHBUF) bit in the SPIx Control Register 1 (SPIxCON1[0]) can be set to enable the Enhanced Buffer mode.

In Enhanced Buffer mode, two FIFO buffers are used for the SPIx Transmit Buffer (SPIxTXB) and the SPIx Receive Buffer (SPIxRXB). SPIxBUF provides access to both the receive and transmit FIFOs. The data transmission and reception in the SPIxSR buffer is identical to that in the Standard Buffer mode. The FIFO depth depends on the data width chosen by the Word/Half-Word Byte Communication Select (MODE[32,16]) bits in the SPIx Control Register 1 (SPIxCON1[11:10]). The FIFO depth varies between devices. For a device with FIFO depth 'X', the MODE field will modify it as follows:

- MODE = 8-bit, FIFO depth = X
- MODE = 16-bit, FIFO depth = X/2
- MODE = 32-bit, FIFO depth = X/4

Note: FIFO depth does not change when variable word length is configured. Refer to Table 19-3 for the value of FIFO depth 'X'.

The SPITBF status bit is set when all of the elements in the transmit FIFO buffer are full, and it is cleared if one or more of those elements are empty. The SPIRBF status bit is set when all of the elements in the receive FIFO buffer are full, and it is cleared if the SPIxBUF buffer is read by the software.

The SPITBE status bit is set if all the elements in the transmit FIFO buffer are empty and is cleared otherwise. The SPIRBE bit is set if all of the elements in the receive FIFO buffer are empty and is cleared otherwise.

There is underrun or overflow protection against reading an empty receive FIFO element or writing a full transmit FIFO element. The SPIxSTAT register provides the SPIx Transmit Underrun bit (SPITUR) and the Receive Overflow Status bit (SPIROV). Depending on the requirements, IGNTUR and IGNROV can be configured for SPI operation to be continued or not at the time of error. When a Transmit Underrun occurs, the last received data or the data in the SPIxURDT register can be transmitted by configuring the URDTEN bit (SPIxCON1[26]).

The Receive Buffer Element Count bits (RXELM[2:0]) in the SPIx Status Register (SPIxSTAT[26:24]) indicate the number of unread elements in the receive FIFO. The Transmit Buffer Element Count bits (TXELM[2:0]) in the SPIx Status Register (SPIxSTAT[18:16]) indicate the number of elements not transmitted in the transmit FIFO.

When configured for non-framed Client mode, it is important to ensure that the software can reload the transmit buffer quickly enough to keep up with the configured transfer rate. If the SPIxTXB is empty at the start of a transaction, then the transmit behavior will be undefined and is likely to cause errors (duplicate transmissions, missed bits, etc.) in the received data.

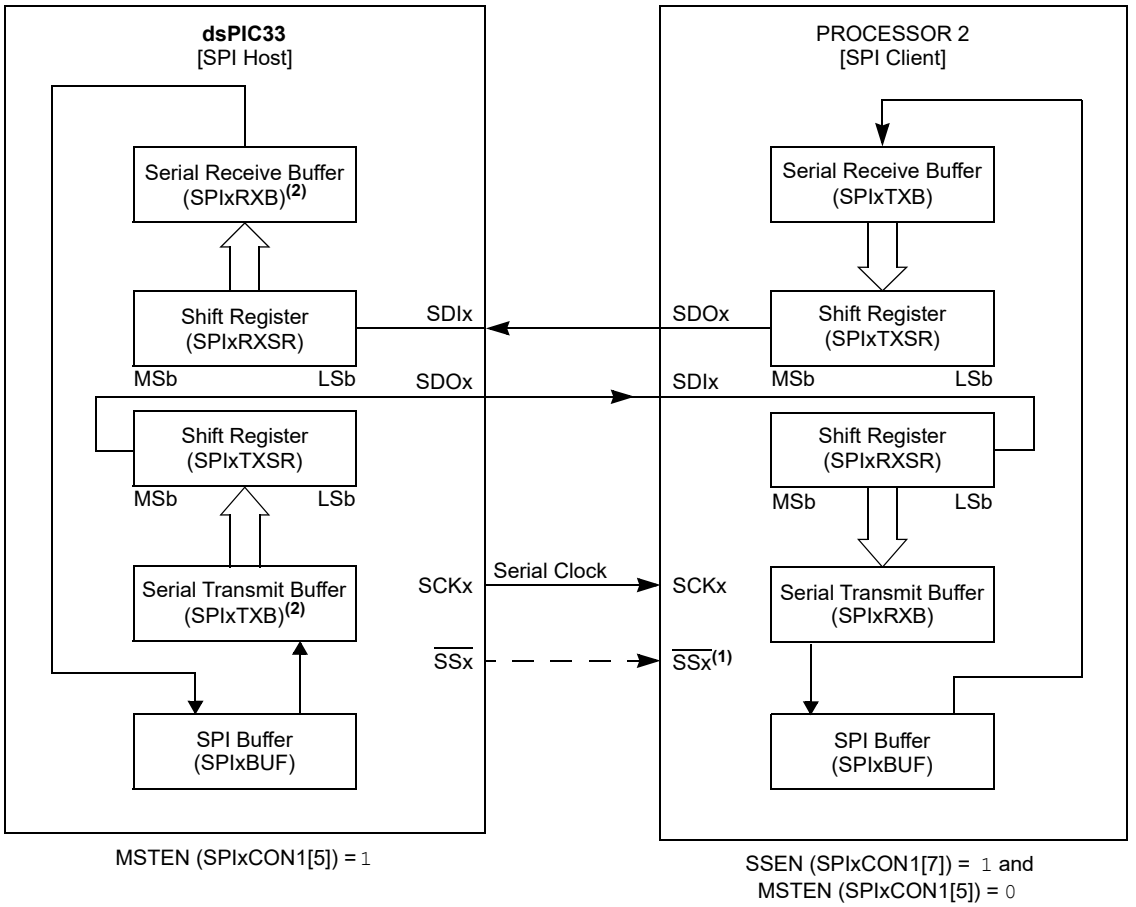
19.4.1.4 Variable Word Length Operation

The SPI module allows variable word length when transmitting and receiving data over an SPI bus. Word length can vary from 2 to 32 bits. Different word lengths can be configured by changing the WLENGTH[4:0] bits (SPIxCON2[4:0]). The number of clock pulses at the SCKx pin will correspond to the word length that is selected.

19.4.1.5 Host and Client Modes

In Host and Client modes, data can be thought of as taking a direct path between the Most Significant bit (MSb) of one module's Shift register and the Least Significant bit (LSb) of the other, and then moving them into the appropriate transmit or receive buffer. The module configured as the host module provides the serial clock and synchronization signals (as required) to the client device. The relationship between the host and client modules is shown in Figure 19-8.

Figure 19-8. SPIx Host/Client Connection Diagram



Notes:

1. Using the $\overline{SSx}$ pin in Client mode of operation is optional.
2. The user must write transmit data to SPIxBUF and read received data from SPIxBUF. The SPIxTXB and SPIxRXB registers are memory mapped to SPIxBUF.

19.4.1.5.1 Host Mode Operation

Perform the following steps to set up the SPI module for Host mode operation:

1. Disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit.
3. Clear the receive buffer.
4. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
5. SPIx interrupts are not going to be used, skip this step. Otherwise, the following additional steps are performed:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.

   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
6. Write the Baud Rate register, SPIxBRG.
7. Clear the SPIROV bit (SPIxSTAT[6]).
8. Write the desired settings to the SPIxCON1 register with MSTEN (SPIxCON1[5]) = 1.
9. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
10. Write the data to be transmitted to the SPIxBUF register. Transmission (and reception) will start as soon as data is written to the SPIxBUF register.

Note: The SPI device must be turned OFF prior to changing the mode from Client to Host. (When using the Client Select mode, the $\overline{SSx}$ pin or another GPIO pin is used to control the Client's $\overline{SSx}$ input. The pin must be controlled in software).

In Host mode, the PBCLK is divided and then used as the serial clock. The division is based on the settings in the SPIxBRG register. The serial clock is output through the SCKx pin to the client devices. Clock pulses are only generated when there is data to be transmitted; except when in Framed mode, when the clock is generated continuously. For further information, refer to 19.4.1.1. SPI Host Mode Clock Frequency.

The Host Mode Client Select Enable (MSSEN) bit in the SPIx Control Register 1 (SPIxCON1[20]) can be set to automatically drive the Client Select signal ($\overline{SSx}$) in Host mode. Clearing this bit disables the Client Select signal support in Host mode. The FRMPOL bit (SPIxCON1[21]) determines the polarity for the Client Select signal in Host mode.

Note: The MSSEN bit is not available on all devices. This bit should not be set when the SPI Framed mode is enabled (i.e., FRMEN = 1).

In devices that do not feature the MSSEN bit, the Client Select signal (in Non-Framed SPI mode) must be generated by using the $\overline{SSx}$ pin or another general purpose I/O pin under software control.

The CKP (SPIxCON1[6]) and CKE (SPIxCON1[8]) bits determine on which edge of the clock data transmission occurs.

Note: The user must turn OFF the SPI device prior to changing the CKE or CKP bits. Otherwise, the behavior of the device is not ensured.

Both data to be transmitted and data that is received are written to, or read from, the SPIxBUF register, respectively.

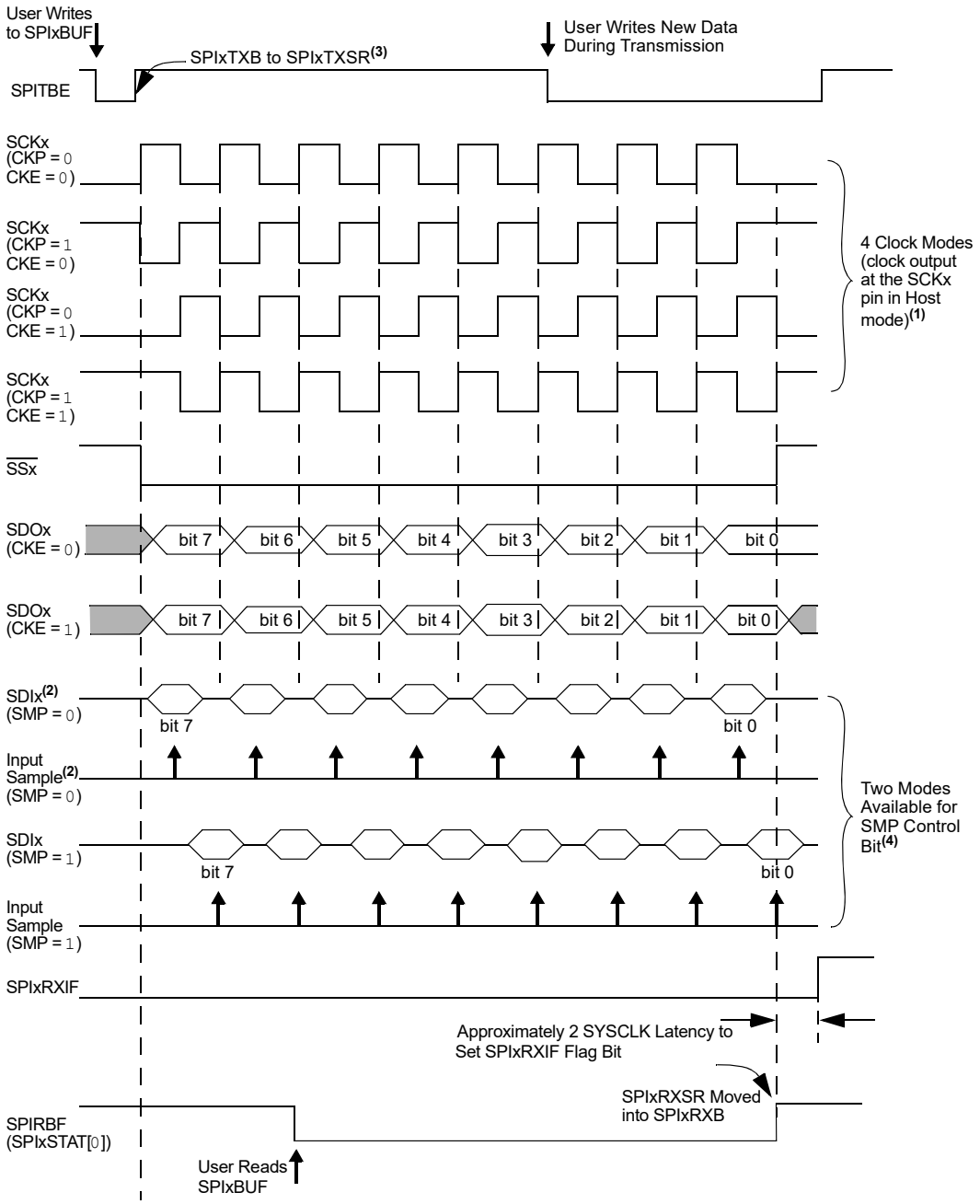
The following progression describes the SPI module operation in Host mode:

1. Once the module is set up for Host mode operation and enabled, data to be transmitted is written to the SPIxBUF register. The SPITBE bit (SPIxSTAT[3]) is cleared.
2. The contents of SPIxTXB are moved to the SPIx Shift register, SPIxTXSR (see Figure 19-9), and the SPITBE bit is set by the module.
3. A series of 8/16/32 clock pulses shifts 8/16/32 bits of transmit data from SPIxTXSR to the SDOx pin and simultaneously shifts the data at the SDIx pin into SPIxRXSR.
4. When the transfer is complete, the following events will occur:
   a. The SPIxRXIF interrupt flag bit is set. SPIx interrupts can be enabled by setting the SPIxRXIE interrupt enable bit. The SPIxRXIF flag is not cleared automatically by the hardware.
   b. Also, when the ongoing transmit and receive operation is completed, the contents of SPIxRXSR are moved to SPIxRXB.
   c. The SPIRBF bit (SPIxSTAT[0]) is set by the module, indicating that the receive buffer is full. Once SPIxBUF is read by the user code, the hardware clears the SPIRBF bit. In Enhanced Buffer mode, the SPIRBE bit (SPIxSTAT[5]) is set when the SPIxRXB FIFO buffer is completely empty and cleared when not empty.

5. If the SPIRBF bit is set (the receive buffer is full) when the SPI module needs to transfer data from SPIxRXSR to SPIxRXB, the module will set the SPIROV bit (SPIxSTAT[6]) indicating an overflow condition.
6. Data to be transmitted can be written to SPIxBUF by the user software at any time, if the SPITBE bit (SPIxSTAT[3]) is set. The write can occur while SPIxTXSR is shifting out the previously written data, allowing continuous transmission. In Enhanced Buffer mode, the SPITBF bit (SPIxSTAT[1]) is set when the SPIxTXB FIFO buffer is completely full and clear when it is not full.

Note: The SPIxTXSR register cannot be written directly by the user. All writes to the SPIxTXSR register are performed through the SPIxBUF register.

Figure 19-9. SPIx Host Mode Operation in 8-Bit Mode (MODE32 = 0, MODE16 = 0)



Notes:

1. Four SPI Clock modes are shown here to demonstrate the functionality of bits, CKP (SPIxCON1[6]) and CKE (SPIxCON1[8]). Only one of the four modes can be chosen for operation.
2. The SDIx and input samples shown here for two different values of the SMP bit (SPIxCON1[9]) are strictly for demonstration purposes. Only one of the two configurations of the SMP bit can be chosen during operation.
3. If there are no pending transmissions, SPIxTXB is transferred to SPIxTXSR as soon as the user writes to SPIxBUF.
4. Operation for 8-bit mode is shown; 16-bit and 32-bit modes are similar.

Example 19-1. Initialization Code for 16-Bit SPI Host Mode

```
/* The following code example will initialize the SPI1 in Host mode. */
_SPI1TXIP = 4;                // Set SPI Interrupt Priorities
SPI1BRG = 0x1;                // use FPB/4 clock frequency
SPI1STATbits.SPIROV = 0;      // clear the Overflow
SPI1CON1 = 0x00000420;        // 16 bits transfer, Host mode,ckp=0,cke=0,smp=0
SPI1IMSKbits.SPITBFEN = 1;    // SPI1 transmit buffer full generates interrupt
event
_SPI1TXIE = 1;                // Enable interrupts
SPI1CON1bits.ON = 1;
// from here, the device is ready to transmit and receive data. Buffer can be
loaded to transmit data.
```

19.4.1.5.2 Client Mode Operation

The following steps are used to set up the SPI module for the Client mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit.
3. Clear the receive buffer.
4. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
5. If using interrupts, the following additional steps are performed:
6. Clear the SPIx interrupt flags/events in the respective IFSx register.
7. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
8. Set the SPIx interrupt enable bits in the respective IECx register.
9. Clear the SPIROV bit (SPIxSTAT[6]).
10. Write the desired settings to the SPIxCON1 register with MSTEN (SPIxCON1[5]) = 0.
11. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
12. Transmission (and reception) will start as soon as the host provides the serial clock.

Note: The SPI module must be turned OFF prior to changing the mode from Host to Client.

In Client mode, data is transmitted and received as the external clock pulses appear on the SCKx pin. The CKP bit (SPIxCON1[6]) and the CKE bit (SPIxCON1[8]) determine on which edge of the clock data transmission occurs.

Both data to be transmitted and data that is received are respectively written into or read from the SPIxBUF register.

The rest of the operation of the module is identical to that in the Host mode, including Enhanced Buffer mode.

Client Mode Additional Features

The following additional features are provided in the Client mode:

Client Select Synchronization

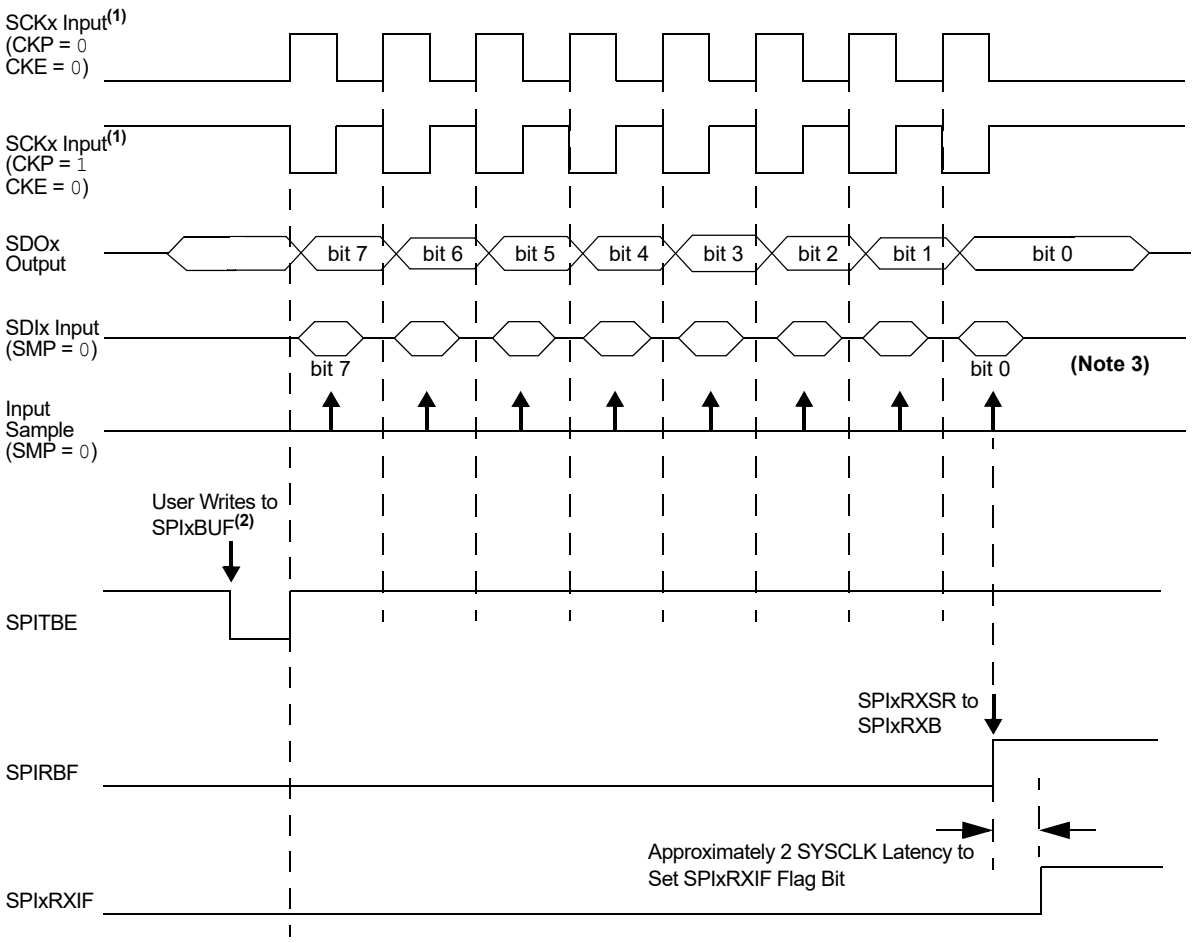
The $\overline{SSx}$ pin allows a Synchronous Client mode. If the SSEN bit (SPIxCON1[7]) is set, transmission and reception are enabled in Client mode only if the $\overline{SSx}$ pin is driven to a low state. The port output or other peripheral outputs must not be driven in order to allow the $\overline{SSx}$ pin to function as an input. If the SSEN bit is set and the $\overline{SSx}$ pin is driven high, the SDOx pin is no longer driven and will tri-state, even if the module is in the middle of a transmission. An aborted transmission will be retried the next time the $\overline{SSx}$ pin is driven low using the data held in the SPIxTXB register. If the SSEN bit is not set, the $\overline{SSx}$ pin does not affect the module operation in Client mode.

SPITBE Status Flag Operation

The SPITBE bit (SPIxSTAT[3]) has a different function in the Client mode of operation. The following describes the function of SPITBE for various settings of the Client mode of operation:

- If SSEN (SPIxCON1[7]) is cleared, the SPITBE bit is cleared when SPIxBUF is loaded by the user code. It is set when the module transfers SPIxTXB to SPIxTXSR. This is similar to the SPITBE bit function in Host mode.
- If SSEN is set, SPITBE is cleared when SPIxBUF is loaded by the user code. However, it is set only when the SPI module completes data transmission. A transmission will be aborted when the $\overline{SSx}$ pin goes high and may be retried at a later time. So, each data word is held in SPIxTXB until all bits are transmitted to the receiver.
  Note: Client Select cannot be used when operating in Frame mode.

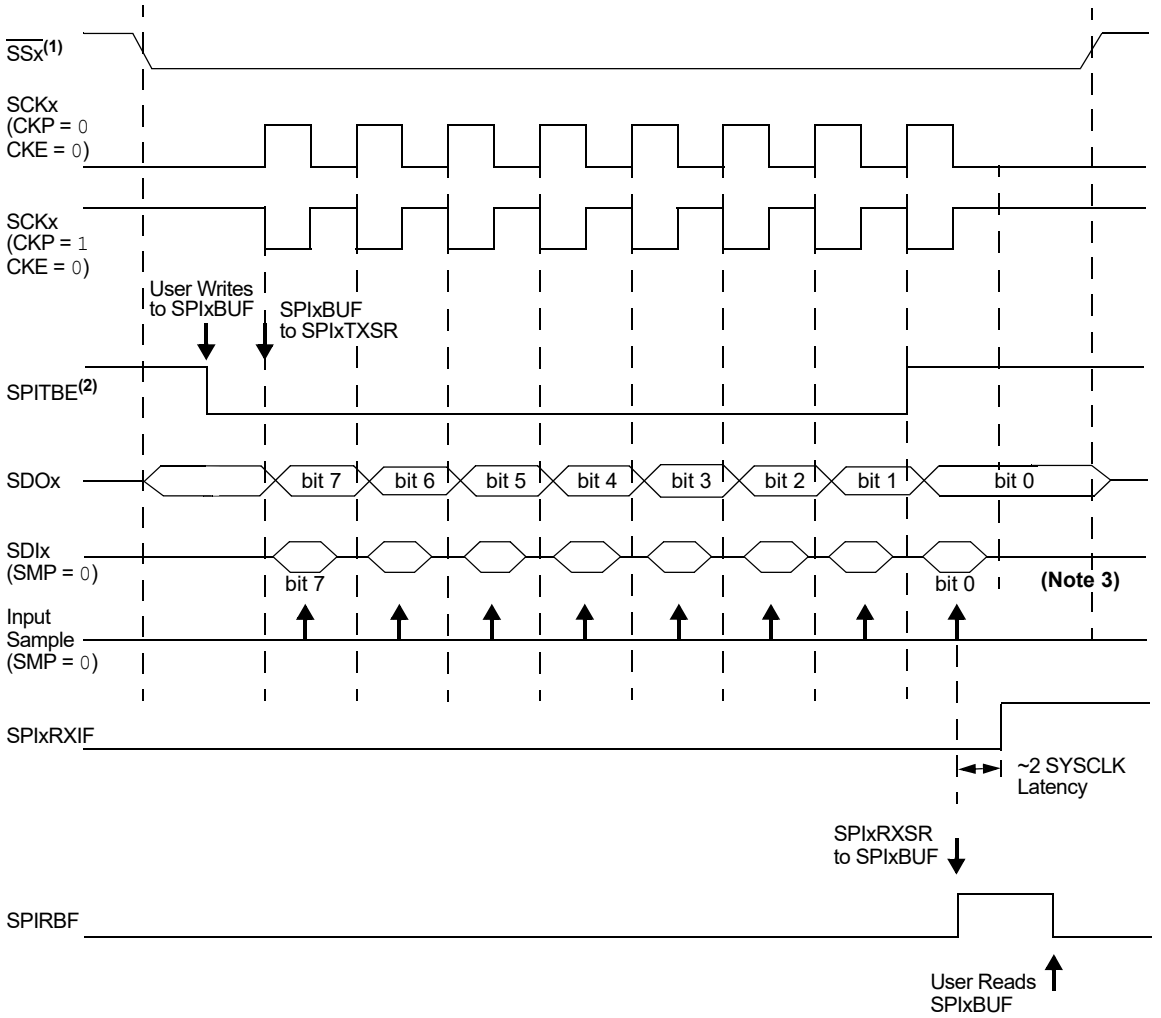
Figure 19-10. SPIx Client Mode Operation in 8-Bit Mode with Client Select Pin Disabled (MODE32 = 0, MODE16 = 0, SSEN = 0)



Notes:

1. Two SPI Clock modes are shown here only to demonstrate the functionality of bits, CKP (SPIxCON1[6]) and CKE (SPIxCON1[8]). Any combination of CKP and CKE bits can be chosen for module operation.
2. If there are no pending transmissions or a transmission is in progress, SPIxBUF is transferred to SPIxTXSR as soon as the user writes to SPIxBUF.
3. Operation for 8-bit mode is shown; 16-bit and 32-bit modes are similar.

Figure 19-11. SPIx Client Mode Operation in 8-Bit Mode with Client Select Pin Disabled (MODE32 = 0, MODE16 = 0, SSEN = 0)



Notes:

1. When the SSEN (SPIxCON1[7]) bit is set to '1', the $\overline{SSx}$ pin must be driven low so as to enable transmission and reception in Client mode.
2. Transmit data is held in SPIxTXB, and SPITBE (SPIxSTAT[3]) remains clear until all bits are transmitted.
3. Operation for 8-bit mode is shown; 16-bit and 32-bit modes are similar.

Example 19-2. Initialization Code for 16-Bit SPI Client Mode

```
/* The following code example will initialize the SPI1 in Client mode. */
_SPI1RXIP = 4;                      //Set SPI Interrupt Priorities
SPI1STATbits.SPIROV = 0;           // clear the Overflow
SPI1CON1 = 0x0400;                 // 16 bits transfer, Client
mode,ckp=0,cke=0,smp=0
SPI1IMSKbits.SPIRBFEN = 1;         // SPI1 receive buffer full generates
interrupt event
_SPI1RXIE = 1;                     // Enable interrupts
SPI1CON1bits.ON = 1;
// from here, the device is ready to transmit and receive data. Buffer can be
loaded to
transmit data.
```

19.4.1.6 SPI Error Handling

When a new data word has been shifted into the SPIx Shift register, SPIxRXSR, and the previous contents of the SPIx Receive register, SPIxRXB, have not been read by the user software, the SPIROV bit (SPIxSTAT[6]) will be set. The module will not transfer the received data from SPIxRXSR to the SPIxRXB. Further data reception is disabled until the SPIROV bit is cleared. The SPIROV bit is not cleared automatically by the module and must be cleared by the user software.

19.4.1.7 SPI Receive Only Operation

Setting the DISSDO control bit (SPIxCON1[12]) disables transmission at the SDOx pin. This allows the SPI module to be configured for a Receive Only mode of operation. The SDOx pin will be controlled by the respective port function if the DISSDO bit is set.

The DISSDO function is applicable to all SPI operating modes.

19.4.1.8 Framed SPI Modes

The module supports a very basic framed SPI protocol while operating in either Host or Client modes. The following features are provided in the SPI module to support Framed SPI modes:

- The FRMEN control bit (SPIxCON1[23]) enables Framed SPI mode and causes the SSx pin to be used as a Frame Synchronization pulse input or output pin. The state of SSEN (SPIxCON1[7]) is ignored.
- The FRMSYNC control bit (SPIxCON1[22]) determines whether the $\overline{SSx}$ pin is an input or an output (i.e., whether the module receives or generates the Frame Synchronization pulse).
- The FRMPOL control bit (SPIxCON1[21]) determines the Frame Synchronization pulse polarity for a single SPI clock cycle.
- The FRMSYPW control bit (SPIxCON1[19]) can be set to configure the width of the Frame Synchronization pulse to one character wide.

The FRMCNT[2:0] control bits (SPIxCON1[18:16]) can be set to configure the number of data characters transmitted per Frame Synchronization pulse.

The following Framed SPI modes are supported by the SPI module:

Frame Host mode

The SPI module generates the Frame Synchronization pulse and provides this pulse to other devices at the $\overline{SSx}$ pin.

Frame Client mode

The SPI module uses a Frame Synchronization pulse received at the $\overline{SSx}$ pin.

The Framed SPI modes are supported in conjunction with the Host and Client modes. Therefore, the following Framed SPI mode configurations are available:

- SPI Host mode and Frame Host mode

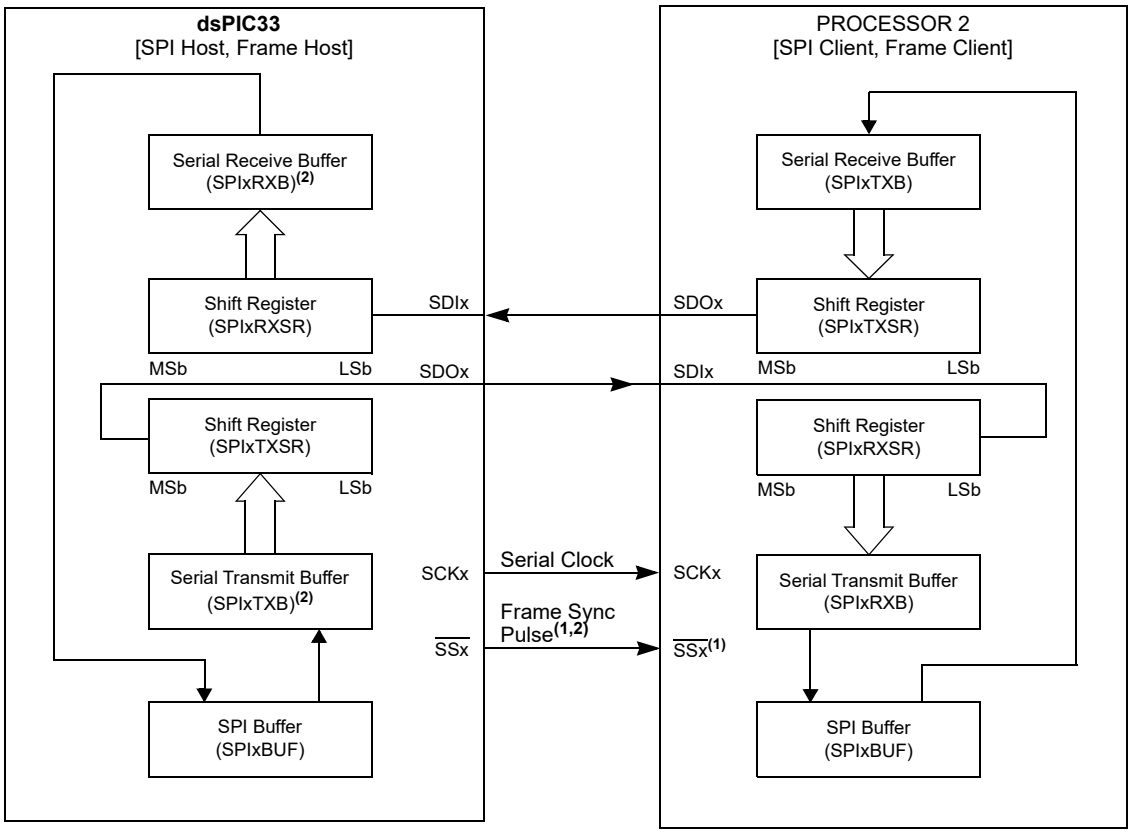
- SPI Host mode and Frame Client mode
- SPI Client mode and Frame Host mode
- SPI Client mode and Frame Client mode

These four modes determine whether or not the SPI module generates the serial clock and the Frame Synchronization pulse.

The ENHBUF bit (SPIxCON1[0]) can be configured to use the Standard Buffering mode or Enhanced Buffering mode in Framed SPI mode.

In addition, the SPI module can be used to interface to external audio DAC/ADC and codec devices in Framed SPI mode.

Figure 19-12. SPIx Host, Frame Host Connection Diagram



Notes:

1. In Framed SPI modes, the $\overline{SSx}$ pin is used to transmit/receive the Frame Synchronization pulse.
2. Framed SPI modes require the use of all four pins (i.e., using the $\overline{SSx}$ pin is not optional).
3. The SPIxTXB and SPIxRXB registers are memory mapped to the SPIxBUF register.

19.4.1.8.1 SCKx in Framed SPI Modes

When FRMEN (SPIxCON1[23]) = 1 and MSTEN (SPIxCON1[5]) = 1, the SCKx pin becomes an output and the SPI clock at SCKx becomes a free-running clock.

When FRMEN = 1 and MSTEN = 0, the SCKx pin becomes an input. The source clock provided to the SCKx pin is assumed to be a free-running clock.

The polarity of the clock is selected by the CKP bit (SPIxCON1[6]). The CKE bit (SPIxCON1[8]) is not used for the Framed SPI modes.

When CKP or CKE = 0, the Frame Sync pulse output and the SDOx data output change on the rising edge of the clock pulses at the SCKx pin. Input data is sampled at the SDIx input pin on the falling edge of the serial clock.

When CKP or CKE = 1, the Frame Sync pulse output and the SDOx data output change on the falling edge of the clock pulses at the SCKx pin. Input data is sampled at the SDIx input pin on the rising edge of the serial clock.

19.4.1.8.2 SPI Buffers in Framed SPI Modes

When FRMSYNC (SPIxCON1[22]) = 0, the SPI module is in the Frame Host mode of operation.

In this mode, the Frame Sync pulse is initiated by the module when the user software writes the transmit data to a SPIxBUF location (thus, writing the SPIxTXB register with transmit data). At the end of the Frame Sync pulse, SPIxTXB is transferred to SPIxTXSR and data transmission/reception begins.

When FRMSYNC = 1, the module is in Frame Client mode. In this mode, the Frame Sync pulse is generated by an external source. When the module samples the Frame Sync pulse, it will transfer the contents of the SPIxTXB register to SPIxTXSR, and data transmission/reception begins. The user must make sure that the correct data is loaded into the SPIxBUF for transmission before the Frame Sync pulse is received.

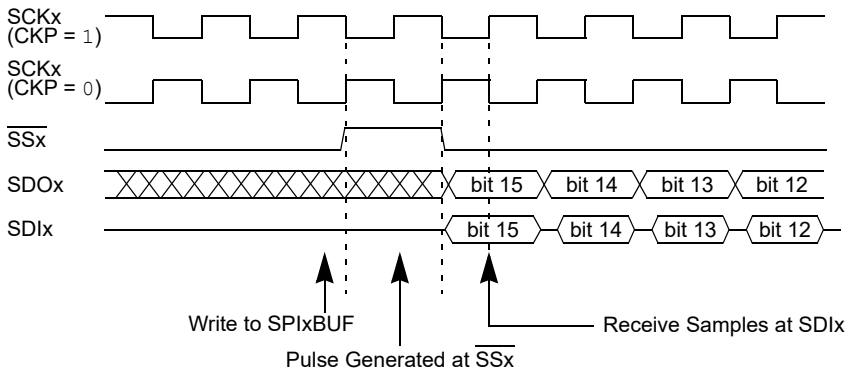
Note: Receiving a Frame Sync pulse will start a transmission, regardless of whether or not data was written to SPIxBUF. If a write was not performed, zeros will be transmitted.

19.4.1.8.3 SPI Host Mode and Frame Host Mode

This Framed SPI mode is enabled by setting the MSTEN bit (SPIxCON1[5]) and the FRMEN bit (SPIxCON1[23]) to '1', and the FRMSYNC bit (SPIxCON1[22]) to '0'. In this mode, the serial clock will be output continuously at the SCKx pin, regardless of whether the module is transmitting. When SPIxBUF is written, the $\overline{SSx}$ pin will be driven active-high or active-low, depending on the FRMPOL bit (SPIxCON1[21]), on the next transmit edge of the SCKx clock.

The $\overline{SSx}$ pin will be high for one SCKx clock cycle. The module will start transmitting data on the next transmit edge of SCKx, as shown in Figure 19-13. A connection diagram indicating signal directions for this operating mode is shown in Figure 19-13.

Figure 19-13. SPIx Host, Frame Host (MODE32 = 0, MODE16 = 1, SPIFE = 0, FRMPOL = 1)



19.4.1.8.4 SPI Host Mode and Frame Client Mode

This Framed SPI mode is enabled by setting the MSTEN bit (SPIxCON1[5]), the FRMEN bit (SPIxCON1[23]) and the FRMSYNC bit (SPIxCON1[22]) to '1'. The SSx pin is an input and it is sampled on the sample edge of the SPI clock. When it is sampled active-high or active-low, depending on the FRMPOL bit (SPIxCON1[21]), data will be transmitted on the subsequent transmit edge of the SPI clock, as shown in Figure 19-14. The SPIx Interrupt Flag, SPIxIF, is set when the transmission is complete. The user must make sure that the correct data is loaded into SPIxBUF for transmission

before the signal is received at the $\overline{SSx}$ pin. A connection diagram indicating signal directions for this operating mode is shown in Figure 19-15.

Figure 19-14. SPIx Host, Frame Client (MODE32 = 0, MODE16 = 1, SPIFE = 0, FRMPOL = 1)

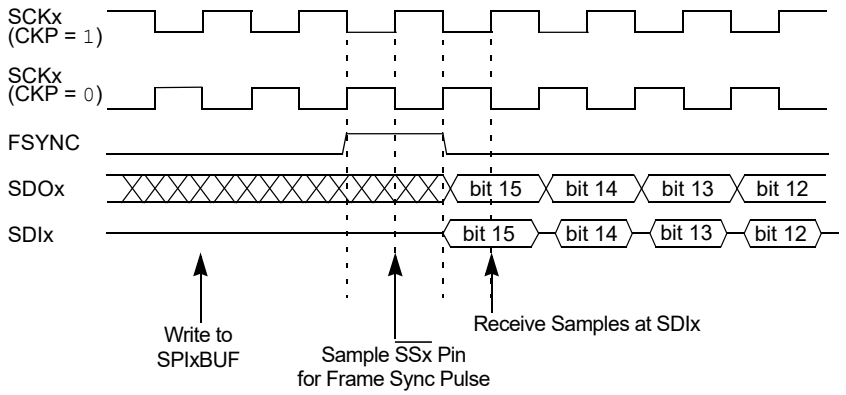
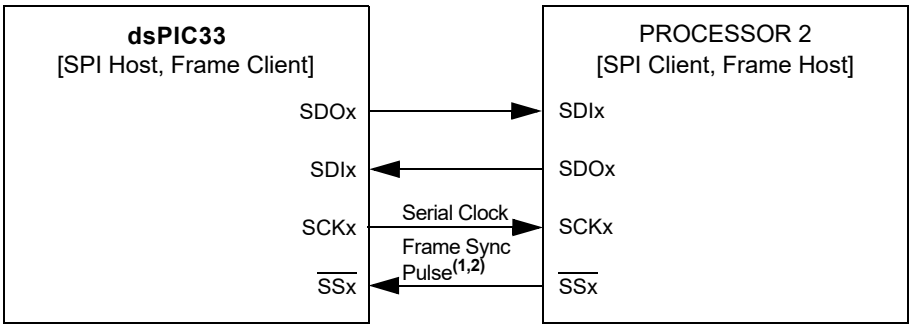


Figure 19-15. SPIx Host, Frame Client Connection Diagram



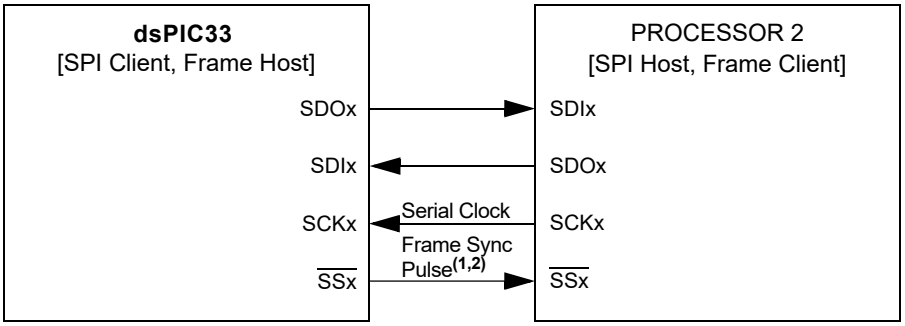
Notes:

1. In Framed SPI modes, the $\overline{SSx}$ pin is used to transmit/receive the Frame Synchronization pulse.
2. Framed SPI modes require the use of all four pins (i.e., using the $\overline{SSx}$ pin is not optional).

19.4.1.8.5 SPI Client Mode and Frame Host Mode

This Framed SPI mode is enabled by setting the MSTEN bit (SPIxCON1[5]) to '0', the FRMEN bit (SPIxCON1[23]) to '1' and the FRMSYNC bit (SPIxCON1[22]) to '0'. The input SPI clock will be continuous in Client mode. The $\overline{SSx}$ pin will be an output when bit, FRMSYNC, is low. Therefore, when SPIxBUF is written, the module will drive the $\overline{SSx}$ pin active-high or active-low, depending on the FRMPOL bit (SPIxCON1[21]), on the next transmit edge of the SPI clock. The $\overline{SSx}$ pin will be driven high for one SPI clock cycle. Data transmission will start on the next SPI clock transmit edge. A connection diagram indicating signal directions for this operating mode is shown in Figure 19-16.

Figure 19-16. SPIx Client, Frame Host Connection Diagram



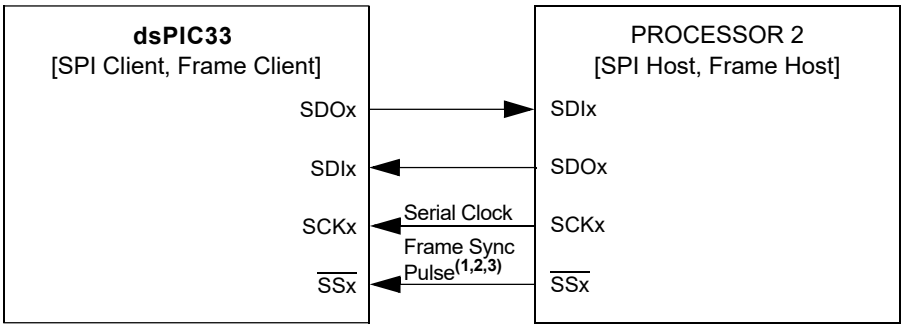
Notes:

1. In Framed SPI modes, the $\overline{SSx}$ pin is used to transmit/receive the Frame Synchronization pulse.
2. Framed SPI modes require the use of all four pins (i.e., using the $\overline{SSx}$ pin is not optional).

19.4.1.8.6 SPI Client Mode and Frame Client Mode

This Framed SPI mode is enabled by setting the MSTEN bit (SPIxCON1[5]) to '0', the FRMEN bit (SPIxCON1[23]) to '1' and the FRMSYNC bit (SPIxCON1[22]) to '1'. Therefore, both the SCKx and $\overline{SSx}$ pins will be inputs. The $\overline{SSx}$ pin will be sampled on the sample edge of the SPI clock. When $\overline{SSx}$ is sampled active-high or active-low, depending on the FRMPOL bit (SPIxCON1[21]), data will be transmitted on the next transmit edge of SCKx. A connection diagram indicating signal directions for this operating mode is shown in Figure 19-17.

Figure 19-17. SPIx Client, Frame Client Connection Diagram



Notes:

1. In Framed SPI modes, the $\overline{SSx}$ pin is used to transmit/receive the Frame Synchronization pulse.
2. Framed SPI modes require the use of all four pins (i.e., using the $\overline{SSx}$ pin is not optional).
3. Client Select is not available when using Frame mode as a client device.

19.4.2 Audio Protocol Interface Mode

The SPI module can be interfaced to most codec devices available today to provide dsPIC33 microcontroller-based audio solutions. The SPI module provides support to the audio protocol functionality through four standard I/O pins. The four pins that make up the audio protocol interface modes are:

- SDIx: Serial Data Input for receiving sample Digital Audio Data (ADCDAT)
- SDOx: Serial Data Output for transmitting Digital Audio Data (DACDAT)
- SCKx: Serial Clock, also known as the Bit Clock (BCLK)
- $\overline{SSx}$: Left/Right Channel Clock (LRCK)

BCLK provides the clock required to drive the data out or into the module, while LRCK provides the synchronization of the frame based on the Protocol mode selected.

In some codecs, Serial Clock (SCK) refers to the Baud/Bit Clock (BCLK). Throughout this section, the signal, $\overline{SSx}$, is to be referred to as LRCK to be consistent with codec naming conventions. The SPI module has the ability to function in Audio Protocol Host and Audio Protocol Client modes. In Host mode, the module generates both the BCLK on the SCKx pin and the LRCK on the $\overline{SSx}$ pin. In certain devices, while in Client mode, the module receives these two clocks from its I^2S partner, which is operating in Host mode.

While in Host mode, the SPI module has the ability to generate its own clock internally through the Host Clock (MCLK) from various internal sources, such as the primary clock, PBCLK, USB clock, FRC and other internal sources. In addition, the SPI module has the ability to provide the MCLK to the codec device, which is a common requirement.

To start the Audio Protocol mode, first disable the peripheral by setting the ON bit (SPIxCON1[15]) = 0. Next, set the AUDEN bit (SPIxCON1[31]) = 1 and then re-enable the peripheral by setting the ON bit = 1.

When configured in Host mode, the leading edge of SCKx and the LRCK is driven out within one SCKx period of starting the audio protocol. Serial data is shifted in or out with timing determined by the Protocol mode set by the AUDMOD[1:0] bits (SPIxCON1[25:24]). If the transmit FIFO is empty, zeros are transmitted.

In Client mode, the peripheral drives zeros out of SDOx but does not transmit the contents of the transmit FIFO until it sees the leading edge of the LRCK, after which time it starts receiving data (provided SDIx has not been disabled). It will continue to transmit zeros as long as the transmit FIFO is empty.

While in Client or Host mode, the SPI module does not generate an underrun on the TX FIFO after start-up. This allows software to set up the SPI, set up the DMA, turn on the SPI module's audio protocol and then turn on the DMA without getting an error.

After the first write to the TX FIFO (SPIxBUF), the SPI enables underrun detection and generation. To keep the RX FIFO empty until the DMA is enabled, set DISSDI (SPIxCON1[4]) = 1. After enabling the DMA, set DISSDI = 0 to start receiving.

19.4.2.1 Host Mode

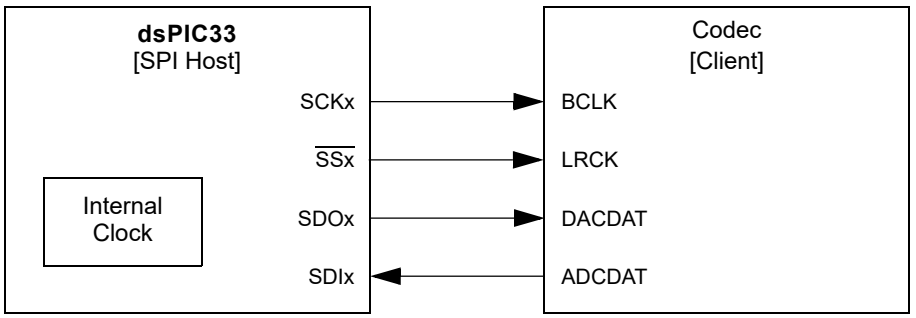
To configure the dsPIC33 device in Audio Protocol Host mode, set both the MSTEN bit (SPIxCON1[5]) and the AUDEN bit (SPIxCON1[31]) to '1'.

A few characteristics of Host mode are:

- This mode enables the device to generate SCKx and LRCK pulses as long as the ON bit (SPIxCON1[15]) = 1.
- The SPI module generates LRCK and SCKx continuously in all cases, regardless of the transmit data while in Host mode.
- The SPI module drives the leading edge of LRCK and SCKx within one SCKx period, and the serial data shifts in and out continuously, even when the TX FIFO is empty.

Figure 19-18 shows a typical interface between Host and Client while in Host mode.

Figure 19-18. Host Generating its Own Clock – Output BCLK and LRCK



19.4.2.2 Client Mode

The SPI module can be configured in Audio Protocol Client mode by setting the MSTEN bit = 0 (SPIxCON1[5]) and the AUDEN bit = 1 (SPIxCON1[31]).

A few characteristics of Client mode are:

- This mode enables the device to receive SCKx and LRCK pulses as long as the ON bit (SPIxCON1[15]) = 1.
- The SPI module drives zeros out of SDOx, but does not shift data out or in (SDIx) until the module receives the LRCK (i.e., the edge that precedes the left channel).
- Once the module receives the leading edge of LRCK, it starts receiving data if DISSDI (SPIxCON1[4]) = 0 and the serial data shifts out continuously, even when the TX FIFO is empty.

Figure 19-19 shows the interface between a SPI module in Audio Client Interface mode to a codec host device.

Figure 19-19. Codec Device as Host Generates Required Clock via External Crystal

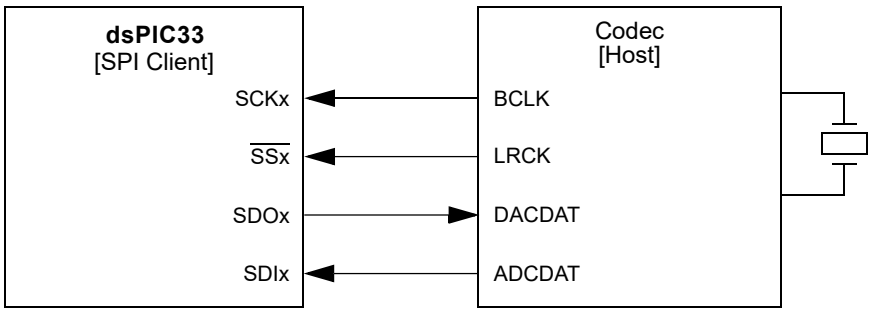
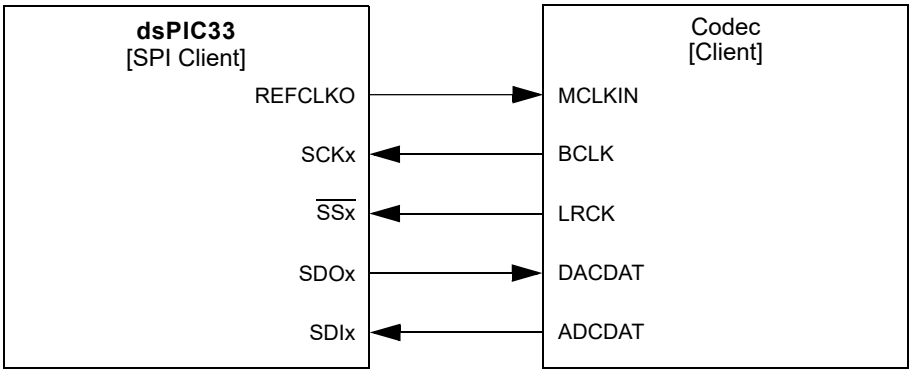


Figure 19-20 shows the interface between an SPI module in Audio Client Interface mode to a codec host device, in which the host clock is being derived from the SPI reference clock out function.

Figure 19-20. Codec Device as Host Derives MCLK from dsPIC33 Reference Clock Out



19.4.2.3 Audio Data Length and Frame Length

While codec devices may generate audio data samples of various word lengths of 8, 16, 20, 24 and 32, the dsPIC33 SPI module supports transmit/receive audio data lengths of 16, 24 and 32.
Note: Actual sample data can be any length, with a maximum of 32 bits, and the data must be packed in one of three (16/24/32) formats.

Table 19-7 illustrates how the MODE[32,16] bits (SPIxCON1[11:10]) control the maximum allowable sample length and frame length (LRCK period on $\overline{SSx}$).

Table 19-7. Audio Data Length vs. LRCK Period

SPIxCON1[11:10]		Data Length (bits)	FIFO Width (bits)	Left/Right Channel Sample Length (bits)	Enhanced Buffer FIFO Depth (samples)(1)	LRCK Period Frame Length (bits)
MODE32	MODE16					
0	0	16	16	≤ 16	X/2	32
0	1	16	16	≤ 32	X/2	64
1	1	24	32	≤ 32	X/4	64
1	0	32	32	≤ 32	X/4	64

Note:

1. FIFO depth varies between devices. The data in the table above is specified considering a device with available FIFO depth of 'X'.

The parameters of the MODE[32,16] bits (SPIxCON1[11:10]) have the following behavior:

- Controls left/right channel data length, frame length
- In 16-bit Sample mode, 32/64-bit frame length is supported
- In 24/32-bit Sample mode, 64-bit frame length is supported
- Defines FIFO width and depth (for example, 24-bit data has a 32-bit wide and X/4-location deep FIFO)
- If the written data is greater than the data selected, the upper bytes are ignored
- If the written data is less than the data selected, the FIFO Pointers change on the write to the Most Significant Byte (MSB) of the selected length

If this data is written to the transmit FIFO in more than one write, the write order must be from Least Significant to Most Significant.

For example, assume that audio data is 24 bits per sample with 8 bits available at a time. According to Table 4-1, the FIFO width is 32 bits per sample. Therefore, the 8 Most Significant bits (MSbs), bits[31:24], in each FIFO sample are ignored.

Data written to unused bytes is ignored. Also, transactions that are only to unused bytes are also ignored. Therefore, a byte write to address offset, 0x0023, is completely ignored and does not cause a FIFO push if the data is less than 32 bits wide.

19.4.2.4 Frame Error/LRCK Errors

The SPI module provides detection of frame/LRCK errors for debugging. The frame/LRCK error occurs when the LRCK edge that is defining a channel start happens before the correct number of bits (as defined by MODE[32,16]).

The SPI module immediately sets the FRMERR bit (SPIxSTAT[12]), pushes data in from the SPIxRXSR register into the SPIxRXB register, and pops data from the SPIxTXB register into the SPIxTXSR register. The module can be configured to detect frame/LRCK related errors by setting the FRMERREN bit (SPIxIMSK[12]).

Note: In Audio Protocol mode, both the BCLK (on the SCKx pin) and the LRCK (on the $\overline{SSx}$ pin) are free running, meaning they are continuous. Normally, the LRCK is a fixed number of BCLKs long. In all cases, the SPI module will realign to the new frame edge and will set the FRMERR bit. If operating in a Non-PCM mode, the SPI module will also push the abbreviated data onto the FIFO when the frame is too short.

19.4.2.5 Audio Protocol Modes

The SPI module supports four Audio Protocol modes and can be operated in any one of these modes:

- I^2S mode
- Left Justified mode
- Right Justified mode
- PCM/DSP mode

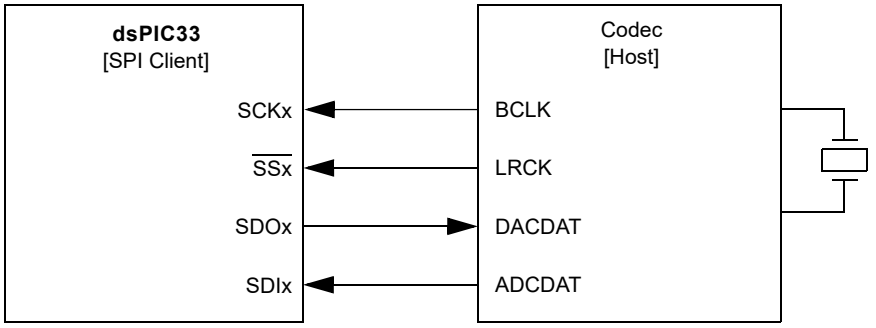
These Audio Protocol modes can be enabled by configuring the AUDMOD[1:0] bits (SPIxCON1[25:24]). These modes enable communication to different types of codecs and control the edge relationships of LRCK and SDIx/SDOx with respect to SCKx.

With respect to data transmit, in all of the Protocol modes, the MSB is first transmitted followed by MSB-1, and so on, until the Least Significant Byte (LSB) transmits. The length of the data is discussed in 19.4.2.3. Audio Data Length and Frame Length. If there are SCKx periods left over after the LSb is transmitted, zeros are sent to fill up the frame.

When in Client mode, the relationship between the BCLK (on the SCKx pin) and the period (or frame length) of the LRCK (on the $\overline{SSx}$ pin) is far less constrained than that of Host mode. In Host mode, the frame length equals 32 or 64 BCLKs, depending on the MODE[32,16] (SPIxCON1[11:10]) bit settings. However, in Client mode, the frame length can be greater than or equal to 32 or 64 BCLKs, but the FRMERR bit (SPIxSTAT[12]) will be set if the frame LRCK edge arrives early.

Figure 19-21 illustrates the general interface between the codec device and the SPI module in Audio mode.

Figure 19-21. SPIx Module in Audio Client Mode – BCLK and WS or LRCK Generated by Host



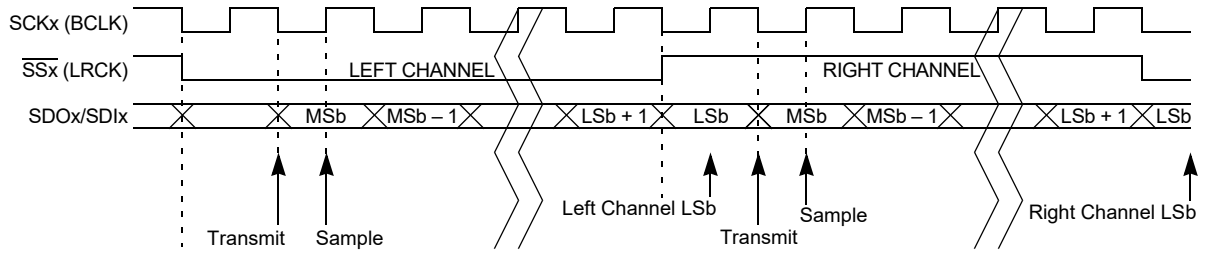
19.4.2.5.1 I²S Mode

The Inter-IC Sound (I²S) protocol enables transmission of two channels of digital audio data over a single serial interface. The I²S protocol defines a 3-wire interface that handles the stereo data using the WS/LRCK line. The I²S specification defines a half-duplex interface that supports transmit or receive but not both at the same time. With both SDOx and SDIx available, full-duplex operation is supported by this peripheral, as shown in Figure 19-22.

- Data Transmit and Clocking:
  - The transmitter shifts the audio sample data's MSb on the first falling edge of SCKx, after an LRCK transition
  - The receiver samples the MSB on the second rising edge of SCKx
  - The left channel data shifts out while LRCK is low and the right channel data is shifted out while LRCK is high
  - The data in the left and right channels consists of a single frame
- Required Configuration Settings:
  To set the module to I²S mode, the following bits must be set:
  - AUDMOD[1:0] = 00 (SPIxCON1[25:24])
  - FRMPOL = 0 (SPIxCON1[21])
  - CKP = 1 (SPIxCON1[6])

Setting these bits enables the SDOx and LRCK ($\overline{SSx}$) transitions to occur on the falling edge of SCKx (BCLK) and sampling of SDIx to occur on the rising edge of SCKx. Refer to the diagrams shown in Figure 19-22.

Figure 19-22. I²S with 16-Bit Data/Channel or 32-Bit Data/Channel



I²S Audio Client Mode of Operation

Use the following steps to set up the SPI module for the I²S Audio Client mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Clear the receive buffer.
5. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
6. If using interrupts, the following additional steps need to be performed:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
7. Clear the SPIROV bit (SPIxSTAT[6]).

8. Write the desired settings to the SPIxCON1 register.
   a. AUDMOD[1:0] bits (SPIxCON1[25:24]) = 00
   b. AUDEN bit (SPIxCON1[15]) = 1
9. Write the desired settings to the SPIxCON1 register:
   a. MSTEN (SPIxCON1[5]) = 0
   b. CKP (SPIxCON1[6]) = 1
   c. MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
   d. Enable SPI operation by setting the ON bit (SPIxCON1[31]).
10. Transmission (and reception) will start as soon as the host provides the BCLK and LRCK.

Example 19-3. I²S Client Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in I2S Client
mode. */
_SPI1RXIP = 4;
SPI1STATbits.SPIROV = 0;      // clear the Overflow
SPI1CON1=0x80000440;          // AUDEN=1, I2S mode, stereo mode,
                              // 16 bits/32 channel transfer, Client mode,CKP
=1
SPI1IMSKbits.SPIRBFEN = 1;    // SPI1 receive buffer full generates interrupt
event
_SPI1RXIE = 1;                // Enable interrupts
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

I²S Audio Host Mode of Operation

A typical application could be to play PCM data (8 kHz sample frequency, 16-bit data, 32-bit frame size) when interfaced to a codec client device. In this case, the SPI module is initialized to generate BCLK @ 625 kbps. Assuming a 20 MHz peripheral clock, F_{PB} = 20e6, the baud rate would be determined using Equation 19-2.

Equation 19-2. Baud Rate Calculation

$$Baud\ Rate = \frac{F_{PB}}{2 \cdot (SPIxBRG + 1)}$$

Solving for the value of SPIxBRG is shown in Equation 19-3.

Equation 19-3. Baud Rate Calculation

$$SPIxBRG = \frac{F_{PB}}{2(Baud\ Rate)} - 1$$

The Baud Rate is now equal to 256e3. Equation 19-4 shows the resulting calculation.

Equation 19-4. Baud Rate Calculation

$$SPIxBRG = \frac{40e6}{2(256e3)} - 1 = 15$$

If the result of Equation 19-4 is rounded to the nearest integer, SPIxBRG is now equal to 15; therefore, the effective Baud Rate is that of Equation 19-5.

Equation 19-5. Baud Rate Calculation

$$\frac{20e6}{2 \cdot (15+1)} = \frac{20e6}{32} = 625000 \; bits \; per \; second$$

The following steps can be used to set up the SPI module for operation in I²S Audio Host mode:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Reset the SPIx Baud Rate Register, SPIxBRG.
5. Clear the receive buffer.
6. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
7. If using interrupts, perform these additional steps:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
8. Clear the SPIROV bit (SPIxSTAT[6]).
9. Write the desired settings to the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to '00' for I²S mode and the AUDEN bit (SPIxCON1[31]) must be set to '1' to enable the audio protocol.
10. Set the SPIxBRG register to 0x4F (to generate approximately 625 kbps sample rate with PBCLK @ 20 MHz).
11. Write the desired settings to the SPIxCON1 register:
    a. MSTEN (SPIxCON1[5]) = 1.
    b. CKP (SPIxCON1[6]) = 1.
    c. MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
    d. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
12. Transmission (and reception) will start immediately after the ON bit is set.

Example 19-4. I²S Host Mode, 625 kbps BCLK, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in I2S Host
mode. */
_SPI1TXIP = 4;
SPI1STATbits.SPIROV = 0;        // clear the Overflow
SPI1BRG = 0x000F;               // to generate 625 kbps sample rate, PBCLK @ 20
MHz
SPI1CON1=0x80000460;            // AUDEN=1, I2S mode, stereo mode,
                                // 16 bits/32 channel transfer, Host mode, CKP
= 1
SPI1IMSKbits.SPITBFEN = 1;      // SPI1 transmit buffer full generates
interrupt event
_SPI1TXIE = 1;                  // Enable interrupts
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

19.4.2.5.2 Left Justified Mode

The Left Justified mode is similar to I2S mode, however, in this mode, the SPI shifts the audio data's MSb on the first SCKx edge that is coincident with an LRCK transition. On the receiver side, the SPI module samples the MSb on the next SCKx edge.

In general, a codec using justified protocols defaults to transmitting data on the rising edge of SCKx and receiving data on the falling edge of SCKx.

- Required Configuration Settings
  To set the module to Left Justified mode, the following bits must to be set:
  - AUDMOD[1:0] = 01 (SPIxCON1[25:24])
  - FRMPOL = 1 (SPIxCON1[21])
  - CKP = 0 (SPIxCON1[6])

This enables the SDOx and LRCK transitions to occur on the rising edge of SCKx. Refer to the sample waveform diagrams shown in Figure 19-23 and Figure 19-24 for 16, 24 and 32-bit audio data transfers.

Figure 19-23. Left Justified with 16-Bit Data/Channel or 32-Bit Data/Channel

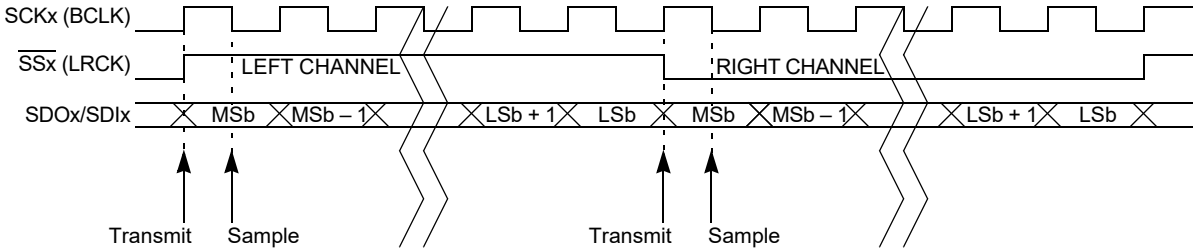
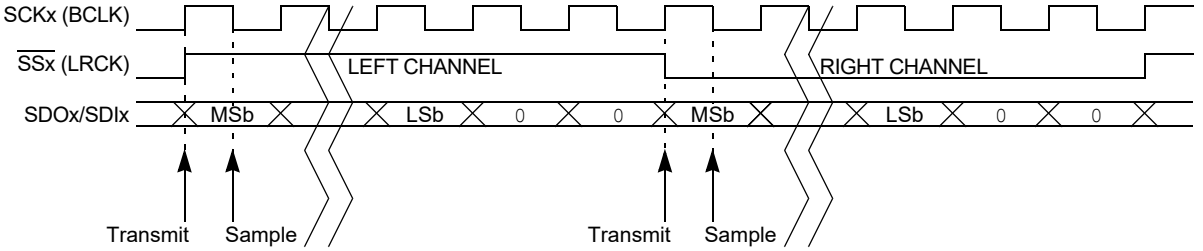


Figure 19-24. Left Justified with 16/24-Bit Data and 32-Bit Channel



Left Justified Audio Client Mode Operation

Use the following steps to set up the SPI module for the Left Justified Audio Client mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Clear the receive buffer.
5. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
6. If using interrupts, the following additional steps are performed:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
7. Clear the SPIROV bit (SPIxSTAT[6]).

8. Write the desired settings in the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to '01' for Left Justified mode and the AUDEN bit (SPIxCON1[31]) must be set to '1' to enable the audio protocol.
9. Write the desired settings to the SPIxCON1 register:
   a. Set to Client mode, MSTEN (SPIxCON1[5]) = 0.
   b. Set Clock Polarity, CKP (SPIxCON1[6]) = 0.
   c. Set Frame Polarity, FRMPOL (SPIxCON1[5]) = 1.
   d. Set MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
   e. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
10. Transmission (and reception) will start as soon as the host provides the BCLK and LRCK.

Example 19-5. Left Justified Client Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in Left
Justified Client mode. */
_SPI1RXIP = 4;
SPI1STATbits.SPIROV = 0;          // clear the Overflow
SPI1CON1=0x81200400;              // AUDEN=1, Left justified mode, stereo
mode,
                                  // FRMPOL = 1,16 bits/32 channel transfer,
SPI1IMSKbits.SPIRBFEN = 1;        // SPI1 receive buffer full generates
interrupt event
_SPI1RXIE = 1;                    // Enable interrupts
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

Left Justified Audio Host Mode Operation

Use the following steps to set up the SPI module for the Left Justified Audio Host mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Reset the SPIx Baud Rate Register, SPIxBRG.
5. Clear the receive buffer.
6. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
7. If using interrupts, the following additional steps are performed:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
8. Clear the SPIROV bit (SPIxSTAT[6]).
9. Write the desired settings in the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to '01' for left justified and the AUDEN bit (SPIxCON1[31]) must be set to '1' to enable the audio protocol.
10. Set the SPIx Baud Rate Register Low, SPIxBRG, to 0x4F (to generate approximately 256 kbps sample rate with PBCLK @ 20 MHz).
11. Write the desired settings to the SPIxCON1 register:
    a. Set to Host mode, MSTEN (SPIxCON1[5]) = 1.
    b. Set Clock Polarity, CKP (SPIxCON1[6]) = 0.
    c. Set Frame Polarity, FRMPOL (SPIxCON1[21]) = 1.
    d. Set MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
    e. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
12. Transmission (and reception) will start immediately after the ON bit is set.

Example 19-6. Left Justified Host Mode, 625 kbps BLCK, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in Left
Justified Host mode. */
_SPI1TXIP = 4;
SPI1STATbits.SPIROV = 0;      // clear the Overflow
SPI1BRG = 0x000F;             // to generate 625 kbps sample rate, PBCLK @ 20
MHz
SPI1CON1 = 0x81200420;        // AUDEN =1, Left Justified mode, stereo mode,
                              // FRMPOL = 1, 16 bits/32 channel transfer,Host
```

```
mode,CKP = 0
SPI1IMSKbits.SPITBFEN = 1;   // SPI1 transmit buffer full generates interrupt
event
_SPI1TXIE = 1;               // Enable interrupts
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

19.4.2.5.3 Right Justified Mode

In Right Justified mode, the SPI module shifts the audio sample data's MSb after aligning the data to the last clock cycle. The bits preceding the audio sample data can be driven to logic level '0' by setting the DISSDO bit (SPIxCON1[12]) to '0'. When DISSDO = 0, the module ignores the unused bit slot.

- Required Configuration Settings
  To set the module to Right Justified mode, the following bits must to be set:
  - AUDMOD[1:0] (SPIxCON1[25:24]) = 10
  - FRMPOL (SPIxCON1[21]) = 1
  - CKP (SPIxCON1[6]) = 0

This enables the SDOx and LRCK transitions to occur on the rising edge of SCKx, after the LSb is aligned to the last clock cycle. Refer to the sample waveform diagrams shown in Figure 19-25 and Figure 19-26 for 16, 24 and 32-bit audio data transfers.

Figure 19-25. Right Justified with 16-Bit Data/Channel or 32-Bit Data/Channel

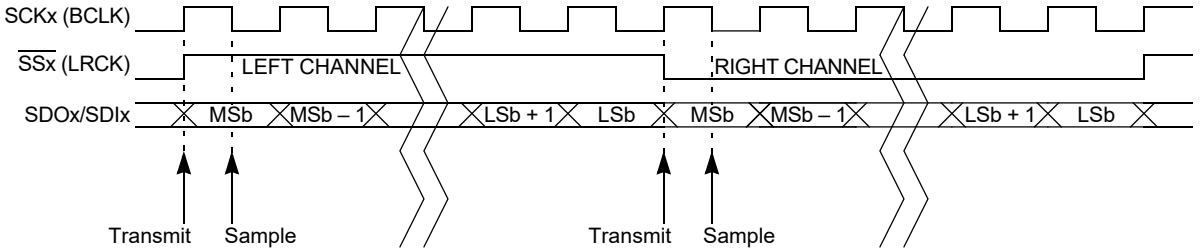
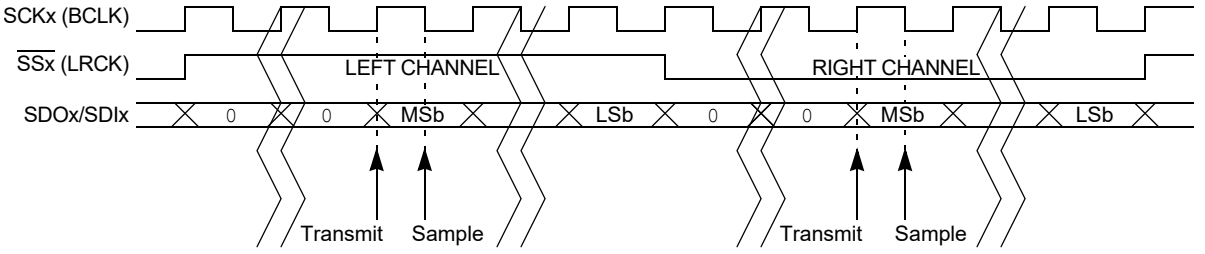


Figure 19-26. Right Justified with 16/24-Bit Data and 32-Bit Channel



Right Justified Audio Client Mode Operation

Use the following steps to set up the SPI module for the Right Justified Audio Client mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Clear the receive buffer.

5. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
6. If using interrupts, perform the following steps:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
7. Clear the SPIROV bit (SPIxSTAT[6]).
8. Write the desired settings in the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to '10' for Right Justified mode and the AUDEN bit (SPIxCON1[31]) must be set to '1' to enable the audio protocol.
9. Write the desired settings to the SPIxCON1 register:
   a. Set to Client mode, MSTEN (SPIxCON1[5]) = 0.
   b. Set Clock Polarity, CKP (SPIxCON1[6]) = 0.
   c. Set Frame Polarity, FRMPOL (SPIxCON1[21]) = 1.
   d. Set MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
   e. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
10. Transmission (and reception) will start as soon as the host provides the BCLK and LRCK.

Example 19-7. Right Justified Client Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in Right
Justified Client mode. */
_SPI1RXIP = 4;
SPI1STATbits.SPIROV = 0;      // clear the Overflow
SPI1CON1 = 0x82200400;        // AUDEN=1, Right Justified mode, stereo
mode,FRMPOL=1,
                              // 16 bits/32 channel transfer,Client mode,ckp=0
SPI1IMSKbits.SPIRBFEN = 1;    // SPI1 receive buffer full generates interrupt
event
_SPI1RXIE = 1;                // Enable interrupts
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

Right Justified Audio Host Mode Operation

Use the following steps to set up the SPI module for the Right Justified Audio Host mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Reset the SPIx Baud Rate Register, SPIxBRG.
5. Clear the receive buffer.
6. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
7. If using interrupts, the following additional steps are performed:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
8. Clear the SPIROV bit (SPIxSTAT[6]).

9. Write the desired settings in the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to '10' for Right Justified mode and the AUDEN bit (SPIxCON1[31]) must be set to '1' to enable the audio protocol.
10. Set the SPIx Baud Rate Register, SPIxBRG, to 0x0F (to generate approximately 256 kbps sample rate with PBCLK @ 20 MHz).
11. Write the desired settings to the SPIxCON1 register:
    a. Set to Host mode, MSTEN (SPIxCON1[5]) = 1.
    b. Set Clock Polarity, CKP (SPIxCON1[6]) = 0.
    c. Set Frame Polarity, FRMPOL (SPIxCON1[21]) = 1.
    d. Set MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
    e. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
12. Transmission (and reception) will start immediately after the ON bit is set.

Example 19-8. Right Justified Host Mode, 625 kbps BLCK, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in Right
Justified Host mode. */
_SPI1TXIP = 4;
SPI1STATbits.SPIROV = 0;        // clear the Overflow
SPI1BRG = 0x000F;               // to generate 625 kbps sample rate, PBCLK @ 20
MHz
SPI1CON1 = 0x82200420;          // AUDEN=1, Right Justified mode,stereo mode,
                                // 16 bits/32 channel transfer,Host mode,ckp=0
SPI1IMSKbits.SPITBFEN = 1;      // SPI1 transmit buffer full generates
interrupt event
_SPI1TXIE = 1;                  // Enable interrupts
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

19.4.2.5.4 PCM/DSP Mode

The PCM/DSP Protocol mode is available for communication with some codecs and certain DSP (Digital Signal Processor) devices. This mode modifies the behavior of LRCK and audio data spacing. In PCM/DSP mode, the LRCK can be a single bit wide (i.e., 1 SCKx) or as wide as the audio data (16, 24, 32 bits). The audio data is packed in the frame with the left channel data immediately followed by the right channel data. The frame length is still either 32 or 64 clocks when this device is the host.

In PCM/DSP mode, the transmitter drives the audio data's (left channel) MSb on the first or second transmit edge (see the SPIFE bit (SPIxCON1[1])) of SCKx (after an LRCK transition). Immediately after the (left channel) LSb, the transmitter drives the (right channel) MSb.

- Required Configuration Settings
  To set the module to Left Justified mode, the following bits must to be set:
  - AUDMOD[1:0] bits (SPIxCON1[25:24]) = 11

Refer to the sample waveform diagrams shown in Figure 19-27 and Figure 19-28 for 16, 24 and 32-bit audio data transfers.

Figure 19-27. PCM/DSP with 16-Bit Data/Channel or 32-Bit Data/Channel

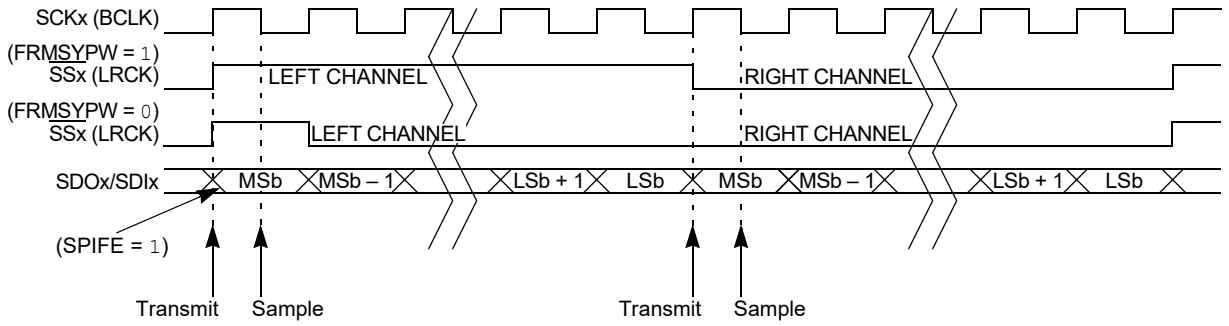
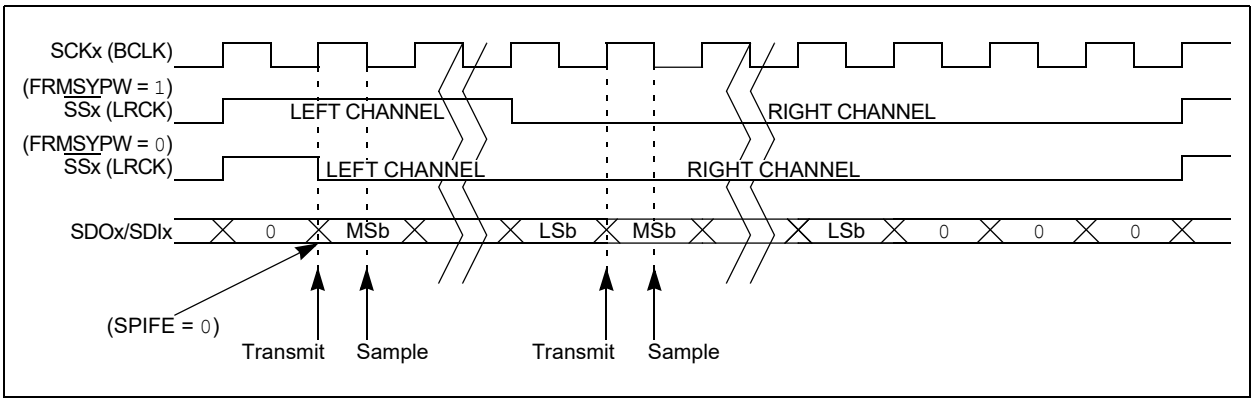


Figure 19-28. PCM/DSP with 16/24-Bit Data and 32-Bit Channel



PCM/DSP Audio Client Mode of Operation

Use the following steps to set up the SPI module for the PCM/DSP Audio Client mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Clear the receive buffer.
5. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
6. If using interrupts, the following additional steps are performed:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
7. Clear the SPIROV bit (SPIxSTAT[6]).
8. Write the desired setting in the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to '11' for DSP/PCM mode and the AUDEN bit (SPIxCON1[31]) must be set to '1' to enable audio protocol.
9. Write the desired settings to the SPIxCON1 register:
   a. Set to Client mode, MSTEN (SPIxCON1[5]) = 0.
   b. Set MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
   c. Enable SPI operation by setting the ON bit (SPIxCON1[15]).

10. Transmission (and reception) will start as soon as the host provides the BCLK and LRCK.

Example 19-9. PCM/DSP Client Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in PCM/DSP
Client Mode. */
_SPI1RXIP = 4;
SPI1STATbits.SPIROV = 0;        // clear the Overflow
SPI1CON1 = 0x83200400;          // AUDEN=1,PCM/DSP mode,stereo mode,FRMPOL=1,
                                // 16 bits/32 channel transfer,Client mode,ckp=0
SPI1IMSKbits.SPIRBFEN = 1;      // SPI1 receive buffer full generates interrupt
event
_SPI1RXIE = 1;                  // Enable interrupts
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

PCM/DSP Audio Host Mode of Operation

Use the following steps to set up the SPI module for the PCM/DSP Audio Host mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Reset the SPIx Baud Rate Register, SPIxBRG.
5. Clear the receive buffer.
6. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
7. If using interrupts, perform the following steps:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
8. Clear the SPIROV bit (SPIxSTAT[6]).
9. Write the desired settings in the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to '11' for DSP/PCM mode and the AUDEN bit (SPIxCON1[31]) must be set to '1' to enable the audio protocol.
10. Set the SPIx Baud Rate Register, SPIxBRG, to 0x0F (to generate approximately 256 kbps sample rate with PBCLK @ 20 MHz).
11. Write the desired settings to the SPIxCON1 register:
    a. Set to Host mode, MSTEN (SPIxCON1[5]) = 1.
    b. Set MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
    c. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
12. Transmission (and reception) will start immediately after the ON bit is set.

Example 19-10. PCM/DSP Host Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in PCM/DSP Host
Mode. */
_SPI1TXIP = 4;
SPI1STATbits.SPIROV = 0;        // clear the Overflow
SPI1BRG = 0x000F;               // to generate 625 kbps sample rate, PBCLK @ 20
MHz
SPI1CON1 = 0x8320;              // AUDEN=1, PCM/DSP mode,stereo mode,FRMPOL=1
SPI1CON1 = 0x0400;              // 16 bits/32 channel transfer,Host mode,ckp=0
SPI1IMSKbits.SPITBFEN = 1;      // SPI1 transmit buffer full generates interrupt
event
_SPI1TXIE=1;                    // Enable interrupts
```

```
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

19.4.2.6 Audio Protocol Mode Features

19.4.2.6.1 BCLK/SCKx and LRCK Generation

BCLK and LRCK generation is a key requirement in Host mode. The frame frequency of SCKx and LRCK is defined by the MODE[32,16] bits (SPIxCON1[11:10]). When the frame is 64 bits, SCKx is 64 times the frequency of LRCK. Similarly, when the frame is 32 bits, SCKx is 32 times the frequency of LRCK. The frequency of SCKx must be derived from the toggling rate of LRCK and the frame size.

For example, to sample 16-bit channel data at 8 kHz with PBCLK = 36.864 MHz, set the SPIxBRG register to 0x47 to generate an 8 kHz LRCK.

19.4.2.6.2 Host Mode Clocking and MCLK

The SPI module as a host has the ability to generate BCLK and LRCK by internally generating using PBCLK (MCLKEN = 0). The SPI module can generate the clock for external codec devices using the reference output, REFCLKO, function (see Figure 19-29), although some codecs may have the ability to generate their own MCLK from a crystal to provide accurate audio sample rates. Figure 19-30 shows that the REFCLKO clock can be used as MCLKIN by the codec.

Figure 19-29. SPIx Host Clock Generation

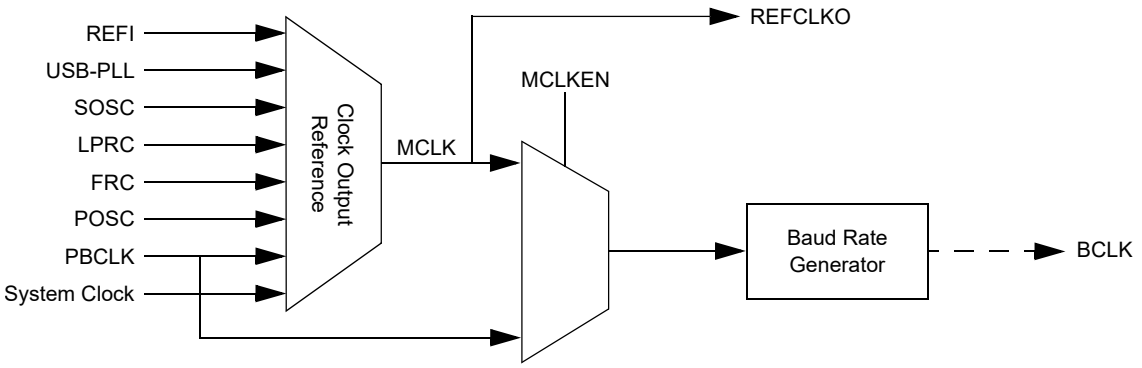
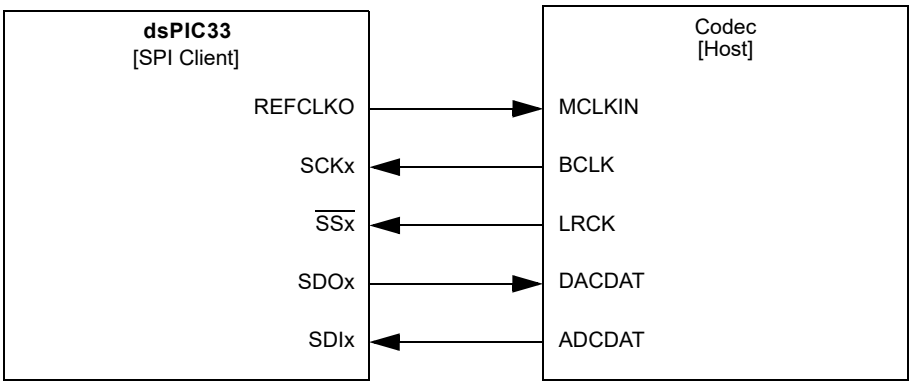


Figure 19-30 shows the interface between an SPI client and a codec host, deriving the clock from the MCLK input interface.

Figure 19-30. SPIx Client and Codec Host – Clock Derived from MCLK



I²S Audio Host Mode of Operation Using REFCLKO

The following steps can be used to set up the SPI module for the I²S Audio Host mode of operation with MCLK enabled. The SPI module is initialized to generate BCLK @ 625 kbps and MCLK is derived from PBCLK using the Reference Oscillator Controller register. A typical application could be to play PCM data (8 kHz sample frequency, 16-bit data, 32-bit frame) when interfaced to a codec client device.

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Reset the Reference Oscillator Controller register, REFOCON.
5. Reset the SPIx Baud Rate Register, SPIxBRG.
6. Clear the receive buffer.
7. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
8. If using interrupts, the following additional steps are performed:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
   d. Clear the SPIROV bit (SPIxSTAT[6]).
9. Write the desired settings in the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to '00' for I²S mode and the AUDEN bit (SPIxCON1[31]) must be set to '1' to enable the audio protocol.
10. Set the Reference Oscillator Controller register, REFOCON:
    a. RODIV[14:0] (REFOCON[30:16]) = 0.
    b. ROEN (REFOCON[15]) = 1, reference oscillator is enabled.
    c. ROOUT (REFOCON[12]) = 1, output is enabled.
11. Set the SPIx Baud Rate Register, SPIxBRG, to 0x1F (to generate approximately 625 kbps sample rate with PBCLK @ 20 MHz).
12. Write the desired settings to the SPIxCON1 register with:
    a. MSTEN (SPIxCON1[5]) = 1.
    b. CKP (SPIxCON1[6]) = 1.
    c. MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
    d. MCLKEN (SPIxCON1[2]) = 1, Host mode.
    e. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
13. Transmission (and reception) will start as soon as the host provides the BCLK and LRCK.

Note: The use of a reference clock output to generate MCLK for the codec may not be a perfect choice. Driving a clock out to an I/O pad induces a jitter that may degrade audio fidelity of the codec. The best solution is for the codec to use a crystal and be the host I²S/audio device.

Example 19-11. I²S Host Mode, 625 kbps BCLK, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in I2S Maaster
mode.
_SPI1TXIP = 4;
SPI1STATbits.SPIROV = 0;      // clear the Overflow
SPI1BRG = 0x000F;             // to generate 625 kbps sample rate, PBCLK @ 20
MHz
SPI1CON1 = 0x80000464;        // AUDEN=1, I2S mode, stereo mode,
```

```
                                // 16 bits/32 channel transfer,
Host mode,ckp=1,MCLKEN =1
REFOCON = 0x8001;               // REFO ROEN = 1, ROSEL = 1 for PBCLK
SPI1IMSKbits.SPITBFEN = 1;      // SPI1 transmit buffer full generates interrupt
event
_SPI1TXIE = 1;                  // Enable interrupts
_SPI1RXIE = 1;
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

19.4.2.7 Mono Mode vs. Stereo Mode

The SPI module enables the audio data transmission in Mono or Stereo mode by setting the AUDMONO bit (SPIxCON1[27]). When the AUDMONO bit is set to '0' (Stereo mode), the SPIx Shift register uses each FIFO location once, which gives each channel a unique stream of data for stereo data. When the AUDMONO bit is set to '1' (Mono mode), the SPIx Shift register uses each FIFO location twice to give each channel the same mono stream of audio data.

Note: Receive data is not affected by AUDMONO bit settings.

19.4.2.8 Streaming Data Support and Error Handling

Most audio streaming applications transmit or receive data continuously. This is required to keep the channel active during the period of operation and ensures the best possible accuracy. Due to streaming audio, the data feeds could be bursty or packet loss can occur causing the module to encounter situations such as underrun. The software needs to be involved to recover from an underrun.

The Ignore Transmit Underrun (IGNTUR) bit (SPIxCON1[28]), when set to a '1', ignores an Underrun condition. This is helpful for cases when software does not care or does not need to know about underrun conditions. When an underrun is encountered, the SPI module sets the SPITUR bit (SPIxSTAT[8]) and when URDTEN = 1 (SPIxCON1[26]), the module remains in an error state until the software clears the state or the ON bit = 0 (SPIxCON1[15]).

During the Underrun condition, the SPI module loads the SPIxTXSR with the data in the SPIxURDT register when the URDTEN bit is set to '1'. If URDTEN is not set to '1', then the last received data during underrun is loaded to SPIxTXSR. The module samples the Underrun condition on channel boundaries, so transmission of SPIxURDT data can start with either the left or right audio channel.

When the condition clears (i.e., SPIxTXB is not empty), the logic loads audio data from the transmit buffer into the SPIxTXSR on the next LRC frame boundary. Because recovery from the Underrun condition occurs on the LRC frame boundary (i.e., at the end of a full left and right channel pair), software must ensure the left and right audio data is always transferred to the FIFO in pairs.

The Ignore Receive Overflow (IGNROV) bit (SPIxCON1[29]), when set to a '1', ignores a Receive Overflow condition. This is useful when there is a general performance problem in the system that software must handle properly. An alternate method to handle the Receive Overflow is by setting the DISSDI bit = 1 (SPIxCON1[4]) when the system does not need to receive audio data. After changing the DISSDI bit on-the-fly, the SPIx Receive Shift register starts a receive on the leading LRCK edge.

If an RX overflow occurs when IGNROV = 0, the I^2S will behave just like it would in SPI mode, that is, it will stop writing to the RX FIFO. However, recovery from overflow is different from SPI mode. When the CPU gets around to reading the RX FIFO, the I^2S will restart receiving into the RX FIFO only when two additional conditions are met:

- The I^2S is on an LRC boundary
- There are a multiple of two locations free in the RX FIFO

These conditions will ensure the received data will start at the beginning of the LEFT channel and there is room to receive the RIGHT channel information immediately following.

19.5 Interrupts

The SPI module has the ability to generate interrupts reflecting the events that occur during the data communication. The following types of interrupts can be generated:

- Receive data available interrupts are signaled by SPIxRXIF. This event occurs when:
  - RX watermark interrupt
  - SPIROV = 1
  - SPIRBF = 1
  - SPIRBE = 1, provided respective mask bits are enabled in SPIxIMSK
- Transmit buffer empty interrupts are signaled by SPIxTXIF. This event occurs when:
  - TX watermark interrupt
  - SPITUR = 1
  - SPITBF = 1
  - SPITBE = 1, provided respective mask bits are enabled in SPIxIMSK
- General interrupts are signaled by SPIxIF. This event occurs when:
  - FRMERR = 1
  - BUSY = 1
  - SRMT = 1, provided respective mask bits are enabled in SPIxIMSK

All of these interrupt flags, which must be cleared in software, are located in the IFSx registers. Refer to the **"Interrupt Controller"** chapter for details.

To enable the SPIx interrupts, use the respective SPIx Interrupt Enable bits, SPIxRXIE, SPIxTXIE and SPIxIF, in the corresponding IECx registers.

The Interrupt Priority Level (IPL) bits must be also be configured using the SPIxIP bits in the corresponding IPCx registers.

When using Enhanced Buffer mode, the SPIx transmit buffer can be configured to interrupt at different FIFO levels using mask bits, TXMSK[2:0] in SPIxIMSK[18:16]. Also, the Transmit Watermark Interrupt bit, TXWIEN (SPIxIMSK[23]), should be enabled.

Similarly, the SPIx receive buffer can be configured to interrupt at different FIFO levels using mask bits, RXMSK[2:0] (SPIxIMSK[26:24]). Also, the Receive Watermark Interrupt, RXWIEN (SPIxIMSK[31]), should be enabled.

19.5.1 Interrupt Configuration

Each SPI module has three dedicated interrupt flag bits: SPIxIF, SPIxRXIF and SPIxTXIF, and corresponding interrupt enable bits, SPIxIE, SPIxRXIE and SPIxTXIE. These bits are used to determine the source of an interrupt and to enable or disable an individual interrupt source. Note that all the interrupt sources for a specific SPI module share one interrupt vector. Each SPI module can have its own priority level independent of other SPI modules.

Note: SPIxTXIF, SPIxRXIF and SPIxIF bits will be set without regard to the state of the corresponding enable bit. The interrupt flag bits can be polled by software if desired.

The SPIxIE, SPIxTXIE and SPIxRXIE bits are used to define the behavior of the interrupt controller when a corresponding SPIxIF, SPIxTXIF or SPIxRXIF bit is set. When the corresponding interrupt enable bit is clear, the interrupt controller does not generate a CPU interrupt for the event. If the interrupt enable bit is set, the interrupt controller will generate an interrupt to the CPU when the corresponding interrupt flag bit is set (subject to the priority as outlined below).

It is the responsibility of the user's software routine that services a particular interrupt to clear the appropriate interrupt flag bit before the service routine is complete.

The priority of each SPI module can be set independently with the SPIxIP[2:0] bits. This priority defines the priority group to which the interrupt source will be assigned. The priority groups range from a value of seven (the highest priority), to a value of zero (which does not generate an interrupt). An interrupt being serviced will be preempted by an interrupt in a higher priority group.

The priority group bits allow more than one interrupt source to share the same priority. If simultaneous interrupts occur in this configuration, the natural order of the interrupt sources within a priority group pair determines the interrupt generated. The natural priority is based on the vector numbers of the interrupt sources. The lower the vector number, the higher the natural priority of the interrupt. Any interrupts that were overridden by natural order will then generate their respective interrupts based on priority and natural order, after the interrupt flag for the current interrupt is cleared.

After an enabled interrupt is generated, the CPU will jump to the vector assigned to that interrupt. The vector number for the interrupt is the same as the natural order number. The CPU will then begin executing code at the vector address. The user's code at this vector address should perform any application-specific operations required and clear interrupt flags, SPIxIF, SPIxTXIF or SPIxRXIF, and then exit.

With Enhanced Buffering mode, the user application should clear the interrupt request flag after servicing the interrupt condition .

If an SPIx interrupt has occurred, the ISR should read the SPIx Data Buffer (SPIxBUF) register and then clear the SPIx interrupt flag.

19.6 Operation in Power-Saving and Debug Modes

19.6.1 Sleep Mode

When the device enters Sleep mode, the system clock is disabled. The exact SPI module operation during Sleep mode depends on the current mode of operation. The following sub-sections describe mode-specific behavior.

19.6.1.1 Host Mode in Sleep Mode

The following items should be noted in Sleep mode:

- The Baud Rate Generator (BRG) is stopped and may be reset.
- On-going transmission and reception sequences are aborted. The module may not resume aborted sequences when Sleep mode is exited.
- Once in Sleep mode, the module will not transmit or receive any new data.

Note: To prevent unintentional abort of transmit and receive sequences, wait for the current transmission to be completed before activating Sleep mode.

19.6.1.2 Client Mode in Sleep Mode

In Client mode, the SPI module operates from the SCKx provided by an external SPI host. Since the clock pulses at SCKx are externally provided for Client mode, the module will continue to function in Sleep mode. It will complete any transactions during the transition into Sleep. On completion of a transaction, the SPIRBF flag is set. Consequently, the SPIxRXIF bit will be set.

If SPIx interrupts are enabled (SPIxRXIE = 1) and the SPI Interrupt Priority Level is greater than the present CPU priority level, the device will wake from Sleep mode and the code execution will resume at the SPIx interrupt vector location. If the SPI Interrupt Priority Level is lower than or equal to the present CPU priority level, the CPU will remain in Idle mode.

The module is not reset on entering Sleep mode if it is operating as a client device. Register contents are not affected when the SPI module is going into or coming out of Sleep mode.

19.6.2 Idle Mode

When the device enters Idle mode, the system clock sources remain functional.

19.6.2.1 Host Mode in Idle Mode

Bit, SPISIDL (SPIxCON1[13]), selects whether the module will stop or continue functioning in Idle mode.

- If SPISIDL = 1, the module will discontinue operation in Idle mode. The module will perform the same procedures when stopped in Idle mode that it does for Sleep mode.
- If SPISIDL = 0, the module will continue operation in Idle mode.

19.6.2.2 Client Mode in Idle Mode

The module will continue operation in Idle mode irrespective of the SPISIDL bit setting. The behavior is identical to the one in Sleep mode.

19.6.3 Debug Mode

19.6.3.1 Operation of SPIxBUF

19.6.3.1.1 Reads During Debug Mode

During Debug mode, SPIxBUF can be read, but the read operation does not affect any status bits. For example, if the SPIRBF bit (SPIxSTAT[0]) is set when Debug mode is entered, it will remain set on an exit from Debug mode, even though the SPIxBUF register was read in Debug mode.

20. Inter-Integrated Circuit (I^2C)

The Inter-Integrated Circuit (I^2C) module is a serial interface useful for communicating with other peripheral or microcontroller (MCU) devices. The external peripheral devices may be serial EEPROMs, display drivers, Analog-to-Digital Converters (ADC) and so on.

The I^2C module can operate as any one of the following in the I^2C system:

- Client Device
- Host Device in a Single Host System (Client May Be Active)
- Host or Client Device in a Multi-Host System (Bus Collision Detection and Arbitration are Available)

Key features of the I^2C module include the following:

- Independent Host and Client Logic
- Supports 100 kHz, 400 kHz and 1MHz Bus Specifications
- 7-bit and 10-bit Device Addresses.
- Client Mode Can be Configured for:
  - Two unique addresses
  - Range of address
  - General call address
  - Address bit masking
- Automatic Clock Stretching Provides Delays for the Processor to Respond to a Client Data Request
- Multi-Host Support Which Prevents Message Losses in Arbitration
- Smart Mode for Minimal User Interaction and Simpler Application Code
- Supports Data Hold Time for SMBus (300 nS or 150 nS) in Client Mode
- Supports SMBus v2.0 and v3.0 Input Voltage Levels.
- Supports the Intelligent Platform Management Interface (IPMI) Standard
- System Management Bus (SMBus) and Power Management Bus (PMBus) Support:
  - Packet Error Checking (PEC) using CRC-8 calculator
  - Clock low timeout, Bus idle timeout and cumulative timeout
  - Frame error detection

20.1 Device-Specific Information

Table 20-1. I^2C Summary Table

I^2C Module Instances	Clock Source	Peripheral Bus Speed
2	Standard Speed Peripheral Clock	Standard

20.2 Architectural Overview

The I^2C bus is a 2-wire serial interface. Figure 20-1 illustrates the schematic of an I^2C connection between a dsPIC33AK128MC106 device and a 24LC256 I^2C serial EEPROM, which is a typical example for any I^2C interface.

The I^2C interface uses a comprehensive protocol to ensure reliable transmission and reception of the data. When communicating, one device acts as the “host” and initiates transfer on the bus and generates the clock signals to permit that transfer, while the other devices act as the “client” responding to the transfer. The clock line, SCLx, is output from the host and input to the client,

although occasionally the client drives the SCLx line. The data line, SDAx, may be output and input from both the host and client.

Because the SDAx and SCLx lines are bidirectional, the output stages of the devices driving the SDAx and SCLx lines must have an open-drain in order to perform the wired-AND function of the bus. External pull-up resistors are used to ensure a high level when no device is pulling the line down.

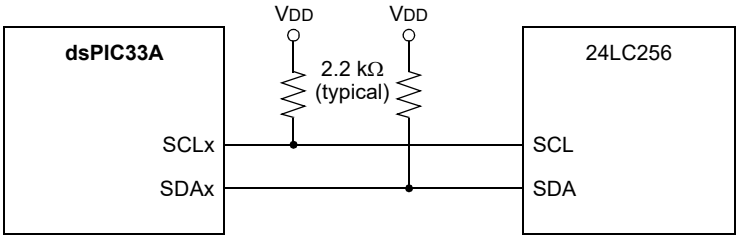
Note: SCLx and SDAx must be configured as digital.

In the I²C interface protocol, each device has an address. When a host needs to initiate a data transfer, it first transmits the address of the device that it wants to "communicate". All of the devices "listen" to see if this is their address. Within this address, bit 0 specifies whether the host wants to read from or write to the client device. The host and client are always in opposite modes (Transmitter or Receiver) of operation during a data transfer. That is, they operate in either of the following two relationships:

- Host-transmitter and client-receiver
- Client-transmitter and host-receiver

In both cases, the host originates the SCLx clock signal.

Figure 20-1. Typical I²C Interconnection Block Diagram



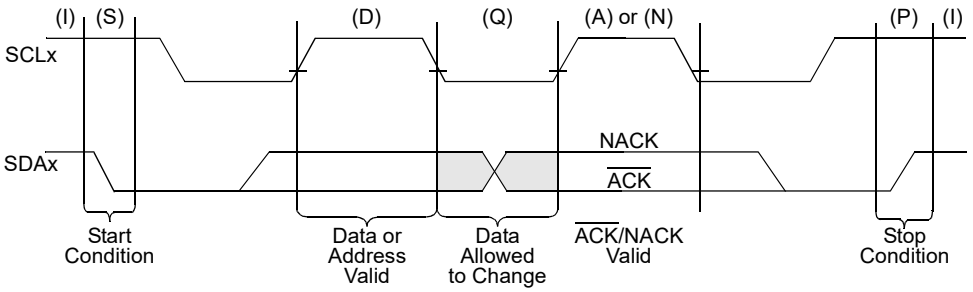
20.2.1 Bus Protocol

The following I²C bus protocol has been defined:

- The data transfer may be initiated only when the bus is not busy.
- During the data transfer, the data line must remain stable whenever the SCLx clock line is high. Any changes in the data line while the SCLx clock line is high will be interpreted as a Start or Stop condition.

Accordingly, the bus conditions are defined as illustrated in Figure 20-2.

Figure 20-2. I²C Bus Protocol States



20.2.1.1 Start Data Transfer (S)

After a bus Idle state, a high-to-low transition of the SDAx line while the clock (SCLx) is high determines a Start condition. All data transfers must be preceded by a Start condition.

20.2.1.2 Stop Data Transfer (P)

A low-to-high transition of the SDAx line while the clock (SCLx) is high determines a Stop condition. All data transfers must end with a Stop condition.

20.2.1.3 Repeated Start (R)

After a Wait state, a high-to-low transition of the SDAx line while the clock (SCLx) is high determines a Repeated Start condition. Repeated Starts allow a host to change bus direction or address a client device without relinquishing control of the bus.

20.2.1.4 Data Valid (D)

After a Start condition, the state of the SDAx line represents valid data when the SDAx line is stable for the duration of the high period of the clock signal. There is one bit of data per SCLx clock.

20.2.1.5 Acknowledge (A) or Not Acknowledge (N)

All data byte transmissions must be Acknowledged ($\overline{ACK}$) or Not Acknowledged (NACK) by the receiver. The receiver will pull the SDAx line low for an $\overline{ACK}$ or release the SDAx line for a NACK. The Acknowledge is a 1-bit period using one SCLx clock.

20.2.1.6 Wait/Data Invalid (Q)

The data on the line must be changed during the low period of the clock signal. The devices may also stretch the clock low time by asserting a low on the SCLx line, causing a Wait on the bus.

20.2.1.7 Bus Idle (I)

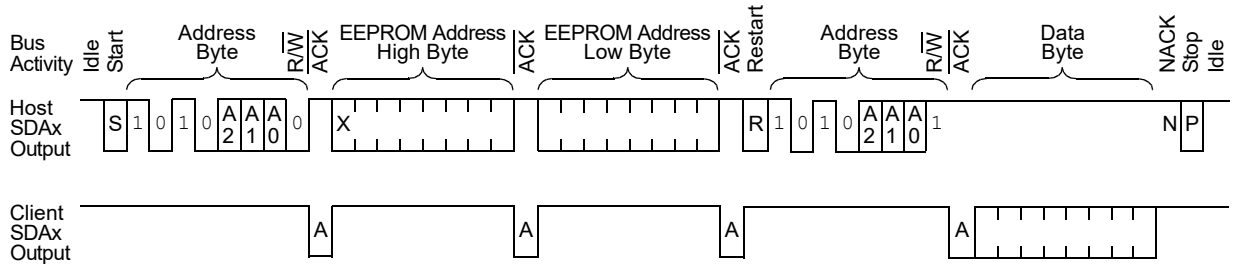
Both data and clock lines remain high after a Stop condition and before a Start condition.

20.2.2 Message Protocol

A typical I^{2}C message is illustrated in Figure 20-3. In this example, the message will read a specified byte from a 24LC256 I^{2}C serial EEPROM. The I^{2}C device will act as the host and the 24LC256 device will act as the client.

Figure 20-3 illustrates the data as driven by the host device and the client device, taking into account that the combined SDAx line is a wired-AND of the host and client data. The host device controls and sequences the protocol. The client device will only drive the bus at specifically determined times.

Figure 20-3. A Typical I^{2}C Message: Read of Serial EEPROM (Random Address Mode)



20.2.2.1 Start Message

Each message is initiated with a Start condition and terminated with a Stop condition. The number of data bytes transferred between the Start and Stop conditions is determined by the host device. As defined by the system protocol, the bytes of the message may have special meaning, such as the device address byte or the data byte.

20.2.2.2 Address Client

In Figure 20-3, the first byte is the device address byte, which must be the first part of any I^{2}C message. It contains a device address and a R/W status bit. Note that R/W = 0 for this first address byte, indicating that the host will be a transmitter and the client will be a receiver.

20.2.2.3 Client Acknowledge

The receiving device is obliged to generate an Acknowledge signal, ACK, after the reception of each byte. The host device must generate an extra SCLx clock, which is associated with this Acknowledge bit.

20.2.2.4 Host Transmit

The next two bytes, sent by the host to the client, are data bytes that contain the location of the requested EEPROM data byte. The client must Acknowledge each of the data bytes.

20.2.2.5 Repeated Start

The client EEPROM has the required address information that is required to return the requested data byte to the host. However, the R/W status bit from the first device address byte specifies the host transmission and the client reception. The direction of the bus must be reversed for the client to send data to the host.

To perform this function without ending the message, the host sends a Repeated Start. The Repeated Start is followed with a device address byte containing the same device address as before, and with R/W = 1, to indicate the client transmission and the host reception.

20.2.2.6 Client Reply

The client transmits the data byte by driving the SDAx line, while the host continues to originate clocks but releases its SDAx drive.

20.2.2.7 Host Acknowledge

During reads, a host must terminate data requests to the client by generating a NACK on the last byte of the message.

20.2.2.8 Stop Message

The host sends a Stop signal to terminate the message and returns the bus to an Idle state.

20.3 I2C System Overview

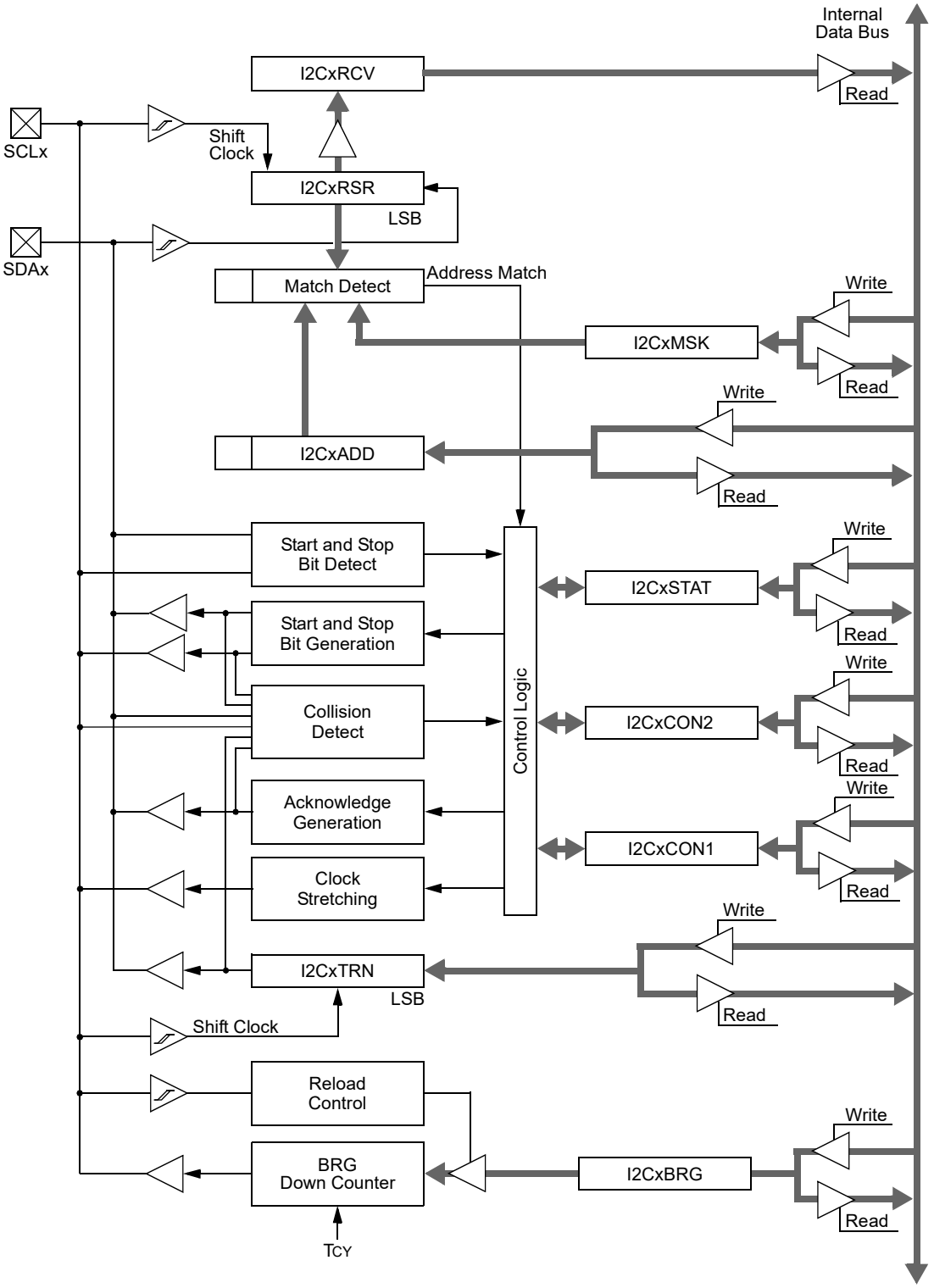
The I^2C module contains an independent I^2C host logic and an I^2C client logic, which generates interrupts based on their events. In the multi-host systems, the user software is simply partitioned into the host controller and the client controller.

When the I^2C host logic is active, the client logic also remains active, detecting the state of the bus and potentially receiving messages from itself in a single host system or from the other hosts in a multi-host system. No messages are lost during the multi-host bus arbitration.

In a multi-host system, the bus collision conflicts with the other hosts in the system when detected, and the module provides a method to terminate and then restart the message.

The I^2C module contains a Baud Rate Generator (BRG). The I^2C BRG does not consume other timer resources in the device. Figure 20-4 illustrates the I^2C module block diagram.

Figure 20-4. I²C Block Diagram



20.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1880	I2C1CON1	31:24							SMBEN[1:0]	
		23:16		PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
		15:8	ON		I2CSIDL	SCLREL	STRICT	A10M	DISSLW	
		7:0	GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
0x1884	I2C1STAT1	31:24								
		23:16								
		15:8	ACKSTAT	TRSTAT	ACKTIM			BCL	GCSTAT	ADD10
		7:0	IWCOL	I2COV	D/A	P	S	R/W	RBF	TBF
0x1888	I2C1ADD	31:24								
		23:16								
		15:8							ADD[9:8]	
		7:0	ADD[7:0]							
0x188C	I2C1MSK	31:24								
		23:16								
		15:8							MSK[9:8]	
		7:0	MSK[7:0]							
0x1890	I2C1HBRG	31:24								
		23:16	I2C1HBRG[23:0]							
		15:8	I2C1HBRG[23:0]							
		7:0	I2C1HBRG[23:0]							
0x1894	I2C1TRN	31:24								
		23:16								
		15:8								
		7:0	I2CTXDATA[7:0]							
0x1898	I2C1RCV	31:24								
		23:16								
		15:8								
		7:0	I2CRXDATA[7:0]							
0x189C	I2C1CON2	31:24	AMODE[1:0]		PECC[1:0]		BSCLTE	HBCTE	CBCTE	EPSZE
		23:16	ACKC[1:0]		HNACKIGN	EOPSC[1:0]		ND/$\overline{A}$	SMEN	BITE
		15:8	PSZ[15:0]							
		7:0	PSZ[15:0]							
0x18A0	I2C1LBRG	31:24								
		23:16	I2C1LBRG[23:0]							
		15:8	I2C1LBRG[23:0]							
		7:0	I2C1LBRG[23:0]							
0x18A4	I2C1INTC	31:24	I2CEIE		CBCLIE	HPCIE	HSCIE	HBCLIE	EOPIE	
		23:16	BSCLTIE	HBCTIE	CBCTIE	BITIE	FRMEIE	NACKIE		CRCIE
		15:8	BCLDIE	HSBCLIE	HSTIE	CLTIE	HACKSIE	CADDRIE		
		7:0	RXIE	TXIE		CDRXIE	CDTXIE		HDTXIE	HDRXIE
0x18A8	I2C1STAT2	31:24	SSPND	CLTACT	HSTACT				HSBCL	EOP
		23:16	BSCLTO	HBCLTO	CBCLTO	BITO	FRME	NACKE		CRC
		15:8	STOPE	STARTE	HSTIF	CLTIF	ERR			
		7:0					SCLCNT[3:0]			
0x18AC	I2C1PEC	31:24								
		23:16								
		15:8	CCRC[7:0]							
		7:0	RCRC[7:0]							
0x18B0	I2C1BTO	31:24	BTOCLKSEL							
		23:16	BSCLTOTMR[23:16]							
		15:8	BSCLTOTMR[15:8]							
		7:0	BSCLTOTMR[7:0]							
0x18B4	I2C1HBCTO	31:24								
		23:16	HBCTOTMR[23:16]							
		15:8	HBCTOTMR[15:8]							
		7:0	HBCTOTMR[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x18B8	I2C1CBCTO	31:24								
		23:16	CBCTOTMR[23:16]							
		15:8	CBCTOTMR[15:8]							
		7:0	CBCTOTMR[7:0]							
0x18BC	I2C1BITO	31:24								
		23:16	BITOTMR[23:16]							
		15:8	BITOTMR[15:8]							
		7:0	BITOTMR[7:0]							
0x18C0	I2C1SDASUT	31:24	SDASUTEN							
		23:16								
		15:8	SDASUT[15:8]							
		7:0	SDASUT[7:0]							
0x18C4 ... 0x18CF	Reserved									
0x18D0	I2C2CON1	31:24							SMBEN[1:0]	
		23:16		PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
		15:8	ON		I2CSIDL	SCLREL	STRICT	A10M	DISSLW	
		7:0	GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
0x18D4	I2C2STAT1	31:24								
		23:16								
		15:8	ACKSTAT	TRSTAT	ACKTIM			BCL	GCSTAT	ADD10
		7:0	IWCOL	I2COV	D/A	P	S	R/W	RBF	TBF
0x18D8	I2C2ADD	31:24								
		23:16								
		15:8							ADD[9:8]	
		7:0	ADD[7:0]							
0x18DC	I2C2MSK	31:24								
		23:16								
		15:8							MSK[9:8]	
		7:0	MSK[7:0]							
0x18E0	I2C2HBRG	31:24								
		23:16	I2C2HBRG[23:0]							
		15:8	I2C2HBRG[23:0]							
		7:0	I2C2HBRG[23:0]							
0x18E4	I2C2TRN	31:24								
		23:16								
		15:8								
		7:0	I2CTXDATA[7:0]							
0x18E8	I2C2RCV	31:24								
		23:16								
		15:8								
		7:0	I2CRXDATA[7:0]							
0x18EC	I2C2CON2	31:24	AMODE[1:0]		PECC[1:0]		BSCLTE	HBCTE	CBCTE	EPSZE
		23:16	ACKC[1:0]		HNACKIGN	EOPSC[1:0]		ND/$\overline{A}$	SMEN	BITE
		15:8	PSZ[15:0]							
		7:0	PSZ[15:0]							
0x18F0	I2C2LBRG	31:24								
		23:16	I2C2LBRG[23:0]							
		15:8	I2C2LBRG[23:0]							
		7:0	I2C2LBRG[23:0]							
0x18F4	I2C2INTC	31:24	I2CEIE		CBCLIE	HPCIE	HSCIE	HBCLIE	EOPIE	
		23:16	BSCLTIE	HBCTIE	CBCTIE	BITIE	FRMEIE	NACKIE		CRCIE
		15:8	BCLDIE	HSBCLIE	HSTIE	CLTIE	HACKSIE	CADDRIE		
		7:0	RXIE	TXIE		CDRXIE	CDTXIE		HDTXIE	HDRXIE
0x18F8	I2C2STAT2	31:24	SSPND	CLTACT	HSTACT				HSBCL	EOP
		23:16	BSCLTO	HBCLTO	CBCLTO	BITO	FRME	NACKE		CRC
		15:8	STOPE	STARTE	HSTIF	CLTIF	ERR			
		7:0					SCLCNT[3:0]			

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x18FC	I2C2PEC	31:24								
		23:16								
		15:8	CCRC[7:0]							
		7:0	RCRC[7:0]							
0x1900	I2C2BTO	31:24	BTOCLKSEL							
		23:16	BSCLTOTMR[23:16]							
		15:8	BSCLTOTMR[15:8]							
		7:0	BSCLTOTMR[7:0]							
0x1904	I2C2HBCTO	31:24								
		23:16	HBCTOTMR[23:16]							
		15:8	HBCTOTMR[15:8]							
		7:0	HBCTOTMR[7:0]							
0x1908	I2C2CBCTO	31:24								
		23:16	CBCTOTMR[23:16]							
		15:8	CBCTOTMR[15:8]							
		7:0	CBCTOTMR[7:0]							
0x190C	I2C2BITO	31:24								
		23:16	BITOTMR[23:16]							
		15:8	BITOTMR[15:8]							
		7:0	BITOTMR[7:0]							
0x1910	I2C2SDASUT	31:24	SDASUTEN							
		23:16								
		15:8	SDASUT[15:8]							
		7:0	SDASUT[7:0]							

20.4.1 I²Cx Control Register

Name: I2CxCON1
Offset: 0x1880, 0x18D0

Notes:

1. Automatically cleared to '0' at the beginning of client transmission; automatically cleared to '0' at the end of client reception.
2. Automatically cleared to '0' at the beginning of client transmission.
3. This bit must be set before any I²C operations can be performed. This bit should be set in a separate instruction from any of the other enable bits.
4. 16 peripheral clock cycles should be waited before performing any operation.

Legend: HC = Hardware Clearable bit

Bit	31	30	29	28	27	26	25	24
							SMBEN[1:0]	
Access							R/W	R/W
Reset							0	0

Bit	23	22	21	20	19	18	17	16
		PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		I2CSIDL	SCLREL	STRICT	A10M	DISSLW	
Access	R/W		R/W	R/W/HC	R/W	R/W	R/W	
Reset	0		0	1	0	0	0	

Bit	7	6	5	4	3	2	1	0
	GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
Access	R/W	R/W	R/W	R/W/HC/HS	R/W/HC/HS	R/W/HC	R/W/HC	R/W/HC
Reset	0	0	0	0	0	0	0	0

Bits 25:24 – SMBEN[1:0] SMBus Input Levels Enable bit

Value	Description
11	Reserved
10	Enable SMBus 3.0 input threshold voltage specification
01	Enable SMBus 2.0 input threshold voltage specification
00	Enable I2C input threshold voltage specification (disable SMBus specific input)

Bit 22 – PCIE Stop Condition Interrupt Enable bit (Client mode only)

Value	Description
1	Enables interrupt on detection of Stop condition
0	Stop detection interrupts are disabled

Bit 21 – SCIE Start Condition Interrupt Enable bit (Client mode only)

Value	Description
1	Enables interrupt on detection of Start or Restart conditions
0	Start detection interrupts are disabled

Bit 20 – BOEN Buffer Overwrite Enable bit (Client mode only)

Value	Description
1	I2CxRCV is updated and an ACK is generated for a received address/data byte, ignoring the state of the I2COV bit only if RBF bit = 0
0	I2CxRCV is only updated when I2COV is clear

Bit 19 – SDAHT SDAx Hold Time Selection bit

Value	Description
1	Minimum of 300 ns hold time on SDAx after the falling edge of SCLx
0	Minimum of 100 ns hold time on SDAx after the falling edge of SCLx

Bit 18 – SBCDE Mode Bus Collision Detect Enable bit (Client mode only)

If, on the rising edge of SCLx, SDAx is sampled low when the module is outputting a high state, the BCL bit is set and the bus goes Idle. This Detection mode is only valid during data and ACK transmit sequences.

Value	Description
1	Enables client bus collision interrupts
0	Client bus collision interrupts are disabled

Bit 17 – AHEN Client Address Hold Enable bit

Value	Description
1	Following the eighth falling edge of SCLx for a matching received address byte, the SCLREL bit (I2CxCON1[12]) will be cleared and the SCLx will be held low
0	Address holding is disabled

Bit 16 – DHEN Client Data Hold Enable bit

Value	Description
1	Following the eighth falling edge of SCLx for a received data byte, client hardware clears the SCLREL bit (I2CxCON1[12]) and SCLx is held low
0	Data holding is disabled

Bit 15 – ON I^2Cx Enable bit (writable from software only)**(3,4)**

Value	Description
1	Enables the I2Cx module and configures the SDAx and SCLx pins as serial port pins
0	Disables the I2Cx module; all I^2C pins are controlled by port functions

Bit 13 – I2CSIDL I^2Cx Stop in Idle Mode bit

Value	Description
1	Discontinues module operation when device enters Idle mode
0	Continues module operation in Idle mode

Bit 12 – SCLREL SCLx Release Control bit (I^2C Client mode only)**(1)**

If STREN = 1:**(2)**

User software may write '0' to initiate a clock stretch and write '1' to release the clock. Hardware clears at the beginning of every client data byte transmission. Hardware clears at the end of every client address byte reception. Hardware clears at the end of every client data byte reception.

If STREN = 0:

User software may only write '1' to release the clock. Hardware clears at the beginning of every client data byte transmission. Hardware clears at the end of every client address byte reception.

Value	Description
1	Releases the SCLx clock

Value	Description
0	Holds the SCLx clock low (clock stretch)

Bit 11 – STRICT I²Cx Strict Reserved Address Rule Enable bit

Value	Description
1	Strict reserved addressing is enforced for reserved addresses. (In Client mode) – The device doesn't respond to reserved address space and addresses falling in that category are NACKed. (In Host mode) – The device is allowed to generate addresses with reserved address space.
0	Reserved addressing would be Acknowledged. (In Client mode) – The device will respond to an address falling in the reserved address space. When there is a match with any of the reserved addresses, the device will generate an ACK. (In Host mode) – Reserved.

Bit 10 – A10M 10-Bit Client Address Flag bit

Value	Description
1	I2CxADD is a 10-bit client address
0	I2CxADD is a 7-bit client address

Bit 9 – DISSLW Slew Rate Control Disable bit

Value	Description
1	Slew rate control is disabled for Standard Speed mode (100 kHz, also disabled for 1 MHz mode)
0	Slew rate control is enabled for High-Speed mode (400 kHz)

Bit 7 – GCEN General Call Enable bit (in I²C Client mode only)

Value	Description
1	Enables interrupt when a general call address is received in I2CxRSR; module is enabled for reception
0	General call address is disabled.

Bit 6 – STREN SCLx Clock Stretch Enable bit
In I²C Client mode only; used in conjunction with the SCLREL bit.

Value	Description
1	Enables clock stretching
0	Disables clock stretching

Bit 5 – ACKDT Acknowledge Data bit
In I²C Host mode during Host Receive mode. The value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
In I²C Client mode when AHEN = 1 or DHEN = 1. The value that the client will transmit when it initiates an Acknowledge sequence at the end of an address or data reception.

Value	Description
1	NACK is sent
0	ACK is sent

Bit 4 – ACKEN Acknowledge Sequence Enable bit
In I²C Host mode only; applicable during Host Receive mode.

Value	Description
1	Initiates Acknowledge sequence on SDAx and SCLx pins and transmits ACKDT data bit
0	Acknowledge sequence is Idle

Bit 3 – RCEN Receive Enable bit (in I²C Host mode only)

Value	Description
1	Enables Receive mode for I²C; automatically cleared by hardware at end of 8-bit receive data byte
0	Receive sequence is not in progress

Bit 2 – PEN Stop Condition Enable bit (in I²C Host mode only)

Value	Description
1	Initiates Stop condition on SDAx and SCLx pins
0	Stop condition is Idle

Bit 1 – RSEN Restart Condition Enable bit (in I²C Host mode only)

Value	Description
1	Initiates Restart condition on SDAx and SCLx pins
0	Restart condition is Idle

Bit 0 – SEN Start Condition Enable bit (in I²C Host mode only)

Value	Description
1	Initiates Start condition on SDAx and SCLx pins
0	Start condition is Idle

20.4.2 I²Cx Status Register

Name: I2CxSTAT1
Offset: 0x1884, 0x18D4

Legend: C = Clearable bit, HS = Hardware Settable bit, HSC = Hardware Settable/Clearable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ACKSTAT	TRSTAT	ACKTIM			BCL	GCSTAT	ADD10
Access	R/HSC	R/HSC	R/HSC			R/C/HSC	R/HSC	R/HSC
Reset	0	0	0			0	0	0

Bit	7	6	5	4	3	2	1	0
	IWCOL	I2COV	D/A	P	S	R/W	RBF	TBF
Access	R/C/HS	R/C/HS	R/HSC	R/HSC	R/HSC	R/HSC	R/HSC	R/HSC
Reset	0	0	0	0	0	0	0	0

Bit 15 – ACKSTAT Acknowledge Status bit (updated in all Host and Client modes)

Value	Description
1	Acknowledge was not received from client
0	Acknowledge was received from client

Bit 14 – TRSTAT Transmit Status bit (when operating as I²C host; applicable to host transmit operation)

Value	Description
1	Host transmit is in progress (eight bits + ACK)
0	Host transmit is not in progress

Bit 13 – ACKTIM Acknowledge Time Status bit (valid in I²C Client mode only)

Value	Description
1	Indicates I²C bus is in an Acknowledge sequence, set on eighth falling edge of SCLx clock
0	Not an Acknowledge sequence, cleared on ninth rising edge of SCLx clock

Bit 10 – BCL Bus Collision Detect bit
(Host/Client mode; cleared when I²C module is disabled, I2CEN = 0)

Value	Description
1	A bus collision has been detected during a host or client transmit operation
0	No bus collision has been detected

Bit 9 – GCSTAT General Call Status bit (cleared after Stop detection)

Value	Description
1	General call address was received
0	General call address was not received

Bit 8 – ADD10 10-Bit Address Status bit (cleared after Stop detection)

Value	Description
1	10-bit address was matched
0	10-bit address was not matched

Bit 7 – IWCOL I2Cx Write Collision Detect bit

Value	Description
1	An attempt to write to the I2CxTRN register failed because the I^2C module is busy; must be cleared in software
0	No collision

Bit 6 – I2COV I2Cx Receive Overflow Flag bit

Value	Description
1	A byte was received while the I2CxRCV register is still holding the previous byte
0	No overflow

Bit 5 – D/$\overline{A}$ Data/$\overline{\text{Address}}$ bit (when operating as I^2C Client)

Value	Description
1	Indicates that the last byte received was data
0	Indicates that the last byte received or transmitted was an address

Bit 4 – P Stop bit
Updated when Start, Reset or Stop is detected; cleared when the I^2C module is disabled, I2CEN = 0.

Value	Description
1	Indicates that a Stop bit has been detected last
0	Stop bit was not detected last

Bit 3 – S I2Cx Start bit
Updated when Start, Reset or Stop is detected; cleared when the I^2C module is disabled, I2CEN = 0.

Value	Description
1	Indicates that a Start (or Repeated Start) bit has been detected last
0	Start bit was not detected last

Bit 2 – R/$\overline{W}$ Read/$\overline{\text{Write}}$ Information bit (when operating as I^2C Client)

Value	Description
1	Read: Indicates the data transfer is output from the client
0	Write: Indicates the data transfer is input to the client

Bit 1 – RBF Receive Buffer Full Status bit

Value	Description
1	Receive is complete, I2CxRCV is full
0	Receive is not complete, I2CxRCV is empty

Bit 0 – TBF Transmit Buffer Full Status bit

Value	Description
1	Transmit is in progress, I2CxTRN is full (eight bits of data)
0	Transmit is complete, I2CxTRN is empty

20.4.3 I²Cx Client Address Register

Name: I2CxADD
Offset: 0x1888, 0x18D8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
							ADD[9:8]	
Access							R/W	R/W
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	ADD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 9:0 – ADD[9:0] Client Device Address bits

20.4.4 I²Cx Client Mode Address Mask Register

Name: I2CxMSK
Offset: 0x188C, 0x18DC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
							MSK[9:8]	
Access							R/W	R/W
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	MSK[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 9:0 – MSK[9:0] I²Cx Mask for Address Bit x Select bits

Value	Description
1	Enables masking for bit x of the incoming message address; bit match is not required in this position
0	Disables masking for bit x; bit match is required in this position

20.4.5 I²Cx SCL High Baud Rate Generator Register

Name: I2CxHBRG
Offset: 0x1890, 0x18E0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	I2CxHBRG[23:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	I2CxHBRG[23:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	I2CxHBRG[23:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – I2CxHBRG[23:0] I²Cx SCL High time Baud Rate Generator Value bits

20.4.6 I²Cx Transmit Data Register

Name: I2CxTRN
Offset: 0x1894, 0x18E4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	I2CTXDATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – I2CTXDATA[7:0] I²C Transmit Data Buffer bits

20.4.7 I²Cx Receive Data Register

Name: I2CxRCV
Offset: 0x1898, 0x18E8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	I2CRXDATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – I2CRXDATA[7:0] I²Cx Receive Data bits

20.4.8 I²Cx Control Register

Name: I2CxCON2
Offset: 0x189C, 0x18EC

Notes:

1. Automatically cleared to '0' at the beginning of client transmission; automatically cleared to '0' at the end of client reception.
2. Automatically cleared to '0' at the beginning of client transmission.
3. When EPSZE is enabled, Smart mode (SMEN=1), clock stretching (STREN=1) and EOP function (EOPSC= "10" or "01" should be enabled.
4. In Host mode , when ACKC != 00, hardware will automatically set ACKEN (I2CxCON1[4]) bit.
5. In Host mode, this bit will be used by PSZ(I2CxCON2[15:0]) to count the data bytes.
6. In Client mode, this bit is used for TX/RX interrupt generation.
   If '1', Interrupt is generated only for data bytes. If '0', Interrupt is generated for both address and data bytes.
7. In Client mode, it should set the ND/A to '1' before enabling transfers through DMA for data transfer.
8. The packet size should be excluding of address byte(s). It should not be changed on fly and should be changed when the bus is in Idle state.
9. In Host mode, the EOPSC and ND/A bit control the PSZ to decrement.
10. For the host, the software has to clear the CRC calculator by setting I2CxCON2.PECC to "11" for new data frame transaction.
11. To use the Auto-Append mode, the PECC needs to be set to "01" at least one data byte earlier than the CRC byte of data.

Legend: HC = Hardware Clearable bit

Bit	31	30	29	28	27	26	25	24
	AMODE[1:0]		PECC[1:0]		BSCLTE	HBCTE	CBCTE	EPSZE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ACKC[1:0]		HNACKIGN	EOPSC[1:0]		ND/$\overline{A}$	SMEN	BITE
Access	R/W	R/W	R/W	R/W/HC	R/W/HC	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PSZ[15:0]							
Access	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PSZ[15:0]							
Access	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC
Reset	0	0	0	0	0	0	0	1

Bits 31:30 – AMODE[1:0] Address Mode bit

Value	Description
11	Reserved
10	The client responds to the range of addresses between and including I2CxADD and I2CxMSK. I2CxMSK is the upper limit
01	The client responds to the two unique addresses in I2CxADD and I2CxMSK
00	I2CxMSK is used as a mask to the I2CxADD register

Bits 29:28 – PECC[1:0] PEC Control bit(10,11)

Value	Description
11	PEC reset
10	PEC append is disabled. CRC-8 calculator will be active. On a read request (receive), calculated CRC-8 is copied into CCRC (I2CxPEC[15:8]) at EOP. On a write request (transmit), calculated CRC-8 is copied into CCRC (I2CxPEC[15:8]) at EOP.
01	Calculated CRC-8 is appended at the end of a packet. On a read request (receive), calculated CRC-8 is copied into CCRC (I2CxPEC[15:8]) and received CRC will be copied into RCRC (I2CxPEC[7:0]) at EOP. On a write Request (transmit), calculated CRC-8 will be automatically appended and also copied into CCRC (I2CxPEC[15:8]) at the end of the data transmission. PEC will be get reset after appending.
00	PEC disabled

Bit 27 – BSCLTE Bus SCL Time-out Enable bit

Value	Description
1	SCL low time-out enabled
0	SCL low time-out disabled

Bit 26 – HBCTE Host Bus SCL Cumulative (Extended time) Low Time-out Enable bit

Value	Description
1	SCL Cumulative low extended time-out enable
0	SCL Cumulative low extended time-out disable

Bit 25 – CBCTE Client Bus SCL Cumulative (Extended time) Low Time-out Enable bit

Value	Description
1	SCL Cumulative low extended time-out enable
0	SCL Cumulative low extended time-out disable

Bit 24 – EPSZE Extended Packet Size Enable bit (Valid for Client Receive mode only)(3)

Value	Description
1	Extended packet size enabled after EOP=1
0	Extended packet size disable

Bits 23:22 – ACKC[1:0] ACK Control bit(4)

Value	Description
11	Host: ACK all the bytes except the CRC byte; for the CRC byte, NACK will be sent Client: ACK all the bytes and for the last byte, which is the CRC byte, ACK or NACK based on CRC result • If the CRC is 0, an ACK is sent • If the CRC is non-zero, a NACK is sent (to be used only on PECC[1:0] = "01" Auto-Append mode)
10	ACK all the bytes and the last byte will be NACKed (to be used only for last packet)
01	ACK all bytes including end of packet (to be used for extended packets)
00	ACK/NACK based on ACKDT and BOEN (client) ACK/NACK based on ACKEN and ACKDT (host)

Bit 21 – HNACKIGN Host NACK Response Ignore Control bit

Value	Description
1	Host treats all NACK responses as ACK
0	Normal operation; host treats NACK responses as NACK only

Bits 20:19 – EOPSC[1:0] End of Packet Status Control bit

Value	Description
11	Reserved
10	I2CxSTAT2.EOP will be set after data bytes and PEC
01	I2CxSTAT2.EOP will be set after data bytes
00	End of packet function is disabled

Bit 18 – ND/$\overline{A}$ Next Data/Address Byte Transmission bit(5,6,7)

Value	Description
1	Next transmission is a data byte transmission
0	Next transmission is an address byte transmission

Bit 17 – SMEN Smart Mode Enable bit

Value	Description
1	Smart mode is enable
0	Smart mode is disabled

Bit 16 – BITE Bus Idle Time-out Enable bit

Value	Description
1	Bus idle time-out enable
0	Bus idle time-out disabled

Bits 15:0 – PSZ[15:0] Packet Size bits(8,9)

Sets the size of the packet to transfer/receive; the valid range is from 0 to 65535 bytes. Use I2CxCONC.PSZ=0 for the SMBus Quick Command protocol.

20.4.9 I²Cx SCL Low Baud Rate Generator Register

Name: I2CxLBRG
Offset: 0x18A0, 0x18F0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	I2CxLBRG[23:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	I2CxLBRG[23:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	I2CxLBRG[23:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – I2CxLBRG[23:0] I²Cx SCL Low time Baud Rate Generator Value bits

20.4.10 I²Cx Interrupt Control Register

Name: I2CxINTC
Offset: 0x18A4, 0x18F4

Bit	31	30	29	28	27	26	25	24
	I2CEIE		CBCLIE	HPCIE	HSCIE	HBCLIE	EOPIE	
Access	R/W		R/W	R/W	R/W	R/W	R/W	
Reset	0		0	0	0	0	0	

Bit	23	22	21	20	19	18	17	16
	BSCLTIE	HBCTIE	CBCTIE	BITIE	FRMEIE	NACKIE		CRCIE
Access	R/W	R/W	R/W	R/W	R/W	R/W		R/W
Reset	0	0	0	0	0	0		0

Bit	15	14	13	12	11	10	9	8
	BCLDIE	HSBCLIE	HSTIE	CLTIE	HACKSIE	CADDRIE		
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bit	7	6	5	4	3	2	1	0
	RXIE	TXIE		CDRXIE	CDTXIE		HDTXIE	HDRXIE
Access	R/W	R/W		R/W	R/W		R/W	R/W
Reset	0	0		0	0		0	0

Bit 31 – I2CEIE Error Interrupt Enable bit

Value	Description
1	Assert I2CIF if I2CEIF asserts. Merges error interrupts onto I2CIF. The ERR bit is not affected by setting this bit
0	Disabled

Bit 29 – CBCLIE Client Bus Collision Message Event Interrupt Enable bit (I²C Client mode only)

Value	Description
1	Enable CLTIF interrupt on bus collision detect
0	Bus collision interrupt is disabled

Bit 28 – HPCIE Host STOP Condition Interrupt Enable bit

Value	Description
1	Assert HSTIF when I2CxSTAT.P is set
0	Host STOP interrupt disabled

Bit 27 – HSCIE Host START Condition Interrupt Enable bit

Value	Description
1	Assert HSTIF when I2CxSTAT.S is set
0	Host START interrupt disabled

Bit 26 – HBCLIE Host Bus Collision Message event Interrupt Enable bit (I²C Host mode only)

Value	Description
1	Enable HSTIF interrupt on bus collision detect
0	Bus collision interrupt is disabled

Bit 25 – EOPIE End of Packet Interrupt Enable bit

Value	Description
1	End of packet interrupt is enabled
0	End of packet interrupt is disabled

Bit 23 – BSCLTIE Bus SCL Low Time-out Enable bit

Value	Description
1	Assert I2CEIF when I2CxSTAT2.BSCLTO is set
0	Bus SCL time-out not enabled

Bit 22 – HBCTIE Host Bus Cumulative Time-out Interrupt Enable bit

Value	Description
1	Assert I2CEIF when I2CxSTAT2.HBCLTO is set
0	Host bus cumulative time-out not enabled

Bit 21 – CBCTIE Client Bus Cumulative Time-out Interrupt Enable bit

Value	Description
1	Assert I2CEIF when I2CxSTAT2.CBCLTO is set
0	Client bus cumulative time-out not enabled

Bit 20 – BITIE Bus Idle Time-out Enable Interrupt bit

Value	Description
1	Assert I2CIF when I2CxSTAT2.BIT is set
0	Bus free interrupt is disabled

Bit 19 – FRMEIE Framing Error Detect Interrupt Enable bit

Value	Description
1	Enable I2CEIF interrupt if FRME is set due to a framing error. A framing error occurs in Client mode if a STOP or START is received during the byte transfer time or $\overline{\text{ACK}}$ transfer time
0	Frame error interrupt is disabled

Bit 18 – NACKIE NACK Detect Interrupt Enable bit

Value	Description
1	Enable I2CEIF interrupt on NACK detect as an Error
0	NACK interrupt is disabled

Bit 16 – CRCIE CRC Error Interrupt Enable bit

Value	Description
1	Enable I2CEIF interrupt on CRC error
0	CRC interrupts are disabled

Bit 15 – BCLDIE Bus Collision Detect Interrupt Enable bit

Value	Description
1	Enable I2CEIF interrupt on bus collision
0	Bus collision interrupts are disabled

Bit 14 – HSBCLIE Host Bus Collision Detect Interrupt Enable bit

Value	Description
1	Enable interrupt during the STOP sequence after receiving bus collision
0	An interrupt during the STOP sequence after receiving bus collision is disabled

Bit 13 – HSTIE Host Interrupt Enable bit

Value	Description
1	Assert I2CIF when I2CxSTAT2.HSTIFis set
0	Host interrupt is disabled

Bit 12 – CLTIE Client Interrupt Enable bit

Value	Description
1	Assert I2CIF when I2CxSTAT2.CLIIF is set
0	Client interrupt is disabled

Bit 11 – HACKSIE Host ACK Sequence Interrupt Enable bit

Value	Description
1	Enable HSTIF interrupt on ACK sequence
0	ACK sequence interrupt is disabled

Bit 10 – CADDRIE Client Address Transaction Interrupt Enable bit

Value	Description
1	Enable CLTIF interrupt on address detect
0	Address detect interrupt is disabled

Bit 7 – RXIE Receive Interrupt Enable bit

Value	Description
1	Enable the I2CRXIF interrupt when I2CxSTAT.RBF is set
0	Disable the I2CRXIF interrupt

Bit 6 – TXIE Transmit Interrupt Enable bit

Value	Description
1	Enable the I2CTXIF interrupt when I2CxSTAT.TBF is cleared
0	Disable the I2CTXIF interrupt

Bit 4 – CDRXIE Client Data Receive Buffer Full Interrupt Enable bit

Value	Description
1	Include the I2CxSTAT.RBF=1 for the CLTIF interrupt
0	Exclude the I2CxSTAT.RBF=1 for the CLTIF interrupt

Bit 3 – CDTXIE Host Data Transmit Buffer Empty Interrupt Enable bit

Value	Description
1	Include the host I2CxSTAT.TBF=0 for the HSTIF interrupt
0	Exclude the I2CxSTAT.TBF=0 for the HSTIF interrupt

Bit 1 – HDTXIE Host Data Transmit Buffer Empty Interrupt Enable bit

Value	Description
1	Include the host I2CxSTAT.TBF=0 for the HSTIF interrupt
0	Exclude the I2CxSTAT.TBF=0 for the HSTIF interrupt

Bit 0 – HDRXIE Host Data Receive Buffer Full Interrupt Enable bit

Value	Description
1	Include the I2CxSTAT.RBF=1 for the HSTIF interrupt
0	Exclude the I2CxSTAT.RBF=1 for the HSTIF interrupt

20.4.11 I²Cx Status Register

Name: I2CxSTAT2
Offset: 0x18A8, 0x18F8

Note:

1. Cleared by any Stop detected on the bus. Cleared by I2CxSTAT2.BSCLTO or I2CxSTAT.BCL conditions. Cleared by any address match failure in 10-bit Addressing mode. Cleared by any address match failure following a Restart detect. Cleared when NACK is either received or transmitted to client. Cleared when client goes to IDLE after ACK/NACK CRC byte.
2. Cleared when BCL is set. Cleared when Stop is sent. Cleared for I2CxSTAT2.BSCLTO condition, after the host successfully sends a STOP condition.
3. It is the user's responsibility to make sure the EOP = "0" to start new packet transmission.
4. ERRF is combined Error Status bit of I2CxSTAT2.BSCLTO + I2CxSTAT2.CBCTO + I2CxSTAT2.HBCTO + I2CxSTAT2.FRAME + I2CxSTAT2.CRC + I2CxSTAT1.BCL + I2CxSTAT2.NACKE.

Legend: C = Clearable bit, HS = Hardware Settable bit, HSC = Hardware Settable/Clearable bit

Bit	31	30	29	28	27	26	25	24
	SSPND	CLTACT	HSTACT				HSBCL	EOP
Access	R/HS/HC	R/HS/HC	R/HS/HC				R/C/HS	R/C/HS
Reset	0	0	0				0	0

Bit	23	22	21	20	19	18	17	16
	BSCLTO	HBCLTO	CBCLTO	BITO	FRME	NACKE		CRC
Access	R/C/HS	R/C/HS	R/C/HS	R/C/HS	R/C/HS	R/C/HS		R/C/HS
Reset	0	0	0	0	0	0		0

Bit	15	14	13	12	11	10	9	8
	STOPE	STARTE	HSTIF	CLTIF	ERR			
Access	R/C/HS	R/HSC	R/HSC	R/C/HSC	R/C/HSC			
Reset	0	0	0	0	0			

Bit	7	6	5	4	3	2	1	0
					SCLCNT[3:0]			
Access					R/W/HS	R/W/HS	R/W/HS	R/W/HS
Reset					0	0	0	0

Bit 31 – SSPND I²C Suspended bit

During a host transaction, I²C bus activity is suspended to allow the user to service the I²C interrupt. During a client transaction, the SCL line is pulled low to clock stretch until the user services the I²C interrupt.

Value	Description
1	The I²C is in a suspended state
0	The I²C is not in a suspended state

Bit 30 – CLTACT Client State Machine Active Status bit

Value	Description
1	Set after the eighth falling SCL edge of a received matching client address
0	Client is idle(1)

Bit 29 – HSTACT Host State Machine Active Status bit

Value	Description
1	Host mode state machine is active; set when host state machine asserts a Start on bus Under these conditions, the host function is enabled and a START/RESTART condition is sent on the I^2C bus followed by data from the transmit buffer
0	Host is Idle(2)

Bit 25 – HSBCL Host Stop Bus Collision Detect Flag bit

Value	Description
1	Host Stop Bus Collision is detected
0	Host Stop Bus Collision is not detected

Bit 24 – EOP End-of-Packet Flag bit(3)

Value	Description
1	End of packet condition detected
0	End of packet condition not detected

Bit 23 – BSCLTO Bus SCL Time-out Flag bit

Value	Description
1	SCL Bus Time-out occurred
0	SCL Bus Time-out has not occurred

Bit 22 – HBCLTO Host Bus Cumulative Extended Time-out Flag bit

Value	Description
1	Host Bus Cumulative Extended Time-out occurred
0	Host Bus Cumulative Time-out has not occurred

Bit 21 – CBCLTO Client Bus Cumulative Extended Time-out Flag bit

Value	Description
1	Client Bus Cumulative Extended Time-out occurred
0	Client Bus Cumulative Extended Time-out has not occurred

Bit 20 – BITO Bus Idle Time-out Flag bit

Value	Description
1	Bus Idle Time-out occurred
0	Bus Idle Time-out has not occurred

Bit 19 – FRME Frame Error Detect Flag bit

Value	Description
1	Frame error detected in Client mode, STOP or START is received during the data byte transfer or data ACK transfer time
0	Frame error has not been detected in Client mode

Bit 18 – NACKE NACK Detect Error Flag bit

Value	Description
1	NACK detected as an Error
0	NACK has not been detected as an Error

Bit 16 – CRC CRC Error Flag bit

Value	Description
1	CRC Error occurred
0	CRC Error has not occurred

Bit 15 – STOPE Stop Condition Detect Event flag bit

Value	Description
1	Stop Condition detected
0	Stop Condition has not been detected

Bit 14 – STARTE Start Condition Detect Event Flag bit

Value	Description
1	Start Condition event detected
0	Start Condition event not detected

Bit 13 – HSTIF Host Detect Interrupt Flag bit

Value	Description
1	Host interrupt detected
0	Host interrupt not detected

Bit 12 – CLTIF Client Detect Interrupt Flag bit

Value	Description
1	Client interrupt detected
0	Client interrupt not detected

Bit 11 – ERR Error Detect Flag bit(4)

Value	Description
1	Error is detected
0	Error is not detected

Bits 3:0 – SCLCNT[3:0] SCL Count bits
Number of SCL clocks to be stored for Frame Error

20.4.12 I²Cx PEC Register

Name: I2CxPEC
Offset: 0x18AC, 0x18FC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CCRC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RCRC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – CCRC[7:0] Calculated Packet Error Check CRC Value bit
Provides read back of the CRC-8 calculator output for the current transaction.

Bits 7:0 – RCRC[7:0] Received Packet Error Check CRC Value bit
Provides read back of the CRC-8 received CRC for the current transaction.

20.4.13 I²Cx Bus Clock Time-Out Register

Name: I2CxBTO
Offset: 0x18B0, 0x1900

Bit	31	30	29	28	27	26	25	24
	BTOCLKSEL							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
	BSCLTOTMR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BSCLTOTMR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BSCLTOTMR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – BTOCLKSEL I²C Bus Time-out Clock Selection bit

Value	Description
1	The I²C BTO clock is the FRC
0	The I²C BTO clock is the peripheral clock

Bits 23:0 – BSCLTOTMR[23:0] I²C Bus Clock Time-out Timer bits
Provides the Reset value for the BSCLTO timer. The timer resets on each falling edge of SCL and starts counting each BTO clock cycle if SCL is low. The timer is disabled when BSCLTE=0.

20.4.14 I²Cx Host Bus Cumulative Time-out Register

Name: I2CxHBCTO
Offset: 0x18B4, 0x1904

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	HBCTOTMR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HBCTOTMR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HBCTOTMR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – HBCTOTMR[23:0] I²C Host Cumulative Bus Time-out Timer bits
Provides the Reset value for the HCBCTO timer. The timer resets on detection of a START or ACK or STOP and continued to run when SCL low is extended and HSTACT=1. The timer is disabled when HBCTE=0.

20.4.15 I²Cx Client Bus Cumulative Time-out Register

Name: I2CxCBCTO
Offset: 0x18B8, 0x1908

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	CBCTOTMR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CBCTOTMR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CBCTOTMR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – CBCTOTMR[23:0] I²C Client Cumulative Bus Time-out Timer bits
Provides the Reset value for the CBCTO timer. The timer resets on detection of a START or STOP and continues to run when SCL is low. The timer is disabled when CBCTE=0.

20.4.16 I²Cx Bus Idle/Free Time-Out Register

Name: I2CxBITO
Offset: 0x18BC, 0x190C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	BITOTMR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BITOTMR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BITOTMR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – BITOTMR[23:0] I²C Bus Idle (inactive/Free) Time-out Timer bit

20.4.17 I²Cx Bus SDA Set-Up Time Register

Name: I2CxSDASUT
Offset: 0x18C0, 0x1910

Notes:

1. Timer runs with peripheral clock.
2. This provides a delay between writing to the transmit buffer and releasing the SCL by hardware; it is used for the release of SCL by hardware.
3. In Smart mode, the SDA set up timer is used by hardware to release SCL.

Bit	31	30	29	28	27	26	25	24
	SDASUTEN							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	SDASUT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SDASUT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – SDASUTEN I²C Bus SDA-to-SCL Set-up Time Enable bit

Bits 15:0 – SDASUT[15:0] I²C Bus SDA-to-SCL Set-up Time Timer bits

20.5 Operation

20.5.1 Enabling I²C Operation

The I²C module is enabled by setting the I2CEN bit (I2CxCON1[15]. The I²C module fully implements all host and client functions. When the module is enabled, the host and client functions are active simultaneously and will respond according to the user software or bus events.

When initially enabled, the module will release the SDAx and SCLx pins, putting the bus into an Idle state. The host functions will remain in an Idle state unless the user software sets the SEN control bit and the data is loaded into the I2CxTRN register. These two actions initiate a host event.

When the host logic is active, the client logic also remains active. Therefore, the client functions will begin to monitor the bus. If the client logic detects a Start event and a valid address on the bus, the client logic will begin a client transaction.

20.5.1.1 I²C I/O Pins

Two pins are used for the bus operation. These are the SCLx pin, which is the clock, and the SDAx pin, which is the data. When the module is enabled, assuming no other module with higher priority has control, the module will assume control of the SDAx and SCLx pins. The user software need not

be concerned with the state of the port I/O of the pins as the module overrides the port state and direction. At initialization, the pins are tri-stated (released).

Note: SCLx and SDAx must be configured as digital.

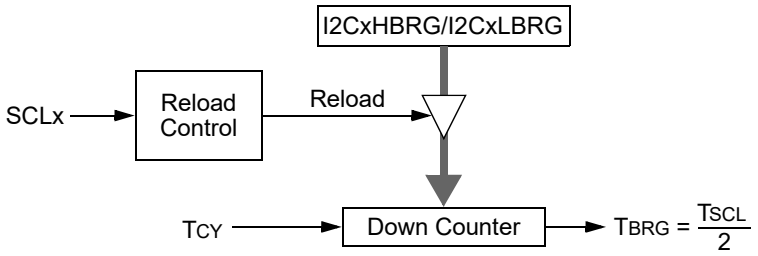
20.5.1.2 Setting Baud Rate When Operating as a Bus Host

When operating as an I^2C host, the module must generate the system SCLx clock. Generally, the I^2C system clocks are specified to be either 100 kHz, 400 kHz or 1 MHz. The system clock rate is specified as the minimum SCLx low time (I2CxLBRG), plus the minimum SCLx high time (I2CxHBRG). In most cases, that is defined by two BRG periods (T_{BRG}).

The reload value for the BRG is the I2CxHBRG/I2CxLBRG register, as illustrated in Figure 20-5. When the BRG is loaded with this value, the generator counts down to zero and stops until another reload has taken place. The BRG is reloaded automatically on baud rate Restart. For example, if clock synchronization is taking place, the BRG will be reloaded when the SCLx pin is sampled high.

Note: The I2CxHBRG/I2CxLBRG register values that are less than four are not supported.

Figure 20-5. Baud Rate Generator Block Diagram



Equation 20-1 shows the formula for computing the BRG reload value.

Equation 20-1. BRG Reload Value Calculation

$$I2CxHBRG/I2CxLBRG[23:0] = \left[\left(\left(\frac{1}{2 \bullet FSCL}\right) - Delay\right) \bullet Fi2c\right] - 3$$

Where:

Typical value of delay is 200 ns.

Note: Equation 20-1 is only for a design guideline. Due to system-dependent parameters, the actual baud rate may differ slightly. Testing is required to confirm that the actual baud rate meets the system requirements; otherwise, the value of the I2CxHBRG/I2CxLBRG register has to be adjusted.

Table 20-2. I^2C Clock Rates

Fi2c	FSCL	I2CxLBRG/I2CxHBRG (Decimal)
200 MHz	1 MHz	57
200 MHz	400 kHz	207
200 MHz	100 kHz	957
100 MHz	1 MHz	27
100 MHz	400 kHz	102
100 MHz	100 kHz	477
50 MHz	1 MHz	12
50 MHz	400 kHz	50
50 MHz	100 kHz	237

.....continued		
Fi2c	FSCL	I2CxLBRG/I2CxHBRG (Decimal)
16 MHz	1 MHz	—
16 MHz	400 kHz	13
16 MHz	100 kHz	73
8 MHz	1 MHz	—
8 MHz	400 kHz	5
8 MHz	100 kHz	35

20.5.2 Communicating As a Host In a Single Host Environment

The I²C module's typical operation in a system is using the I²C to communicate with an I²C peripheral, such as an I²C serial memory. In an I²C system, the host controls the sequence of all data communication on the bus. In this example, the dsPIC33AK128MC106 device and its I²C module have the role of the single host in the system. As the single host, it is responsible for generating the SCLx clock and controlling the message protocol.

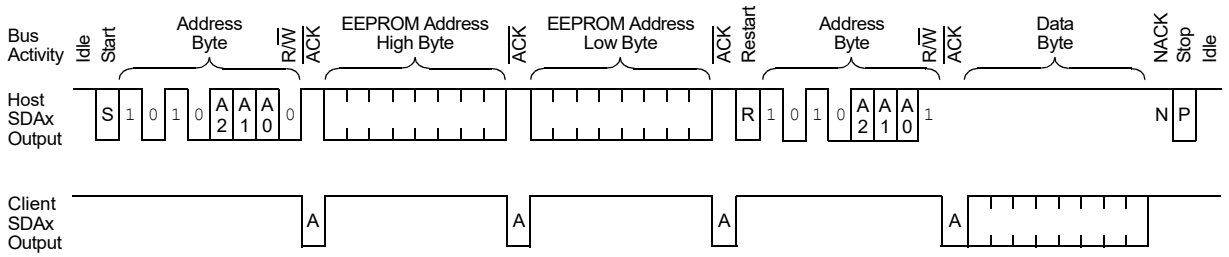
The I²C module controls individual portions of the I²C message protocol; however, sequencing of the components of the protocol to construct a complete message is performed by the user software.

For example, a typical operation in a single host environment is to read a byte from an I²C serial EEPROM. Figure 20-6 illustrates the example message.

To accomplish this message, the user software will sequence through the following steps:

1. Assert a Start condition on SDAx and SCLx.
2. Send the I²C device address byte to the client with a write indication.
3. Wait for and verify an Acknowledge from the client.
4. Send the serial memory address high byte to the client.
5. Wait for and verify an Acknowledge from the client.
6. Send the serial memory address low byte to the client.
7. Wait for and verify an Acknowledge from the client.
8. Assert a Repeated Start condition on SDAx and SCLx.
9. Send the device address byte to the client with a read indication.
10. Wait for and verify an Acknowledge from the client.
11. Enable the host reception to receive serial memory data.
12. Generate an ACK or NACK condition at the end of a received byte of data.
13. Generate a Stop condition on SDAx and SCLx.

Figure 20-6. A Typical I²C Message: Read of Serial EEPROM (Random Address Mode)



The module supports Host mode communication with the inclusion of the Start and Stop generators, data byte transmission, data byte reception, Acknowledge generator and a BRG.

Generally, the user software will write to a control register to start a particular step, then wait for an interrupt or poll status to wait for completion. These operations are discussed in the subsequent sections.

Note: The I²C module does not allow queuing of events. For example, the user software is not allowed to initiate a Start condition and immediately write the I2CxTRN register to initiate transmission before the Start condition is complete. In this case, the I2CxTRN register will not be written to and the IWCOL status bit (I2CxSTAT1[7]) will be set, indicating that this write to the I2CxTRN register did not occur.

20.5.2.1 Generating Start Bus Event

To initiate a Start event, the user software sets the SEN bit (I2CxCON1[0]. Prior to setting the Start bit, the user software should check the bus free timer flag BITO (I2CxSTAT2) to ensure that the bus is in an Idle state or wait until the bus is in an Idle state.

Figure 20-7 illustrates the timing of the Start condition.

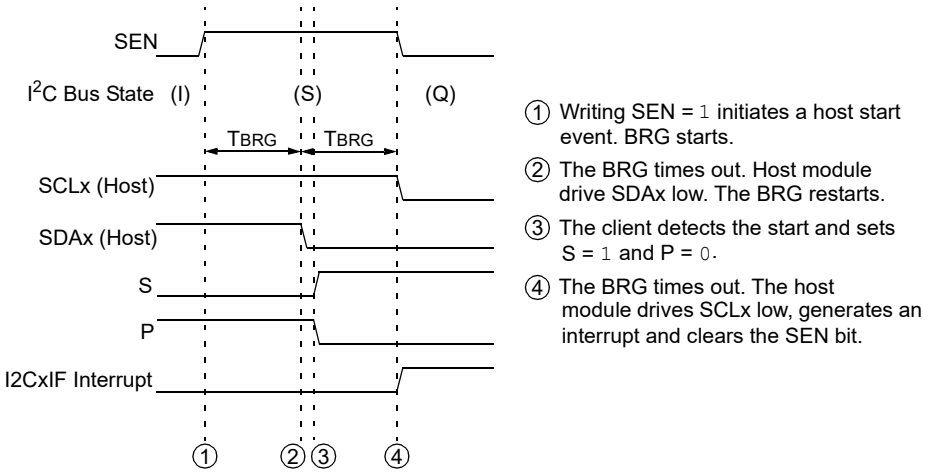
- Client logic detects the Start condition, sets the S status bit (I2CxSTAT1[3]) and clears the P status bit (I2CxSTAT1[4])
- STARTE(I2CxSTA2[14]) bit is set
- The SEN bit is automatically cleared at completion of the Start condition
- The I2CxIF interrupt is generated at completion of the Start condition if HSCIE (I2CxINTC[27]) bit and HSTIE(I2CxINTC[13]) are enabled
- HSTACT (I2CxSTAT2[29]) bit will be set on completion of Start condition
- After the Start condition, the SDAx line and SCLx lines are left low (Q state)

20.5.2.1.1 IWCOL Status Flag

If the user software writes to the I2CxTRN register when a Start sequence is in progress, the IWCOL status bit (I2CxSTAT1[7]) is set and the contents of the transmit buffer are unchanged (the write does not occur).

Note: As the queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the Start condition is complete.

Figure 20-7. Host Start Timing Diagram



20.5.2.2 Host Transmission

User software should configure data packet size in PSZ (I2CxCON2) excluding address byte and PEC, if enabled. The transmission of a data byte, a 7-bit device address byte or the second byte of a 10-bit address is accomplished by writing the appropriate value to the I2CxTRN register. Before

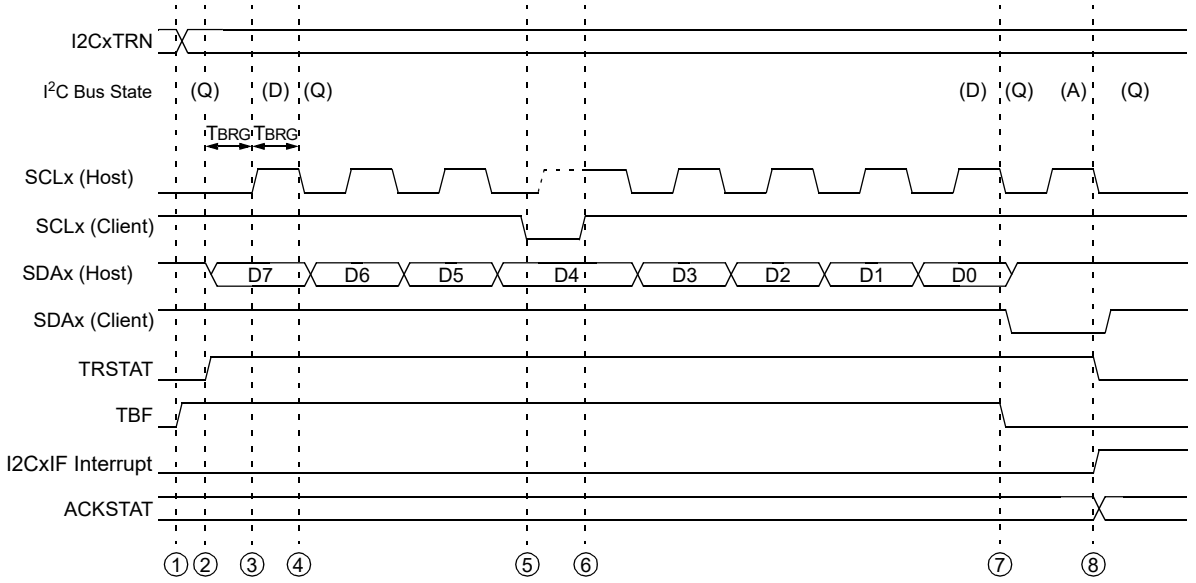
transmitting, ND/$\overline{A}$ (I2CxCON2[18]) has to be cleared/set to indicated address/data transmission. Loading the I2CxTRN register will start the following process:

1. The user software loads the I2CxTRN register with the data/address byte to transmit.
2. Writing to the I2CxTRN register sets the TBF bit (I2CxSTAT1[0]).
3. The data/address byte is shifted out through the SDAx pin until all eight bits are transmitted. Each bit of address or data will be shifted out onto the SDAx pin after the falling edge of SCLx.
4. On the ninth SCLx clock, the module shifts in the ACK bit from the client device and writes its value into the ACKSTAT status bit (I2CxSTAT1[15]).
5. The module generates the I2CxIF interrupt at the end of the ninth SCLx clock cycle if HDTXIE (I2CxINTC[1]) bit and HSTIE (I2CxINTC[13]) are enabled.

The module does not generate or validate the data bytes. The contents and usage of the bytes are dependent on the state of the message protocol maintained by the user software.

The sequence of events that occur during host transmission are provided in Figure 20-8.

Figure 20-8. Host Transmission Timing Diagram



① Writing to the I2CxTRN register will start a host transmission event. The TBF status bit is set.

② The BRG starts. The Most Significant Byte (MSB) of the I2CxTRN register drives SDAx. SCLx remains low. The TRSTAT status bit is set.

③ The BRG times out. SCLx is released and the BRG restarts.

④ The BRG times out. SCLx is driven low. After SCLx is detected low, the next bit of the I2CxTRN register drives SDAx.

⑤ While SCLx is low, the client can also pull SCLx low to initiate a Wait (clock stretch).

⑥ Host has already released SCLx and client can release to end the Wait . The BRG restarts.

⑦ At the falling edge of the eighth SCLx clock, the host releases SDAx. The TBF status bit is cleared. The client drives an $\overline{ACK}$/NACK

⑧ At the falling edge of the ninth SCLx clock, the host generates the interrupt. SCLx remains low until the next event. The client releases SDAx and the TRSTAT status bit is clear.

20.5.2.2.1 Sending a 7-Bit Address to the Client

Sending a 7-bit device address involves sending one byte to the client. A 7-bit address byte must contain the 7 bits of the I^2C device address and a R/W status bit that defines whether the message will be a write to the client (host transmission and client reception) or a read from the client (client transmission and host reception).

Note: In a 7-Bit Addressing mode, each node using the I^2C protocol should be configured with a unique address that is stored in the I2CxADD register.
While transmitting the address byte, the host must shift the address bits [7:0] left by 1 bit and configure bit 0 as the R/$\overline{W}$ bit.

20.5.2.2.2 Sending a 10-Bit Address to the Client

Sending a 10-bit device address involves sending two bytes to the client. The first byte contains five bits of the I^2C device address reserved for 10-Bit Addressing modes and two bits of the 10-bit address. As the next byte, which contains the remaining eight bits of the 10-bit address, must be received by the client, the R/W status bit in the first byte must be '0', indicating host transmission and client reception. If the message data is also directed toward the client, the host can continue sending data. However, if the host expects a reply from the client, a repeated start sequence and upper 10-bit address with the R/W status bit at '1' will change the R/W state of the message to a read of the client.

Note: In a 10-Bit Addressing mode, each node using the I^2C protocol should be configured with a unique address that is stored in the I2CxADD register.
While transmitting the first address byte, the host must shift the address bits[9:8] left by one bit and configure bit 0 as the R/W bit.

20.5.2.2.3 Receiving Acknowledge from the Client

On the falling edge of the eighth SCLx clock, the TBF status bit is cleared and the host will deassert the SDAx pin, allowing the client to respond with an Acknowledge. The host will then generate a ninth SCLx clock.

This allows the client device being addressed to respond with an $\overline{ACK}$ bit during the ninth bit time if an address match occurs or data was received properly. A client sends an Acknowledge when it has recognized its device address (including a general call) or when the client has properly received its data.

The status of $\overline{ACK}$ is written into the ACKSTAT bit (I2CxSTAT1[15]) on the falling edge of the ninth SCLx clock. After the ninth SCLx clock, the module generates the I2CxIF interrupt if HDTXIE (I2CxINTC[1]) bit and HSTIE(I2CxINTC[13]) are enabled, and enters into the Idle state until the next data byte is loaded into the I2CxTRN register.

20.5.2.2.4 ACKSTAT Status Flag

The ACKSTAT bit (I2CxSTAT1[15]) is cleared when the client has sent an Acknowledge ($\overline{ACK}$ = 0) and is set when the client does not Acknowledge ($\overline{ACK}$ = 1).

20.5.2.2.5 TBF Status Flag

When transmitting, the TBF status bit (I2CxSTAT1[0]) is set when the CPU writes to the I2CxTRN register and is cleared when all eight bits are shifted out.

20.5.2.2.6 IWCOL Status Flag

If the user software attempts to write to the I2CxTRN register when a transmit is already in progress (that is, the module is still shifting a data byte), the IWCOL status bit (I2CxSTAT1[7]) is set and the contents of the buffer are unchanged (the write does not occur). The IWCOL status bit must be cleared in the user software.

Note: Because queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the transmit condition is complete.

20.5.2.2.7 Data Transmission to Client

User software should set ND/$\overline{A}$ before data are transmitted and I2CxTRN is then loaded with the data. Packet size PSZ (I2CxSTAT2) decrements on each data transfer.

The host can be configured to ignore the NACK client response to continue transmits to the client. The HNACKIGN(I2CxCON2[21]) has to be set and the transmit can then continue.

Once packet size becomes zero, end of packet (EOP) will be set (I2CxSTAT2[24]) if enabled (EOPSC, I2CxCON2[20:19]).

If the transmit buffer (I2CxTRN) is empty and packet size (I2CxCON2[15:0]) has not reached zero, the I2CxSTAT.SSPND bit is set and the host drives SCL low. Once the transmit buffer (I2CxTRN) is loaded, I2CxSTAT.SSPND will be cleared and transmission will be resumed (see 20.6.1. Host Transmission (7-bit Address) and 20.6.2. Host Transmission (10-bit Address)).

20.5.2.3 Receiving Data from a Client Device

The host can receive data from the client device after the host has transmitted the client address with a R/W status bit value of '1'. After receiving ACK from the client for the address, configure packet size (PSZ,(I2CxCON2[15:0]) and then enable RCEN bit (I2CxCON1[3]) for the first data byte to be received. If SMART mode (SMEN, I2CxCON2 [17]) is enabled, then the hardware will automatically set RCEN(I2CxCON1[3]) based on the status of RBF(I2CxSTAT1[1]) and PSZ (I2CxCON2[15:0]) for remaining bytes; otherwise, the user software should set RCEN(I2CxCON1[3]) for every byte to be received. The host logic begins to generate clocks and before each falling edge of the SCLx, the SDAx line is sampled and data is shifted into the I2CxRSR register.

Note: The lower five bits of the I2CxCON1 register must be '0' before attempting to set the RCEN bit. This ensures that the host logic is inactive.

After the falling edge of the eighth SCLx clock, the following events occur:

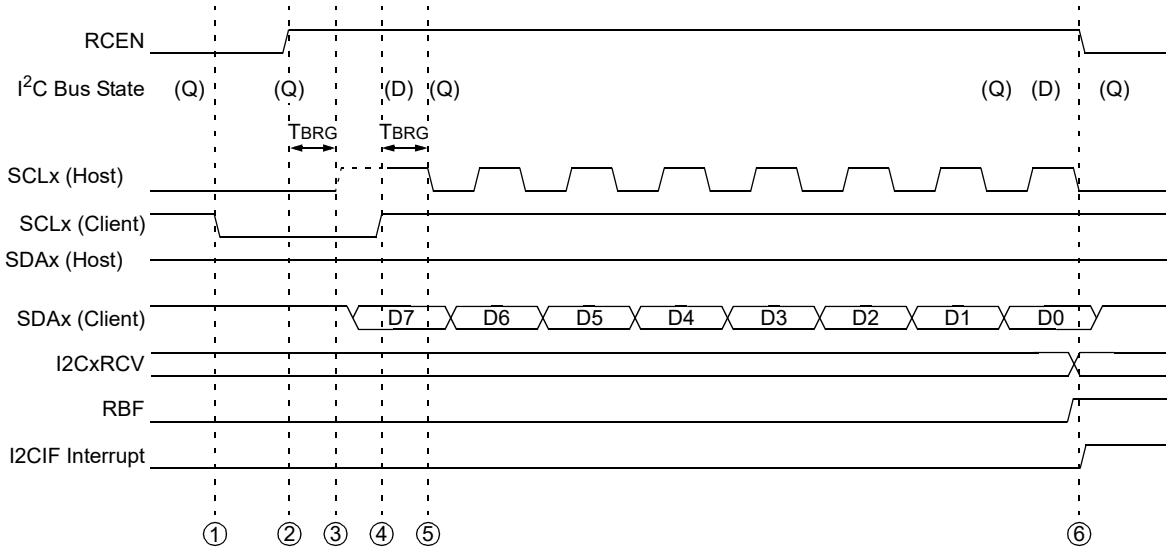
- The RCEN bit is automatically cleared
- The contents of the I2CxRSR register transfer into the I2CxRCV register
- The RBF status bit (I2CxSTAT1[1]) is set
- The I^2C module generates the I2CxIF interrupt if HDRXIE (I2CxINTC[0]) bit and HSTIE(I2CxINTC[13]) are enabled

When the CPU reads the receive buffer (I2CxRCV), the RBF status bit is automatically cleared. The user software can process the data and then execute an Acknowledge sequence.

If SMART mode is enabled, the hardware will automatically set RCEN(I2CxCON1[3]) based on the status of RBF(I2CxSTAT1[1]) and PSZ (I2CxCON2[15:0]) for remaining bytes; otherwise the user software should set RCEN(I2CxCON1[3]) for every byte to be received.

If the receive buffer (I2CxRCV) is full and packet size (I2CxCON2[15:0]) has not reached zero, the I2CxSTAT.SSPND bit is set and the host drives SCL low. The user should remove the Receiver Full condition by reading the receive buffer (I2CxRCV). Once the full condition is removed, I2CxSTAT.SSPND will be cleared by hardware. Once packet size becomes zero, end of packet (EOP) will be set (I2CxSTAT2[24]), if enabled (EOPSC, I2CxCON2[20:19]). The sequence of events that occurs during host reception is illustrated in Figure 20-9.

Figure 20-9. Host Reception Timing Diagram



1. Typically, the client can pull SCLx low (clock stretch) to request a Wait to prepare the data response. The client will drive the MSB of the data response on SDAx when ready.
2. Writing the RCEN bit will start a host reception event. The BRG starts. SCLx remains low.
3. The BRG times out. The host attempts to release SCLx.
4. When the client releases SCLx, the BRG restarts.
5. The BRG times out. The MSB of the response is shifted to the I2CxRSR register. SCLx is driven low for the next baud interval.
6. At the falling edge of the eighth SCLx clock, the I2CxRSR register is transferred to the I2CxRCV register. The module clears the RCEN bit. The RBF status bit is set. The host generates the interrupt.

20.5.2.3.1 RBF Status Flag

When receiving data, the RBF status bit (I2CxSTAT1[1]) is set when a device address or data byte is loaded into the I2CxRCV register from the I2CxRSR register. It is cleared when the user software reads the I2CxRCV register.

20.5.2.3.2 I2COV Status Flag

If another byte is received in the I2CxRSR register while the RBF status bit remains set and the previous byte remains in the I2CxRCV register, the I2COV status bit (I2CxSTAT1[6]) is set and the data in the I2CxRSR register is lost.

Leaving the I2COV status bit set does not inhibit further reception. If the RBF status bit is cleared by reading the I2CxRCV register, and the I2CxRSR register receives another byte, that byte will be transferred to the I2CxRCV register.

20.5.2.3.3 IWCOL Status Flag

If the user software writes to the I2CxTRN register when a receive is already in progress (that is, the I2CxRSR register is still shifting in a data byte), the IWCOL status bit (I2CxSTAT1[7]) is set and the contents of the buffer are unchanged (the write does not occur) (see 20.6.3. Host Reception (7-bit Address) and 20.6.4. Host Reception (10-bit Address)).

Note: Because queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the data reception condition is complete.

20.5.2.4 Acknowledge Generation

User software should configure an Acknowledge sequence using ACKC bits(I2CxCON2[23:22]). When ACKC is configured with "00", setting the ACKEN bit (I2CxCON1[4]) enables the generation of a host Acknowledge sequence.

Note: The lower five bits of the I2CxCON or I2CxCONL register must be '0' (host logic inactive) before attempting to set the ACKEN bit.

Figure 20-10 illustrates an $\overline{ACK}$ sequence and Figure 20-11 illustrates a NACK sequence. The ACKDT bit (I2CxCON1[5] specifies an $\overline{ACK}$ or NACK sequence.

After two baud periods, the ACKEN bit is automatically cleared and the module generates the I2CxIF interrupt, if HACKSIE(I2CxINTC[11]) bit and HSTIE(I2CxINTC[13]) are enabled.

Figure 20-10. Host Acknowledge ($\overline{ACK}$) Timing Diagram

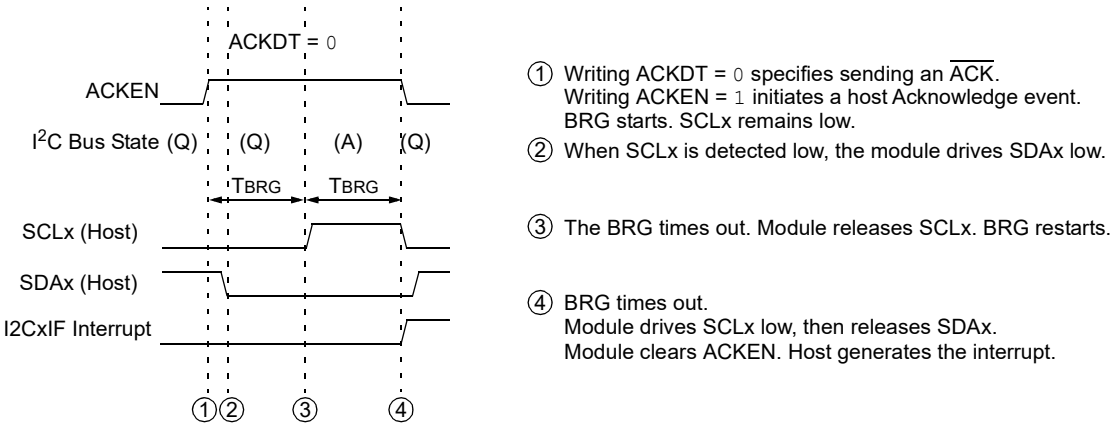
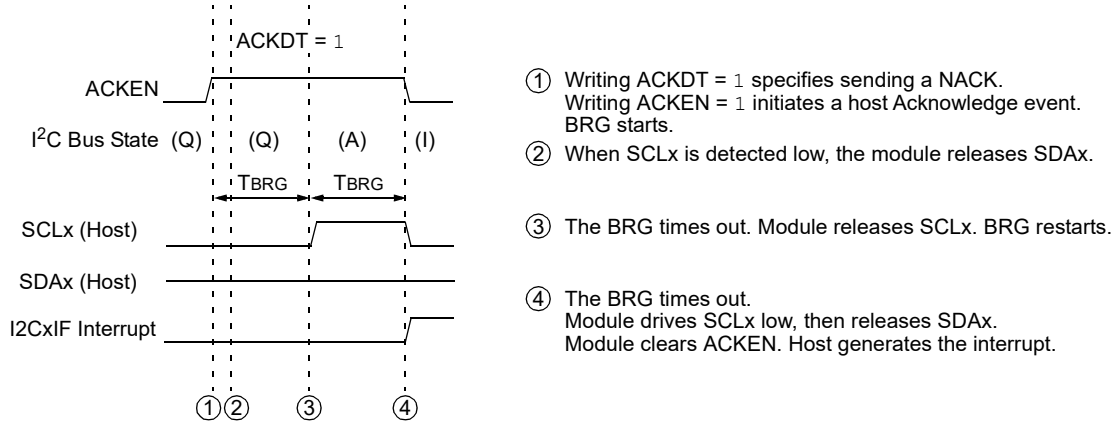


Figure 20-11. Host Not Acknowledge (NACK) Timing Diagram



20.5.2.4.1 IWCOL Status Flag

If the user software writes to the I2CxTRN register when a receive is already in progress (that is, the I2CxRSR register is still shifting in a data byte), the IWCOL status bit (I2CxSTAT1[7]) is set and the contents of the buffer are unchanged (the write does not occur) (see 20.6.3. Host Reception (7-bit Address) and 20.6.4. Host Reception (10-bit Address)).

Note: Because queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the data reception condition is complete.

20.5.2.5 Generating a Stop Bus Event

Setting the PEN bit (I2CxCON1[2]) enables the generation of a host Stop sequence.

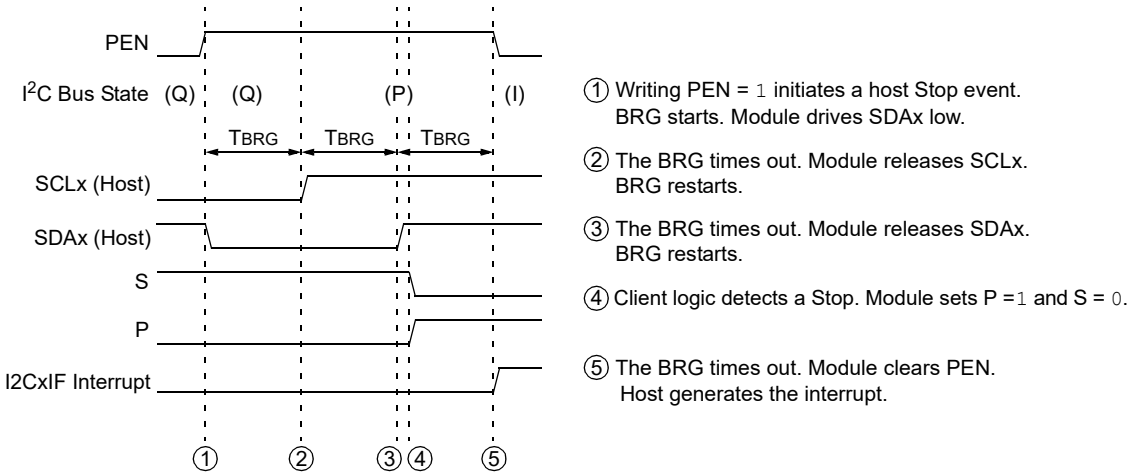
Note: The lower five bits of the I2CxCON1 register must be '0' (host logic inactive) before attempting to set the PEN bit.

When the PEN bit is set, the host generates the Stop sequence, as illustrated in Figure 20-12.

- The client detects the Stop condition, sets the P status bit (I2CxSTAT1[4]) and clears the S status bit (I2CxSTAT1[3])
- STOPE(I2CxSTAT2[15]) bit is set
- The PEN bit is automatically cleared
- The HSTACT(I2CxSTAT2[29]) bit is cleared when STOP is sent.
- The module generates the I2CxIF interrupt if HPCIE(I2CxINTC[28]) bit and HSTIE(I2CxINTC[13]) are enabled

Note: Because queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the Stop condition is complete.

Figure 20-12. Host Stop Timing Diagram



20.5.2.5.1 IWCOL Status Flag

If the user software writes to the I2CxTRN register when a receive is already in progress (that is, the I2CxRSR register is still shifting in a data byte), the IWCOL status bit (I2CxSTAT1[7]) is set and the contents of the buffer are unchanged (the write does not occur) (see 20.6.3. Host Reception (7-bit Address) and 20.6.4. Host Reception (10-bit Address)).

Note: Because queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the data reception condition is complete.

20.5.2.6 Generating a Repeated Start Bus Event

Setting the RSEN bit (I2CxCON1[1]) enables the generation of a host repeated start sequence, as illustrated in Figure 20-13.

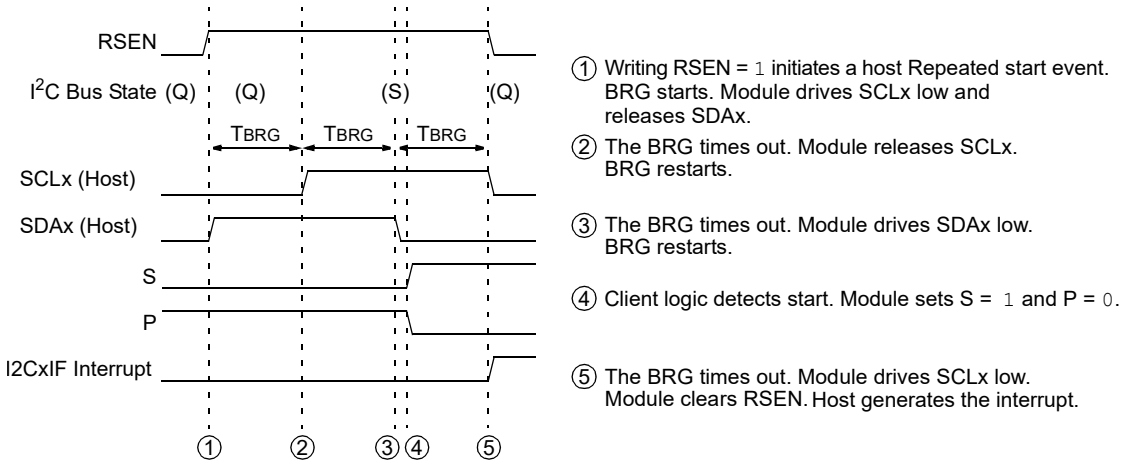
Note: The lower five bits of the I2CxCON1 register must be '0' (host logic inactive) before attempting to set the RSEN bit.

To generate a Repeated Start condition, the user software sets the RSEN bit. The host module asserts the SCLx pin low. When the module samples the SCLx pin low, the module releases the SDAx pin for 1 T_{BRG}. When the BRG times out and the module samples SDAx high, the module deasserts the SCLx pin. When the module samples the SCLx pin high, the BRG reloads and begins counting. SDAx and SCLx must be sampled high for 1 T_{BRG}. This action is then followed by assertion of the SDAx pin low for 1 T_{BRG} while SCLx is high.

The following is the repeated start sequence:

1. The client detects the Start condition, sets the S status bit (I2CxSTAT1[3]) and clears the P status bit (I2CxSTAT1[4]).
2. The RSEN bit is automatically cleared.
3. The I²C module generates the I2CxIF interrupt if HSCIE(I2CxINTC[27]) bit and HSTIE(I2CxINTC[13]) are enabled.

Figure 20-13. Host Repeated Start Timing Diagram



20.5.2.6.1 IWCOL Status Flag

If the user software writes to the I2CxTRN register when a receive is already in progress (that is, the I2CxRSR register is still shifting in a data byte), the IWCOL status bit (I2CxSTAT1[7]) is set and the contents of the buffer are unchanged (the write does not occur) (see 20.6.3. Host Reception (7-bit Address) and 20.6.4. Host Reception (10-bit Address)).

Note: Because queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the data reception condition is complete.

20.5.2.7 Building Complete Host Messages

As described in 20.5.2. Communicating As a Host In a Single Host Environment, the user software is responsible for constructing messages with the correct message protocol. The module controls individual portions of the I²C message protocol; however, sequencing of the components of the protocol to construct a complete message is performed by the user software.

The user software can use polling or interrupt methods while using the module. The timing diagrams shown in this document use interrupts for detecting various events.

The user software can use the SEN, RSEN, PEN, RCEN and ACKEN bits (Least Significant five bits of the I2CxCON1 register), and the TRSTAT status bit as a 'state' flag when progressing through a message. For example, Table 20-3 shows some example state numbers associated with bus states.

Table 20-3. Host Message Protocol States

Example State Number(1)	I2CxCON[4:0] or I2CxCONL[4:0]	TRSTAT (I2CxSTAT[14])	State
0	00000	0	Bus Idle or Wait
1	00001	N/A	Sending Start Event
2	00000	1	Host Transmitting

Note: The example state numbers are for reference only. The user software can assign the state numbers as desired.

.....continued

Example State Number(1)	I2CxCON[4:0] or I2CxCONL[4:0]	TRSTAT (I2CxSTAT[14])	State
3	00010	N/A	Sending Repeated Start Event
4	00100	N/A	Sending Stop Event
5	01000	N/A	Host Reception
6	10000	N/A	Host Acknowledgment

Note: The example state numbers are for reference only. The user software can assign the state numbers as desired.

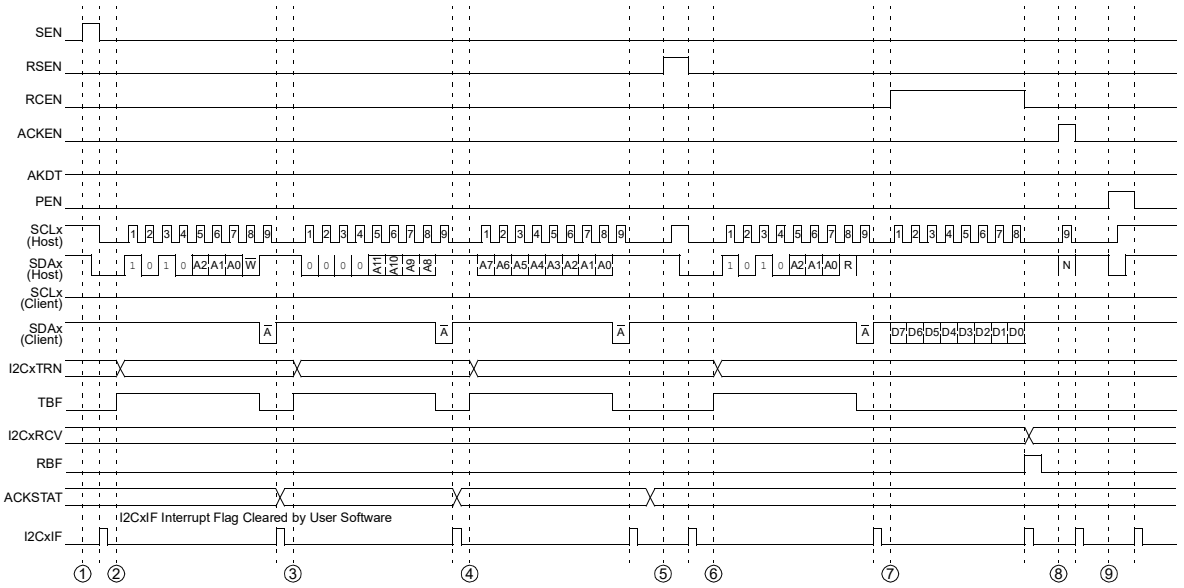
The user software will begin a message by issuing a Start condition. The user software will record the state number corresponding to the Start.

As each event completes and generates an interrupt, the interrupt handler may check the state number. Therefore, for a Start state, the interrupt handler will confirm execution of the start sequence and then start a host transmission event to send the I^2C device address, changing the state number to correspond to the host transmission.

On the next interrupt, the interrupt handler will again check the state, determining that a host transmission just completed. The interrupt handler will confirm successful transmission of the data and then move on to the next event, depending on the contents of the message. In this manner, on each interrupt, the interrupt handler will progress through the message protocol until the complete message is sent.

Figure 20-14 provides a detailed examination of the same message sequence as shown in Figure 20-6. Figure 20-15 provides a few simple examples of the messages using a 7-bit addressing format. Figure 20-18 provides an example of a 10-bit addressing format message sending data to a client. Figure 20-19 provides an example of a 10-bit addressing format message receiving data from a client.

Figure 20-14. Host Message (Typical I²C Message: Read of Serial EEPROM)



① Setting the SEN bit begins a Start event.

② Writing the I2CxTRN register starts a host transmission. The data is the serial EEPROM device address byte, with the R/W status bit clear, indicating a write.

③ Writing the I2CxTRN register starts a host transmission. The data is the first byte of the EEPROM data address.

④ Writing the I2CxTRN register starts a host transmission. The data is the second byte of the EEPROM data address.

⑤ Setting the RSEN bit starts a Repeated Start event.

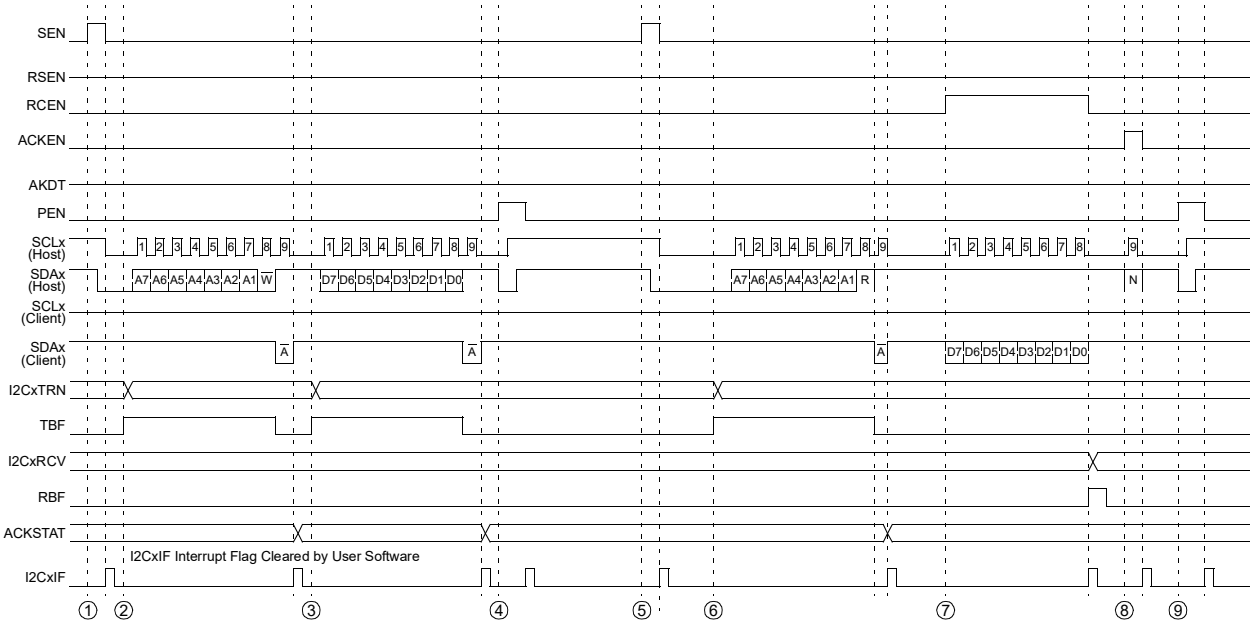
⑥ Writing the I2CxTRN register starts a host transmission. The data is a resend of the serial EEPROM device address byte, but with R/W status bit set, indicating a read.

⑦ Setting the RCEN bit starts a host reception. On interrupt, the user software reads the I2CxRCV register, which clears the RBF status bit.

⑧ Setting the ACKEN bit starts an Acknowledge event. ACKDT = 1 to send a NACK.

⑨ Setting the PEN bit starts a host Stop event.

Figure 20-15. Host Message (7-Bit Address: Transmission and Reception)



① Setting the SEN bit begins a Start event.

② Writing the I2CxTRN register starts a host transmission. The data is the address byte with the R/$\overline{W}$ status bit clear.

③ Writing the I2CxTRN register starts a host transmission. The data is the message byte.

④ Setting the PEN bit starts a host Stop event.

⑤ Setting the SEN bit begins a Start event. An interrupt is generated on completion of the Start event.

⑥ Writing the I2CxTRN register starts a host transmission. The data is the address byte with the R/$\overline{W}$ status bit set.

⑦ Setting the RCEN bit starts a host reception.

⑧ Setting the ACKEN bit starts an Acknowledge event. ACKDT = 1 to send a NACK.

⑨ Setting the PEN bit starts a host Stop event.

Figure 20-16. I²C Host, 7-Bit Address, Read with RXIF without CRC

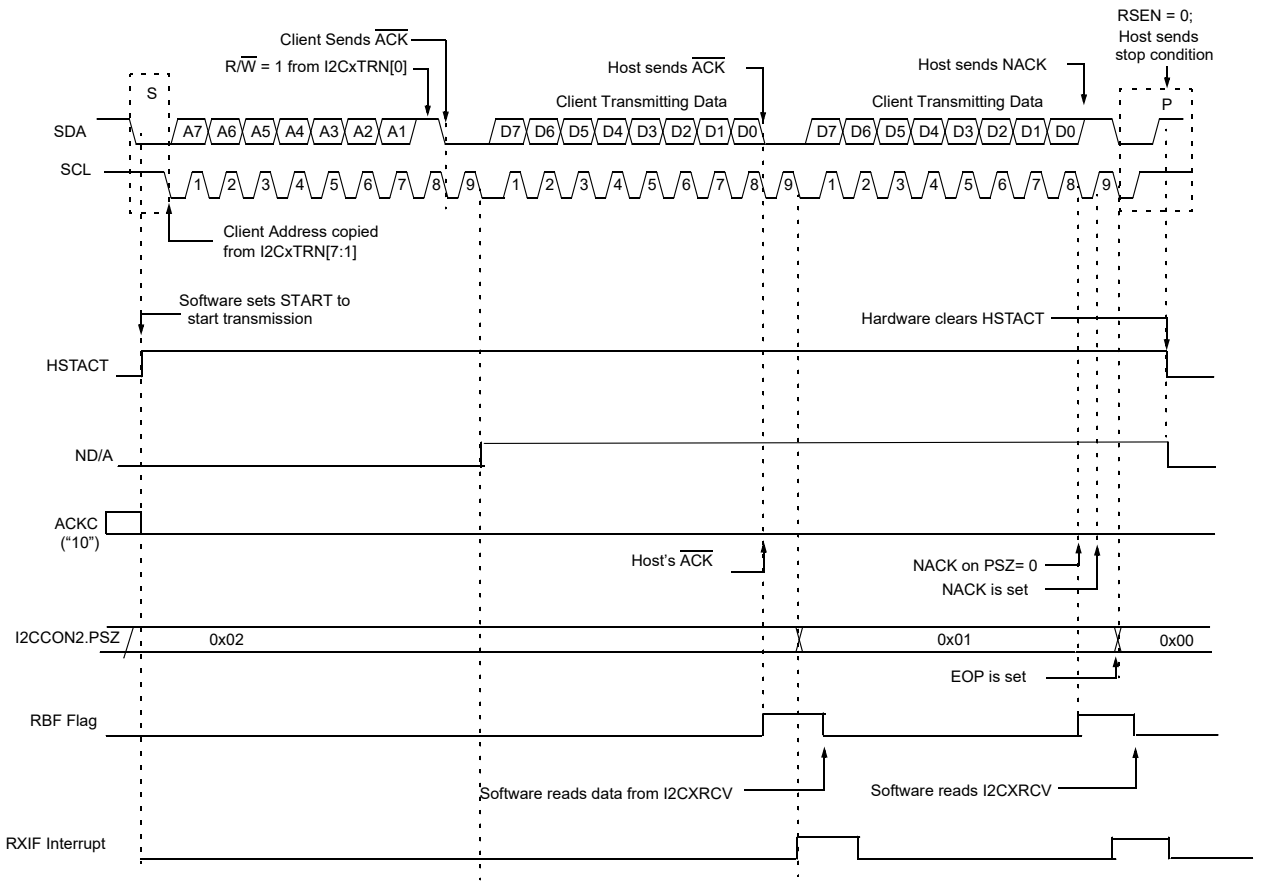


Figure 20-17. I2C Host, 7-Bit Address, Write with TXIF without CRC

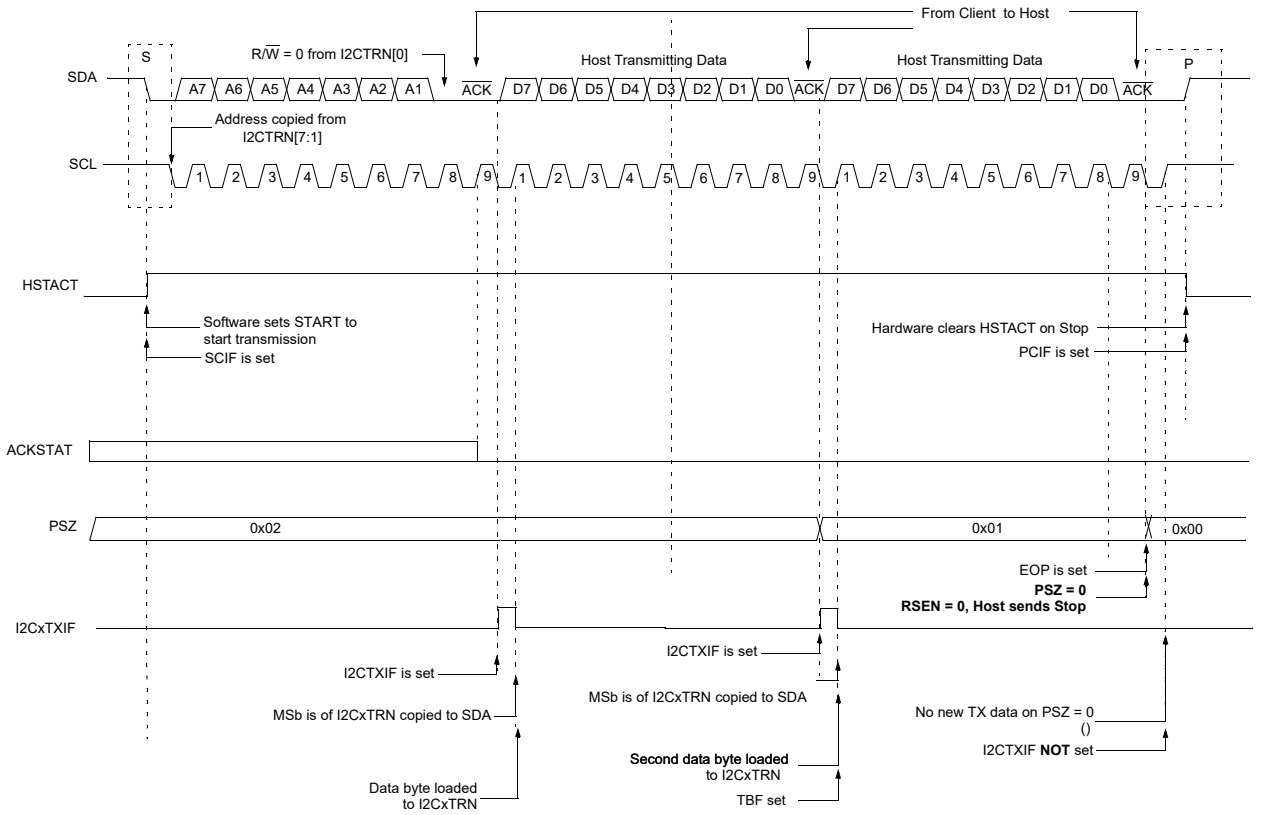
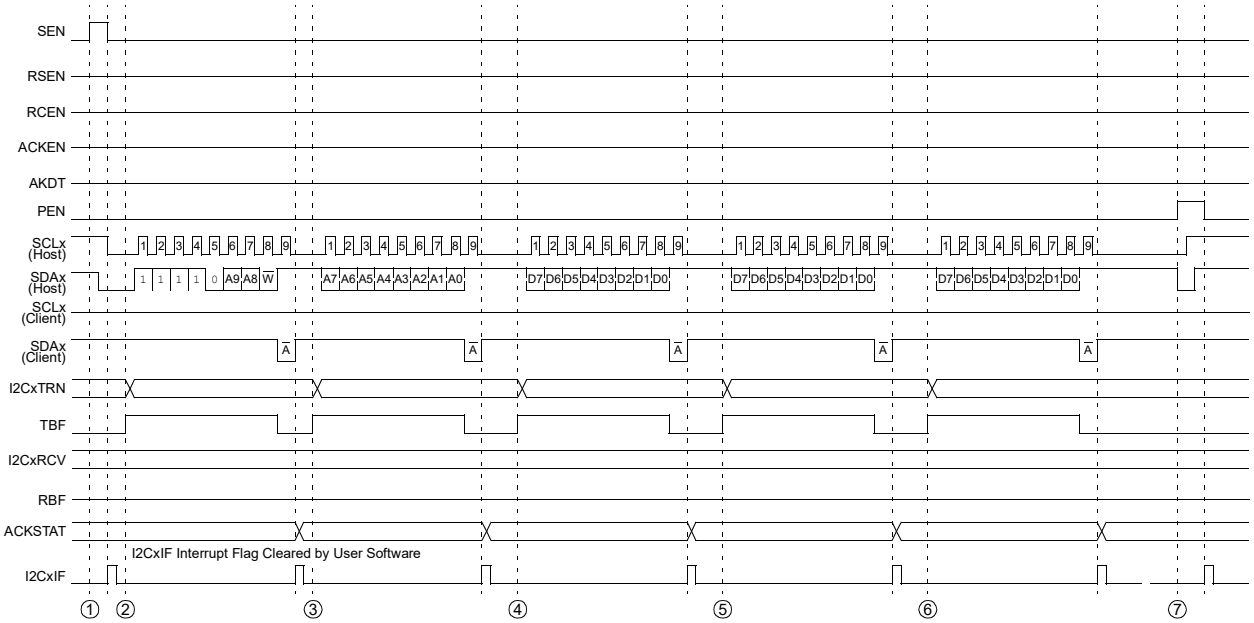


Figure 20-18. Host Message (10-Bit Transmission)



① Setting the SEN bit begins a Start event.

② Writing the I2CxTRN register starts a host transmission. The data is the first byte of the address.

③ Writing the I2CxTRN register starts a host transmission. The data is the second byte of the address.

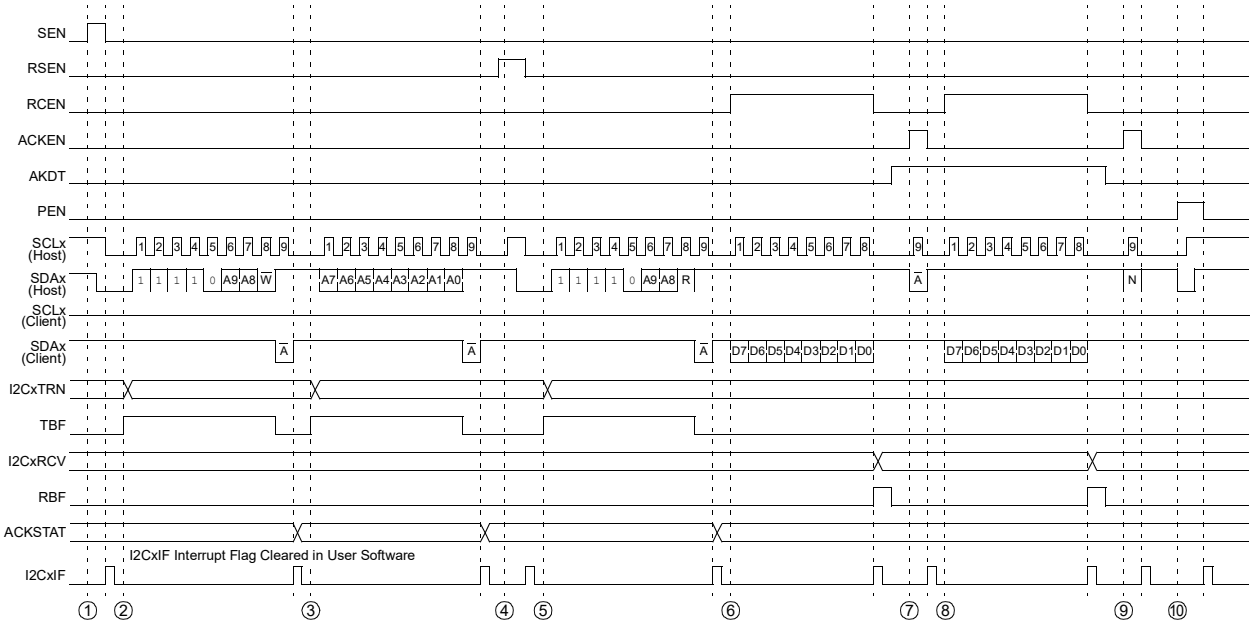
④ Writing the I2CxTRN register starts a host transmission. The data is the first byte of the message data.

⑤ Writing the I2CxTRN register starts a host transmission. The data is the second byte of the message data.

⑥ Writing the I2CxTRN register starts a host transmission. The data is the third byte of the message data.

⑦ Setting the PEN bit starts a host Stop event.

Figure 20-19. Host Message (10-Bit Reception)



① Setting the SEN bit begins a Start event.

② Writing the I2CxTRN register starts a host transmission. The data is the first byte of the address with the R/$\overline{W}$ status bit cleared.

③ Writing the I2CxTRN register starts a host transmission. The data is the second byte of the address.

④ Setting the RSEN bit starts a host Restart event.

⑤ Writing the I2CxTRN register starts a host transmission. The data is a resend of the first byte with the R/$\overline{W}$ status bit set.

⑥ Setting the RCEN bit starts a host reception. On interrupt, the user software reads the I2CxRCV register, which clears the RBF status bit.

⑦ Setting the ACKEN bit starts an Acknowledge event. ACKDT = 0 to send $\overline{ACK}$.

⑧ Setting the RCEN bit starts a host reception.

⑨ Setting the ACKEN bit starts an Acknowledge event. ACKDT = 1 to send NACK.

⑩ Setting the PEN bit starts a host Stop event.

20.5.3 Communicating As a Host In A Multi-Host Environment

The I^2C protocol allows more than one host to be attached to a system bus. Taking into account that a host can initiate message transactions and generate clocks for the bus, the protocol has methods to account for situations where more than one host is attempting to control the bus. The clock synchronization ensures that multiple nodes can synchronize their SCLx clocks to result in one common clock on the SCLx line. The bus arbitration ensures that if more than one node attempts a message transaction, only one node will be successful in completing the message. The other nodes lose bus arbitration and are left with a bus collision.

20.5.3.1 Multi-Host Operation

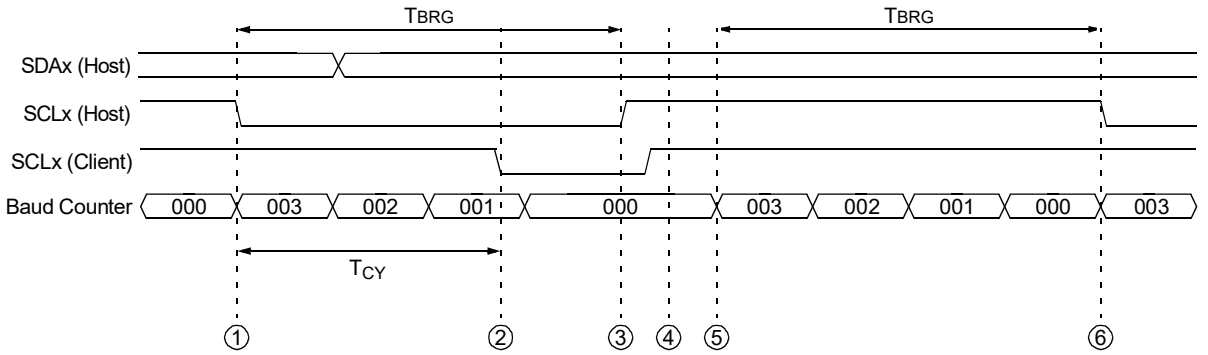
The host module has no special settings to enable the multi-host operation. The module performs the clock synchronization and bus arbitration at all times. If the module is used in a single host environment, clock synchronization only occurs between the host and clients and bus arbitration does not occur.

20.5.3.2 Master Clock Synchronization

In a multi-host system, different hosts can have different baud rates. The clock synchronization ensures that when these hosts are attempting to arbitrate the bus, their clocks will be coordinated.

The clock synchronization occurs when the host deasserts the SCLx pin (SCLx intended to float high). When the SCLx pin is released, the BRG is suspended from counting until the SCLx pin is actually sampled high. When the SCLx pin is sampled high, the BRG is reloaded with the contents of I2CxHBRG[23:0] and I2CxLBRG[23:0] begins counting. This ensures that the SCLx high time will always be at least one BRG rollover count in the event that the clock is held low by an external device, as illustrated in Figure 20-20.

Figure 20-20. Baud Rate Generator Timing with Clock Synchronization



① The baud counter decrements twice per T_{CY}. On rollover, the host SCLx will transition.

② The client has pulled SCLx low to initiate a Wait.

③ At what would be the host baud counter rollover, detecting SCLx low holds the counter.

④ Logic samples SCLx once per T_{CY}. Logic detects SCLx high.

⑤ The baud counter rollover occurs on the next cycle.

⑥ On the next rollover, the host SCLx will transition.

20.5.3.3 Bus Arbitration and Bus Collision

The bus arbitration supports the multi-host system operation. The wired-AND nature of the SDAx line permits arbitration. Arbitration takes place when the first host outputs '1' on SDAx by letting the SDAx float high, and simultaneously, the second host outputs '0' on SDAx by pulling SDAx low. The SDAx signal will go low. In this case, the second host has won bus arbitration. The first host has lost bus arbitration, and thus, has a bus collision.

For the first host, the expected data on SDAx is '1', while the data sampled on SDAx is '0'. This is the definition of a bus collision.

The first host will set the BCL bit (I2CxSTAT1[10]) and generate an error (I2CxEIF) interrupt if the BCLDIE (I2CxINTC[15]) bit is enabled. The Host module will reset the I^2C port to its Idle state.

In multi-host operation, the SDAx line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed by the host logic, with the result placed in the BCL status bit.

The states where arbitration can be lost are:

- Start condition
- Repeated Start condition
- Address, Data or Acknowledge bit
- Stop condition

20.5.3.4 Detecting Bus Collisions and Resending Messages

When a bus collision occurs, the host module sets the BCL status bit and generates an error (I2CxEIF) interrupt if the BCLDIE (I2CxINTC[15]) bit is enabled. If a bus collision occurs during a byte transmission, the transmission is stopped, the TBF status bit is cleared, and the SDAx and SCLx pins are deasserted. If a bus collision occurs during a Start, Repeated Start, Stop or Acknowledge condition, the condition is aborted, the respective control bits in the I2CxCON register are cleared, and the SDAx and SCLx lines are deasserted.

If the user software is expecting an interrupt at the completion of the host event, the user software can check the BCL status bit to determine if the host event completed successfully or a bus collision

occurred (or it may branch to a bus collision interrupt on bus collision), or a host interrupt occurred in case of a successful host event. If a bus collision occurs, the user software must abort sending the rest of the pending message and prepare to resend the entire message sequence, beginning with the Start condition, after the bus returns to the Idle state. The user software can monitor the S and P status bits to wait for an Idle bus. When the user software executes the host Interrupt Service Routine (ISR) and the I^2C bus is free, the user software can resume communication by asserting a Start condition.

20.5.3.5 Bus Collision During a Start Condition

Before issuing a Start condition, the user software should verify an Idle state of the bus using the S and P status bits. Two hosts may attempt to initiate a message at a similar point in time. Typically, the hosts will synchronize clocks and continue arbitration into the message until one loses arbitration. Any of the following conditions can cause a bus collision to occur during a Start:

- If the SDAx and SCLx pins are at a low logic state at the beginning of the Start condition
- If the SCLx line is at a low logic state before the SDAx line is driven low

In either case, the host that loses arbitration during the Start condition generates a bus collision interrupt.

20.5.3.6 Bus Collision During a Repeated Start Condition

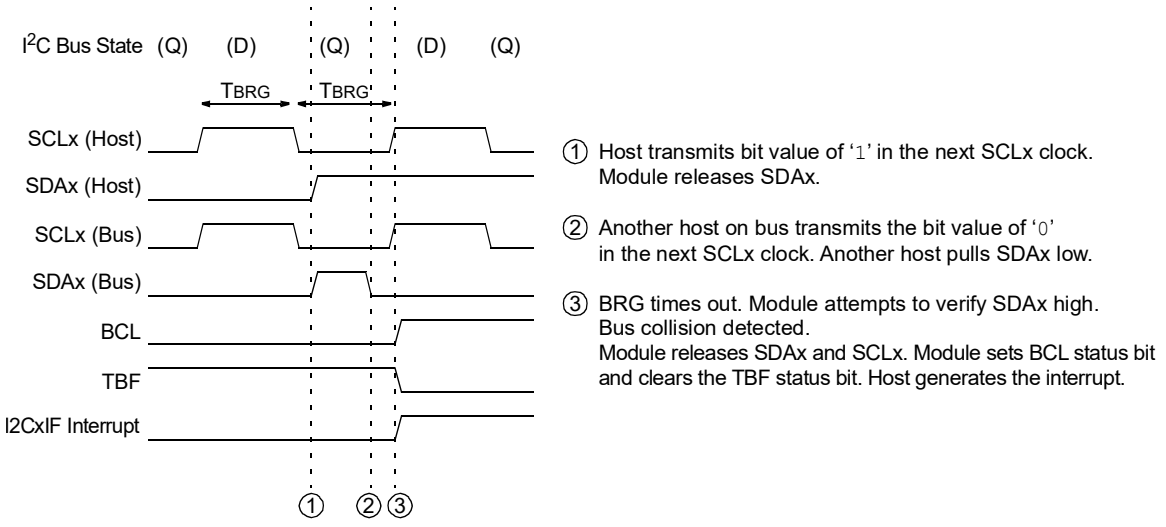
When the two hosts do not collide throughout an address byte, a bus collision can occur when one host attempts to assert a Repeated Start while another transmits data. In this case, the host generating the Repeated Start loses arbitration and generates a bus collision interrupt.

20.5.3.7 Bus Collision During Message Bit Transmission

The most typical case of data collision occurs while the host is attempting to transmit the device address byte, a data byte or an Acknowledge bit.

If the user software is properly checking the bus state, it is unlikely that a bus collision will occur on a Start condition. However, because another host can, at the same time, check the bus and initiate its own Start condition, it is likely that SDAx arbitration will occur and synchronize the Start of two hosts. In this condition, both hosts begin and continue to transmit their messages until one host loses arbitration on a message bit. The SCLx clock synchronization keeps the two hosts synchronized until one loses arbitration. Figure 20-21 illustrates an example of the message bit arbitration.

Figure 20-21. Bus Collision During Message Bit Transmission



20.5.3.8 Bus Collision During a Stop Condition

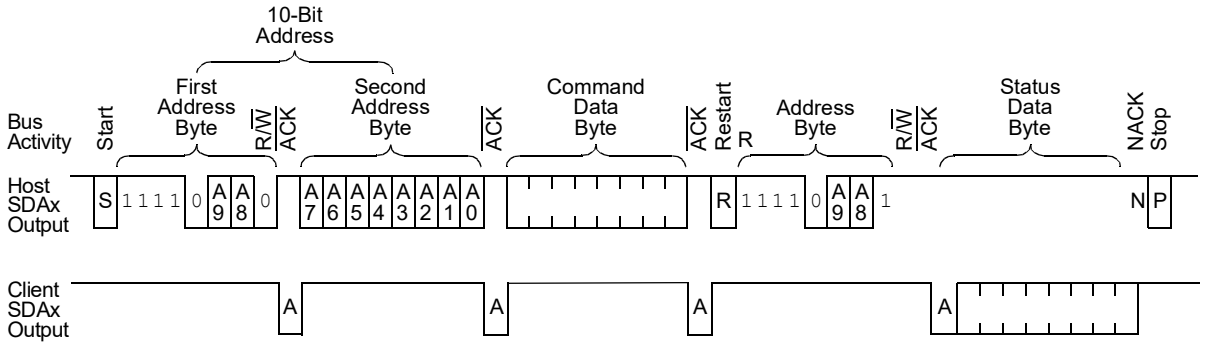
If the host software loses track of the state of the I2C bus, many existing conditions can cause a bus collision during a Stop condition. In this case, the host generating the Stop condition will lose arbitration and generate a bus collision interrupt. The HSBCL (I2CxSTAT2[25]) bit will be set to collision during stop condition.

20.5.4 Communicating As a Client

In some systems, particularly where multiple processors communicate with each other, the dsPIC33AK128MC106 device can communicate as a client, as illustrated in Figure 20-22. When the I^2C module is enabled, the client is active. The client cannot initiate a message; it can only respond to a message sequence initiated by a host. The host requests a response from a particular client as defined by the device address byte in the I^2C protocol. The client replies to the host at the appropriate times as defined by the protocol.

As with the host module, sequencing the components of the protocol for the reply is a user software task. However, the client detects when the device address matches the address specified by the user software for that client.

Figure 20-22. A Typical Client I^2C Message: Multiprocessor Command/Status



After a Start condition, the client receives and checks the device address. The client can specify either a 7-bit address or a 10-bit address. When a device address is matched, the module will generate an interrupt to notify the user software that its device is selected. Based on the R/W status bit sent by the host, the client either receives or transmits data. If the client is to receive data, the client automatically generates the Acknowledge (ACK), loads the I2CxRCV register with the received value currently in the I2CxRSR register and notifies the user software through an interrupt. If the client is to transmit data, the user software must load the I2CxTRN register.

20.5.4.1 Sampling Receive Data

All the incoming bits are sampled with the rising edge of the clock (SCLx) line.

20.5.4.2 Detecting Start and Stop Conditions

The client detects the Start and Stop conditions on the bus and indicates that status on the S status bit (I2CxSTAT1[3]), STARTE (I2CxSTAT2[14]) and P status bit (I2CxSTAT1[4]), STOPE (I2CxSTAT2[15]). The Start (S) and Stop (P) status bits are cleared when a Reset occurs or when the module is disabled. After detection of a start or repeated start event, the S status bit is set and the P status bit is cleared.

After detection of a Stop event:

- The P status bit is set and the S status bit is cleared
- The CLTACT(I2CxSTAT2[30]) bit is cleared

20.5.4.2.1 Interrupt On Start/Repeated Start and Stop Conditions (Client Mode)

The user software is notified through a client interrupt if the SCIE bit (I2CxCON1[21]) is set for a Start/Repeated Start condition or if the PCIE bit (I2CxCON1[22]) is set for a Stop condition.

20.5.4.3 Detecting the Address

Once the module has been enabled, the client waits for a Start condition to occur. After a Start, depending on the A10M bit (I2CxCON1[10]), the client attempts to detect a 7-bit or 10-bit address. The client compares one received byte for a 7-bit address or two received bytes for a 10-bit address. A 7-bit address also contains a R/W status bit that specifies the direction of the data transfer after the address. If R/W = `0`, a write is specified and the client receives data from the host. If R/W = `1`, a read is specified and the client sends data to the host. The 10-bit address contains a R/W status bit; however, by definition, it is always R/W = `0` because the client must receive the second byte of the 10-bit address.

20.5.4.3.1 Addressing Modes

Client can be configured into different addressing modes using AMODE (I2CxCON2[31:30]). Received address will be compared to address in:

- I2CxADDR when AMODE is `00`
- I2CxADDR or I2CxAMSK when AMODE is `01`
- Range of address specified I2CxADDR and I2CxAMSK when AMODE is `10`

20.5.4.3.2 Client Address Masking

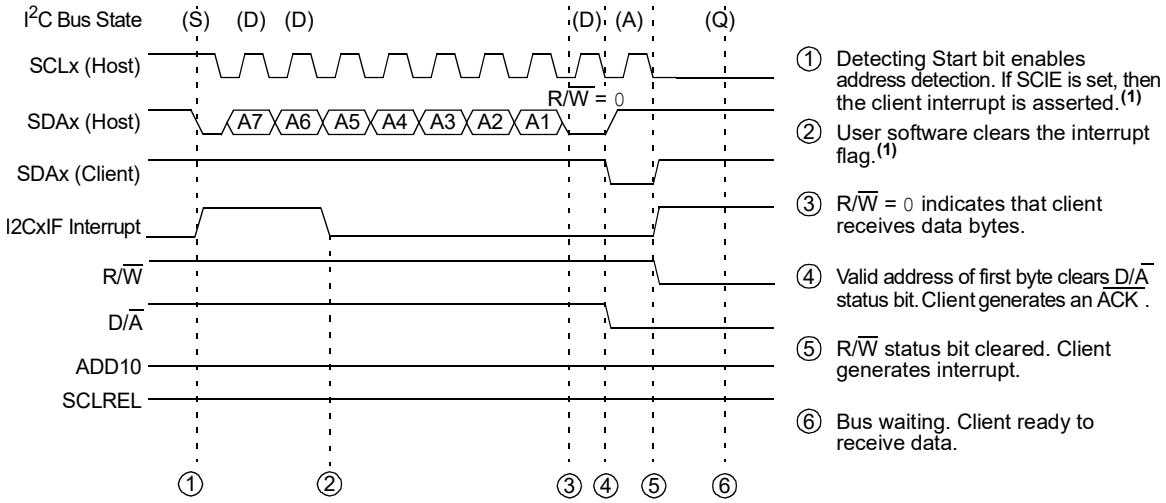
The I2CxMSK register masks the address bit positions, designating them as “don’t care” bits for both 10-bit and 7-bit Addressing modes. When a bit in the I2CxMSK register is set (= `1`), the client responds when the bit in the corresponding location of the address is a ‘`0`’ or ‘`1`’. For example, in 7-bit Client mode with I2CxMSK = `0100000`, the client module Acknowledges addresses, ‘`0000000`’ and ’`0100000`’, as valid.

20.5.4.3.3 7-Bit Address and Client Write

After the Start condition, the module shifts 8 bits into the I2CxRSR register, as illustrated in Figure 20-23. The value of the I2CxRSR register is evaluated against that of the I2CxADD and I2CxMSK registers on the falling edge of the eighth clock (SCLx). If the address is valid (that is, an exact match between unmasked bit positions), the following events occur:

- An ACK is generated if the AHEN bit is clear
- The D/A and R/W status bits are cleared
- The CLTACT (I2CxSTAT2[30]) bit is set
- The module generates the I2CxIF interrupt on the falling edge of the ninth SCLx clock if CADDRIE (I2CxINTC[10]) bit and CSTIE(I2CxINTC[12]) are enabled
- The module waits for the host to send data

Figure 20-23. Client Write 7-Bit Address Detection Timing Diagram

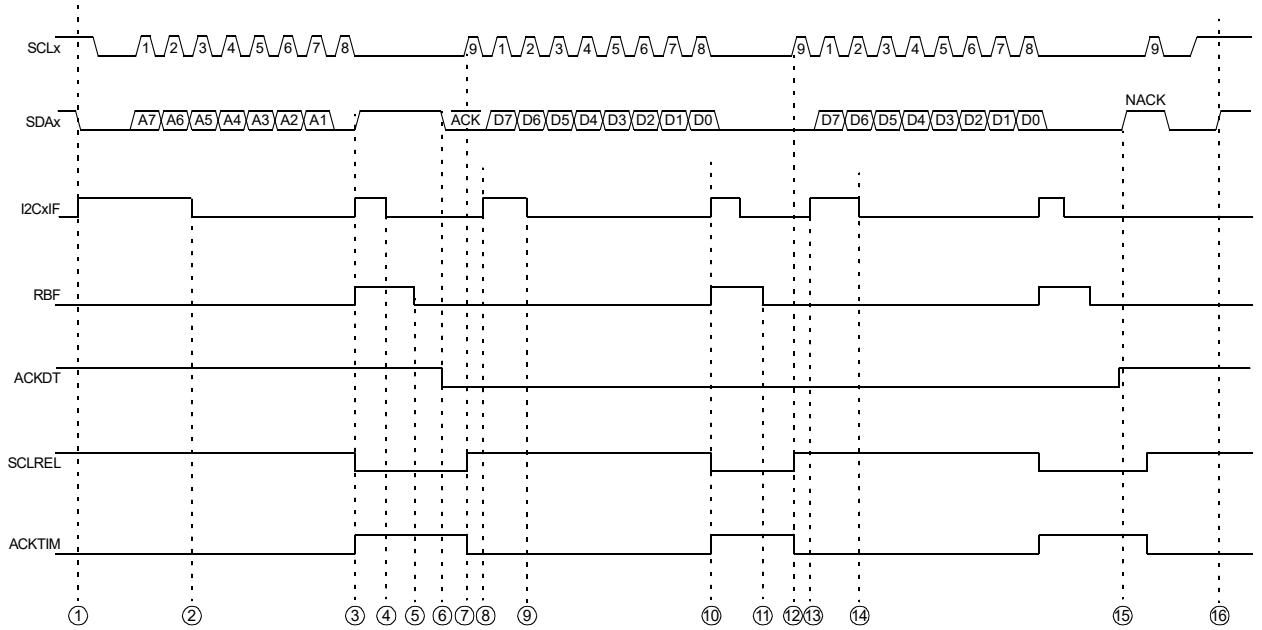


20.5.4.3.4 7-Bit Address and Client Write with the AHEN and DHEN Bits

The client device reception, with the AHEN and DHEN bits set, operates with extra interrupts and clock stretching added after the eighth falling edge of SCLx. These additional interrupts allow the client software to decide whether it wants to ACK the receive address or data byte rather than the hardware.

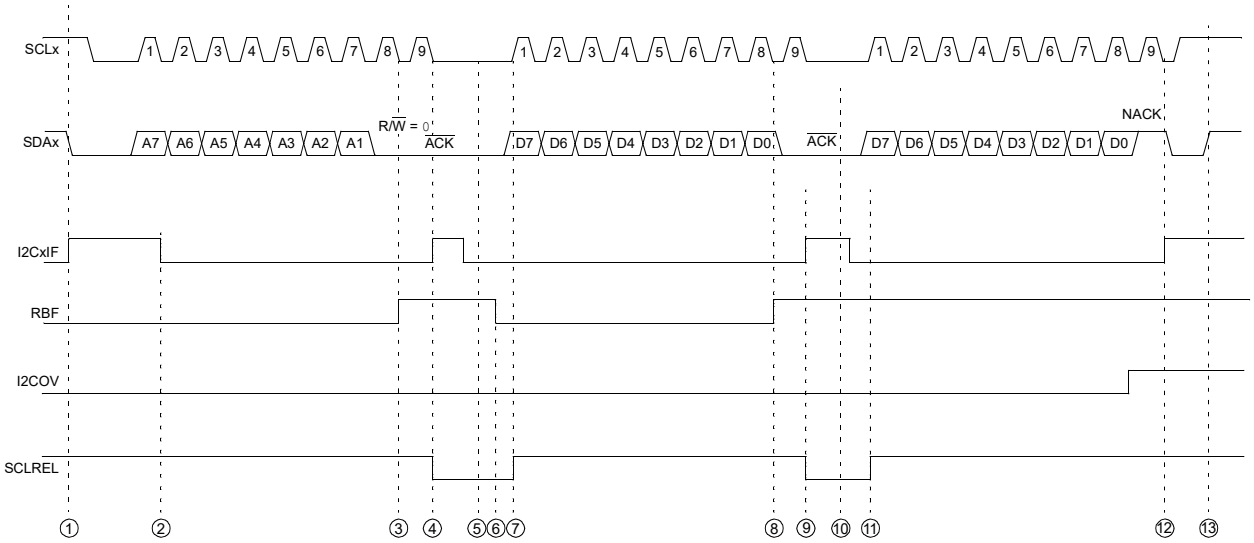
Note: The I2CxIF interrupt is still set after the ninth falling edge of the SCLx clock, even if there is no clock stretching and the RBF bit has been cleared. The I2CxIF interrupt is not asserted if a NACK is sent to the host.

Figure 20-24. I²C Client, 7-Bit Address, Reception (STREN = 0, AHEN = 1, DHEN = 1)



1. Detecting Start bit enable address detection, interrupt flag is set if SCEN is set.
2. User software clears the interrupt flag.
3. Client receives the address byte with R/$\overline{W}$ = 0. Hardware clears SCLREL. Interrupt flag is asserted. ACKTIM is asserted. I2CxRCV is loaded with I2CxRSR and RBF is asserted.
4. User software clears the interrupt flag.
5. User software reads I2CxRCV, that clears the RBF flag.
6. ACKDT is written with $\overline{ACK}$ by user software.
7. User software sets SCLREL bit to release clock, ACKTIM is cleared by hardware.
8. Interrupt flag is set (not set if NACK is received).
9. User software clears the interrupt flag.
10. If DHEN = 1, hardware clears the SCLREL bit. I2CxRCV is loaded with I2CxRSR; ACKTIM is asserted at the end of 8th falling edge of SCLx by hardware.
11. User software reads I2CxRCV; clears the RBF flag.
12. User software releases the SCLREL bit, ACKTIM is cleared by hardware.
13. Interrupt flag is set.
14. User software clears the interrupt flag.
15. NACK.
16. Client recognizes the Stop event.

Figure 20-25. I²C Client, 7-Bit Address, Reception (STREN = 1, AHEN = 0, DHEN = 0)



① Detecting Start bit, enables address detection, interrupt is set if SCEN is set.
② User software clears the interrupt flag.
③ RBF is set on the 8th falling clock, address is loaded into I2CxRCV. RBF is asserted.
④ Interrupt is asserted.
⑤ SCLx is stretched low until SCLREL is set.
⑥ User software reads the I2CxRCV buffer, that clears the RBF flag.
⑦ User software releases the SCLx line by writing SCLREL to '1'.
⑧ Data is loaded into I2CxRCV. RBR flag is asserted.
⑨ On the 9th falling clock edge, interrupt is asserted.
⑩ SCLx is stretched and held low until SCLREL is set.
⑪ User software releases SCLx line by writing SCLREL to '1'.
⑫ NACK is received (SCLx is not stretched to low).
⑬ Client recognizes the Stop event.

20.5.4.3.5 7-Bit Address and Client Read

When a client read is specified by having R/W = 1 in a 7-bit address byte, the process of detecting the device address is similar to that of a client write, as illustrated in Figure 20-26. If the addresses match, the following events occur:

- An $\overline{ACK}$ is generated if the AHEN bit is clear
- The D/$\overline{A}$ status bit is cleared and the R/W status bit is set
- The module generates the I2CxIF interrupt on the falling edge of the ninth SCLx clock if the CADDRIE (I2CxINTC[10]) bit and CSTIE(I2CxINTC[12]) are enabled

Because the client is expected to reply with data at this point, it is necessary to suspend the operation of the I²C bus to allow the user software to prepare a response. This is done automatically when the module clears the SCLREL bit. With SCLREL low, the client will pull down the SCLx clock line, causing a Wait on the I²C bus. The SSPND (I2CxSTAT2[31]) bit will be set by the hardware to indicate clock stretching. Packet size (PSZ, I2CxCON2[15:0]) and ND/$\overline{A}$ needs to be configured and then the user software writes the I2CxTRN register with the response data. SSPND (I2CxSTAT2[31]) bit will be automatically cleared by hardware. If Smart Mode (SMEN, I2CxCON2[17]) is disabled, the user must set SCLREL to release the clock or the hardware will automatically set SCLREL to release the clock.

Data setup time (TSU:DAT) can be configured by writing the required setup time to SDASUT(I2CXSDASUT[15:0]) and then enabling SDASUTEN (I2CXSDASUT[31]). In SMART mode (SMEN, I2CxCON2 [17]), hardware will automatically set SCLREL after data setup time.

Once packet size becomes zero, the end of packet (EOP) will be set (I2CxSTAT2[24]).

Note: For more information on the AHEN and DHEN bits, refer to section 20.5.4.3.4. 7-Bit Address and Client Write with the AHEN and DHEN Bits.
The SCLREL bit will automatically clear after detecting the client read address, irrespective of the state of the STREN bit.

Figure 20-26. Client Read 7-Bit Address Detection Timing Diagram (AHEN = 0)

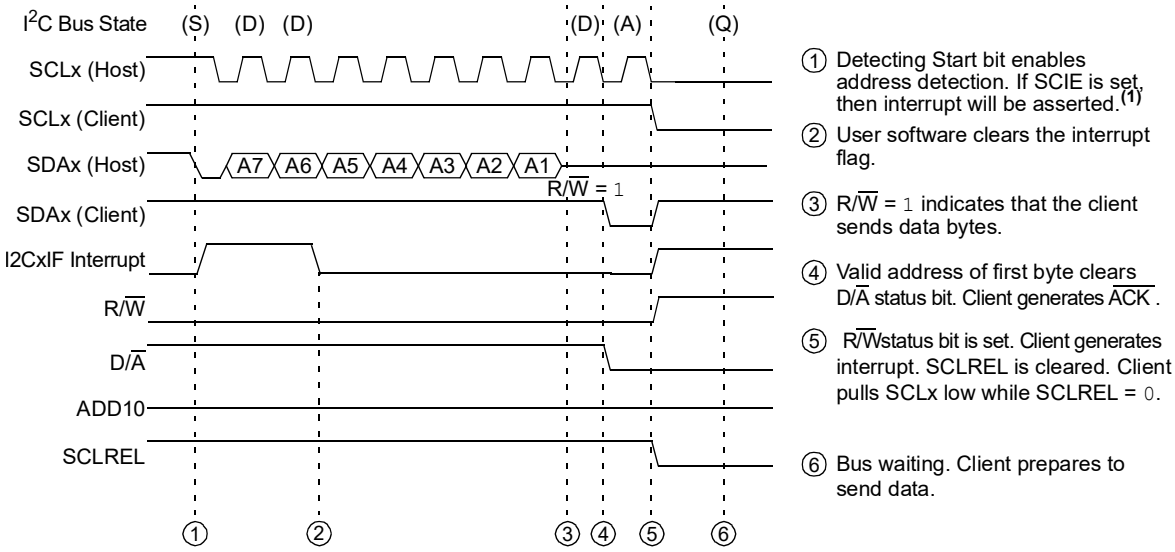
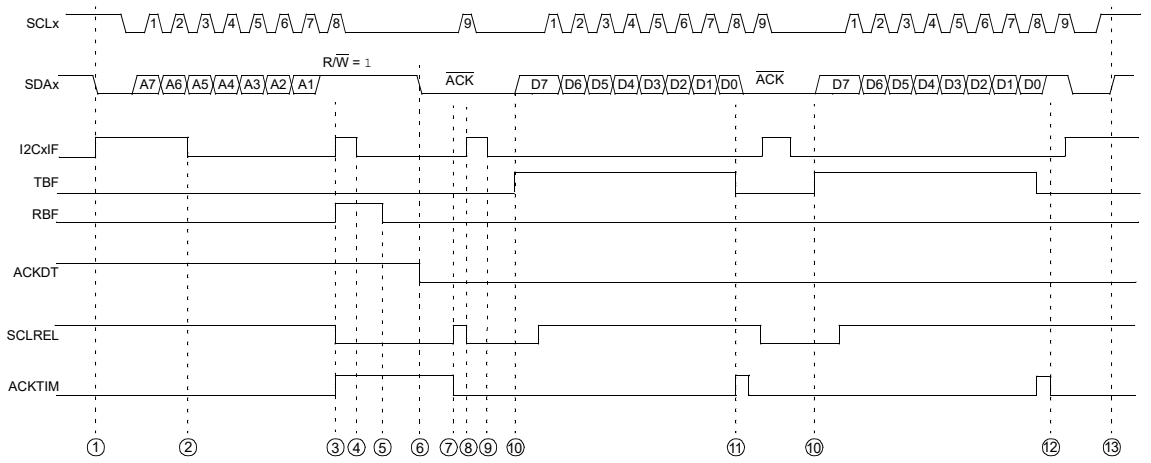


Figure 20-27. I²C Client, 7-Bit Address, Transmission (AHEN = 1)



1. Detecting Start bit enables address detection, interrupt is set if the SCIE bit is set.
2. User software clears the interrupt flag.
3. Client receives the address byte with R/$\overline{W}$ = 1. Hardware clears SCLREL to suspend host clock. ACKTIM and interrupt flag are asserted.
4. User software clears the interrupt flag.
5. Software reads the I2CxRV register, that clears the RBF flag.
6. ACKDT is written with $\overline{ACK}$.
7. User software sets SCLREL to release clock hold. Host clocks in the Acknowledgment sequence. ACKTIM is cleared by hardware.
8. Hardware clears SCLREL to suspend host clock if R/$\overline{W}$ = 1.
9. User software clears the interrupt flag.
10. User software loads the I2CxTRN register with response data. TBF = 1 indicates that the buffer is full.
11. After last bit, module clears TBF bit, indicating buffer is available for next byte.
12. At the end of ninth clock, if host sent NACK, no more data is expected. Module does not suspend the clock.
13. Module recognizes Stop event.

20.5.4.3.6 10-Bit Addressing Mode

In 10-Bit Addressing mode, the client must receive two device address bytes, as illustrated in Figure 20-28. The five Most Significant bits (MSbs) of the first address byte specify a 10-bit address. The R/W status bit of the address must specify a write, causing the client device to receive the second address byte. For a 10-bit address, the first byte would equal, '11110 A9 A8 0', where A9 and A8 are the two MSbs of the address.

The I2CxMSK register can mask any bit position in a 10-bit address. The 2 MSbs of the I2CxMSK register are used to mask the MSbs of the incoming address received in the first byte. The remaining byte of the register is then used to mask the lower byte of the address received in the second byte.

Following the Start condition, the module shifts eight bits into the I2CxRSR register. The value of the I2CxRSR[2:1] bits is evaluated against the value of the I2CxADD[9:8] and I2CxMSK[9:8] bits, while the value of the I2CxRSR[7:3] bits is compared to '11110'. Address evaluation occurs on the falling edge of the eighth SCLx clock. For the address to be valid, the I2CxRSR[7:3] bits must be equal to '11110', while the I2CxRSR[2:1] bits must exactly match any unmasked bits in the I2CxADD[9:8] bits (if both bits are masked, a match is not needed). If the address is valid, the following events occur:

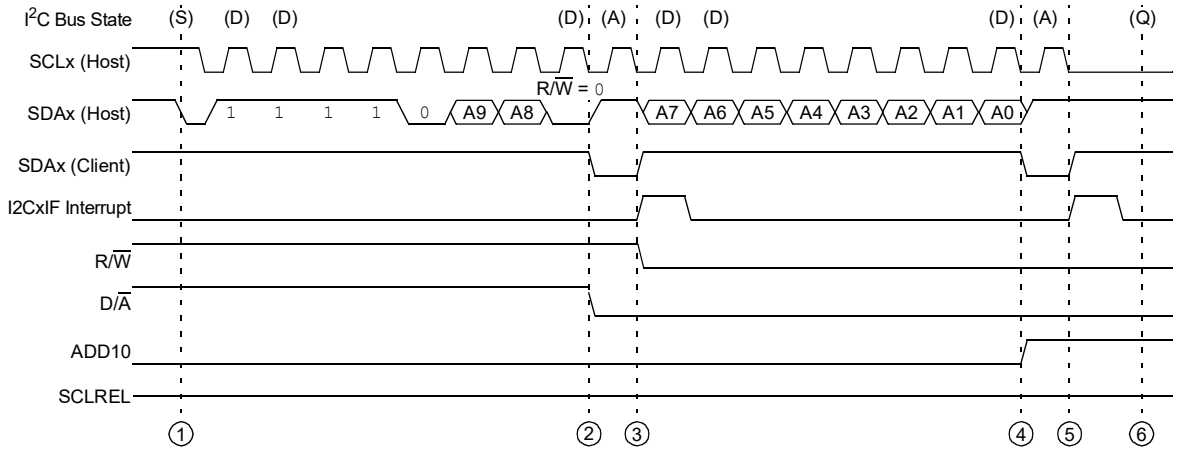
- An $\overline{ACK}$ is generated
- The D/$\overline{A}$ and R/$\overline{W}$ status bits are cleared
- The module generates the I2CxIF interrupt on the falling edge of the ninth SCLx clock if the CADDRIE (I2CxINTC[10]) bit and CSTIE(I2CxINTC[12]) are enabled.

The module does generate an interrupt after the reception of the first byte of a 10-bit address; however, this interrupt is of little use.

The module will continue to receive the second byte into the I2CxRSR register. This time, the I2CxRSR[7:0] bits are evaluated against the I2CxADD[7:0] and I2CxMSK[7:0] bits. If the lower byte of the address is valid, as previously described, the following events occur:

- An $\overline{ACK}$ is generated
- The ADD10 status bit is set
- The module generates the I2CxIF interrupt on the falling edge of the ninth SCLx clock if the CADDRIE (I2CxINTC[10]) bit and CSTIE (I2CxINTC[12]) are enabled.
- The module will wait for the host to send data or initiate a Repeated Start condition
  Note: Following a Repeated Start condition in 10-Bit Addressing mode, the client only matches the first 7-bit address, '11110 A9 A8 0'.

Figure 20-28. 10-Bit Address Detection Timing Diagram (AHEN = 0)



(1) Detecting the Start bit enables address detection.

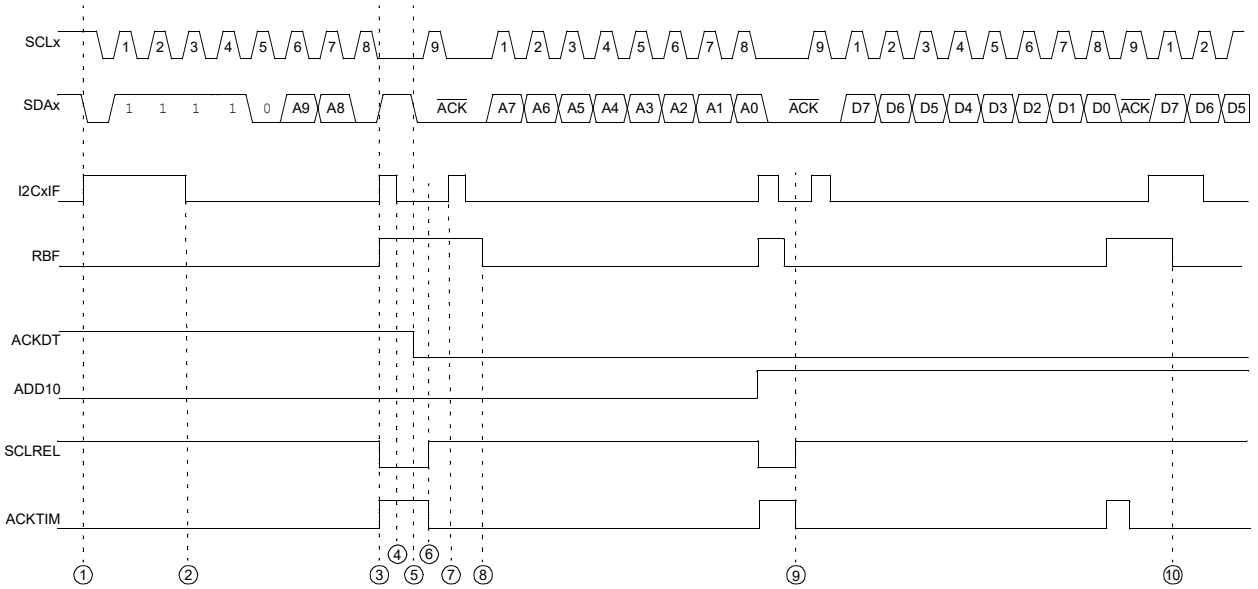
(2) Address match of first byte clears the D/$\overline{A}$ status bit and causes client logic to generate an $\overline{ACK}$.

(3) Reception of first byte clears the R/$\overline{W}$ status bit. Client logic generates an interrupt.

(4) Address match of first and second byte sets the ADD10 status bit and causes client logic to generate an $\overline{ACK}$.

(5) Reception of second byte completes the 10-bit address. Client logic generates an interrupt.

Figure 20-29. I²C Client, 10-Bit Address, Reception (STREN = 0, AHEN = 1, DHEN = 0)



① Detecting the Start bit enables address detection; interrupt is set if the SCEN bit is set.

② User software clears the interrupt flag.

③ Client receives first address byte. Write indicated. Interrupt flag is asserted. ACKTIM is asserted. If AHEN = 1, client suspends clock. SCLREL is cleared by hardware.

④ User software clears the interrupt flag.

⑤ ACKDT is written with $\overline{ACK}$ by user software.

⑥ User software sets SCLREL to release clock hold.

⑦ Client interrupt is asserted.

⑧ User software reads I2CxRCV buffer, that clears RBF flag.

⑨ Client Acknowledges the second address byte.

⑩ User software reads data from the I2CxRCV register.

20.5.4.3.7 Client Mode Bus Collision

On a read request from the host, the client begins shifting data out on the SDAx line. If a bus collision is detected and the SBCDE bit (I2CxCON1[18]) is set, then the I2CxBCIF bit will be set. After detecting the bus collision, the client goes into Idle mode and waits to be addressed again. User software can use the I2CxBCIF bit or vectors to the bus collision interrupt to handle a client bus collision.

20.5.4.3.8 General Call Operation

The addressing procedure for the I²C bus is such that the first byte after a Start condition usually determines which client device the host is addressing. The exception is the general call address, which can address all devices. When this address is used, all the enabled devices respond with an Acknowledge. The general call address is one of the eight addresses reserved for specific purposes by the I²C protocol. It consists of all '0's with R/W = 0. The general call is always a client write operation.

The general call address is recognized when the General Call Enable bit, GCEN (I2CxCON1[7], is set, as illustrated in Figure 20-30. Following a Start bit detect, eight bits are shifted into the I2CxRSR register, and the address is compared against the I2CxADD register and the general call address.

If the general call address matches, the following events occur:

- An $\overline{ACK}$ is generated
- The client will set the GCSTAT status bit (I2CxSTAT1[9])
- The D/$\overline{A}$ and R/$\overline{W}$ status bits are cleared
- The module generates the I2CxIF interrupt on the falling edge of the ninth SCLx clock if the CADDRIE (I2CxINTC[10]) bit and CSTIE(I2CxINTC[12]) are enabled

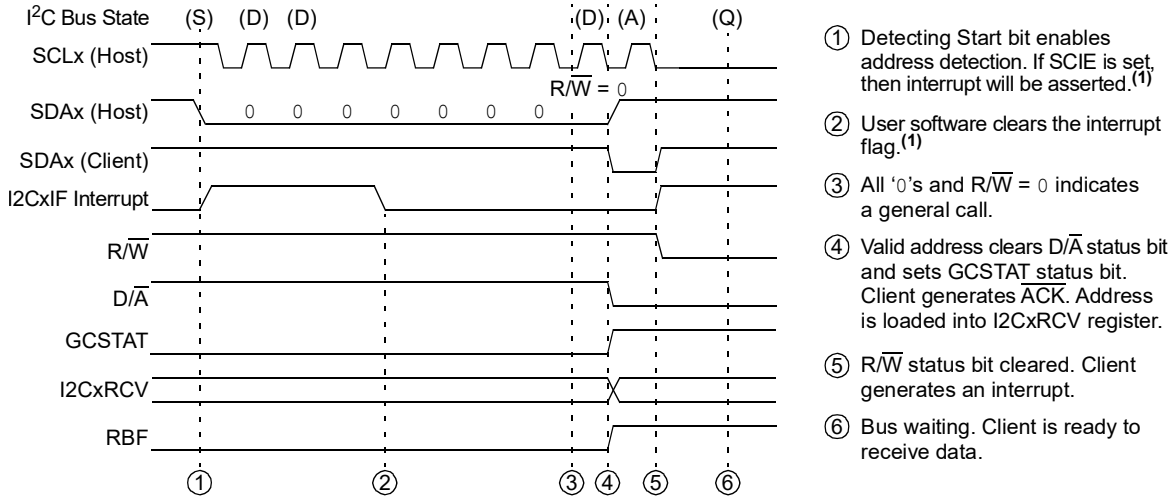
- The I2CxRSR register is transferred to the I2CxRCV register and the RBF status bit (I2CxSTAT[1]) is set (during the eighth bit)
- The module waits for the host to send data

When the interrupt is serviced, the cause for the interrupt can be checked by reading the contents of the GCSTAT status bit to determine if the device address was device-specific or a general call address.

Notes:

1. General call addresses are 7-bit addresses. If configuring the client for 10-bit addresses and the A10M and GCEN bits are set, the client will continue to detect the 7-bit general call address.
2. The client will Acknowledge the general call address (7-bit address, 0x00) only if GCEN is set and independent of the STRICT and A10M bits.

Figure 20-30. General Call Address Detection Timing Diagram (GCEN = 1, AHEN =0)



20.5.4.3.9 Strict Support

The client module Acknowledges all the addresses, including the reserved addresses, when STRICT reserved addressing is not enforced (STRICT = 0). The client device does not Acknowledge the reserved address space if the STRICT bit (I2CxCON1[11]) is set.

Table 20-4. Client Response to Reserved Addresses

STRICT Bit	I2CxADD Client Address	Received Address into I2CxRSR	Client Acknowledge
x	0x1F	0x1F	ACK
1	0x1F	C-Bus Address	NACK
1	C-Bus Address	C-Bus Address	NACK
0	C-Bus Address	C-Bus Address	ACK
0	C-Bus Address	0x1F	NACK
0	0x1F	C-Bus Address	NACK

Note: When the STRICT bit is cleared, the ACK signal is generated only if the address is matched, even for reserved addresses. The client device does not generate an ACK if there is an address mismatch, even if the address is a reserved address. Irrespective of the STRICT bit setting and whether the address is reserved or not, an ACK signal is generated for a proper address match.

20.5.4.3.10 When an Address is Invalid

If a 7-bit address does not match the contents of the I2CxADD[6:0] bits, the client will return to an Idle state and ignore any activity on the I²C bus until after the Stop condition.

If the first byte of a 10-bit address does not match the contents of the I2CxADD[9:8] bits, the client will return to an Idle state and ignore all bus activity until after the Stop condition.

If the first byte of a 10-bit address matches the contents of the I2CxADD[9:8] bits, but the second byte of the 10-bit address does not match the I2CxADD[7:0] bits, the client will return to an Idle state and ignore all bus activity until after the Stop condition.

20.5.4.3.11 Addresses Reserved from Masking

Even when enabled, there are several addresses that are ignored by the I²C module. For these addresses, an Acknowledge will not be issued independent of the mask setting. These addresses are listed in Table 20-5.

Table 20-5. Reserved I²C Bus Addresses(3)

7-Bit Address Mode		
Client Address	**R/W Bit**	**Description**
0000 000	0	General Call Address(1)
0000 000	1	Start Byte
0000 001	x	C-Bus Address
0000 010	x	Reserved
0000 011	x	Reserved
0000 1xx	x	HS Mode Master Code
1111 1xx	x	Reserved
1111 0xx	x	10-Bit Client Upper Byte(2)

Notes:

1. Address will be Acknowledged only if GCEN = 1.
2. A match on this address can only occur as the upper byte in 10-Bit Addressing mode.
3. These addresses will not be Acknowledged, independent of mask settings and STRICT = 1.

20.5.4.4 Receiving Data from a Host Device

When the R/$\overline{W}$ status bit of the device address byte is '0' and an address match occurs, the R/$\overline{W}$ status bit (I2CxSTAT1[2]) is cleared. The client enters a state waiting for data to be sent by the host. After the device address byte, the contents of the data byte are defined by the system protocol and are only received by the client. User software should configure data packet size in PSZ (I2CxCON2).

The client shifts eight bits into the I2CxRSR register. On the falling edge of the eighth clock (SCLx), the following events occur:

- The module begins to generate an $\overline{ACK}$ or NACK.
- The RBF status bit (I2CxSTAT1[1]) is set to indicate received data.
- The I2CxRSR register byte is transferred to the I2CxRCV register for access by the user software.
- The D/$\overline{A}$ status bit is set.
- A client interrupt is generated. User software can check the status of the I2CxSTAT register to determine the cause of the event and then clear the I2CxIF interrupt flag if the CDRXIE (I2CxINTC[4]) bit and CSTIE (I2CxINTC[12]) are enabled.
- The module waits for the next data byte.
- Once packet size becomes zero, end of packet (EOP) will be set (I2CxSTAT2[24]).

20.5.4.4.1 Acknowledge Generatrion

Normally, the client Acknowledges all the received bytes by sending an $\overline{ACK}$ on the ninth SCLx clock. If the receive buffer is overrun, the client does not generate this $\overline{ACK}$. The overrun is indicated if either (or both) of the following occur:

- The Receive Buffer Full bit, RBF (I2CxSTAT1[1]), was set before the transfer was received
- The Receive Overflow bit, I2COV (I2CxSTAT1[6]), was set before the transfer was received

Table 20-6 shows what happens when a data transfer byte is received, given the status of the RBF and I2COV status bits. If the RBF status bit is already set when the client attempts to transfer to the I2CxRCV register, the transfer does not occur, but the interrupt is generated and the I2COV status bit is set. If both the RBF and I2COV status bits are set, the client acts similarly. The shaded cells show the condition where user software did not properly clear the overflow condition.

Reading the I2CxRCV register clears the RBF status bit. The I2COV status bit is cleared by writing to a '0' through user software.

Note: If the BOEN bit (I2CxCON1[20]) is set, then the I2COV bit (I2CxSTAT1[6]) is ignored and only the RBF bit (I2CxSTAT1[1]) determines whether the module will Acknowledge the message or not.

20.5.4.4.2 Receive Buffer Overwrite (I^2C Client Mode Only)

If the BOEN bit (I2CxCON1[20]) is set, then the I2COV bit (I2CxSTAT1[6]) is ignored. If the RBF bit (I2CxSTAT1[1]) is not set, then the $\overline{ACK}$ is generated for the receive address or data; the I2CxRCV buffer is updated with I2CxRSR.

Table 20-6. Data Transfer Received Byte Actions

Status Bits as Data Byte Received			Transfer I2CxRSR to I2CxRCV	Generate ACK	Generate I2CxIF Interrupt (interrupt occurs if enabled)	Set RBF	Set I2COV
BOEN(1)	RBF	I2COV					
x	0	0	Yes	Yes	Yes	Yes	No change
x	1	0	No	No	Yes	No change	Yes
x	1	1	No	No	Yes	No change	Yes
0	0	1	Yes	No	Yes	Yes	No change
1	0	1	Yes	Yes	Yes	Yes	No change

20.5.4.4.3 Wait States During Client Receptions

When the client receives a data byte, the host can potentially begin sending the next byte immediately. This allows the user software controlling the nine client SCLx clock periods to process the previously received byte. If this is not enough time, the client software may want to generate a bus Wait period.

The STREN bit (I2CxCON1[6]) enables a bus Wait to occur on client receptions. When STREN = 1 at the falling edge of the ninth SCLx clock of a received byte, the client clears the SCLREL bit. Clearing the SCLREL bit causes the client to pull the SCLx line low, initiating a Wait. The SCLx clock of the host and client will synchronize, as provided in 20.5.3.2. Master Clock Synchronization.

When the user software is ready to resume reception, the user software sets the SCLREL bit. This causes the client to release the SCLx line and the host resumes clocking. If SMART mode (SMEN, I2CxCON2[17]) is enabled, hardware will automatically release SCL by setting SCLREL when the received byte is read (RBF=0) depending on EOP (I2CxSTAT2[24]) and EPSZE (I2CxCON2[24]).

20.5.4.4.4 Examples Messages of Client Reception

Receiving a client message is an automatic process. The user software handling the client protocol uses the client interrupt to synchronize to the events.

When the client detects the valid address, the associated interrupt will notify the user software to expect a message. Upon receiving the data, as each data byte transfers to the I2CxRCV register, an interrupt notifies the user software to unload the buffer.

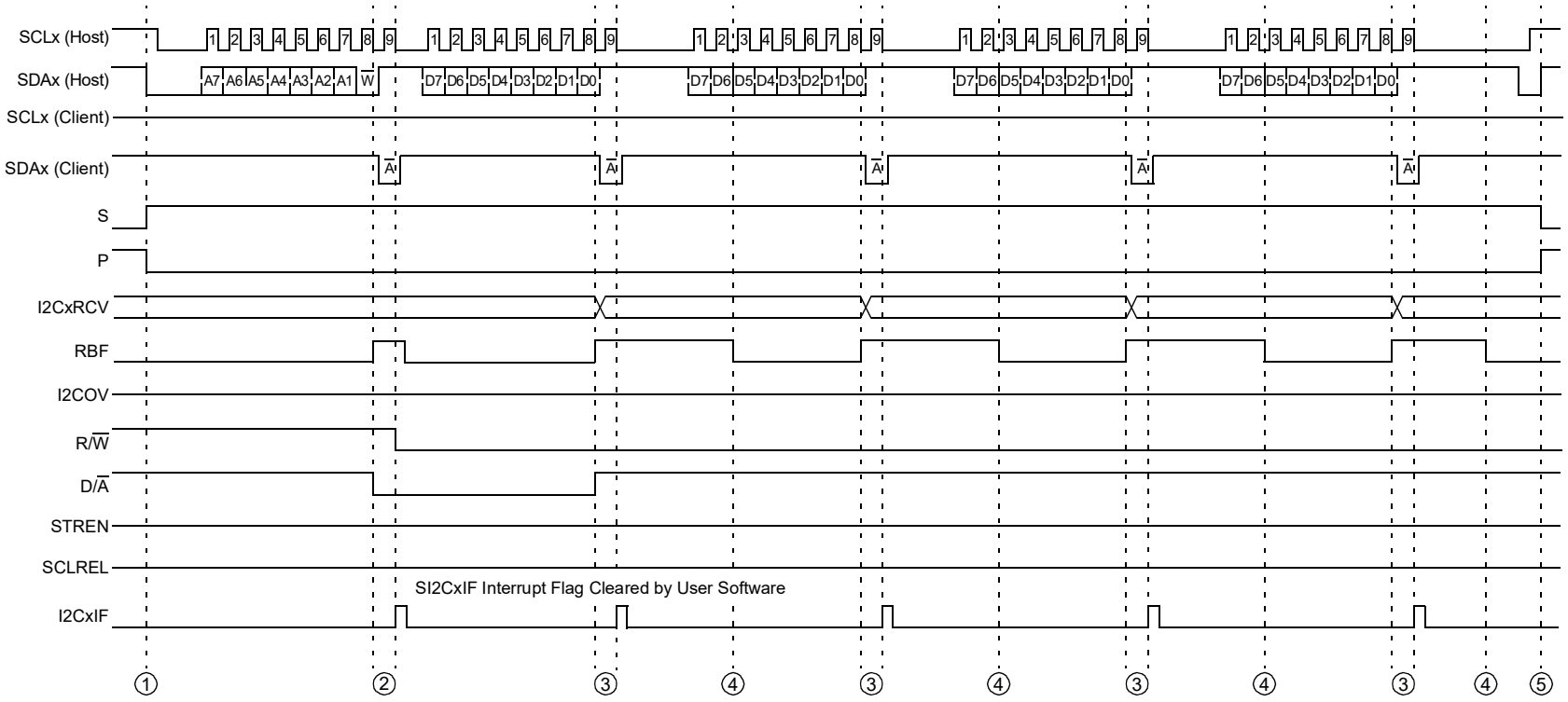
Figure 20-31 illustrates a simple receive message. Because it is a 7-bit address message, only one interrupt occurs for the address bytes. Then, interrupts occur for each of four data bytes. At an interrupt, the user software may monitor the status bits, RBF (I2CxSTAT1[1]), D/$\overline{A}$ (I2CxSTAT1[5]) and R/$\overline{W}$ (IC2xSTAT1[2]), to determine the condition of the byte received.

Figure 20-32 illustrates a similar message using a 10-bit address. In this case, two bytes are required for the address.

Figure 20-33 illustrates a case where the user software does not respond to the received byte and the buffer overruns. On reception of the second byte, the module will automatically NACK the host transmission. Generally, this causes the host to resend the previous byte. The I2COV status bit (I2CxSTAT1[6]) indicates that the buffer has overrun. The I2CxRCV register buffer retains the contents of the first byte. On reception of the third byte, the buffer is still full, and again, the module will NACK the host. After this, the user software finally reads the buffer. Reading the buffer will clear the RBF status bit; however, the I2COV status bit remains set. The user software must clear the I2COV status bit (I2CxSTAT1[6]). The next received byte is moved to the I2CxRCV register buffer and the module responds with an $\overline{ACK}$.

Figure 20-34 highlights clock stretching while receiving data. In the previous examples, the STREN bit (I2CxCON1[6]) is equal to '0', which disables clock stretching on receive messages. In this example, the user software sets STREN to enable clock stretching. When STREN = 1, the module will automatically clock stretch after each received data byte, allowing the user software more time to move the data from the buffer. If RBF = 1 at the falling edge of the ninth clock, the module automatically clears the SCLREL bit (I2CxCON1[12]) and pulls the SCLx bus line low. As shown with the second received data byte, if the user software can read the buffer and clear the RBF status bit before the falling edge of the ninth clock, the clock stretching will not occur. The user software can also suspend the bus at any time. By clearing the SCLREL bit, the module pulls the SCLx line low after it detects the bus SCLx low. The SCLx line remains low, suspending transactions on the bus until the SCLREL bit is set. See 20.6.5. Client Reception (7-bit Address) and 20.6.6. Client Reception (10-bit Address) for application examples.

Figure 20-31. Client Message (Write Data to Client: 7-Bit Address; Address Matches; A10M = 0, GCEN = 0, IPMIEN = 0, AHEN = 0, DHEN = 0, STRICT = 0 and BOEN = 0)



① Client recognizes Start event; S and P status bits set/clear accordingly.

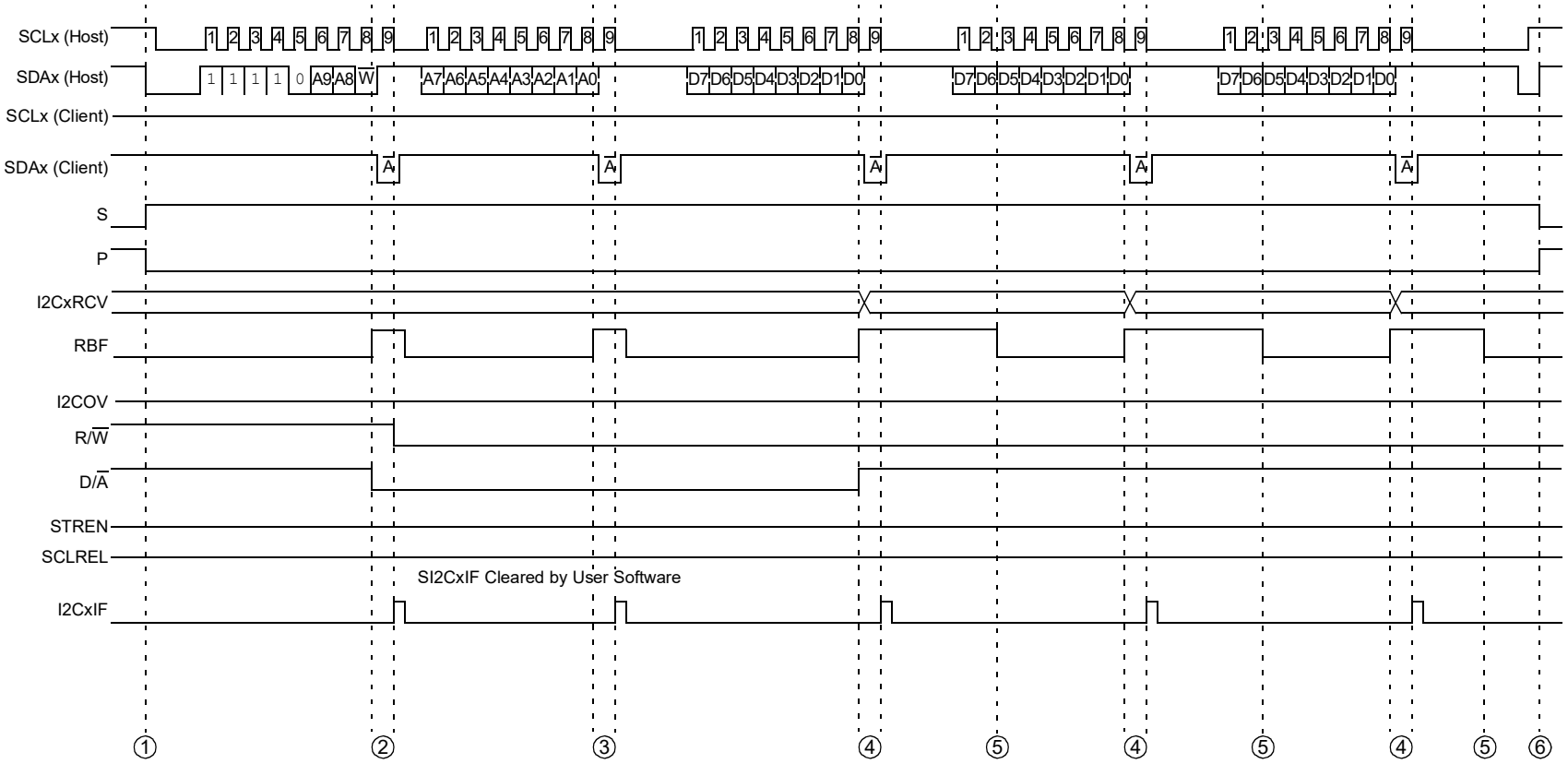
② Client receives address byte. Address matches. Client Acknowledges and generates interrupt. Address byte is moved to the I2CxRCV register and must be read by user software to prevent buffer overflow.

③ Next received byte is message data. The byte moved to the I2CxRCV register sets the RBF status bit. Client generates interrupt. Client Acknowledges reception.

④ User software reads the I2CxRCV register. RBF status bit clears.

⑤ Client recognizes Stop event; S and P status bits set/clear accordingly.

Figure 20-32. Client Message (Write Data to Client: 10-Bit Address; Address Matches; A10M = 1, GCEN = 0, IPMIEN = 0, AHEN = 0, DHEN = 0, STRICT = 0 and BOEN = 0)



① Client recognizes Start event, S and P bits set/clear accordingly.

② Client receives address byte. High-order address matches. Client Acknowledges and generates interrupt. Address byte is moved to the I2CxRCV register and is read by user software to prevent buffer overflow.

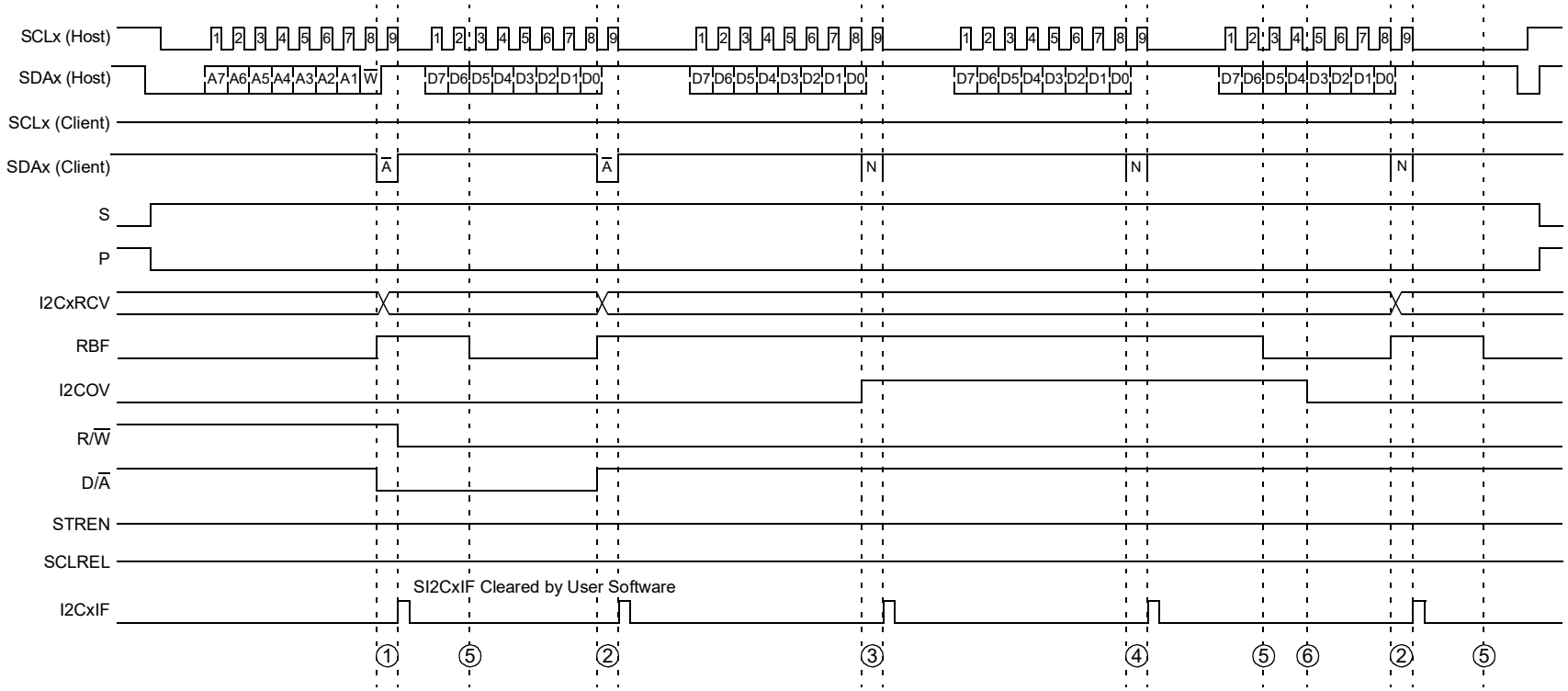
③ Client receives address byte. Low-order address matches. Client Acknowledges and generates interrupt. Address byte is moved to the I2CxRCV register and is read by user software to prevent buffer overflow.

④ Next received byte is message data. Byte moved to the I2CxRCV register, sets RBF. Client Acknowledges and generates interrupt.

⑤ User software reads the I2CxRCV register. RBF bit clears.

⑥ Client recognizes Stop event, S and P bits set/clear accordingly.

Figure 20-33. Client Message (Write Data to Client: 7-Bit Address; Buffer Overrun; A10M = 0, GCEN = 0, IPMIEN = 0, AHEN = 0, DHEN = 0, STRICT = 0 and BOEN = 0)



① Client receives address byte. Address matches. Client generates interrupt. Address byte is moved to the I2CxRCV register and must be read by user software to prevent buffer overflow.

② Next received byte is message data. The byte is moved to the I2CxRCV register, sets RBF. Client generates interrupt. Client Acknowledges reception.

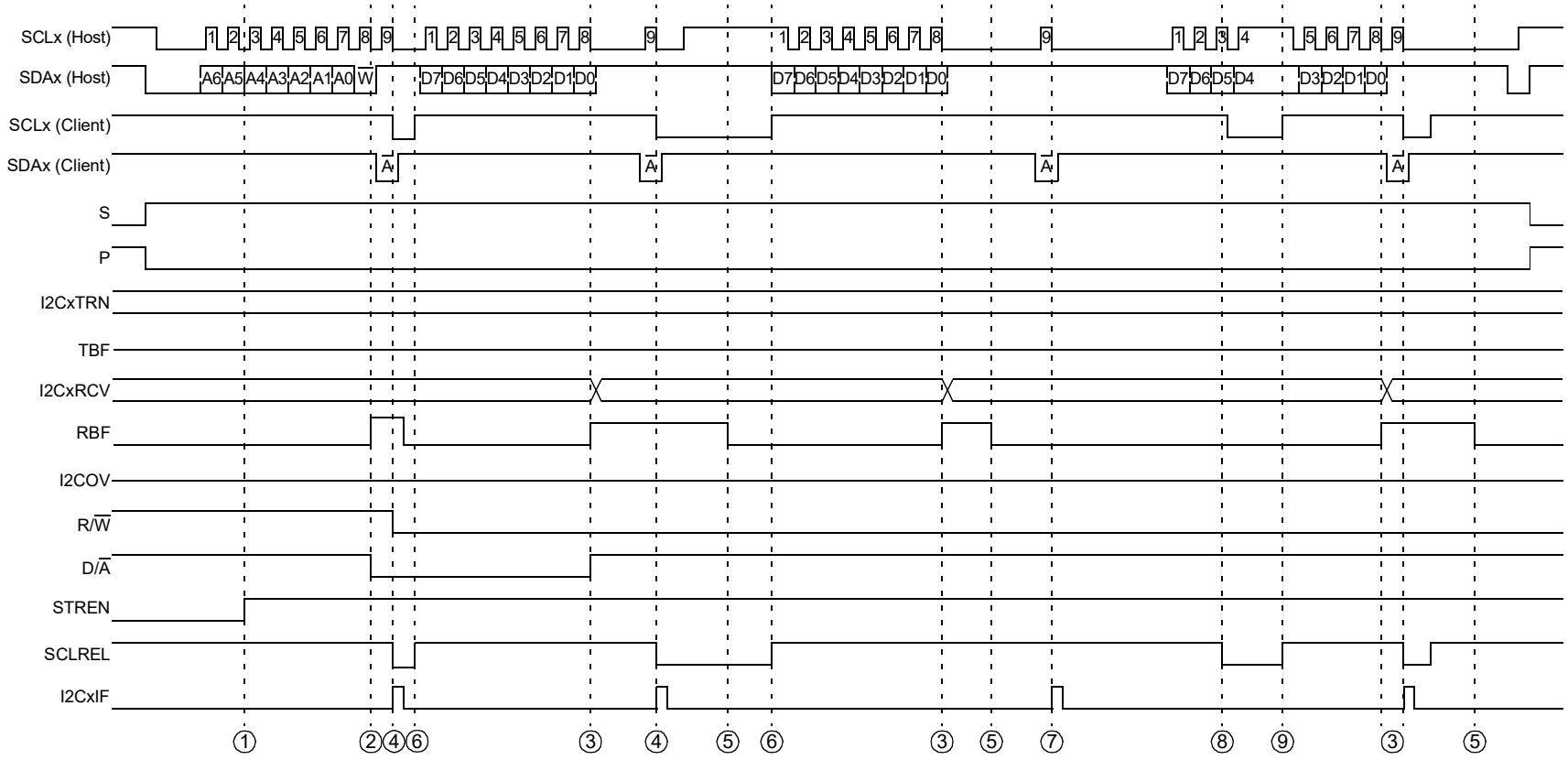
③ Next byte received before I2CxRCV is read by software. I2CxRCV register is unchanged. I2COV overflow bit is set. Client generates interrupt. Client sends NACK for reception.

④ Next byte also received before I2CxRCV is read by software. I2CxRCV register is unchanged. Client generates interrupt. Client sends NACK for reception. The host state machine should not be programmed to send another byte after receiving a NACK in this manner. Instead, it should abort the transmission with a Stop condition or send a Repeated Start condition and attempt to retransmit the data.

⑤ User software reads the I2CxRCV register. RBF bit clears.

⑥ User software clears the I2COV bit. Reception will still not be able to proceed normally until the module sees a Stop/Repeated Start bit. If neither of these conditions is met, an additional transmission will be received correctly, but sends a NACK and sets the I2COV bit again.

Figure 20-34. Client Message (Write Data to Client: 7-Bit Address; Clock Stretching Enabled; A10M = 0, GCEN = 0, IPMIEN = 0, AHEN = 0, DHEN = 0, STRICT = 0 and BOEN = 0)



① User software sets the STREN bit to enable clock stretching.

② Client receives address byte. I2CxRCV register is read by user software to prevent buffer overflow.

③ Next received byte is message data. The byte moves to the I2CxRCV register, sets RBF.

④ Because RBF = 1 at ninth clock, automatic clock stretch begins. Client clears SCLREL bit. Client pulls SCLx line low to stretch clock.

⑤ User software reads I2CxRCV register. RBF bit clears.

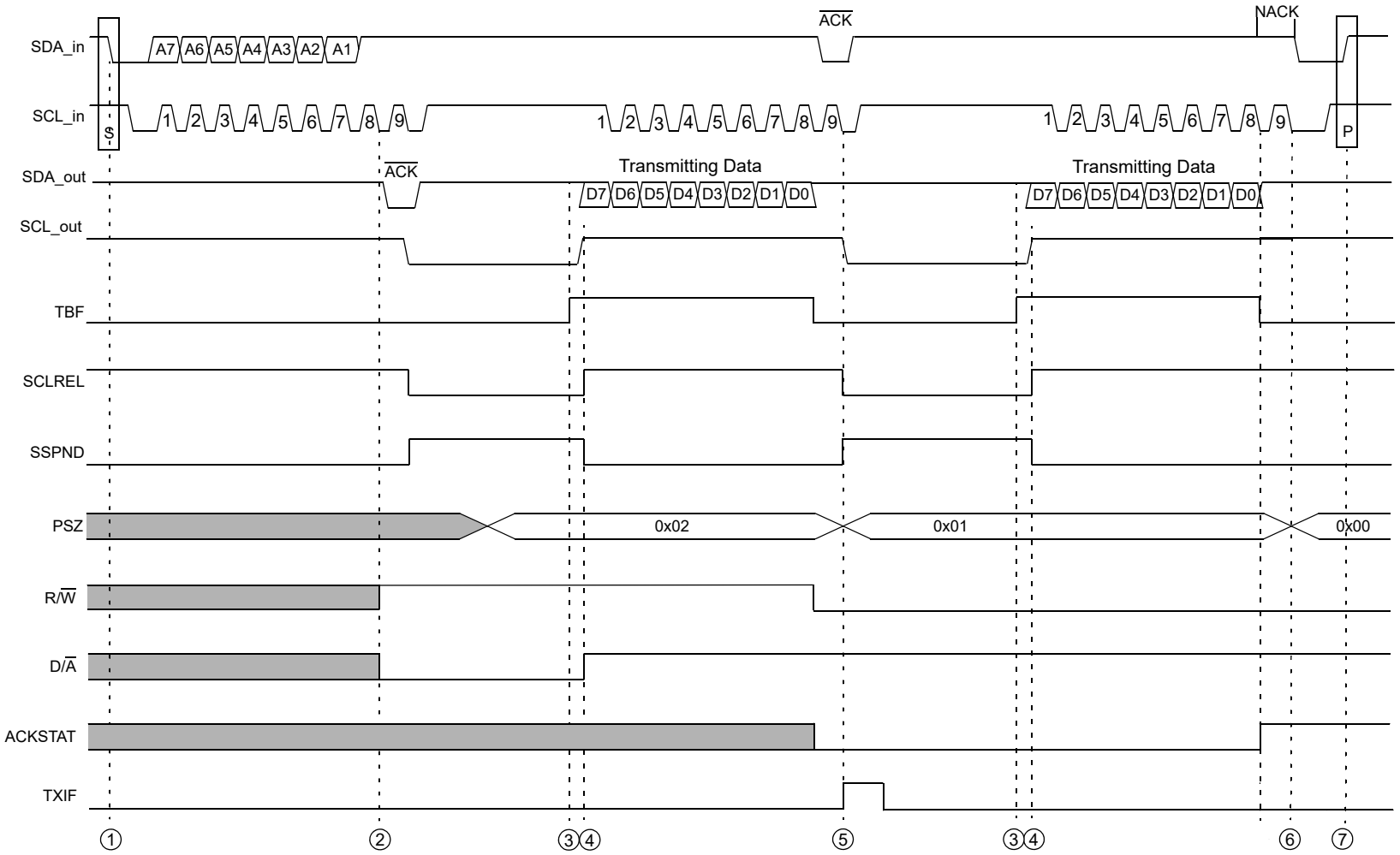
⑥ User software sets SCLREL bit to release clock.

⑦ Client does not clear SCLREL because RBF = 0 at this time.

⑧ User software may clear SCLREL to cause a clock hold. Module must detect SCLx low before asserting SCLx low.

⑨ User software may set SCLREL to release a clock hold.

Figure 20-35. I²C Client 7-Bit Address Transmission with Smart Mode Enabled (SMEN = 1)



① Client recognizes Start event, S and P bits set/clear accordingly.

② Client receives address byte. Address matches. Address byte is moved to I2CxRCV register and is read by user software to prevent buffer overflow. R/W = 1 to indicate read from Client.

③ User software writes I2CxTRN with response data. TBF = 1 indicates that buffer is full. Writing I2CxTRN sets D/A, indicating a data byte.

④ SCLREL will be set by hardware automatically to release the clock. SCLREL will be released after data setup if configured. SSPND will be cleared to indicate clock release.

⑤ I2CxTXIF will be set after transmitting data.

⑥ EOP will be set.

⑦ Client recognizes Stop event; S and P bits set/clear accordingly.

20.5.4.5 Sending Data to a Host Device

When the R/W status bit of the incoming device address byte is '1' and an address match occurs, the R/W status bit (I2CxSTAT1[2]) is set. At this point, the host device is expecting the client to respond by sending a byte of data. The contents of the byte are defined by the system protocol and are only transmitted by the client.

When the interrupt from the address detection occurs, the user software can write a byte to the I2CxTRN register to start the data transmission.

The client sets the TBF status bit (I2CxSTAT1[0]). The eight data bits are shifted out on the falling edge of the SCLx input. This ensures that the SDAx signal is valid during the SCLx high time. When all eight bits have been shifted out, the TBF status bit is cleared.

The client detects the Acknowledge from the host-receiver on the rising edge of the ninth SCLx clock.

If the SDAx line is low, indicating an $\overline{ACK}$, the host is expecting more data and the message is not complete. The module generates a client interrupt and the ACKSTAT status bit (I2CxSTAT1[15]) can be inspected to determine whether more data is being requested.

A client interrupt is generated on the falling edge of the ninth SCLx clock. User software must check the status of the I2CxSTAT register and clear the I2CxIF interrupt flag if the CDTXIE (I2CxINTC[3]) bit and CSTIE(I2CxINTC[12]) are enabled.

If the SDAx line is high, indicating a NACK, the data transfer is complete. User software should not write further data to the I2CxTRN register. The client resets and generates an interrupt, and it waits for detection of the next Start bit.

20.5.4.5.1 Wait States During Client Transmissions

During a client transmission message, the host expects return data immediately after detection of the valid address with R/W = 1. Because of this, the client automatically generates a bus Wait whenever the client returns data.

The automatic Wait occurs at the falling edge of the ninth SCLx clock of a valid device address byte or transmitted byte, Acknowledged by the host, indicating expectation of more transmit data.

The client clears the SCLREL bit (I2CxCON1[12]). Clearing the SCLREL bit causes the client to pull the SCLx line low, initiating a Wait. The SCLx clock of the host and client will synchronize, as shown in 20.5.3.2. Master Clock Synchronization.

When the user software loads the I2CxTRN register and is ready to resume transmission, the user software sets the SCLREL bit. This causes the client to release the SCLx line, and the host resumes clocking.

Note: The user software must provide a delay between writing to the transmit buffer and setting the SCLREL bit. This delay must be greater than the minimum setup time for client transmissions, as specified in the **"Electrical Characteristics"** chapter.

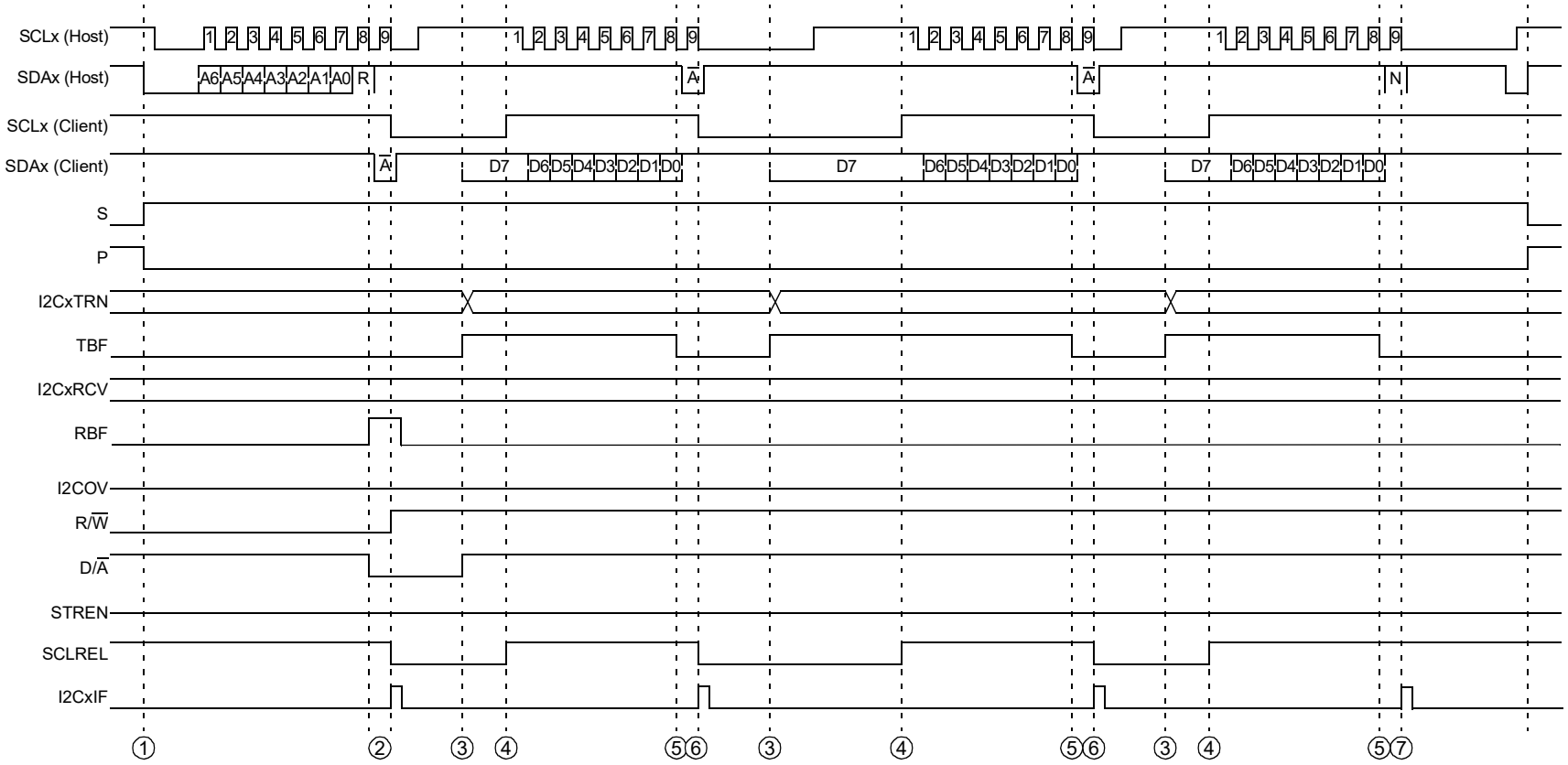
20.5.4.5.2 Example Messages of Client Transmission

The client transmissions for 7-bit address messages are illustrated in Figure 20-36. When the address matches and the R/W status bit of the address indicates a client transmission, the module automatically initiates clock stretching by clearing the SCLREL bit and generates an interrupt to indicate that a response byte is required. The user software writes the response byte into the I2CxTRN register. As the transmission completes, the host responds with an $\overline{ACK}$. If the host replies with an $\overline{ACK}$, the host expects more data, and the module again clears the SCLREL bit and generates another interrupt. If the host responds with a NACK, no more data is required and the module will not stretch the clock, but will generate an interrupt.

The client transmissions for 10-bit address messages require the client to first recognize a 10-bit address. Because the host must send two bytes for the address, the R/$\overline{W}$ Status bit in the first byte of the address specifies a write. To change the message to a read, the host sends a Repeated Start and repeats the first byte of the address with the R/$\overline{W}$ status bit, specifying a read. At this point,

the client transmission begins as illustrated in Figure 20-37. See 20.6.7. Client Transmission (7-bit Address) and 20.6.8. Client Transmission (10-bit Address) for application examples.

Figure 20-36. Client Message (Read Data from Client: 7-Bit Address)



① Client recognizes Start event, S and P bits set/clear accordingly.

② Client receives address byte. Address matches. Client generates interrupt. Address byte is moved to I2CxRCV register and is read by user software to prevent buffer overflow. R/$\overline{W}$ = 1 to indicate read from client. SCLREL = 0 to suspend host clock.

③ User software writes I2CxTRN with response data. TBF = 1 indicates that buffer is full. Writing I2CxTRN sets D/$\overline{A}$, indicating a data byte.

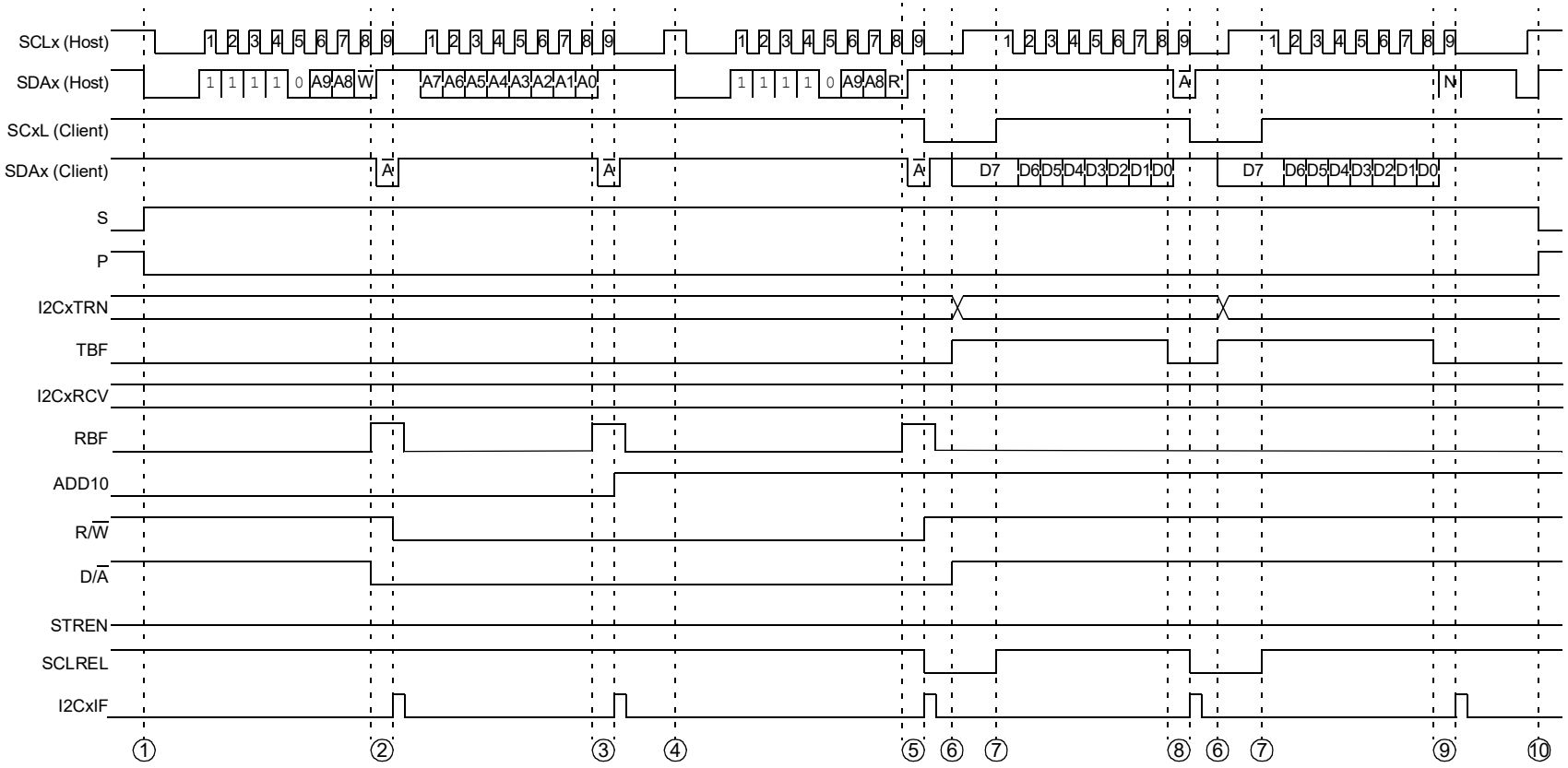
④ User software sets SCLREL to release clock hold. Host resumes clocking and client transmits data byte.

⑤ After last bit, module clears TBF bit, indicating buffer is available for next byte.

⑥ At the end of ninth clock, if the host has sent an $\overline{ACK}$, module clears SCLREL to suspend clock. Client generates interrupt.

⑦ At the end of ninth clock, if host sent a NACK, no more data expected. User software should stop writing to I2CxTRN. Module does not suspend clock and will generate an interrupt.

Figure 20-37. Client Message (Read Data from Client: 10-Bit Address)



① Client recognizes Start event; S and P bits set/clear accordingly.

② Client receives first address byte. Write indicated. Client Acknowledges and generates interrupt. User software reads I2CxRCV register.

③ Client receives address byte. Address matches. Client Acknowledges and generates interrupt. User software reads I2CxRCV register.

④ Host sends a Repeated Start to redirect the message.

⑤ Client receives resend of first address byte. User software reads I2CxRCV register. Read indicated. Client suspends clock.

⑥ User software writes I2CxTRN with response data.

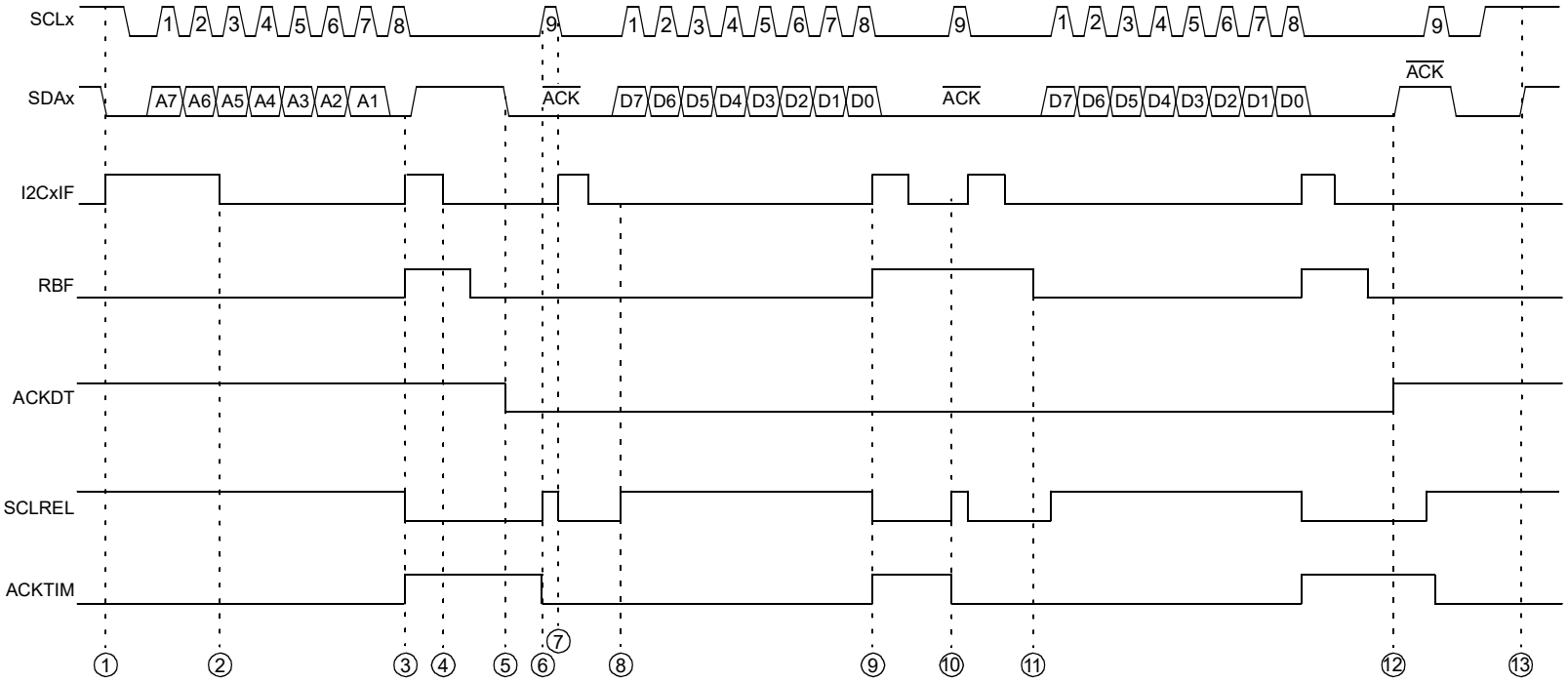
⑦ User software sets SCLREL to release clock hold. Host resumes clocking and client transmits data byte.

⑧ At the end of ninth clock, if host sent an $\overline{\text{ACK}}$, module clears SCLREL to suspend clock. Client generates interrupt.

⑨ At the end of ninth clock, if host sent a NACK, no more data expected. User software should stop writing to I2CxTRN. Module does not suspend clock and will generate an interrupt.

⑩ Client recognizes Stop event; S and P bits set/clear accordingly.

Figure 20-38. I²C Client, 7-Bit Address, Reception (STREN = 1, AHEN = 1, DHEN = 1)



① Detecting Start bit, enables address detection, interrupt is set if SCEN is set.

② User software clears the interrupt flag.

③ Client receives first address byte. Write Indicated. If AHEN = 1, SCLREL is cleared by hardware. ACKTIM and interrupt are asserted.

④ User software clears the interrupt flag.

⑤ ACKDT is written with an $\overline{\text{ACK}}$ by user software.

⑥ User software sets SCLREL to release the clock hold; ACKTIM is cleared by hardware.

⑦ Hardware stretches the clock after $\overline{\text{ACK}}$ (if STREN = 1).

⑧ User software sets SCLREL to release clock hold.

⑨ I2CxRCV is loaded with I2CxRSR. RBF is set. If DHEN = 1, SCLREL is cleared by hardware and ACKTIM is asserted.

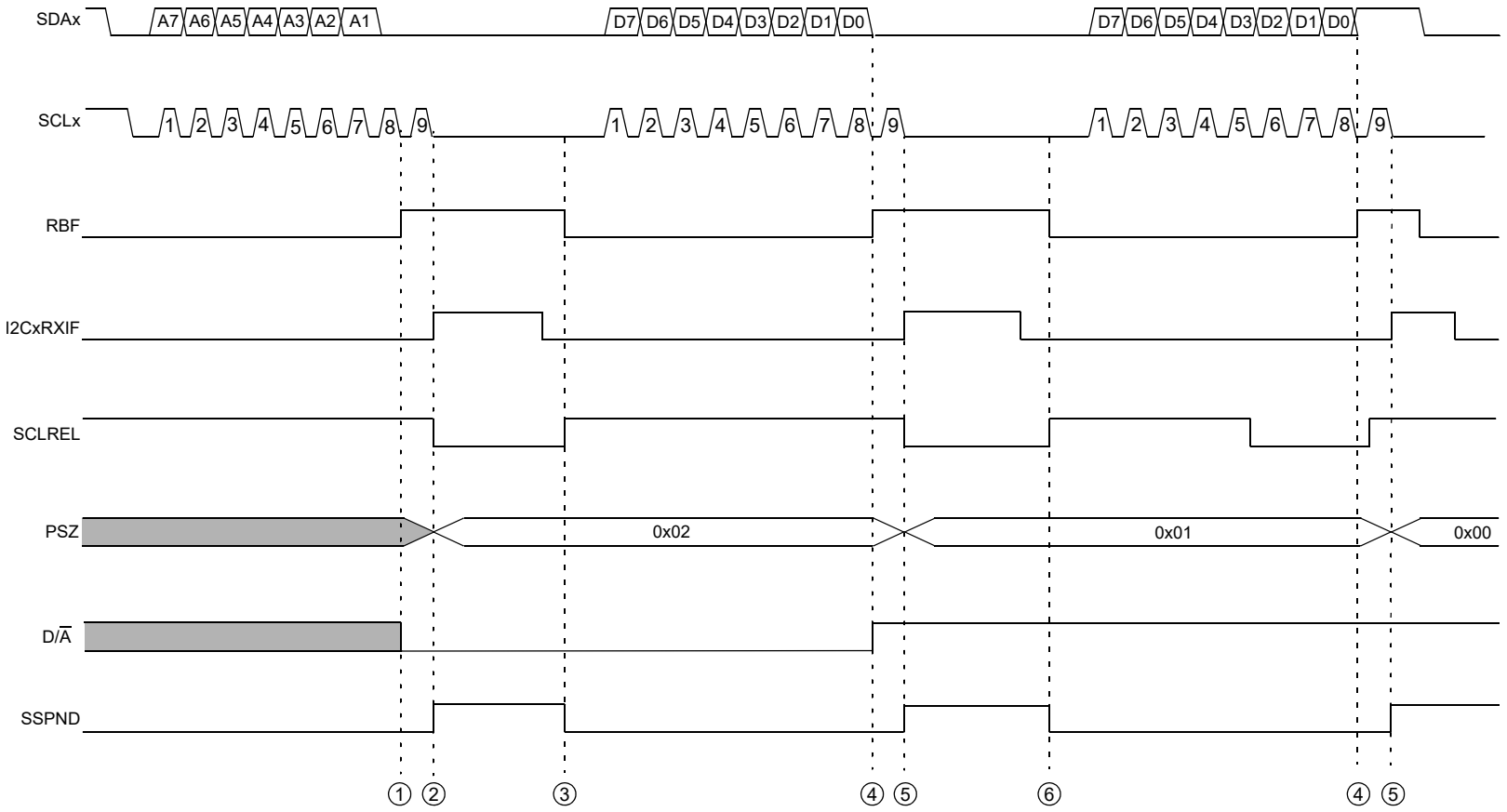
⑩ User software sets SCLREL to release the clock hold. ACKTIM is cleared by hardware. After Acknowledgment, hardware stretches the clock.

⑪ User software reads I2CxRCV, that clears the RBF flag.

⑫ NACK sent by host.

⑬ Module recognizes the Stop event.

Figure 20-39. I²C Client 7-Bit Address Reception with Smart Mode Enabled (SMEN = 1)



1. Client receives address byte, RBF= 1.
2. RBF = 1 at ninth clock, clock will be suspended SSPND = 1. Automatic clock stretch begins. Client clears SCLREL bit. Client pulls SCLx line low to stretch clock.
3. User software reads I2CxRCV register. RBF bit clears. Client will stop suspending clock, SSPND= 0. Client sets SCLREL bit to release the clock.
4. Client receives data byte, RBF= 1.
5. RBF = 1 at ninth clock, clock will be suspended SSPND = 1. Automatic clock stretch begins. Client clears SCLREL bit. Client pulls SCLx line low to stretch clock.
6. User software reads I2CxRCV register. RBF bit clears. Client will stop suspending clock, SSPND= 0. Client sets SCLREL bit to release the clock.

20.5.4.6 Frame Error

Frame error is used in the PMBus to detect fault. Frame error will detect:

- Sending Too Few Bits : If a Start or Stop condition interrupts the transmission while a device is writing to a PMBus device before a complete byte has been sent, this is a data transmission fault.
- Reading Too Few Bits : If a Start or Stop condition interrupts the transmission while a device is reading from a PMBus device before a complete byte has been read, this is a data reception fault. In these conditions the FRME(I2CxSTAT2[19]) bit will be set.
- When the frame error occurs due to a Stop condition, the receiver returns to an Idle state and waits for the Start condition.
- When the frame error occurs due to a Start condition, the client is considered as a new Start condition and moves to the address receive start sequence.
- On frame error, the number of SCL clocks will be stored in SCLCNT(I2CxSTAT2[3:0]).

20.5.5 Packet Error Checking (PEC)

The Packet Error Checking mechanism improves reliability and communication robustness. CRC-8 packet error checking based on SMBus v3.0 specification is available. PEC is calculated on all the message bytes (including addresses and read/write bits). The PEC calculation does not include $\overline{\text{ACK}}$/NACK bits, Start/Stop or Repeated Start conditions. PEC is available in both Host and Client mode.

During client/host transmission, PEC can be enabled or disabled to automatically append at the end of packet using PECC (I2CxCON2[29:28]) bits.

During reception, when PECC[1:0] is `01`, the calculated PEC value will be written into I2CxPEC.CCRC at the end of the packet. Since the received packet will contain a PEC byte at the end of the packet, the calculated PEC should be `0` if the packet was received correctly. Any mismatch in calculated PEC and received PEC, CRC(I2CxSTAT[16]) will be set.

20.5.6 Bus Timeout

Different bus timeout options are available, which are required for SMBus and PMBus protocols. There are four timeouts available: SCL low timeout, host extended timeout, client extended timeout and Bus Free/Idle timeout.

Note: The timers values should be written by software before enabling the timers.

20.5.6.1 SCL Low Timeout ($t_{TIMEOUT}$)

SCL low timeout ($t_{TIMEOUT}$) allows a host or client to conclude that a defective device is holding the clock low indefinitely or that a host is intentionally trying to drive devices off the bus. It is highly recommended that a client device release the bus when it detects any single clock held low longer than $t_{TIMEOUT, MIN}$. Devices that have detected this condition must reset their communication interface and be able to receive a new Start condition in no later than $t_{TIMEOUT, MAX}$.

SCL low timeout can be enabled by writing the timeout value to BSCLTOTMR (I2CxBSCLTO[23:0]) and then setting the BSCLTE (I2CxCON2[27]) bit. The timer will continue to run until SCL stays low or the timer value reloads. If the timer reaches `0` before reloading, then the timeout BSCLTO (I2CxSTAT2[23]) flag and ERR (I2CxSTAT2[11]) are set. The BSCLTIE (I2CxINTC[23]) bit can be enabled to generate an error interrupt (I2CxEIF).

When timeout occurs in Client mode:

- SCL will be released (SCLREL=`1`), irrespective of STREN, by hardware automatically
- SSPND (I2CxSTAT2[31]) will be cleared
- SDA will be released after SCL based on the SDASU timer value specification
- CLTACT (I2CxSTAT2[30]) will be cleared

When timeout occurs in Host mode:

- SSPND (I2CxSTAT2[31]) will be cleared
- Hardware automatically sends Stop bit to terminate the current transaction

Note: SCL low timeout can wake up the device from sleep in the client address phase.

20.5.6.2 Client Extended Timeout ($t_{LOW:SEXT:}$)

Client extended timeout allows a host or client to extend its clock cycles measured from the initial Start to the Stop. Client extended timeout can be enabled by the writing timeout value to CBCTOTMR (I2CXCBCTO[23:0]) and then enabling the CBCTE (I2CxCON2[25]) bit. The timer continues to run until it detects Start or Stop and then the timer values reload. If the timer reaches 0 before reloading, then the timeout CBCLTO (I2CxSTAT2[21]) flag and ERR (I2CxSTAT2[11]) are set. The CBCTIE (I2CxINTC[21]) bit can be enabled to generate an error interrupt (I2CxEIF).

20.5.6.3 Host Extended Timeout ($t_{LOW:MEXT:}$)

Host extended timeout is used for clock extension within one byte in a message as measured from:

START to $\overline{ACK}$

$\overline{ACK}$ to $\overline{ACK}$

$\overline{ACK}$ to STOP

Host extended timeout can be enabled by writing the timeout value to HBCTOTMR (I2CXHBCTO[23:0]) and then enabling the HBCTE (I2CxCON2[26]) bit. The timer continues to run until it detects Start or $\overline{ACK}$ or Stop and then the timer values reload. If the timer reaches 0 before reloading, then the timeout HBCLTO (I2CxSTAT2[22]) flag and ERR (I2CxSTAT2[11]) are set. The HBCTIE (I2CxINTC[22]) bit can be enabled to generate an error interrupt (I2CxEIF).

20.5.6.4 Bus IDLE Timeout

Bus Idle timer can be used to define the bus free time (T_{BUF}) between Stop and Start. Bus free time, depending on the baud rate, should be loaded to BITOPR (I2CXBITO[23:0]) and then the bus idle timer should be enabled by setting BITE (I2CxCON2[16]). The bus Idle timeout starts when the host software initiates a Start condition. BITO (I2CxSTAT2[20]) will be set after the defined bus Idle time. BITIE (I2CxINTC[20]) bit can be enabled to generate a generic interrupt (I2CxIF).

20.5.7 DMA Operation

The DMA can be used to transfer the data without the participation of the CPU. The I2CxTXIF and I2CxRXIF interrupts can be used to trigger DMA.

Note:

1. Packet size (PSZ) should be configured before data transmit/receive.
2. ND/$\overline{A}$ should be configured accordingly to indicate address/data transmission.
3. It is recommended to use Smart mode when using DMA to reduce CPU interaction.

20.5.7.1 Client Mode Transmission with DMA

Once the client is addressed by the host, ND/$\overline{A}$ (I2CxCON2[18]) should be set and then the first data should be loaded by the software. Client hardware automatically clears SCLREL and generates the transmit interrupt (I2CxTXIF) after the $\overline{ACK}$ if the error transmission (I2CEF(I2CxSTAT2[11]) is not set and has not reached the packet size = 0. The transmit buffer status bit TBF(I2CxSTAT1[0]) is clear when the I2CxTRN does not contain any transmit data. At this point, the transmit interrupt (I2CxTXIF) can trigger DMA to load another byte into the buffer and the transmit buffer full (TBF) is set. Client hardware sets SCLREL based on data setup time:

- The SDASUT(I2CxSDASUT[15:0]) value, when the TBF(I2CxSTAT1[0]) is set and Smart mode is enabled (SMEN I2CxCON2[17])

20.5.7.2 Client Mode Reception with DMA

Once the client is addressed by the host, ND/$\overline{A}$ (I2CxCON2[18]) is set in the software. On receiving data from the host, the RBF (I2CxSTAT1[1]) bit will be set, and if the I2COV (I2CxSTAT1[6]) bit is

cleared, the receive interrupt (I2CxRXIF) triggers the DMA to read the I2CRCV register into the data memory.

Note:

1. In Smart mode with DMA, it is recommended to configure DHEN = 0 and STREN = 1 to get a single interrupt.

20.5.7.3 Host Mode Transmission with DMA

Once the host has addressed the client, ND/$\overline{A}$ (I2CxCON2[18]) has to be set to the indicated data transmission and then load the first data to the transmit buffer. The Transmit Buffer Status bit TBF (I2CxSTAT1[0]) is cleared when the I2CxTRN does not contain any transmit data. At this point, the transmit interrupt (I2CxTXIF) can trigger DMA to load another byte into the buffer. The host hardware sets the TBF bit, and the DMA continues to write the next data until the packet size (PSZ) becomes 0.

20.5.7.4 Host Mode Reception with DMA

Once the host has addressed the client, on receiving data from the client RBF (I2CxSTAT1[1]), the bit will be set and the receive interrupt (I2CxRXIF) triggers the DMA to read the I2CRCV register into the data memory.

Notes:

1. In Smart mode, receive enable (RCEN) will be set by the hardware automatically based on receive buffer status and packet size. For the first data byte, user software should enable receive by setting RCEN (I2CxCON1[3]).
2. To reduce CPU interaction when using DMA, it is recommended to use $\overline{ACK}$ Control bits (I2CxCON2.ACKC) to automatically send an acknowledgment.

20.5.8 Connection Considerations For I²C Bus

Because the I²C bus is a wired-AND bus connection, Pull-up Resistors (R_P) on the bus are required, as illustrated in Figure 20-40. The Series Resistors (R_S) are optional and are used to improve the Electrostatic Discharge (ESD) susceptibility.

The values of the resistors, R_P and R_S, depend on the following parameters:

- Supply voltage
- Bus capacitance
- Number of connected devices (input current + leakage current)
- Input level selection (I²C or System Management Bus (SMBus))

Because the device must be able to pull the bus low against R_P, current drawn by R_P must be greater than the I/O pin minimum sink current, I_{OL} at V_{OLMAX}, for the device output stage. Equation 20-2 shows the formula for computing the minimum pull-up resistance.

Equation 20-2. Minimum Pull-up Resistance

$$R_{PMIN} = \frac{(V_{DDMAX} - V_{OLMAX})}{I_{OL}}$$

In a 400 kHz system, a minimum rise time specification of 300 ns exists; in a 100 kHz system, the specification is 1000 ns. Because R_P must pull the bus up against the total Capacitance, C_B, with a maximum rise time of 300 ns to (V_{DD} – 0.7V), the Maximum Resistance for the Pull-up (R_{PMAX}) is computed using the formula as shown in Equation 20-3.

Equation 20-3. Maximum Pull-up Resistance

$$\frac{-tR}{C_B * [ln(1 - (V_{DDMAX} - V_{ILMAX}))]}$$

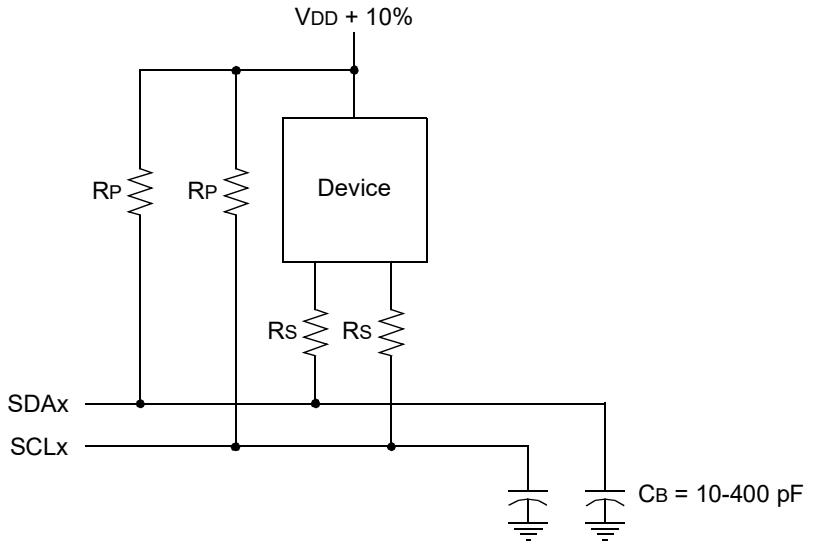
The maximum value for R_S is determined by the desired noise margin for the low level. Rs cannot drop enough voltage to make the device V_{OL} and the voltage across R_S more than the maximum V_{IL}. Equation 20-4 shows the formula for computing the maximum value for R_S.

Equation 20-4. Maximum Series Resistance

$$RSMAX = \frac{(VILMAX - VOLMAX)}{IOLMAX}$$

Note: The SCLx clock input must have a minimum high and low time for proper operation. Refer to the **"Electrical Characteristics"** chapter for more information on the high and low times of the I²C bus specification and requirements of the I²C module and I/O pins.

Figure 20-40. Sample Device Configuration for I²C Bus



20.5.8.1 Integrated Signal Conditioning

The SCLx and SDAx pins have an input glitch filter. The I²C bus requires this filter in both the 100 kHz and 400 kHz systems.

When operating on a 400 kHz bus, the I²C bus specification requires a slew rate control of the device pin output. This slew rate control is integrated into the device. If the DISSLW bit (I2CxCON1[9]) is cleared, the slew rate control is active. For other bus speeds, the I²C bus specification does not require slew rate control and the DISSLW bit should be set.

Some system implementations of I²C buses require different input levels for V_{ILMAX} and V_{IHMIN}. In a normal I²C system, V_{ILMAX} is 0.3 V_{DD} and V_{IHMIN} is 0.7 V_{DD}. By contrast, in a SMBus system, V_{ILMAX} is set at 0.8V while V_{IHMIN} is set at 2.1V.

The SMBEN bit (I2CxCON1[25:24]) controls the input levels. Setting SMBEN (= 1) changes the input levels to SMBus specifications.

20.5.9 SMBus Support

The dsPIC33AK128MC106 family devices have support for SMBus through options in the input voltage thresholds.

Table 20-7. I²C Pin Voltage Threshold

	SMEN SFR Bit (I2CxCONL[8])
I²C (default)	0

.....continued

	SMEN SFR Bit (I2CxCONL[8])
SMBus 2.0	1
SMBus 3.0	1

20.5.10 Peripheral Module Disable (PMDx) Registers

The Peripheral Module Disable (PMDx) registers provide a method to disable the I^{2}C modules by stopping all clock sources supplied to that module. When a peripheral is disabled through the appropriate PMDx control bit, the peripheral is in a minimum power consumption state. The control and status registers associated with the peripheral are also disabled, so writes to those registers will have no effect and read values will be invalid. A peripheral module is only enabled if the I2CxMD bit in the PMDx register is cleared.

20.5.11 Effects of a Reset

A Reset disables the I^{2}C module and terminates any active or pending message activity. Refer to the I2Cx Control (I2CxCON1 or I2CxCON2) and I2Cx Status (I2CxSTAT) registers' definitions for the Reset conditions of those registers.

Note: In this discussion, 'Idle' refers to the CPU power-saving state. The lower case 'idle' refers to the time when the I^{2}C module is not transferring data on the bus.

20.6 Application Examples

20.6.1 Host Transmission (7-bit Address)

Example 20-1. Host Transmission (7-bit Address)

```
#include <xc.h>


#define mCLIENT_ADDRESS 0x4C

#define INIT              0
#define ADDRESS_PHASE     1
#define DATA_WRITE        2


unsigned char data[10], phase;
unsigned char count = 0;

int main(void) {
    /*Configure I2C pins as digital*/


    /*I2C1 configured as Host */

    I2C1LBRG = 38; // 100kHz @ 8MIPS
    I2C1HBRG = 38; // 100kHz @ 8 MIPS

    /* Configure Bus IDle timeout*/

    I2C1CON2bits.BITE = 1;
    I2C1BITObits.BITOPR = 76;
    /* Configured interrupt enable bits*/
    I2C1INTCbits.HACKSIE = 1; // Assert HSTIF on ACK seq
    I2C1INTCbits.HDTXIE = 1; // Assert HSTIF on TX
    I2C1INTCbits.HSCIE = 1; // Assert HSTIF on start
    I2C1INTCbits.HSTIE = 1; // Assert I2CxIF when HSTIF is set

    I2C1CON2bits.PSZ = 10; // Packet size
    I2C1CON2bits.EOPSC = 0b01; // End of packet will be set after data bytes
    I2C1CON1bits.ON = 1; // Enable I2C

    IFS2bits.I2C1IF = 0; // Clear I2C general interrupt
    IEC2bits.I2C1IE = 1; // Enable I2C general interrupt
```

```
    /*wait for bus idle */
    while (!I2C1STAT2bits.BITO);

    phase = ADDRESS_PHASE;
    I2C1CON1bits.SEN = 1; // Send Start bit

    while (1) {

    }
}

void __attribute__((interrupt, no_auto_psv)) _I2C1Interrupt(void) {
    IFS2bits.I2C1IF = 0;

    switch (phase) {
        case ADDRESS_PHASE:
            /*Verify if start has been sent*/
            if (I2C1STAT2bits.STARTE) {
                /* Transmit client address with RW =0 , writing to client*/
                I2C1TRN = (mCLIENT_ADDRESS << 1) | 0;
                phase = DATA_WRITE;
            }
            break;
        case DATA_WRITE:
            /* Set NDA to indicate next byte is data */
            if (I2C1CON2bits.NDA != 1) {
                I2C1CON2bits.NDA = 1;
            }
            /* If Packet size is 0, EOP will be asserted,send STOP on EOP*/
            if (I2C1STAT2bits.EOP) {
                I2C1CON1bits.PEN = 1;

            } else {
                if ((I2C1STAT1bits.ACKSTAT == 0) && (!I2C1STAT1bits.TBF) && (!
I2C1STAT1bits.TRSTAT)) {
                    I2C1TRN = count++; // Send data to client
                }
            }
            break;
    }
}
```

20.6.2 Host Transmission (10-bit Address)

Example 20-2. Host Transmission (10-bit Address)

```
#include <xc.h>


#define INIT                            0
#define ADDRESS_PHASE_UPPER_10BIT       1
#define ADDRESS_PHASE_LOWER_10BIT       2
#define DATA_WRITE                      3


unsigned char count=0;

#define mCLIENT_ADDRESS 0x14C


unsigned char data[10],phase;

int main(void)
{
/*Configure I2C pins as digital*/



    /*I2C1 configured as Host */

    I2C1LBRG = 38;                  // 100kHz @ 8MIPS
    I2C1HBRG = 38;                  // 100kHz @ 8 MIPS
```

```
/* Configure Bus IDle timeout*/

    I2C1CON2bits.BITE = 1;
    I2C1BITObits.BITOPR = 76;
    /* Configured interrupt enable bits*/
    I2C1INTCbits.HACKSIE = 1; // Assert HSTIF on ACK seq
    I2C1INTCbits.HDTXIE = 1; // Assert HSTIF on TX
    I2C1INTCbits.HSCIE = 1; // Assert HSTIF on start
    I2C1INTCbits.HSTIE = 1; // Assert I2CxIF when HSTIF is set

    I2C1CON2bits.PSZ = 10; // Packet size
    I2C1CON2bits.EOPSC = 0b01; // End of packet will be set after data bytes
    I2C1CON1bits.ON = 1; // Enable I2C

    IFS2bits.I2C1IF = 0; // Clear I2C general interrupt
    IEC2bits.I2C1IE = 1; // Enable I2C general interrupt

    /*wait for bus idle */
    while(!I2C1STAT2bits.BITO);

     phase =  ADDRESS_PHASE_UPPER_10BIT;
    I2C1CON1bits.SEN= 1;         // Send Start bit


    while(1);
}



void __attribute__((interrupt, no_auto_psv)) _I2C1Interrupt(void)
{
    IFS2bits.I2C1IF = 0;

    switch(phase)
    {

        case ADDRESS_PHASE_UPPER_10BIT:
         /*Verify if start has been sent*/
            if(I2C1STAT2bits.STARTE)
            {
                /* Transmit client address with RW =0 , writing to client*/
                I2C1TRN = ((((mCLIENT_ADDRESS >> 8) & 0x03)<< 1) | 0 ) +
0b11110000  ;
                phase = ADDRESS_PHASE_LOWER_10BIT;
            }
            break;
        case ADDRESS_PHASE_LOWER_10BIT:
        /*Verify if the upper 10bit address is ACKed*/
            if (I2C1STAT1bits.ACKSTAT ==0)
            {
                I2C1TRN = mCLIENT_ADDRESS & 0xFF  ;
                phase = DATA_WRITE;
            }
            break;
        case DATA_WRITE:
            /* Set NDA to indicate next byte is data
*/
            if(I2C1CON2bits.NDA !=1)
            {
                I2C1CON2bits.NDA = 1;
            }
            /* If Packet size is 0, EOP will be asserted,send STOP on
EOP*/
            if(I2C1STAT2bits.EOP)
            {
                I2C1CON1bits.PEN =1;
            }
            else
            {
                if ((I2C1STAT1bits.ACKSTAT ==0) && (!I2C1STAT1bits.TBF) && (!
I2C1STAT1bits.TRSTAT))
                {
                   I2C1TRN = count++;            // Send data to client
                }
            }
            break;
```

```
    }
}
```

20.6.3 Host Reception (7-bit Address)

Example 20-3. Host Reception (7-bit Address)

```
#include <xc.h>


#define mCLIENT_ADDRESS 0x4C

#define INIT              1
#define ADDRESS_PHASE     2
#define DATA_READ         3

unsigned int hostReceived[20], phase;
unsigned char count = 0;

int main(void) {

/*Configure I2C pins as digital*/



    /*I2C1 configured as Host */

    I2C1LBRG = 38;                      // 100kHz @ 8MIPS
    I2C1HBRG = 38;                      // 100kHz @ 8 MIPS

/* Configure Bus IDle timeout*/
    I2C1CON2bits.BITE = 1;
    I2C1BITObits.BITOPR = 76;
/* Configured interrupt enable bits*/
    I2C1INTCbits.HACKSIE = 1;           // Assert HSTIF on ACK seq
    I2C1INTCbits.HDTXIE = 1;            // Assert HSTIF on TX
    I2C1INTCbits.HDRXIE = 1;            // Assert HSTIF on RX
    I2C1INTCbits.HSCIE = 1;             // Assert HSTIF on start
    I2C1INTCbits.HSTIE = 1;             // Assert I2CxIF when HSTIF is set


    I2C1CON2bits.SMEN = 1;              // Smart mode enabled
    I2C1CON2bits.ACKC = 0b10;           // ACK all the bytes expect last byte
    I2C1CON2bits.PSZ = 10;              // Packet size
    I2C1CON2bits.EOPSC = 0b01;          // End of packet will be set after
data bytes
    I2C1CON1bits.ON = 1;                // Enable I2C

    IFS2bits.I2C1IF = 0;                // Clear I2C general interrupt
    IEC2bits.I2C1IE = 1;                // Enable I2C general interrupt


    /*wait for bus idle */
    while (!I2C1STAT2bits.BITO);

    phase = ADDRESS_PHASE;
    I2C1CON1bits.SEN = 1; // Send Start bit

    while (1) {

    }
}


void __attribute__((interrupt, no_auto_psv)) _I2C1Interrupt(void) {
    IFS2bits.I2C1IF = 0;

    switch (phase) {
        case ADDRESS_PHASE:
            /*Verify if start has been sent*/
            if (I2C1STAT2bits.STARTE) {
                phase = DATA_READ;
                /* Transmit client address with RW =1 , read  client*/
                I2C1TRN = (mCLIENT_ADDRESS << 1) | 1;
```

```
            }
            break;
        case DATA_READ:
            /* Set NDA to indicate next byte is data */
            if (I2C1CON2bits.NDA != 1) {
                I2C1CON2bits.NDA = 1;
                /*Enable receive for the 1st time*/
                I2C1CON1bits.RCEN = 1;
            }
/*Read the received data*/
            if (I2C1STAT1bits.RBF) {
                hostReceived[count++] = I2C1RCV;
            }
            /* If Packet size is 0, EOP will be asserted,send STOP on EOP*/
            if (I2C1STAT2bits.EOP == 1) {
                I2C1CON1bits.PEN = 1;
            }

            break;
    }

}
```

20.6.4 Host Reception (10-bit Address)

Example 20-4. Host Reception (10-bit Address)

```
#include <xc.h>

#define INIT                                0
#define ADDRESS_PHASE_UPPER_10BIT_WRITE     1
#define ADDRESS_PHASE_LOWER_10BIT           2
#define RESTART                             3
#define ADDRESS_PHASE_UPPER_10BIT_READ      4
#define DATA_READ                           5


unsigned char hostReceived[20], phase;
unsigned char count = 0;
#define mCLIENT_ADDRESS 0x14C

int main(void) {

    /*Configure I2C pins as digital*/


    /*I2C1 configured as Host */

    I2C1LBRG = 38; // 100kHz @ 8MIPS
    I2C1HBRG = 38; // 100kHz @ 8 MIPS


    /* Configure Bus IDle timeout*/
    I2C1CON2bits.BITE = 1;
    I2C1BITObits.BITOPR = 76;
    /* Configured interrupt enable bits*/

    I2C1INTCbits.HACKSIE = 1; // Assert HSTIF on ACK seq
    I2C1INTCbits.HDTXIE = 1; // Assert HSTIF on TX
    I2C1INTCbits.HDRXIE = 1; // Assert HSTIF on RX
    I2C1INTCbits.HSCIE = 1; // Assert HSTIF on start
    I2C1INTCbits.HSTIE = 1; // Assert I2CxIF when HSTIF is set


    I2C1CON2bits.SMEN = 1; // Smart mode enabled
    I2C1CON2bits.ACKC = 0b10; // ACK all the bytes expect last byte
    I2C1CON2bits.PSZ = 10; // Packet size
    I2C1CON2bits.EOPSC = 0b01; // End of packet will be set after data bytes
    I2C1CON1bits.ON = 1; // Enable I2C

    IFS2bits.I2C1IF = 0; // Clear I2C general interrupt
    IEC2bits.I2C1IE = 1; // Enable I2C general interrupt
```

```
    /*wait for bus idle */
    while (!I2C1STAT2bits.BITO);

    phase = ADDRESS_PHASE_UPPER_10BIT_WRITE;
    I2C1CON1bits.SEN = 1; // Send Start bit




    while (1);
}

void __attribute__((interrupt, no_auto_psv)) _I2C1Interrupt(void) {
    IFS2bits.I2C1IF = 0;

    switch (phase) {

        case ADDRESS_PHASE_UPPER_10BIT_WRITE:
            /*Verify if start has been sent*/
            if (I2C1STAT2bits.STARTE) {
                /* Transmit client address with RW =0 , write  client*/
                I2C1TRN = ((((mCLIENT_ADDRESS >> 8) & 0x03) << 1) | 0) +
0b11110000;
                phase = ADDRESS_PHASE_LOWER_10BIT;
            }
            break;
        case ADDRESS_PHASE_LOWER_10BIT:
            /*Verify if the upper 10bit address is ACKed*/
            if (I2C1STAT1bits.ACKSTAT == 0) {
                I2C1TRN = mCLIENT_ADDRESS & 0xFF;
                phase = RESTART;
            }
            break;
        case RESTART:
            I2C1CON1bits.RSEN = 1; // Send restart
            phase = ADDRESS_PHASE_UPPER_10BIT_READ;
            break;
        case ADDRESS_PHASE_UPPER_10BIT_READ:
            /* Transmit client address with RW =1 , read  client*/
            I2C1TRN = ((((mCLIENT_ADDRESS >> 8) & 0x03) << 1) | 1) +
0b11110000;
            phase = DATA_READ;
            break;
        case DATA_READ:
            /* Set NDA to indicate next byte is data */
            if (I2C1CON2bits.NDA != 1) {
                I2C1CON2bits.NDA = 1;
                /*Enable receive for the 1st time*/
                I2C1CON1bits.RCEN = 1;
            }
            /*Read the received data*/
            if (I2C1STAT1bits.RBF) {
                hostReceived[count++] = I2C1RCV;
            }
            /* If Packet size is 0, EOP will be asserted,send STOP on EOP*/
            if (I2C1STAT2bits.EOP) {
                I2C1CON1bits.PEN = 1;
            }

            break;
    }
}
```

20.6.5 Client Reception (7-bit Address)

Example 20-5. Client Reception (7-bit Address)

```
#include <xc.h>


#define mCLIENT_ADDRESS 0x4C
```

```
#define INIT              0
#define ADDRESS_PHASE     1
#define DATA_WRITE        2


volatile unsigned char gReceived[20], gRcv_count,count = 0;

int main(void) {

    /*I2C1 configured as client*/


    I2C1ADD = mCLIENT_ADDRESS; // Configure Client address
    I2C1INTCbits.CADDRIE = 1; // Assert CLTIF on address detect
    I2C1INTCbits.CDRXIE = 1; // Assert CLTIF on received byte
    I2C1INTCbits.CLTIE = 1; // Assert I2CxIF when CLTIF is asserted


    I2C1CON2bits.SMEN = 1; // Enable smart mode
    I2C1CON2bits.PSZ = 10; // Packet size

    I2C1CON1bits.ON = 1; // Enable I2C


    IFS2bits.I2C1IF = 0; // Clear I2C general interrupt
    IEC2bits.I2C1IE = 1; // Enable I2C general interrupt

    while (1) {

    }
}

void __attribute__((__interrupt__, no_auto_psv)) _I2C1Interrupt(void) {

    IFS2bits.I2C1IF = 0;
    /* Read received address*/
    if ((I2C1STAT1bits.RBF)&& (!I2C1STAT1bits.D_A)) {
        gRcv_count = 0;
        gReceived[gRcv_count] = I2C1RCV;
        gRcv_count++;

    }        /* Read received data byte*/
    else if ((I2C1STAT1bits.RBF)&& (I2C1STAT1bits.D_A)) {
        gReceived[gRcv_count] = I2C1RCV;
        gRcv_count++; // Read received data

    }
}
```

20.6.6 Client Reception (10-bit Address)

Example 20-6. Client Reception (10-bit Address)

```
#include <xc.h>

#define INIT                            0
#define ADDRESS_PHASE_UPPER_10BIT       1
#define ADDRESS_PHASE_LOWER_10BIT       2
#define DATA_WRITE                      3
volatile unsigned char gReceived[20], gRcv_count, count = 0;

#define mCLIENT_ADDRESS 0x14C

int main(void) {

     /*Configure I2C pins as digital*/

    /*I2C1 configured as client*/


    I2C1ADD = mCLIENT_ADDRESS; // Configure Client address
    I2C1INTCbits.CADDRIE = 1; // Assert CLTIF on address detect
    I2C1INTCbits.CDRXIE = 1; // Assert CLTIF on received byte
```

```
    I2C1INTCbits.CLTIE = 1; // Assert I2CxIF when CLTIF is asserted



    I2C1CON2bits.SMEN = 1; // Enable smart mode
    I2C1CON2bits.PSZ = 10; // Packet size


    I2C1CON1bits.A10M = 1; // 10it addressing
    I2C1CON1bits.ON = 1; // Enable I2C


    IFS2bits.I2C1IF = 0; // Clear I2C general interrupt
    IEC2bits.I2C1IE = 1; // Enable I2C general interrupt

    while (1);
}

void __attribute__((__interrupt__, no_auto_psv)) _I2C1Interrupt(void) {

    IFS2bits.I2C1IF = 0;
    /* Read received address*/
    if ((I2C1STAT1bits.RBF)&& (!I2C1STAT1bits.D_A)) {
        gRcv_count = 0;
        gReceived[gRcv_count] = I2C1RCV;
        gRcv_count++;

    }/* Read received data byte*/
    else if ((I2C1STAT1bits.RBF)&& (I2C1STAT1bits.D_A)) {
        gReceived[gRcv_count] = I2C1RCV;
        gRcv_count++; // Read received data

    }

}
```

20.6.7 Client Transmission (7-bit Address)

Example 20-7. Client Transmission (7-bit Address)

```
#include <xc.h>


#define mCLIENT_ADDRESS 0x4C

#define INIT              1
#define ADDRESS_PHASE     2
#define DATA_READ         3

volatile unsigned char gReceived[20], gRcv_count, gTrans_count, count = 0;

int main(void) {
    /*Configure I2C pins as digital*/

    /*I2C1 configured as client*/

    I2C1ADD = mCLIENT_ADDRESS; // Configure Client address
    I2C1INTCbits.CADDRIE = 1; // Assert CLTIF on address detect
    I2C1INTCbits.CDRXIE = 1; // Assert CLTIF on received byte
    I2C1INTCbits.CDTXIE = 1; // Assert CLTIF on transmit byte
    I2C1INTCbits.CLTIE = 1; // Assert I2CxIF when CLTIF is asserted

    I2C1CON2bits.SMEN = 1; // Enable smart mode
    I2C1CON2bits.PSZ = 10; // Packet size


    I2C1CON1bits.ON = 1; // Enable I2C


    IFS2bits.I2C1IF = 0; // Clear I2C general interrupt
    IEC2bits.I2C1IE = 1; // Enable I2C general
interrup

    while (1) {
```

```
    }
}

void __attribute__((__interrupt__, no_auto_psv)) _I2C1Interrupt(void) {

    IFS2bits.I2C1IF = 0;
    /* Read received address*/
    if ((I2C1STAT1bits.RBF)&& (!I2C1STAT1bits.D_A)) {
        gTrans_count = 0;
        gRcv_count = 0;
        gReceived[gRcv_count] = I2C1RCV;
        gRcv_count++;

    }/* Read received data byte*/
    else if ((I2C1STAT1bits.RBF)&& (I2C1STAT1bits.D_A)) {
        gReceived[gRcv_count] = I2C1RCV;
        gRcv_count++; // Read received data

    }
    /* transmit byte if R_W is set */
    if ((I2C1STAT1bits.R_W) &&(!I2C1STAT1bits.ACKSTAT)) {
        I2C1TRN = gTrans_count++; // transmit data

    }
}

s
```

20.6.8 Client Transmission (10-bit Address)

Example 20-8. Client Transmission (10-bit Address)

```
#include <xc.h>

#define INIT                                0
#define ADDRESS_PHASE_UPPER_10BIT_WRITE     1
#define ADDRESS_PHASE_LOWER_10BIT           2
#define RESTART                             3
#define ADDRESS_PHASE_UPPER_10BIT_READ      4
#define DATA_READ                           5

#define mCLIENT_ADDRESS 0x14C

volatile unsigned char gReceived[256], gRcv_count, gTrans_count, count = 0;


int main(void) {
    /*Configure I2C pins as digital*/




    /*I2C1 configured as client*/

    I2C1ADD = mCLIENT_ADDRESS; // Configure Client address
    I2C1INTCbits.CADDRIE = 1; // Assert CLTIF on address detect
    I2C1INTCbits.CDRXIE = 1; // Assert CLTIF on received byte
    I2C1INTCbits.CDTXIE = 1; // Assert CLTIF on transmit byte
    I2C1INTCbits.CLTIE = 1; // Assert I2CxIF when CLTIF is asserted

    I2C1CON2bits.SMEN = 1; // Enable smart mode
    I2C1CON2bits.PSZ = 10; // Packet size

    I2C1CON1bits.A10M = 1; // 10bit addressing
    I2C1CON1bits.ON = 1; // Enable I2C



    while (1);
}

void __attribute__((__interrupt__, no_auto_psv)) _I2C1Interrupt(void) {
```

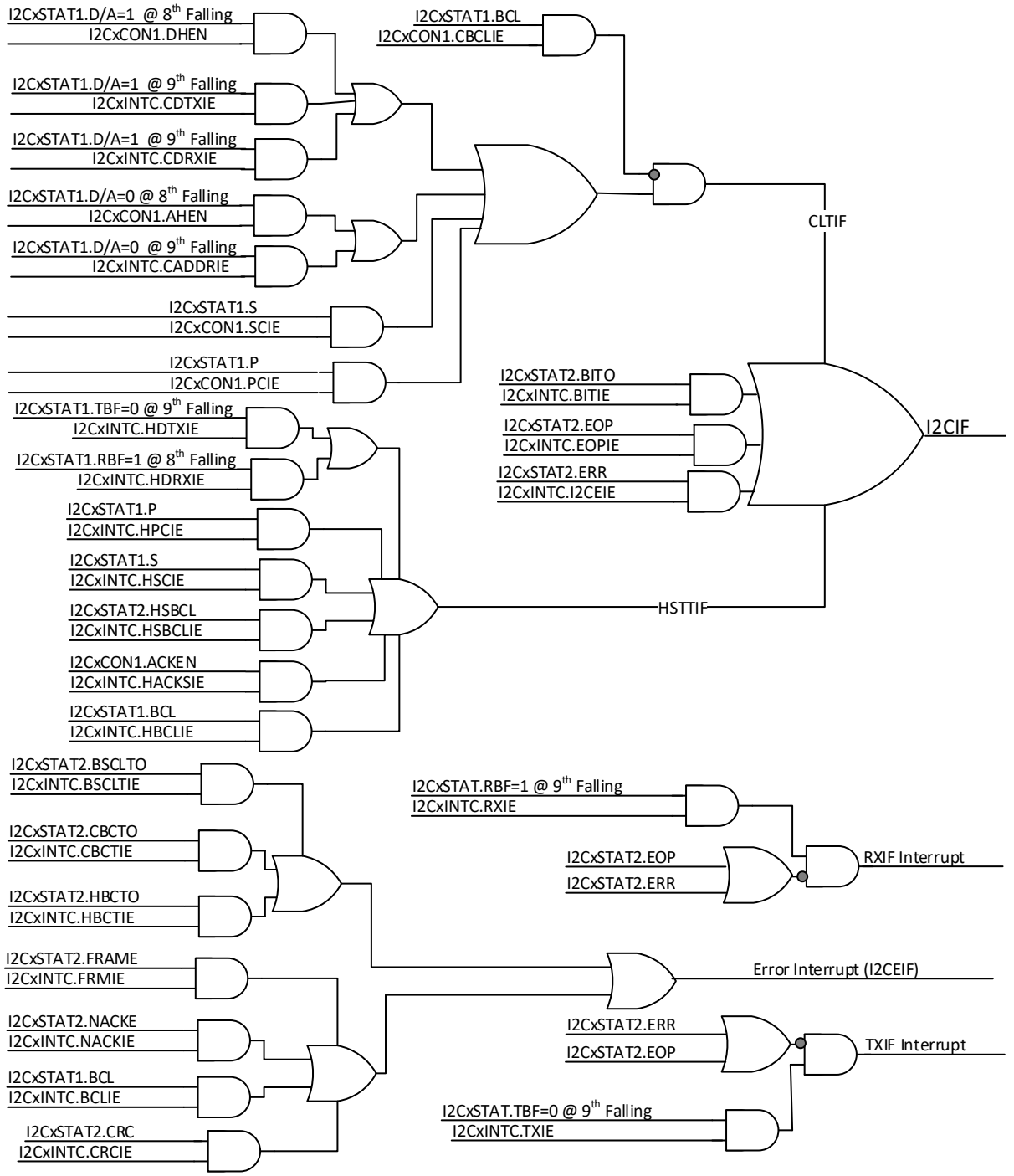
```
    IFS2bits.I2C1IF = 0;
    /* Read received address*/
    if ((I2C1STAT1bits.RBF)&& (!I2C1STAT1bits.D_A)) {
        gTrans_count = 0;
        gRcv_count = 0;
        gReceived[gRcv_count] = I2C1RCV;
        gRcv_count++;

    }/* Read received data byte*/
    else if ((I2C1STAT1bits.RBF)&& (I2C1STAT1bits.D_A)) {
        gReceived[gRcv_count] = I2C1RCV;
        gRcv_count++; // Read received data

    }
    /* transmit byte if R_W is set */
    if ((I2C1STAT1bits.R_W) &&(!I2C1STAT1bits.ACKSTAT)) {
        I2C1TRN = gTrans_count++; // transmit data

    }
}
```

20.7 Interrupts

I^2C has four separate interrupts. To determine which event has caused the interrupt, the associated flag needs to be read and evaluated. All interrupt sources are shown in Figure 20-41.

- Receive Buffer Full Interrupt (I2CTXIF)
- Transmit Buffer Empty Interrupt (I2CRXIF)
- Generic Module Interrupt (I2CIF)
- Error Interrupt (I2CEIF)

Figure 20-41. Interrupt Logic



- The Receive interrupt will be generated when EOP=1 to read the last byte by DMA
- The First Transmit Data byte write is expected write by the CPU
- The Transmit interrupt is not generated after the last byte Transmit
- The HACKSIE control the ACK Sequence completion interrupt generated at @9th falling edge after transmitting the ACK|NACK in the host receive mode.
- The TXIF,RXIF and I2CEIF are not in the status bits, those are actual interrupt outputs.
- Data transmission interrupt is not generated to get CRC value from CPU or DMA in the CRC Auto append mode since the transmitted CRC value is already present in the CRC calculator.

20.8 Operation in Power-Saving Modes

20.8.1 Sleep Mode in Client Mode

The I^2C module can wake-up from Sleep mode on detecting a valid client address match. Because all bit shifting is done with reference to the external SCLx signal, generated by an I^2C host, transmissions and receptions can continue even while in Sleep mode. SCL low timeout interrupt can wake up the client to release the bus to an Idle state. This can be enabled by setting BSCLTE (I2CxCON2[27])

Note: As per the client I^2C behavior, a client interrupt is generated only on an address match. Therefore, when an I^2C client is in Sleep mode and it receives a message from the host, the clock required to match the received address is derived from the host. Only on an address match will the interrupt be generated and the device can wake-up from Sleep, provided the interrupt has been enabled and an ISR has been defined.

20.8.2 Sleep Mode in Host Mode

If Sleep occurs in the middle of a host transmission and the state machine is partially into a transmission as the clocks stop, the behavior of the module will be undefined. Similarly, if Sleep occurs in the middle of a host reception, the module behavior will also be undefined. The transmitter and receiver will stop at Sleep when in Host mode. Register contents are not affected by going into Sleep mode or coming out of Sleep mode; there is no automatic way to prevent Sleep entry if a transmission or reception is pending. The user software must synchronize the Sleep entry with I^2C operation to avoid undefined module behavior.

20.8.3 When the Device Enters Idle Mode

When the device executes a PWRSAV 1 instruction, the device enters Idle mode. The module enters a power-saving state in Idle mode, depending on the I2CSIDL bit (I2CxCON1[13]). If I2CSIDL = `1`, the module enters a power-saving mode, similar to actions while entering Sleep mode. If I2CSIDL = `0`, the module does not enter a power-saving mode and continues to operate normally.

21. Single-Edge Nibble Transmission (SENT)

SENT is a unidirectional, single-wire communications protocol that is based on SAE J2716, “SENT – Single-Edge Nibble Transmission for Automotive Applications”. The protocol is designed for point-to-point transmission of signal values using a signal system based on successive falling edges. It allows for high-resolution data transmission with a lower system cost than available serial data solutions, and it is intended for use in applications where data need to be communicated from a sensor to a central controller, such as an Engine Control Unit (ECU).

The 16-bit SENT module is a dedicated hardware implementation of SAE J2716. The module can be configured for three main modes of operation:

- Asynchronous Transmitter (default)
- Synchronous Transmitter
- Receiver

The module also includes these features:

- Automatic Data Rate Synchronization
- Optional Automatic Detection of CRC Errors in Receive Mode
- Optional Hardware Calculation of CRC in Transmit Mode
- Support for Optional Pause Pulse Period
- Data Buffering for One Message Frame
- Selectable Data Length for Transmit/Receive from Three to Six Nibbles
- Automatic Detection of Framing Errors
- Separately Mappable Input and Output Functions on Devices with Peripheral Pin Select (PPS)

21.1 Device-Specific Information

Table 21-1. SENT Summary Table

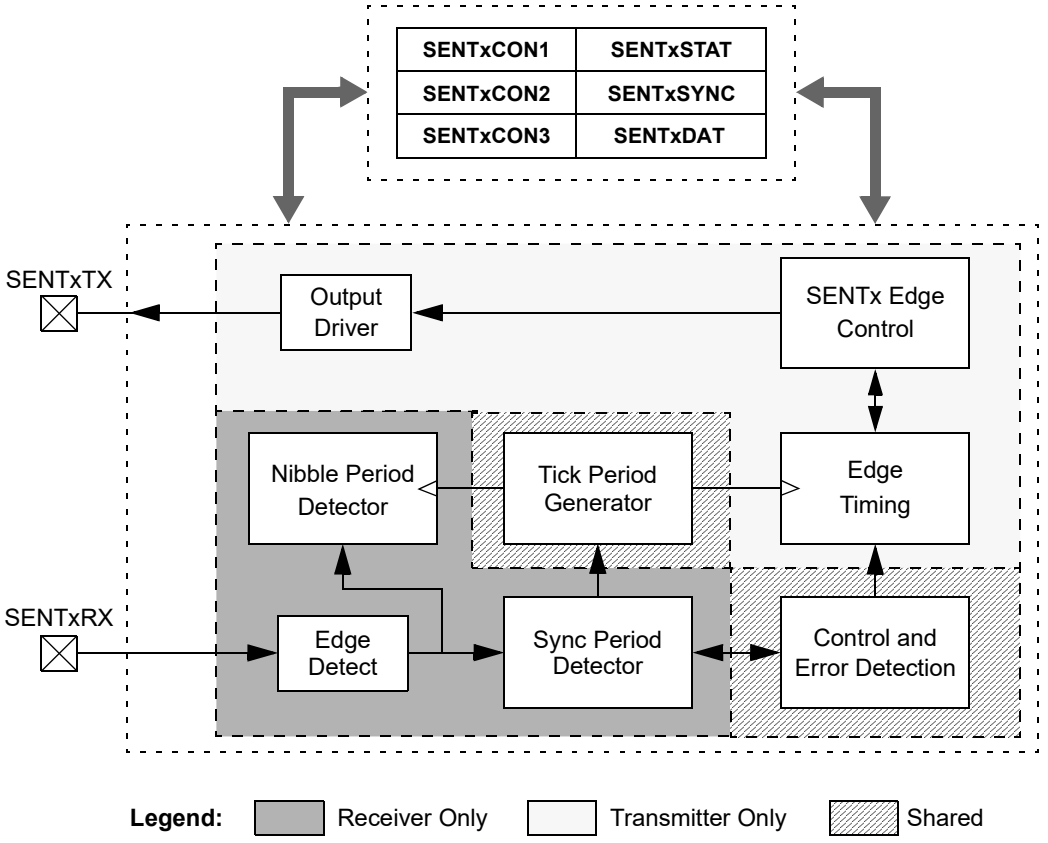
SENT Module Instances	PPS Available	Peripheral Bus Speed	Clock Source
2	All Instances	Standard	Standard Speed Peripheral Clock

21.2 Architectural Overview

SENT messages are encoded and decoded based on the time between falling edges. The protocol’s timing is based on a predetermined time unit, T_{TICK}, which can vary from 3 to 90 µs. Both the transmitter and receiver must be preconfigured for the same value of T_{TICK}. The SENT specification allows messages to be validated with up to a 20% variation in T_{TICK}. This allows for the transmitter and receiver to run from different clocks that may be inaccurate and drift with time and temperature.

An overview of the SENT module is shown in Figure 21-1.

Figure 21-1. SENTx Module Block Diagram

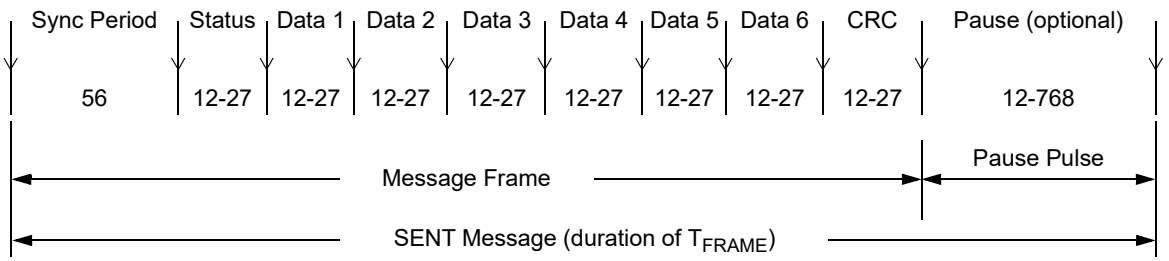


A SENT message consists of the following:

- A Synchronization/Calibration Period (“pulse”) of 56 Tick Times
- A Status Nibble of 12 to 27 Tick Times
- Up to Six Data Nibbles of 12 to 27 Tick Times
- A CRC Nibble of 12 to 27 Tick Times
- An Optional Pause Pulse Period of 12 to 768 Tick Times

The period from the start of the Sync period to the end of the CRC nibble comprises the message frame. When the optional Pause period is present, this makes one SENT message with a length of T_{FRAME} (generally expressed in µs). Figure 21-2 shows the construction of a typical six-nibble data frame, with the numbers representing the minimum or maximum number of tick times for each section.

Figure 21-2. SENT Message Format



The Sync period starts the message frame and is used for synchronization of T_{TICK} between the transmitter and receiver. When configured for Transmit mode, the module drives the line low for five ticks and to a High-Impedance state for 51 ticks.

A four-bit status nibble follows the Sync pulse and may be used for device status, identification or alternatively, used as additional data. The status nibble is formatted the same as a data nibble.

After the status nibble is one or more (up to six) data nibbles. These are four bits in length and are encoded as the data value plus 12 ticks. This yields a minimum value of 12 ticks for 0h and a maximum value of 27 ticks for Fh. When configured for Transmit mode, the module drives the line low for five ticks and into a High-Impedance state for the remaining 7 to 22 ticks.

The CRC data nibble follows the data payload. This is a 4-bit CRC of the six data nibbles only. The CRC is calculated using a polynomial of, $x^4 + x^3 + x^2 + 1$, with a seed value of '0101'. It is then padded with '0' to help detect shift errors. The CRC nibble is formatted the same as a data nibble.

Since the data values are encoded in the time between falling edges, the SENT protocol may produce a variable length message. In some applications, the Pause pulse period is used to pad the message length so that messages will always be received at a constant time interval. The module provides support to automatically calculate the Pause duration needed for periodic transmissions. When configured for Transmit mode, the module drives the line low for five ticks and into a High-Impedance state for the remaining Pause time.

Note: A SENT message frame will always have a status and CRC nibble. The shortest message frame with one data nibble (SENTxCON1[2:0] = 001) will have a length of one Sync and three nibbles.

21.2.1 Short and Enhanced Serial Message Formats

The J2716 specification defines two optional message formats: a Short Serial Message format and an Enhanced Serial Message format. Both of these message formats encode a longer serial message using two bits of the status nibble. A single message is encoded using multiple SENT data frames.

If the module is configured as a transmitter, the application must encode these serial messages by writing the appropriate value. If the module is configured as a receiver, the application must decode these serial messages by storing and analyzing the contents of the received status nibbles.

21.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x19C0	SENT1CON1	31:24								
		23:16								
		15:8	ON		SIDL		RCVEN	TXM	TXPOL	CRCEN
		7:0	PPP	SPCEN		PS		NIBCNT[2:0]		
0x19C4	SENT1CON2	31:24								
		23:16								
		15:8	TICKTIME/SYNCMAX[15:8]							
		7:0	TICKTIME/SYNCMAX[7:0]							
0x19C8	SENT1CON3	31:24								
		23:16								
		15:8	FRAMETIME/SYNCMIN[15:8]							
		7:0	FRAMETIME/SYNCMIN[7:0]							
0x19CC	SENT1STAT	31:24								
		23:16								
		15:8								
		7:0	PAUSE	NIB[2:0]			CRCERR	FRMERR	RXIDLE	SYNCTXEN
0x19D0	SENT1SYNC	31:24								
		23:16								
		15:8	SENTSYNC[15:8]							
		7:0	SENTSYNC[7:0]							
0x19D4	SENT1DAT	31:24	STAT[3:0]				DATA1[3:0]			
		23:16	DATA2[3:0]				DATA3[3:0]			
		15:8	DATA4[3:0]				DATA5[3:0]			
		7:0	DATA6[3:0]				CRC[3:0]			
0x19D8 ... 0x19DF	Reserved									
0x19E0	SENT2CON1	31:24								
		23:16								
		15:8	ON		SIDL		RCVEN	TXM	TXPOL	CRCEN
		7:0	PPP	SPCEN		PS		NIBCNT[2:0]		
0x19E4	SENT2CON2	31:24								
		23:16								
		15:8	TICKTIME/SYNCMAX[15:8]							
		7:0	TICKTIME/SYNCMAX[7:0]							
0x19E8	SENT2CON3	31:24								
		23:16								
		15:8	FRAMETIME/SYNCMIN[15:8]							
		7:0	FRAMETIME/SYNCMIN[7:0]							
0x19EC	SENT2STAT	31:24								
		23:16								
		15:8								
		7:0	PAUSE	NIB[2:0]			CRCERR	FRMERR	RXIDLE	SYNCTXEN
0x19F0	SENT2SYNC	31:24								
		23:16								
		15:8	SENTSYNC[15:8]							
		7:0	SENTSYNC[7:0]							
0x19F4	SENT2DAT	31:24	STAT[3:0]				DATA1[3:0]			
		23:16	DATA2[3:0]				DATA3[3:0]			
		15:8	DATA4[3:0]				DATA5[3:0]			
		7:0	DATA6[3:0]				CRC[3:0]			

21.3.1 SENTx Control Register 1

Name: SENTxCON1
Offset: 0x0019C0, 0x0019E0

Notes:

1. These bits have no function when RCVEN = 1.
2. This bit has no function when RCVEN = 0.
3. CCP3 OC output is internally connected with SENT1OUT pin, and CCP4 OC output is internally connected with SENT2OUT pin.

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON		SIDL		RCVEN	TXM	TXPOL	CRCEN
Access	R/W		R/W		R/W	R/W	R/W	R/W
Reset	0		0		0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PPP	SPCEN		PS		NIBCNT[2:0]		
Access	R/W	R/W		R/W		R/W	R/W	R/W
Reset	0	0		0		1	1	0

Bit 15 – ON SENTx Enable bit

Value	Description
1	Module is enabled
0	Module is disabled

Bit 13 – SIDL SENTx Stop in Idle Mode bit

Value	Description
1	Module stops operation in Idle mode
0	Module continues operation in Idle mode

Bit 11 – RCVEN SENTx Receive Enable bit

Value	Description
1	Module operates as a receiver
0	Module operates as a transmitter

Bit 10 – TXM SENTx Transmit Mode bit(1)

Value	Description
1	Module transmits data frame only when triggered using the SYNCTXEN status bit
0	Module transmits data frame continuously while enabled

Bit 9 – TXPOL SENTx Transmit Polarity bit(1)

Value	Description
1	Idle state of data output pin is low
0	Idle state of data output pin is high

Bit 8 – CRCEN CRC Enable bit

In Receive Mode (RCVEN = 1):
1 = CRC verification is performed using the J2716 method
0 = CRC verification is not performed
In Transmit Mode (RCVEN = 0):
1 = CRC is calculated using the J2716 method
0 = CRC is not calculated

Bit 7 – PPP Pause Pulse Present bit

Value	Description
1	SENTx messages transmitted/received with Pause pulse
0	SENTx messages transmitted/received without Pause pulse

Bit 6 – SPCEN Short PWM Code Enable bit(2,3)

Value	Description
1	SPC control from external source is enabled
0	SPC control from external source is disabled

Bit 4 – PS Prescale Select bit

Value	Description
1	1:4 (module clock is $T_{CY}/4$)
0	1:1 (module clock is T_{CY})

Bits 2:0 – NIBCNT[2:0] Nibble Count Control bits

Value	Description
111	Reserved: do not use
110	Six data nibbles per data packet
101	Five data nibbles per data packet
100	Four data nibbles per data packet
011	Three data nibbles per data packet
010	Two data nibbles per data packet
001	One data nibble per data packet
000	Reserved: do not use

21.3.2 SENTx Control Register 2

Name: SENTxCON2
Offset: 0x0019C4, 0x0019E4

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	TICKTIME/SYNCMAX[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	TICKTIME/SYNCMAX[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 15:0 – TICKTIME/SYNCMAX[15:0]

Module in Transmit Mode (RCVEN = 0):
TICKTIME[15:0]: This register value specifies the period for the tick clock generator.
Module in Receive Mode (RCVEN = 1):
SYNCMAX[15:0]: This register value specifies the maximum time limit for a valid Sync period.

21.3.3 SENTx Control Register 3

Name: SENTxCON3
Offset: 0x0019C8, 0x0019E8

Note:

1. The module will not produce a Pause period with less than 12 ticks, regardless of the FRAMETIME[15:0] value. FRAMETIME[15:0] values beyond 2047 will have no effect on the length of a data frame.

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	FRAMETIME/SYNCMIN[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	FRAMETIME/SYNCMIN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 15:0 – FRAMETIME/SYNCMIN[15:0]

Module in Transmit Mode (RCVEN = 0):
FRAMETIME[15:0]: This register value specifies the total number of ticks in a data frame if PPP = 1.(1)
Module in Receive Mode (RCVEN = 1):
SYNCMIN[15:0]: This register value specifies the minimum time limit for a valid Sync period.

21.3.4 SENTx Status Register

Name: SENTxSTAT
Offset: 0x0019CC, 0x0019EC

Note:
1. This bit is read-only in Receive mode and writable (settable and clearable) in Transmit mode.

Legend: R = Readable bit, W = Writable bit, C = Clearable bit, HC = Hardware Clearable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	PAUSE	NIB[2:0]			CRCERR	FRMERR	RXIDLE	SYNCTXEN
Access	R	R	R	R	R/C	R/C	R	R/W/HC
Reset	0	0	0	0	0	0	0	0

Bit 7 – PAUSE Pause Period Status bit

Value	Description
1	The module is transmitting/receiving a Pause period
0	The module is not transmitting/receiving a Pause period

Bits 6:4 – NIB[2:0] Nibble Status bits

In Transmit Mode (RCVEN = 0):
111 = Module is transmitting CRC nibble
110 = Module is transmitting Data Nibble 6
101 = Module is transmitting Data Nibble 5
100 = Module is transmitting Data Nibble 4
011 = Module is transmitting Data Nibble 3
010 = Module is transmitting Data Nibble 2
001 = Module is transmitting Data Nibble 1
000 = Module is transmitting status nibble or Pause period, or is not transmitting

In Receive Mode (RCVEN = 1):
111 = Module is receiving CRC nibble or was receiving this nibble when an error occurred
110 = Module is receiving Data Nibble 6 or was receiving this nibble when an error occurred
101 = Module is receiving Data Nibble 5 or was receiving this nibble when an error occurred
100 = Module is receiving Data Nibble 4 or was receiving this nibble when an error occurred
011 = Module is receiving Data Nibble 3 or was receiving this nibble when an error occurred
010 = Module is receiving Data Nibble 2 or was receiving this nibble when an error occurred
001 = Module is receiving Data Nibble 1 or was receiving this nibble when an error occurred

000 = Module is receiving Status nibble or waiting for Sync

Bit 3 – CRCERR CRC Status bit (Receive mode only)

Value	Description
1	A CRC error occurred for the data nibbles in the SENTxDAT register
0	No CRC error has occurred

Bit 2 – FRMERR Framing Error Status bit (Receive mode only)

Value	Description
1	A data nibble was received with less than 12 tick periods or greater than 27 tick periods
0	No framing error has occurred

Bit 1 – RXIDLE Receiver Idle Status bit (Receive mode only)

Value	Description
1	The SENTx data bus has been Idle (high) for a period of SYNCMAX or greater
0	The SENTx data bus is not Idle

Bit 0 – SYNCTXEN Synchronization Period Status/Transmit Enable bit(1)

In Receive Mode (RCVEN = 1):
1 = A valid synchronization period was detected, the module is receiving nibble data
0 = No synchronization period has been detected, the module is not receiving nibble data
In Synchronous Transmit Mode (RCVEN = 0, TXM = 1):
1 = The module is transmitting a SENT data frame
0 = The module is not transmitting a data frame; software may set SYNCTXEN to start another data frame transmission
In Asynchronous Transmit Mode (RCVEN = 0, TXM = 0):
SYNCTXEN always reads as '1', indicating the module is transmitting frames continuously.

21.3.5 SENTx Synchronization Time Capture Register

Name: SENTxSYNC
Offset: 0x0019D0, 0x0019F0

Note:

1. These register bits are not available in Transmit mode (RCVEN = 0).

Legend: R = Readable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	SENTSYNC[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SENTSYNC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – SENTSYNC[15:0] Captured Sync Period bits(1)

21.3.6 SENTx Data Register

Name: SENTxDAT
Offset: 0x0019D4, 0x0019F4

Notes:

1. Register bits are read-only in Receive mode (RCVEN = 1).
2. In Transmit mode, the CRC[3:0] bits are read-only when automatic CRC calculation is enabled (RCVEN = 0, CRCEN = 1).

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	STAT[3:0]				DATA1[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA2[3:0]				DATA3[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA4[3:0]				DATA5[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA6[3:0]				CRC[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:28 – STAT[3:0] Status Nibble Data bits(1)

Bits 27:24 – DATA1[3:0] Data Nibble #1 Data bits(1)

Bits 23:20 – DATA2[3:0] Data Nibble #2 Data bits(1)

Bits 19:16 – DATA3[3:0] Data Nibble #3 Data bits(1)

Bits 15:12 – DATA4[3:0] Data Nibble #4 Data bits(1)

Bits 11:8 – DATA5[3:0] Data Nibble #5 Data bits(1)

Bits 7:4 – DATA6[3:0] Data Nibble #6 Data bits(1)

Bits 3:0 – CRC[3:0] CRC Nibble Data bits(1,2)

21.4 Operation

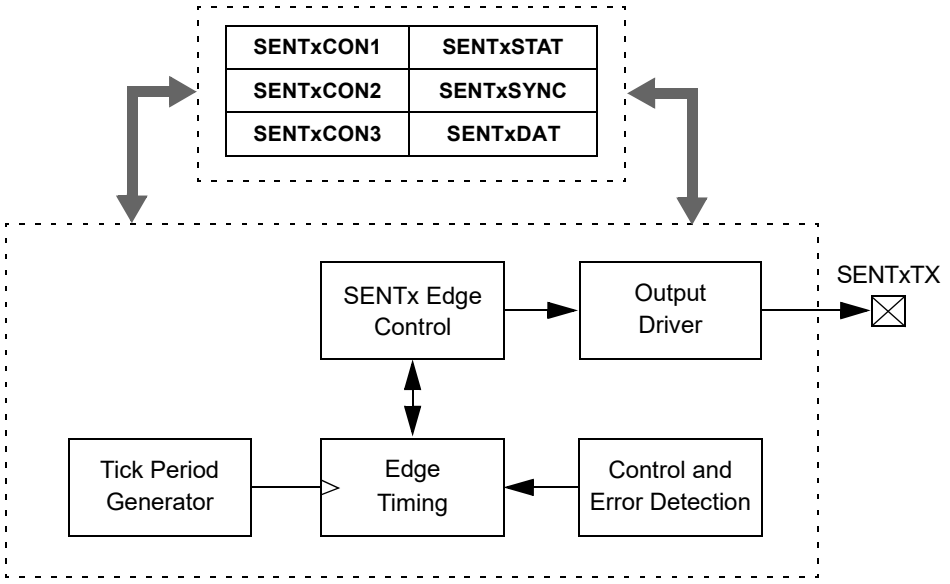
21.4.1 Transmit Mode

When RCVEN (SENTxCON1[11]) = 0, the module operates as a transmitter. Message frames are generated using the Configuration and Data registers.

The module has two transmit operating modes selected by the TXM bit (SENTxCON1[10]). Asynchronous mode (TXM = 0) continuously sends data message frames when the ON bit (SENTxCON1[15]) is set. Synchronous mode (TXM = 1) sends messages under software control to support additional capabilities, including Short PWM Code (SPC).

A block diagram of Transmit mode is shown in Figure 21-3.

Figure 21-3. SENTx Transmit Mode Block Diagram



21.4.1.1 Timing Calculations for Transmit Mode

In Transmit mode, SENTxCON2 and SENTxCON3 hold the values for TICKTIME[15:0] and FRAMETIME[15:0]. The tick period used by the SENT transmitter (T_{TICK}) is set by writing TICKTIME to the SENTxCON2 register. The tick period calculations are shown in Equation 21-1. The F_{CLK} value is either F_{CY} or $F_{CY}/4$, depending on the value of the Prescaler Enable bit, PS (SENTxCON1[4]).

Equation 21-1. Tick Period Calculation

$$TICKTIME = (T_{TICK} \bullet F_{CLK}) - 1$$

Where:

$$F_{CLK} = F_{CY}/Prescaler$$

If the Pause pulse is to be used, a frame period (T_{FRAME}) must be defined. This is done by writing the value of FRAMETIME[15:0] to SENTxCON3. The formulas used to calculate the value of FRAMETIME (in the same units as T_{TICK}) are shown in Equation 21-2. The FRAMETIME ranges for all settings are summarized in Table 21-2.

Equation 21-2. Calculating T_{FRAME}

$$FRAMETIME = T_{FRAME}\ (\mu s)/T_{TICK}$$

Where:

$$848 + 12N \geq FRAMETIME \geq 122 + 27N, \text{ for } 1 \leq N \leq 6$$

Table 21-2. Range of FRAMETIME[15:0] Values

Number of Data Nibbles	Min. FRAMETIME[15:0] Value	Max. FRAMETIME[15:0] Value
1	149	860
2	176	872
3	203	884
4	230	896
5	257	908
6	284	920

21.4.1.2 CRC Calculation

The module can optionally calculate the CRC using the recommended method shown in Example 21-1. The CRC is calculated when the CRCEN bit (SENTxCON1<8>) is set and the CRC[3:0] (SENTxDAT[3:0]) register bits become read-only. The hardware computed CRC value is indicated in the CRC[3:0] register bits when the calculation is finished.

When CRCEN = 0, no CRC is computed in hardware and the CRC[3:0] bits become writable by software. The application must compute a CRC value and write it to CRC[3:0].

Example 21-1. Recommended J2716 CRC Implementation

```
#define NUM_NIBBLES 6
// Array holding received nibbles
rec_data[NUM_NIBBLES];
// CRC lookup table
crc_table = {0,13,7,10,14,3,9,4,1,12,6,11,15,2,8,5};
// Initialize checksum to seed value
Checksum = 5;
// For each data nibble, bit-wise XOR with lookup value from table
for(i=0;i<NUM_NIBBLES;i++)
{
    Checksum = rec_data[i] ^ crc_table[Checksum];
}
// Bit-wise XOR with additional 0 value
Checksum = 0 ^ crc_table[Checksum];
```

21.4.1.3 Transmitter Status Bits

The SENTxSTAT register provides status information and control when in Transmit mode. The NIB[2:0] status bits (SENTxSTAT[6:4]) display the current data nibble transmitting during a message frame. If the Pause period is enabled (PPP (SENTxCON1[7]) = 1), the PAUSE bit (SENTxSTAT[7]) indicates when a Pause period transmission is in progress.

The SYNCTXEN bit (SENTxSTAT[0]) is used to initiate a synchronous transmission when TXM is set. SYNCTXEN is automatically cleared by hardware when all data nibbles, the CRC nibble and the Pause period have completed.

21.4.1.4 Transmit Polarity Option

The polarity of the SENT data pin can be inverted by setting the TXPOL bit (SENTxCON1[9]). This feature can be useful for implementing an external transistor drive circuit. Note that the High-Impedance (Idle) condition on the line remains in the same state (high) due to the external pull-up resistor network.

21.4.1.5 Transmit Output Pin and PPS

On devices with Peripheral Pin Select (PPS), the SENTx Transmit (SENTxTX) pin function is a remappable feature. To use the module in Transmit mode, SENTxTX must be mapped to an available I/O pin using the appropriate RPORx register.

21.4.1.6 Asynchronous Transmitter Mode

The module is, by default, configured as an asynchronous transmitter. In this mode, the module continuously transmits message frames as long as the ON bit is set (SENTxCON1[15]). The final falling edge of the CRC nibble also serves as the first falling edge of the Sync pulse. An interrupt is generated at the completion of the CRC nibble.

Figure 21-4 and Figure 21-5 show the relationship between the control, status and interrupt events.

Figure 21-4. SENTx Data Transmission, Asynchronous Mode

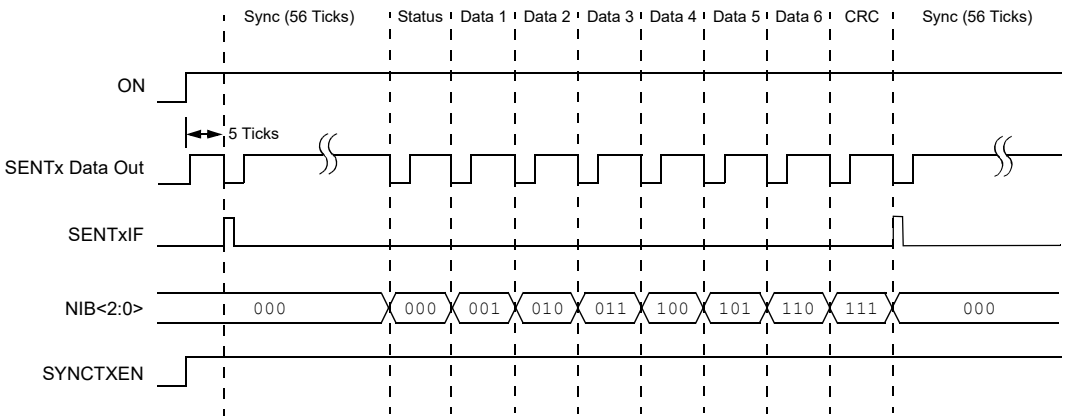
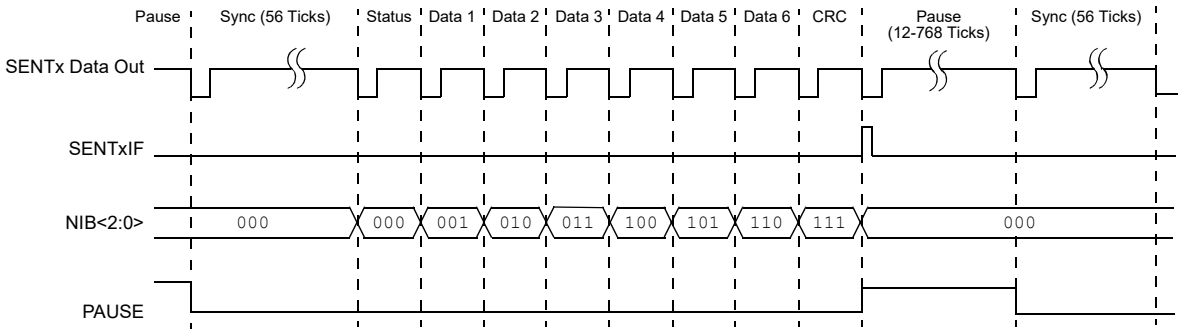


Figure 21-5. SENTx Data Transmission with Pause Period



To fully configure the module, the following must be known:

- Tick Time, T_{TICK} (Equation 21-1)
- Number of Data Nibbles, *N*
- Hardware or Application Calculated CRC
- Use of Pause Period for Fixed Message Period
- The Overall Duration of the SENT Message if the Pause Period is Present (Equation 21-2)

To initialize the module:

1. Clear RCVEN (SENTxCON1[11]) for Transmit mode.
2. Clear TXM (SENTxCON1[10]) for asynchronous transmit.
3. Write the value of the desired frame length to NIBCNT[2:0] (SENTxCON1<2:0>).
4. Set or clear CRCEN (SENTxCON1[8]) to configure the module for hardware or software CRC calculation.
5. Write the value for the desired TICKTIME to SENTxCON2.
6. If the optional Pause period is required, set PPP (SENTxCON1[7]) to enable the feature, then write the value of FRAMETIME to SENTxCON3.
7. Enable the SENT interrupt(s) and set the interrupt priority.
8. Write the initial status and data values to SENTxDAT. If application-based CRC is being used (CRCEN = 0), also calculate the message CRC and write it to CRC<[:0].
9. Set ON (SENTxCON1[15]) to enable the module.

Updates to SENTxDAT must be performed after the completion of the CRC and before the next message frame's status nibble. The recommended method is to use the message frame completion interrupt to trigger data writes.

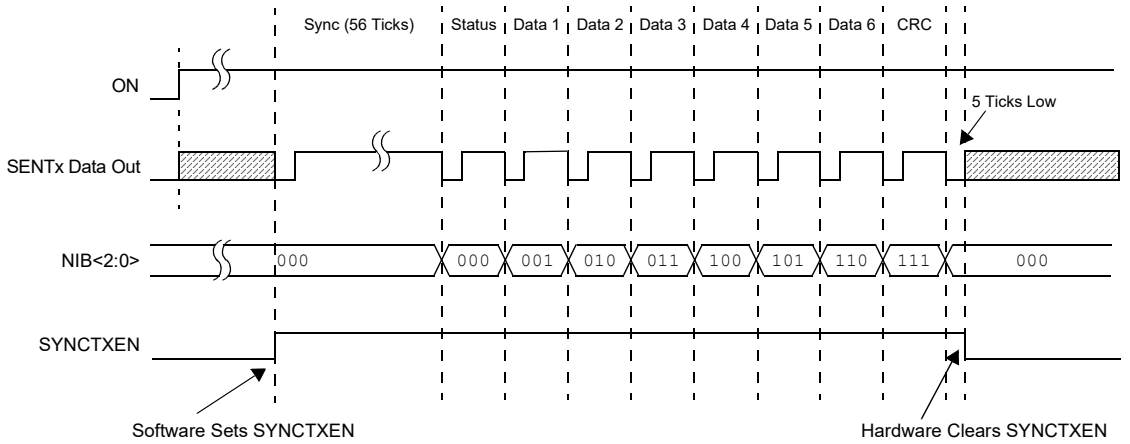
Example 21-2. SENT1 Asynchronous Transmission Code

```
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#define mFclk (4E+6)
#define mTickTime (70E-6)
#define mFrameTime (25E-3)
int main(void) {
    _RP23R = 57; // Assign SENT1OUT to pin RP23
    SENT1CON1bits.RCVEN = 0; // Module operates as a transmitter
    SENT1CON1bits.TXM = 0; // Asynchronous Transmit
    SENT1CON1bits.NIBCNT = 6; // 6 data nibbles per data packet
    SENT1CON1bits.CRCEN = 1; // CRC is calculated using the J2716 method
    SENT1CON1bits.PPP = 1; // SENTx messages transmitted with Pause Pulse
    SENT1CON1bits.PS = 0; // Module clock is TCY
    SENT1CON2 = ( (mTickTime * mFclk) - 1); // TICKTIME
    SENT1CON3 = ( mFrameTime / mTickTime) ; // FRAMETIME
    _SENT1IE = 1; // Enable SENT1 interrupt
    _SENT1IP = 4; // SENT interrupt priority
    SENT1DATbits.STAT = 0; // Status Nibble
    SENT1DATbits.DATA1 = 1; // Data Nibble 1
    SENT1DATbits.DATA2 = 2; // Data Nibble 2
    SENT1DATbits.DATA3 = 3; // Data Nibble 3
    SENT1DATbits.DATA4 = 4; // Data Nibble 4
    SENT1DATbits.DATA5 = 5; // Data Nibble 5
    SENT1DATbits.DATA6 = 6; // Data Nibble 6
    SENT1CON1bits.ON = 1; // Enable SENT module
    while(1);
    return 0;
}
void __attribute__((__interrupt__,__auto_psv__)) _SENT1Interrupt (void)
{
    _SENT1IF = 0; // Clear interrupt flag
    // Update SENT1DAT here
}
```

21.4.1.7 Synchronous Transmitter Mode

The module can be alternatively configured as a synchronous transmitter. In this mode, the module will transmit only one message frame each time the SYNCTXEN bit (SENTxSTAT[0]) is set. When the data frame is complete, the SYNCTXEN bit will be cleared in hardware. The line will be driven low for five ticks to complete the CRC nibble and then the line will tri-state and remain in the Idle state until SYNCTXEN is set again. An interrupt is generated, five ticks after the completion of the CRC nibble. Figure 21-6 shows the relationship between the control, status and interrupt events.

Figure 21-6. SENTx Data Transmission, Synchronous Mode



To fully configure the module, the following must be known:

- Tick Time, T_{TICK} (Equation 21-1)
- Number of Data Nibbles
- Hardware or Application Calculated CRC

To initialize the module for synchronous transmission:

1. Clear the RCVEN bit (SENTxCON1[11]) for Transmit mode.
2. Set the TXM bit (SENTxCON1[10]) for synchronous transmit operation.
3. Write the desired data frame length to NIBCNT[2:0] (SENTxCON1[2:0]).
4. Set or clear CRCEN (SENTxCON1[8]) to configure the module for hardware or software CRC calculation.
5. Write the desired value for TICKTIME to SENTxCON2.
6. Enable the SENT interrupts and set the interrupt priority.
7. Set the ON bit (SENTxCON1[15]) to enable module.

When the application is ready to transmit data:

1. Write the data to be transmitted to the SENTxDAT register.
2. Set the SYNCTXEN bit to begin transmission.

Updates to SENTxDAT must be performed after the completion of the CRC and before the next message frame's status nibble. The recommended method is to use the message frame completion interrupt or poll the SYNCTXEN bit.

Note: Software may need to wait additional time before starting a new message frame. The J2716 specification allows up to 18 µs of rise time on the SENT data line. The rise time will be a function of the external pull-up resistor and EMI filtering on the SENT data line. It is recommended that the wait time be longer than one Sync time of 56 ticks + 20%.

Example 21-3. SENT1 Synchronous Transmission Code

```
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#define mFclk (4E+6)
#define mTickTime (70E-6)
#define mFrameTime (25E-3)
int main(void) {
    _RP23R = 57; // Assign SENT1OUT to pin RP23
    SENT1CON1bits.RCVEN = 0; // Module operates as a transmitter
    SENT1CON1bits.TXM = 1; // Synchronous Transmit
    SENT1CON1bits.NIBCNT = 6; // 6 data nibbles per data packet
    SENT1CON1bits.CRCEN = 1; // CRC is calculated using the J2716 method
    SENT1CON1bits.PPP = 1; // SENTx messages transmitted with Pause Pulse
    SENT1CON1bits.PS = 0; // Module clock is TCY
    SENT1CON2 = ( (mTickTime * mFclk) - 1); // TICKTIME
    SENT1CON3 = ( mFrameTime / mTickTime) ; // FRAMETIME
    _SENT1IE = 1; // Enable SENT1 interrupt
    _SENT1IP = 4; // SENT interrupt priority
    SENT1DATbits.STAT = 0; // Status Nibble
    SENT1DATbits.DATA1 = 1; // Data Nibble 1
    SENT1DATbits.DATA2 = 2; // Data Nibble 2
    SENT1DATbits.DATA3 = 3; // Data Nibble 3
    SENT1DATbits.DATA4 = 4; // Data Nibble 4
    SENT1DATbits.DATA5 = 5; // Data Nibble 5
    SENT1DATbits.DATA6 = 6; // Data Nibble 6
    SENT1CON1bits.ON = 1; // Enable SENT module
    SENT1STATbits.SYNCTXEN = 1; // Initiate Synchronous Transmission
    while(1);
    return 0;
}
void __attribute__((__interrupt__,__auto_psv__)) _SENT1Interrupt (void)
{
    _SENT1IF = 0; // Clear interrupt flag
    // Update SENT1DAT here
}
```

21.4.1.7.1 Short PWM Code (SPC) Support

Short PWM Code can be implemented with user software using Synchronous mode. SPC allows bidirectional communication, such as allowing the receiver to request a message frame from the transmitter, change modes or to calibrate a sensor.

The SPC pulse is an active-low pulse initiated by the receiver. Since the module does not provide hardware functionality for the detection of SPC pulses, the application will need to detect the SPC pulses on the SENT data pin.

When the transmitter is Idle, user software can use one of these methods to detect an SPC pulse on the SENT data pin:

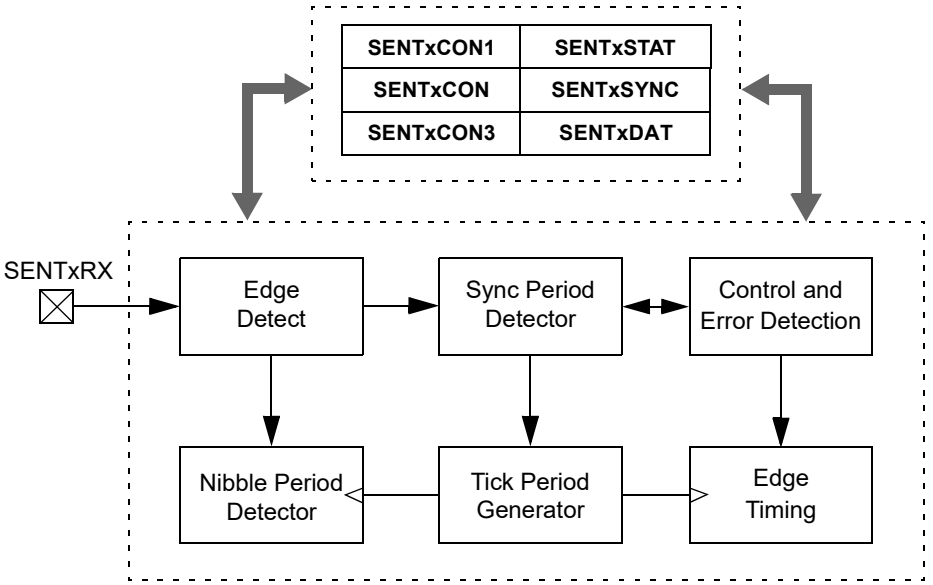
- Poll the data pin using the PORTx register associated with the pin
- Enable an input capture peripheral that is multiplexed on the SENT data pin
- Enable a Change Notification (CN) input or a comparator that is multiplexed on the SENT data pin

After an SPC pulse is found, the application disables any peripherals associated with SPC pulse detection, then initiates a SENT transmission by setting the SYNCTXEN bit.

21.4.2 Receive Mode

The module can be operated as a receiver when RCVEN (SENTxCON1[11]) = 1. If the serial data are valid, they are decoded by the module and made available in the SENTxDAT register. Error checking and status information are made available in the SENTxSTAT and SENTxSYNC registers. The captured Sync period value is readable in the SENTxSYNC register. A block diagram of the module in Receive mode is shown in Figure 21-7.

Figure 21-7. SENTx Receive Mode Block Diagram



21.4.2.1 Receive Mode Timing Calculations

When configured for Receive mode, the SENTxCON2 and SENTxCON3 registers are used to hold the maximum (SYNCMAX[15:0]) and minimum (SYNCMIN[15:0]) boundary values of the Sync period for validation of the Sync pulse. SYNCMIN and SYNCMAX are ±20% of the nominal Sync period. Received Sync periods outside this window will be rejected by the receiver.

The equation for SYNCMIN and SYNCMAX is shown in Equation 21-3. As in the Transmit modes, the value for F_{RCV} is either F_{CY} or $F_{CY}/4$, depending on the setting of the PS bit.

Equation 21-3. Calculating SYNCMIN and SYNCMAX

$$\text{SyncCount} = 8 \bullet F_{RCV} \bullet T_{TICK}$$
$$\text{SYNCMIN} = 0.8 \bullet \text{SyncCount}$$
$$\text{SYNCMAX} = 1.2 \bullet \text{SyncCount}$$

Where:

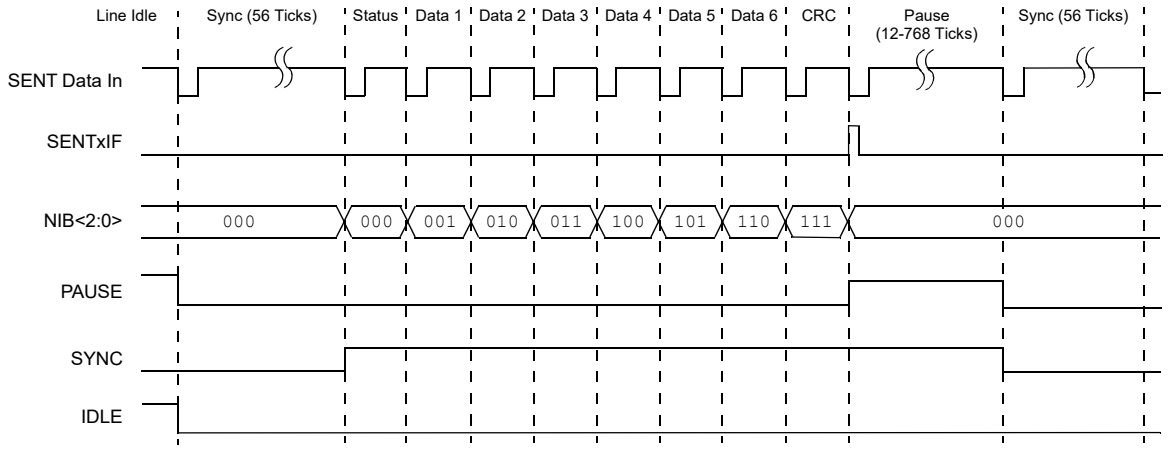
$$F_{RCV} = F_{CY}/\text{Prescaler}$$

Note: The SENT protocol allows no more than 1.5625% (1 part in 64) timing variation between successive messages. To verify this condition is met, save the value captured in SENTxSYNC after each data frame is received, then compare it to the value received during the next data frame.

21.4.2.2 Receiver Status

When the module is configured as a receiver (RCVEN = 0), the status of the received message (nibble status, line state, Sync and Pause states) is stored in the SENTxSTAT register. Figure 21-8 shows the relationship between the SENT data in the signal and the SENTxSTAT status bits.

Figure 21-8. SENTx Data Reception with Pause Period



Note: The receiver is in the Idle state when the SENT data line has been high for more than the maximum time allowed for a Sync period (SYNCMAX bits register value).

21.4.2.3 Receive Input Pin and PPS

On devices with Peripheral Pin Select (PPS), the SENT Receive (SENTxRX) pin function is an independently remappable feature. To use the module in Receive mode, SENTxRX must be mapped to an available I/O pin using the appropriate RPINRx or RPORx register. If Short PWM Code support is required, a bidirectional mappable pin must be used in order to support the pulse output.

21.4.2.4 Receive Setup Procedure

To fully configure the module, the following must be known:

- T_{TICK} (Equation 21-1)
- Number of Data Nibbles
- Hardware or Application Calculated CRC Validation
- Pause Period Present

Note: Application software can be used to implement an alternate CRC algorithm. In these instances, disable hardware CRC checking by clearing CRCEN (SENTxCON1[8]). The received CRC value (SENTxDAT[3:0]) can be read and compared against the application calculated CRC value.

To initialize the module for Receive mode:

1. Set RCVEN (SENTxCON1[11]) for Receive mode.
2. Write the desired data frame length to NIBCNT[2:0] (SENTxCON1[2:0]).
3. Set or clear CRCEN (SENTxCON1[8]) to configure the module for hardware or software CRC calculation.
4. If the Pause period is present, set PPP (SENTxCON1[7]).
5. Write the value of SYNCMAX (nominal Sync period + 20%) to SENTxCON2.
6. Write the value of SYNCMIN (nominal Sync period – 20%) to SENTxCON3.
7. Enable the SENT interrupts and set the interrupt priority.
8. Set the ON bit (SENTxCON1[15]) to enable the module.

Incoming data are read from the SENTxDAT register after the completion of the CRC and before the next message frame's status nibble. The recommended method is to use the message frame completion interrupt trigger.

Example 21-4. SENT1 Reception Code

```
#include<xc.h>
#define mFclk (4E+6)
#define mTickTime (70E-6)
#define mFrameTime (25E-3)
#define mSyncCount (8 * mFclk * mTickTime)
#define mSyncMin (0.8 * mSyncCount)
#define mSyncMax (1.2 * mSyncCount)
uint8_t ReceivedData[6];
uint8_t i;
int main(void) {
    _SENT1R = 24; // RP24 as SENT1 RX pin
    SENT1CON1bits.RCVEN = 1; // Module operates as a receiver
    SENT1CON1bits.NIBCNT = 6; // 6 data nibbles per data packet
    SENT1CON1bits.CRCEN = 1; // CRC is calculated using the J2716 method
    SENT1CON1bits.PPP = 1; // SENTx messages transmitted with Pause Pulse
    SENT1CON1bits.PS = 0; // Module clock is TCY
    SENT1CON2 = mSyncMax; // TICKTIME
    SENT1CON3 = mSyncMin; // FRAMETIME
    _SENT1IE = 1; // Enable SENT1 interrupt
    _SENT1IP = 4; // set SENT interrupt priority
    SENT1CON1bits.ON = 1; // Enable SENT module
    while(1);
    return 0;
}
void __attribute__((__interrupt__,__auto_psv__)) _SENT1Interrupt (void)
{
    _SENT1IF = 0; // Clear interrupt flag
    i=0;
    ReceivedData[i++] = SENT1DATbits.DATA1; // Read Data Nibble 1
    ReceivedData[i++] = SENT1DATbits.DATA2; // Read Data Nibble 2
    ReceivedData[i++] = SENT1DATbits.DATA3; // Read Data Nibble 3
    ReceivedData[i++] = SENT1DATbits.DATA4; // Read Data Nibble 4
    ReceivedData[i++] = SENT1DATbits.DATA5; // Read Data Nibble 5
    ReceivedData[i++] = SENT1DATbits.DATA6; // Read Data Nibble 6
}
```

21.4.2.5 Error Handling

The module has the capability to automatically detect and flag framing errors and CRC mismatch. A framing error is the detection of a status or data nibble period, less than 12 ticks or greater than 27 ticks. If a framing error is detected, the FRMERR bit (SENTxSTAT[2]) is set and a receive error interrupt is generated. After a framing error has been detected, the module clears SYNCTXEN (SENTxSTAT[0]) and begins looking for another valid Sync period. The FRMERR bit remains set until another valid Sync period has been detected and SYNCTXEN = 1. The application may optionally clear the FRMERR bit.

Note: If the PPP bit is set, the module will not generate a framing error on successive valid Sync pulses. This is due to the possibility that a Pause pulse could be interpreted as a valid Sync and the following actual Sync pulse could be interpreted as a frame error. Framing errors for nibble data will still be detected when PPP = 1.

If CRC verification fails, the CRCERR bit (SENTxSTAT[3]) is set and a receive error interrupt is generated. The CRCERR bit remains set until a valid Sync period for a new message has been received. Software may optionally clear the CRCERR bit. The CRCEN bit (SENTxCON1[8]) must be set to receive an interrupt on a CRC error.

21.4.2.6 Short PWM Code (SPC) Support

The SENT module provides support for implementing SPC with assistance from other external peripherals. The SPCEN (SENTxCON1[6]) bit enables an external Output Compare (OC) peripheral to control the SENT data input pin. In general, a specific OC module is linked in hardware to a specific SENTx module.

To initialize the SENT module for SPC operation:

1. Set the SPCEN (SENTxCON1[6]) bit to enable control of the SENT data pin by an external source.

2. For devices with Peripheral Pin Select, map the SENTxTX function to the same I/O pin as SENTxRX.
3. Configure the Output Compare module as follows:
   a. Configure the module for Triggered mode.
   b. Configure for a single-shot, active-high pulse.
   c. Set the Period and Duty Cycle registers for the desired pulse duration.

After configuration, to use the SPC pulse trigger:

1. Verify that the line is in a High-Impedance state by polling the RXIDLE bit (SENTxSTAT[1]).
2. Set the Trigger bit of the OC module to trigger the SPC pulse.

During the active period of the SPC pulse, the SENT receiver edge detection is disabled and the SENT data input pin is driven low by the module. At this time, the receiver logic is reset to prepare for a new data frame. When the pulse is completed, the module releases control of the SENT data input pin and input edge detection is re-enabled, so a data frame can be received from the sensor.

Note: To implement the SPC protocol, the SENT transmitting device(s) must leave the data bus in a High-Impedance state after the falling edge that completes the CRC nibble period. At this time, the SENT data line will be pulled high by the external pull-up resistor. The data bus should not be driven by any transmitter devices until the receiver device requests data by placing a low pulse on the SENT data line.

It is also possible to manually control the SENT data pin to implement the SPC protocol. This can be done as follows:

1. Clear the ON bit to disable receiver operation.
2. Manipulate the PORTx and TRISx registers associated with the SENT data pin to drive the data pin low for the desired time.
3. Return the SENT data pin to a High-Impedance condition using the TRISx register.
4. Set the ON bit to resume receiver operation.

21.5 Application Examples

Example 21-5. SENT Transmission (SPC Pulse Reception)

```
#include <xc.h>
#define FCY (8E+06)
#include <libpic30.h>

#define mFclk (4E+6)
#define mTickTime (50E-6)
#define mFrameTime (25E-3)
#define mSPCPulseWidth (56 * mTickTime)
void InputCapture_configure(void);
void SENT_Tx_configure(void);
uint8_t count = 0;
uint16_t falling_edge_capture = 0, rising_edge_capture = 0,delta,firstRead;
float pulseWidth;

int main(void) {
    SENT_Tx_configure(); // Configure SENT for Transmission
    InputCapture_configure(); // Configure Input Capture
    while (1);
    return 0;
}

void InputCapture_configure(void) {
    _ICM1R = 27; // RP27 as Input Capture 1
    CCP1CON1bits.CCSEL = 1; // Input capture mode
    CCP1CON1bits.CLKSEL = 0; // Set the clock source (Tcy)
    CCP1CON1bits.T32 = 0; // 16-bit Dual Timer mode
    CCP1CON1bits.MOD = 3; // Capture every edge of the event
    CCP1CON2bits.ICS = 0; // Capture rising edge on the Pin
    CCP1CON1bits.OPS = 0; // Interrupt on every input capture event
    CCP1CON1bits.TMRPS = 0; // Set the clock pre-scaler (1:1)
    CCP1CON1bits.ON = 1; // Enable CCP/input capture
    CCP1TMR = 0;
    __delay_ms(10);
```

```
    _CCP1IF = 0; // Clear CCP interrupt flag
    _CCP1IE = 1; // Enable CCP interrupt

}

void SENT_Tx_configure(void) {
    _RP27R = 57; // RP27 as SENT1 output
    SENT1CON1bits.RCVEN = 0; // Module operates as a transmitter
    SENT1CON1bits.TXM = 1; // Synchronous Transmit
    SENT1CON1bits.NIBCNT = 6; // 6 data nibbles per data packet
    SENT1CON1bits.CRCEN = 1; // CRC is calculated using the J2716 method
    SENT1CON1bits.PPP = 1; // SENTx messages transmitted with Pause Pulse
    SENT1CON1bits.PS = 0; // Module clock is TCY
    SENT1CON2 = ((mTickTime * mFclk) - 1); // TICKTIME
    SENT1CON3 = (mFrameTime / mTickTime); // FRAMETIME
    _SENT1IE = 1; // Enable SENT1 interrupt
    SENT1CON1bits.ON = 1; // Enable SENT module
    SENT1DATbits.STAT = 0; // Status Nibble
    SENT1DATbits.DATA1 = 1; // Data Nibble 1
    SENT1DATbits.DATA2 = 2; // Data Nibble 2
    SENT1DATbits.DATA3 = 3; // Data Nibble 3
    SENT1DATbits.DATA4 = 4; // Data Nibble 4
    SENT1DATbits.DATA5 = 5; // Data Nibble 5
    SENT1DATbits.DATA6 = 6; // Data Nibble 6
}

void __attribute__((__interrupt__, __no_auto_psv__)) _CCP1Interrupt(void) {

    _CCP1IF = 0; // Clear interrupt flag
    count++;
    if (count == 1) // Falling edge of the SPC pulse
    {
        firstRead = CCP1BUF;
        falling_edge_capture = CCP1BUF;
    } else if (count == 2)// Raising edge of the SPC pulse
    {
        count = 0; // Clear count
        rising_edge_capture = CCP1BUF;
        if (rising_edge_capture <= falling_edge_capture) {
            delta = (0xFFFF - falling_edge_capture) + rising_edge_capture;// Rollover case
        } else {
            delta = rising_edge_capture - falling_edge_capture;// Non-rollover case
        }
        pulseWidth = delta / mFclk; // Calculate the pulse width

        // Check if pulse width is within the range
        if ((mSPCPulseWidth * 1.2) >= pulseWidth && (mSPCPulseWidth * 0.8) <= pulseWidth) {
            _CCP1IE = 0; // Disable CCP interrupt
            CCP1CON1bits.ON = 0; // Disable CCP/input capture
            SENT1STATbits.SYNCTXEN = 1; // Initiate Synchronous Transmission
        }
    }
}

void __attribute__((__interrupt__, __no_auto_psv__)) _SENT1Interrupt(void) {
    _SENT1IF = 0; // Clear SENT1 interrupt flag
    while (SENT1STATbits.PAUSE == 1); // Wait till PAUSE pulse is transmitted
    CCP1CON1bits.ON = 1; // Enable CCP/input capture
    CCP1TMR = 0;
    __delay_ms(1);
    _CCP1IF = 0; // Clear CCP interrupt flag
    _CCP1IE = 1; // Enable CCP interrupt
}
```

Example 21-6. SENT Reception (SPC Pulse Transmission)

```
#include <xc.h>
#define FCY (8E+6)
#include<libpic30.h>

#define mFclk (4E+6)
#define mTickTime (50E-6)
#define mFrameTime (25E-3)
#define mSyncCount (8 * mFclk * mTickTime)
#define mSyncMin (0.8 * mSyncCount)
#define mSyncMax (1.2 * mSyncCount)
void OutputCompare_configure();
void SENT_RX_configure();
void SendSPCPulse();
uint8_t mReceivedData[7];
uint8_t i;

int main(void) {
```

```
    ANSELB = 0;
    _TRISC6 = 1; // Connect a Pull-Up switch to RC15
    OutputCompare_configure(); // Configure Output Compare
    SENT_RX_configure(); // Configure SENT for Reception
    while(_RC6 == 0); // Wait till pull-up switch is pressed
    SendSPCPulse(); // Send SPC pulse
    while(1)
    {
        if(SENT1STATbits.RXIDLE == 1) // Check if the line is Idle
        {
            //Receiver can request for SENT data through SendSPCPulse()
function
            SendSPCPulse();
        }
    }
    return 0;
}
void OutputCompare_configure(){
    _CCP3IF = 0; // Clear CCP interrupt flag
    _CCP3IE = 1; // Enable CCP interrupt
    // Set MCCP operating mode
    CCP3CON1bits.CCSEL = 0; // Set MCCP operating mode (OC mode)
    CCP3CON1bits.MOD = 0b100; // Set mode
    //Configure MCCP Timebase
    CCP3CON1bits.T32 = 0; // Set timebase width (16-bit)
    CCP3CON1bits.TMRSYNC = 0; // Set timebase synchronization
    CCP3CON1bits.CLKSEL = 0b000; // Set the clock source (Tcy)
    CCP3CON1bits.TMRPS = 0b00; // Set the clock pre-scaler (1:1)
    CCP3CON1bits.ONESHOT = 1;
    CCP3CON1bits.TRIGEN = 1; // Set Sync/Triggered mode (Synchronous)
    CCP3CON1bits.SYNC = 0b1001; // Select Sync/Trigger source
    //Configure MCCP output for PWM signal
    CCP3CON2bits.OCAEN = 1; // Enable desired output signals (OC1A)
    CCP3CON3bits.POLACE = 1; //Configure output polarity (Active High)
    CCP3CON3bits.OSCNT = 0;
    CCP3TMR = 0x0000; //Initialize timer prior to enable module.
    CCP3RA = (56 * mTickTime * mFclk); // Set the rising edge compare value
    CCP3RB = (56 * mTickTime * mFclk)*2; // Set the falling edge compare value
    CCP3PR = CCP3RB; //Configure timebase period
    CCP3CON1bits.ON = 1; // Enable MCCP module
}
void SENT_RX_configure()
{
    _RP26R = 57; // RP26 as SENT1 output
    _SENT1R = 26; // RP26 as SENT1 input
    SENT1CON1bits.RCVEN = 1; // Module operates as a receiver
    SENT1CON1bits.NIBCNT = 6; // 6 data nibbles per data packet
    SENT1CON1bits.CRCEN = 1; // CRC is calculated using the J2716 method
    SENT1CON1bits.PPP = 1; // SENTx messages transmitted with Pause Pulse
    SENT1CON1bits.PS = 0; // Module clock is TCY
    SENT1CON1bits.SPCEN = 1; // Enable SPC
    SENT1CON2 = mSyncMax; // TICKTIME
    SENT1CON3 = mSyncMin; // FRAMETIME
    _SENT1IE = 1; // Enable SENT1 interrupt
    SENT1CON1bits.ON = 1; // Enable SENT module
}
void SendSPCPulse()
{
    CCP3STATbits.TRSET = 1; // Set the Trigger
    while(CCP3STATbits.CCPTRIG);
}

void __attribute__((__interrupt__,__no_auto_psv__)) _CCP3Interrupt (void)
{
    _CCP3IF = 0; // Clear interrupt flag
    CCP3STATbits.TRSET = 0; // Set the Trigger
    CCP3CON1bits.ON = 0; // Disable MCCP module
}
void __attribute__((__interrupt__,__no_auto_psv__)) _SENT1Interrupt (void)
{
    _SENT1IF = 0; // Clear interrupt flag
    i=0;
    mReceivedData[i++] = SENT1DATbits.STAT; // Read STAT
    mReceivedData[i++] = SENT1DATbits.DATA1; // Read DATA1
    mReceivedData[i++] = SENT1DATbits.DATA2; // Read DATA2
    mReceivedData[i++] = SENT1DATbits.DATA3; // Read DATA3
    mReceivedData[i++] = SENT1DATbits.DATA4; // Read DATA4
    mReceivedData[i++] = SENT1DATbits.DATA5; // Read DATA5
```

```
    mReceivedData[i++] = SENT1DATbits.DATA6; // Read DATA6
    while(SENT1STATbits.PAUSE == 1); // Wait till PAUSE pulse is received
    CCP3CON1bits.ON = 1; // Enable MCCP module
}
```

21.6 Interrupts

Each SENT module has two interrupts associated with its operation: the SENTx Transmit/Receive Interrupt Flag (SENTxIF) and the SENTx Error Interrupt Flag (SENTxEIF). Setting the corresponding SENTx Interrupt Enable bits (SENTxIE and SENTxEIE) allows the module to generate device-level interrupts.

The transmit/receive interrupt is generated after the transmission of a message frame in Transmit mode or the successful reception of a message frame in Receive mode.

The error interrupt is generated after a frame error or a CRC error in Receive mode. There are no error interrupts generated in Transmit mode.

21.7 Operation in Power-Saving Modes

21.7.1 Sleep Mode

The SENT module does not support operation when the device is in Sleep mode. If the application requires the device to enter Sleep mode, the module should be halted by clearing the ON bit (SENTxCON1[15]). If operated as a transmitter, the application can wait until the module is done transmitting a message frame before entering Sleep.

21.7.2 Idle Mode

The SENT module provides two options of operation when the device enters Idle mode. If SIDL (SENTxCON1[13]) = 1, module operation stops when the device enters Idle mode. The same system considerations described in Sleep mode apply.

If SIDL = 0, the module will continue the transmission or reception process after the device enters Idle mode. When operating in Transmitter mode, the module will continue to send messages with the data contained in SENTxDAT unless the device wakes from Idle to write new data. If the module is operating in Receive mode, any old data in the SENTxDAT register will be lost unless the device wakes from Idle to read the data.

21.8 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the SENT module to be turned off and any message frames in progress to be aborted. All SENT pins that are multiplexed with analog inputs are configured as analog inputs. The corresponding TRISx bits are also set, effectively making all pins inputs.

22. Bidirectional Serial Synchronous (BiSS) Module

The Bidirectional Serial Synchronous (BiSS) protocol is an open source digital interface for sensors and actuators. This protocol and module are compatible with the industrial standard serial synchronous interface (SSI).

The BiSS interface is used in position control applications. The interface enables a complete closed-loop position control system by providing the real-time position feedback to the host of the control motor. These applications can include:

- Bidirectional Communication in Multi Sensor Systems
- Linear and Rotary Encoders
- Motor Feedback Systems
- PLC Systems
- Drives

The Bidirection Serial Synchronous interface consists of the following key features:

- Bidirectional Data Communication that is Serial, Synchronous, and Continuous.
- 4 Serial Communication Lines
- 8 Data Channel Support with Ability to Add More
- 64-bit Sensor Data Lengths
- Cyclic at High Speeds
- Speeds Up to 10 MHz (BiSS) or 1.5 MHz (SSI)
- Auto Sense Frequency
- Variable Clock Rate
- Line Delay Compensation
- Safety Features Such as CRC, Error Flags and Warning Flags
- Ready to Use with RS485/422 Physical Interfaces
- BiSS/BiSS-C and SSI Compatibility
- Host and Client Configurations (Point-to-Point)
- Host and Multiple Clients (Point-to-Point)
- Host and Multiple Clients on Bus (Bus Configuration or BiSS)

22.1 Device-Specific Information

Table 22-1. BiSS Summary Table

BiSS Module Instances	Peripheral Bus Speed	Clock Source
1	Standard	See Table 22-2

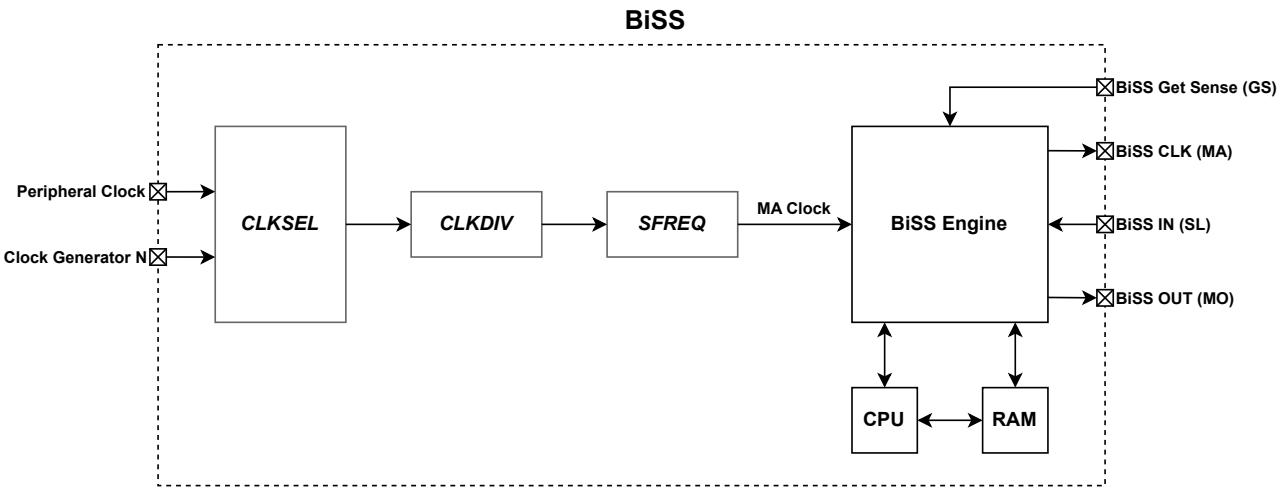
Table 22-2. CLKSEL Selection bit

Value	Description
1	CLKGEN 11
0	Standard Speed Peripheral Clock

22.2 Architectural Overview

BiSS is a four channel serial interface. Figure 22-1 provides a schematic of the BiSS connection between a dsPIC33A device and two BiSS sensors.

Figure 22-1. BiSS High-Level Block Diagram



Note: Refer to 22.8. Terminology for signal abbreviations in BiSS module figures.

The devices in the BiSS system can operate in the following relationships:

- Point-to-Point – Single Client
- Point-to-Point – Multiple Clients
- Bus Configuration – Multiple Clients

Figure 22-2. Point-to-Point – Single Client

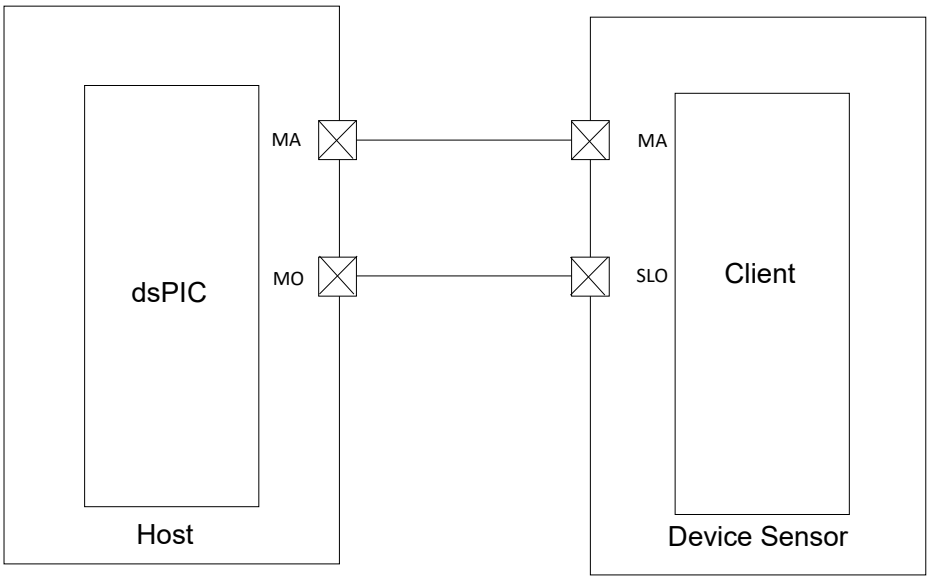


Figure 22-3. Point to Point – Multiple Clients

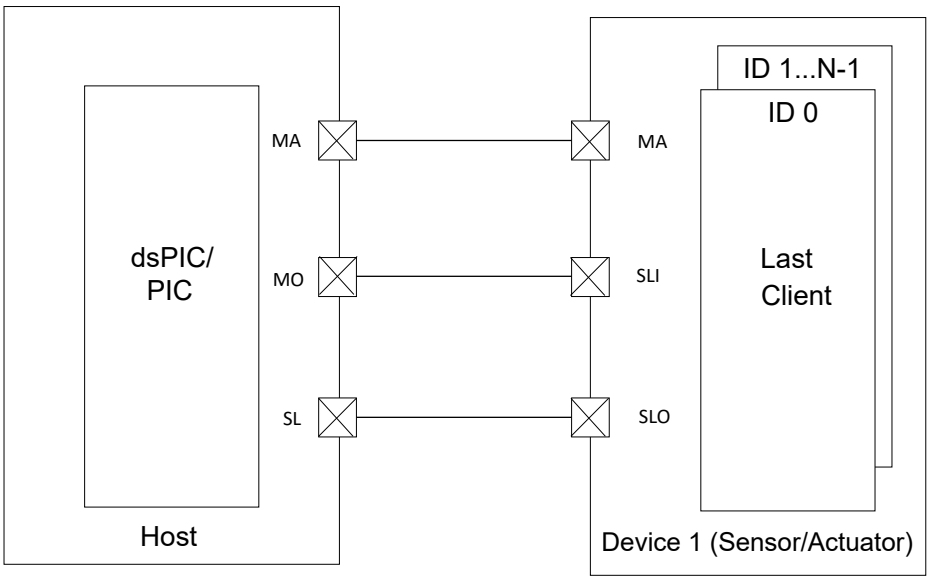
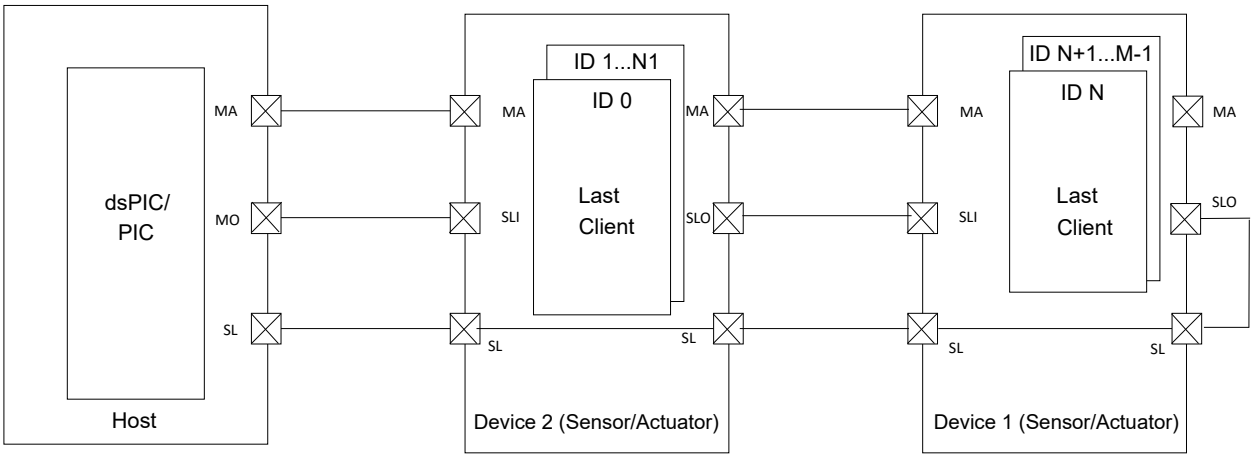


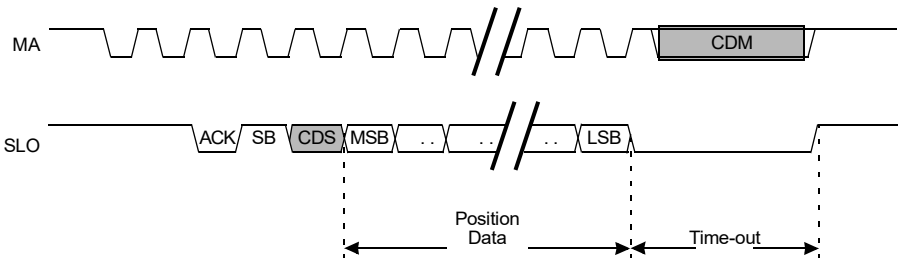
Figure 22-4. Bus Configuration – Multiple Clients



22.2.1 BiSS Frame

In the following point-to-point configuration, a simple example of a BiSS frame is shown. Figure 22-5 shows the clock generated on MA and transmitted to the client. In this case, the client is a sensor, and a packet of data is sent from the sensor or client back to the host.

Figure 22-5. Protocol Format



Note: See 22.2.4. Bus Protocol Details for more information regarding the BiSS frame label definitions.

22.2.2 BiSS Supported Modes

The BiSS module supports BiSS/BiSS-C and SSI modes. There are a few differences between BiSS and SSI, but the main considerations are speed and whether control communication is needed. BiSS can operate at a maximum of 10 MHz, while SSI can operate at a maximum of 1.5 MHz. BiSS can communicate data every cycle and can also communicate control data or commands. SSI only communicates data each cycle. Setting the CFGCHx bits in the BiCHCON register for each channel configures the channel in the desired BiSS operating mode.

22.2.3 BiSS Data Types

There are two major types of transmission types, or packets, that are used in BiSS. These transmission types can be separated into Single Cycle Data and Control Data. The Single Cycle Data and Control Data form the complete BiSS data packet.

22.2.3.1 Single Cycle Data (SCD)

The Single Cycle Data is the primary data and is newly generated and completely transmitted in each cycle. This data channel is primarily used for fast and cyclical sensor or actuator data. Since the data is cyclical, meaning it is based on repeated intervals of time, rather than based on specific

state conditions, the data is predictable and able to be operated at high speeds. Sensor data can be considered a data transmission from a client to a host. Actuator data can be considered as a data transmission from a host to a client.

22.2.3.2 Single Cycle Data (SCD) Gray Code

Normally, BiSS uses a binary encoder and sends an updated complete binary value for each SCD. If desired, in SSI mode, a gray code encoder can be enabled. Enabling the gray code encoder allows only a single bit to be changed from the previous SCD, allowing for less errors during data transmission. The gray code encoder is enabled by setting the GRAY bit in the respective BiCLTCONx register.

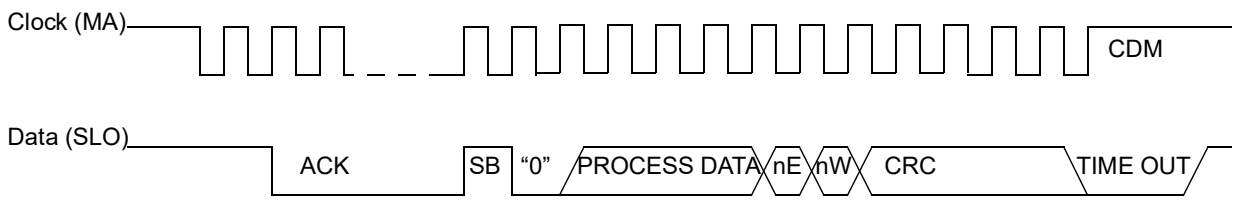
22.2.3.3 Control Data (CD)

The Control Data has two different types: Control Data Host (CDM) and Control Data Client (CDS). The Control Data Host is transmitted one bit per BiSS frame, or cycle, on the MA line during time-out on the SLO line. The Control Data Client is transmitted on the SLO line one bit per BiSS frame, or cycle, after the Start bit but before the data is transmitted. After multiple BiSS frames or cycles, an entire control frame is constructed from the individual CDM and CDS bits that were sent per cycle.

22.2.4 Bus Protocol Details

Figure 22-6 shows a simple example of the BiSS module operating in point-to-point configuration to illustrate the different components that combine to form a complete BiSS frame.

Figure 22-6. Example of BiSS Protocol



Note: Clock and data are both active high.

22.2.4.1 Acknowledge (ACK)

When a host or client device in the serial chain sends a Start bit, the client must provide an Acknowledge of the Start. If multiple clients are in a chain and connected to the host, the host sends a Start, the last client in the chain sends an Acknowledge and a Start and then the next client sends an Acknowledge and Start; this continues until the first client sends an Acknowledge and a Start.

22.2.4.2 Start bit (SB)

The Start bit signifies that a CDS bit, followed by a data transfer, is about to occur.

22.2.4.3 Control Data Client (CDS)

The CDS bit is the first bit after the Start bit and “0” is always the state transmitted by the host. This bit is used by clients for ID confirmation and replies to host control data. This bit can be "1" or "0" when transmitted by the client. The eventual combination from each cycle’s CDS bits creates an entire control communication packet. Please see the 22.4.3.1. Register Communication section for more information.

22.2.4.4 Process Data

Process data is the data from the client’s sensors/actuators that is transmitted to or from the host. Bit length is determined by the number of clients in the system. For a single client, this is equivalent to the Single Cycle Data (SCD); this is the cycle-to-cycle data that is transmitted every cycle. An example of data that is transmitted every cycle could be the encoded position of a sensor.

22.2.4.5 Error bit (nE)

An Error bit is an optional error detection bit that can be enabled. This bit is inverted and transmitted after the main data packet.

The Error bit and 22.2.4.6. Warning bit (nW) allow for precautionary measures during every cycle. This allows the host and clients to respond to an error or warning during every cycle instead of waiting for any control data.

22.2.4.6 Warning bit (nW)

The Warning bit is an optional bit that can be enabled. Similar to the 22.2.4.5. Error bit (nE), this bit is also inverted and transmitted after the Error bit.

22.2.4.7 Cyclic Redundancy Check (CRC)

If enabled, after the nW bit, the CRC for data can be transmitted. The CRC is used to provide additional safety measures. More details can be found in 22.4.6. CRC.

22.2.4.8 Time Out

Once all data has been sent in addition to the optional bits, there should be a time-out on the SLO line while the host transmits the CDM bit. In BiSS time-out, no further clock pulses are sent to the MA by the host. The inverse state of the MA line during the BiSS time-out is the state of the CDM bit. At the end of the data transmission, the host sets its output MO to the Idle state '1'. The clients then pass on this '1' received at SLI to their output SLO as soon as they have detected the expiration of the time-out. This ensures the BiSS time-out on the line SL is only signaled to the host when all connected clients have detected the time-out. When the BiSS time-out expires, all clients return to the Idle state; all lines are set to the high signal level ('1') in the process."

22.2.4.9 Control Data Main (CDM)

The CDM bit for control communication is the inverted state of the MA clock line during the BiSS time-out. Each cycle-to-cycle CDM bit eventually combines together to create an entire CDM control packet; this is how the host communicates control information to clients.

22.2.5 dsPIC Setup

The following conditions are required to start the BiSS module:

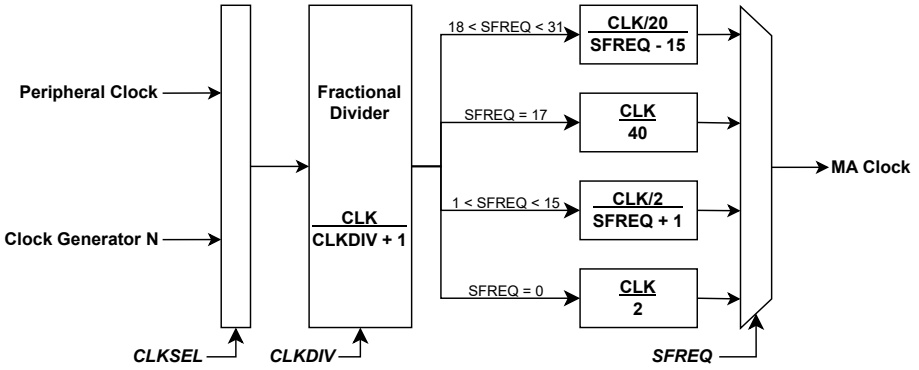
- I/O pins and PPS must be set
- Interrupts must be accounted for
- Communication frequencies must be specified

Once these conditions are met, the data transmission needs to be configured.

22.2.5.1 Clocking and Baud Rate Configuration

The BiSS module supports multiple clock sources to drive the single cycle data frequency. The clock tree with relevant registers is shown in Figure 22-7. The code is equivalent to the decimal number that the register is set.

Figure 22-7. Clock Tree



22.2.5.1.1 Clock Frequency Generation

The BiSS module supports multiple clock sources that can be selected using CLKSEL register bits. After the clock source is selected, the first clock divider can be set up by setting the CLKDIV register. The BiSS module is expected to receive a maximum of 20 MHz CLK from the CLKDIV block. Select the CLKDIV value so that the CLK value is not more than 20 MHz. Setting the SFREQ register configures the SCD frequency (MA Clock). With a module divided clock frequency of 20 MHz, the clock frequency at MA ranges from 10 MHz down to 62.5 kHz.

22.2.5.1.2 Protocol Interface Pins

MA is an output from the host and represents the host clock that will be transmitted to all clients simultaneously. MA is also the line where the control data from the host is transmitted.

MO is the host data output and is the line where the host will transmit actuator data (host to client operation).

SL or SLO (depending on SSI or BiSS mode) is the client return (SSI) or the client output (BiSS). SL/SLO is the line in which sensor data will be transmitted from the sensors to the host (client to host operation).

22.2.5.1.3 BiSS I/O Pins

Three pins are used for the bus operation: MA (Host Clock), MO (Host Output Data) and SL (Return Data).

22.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1F00	B1SCDATA0L	31:24	SCDATA0L[31:24]							
		23:16	SCDATA0L[23:16]							
		15:8	SCDATA0L[15:8]							
		7:0	SCDATA0L[7:0]							
0x1F04	B1SCDATA0H	31:24	SCDATA0H[31:24]							
		23:16	SCDATA0H[23:16]							
		15:8	SCDATA0H[15:8]							
		7:0	SCDATA0H[7:0]							
0x1F08	B1SCDATA1L	31:24	SCDATA1L[31:24]							
		23:16	SCDATA1L[23:16]							
		15:8	SCDATA1L[15:8]							
		7:0	SCDATA1L[7:0]							
0x1F0C	B1SCDATA1H	31:24	SCDATA1H[31:24]							
		23:16	SCDATA1H[23:16]							
		15:8	SCDATA1H[15:8]							
		7:0	SCDATA1H[7:0]							
0x1F10	B1SCDATA2L	31:24	SCDATA2L[31:24]							
		23:16	SCDATA2L[23:16]							
		15:8	SCDATA2L[15:8]							
		7:0	SCDATA2L[7:0]							
0x1F14	B1SCDATA2H	31:24	SCDATA2H[31:24]							
		23:16	SCDATA2H[23:16]							
		15:8	SCDATA2H[15:8]							
		7:0	SCDATA2H[7:0]							
0x1F18	B1SCDATA3L	31:24	SCDATA3L[31:24]							
		23:16	SCDATA3L[23:16]							
		15:8	SCDATA3L[15:8]							
		7:0	SCDATA3L[7:0]							
0x1F1C	B1SCDATA3H	31:24	SCDATA3H[31:24]							
		23:16	SCDATA3H[23:16]							
		15:8	SCDATA3H[15:8]							
		7:0	SCDATA3H[7:0]							
0x1F20 ... 0x1F7F	Reserved									
0x1F80	B1RDATA0	31:24	RDATAy[23:16]							
		23:16	RDATAy[15:8]							
		15:8	RDATAy[7:0]							
		7:0	RDATAy/IDS0[7:0]							
0x1F84	B1RDATA1	31:24	RDATAy[23:16]							
		23:16	RDATAy[15:8]							
		15:8	RDATAy[7:0]							
		7:0	RDATAy/IDS1[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1F88	B1RDATA2	31:24	RDATAy[23:16]							
		23:16	RDATAy[15:8]							
		15:8	RDATAy[7:0]							
		7:0	RDATAy/IDS2[7:0]							
0x1F8C	B1RDATA3	31:24	RDATAy[23:16]							
		23:16	RDATAy[15:8]							
		15:8	RDATAy[7:0]							
		7:0	RDATAy/IDS3[7:0]							
0x1F90	B1RDATA4	31:24	RDATAy[23:16]							
		23:16	RDATAy[15:8]							
		15:8	RDATAy[7:0]							
		7:0	RDATAy/IDS4[7:0]							
0x1F94	B1RDATA5	31:24	RDATAy[23:16]							
		23:16	RDATAy[15:8]							
		15:8	RDATAy[7:0]							
		7:0	RDATAy/IDS5[7:0]							
0x1F98	B1RDATA6	31:24	RDATAy[23:16]							
		23:16	RDATAy[15:8]							
		15:8	RDATAy[7:0]							
		7:0	RDATAy/IDS6[7:0]							
0x1F9C	B1RDATA7	31:24	RDATAy[23:16]							
		23:16	RDATAy[15:8]							
		15:8	RDATAy[7:0]							
		7:0	RDATAy/IDS7[7:0]							
0x1FA0 ... 0x1FBF	Reserved									
0x1FC0	B1CLTCON0	31:24	CRCSEED0[15:8]							
		23:16	CRCSEED0[7:0]							
		15:8	CRCSEL0	CRCLEN0[6:0]/CRCPOLY0[6:0]						
		7:0	GRAY0/LSTOP0	SCDEN0	SCDLEN0[5:0]					
0x1FC4	B1CLTCON1	31:24	CRCSEED1[15:8]							
		23:16	CRCSEED1[7:0]							
		15:8	CRCSEL1	CRCLEN1[6:0]/CRCPOLY1[6:0]						
		7:0	GRAY1/LSTOP1	SCDEN1	SCDLEN1[5:0]					
0x1FC8	B1CLTCON2	31:24	CRCSEED2[15:8]							
		23:16	CRCSEED2[7:0]							
		15:8	CRCSEL2	CRCLEN2[6:0]/CRCPOLY2[6:0]						
		7:0	GRAY2/LSTOP2	SCDEN2	SCDLEN2[5:0]					
0x1FCC	B1CLTCON3	31:24	CRCSEED3[15:8]							
		23:16	CRCSEED3[7:0]							
		15:8	CRCSEL3	CRCLEN3[6:0]/CRCPOLY3[6:0]						
		7:0	GRAY3/LSTOP3	SCDEN3	SCDLEN3[5:0]					

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1FD0 ... 0x1FDF	Reserved									
0x1FE0	B1RCCON	31:24			REGNUM[5:0]					
		23:16	WNR	STRTADDR[6:0]						
		15:8								
		7:0								
0x1FE4	B1CTRLCON	31:24							NOCRC	BANKSWEN
		23:16	Reserved[2:0]			SFREQ[4:0]				
		15:8	CTS	PROTOSEL	CLNTID[2:0]				MOEN	HOLDCDM
		7:0	CHSEL[7:0]							
0x1FE8	B1CCON	31:24								
		23:16								
		15:8	MODELAY[7:0]							
		7:0	FREQAGS[7:0]							
0x1FEC	B1CHCON	31:24	ACTSEN[7:0]							
		23:16	CLCHCFG7[1:0]		CLCHCFG6[1:0]		CLCHCFG5[1:0]		CLCHCFG4[1:0]	
		15:8	CLCHCFG3[1:0]		CLCHCFG2[1:0]		CLCHCFG1[1:0]		CLCHCFG0[1:0]	
		7:0	CLNTLOC[7:2]							CLNTLOC1
0x1FF0	B1STAT	31:24	CDMTO	CDSSEL	REGBYTESV[5:0]					
		23:16	CLSCDV7		CLSCDV6		CLSCDV5		CLSCDV4	
		15:8	CLSCDV3		CLSCDV2		CLSCDV1		CLSCDV0	
		7:0	ERR	AGSERR	DLYERR	SCDTXERR	REGERR		REGEND	EOT
0x1FF4	B1INSTR	31:24								
		23:16								
		15:8	Reserved[3:0]							
		7:0	BREAK	BNKLOCK	SWBANK	INIT	INSTR[2:0]			AGS
0x1FF8	B1CHSTAT	31:24								BKSWERR
		23:16								
		15:8	CDS7	SL7	CDS6	SL6	CDS5	SL5	CDS4	SL4
		7:0	CDS3	SL3	CDS2	SL2	CDS1	SL1	CDS0	
0x1FFC	B1CON	31:24	INSTRWA	INSTRWE	REGAE			TXRDEN	SCDRST	REGRST
		23:16	CLKDIV[7:0]							
		15:8	ON		SLPEN	SIDL	CDM	CDS	SENSESEL	GETSENSE
		7:0	DISSI	DISMA	DISSO	REGACC	BNKNUM	ACTIVE		CLKSEL

22.3.1 BiSS Single Cycle Data Register

Name: B1SCDATAxL
Offset: 0x1F00, 0x1F08, 0x1F10, 0x1F18

Note:

1. x = Client Number, 0-3, each client has maximum of 64-bits

Bit	31	30	29	28	27	26	25	24
	SCDATAxL[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SCDATAxL[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SCDATAxL[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SCDATAxL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – SCDATAxL[31:0] Single Cycle Data for Sensor/Actuator Data(1)

22.3.2 BiSS Single Cycle Data Register

Name: B1SCDATAxH
Offset: 0x1F04, 0x1F0C, 0x1F14, 0x1F1C

Note:

1. x = Client Number, 0-3, each client has maximum of 64-bits

Bit	31	30	29	28	27	26	25	24
	SCDATAxH[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SCDATAxH[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SCDATAxH[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SCDATAxH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – SCDATAxH[31:0] Single Cycle Data for Sensor/Actuator Data

22.3.3 BiSS 1 Register Data Register

Name: B1RDATAx
Offset: 0x1F80, 0x1F84, 0x1F88, 0x1F8C, 0x1F90, 0x1F94, 0x1F98, 0x1F9C

Notes:

1. x = BiSS Register Number or IDS number, 0-7
2. y = BiSS Register Data Number, 0-31

Bit	31	30	29	28	27	26	25	24
	RDATAy[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RDATAy[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RDATAy[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RDATAy/IDSx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:8 – RDATAy[23:0] Register Data

Bits 7:0 – RDATAy/IDSx[7:0] Register Data and IDSx
Bits 7-0 are shared by RDATAy and IDSx

22.3.4 BiSS Client Configuration Register

Name: B1CLTCONx
Offset: 0x1FC0, 0x1FC4, 0x1FC8, 0x1FCC

Note:
1. x = Client Number, 0 to 3

Bit	31	30	29	28	27	26	25	24
	CRCSEEDx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CRCSEEDx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCSELx	CRCLENx[6:0]/CRCPOLYx[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	GRAYx/ LSTOPx	SCDENx	SCDLENx[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CRCSEEDx[15:0]
Polynomial SCD CRC Calculation Start Value bit

Bit 15 – CRCSELx Select CRC bit

Value	Description
1	CRC polynomial(7:1) in CRCPOLYx
0	CRC bit length in CRCLENx will apply dedicated CRC polynomials

Bits 14:8 – CRCLENx[6:0]/CRCPOLYx[6:0] Polynomial Selection by Length for SCD CRC Check bit/Polynomial for SCD CRC Check bits
CRCLENx and CRCPOLYx share the same register location.

Value	Description
	When CRCLENx is selected:
0010000	CRC polynomial 0b1.1001.0000.1101.1001 = 0x190D9
0001000	CRC polynomial 0b1.0010.1111 = 0x12F
0000111	CRC polynomial 0b1000.1001 = 0x89
0000110	CRC polynomial 0b100.0011 = 0x43
0000101	CRC polynomial 0b10.0101 = 0x25
0000100	CRC polynomial 0b1.0011 = 0x13
0000011	CRC polynomial 0b1011 = 0xB
0000000	CRC for single cycle data not present, CRC verification deactivated, CRCSELx = 0b0
	When CRCPOLYx is selected:
11111111–0000001	CRC polynomial for single cycle data

Value	Description
0000000	CRC polynomial 0x00 not applicable with CRCSELx = 0b1

Bit 7 – GRAYx/LSTOPx Enable SCD Gray to Binary Conversion bit (SSI only)/Leading STOP Bit on Single Cycle bit (actuator data only)

GRAYx and LSTOPx share the same register location.

Value	Description
	When GRAYx is selected
1	SSI single cycle data gray coded
0	SSI single cycle data binary coded
	When LSTOPx is selected:
1	Leading STOP Bit on Single Cycle Actuator Data (Do not send STOP bit before Actuator data
0	No Leading STOP Bit on Single Cycle Actuator Data (Send STOP bit before Actuator data)

Bit 6 – SCDENx Enable Single Cycle Data bit

Value	Description
1	Single cycle data available
0	Single cycle data not available

Bits 5:0 – SCDLENx[5:0] Single Cycle Data Length bit

Value	Description
111111	Single cycle data length = 64
...	
000001	Single cycle data length = 2
000000	Single cycle data length = 1

22.3.5 BiSS Register Communication Configuration Register

Name: B1RCCON
Offset: 0x1FE0

Bit	31	30	29	28	27	26	25	24
			REGNUM[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	WNR	STRTADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 29:24 – REGNUM[5:0] Register Quantity of Access bits

Bit 23 – WNR Write/Read Register Selector bit

Value	Description
1	Write register data
0	Read register data

Bits 22:16 – STRTADDR[6:0] Register Access Start Address bits

22.3.6 BiSS Control Communication Configuration Register

Name: B1CTRLCON
Offset: 0x1FE4

Bit	31	30	29	28	27	26	25	24
							NOCRC	BANKSWEN
Access							R/W	R/W
Reset							0	0

Bit	23	22	21	20	19	18	17	16
	Reserved[2:0]			SFREQ[4:0]				
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CTS	PROTOSEL	CLNTID[2:0]				MOEN	HOLDCDM
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	CHSEL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 25 – NOCRC CRC for SCD Not to be Stored in RAM bit

Value	Description
1	All client CRC of SCD not stored in RAM
0	CRC of SCD is stored in RAM (only applicable with active CRC verification and CRC polynomial > 0)

Bit 24 – BANKSWEN Single RAM Bank Enabled bit

Value	Description
1	One RAM bank is used for SCD
0	Two RAM banks are used for SCD

Bits 23:21 – Reserved[2:0] Read as '0'

Bits 20:16 – SFREQ[4:0] Sensor Data Clock Frequency (FSCD) bit
With CLK = 20 MHz, clock frequencies ranging from 62.5 kHz to 10 MHz can be selected for sensor data transmission.

Value	Description
11111 – 10010	CLK / 20 / (Value - 15)
10001	CLK / 40
10000	Not permitted
01111 – 00001	CLK / 2 / (Value + 1)
00000	CLK/2

Bit 15 – CTS Register Transmission or Instruction Selector bit

Value	Description
1	Register communication

Value	Description
0	Command/instruction communication

Bit 14 – PROTOSEL Register Access Protocol Selection A/B or C Selector bit

Value	Description
1	Register communication BiSS C
0	Reserved

Bits 13:12 – CMD[1:0] Command/Instruction bit

Bits 13:11 – CLNTID[2:0] Client Selector bit - client ID of accessed client

Bit 11 – IDADIS ID-Acknowledge Disable bit

Value	Description
1	Immediate execution
0	The client's feedback (IDA) is tested before execution (EX bit after IDA)

Bit 9 – MOEN MO Enable - Enable Output at MOx for Actuator Data or Delayed Start bit

Value	Description
1	Parameterized processing time by host when MO signal active (length = MODELAY)
0	MO forced to low

Bit 8 – HOLDCDM Hold CDM (Control Data Host) - Length of CDM bit

Value	Description
1	Clock line consistent with CDM bit until start of next cycle
0	Clock line high at end of cycle

Bits 7:0 – CHSEL[7:0] Channel Selection bit

Value	Description
1	Channel N used for communication
0	Channel N not used for communication

22.3.7 BiSS Communication Configuration Register

Name: B1CCON
Offset: 0x1FE8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	MODELAY[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FREQAGS[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – MODELAY[7:0] Delay of Start Bit at Output MOx bits

Value	Description
11111111-00000001	Value * 1 /FSCD
00000000	No start delay

Bits 7:0 – FREQAGS[7:0] AutoGetSense Frequency - Controls Automatic SCD Timing bits

Value	Description
0x80-0xFF	CLK/ (625 * (Value+1)
0x7D	AGSINFINITE - Requires trigger event to start next SCD cycle, such as GETSENSE or INSTR
0x7C	AGSMIN - Host automatically restarts the next SCD cycle after the prior finishes
0x00-0x7B	CLK/ (20 * (FREQAGS(6:0)+1))

22.3.8 BiSS Channel Configuration Register

Name: B1CHCON
Offset: 0x1FEC

Bit	31	30	29	28	27	26	25	24
	ACTSEN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CLCHCFG7[1:0]		CLCHCFG6[1:0]		CLCHCFG5[1:0]		CLCHCFG4[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CLCHCFG3[1:0]		CLCHCFG2[1:0]		CLCHCFG1[1:0]		CLCHCFG0[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLNTLOC[7:2]							CLNTLOC1
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	0	1

Bits 31:24 – ACTSEN[7:0] Sensor or Actuator Data Selector

Value	Description
1	Actuator data client 1...8
0	Sensor data client 1...8

Bits 23:22 – CLCHCFG7[1:0] Channel 7 Configuration bits

Value	Description
11	Channel is not used
10	SSI
01	BiSS C
00	Reserved

Bits 21:20 – CLCHCFG6[1:0] Channel 6 Configuration bits

Value	Description
11	Channel is not used
10	SSI
01	BiSS C
00	Reserved

Bits 19:18 – CLCHCFG5[1:0] Channel 5 Configuration bits

Value	Description
11	Channel is not used
10	SSI
01	BiSS C
00	Reserved

Bits 17:16 – CLCHCFG4[1:0] Channel 4 Configuration bits

Value	Description
11	Channel is not used
10	SSI
01	BiSS C
00	Reserved

Bits 15:14 – CLCHCFG3[1:0] Channel 3 Configuration bits

Value	Description
11	Channel is not used
10	SSI
01	BiSS C
00	Reserved

Bits 13:12 – CLCHCFG2[1:0] Channel 2 Configuration bits

Value	Description
11	Channel is not used
10	SSI
01	BiSS C
00	Reserved

Bits 11:10 – CLCHCFG1[1:0] Channel 1 Configuration bits

Value	Description
11	Channel is not used
10	SSI
01	BiSS C
00	Reserved

Bits 9:8 – CLCHCFG0[1:0] Channel 0 Configuration bits

Value	Description
11	Channel is not used
10	SSI
01	BiSS C
00	Reserved

Bits 7:1 – CLNTLOC[7:2] Client Location bits

Value	Description
1	Following clients are assigned to the next channel
0	Clients are assigned to this channel

Bit 0 – CLNTLOC1 Client Location bit

Value	Description
1	Following clients are assigned to the next channel
0	Clients are assigned to this channel

22.3.9 BiSS Communication Status Register

Name: B1STAT
Offset: 0x1FF0

Bit	31	30	29	28	27	26	25	24
	CDMTO	CDSSEL	REGBYTESV[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CLSCDV7		CLSCDV6		CLSCDV5		CLSCDV4	
Access	R/W		R/W		R/W		R/W	
Reset	0		0		0		0	

Bit	15	14	13	12	11	10	9	8
	CLSCDV3		CLSCDV2		CLSCDV1		CLSCDV0	
Access	R/W		R/W		R/W		R/W	
Reset	0		0		0		0	

Bit	7	6	5	4	3	2	1	0
	ERR	AGSERR	DLYERR	SCDTXERR	REGERR		REGEND	EOT
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit 31 – CDMTO CDM Timeout Reached bit

Value	Description
1	CDMTIMEOUT reached
0	CDMTIMEOUT not reached

Bit 30 – CDSSEL Selected Channel CDS bit

Value	Description
1	CDS of the selected channel = 1
0	CDS of the selected channel = 0

Bits 29:24 – REGBYTESV[5:0] Valid Register Data Transmitted Before Error bits

Value	Description
0x01-0x3F	After transfer, number of successfully transferred registers before register communication error
0x00	After transfer, no register communication error

Bit 23 – CLSCDV7 Client 7 Single Cycle Data Valid bit

Value	Description
1	Single cycle data valid
0	Single cycle data invalid

Bit 21 – CLSCDV6 Client 6 Single Cycle Data Valid bit

Value	Description
1	Single cycle data valid
0	Single cycle data invalid

Bit 19 – CLSCDV5 Client 5 Single Cycle Data Valid bit

Value	Description
1	Single cycle data valid
0	Single cycle data invalid

Bit 17 – CLSCDV4 Client 4 Single Cycle Data Valid bit

Value	Description
1	Single cycle data valid
0	Single cycle data invalid

Bit 15 – CLSCDV3 Client 3 Single Cycle Data Valid bit

Value	Description
1	Single cycle data valid
0	Single cycle data invalid

Bit 13 – CLSCDV2 Client 2 Single Cycle Data Valid bit

Value	Description
1	Single cycle data valid
0	Single cycle data invalid

Bit 11 – CLSCDV1 Client 1 Single Cycle Data Valid bit

Value	Description
1	Single cycle data valid
0	Single cycle data invalid

Bit 9 – CLSCDV0 Client 0 Single Cycle Data Valid bit

Value	Description
1	Single cycle data valid
0	Single cycle data invalid

Bit 7 – ERR Transmission Error bit

Value	Description
1	Error
0	No Error

Bit 6 – AGSERR AGS Error - Unable to Start SCD Frame bit

Value	Description
1	AGS error
0	No AGS error

Bit 5 – DLYERR Delay Error bit - Missing Start bit During Register Communication

Value	Description
1	Delay error
0	No Delay error

Bit 4 – SCDTXERR SCD Transmission Error bit

Value	Description
1	Error in last single cycle data transmission
0	No Error in last single cycle data transmission

Bit 3 – REGERR Register Communication Error bit

Value	Description
1	Error in last register data transmission
0	No Error in last register data transmission

Bit 1 – REGEND End Of Register Communication Error bit

Value	Description
1	Register data transmission completed
0	No valid register data available

Bit 0 – EOT End Of Transmission bit

Value	Description
1	Data transmission finished
0	Data transmission active

22.3.10 BiSS Instruction Register

Name: B1INSTR
Offset: 0x1FF4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	Reserved[3:0]							
Access	R	R	R	R				
Reset	0	0	0	0				

Bit	7	6	5	4	3	2	1	0
	BREAK	BNKLOCK	SWBANK	INIT	INSTR[2:0]			AGS
Access	R/W	R/W	R/W	R/W	R/W			R/W
Reset	0	0	0	0	0			0

Bits 15:12 – Reserved[3:0] Read as '0'

Bit 7 – BREAK Data Transmission Interrupt bit

Value	Description
1	Abort data transmission
0	No change

Bit 6 – BNKLOCK Inhibit RAM Bank Switching bit

Value	Description
1	Bank switching lock
0	No bank switching lock

Bit 5 – SWBANK Switch RAM Bank bit

Value	Description
1	RAM banks are switched
0	RAM banks are not switched

Bit 4 – INIT Start INIT Sequence bit

Value	Description
1	Initialize data channel
0	No changes on the data channel

Bits 5:3 – INSTR[2:0] SCD Control Instruction bit

Bit 0 – AGS Automatic Get Sensor Data bit

Value	Description
1	Automatic data transmission
0	No automatic data transmission

22.3.11 BiSS Channel Status Register

Name: B1CHSTAT
Offset: 0x1FF8

Bit	31	30	29	28	27	26	25	24
								BKSWERR
Access								R
Reset								0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CDS7	SL7	CDS6	SL6	CDS5	SL5	CDS4	SL4
Access	R	R	R	R	R	R	R	R
Reset	0	x	0	x	0	x	0	x

Bit	7	6	5	4	3	2	1	0
	CDS3	SL3	CDS2	SL2	CDS1	SL1	CDS0	
Access	R	R	R	R	R	R	R	
Reset	0	x	0	x	0	x	0	

Bit 24 – BKSWERR Switch Bank Failed bit

Value	Description
1	Bank switching (SCD) not successful
0	Bank switching (SCD) successful

Bit 15 – CDS7 Channel 7 CDS bit

Value	Description
1	CDS7 = 1
0	CDS7 = 0

Bit 14 – SL7 SL7 Input Line State bit

Value	Description
1	SL7 line level high
0	SL7 line level low

Bit 13 – CDS6 Channel 6 CDS bit

Value	Description
1	CDS6 = 1
0	CDS6 = 0

Bit 12 – SL6 SL6 Input Line State bit

Value	Description
1	SL6 line level high
0	SL6 line level low

Bit 11 – CDS5 Channel 5 CDS bit

Value	Description
1	CDS5 = 1
0	CDS5 = 0

Bit 10 – SL5 SL5 Input Line State bit

Value	Description
1	SL5 line level high
0	SL5 line level low

Bit 9 – CDS4 Channel 4 CDS bit

Value	Description
1	CDS4 = 1
0	CDS4 = 0

Bit 8 – SL4 SL4 Input Line State bit

Value	Description
1	SL4 line level high
0	SL4 line level low

Bit 7 – CDS3 Channel 3 CDS bit

Value	Description
1	CDS3 = 1
0	CDS3 = 0

Bit 6 – SL3 SL3 Input Line State bit

Value	Description
1	SL3 line level high
0	SL3 line level low

Bit 5 – CDS2 Channel 2 CDS bit

Value	Description
1	CDS2 = 1
0	CDS2 = 0

Bit 4 – SL2 SL2 Input Line State bit

Value	Description
1	SL2 line level high
0	SL2 line level low

Bit 3 – CDS1 Channel1 CDS bit

Value	Description
1	CDS1 = 1
0	CDS1 = 0

Bit 2 – SL1 SL1 Input Line State bit

Value	Description
1	SL1 line level high
0	SL1 line level low

Bit 1 – CDS0 Channel 0 CDS bit

Value	Description
1	CDS0 = 1
0	CDS0 = 0

Bit 1 – SL0 SL0 Input Line State bit

Value	Description
1	SL0 line level high
0	SL0 line level low

22.3.12 BiSS Configuration Register

Name: B1CON
Offset: 0x1FFC

Notes:

1. It is not recommended to set "00" for CLKDIV bits.
2. Input to the CLKDIV depends on the CLKSEL value.
3. The clock frequency (CLK) derived from CLKDIV must be maximum of 20 MHz.

Bit	31	30	29	28	27	26	25	24
	INSTRWA	INSTRWE	REGAE			TXRDEN	SCDRST	REGRST
Access	R	R/C	R/C			R/W	R/W	R
Reset	0	0	0			0	0	0

Bit	23	22	21	20	19	18	17	16
	CLKDIV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SLPEN	SIDL	CDM	CDS	SENSESEL	GETSENSE
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	1		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DISSI	DISMA	DISSO	REGACC	BNKNUM	ACTIVE		CLKSEL
Access	R/W	R/W	R/W	R/W	R/W	R/W		R/W
Reset	0	0	0	0	0	0		0

Bit 31 – INSTRWA Instruction Write Active bit

Value	Description
1	Previous write is not complete
0	Previous write is complete

Bit 30 – INSTRWE Instruction Write Error bit

Value	Description
1	Disallowed attempt to write Instruction register (must reset in software)
0	No instruction write error

Bit 29 – REGAE Register Mode Data Access Error bit

Value	Description
1	Disallowed attempt to access Register mode data (must reset in software)
0	No register access error

Bit 26 – TXRDEN Transmit RAM CPU Read Enable bit

Value	Description
1	Transmit RAM is allowed to read by CPU
0	Receive RAM is allowed to read by CPU

Bit 25 – SCDRST SCD RAM Reset bit

Write to '1' to reset the Register communication RAM data buffers.
Reading this bit returns '1' if the Register RAM is in Reset, and '0' if Reset completed. Writing '0' to this bit has to come out of reset when this bit is '1', otherwise writing '0' to this bit has no effect.

Bit 24 – REGRST Register RAM Reset bit

Write to '1' to reset the Register communication RAM data buffers.
Reading this bit returns '1' if the Register RAM is in Reset, and '0' if Reset completed. Writing '0' to this bit has to come out of reset when this bit is '1', otherwise Writing '0' to this bit has no effect.

Bits 23:16 – CLKDIV[7:0] Clock divider bits(1,2,3)

Bit 15 – ON Module Enable bit

Value	Description
1	Module is enabled
0	Module is disabled

Bit 13 – SLPEN Module Sleep Enable bit

Value	Description
1	Module operates in Sleep mode
0	Module disabled in Sleep mode

Bit 12 – SIDL Module Stop in Idle Mode Enable bit

Value	Description
1	Module disabled in Idle mode
0	Module operates in Idle mode

Bit 11 – CDM Control Data bit Host

Value	Description
1	Host sends the control data bit (CDM)
0	Host has not sent the control data bit (CDM)

Bit 10 – CDS Control Data bit Client

Value	Description
1	Client sends the control data bit (CDS)
0	Client has not sent the control data bit (CDS)

Bit 9 – SENSESEL Sense Selection bits

Value	Description
1	Use external Sense
0	Use software Sense

Bit 8 – GETSENSE Software Get-Sense bits

Value	Description
1	Software data transmission triggered
0	Software data transmission is not triggered

Bit 7 – DISSI Disable SI Input Port bit

Value	Description
1	SI pin is not controlled by the BiSS module; pin is controlled by port
0	SI pin is controlled by the BiSS module

Bit 6 – DISMA Disable MA Output Port bit

Value	Description
1	MA pin is not controlled by the BiSS module; pin is controlled by Port
0	MA pin is controlled by the BiSS module

Bit 5 – DISSO Disable SO(MO) Output Port bit

Value	Description
1	MO pin is not controlled by the BiSS module; pin is controlled by Port
0	MO pin is controlled by the BiSS module

Bit 4 – REGACC Register RAM Access Status bit

Value	Description
1	BiSS Protocol Engine is accessing register RAM
0	BiSS Protocol Engine is not accessing register RAM

Bit 3 – BNKNUM SCD RAM Access Status bit

Value	Description
1	BiSS Protocol Engine is accessing RAM1
0	BiSS Protocol Engine is accessing RAM2

Bit 2 – ACTIVE BiSS Active Status bit

Value	Description
1	The BiSS module is active
0	The BiSS module is inactive

Bit 0 – CLKSEL Macro Baud Clock Selection bit
See Table 22-2.

22.4 Operations

One of the advantages of the host using BiSS is that data can be continuously sent from an actuator or a sensor, while also accessing registers and sending commands. With the combination of the two data types, sensor/actuator data and control/register data, it is important to know each separately and how they interact to form a BiSS frame.

22.4.1 General Setup

The following sections describe the setup of the SCD communication and can be used to configure the different communication types. Since each control communication requires an SCD communication, this setup portion is also relevant to the control communication configuration.

22.4.1.1 Peripheral Pin Select (PPS)

This device has PPS to allow virtual mapping pins to any available PPS port. Virtually mapped pins allow a versatile pinout, but must be configured properly before any BiSS communication can occur.

22.4.1.2 Clock and SCD Frequency

The BiSS clock is generated from the clock source selected using CLKSEL bits. Use the CLKDIV register bits to divide this clock further. It is the user's responsibility to select the appropriate value of CLKDIV to generate a 20 MHz protocol clock based on BiSS clock selection.

After the clock division, the clock is fed through the SFREQ divider and becomes the clock at which Single Cycle Data will be transmitted.

22.4.1.3 Getsense Frequency

In addition to setting the SCD frequency, it also necessary to set the desired sample rate for getsense. The SCD frequency is the rate of the individual SCD transmission, but the rate that the

SCD transmissions are triggered is the getsense frequency; this is also the rate that the B1SCDATAxX registers are updated. There are three selections to choose from: AGSMIN, AGSINFINITE and a user value.

22.4.1.3.1 User Value

The user value works like a clock divider and allows the user to set a value for the divider to determine when getsense will trigger.

22.4.1.3.2 AGSMIN

The hexadecimal value equivalent of AGSMIN is 0x7C. This setting allows the BiSS engine to automatically trigger getsense as soon as possible according to the SCD frequency selected.

22.4.1.3.3 AGSINF

The hexadecimal value equivalent of AGSINF is 0x7D. This setting prevents getsense from automatically triggering and is used when the user wants to manually trigger getsense.

22.4.2 Sensor Mode/Actuator Communication

In Sensor mode, the bus is operated like the SSI protocol. The host sends clock pulses, and eventually the data line drops low to Acknowledge and then data transmission occurs. The MA transmits clock pulses that are transmitted to all clients on the bus. The host triggers a start condition and the clients respond and transmit their data.

22.4.2.1 Configuring Single Cycle Data

Client location one is set by default and no further action is needed if a single client is used. Channel one must be configured for the protocol used, which requires the user to use the CLCHCFG bit to select between BiSS C and SSI. At this point, the actuator and sensor selection can be configured using ACTSEN. By default, this is set for sensor data.

Next, the appropriate values must be configured for the SCD and CRC. Depending on what hardware is being used as the client, there are multiple options available for SCD and CRC selections. For this example, consider a common setup where the client expects 28 bits of sensor data to be transmitted, with an error and warning bit, along with the very common 0x43 CRC polynomial which accounts for six bits.

For this BiSS module, the SCD length is considered the 28 bits with two additional bits from the warning and error bits which combined create an SCD of 30. The appropriate bits can now be setup which include, SCDLEN, CRCSEL and CRCLEN; then the module can be enabled by using SCDEN.

22.4.2.2 Reading a Sensor Value

Once all the settings are configured, values from the sensor can be read. The simplest way to read a value is to turn on the BiSS module, set the AGS bit and read from the B1SCDATAxX register. This is the minimum required to read a value, however, the following additional steps are highly recommended:

1. After configuring all settings, turn on the BISS module using the ON bit. This is also a good time to monitor the SLx Status bit. The SLx bit quickly checks the status of the SL line to determine if the client is pulling the line high, which is required for successful BiSS transmission.
   Note: Monitoring the SLx status bit is used to determine if the client is powered on and connected.
2. Initialize the BiSS module using the INIT bit. Initialization has two useful purposes: when the INIT bit is set, the status information is reset to provide assurance that the BiSS module starts off without any erroneous errors. Initialization also sends two pulses that allow the BiSS engine to determine the line delay introduced on MA. This value can immediately be read in the B1SCDATA0L register.
3. After the BiSS module is initialized, poll INSTRWA. It is critical to operations that the instructions are not written to while still in progress. Then the AGS bit can be set to start triggering the getsense to allow the user to read the B1SCDATAxX registers for the sensor values.

22.4.2.3 Reading SCDATA Registers

There are also two suggested methods for reading an SCD register:

- The register can be read directly within the host code from SCDATA, or
- The BiSS interrupt can be configured to trigger at the end of every BiSS transmission

To configure the BiSS interrupt, set the interrupt enable, clear the interrupt flag and confirm there is an interrupt service routine (ISR). In both methods, it is recommended to poll for an end of transmission (_EOT) to verify a successful transmission. If CRC is being used, it is also recommended to poll after the Client SCD valid bit (_CLSCDVx) is set. This bit is set after a successful CRC value is detected. Once these are polled and successfully checked, the B1SCDATA0L and B1SCDATA0H can be read by the user.

22.4.2.4 Transmitting Sensor Data

To summarize the following sections, use the following steps and the 22.5.1. Sensor Communication Code Example.

1. Configure BiSS PPS.
2. Configure SCD frequency and getsense frequency.
3. Select the client location, the protocol used on the channel and determine if the client device is a sensor or an actuator.
4. Configure the SCD length.
5. Configure the CRC used and the CRC length.
6. Turn the module on.
7. Initialize the BiSS module and poll INSTRWA (can read B1SCDATAxX now if line delay calculations are desired later).
8. Turn AGS on.
9. Poll end of transmission, EOT, Status bit and/or CRC Valid bit to confirm a successful transmission.
10. Read value from B1SCDATAxX.

22.4.2.5 Manual Getsense

AGS is an easy way to quickly get sensor data from a device, but sometimes only single getsenses are desired. In this case, enable BISS ON and set the AGS frequency to 0x7D, which is also AGSINF. This forces AGS wait for a manually triggered getsense. To enable a manually triggered getsense, set AGS, and once the manually triggered getsense is desired, the _GETSENSE bit must be set. Setting the _GETSENSE bit triggers the clock to generate enough pulses for SCD and to update B1SCDATAxX. Then, getsense must be cleared by the user to be able to set it again and trigger another getsense.

22.4.3 Control Communication

Control communication is built from multiple SCD communications. Each time an SCD is sent, there are bits that are stored. Once enough bits are stored, then a full control communication is formed. The addition of a control frame allows the ability to read and write to client registers and also send commands to addresses or all clients. To create this control frame, each CDM and CDS bit from the BiSS frames are combined to create a control “packet.” To initiate a control frame, 14 zeroes must be transmitted by the host as the CDM. Because of this, each control communication has SCD communication, which means to setup control communication, the SCD communication still must be set up. In this case, the setup for sensor communication outlined above can be used, and then the control communication for reading and writing to registers can be focused on next.

22.4.3.1 Register Communication

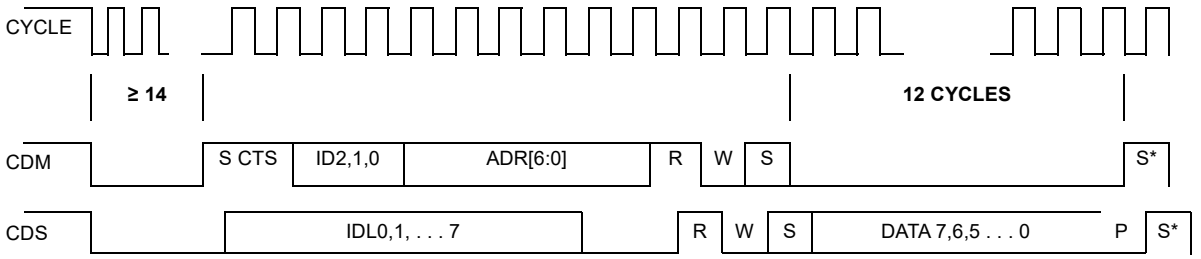
After the SCD/sensor communication has been setup, the additional setup for reading registers can be completed. CTS must be set up before any communication occurs. CTS selects between only

register communication, which is communication between host and client registers, or command communication, which is a bus style communication. The client ID must also be set to make sure the exact device is communicated with; this is done in the CLNTID register.

22.4.3.1.1 Read Register

Adding Register mode to Sensor mode allows SSI to evolve into BiSS. Register mode allows control of the reading and writing of registers by "saving" multiple CDS and CDM bits from multiple BiSS frames and combining them to form a full control frame. Therefore, Register mode always includes Sensor mode. There are different ways to use Register mode, such as reading and writing to registers and sending commands (see Initiating a Register Read).

Figure 22-8. Register Read



Note:

1. The Control Select bit (CTS) is one (CTS =1) for the register communication
2. The secondary register access has only three ID select bits and seven bits of register address
3. * A stop bit (P = 0) is at the end of the frame. Following the stop bit, there can be an optional Start bit if the main must perform a sequential access of additional secondary registers.
4. After the Start bit, the main sends 12 0 bits and one stop bit. The secondary responds with eight bits of data protected with four bits of CRC. After the start bit, the main sends 12 0 bits and one stop bit. The secondary responds with eight bits of data protected with four bits of CRC.

Initiating a Register Read

Like standard SCD communication, INIT should be set, INSTRWA should be polled and AGS should be set.

Before this instruction is set, CDMTO should be polled. This status bit allows the user to poll CDMTO to confirm that the required 14 CDM low bits have been transmitted. Once this is set up, an instruction value can be written into the INSTR register to instruct the BiSS engine the type of instruction/command that will be executed. INSTR should not be written to while it is in progress of fulfilling an instruction, therefore INSTRWA should be polled again.

Communicating Register Read

To summarize the following sections, use the following steps and the 22.5.2. Read Register Code Example.

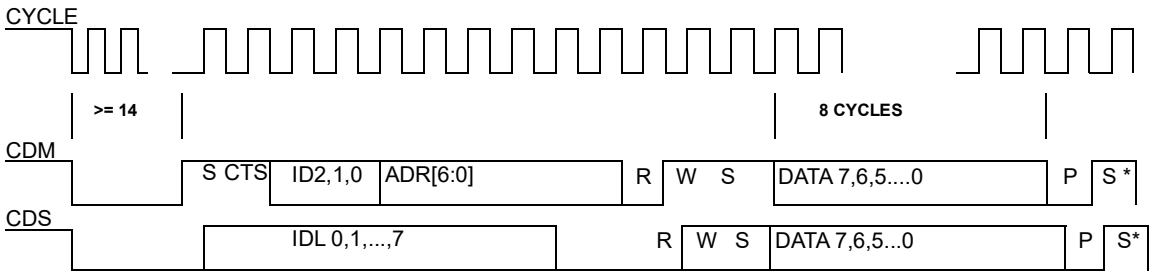
1. Configure BiSS PPS.
2. Configure SCD frequency and getsense frequency.
3. Select the client location, the protocol used on the channel and determine if the client device is a sensor or an actuator.
4. Configure the SCD length.
5. Configure the CRC used and the CRC length.

6. Select a start address to read registers from and how many consecutive bits are desired to be read.
7. Set the read or write bit to read.
8. Set the CTS bit to register communication and the Prototype Select bit to BiSS C.
9. Set the client ID.
10. Turn the module on.
11. Initialize the BiSS module and poll INSTRWA (can read B1SCDATAxX now if line delay compensation is desired)
12. Set AGS to start AutoGetsense
13. Poll CDMTO to confirm 14 CDM low bits have been received.
14. Set the instruction register to the instruction for reading and poll the INSTRWA bit to make sure nothing is written to INSTR while current instruction is in progress.
15. Poll the end of transmission status bit and/or CRC Valid bit and/or Register End bit, REGEND, to confirm that a successful SCD receive and a successful register communication are received.
16. Read values from B1SCDATAxX and B1RDATAx.

22.4.3.1.2 Write Registers

The same setup from the regular single cycle data example can be applied to writing to a BiSS C client's registers. Once set up, the specific registers related to writing to registers can be considered.

Figure 22-9. Register Write



Note:

Note:

1. The Control Select bit (CTS) is one (CTS =1) for the register communication
2. The client register access has only three ID select bits and seven bits of register address
3. * A Stop bit (P = 0) is at the end of the frame. Following the Stop bit, there can be an optional Start bit if the host must perform a sequential access of additional secondary registers.

Additional Setup for Register Write Communication

The TX Read Enable bit must be set to allow the TX buffers to be written to. Otherwise, this bit is cleared and prevents unwanted writes while reading. The Write/Read Register Select bit also needs to be set to determine the direction of communication.

In both read and write, BiSS can consecutively increment and read/write register addresses. To do this, a starting address should be designated using the Start Address bit. In the 22.5.2. Read Register Code Example, the register reading started at 0x00. In this example, the register reading starts at 0x50. Since reading/writing has been determined to occur consecutively in the BiSS module, the user can select how many registers to read/write consecutively by setting the register number register. The register number register is not equivalent to bits. A single register communication for REGNUM purposes is eight bits (see REGNUM). For the same reasons listed in the read register example, the CTS, PROTOSEL and CLNTID registers can all be set now.

REGNUM

For this BiSS engine and communication between registers, the REGNUM register determines that a single register is considered eight bits and cannot be 0. If REGNUM is set to 0, a single REGNUM register will be transmitted, which is equivalent to eight bits.

Initializing a Register Write

To initiate a write, follow the same steps Initiating a Register Read. When initializing a register write, B1RDATAx must be set to the desired value to be written. Also, the INSTR register must be entered with the instruction value for writing instead of reading. The remaining steps are the same as the read example.

Communicating Register Write

To summarize the following sections, use the following steps and the 22.5.3. Write Register Code Example.

1. Configure BiSS PPS
2. Configure SCD frequency and getsense frequency.
3. Select the client location, the protocol used on the channel and determine if the client device is a sensor or an actuator.
4. Configure the SCD length.
5. Configure the CRC used and the CRC length.
6. Select a start address to write registers from and select how many consecutive bits are desired to be written.
7. Set the read or write bit to write. Set the CTS bit to register communication and the Prototype Select bit to BiSS C.
8. Set the client ID.
9. Set TX Read Enable bit to allow writing to the B1SCDATAxX registers.
10. Turn the BiSS module on.
11. Initialize the BiSS module with INIT and poll INSTRWA (can read B1SCDATAxX right now if line delay compensation is desired).
12. Set the AGS bit to start Autogetsense.
13. Set B1RDATAxX to the desired value to be written to the client register.
14. Poll CDMTO to confirm 14 CDM low bits have been received.
15. Set the Instruction register to the instruction for writing and poll the INSTRWA Status bit to make sure nothing is written to INSTR while current instruction is in progress.
16. Poll end of transmission, EOT, Status bit and/or CRC Valid bit and/or Register End bit, REGEND, to confirm a successful SCD receive and a successful register communication has been transmitted.
17. Read value from B1SCDATAxX and B1RDATAx.

22.4.4 Command Communication

The BiSS module allows commands to be sent simultaneously to four clients in any desired combination. This can be achieved by addressing specific clients or broadcasting to all clients on the bus.

Table 22-3. Broadcast Command Options

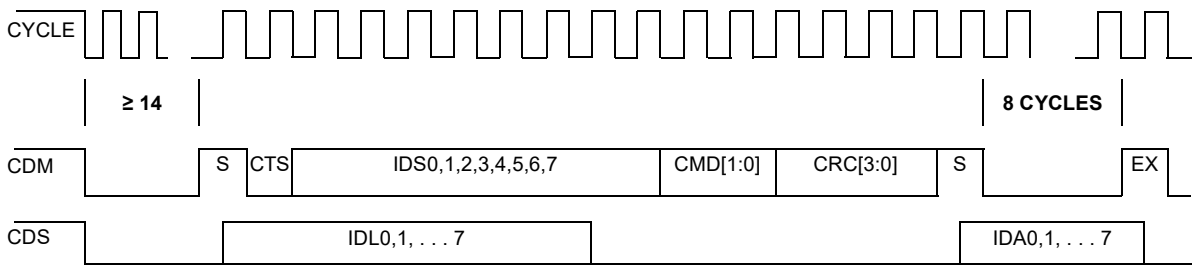
Command	Description
11	Reserved
10	All bus couplers switched to bypass

.....continued	
Command	**Description**
01	Register and command communication is activated
00	Data channels are deactivated

Table 22-4. Addressed Command Options

Command	Description
11	Definable for each secondary
10	The addressed bus couple switches from bypass to line operation
01	Register and command communication of addressed secondary is deactivated
00	Addressed data channel is activated

Figure 22-10. Command Frame



Note:

1. The Control Select bit (CTS) is zero (CTS =0) for the command frame
2. The secondary ID is assigned according to the sequence in the chain
3. To send commands to all secondaries (broadcast) IDS bits= 00000000 (none of the IDS bits is set.)

22.4.4.1 Broadcast Communication

Follow the same setup procedures as described in 22.4.3.1. Register Communication with a few modifications. Set the value of the command desired using the Command bit and changing CTS from '1' for register communication to '0' for command communication, then setting the IDSx to '0'. A '0' in the IDSx register signals a broadcast command that will go to all clients. For an example, see 22.5.4. Broadcast Command Code Example.

22.4.4.2 Addressed Communication

The Addressed Command mode is the same as the Broadcast mode, except instead of '0' in IDSx, an actual address is used. This will send a command to the specific client addressed. For an example, see 22.5.5. Addressed Command Code Example.

22.4.5 Additional BiSS Module Features

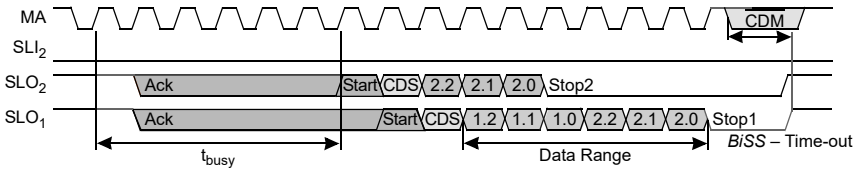
In addition to several communication options, the BiSS module also has options for processing times, line delays and how the memory banks are used.

22.4.5.1 Processing Time Requests

Before sending sensor data, a client can request additional processing time. This request can be used in an instance where the client needs additional time for analog-to-digital conversion or memory access. To request additional processing time, the client can hold the ACK signal low for an additional clock cycle. This, in turn, delays the Start bit, which is the indicator that additional processing time is requested.

If a device in point-to-point configuration consists of several clients, then all but the last client will pass through the data of their predecessor received at SLI and then send this to SLO following its own data. The client with the longest processing time should be used as the last client since it determines the total processing time.

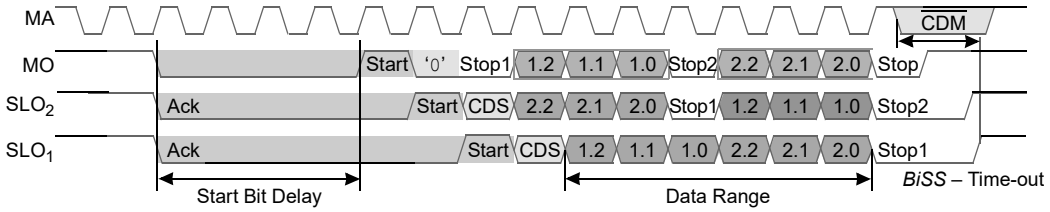
Figure 22-11. Requests for Processing Time (Point-To-Point Configuration)



22.4.5.2 Processing Time Configuration

In the bus configuration, the host delays the output of the Start bit to MO. For this purpose, the host is configured to the maximum delay time of all connected clients during the bus establishment.

Figure 22-12. Configurable Processing Time (Bus Configuration)



22.4.5.3 Line Delay Compensation

Due to various cable lengths expected in BiSS applications, automatic line delay compensation has been included as part of the BiSS engine. Automatic line delay compensation can be calculated as the difference in time from when the client first receives a Clock signal from the host to the time when the host first receives an Acknowledge from the client (see Figure 22-13). The included INIT bit within the BiINSTR register not only resets status registers, but also initiates the line delay compensation by initializing the channel. The INIT bit can be used and then read from the BiSCDATA0L register to read the detected delay value. To see the behavior of the INIT sequence, see Figure 22-14.

Figure 22-13. Line Delay Compensation

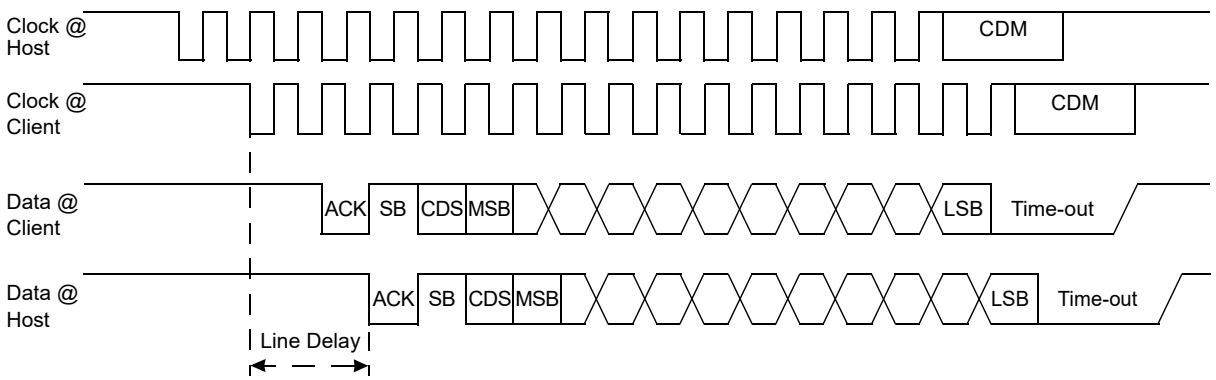
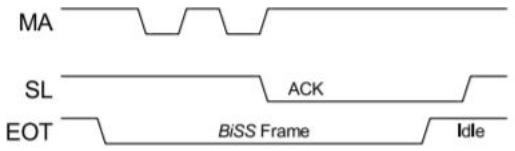


Figure 22-14. BiSS C Initialization Sequence



22.4.5.4 Clock Frequencies for RS422 Interfaces

BiSS is commonly used with RS422 and RS485 interfaces. BiSS is able to match the SSI protocol easily as long as proper lengths of cable and speeds are set. For an approximation of different cable lengths for RS422 interfaces, refer to Table 22-5.

Table 22-5. RS422 Interfaces

Cable Length	Clock Frequency
up to 10m	10 MHz
up to 25m	5 MHz
up to 60m	2 MHz
up to 100m	1 MHz
up to 200m	500 KHz
up to 500m	200 KHz
up to 1000m	100 KHz

Note:
1. Data rate and clock frequency depends on the physical layer used for BiSS communications.

22.4.5.5 Memory Bank Switching

Since BiSS can operate at high speeds of transmission, the BiSS module incorporates memory bank switching. The BiSS module has two memory banks. The status of which memory bank is active can be read by monitoring the BANKSEL bit. It is critical that the registers are not written to during bank switching; therefore, there is also a register access bit to monitor if registers are currently being accessed. By default, bank switching is enabled, but the user can disable it using the SBANK bit. The bank switching feature is also able to hold the bank switch during reads to confirm that data integrity is maintained. There is also the ability to manually switch banks with the SWBANK bit. These bits combine to allow for different ways to maintain control of the module's bank switching.

22.4.6 CRC

Optionally, Cyclic Redundancy Checking can be enabled using the CRCLEN bits within BiCLTCONx for data channels and control channels. This is one way to help determine if any issue has occurred during transmission. To implement this function, a checksum is calculated from the desired transmission and then converted into its CRC value. This value can then be checked during transmission to determine if any issues have already occurred. Depending on what CRC (Data Channel or Control) is being transmitted determines where in the protocol these values will be transmitted. The type of CRC also determines how many bits are available. Up to 17 bits for Data Channel CRC and four bits for Control CRC. The length of the CRC data channel can be set in the CRCLEN bits within BiCLTCONx. Settings for CRC can be configured in the BiCLTCONx and BiCTRLCONx registers.

If there is sufficient free memory available for CRC data, the read and inverted CRC bits are stored beginning at the highest address downwards. The storage of the CRC can be disabled with NOCRC.

22.5 Application Examples

22.5.1 Sensor Communication Code Example

```
// Read BiSS C SCD Frame, Sensor Data
int startupDelay;                    // Variable to read startup delay value into
int sensorValue;                    // Variable to store sensor values

B1CTRLCONbits.PROTOSEL = 1;         // Setting the protocol selection to BiSS C

// Configuring Host Clock for BiSS Module
B1CONbits.CLKDIV = 5;                // Setting CLKDIV divider
B1CTRLCONbits.SFREQ = 5;                // Setting SFREQ divider
B1CCONbits.FREQAGS = 0x7C;            // Setting FREQAGS to AGSMIN, host will automatically
trigger another SCD cycle once the previous has finished

// Configuring BiSS First Channel, Channel 0
B1CTRLCONbits.CHSEL = 1;            // Setting CTRLCON such that 1st channel, Channel 0, is
being used for communication
B1CHCONbits.CLCHCFG0 = 1;             // Setting CLCHCFG0 such that 1st channel, Channel 0,
is configured for BiSS C
B1CLTCON0bits.SCDEN0 = 1;            // Enabling SCD data
B1CLTCON0bits.SCDLEN0 = 29;            // Configuring SCD data length, 28 data bits, 1 error
bit, 1 warning bit
B1CLTCON0bits.CRCSEL0 = 0;            // Setting CRCSEL such that the CRC value will be
calculated based on CRCLEN dedicated polynomial
B1CLTCON0bits.CRCLEN0 = 0b0000110;    // Setting CRC polynomial to dedicated value, 0x43
B1CTRLCONbits.CLNTID = 0x00;        // Setting the Client ID for Channel 0, in this case the
Client ID is 0x00

// Asserting BiSS Communication
B1CONbits.ON = 1;                    // Turn on the BiSS Module
B1INSTRbits.INIT = 1;                // Initialize sequence which resets status registers and
initializes a line delay sense
while(B1CONbits.INSTRWA == 1);        // Waiting for initialization/instruction to complete
before any further write to instruction register
startupDelay = B1SCDATA0L;            // Optional to read startup delay and save to variable
B1INSTRbits.AGS = 1;                  // Starting automatic data transmission

int i = 10;                           // Creating while loop to read sensor ten times
while(i--){
    sensorValue = B1SCDATA0L;        // Reading sensor value into variable
    while(B1STATbits.EOT == 0);        // Waiting for the End of Transmission status bit to
confirm receive has occurred
    while(B1STATbits.CLSCDV0 == 0);    // Waiting for the Client SCD Valid bit to confirm a
successful receive has occurred
}
```

22.5.2 Read Register Code Example

```
// Read BiSS C Register Data
int registerValue;                    // Variable to store register value

B1CTRLCONbits.PROTOSEL = 1;         // Setting the protocol selection to BiSS C

// Configuring Host Clock for BiSS Module
B1CONbits.CLKDIV = 5;                // Setting CLKDIV divider
B1CTRLCONbits.SFREQ = 5;                // Setting SFREQ divider
B1CCONbits.FREQAGS = 0x7C;            // Setting FREQAGS to AGSMIN, host will automatically
trigger another SCD cycle once the previous has finished

// Configuring BiSS First Channel, Channel 0
B1CTRLCONbits.CHSEL = 1;            // Setting CTRLCON such that 1st channel, Channel 0, is
being used for communication
B1CHCONbits.CLCHCFG0 = 1;             // Setting CLCHCFG0 such that 1st channel, Channel 0,
is configured for BiSS C
B1CLTCON0bits.SCDEN0 = 1;            // Enabling SCD data
B1CLTCON0bits.SCDLEN0 = 29;            // Configuring SCD data length, 28 data bits, 1 error
bit, 1 warning bit
B1CLTCON0bits.CRCSEL0 = 0;            // Setting CRCSEL such that the CRC value will be
calculated based on CRCLEN dedicated polynomial
B1CLTCON0bits.CRCLEN0 = 0b0000110;    // Setting CRC polynomial to dedicated value, 0x43
B1CTRLCONbits.CLNTID = 0x00;        // Setting the Client ID for Channel 0, in this case the
Client ID is 0x00

// Setup for Register Read
B1CTRLCONbits.CTS = 1;                // Setting CTS for Register communication
B1RCCONbits.WNR = 0;                // Setting WNR for Read Register
```

```
B1RCCONbits.STRTADDR = 0x00;        // Start reading data at start address of 0x00 in BiSS
Client
B1RCCONbits.REGNUM = 0;               // Declare how many registers to read from the BiSS
Client, 0 means read 1 register

// Asserting BiSS Communication
B1CONbits.ON = 1;                    // Turn on the BiSS Module
B1INSTRbits.INIT = 1;                // Initialize sequence which resets status registers and
initializes a line delay sense
while(B1CONbits.INSTRWA == 1);        // Waiting for initialization/instruction to complete
before any further write to instruction register
B1INSTRbits.AGS = 1;                  // Starting automatic data transmission

// Initializing Register Read
while(B1STATbits.CDMTO == 0);         // Waiting until CDM Time Out to Read Register
B1INSTRbits.INSTR = 0b100;             // Using INSTR code to assert a register read
while(B1CONbits.INSTRWA == 1);        // Waiting for instruction to complete before any
further write to instruction register

// Waiting for Valid Receive
while(B1STATbits.EOT == 0);            // Waiting for the End of Transmission status bit to
confirm receive has occurred
while(B1STATbits.CLSCDV0 == 0);        // Waiting for the Client SCD Valid bit to confirm a
successful receive has occurred
while(B1STATbits.REGEND == 0);        // Waiting for the register receive to complete

registerValue = B1RDATA0;            // Saving register value to a variable
```

22.5.3 Write Register Code Example

```
// Write BiSS C Register Data
B1CTRLCONbits.PROTOSEL = 1;         // Setting the protocol selection to BiSS C

// Configuring Host Clock for BiSS Module
B1CONbits.CLKDIV = 5;                // Setting CLKDIV divider
B1CTRLCONbits.SFREQ = 5;             // Setting SFREQ divider
B1CCONbits.FREQAGS = 0x7C;            // Setting FREQAGS to AGSMIN, host will automatically
trigger another SCD cycle once the previous has finished

// Configuring BiSS First Channel, Channel 0
B1CTRLCONbits.CHSEL = 1;            // Setting CTRLCON such that 1st channel, Channel 0, is
being used for communication
B1CHCONbits.CLCHCFG0 = 1;             // Setting CLCHCFG0 such that 1st channel, Channel 0,
is configured for BiSS C
B1CLTCON0bits.SCDEN0 = 1;            // Enabling SCD data
B1CLTCON0bits.SCDLEN0 = 29;            // Configuring SCD data length, 28 data bits, 1 error
bit, 1 warning bit
B1CLTCON0bits.CRCSEL0 = 0;            // Setting CRCSEL such that the CRC value will be
calculated based on CRCLEN dedicated polynomial
B1CLTCON0bits.CRCLEN0 = 0b0000110;    // Setting CRC polynomial to dedicated value, 0x43
B1CTRLCONbits.CLNTID = 0x00;        // Setting the Client ID for Channel 0, in this case the
Client ID is 0x00

// Setup for Register Write
B1CTRLCONbits.CTS = 1;                // Setting CTS for Register communication
B1RCCONbits.WNR = 1;                // Setting WNR for Write Register
B1RCCONbits.STRTADDR = 0x50;        // Start writing data at start address of 0x50 in BiSS
Client
B1RCCONbits.REGNUM = 0;               // Declare how many registers to write from the BiSS
Client, 0 means read 1 register

B1CONbits.TXRDEN = 1;                // Setting the transmit RAM enable bit to enable
transmission

// Asserting BiSS Communication
B1CONbits.ON = 1;                    // Turn on the BiSS Module
B1INSTRbits.INIT = 1;                // Initialize sequence which resets status registers and
initializes a line delay sense
while(B1CONbits.INSTRWA == 1);        // Waiting for initialization/instruction to complete
before any further write to instruction register
B1INSTRbits.AGS = 1;                  // Starting automatic data transmission

B1RDATA0 = 0x23;                     // Setting the desired register value to transmit

// Initializing Register Write
while(B1STATbits.CDMTO == 0);         // Waiting until CDM Time Out to Write Register
```

```
B1INSTRbits.INSTR = 0b100;              // Using INSTR code to assert a register write
while(B1CONbits.INSTRWA == 1);         // Waiting for instruction to complete before any
further write to instruction register

// Waiting for Valid Transmissions
while(B1STATbits.EOT == 0);             // Waiting for the End of Transmission status bit to
confirm transmission has occurred
while(B1STATbits.CLSCDV0 == 0);         // Waiting for the Client SCD Valid bit to confirm a
successful transmission has occurred
while(B1STATbits.REGEND == 0);         // Waiting for the register transmission to complete
```

22.5.4 Broadcast Command Code Example

```
// Send Broadcast Command
B1CTRLCONbits.PROTOSEL = 1;          // Setting the protocol selection to BiSS C

// Configuring Host Clock for BiSS Module
B1CONbits.CLKDIV = 5;                 // Setting CLKDIV divider
B1CTRLCONbits.SFREQ = 5;             // Setting SFREQ divider
B1CCONbits.FREQAGS = 0x7C;             // Setting FREQAGS to AGSMIN, host will automatically
trigger another SCD cycle once the previous has finished

// Configuring BiSS First Channel, Channel 0
B1CTRLCONbits.CHSEL = 1;             // Setting CTRLCON such that 1st channel, Channel 0, is
being used for communication
B1CHCONbits.CLCHCFG0 = 1;              // Setting CLCHCFG0 such that 1st channel, Channel 0,
is configured for BiSS C
B1CLTCON0bits.SCDEN0 = 1;             // Enabling SCD data
B1CLTCON0bits.SCDLEN0 = 29;             // Configuring SCD data length, 28 data bits, 1 error
bit, 1 warning bit
B1CLTCON0bits.CRCSEL0 = 0;             // Setting CRCSEL such that the CRC value will be
calculated based on CRCLEN dedicated polynomial
B1CLTCON0bits.CRCLEN0 = 0b0000110;     // Setting CRC polynomial to dedicated value, 0x43
B1CTRLCONbits.CLNTID = 0x01;         // Setting the Client ID for Channel 0, in this case the
Client ID is 0x01

// Setup for Broadcast Command
B1CTRLCONbits.CTS = 0;                 // Setting CTS for Command communication
B1RCCONbits.WNR = 0;                 // Setting WNR for Read Register (IS THIS NEEDED?)
B1RCCONbits.STRTADDR = 0x00;         // Start reading data at start address of 0x00 in BiSS
Client
B1RCCONbits.REGNUM = 0;                // Declare how many registers to read from the BiSS
Client, 0 means read 1 register
B1CTRLCONbits.CMD = 2;                 // Setting what command value to send, this should be
described in Client documentation

// Turning the module on and starting communication
B1CONbits.ON = 1;

B1INSTRbits.INIT = 1;                // Initialize sequence which resets status registers and
initializes a line delay sense
while(B1CONbits.INSTRWA == 1);         // Waiting for initialization/instruction to complete
before any further write to instruction register
B1INSTRbits.AGS = 1;                   // Starting automatic data transmission

// Initializing Register Read
while(B1STATbits.CDMTO == 0);          // Waiting until CDM Time Out to Read Register
B1INSTRbits.INSTR = 0b100;              // Using INSTR code to send command(INSTR value is the
same as read register command but CTS = 1 means send a command)
while(B1CONbits.INSTRWA == 1);         // Waiting for instruction to complete before any
further write to instruction register

while(B1STATbits.REGEND == 0);         // Waiting for the register transmission to complete
```

22.5.5 Addressed Command Code Example

```
// Send Addressed Command
B1CTRLCONbits.PROTOSEL = 1;          // Setting the protocol selection to BiSS C

// Configuring Host Clock for BiSS Module
B1CONbits.CLKDIV = 5;                 // Setting CLKDIV divider
B1CTRLCONbits.SFREQ = 5;             // Setting SFREQ divider
B1CCONbits.FREQAGS = 0x7C;             // Setting FREQAGS to AGSMIN, host will automatically
trigger another SCD cycle once the previous has finished
```

```
// Configuring BiSS First Channel, Channel 0
B1CTRLCONbits.CHSEL = 1;            // Setting CTRLCON such that 1st channel, Channel 0, is
being used for communication
B1CHCONbits.CLCHCFG0 = 1;             // Setting CLCHCFG0 such that 1st channel, Channel 0,
is configured for BiSS C
B1CLTCON0bits.SCDEN0 = 1;            // Enabling SCD data
B1CLTCON0bits.SCDLEN0 = 29;            // Configuring SCD data length, 28 data bits, 1 error
bit, 1 warning bit
B1CLTCON0bits.CRCSEL0 = 0;            // Setting CRCSEL such that the CRC value will be
calculated based on CRCLEN dedicated polynomial
B1CLTCON0bits.CRCLEN0 = 0b0000110;    // Setting CRC polynomial to dedicated value, 0x43
B1CTRLCONbits.CLNTID = 0x01;        // Setting the Client ID for Channel 0, in this case the
Client ID is 0x01

// Setup for Addressed Command
B1CTRLCONbits.CTS = 0;                // Setting CTS for Command communication
B1RCCONbits.WNR = 0;                // Setting WNR for Read Register
B1RCCONbits.STRTADDR = 0x00;        // Start reading data at start address of 0x00 in BiSS
Client
B1RCCONbits.REGNUM = 0;                // Declare how many registers to read from the BiSS
Client, 0 means read 1 register
B1CTRLCONbits.CMD = 2;                // Setting what command value to send, this should be
described in Client documentation
B1IDS0bits.IDS0 = 0x80;             // Setting the Client address for sending command

// Turning the module on and starting communication
B1CONbits.ON = 1;

B1INSTRbits.INIT = 1;               // Initialize sequence which resets status registers and
initializes a line delay sense
while(B1CONbits.INSTRWA == 1);        // Waiting for initialization/instruction to complete
before any further write to instruction register
B1INSTRbits.AGS = 1;                // Starting automatic data transmission

// Initializing Register Read
while(B1STATbits.CDMTO == 0);         // Waiting until CDM Time Out to Read Register
B1INSTRbits.INSTR = 0b100;             // Using INSTR code to send command(INSTR value is the
same as read register command but CTS = 1 means send a command)
while(B1CONbits.INSTRWA == 1);        // Waiting for instruction to complete before any
further write to instruction register
```

22.6 Interrupts

The BiSS module generates two interrupts: end of transmission (EOT) and transmission error (ERR). These interrupts can be used by clearing and enabling their appropriate interrupt flags.

22.6.1 End Of Transmission (EOT) Interrupt

The EOT interrupt is enabled using the relevant IECx register, and the flag is cleared and set in the appropriate IFSx register. This interrupt is asserted each time a successful transmission has completed.

22.6.2 Transmission Error (ERR) Interrupt

The ERR interrupt is enabled using the relevant IECx register, and the flag is cleared and set in the appropriate IFSx register. This interrupt is asserted when any of the BiSS error bits are set. The user can read the status flags to determine the specific error.

22.7 Power Saving Modes

BiSS provides support in power-saving modes, including the capability to run in Sleep and Idle modes.

22.7.1 Sleep

Setting the SLPEN bit within the BiCON register will keep the module active in Sleep mode.

22.7.2 Idle

Setting the SIDL bit within the BiCON register will turn the module off in Idle mode.

22.8 Terminology

Table 22-6. Terminology

Acronym	Description
MA	Clock output of the host
MO	Main data output of the host
SL	Data output
SLI	Data input of the client
SLO	Data output of the client
BiSS	Bidirectional Serial Synchronous
SSI	Serial Synchronous Interface
CDM	Control Data Host
CDS	Control Data Client
CRC	Cyclic Redundancy Check
nE	Error bit
nW	Warning bit
SCD	Single-Cycle Data
CD	Control Data
CTS	Control Select bit
ID	Identification
IDS	ID Select
IDA	ID Acknowledge
IDL	ID Lock
ADR	Address
S	Start
P	Stop
CMD	Command
EX	Execute
*S	Optional Start
Cyclical	Occurring in cycles, in a predictive and deterministic manner
Isochronous	Occurring at the same time

23. Timer1

The Timer1 module is a 32-bit timer that can be configured for both synchronous and asynchronous operation. The Timer1 module features include:

- Asynchronous and Synchronous Operation
- External Clock
- Various Timer/Counter Modes
- Adds SFR Control for Various Clock Sources

23.1 Device-Specific Information

Table 23-1. Timer1 Summary Table

Timer1 Module Instances	Peripheral Bus Speed	Clock Source
1	Standard	Standard Speed Peripheral Clock

Table 23-2. Timer1 Clock Source Select bit

Value	Description
1	External clock source selected by TECS[1:0]
0	Standard speed peripheral clock

23.2 Architectural Overview

The Timer1 module found in dsPIC33A family devices shares many features with a classic Type A timer. There is typically a single Timer1 module implemented for each CPU core present in a device. Timers are useful for generating accurate time-based periodic interrupt events for software applications or real-time operating systems. Other uses include counting external pulses or accurate timing measurement of external events by using the timer's gate feature.

The timers found in some dsPIC33A family devices include the following features:

- Asynchronous and Synchronous Operation
- Software-Selectable Internal or External Clock Sources
- Programmable Interrupt Generation and Priority
- Gated External Pulse Counter
- Operation During Sleep Mode
- Software Prescalers: 1:1, 1:8, 1:64 and 1:256

The unique features of the Timer1 module allow it to be used for Real-Time Clock (RTC) applications. Figure 23-1 illustrates the block diagram of a Timer1 module and Figure 23-2 shows the Timer1 clock input logic.

Figure 23-1. Timer1 Block Diagram

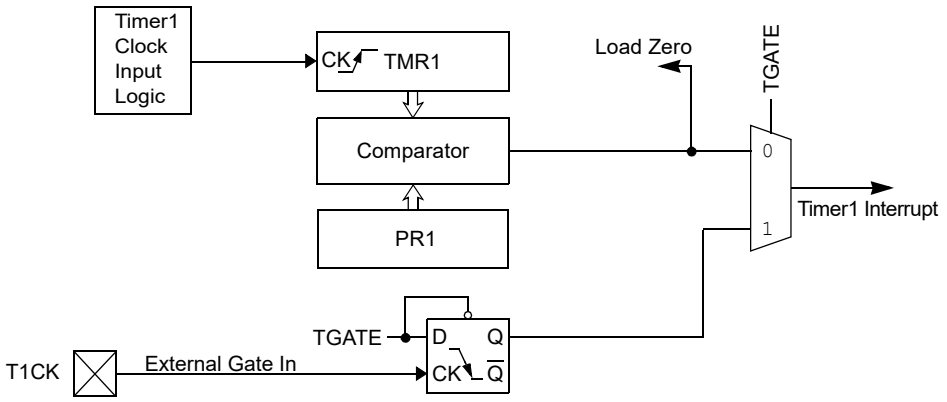
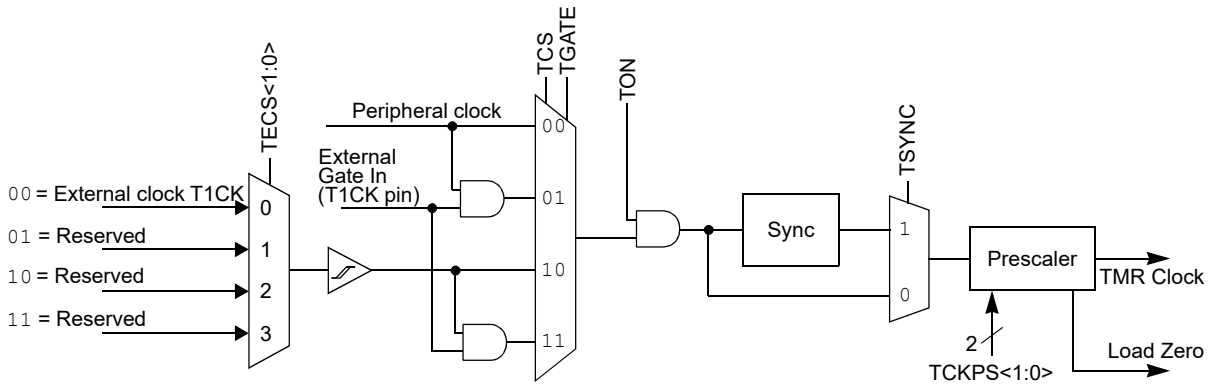


Figure 23-2. Timer1 Clock Input Logic



23.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1E00	T1CON	31:24								
		23:16								
		15:8	ON		SIDL	TMWDIS	TMWIP	PRWIP	TECS[1:0]	
		7:0	TGATE		TCKPS[1:0]			TSYNC	TCS	
0x1E04	TMR1	31:24	TMR[31:24]							
		23:16	TMR[23:16]							
		15:8	TMR[15:8]							
		7:0	TMR[7:0]							
0x1E08	PR1	31:24	PR[31:24]							
		23:16	PR[23:16]							
		15:8	PR[15:8]							
		7:0	PR[7:0]							

23.3.1 Timer1 Control Register

Name: T1CON
Offset: 0x1E00

Note:

1. When Timer1 is enabled in External Synchronous Counter mode (TCS = 1, TSYNC = 1, TON = 1), any attempts by user software to write to the TMR1 register are ignored.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	TMWDIS	TMWIP	PRWIP	TECS[1:0]	
Access	R/W		R/W	R/W	R	R	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TGATE		TCKPS[1:0]			TSYNC	TCS	
Access	R/W		R/W	R/W		R/W	R/W	
Reset	0		0	0		0	0	

Bit 15 – ON Timer1 On bit(1)

Value	Description
1	Starts 32-bit Timer1
0	Stops 32-bit Timer1

Bit 13 – SIDL Timer1 Stop in Idle Mode bit

Value	Description
1	Discontinues module operation when device enters Idle mode
0	Continues module operation in Idle mode

Bit 12 – TMWDIS Asynchronous Timer1 Write Disable bit

Value	Description
1	Timer writes are ignored while a posted write to TMR1 or PR1 is synchronized to the asynchronous clock domain
0	Back-to-back writes are enabled in Asynchronous mode

Bit 11 – TMWIP Asynchronous Timer1 Write in Progress bit

Value	Description
1	Write to the timer in Asynchronous mode is pending
0	Write to the timer in Asynchronous mode is complete

Bit 10 – PRWIP Asynchronous Period Write in Progress bit

Value	Description
1	Write to the Period register in Asynchronous mode is pending
0	Write to the Period register in Asynchronous mode is complete

Bits 9:8 – TECS[1:0] Timer1 Extended Clock Select bits

Value	Description
11	Reserved
10	Reserved
01	Reserved
00	External clock comes from the T1CK pin

Bit 7 – TGATE Timer1 Gated Time Accumulation Enable bit

When TCS = 1:

This bit is ignored.

When TCS = 0:

Value	Description
1	Gated time accumulation is enabled
0	Gated time accumulation is disabled

Bits 5:4 – TCKPS[1:0] Timer1 Input Clock Prescale Select bits

Value	Description
11	1:256
10	1:64
01	1:8
00	1:1

Bit 2 – TSYNC Timer1 External Clock Input Synchronization Select bit(1)

When TCS = 0:

This bit is ignored.

When TCS = 1:

Value	Description
1	Synchronizes the external clock input
0	Does not synchronize the external clock input

Bit 1 – TCS Timer1 Clock Source Select bit(1)

See Table 23-2

23.3.2 Timer1 Counter Register

Name: TMR1
Offset: 0x1E04

Bit	31	30	29	28	27	26	25	24
	TMR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TMR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TMR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TMR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – TMR[31:0] Timer Value bits

23.3.3 Period Register 1

Name: PR1
Offset: 0x1E08

Bit	31	30	29	28	27	26	25	24
	PR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PR[31:0] Period Register bits

23.4 Operation

The Timer1 modules found in dsPIC33A family devices support the following modes of operation:

- Synchronous Clock Counter
- Synchronous External Clock Counter
- Gated Timer
- Asynchronous External Counter

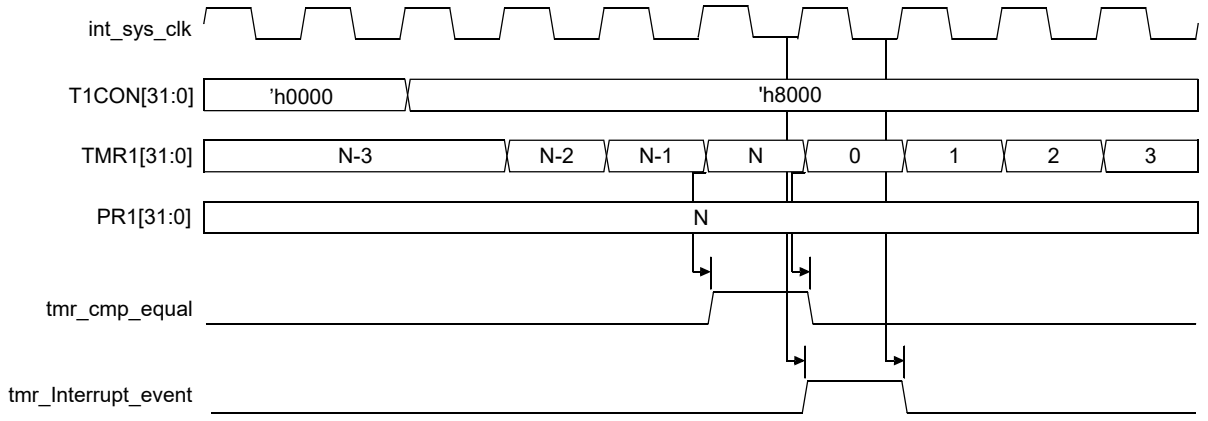
The Timer modes are determined by the following bits:

- TCS (T1CON[1]): Timer1 Clock Source Select bit
- TGATE (T1CON[7]): Timer1 Gated Time Accumulation Enable bit
- TSYNC (T1CON[2]): Timer1 External Clock Input Synchronization Selection bit

23.4.1 Synchronous Clock Counter Mode

When T1CON.TCS = 0 and T1CON.TON = 1, the timer increments on every rising edge of the internal system clock sys_clk/2 (int_sys_clk) up to a match value, preloaded into the period register PR1, then rolls over and continues. The use of period registers allows for any timer value to be reached as the maximum while providing a specific period/time interval to be repeated with no firmware intervention (refer to Figure 23-3).

Figure 23-3. Synchronous Clock Counter Mode Timing Diagram (1:1 Prescale)



For the maximum timer period, load the period register with 0xFFFF_FFFF. This incrementing sequence repeats until the timer is disabled by being turned off (T1CON.TON = 0), stopped in Idle (T1CON.SIDL = 1) or the int_sys_clk is turned off (Sleep mode). The timer count is not reset when the module is turned off.

When the CPU goes into Sleep mode or Idle mode with T1CON.SIDL = 1, the timer will stop incrementing. The timer module logic will resume the incrementing sequence upon termination of the CPU Idle or Sleep mode. If the CPU goes into Idle mode with T1CON.SIDL = 0, the timer will continue to operate normally.

23.4.1.1 Synchronous Clock Counter Considerations

The user may write the timer register to initialize the timer. Since the timer is configured to increment from the Int_sys_clk, the module's interrupt output will be synchronous to the rising edge of int_sys_clk and no interrupt latency will be added by the module.

23.4.1.2 Synchronous Counter Initialization Steps

The following steps must be performed to configure the timer for Synchronous Timer mode:

1. Clear the TON control bit (T1CON[15] = 0) to disable the timer.
2. Clear the TCS control bit (T1CON[1] = 0) to select the internal int_sys_clk source.
3. Select the desired timer input clock prescale.
4. Load/clear the Timer1 register, TMR1.
5. Load the Timer1 Period register, PR1, with the desired 32-bit match value.
6. If interrupts are used:
   a. Clear the T1IF Interrupt Flag bit in the IFSx register.
   b. Configure the Interrupt Priority Levels in the IPCx register.
   c. Set the T1IE Interrupt Enable bit in the IECx register.
7. Set the TON control bit (T1CON[15] = 1) to enable the timer.

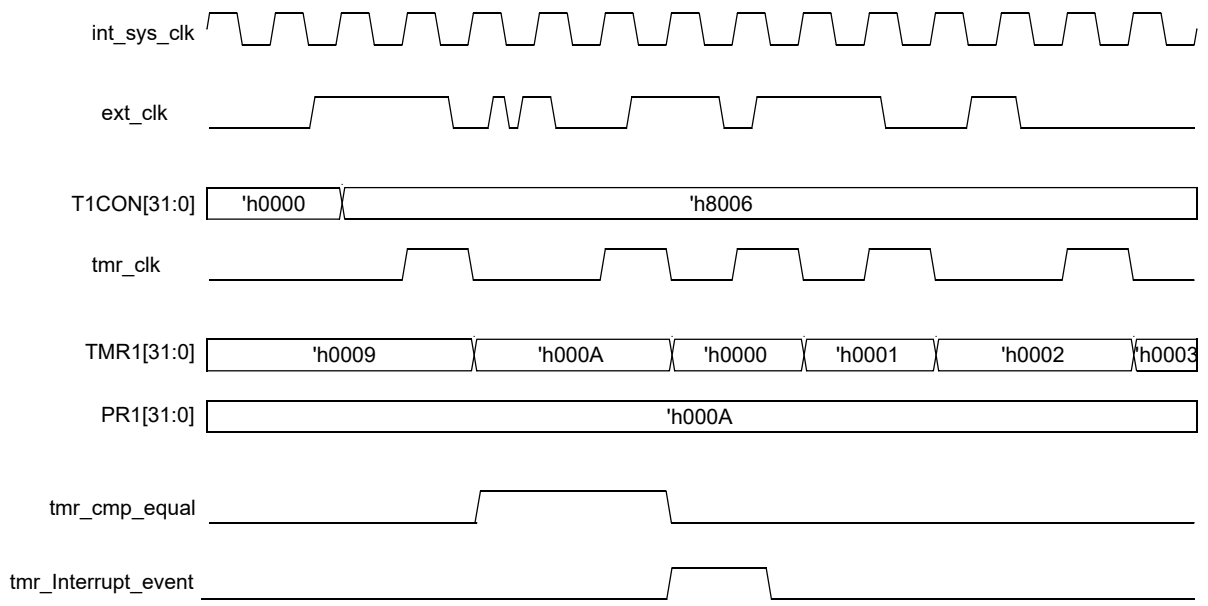
Example 23-1. Synchronous Clock Counter Example Code

```
T1CON = 0x0;                    // Stop timer and clear control register,
// set prescaler at 1:1, internal clock source
TMR1 = 0x0;                       // Clear timer register
PR1 = 0xFFFFFFFF;                   // Load period register
T1CONbits.TON = 1;                  // Start timer
```

23.4.2 Synchronous External Clock Counter Mode

When T1CON.TCS = 1, T1CON.TON = 1 and T1CON.TSYNC = 1, the timer increments on the synchronized rising edge of the applied external clock (ext_clk) signal. The synchronization of the input signal occurs with the int_sys_clk signal. The timer counts up to a match value preloaded in the period register, then resets and continues. This incrementing sequence repeats until the timer is disabled (refer to Figure 23-4).

Figure 23-4. Synchronized External Clock Mode Timing Diagram



When the CPU goes into Sleep mode, or when the timer is configured for the Synchronous mode of operation and the CPU goes into Idle mode with T1CON.SIDL = 1, the timer will stop incrementing. The timer module logic will resume the incrementing sequence upon termination of the CPU Idle/ Sleep mode. If T1CON.SIDL = 0, the timer will continue to operate.

23.4.2.1 Synchronous External Clock Counter Considerations

In Synchronous External Clock Counter mode, the externally supplied clock, ext_clk, must be half the int_sys_clk frequency or slower. Additionally, the external clock high and low times must be at least one full int_sys_clk period each for reliable operation. If the external clock glitches (positive or negative) or operates faster than half the peripheral clock, the timer may or may not increment. In no case will the timer increment faster than half the int_sys_clk.

When the timer is configured to increment from the synchronized external clock, the interrupt latency will be '0', relative to the TMR1 rollover (transition from TMR1 = PR1 to '0'). As the timer will be rolling over from a synchronized clock, the interrupt output will be synchronous to the int_sys_clk rising edge.

When T1CON.TCS = 1, T1CON.ON = 1, T1CON.TCKPS[1:0] ≠ 00 and T1CON.TSYNC = 1, the timer increments on the synchronized rising edge of the prescaler output. The synchronization of the input signal occurs with the internal int_sys_clk signal prior to the prescaler. The timer counts up to a match value preloaded in the period register, then resets and continues. This incrementing sequence repeats until the timer is disabled.

23.4.2.2 Synchronous External Counter Initialization Steps

The following steps must be performed to configure the timer for Synchronous Counter mode:

1. Clear the TON Control bit (T1CON[15] = 0) to disable the timer.

2. If available on the device, set the External Clock source using the TECS[1:0] bits (T1CON[9:8]).
3. Set the TCS Control bit (T1CON[1] = 1) to enable the clock selection.
4. Set the TSYNC Control bit (T1CON[2] = 1) to enable clock synchronization.
5. Select the desired timer input clock prescale.
6. Load/clear the Timer1 register, TMR1.
7. If using period match:
   a. Load the Timer1 Period register, PR1, with the desired 32-bit match value.
8. If interrupts are used:
   a. Clear the T1IF Interrupt Flag bit in the IFSx register.
   b. Configure the Interrupt Priority Levels in the IPC1 register.
   c. Set the T1IE Interrupt Enable bit in the IEC1 register.
9. Set the TON Control bit (T1CON[15] = 1) to enable the timer.

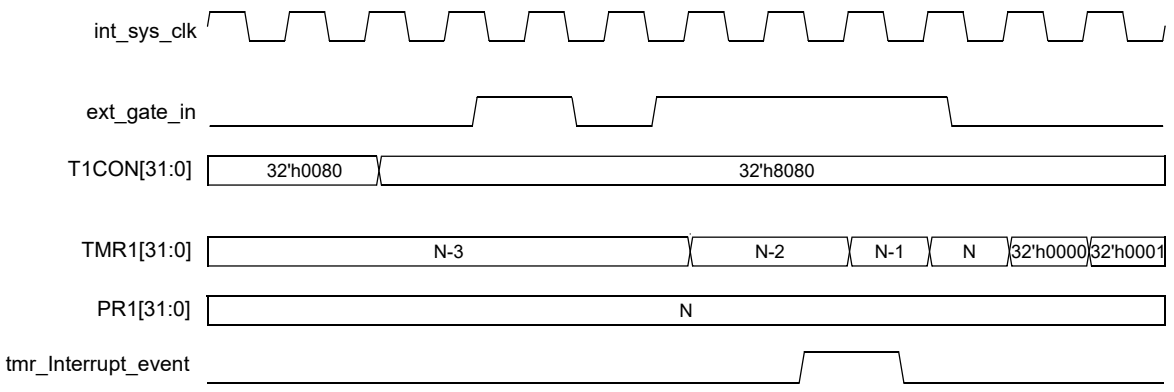
Example 23-2. Synchronous External Counter Example Code

```
T1CON = 0x0;                // Stop timer and clear control register
T1CON = 0x00000006;                 // Set prescaler at 1:1, external clock
source
TMR1 = 0x0;                       // Clear timer register
PR1 = 0xFFFFFFFF;                   // Load period register
T1CONbits.TON = 1;                  // Start timer
```

23.4.3 Gated Timer Mode

The timer can be placed in Gated Time Accumulation Operational mode to enable the timer clock source when the gate signal is asserted high. The Timer Control bit T1CON.TGATE must be set to enable this mode. The timer must be enabled, T1CON.TON = 1 (refer to Figure 23-5).

Figure 23-5. Timer Gate Mode Timing Diagram



When configured for this mode, gate operation starts on a rising edge of the signal applied to the T1CK input and terminates on the falling edge of the signal applied to the T1Ck input. The timer will increment while the external gate signal is high. The gate signal must have a minimum high and low time greater than the timer input clock period to ensure that the T1CK input will have sufficient setup time to be sampled by the rising edge of the clock. The edges of the gate signal may occur asynchronously to the timer clock source.

The falling edge of the gate signal generates an interrupt. The latency of the interrupt is one to two time clock cycles after the falling edge. If TGATE = 1, T1CK = 0 and TON is set, an interrupt will be generated despite no falling edge existing on the gate input. The falling edge of the T1CK terminates

the count operation, but does not reset the timer. The user must reset the timer if desiring to start from zero.

23.4.3.1 Gated Timer Mode Clock Considerations

The timer may be using the prescaler for a slower timer increment rate. Note that the gated timer clock is routed through the prescaler. When using the prescaler, the prescaler will simply retain its current count value as the gate is low. Note that if the user writes the TMR1[15:0] register, the prescaler will be reset. Using the prescaler will not affect the function or timing of the falling edge interrupt.

When the CPU goes into Sleep mode or Idle mode with T1CON.SIDL = 1, the timer will stop incrementing. The timer module logic will resume the incrementing sequence upon termination of the CPU Idle/Sleep mode. If T1CON.SIDL = 0, the timer will continue to operate normally.

The period matching function remains operational in gate mode operation. If the timer matches the period, the timer will reset, but the period match does not generate an interrupt.

Note: Gated Timer mode is overridden if the Timer1 Clock Source Select bit (TCS) is set to an external clock source, TCS = 1. For Gated Timer mode operation, the internal clock source must be selected (TCS = 0).

23.4.3.2 Gated Timer Initialization Steps

The following steps must be performed to configure the timer for Gated Timer mode:

1. Clear the TON Control bit (TxCON[15] = 0) to disable the timer.
2. Clear the TCS Control bit (TxCON[1] = 0) to select the internal int_sys_clk source.
3. Set the TGATE Control bit (T1CON[7] = 1) to enable Gated Timer mode.
4. Select the desired prescaler.
5. Clear the Timer1 register, TMR1.
6. Load the Timer1 Period register, PR1, with the desired 32-bit match value.
7. If interrupts are used:
   a. Clear the T1IF Interrupt Flag bit in the IFSx register.
   b. Configure the Interrupt Priority Levels in the IPCx register.
   c. Set the T1IE Interrupt Enable bit in the IECx register.
8. Set the TON Control bit (TxCON[15] = 1) to enable the timer.

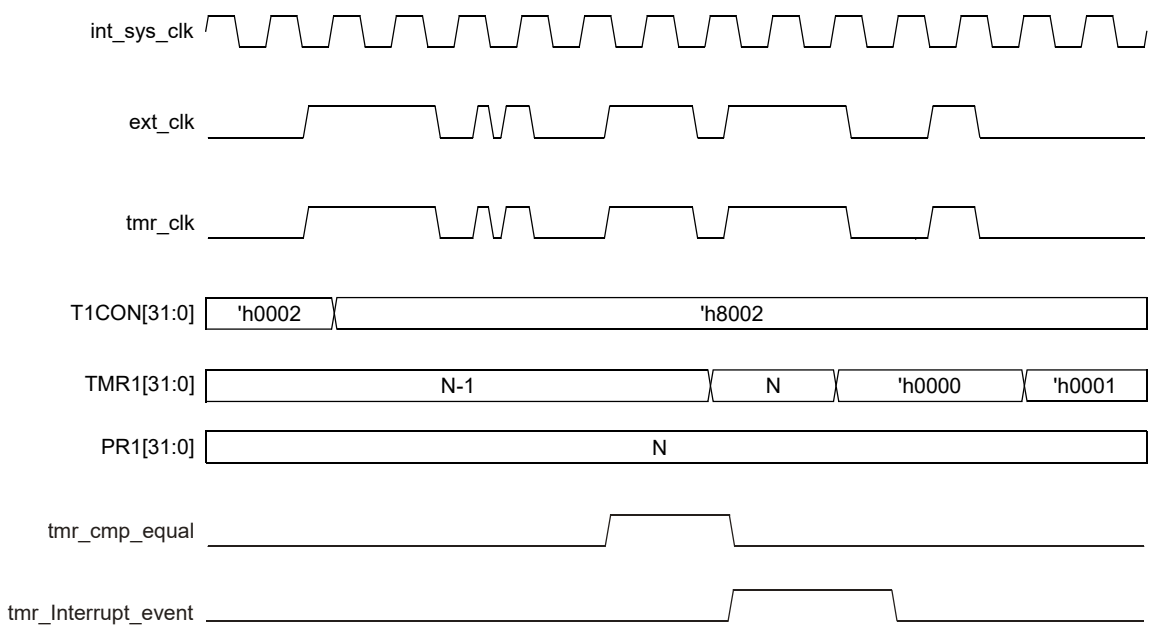
Example 23-3. Gated Timer Example Code

```
T1CON = 0x0;             // Stop timer and clear control register
T1CON = 0x00000080;              // Gated Timer mode, prescaler at 1:1,
internal clock source
TMR1 = 0x0;                 // Clear timer register
PR1 = 0xFFFFFFFF;              // Load period register with 32-bit match value
T1CONbits.TON = 1;             // Start timer
```

23.4.4 Asynchronous Clock Counter Mode

When T1CON.TCS = 1, T1CON.ON = 1 and T1CON.TSYNC = 0, the timer increments on every rising edge of the applied external clock signal. The timer counts up to a match value preloaded in the respective period register, then rolls over and continues. This incrementing sequence repeats until the timer is disabled (refer to Figure 23-6).

Figure 23-6. Asynchronous External Clock Mode Timing Diagram



When the timer is configured for asynchronous operation and the CPU goes into Idle mode with T1CON.SIDL = 1 or sleep, the timer continues incrementing.

23.4.4.1 Asynchronous Mode TMR1 Read and Write Operations

Two bits in the TxCON register can be used to ensure that writes during timer operation will not cause the timer value to be corrupted. The TxCON.TMWDIS bit, when set, will prevent a write to the timer and period registers when a previous write to the timer is awaiting synchronization into the asynchronous timer clock domain. The TxCON.TMWIP and TxCON.PRWIP bits indicate when write synchronization is complete and it is safe to write another value to the timer register and period register. These bits have no effect in Synchronous Clock Counter modes.

The Timer1 Write in Progress bit (TWIP) provides two options for safely writing to the TMR1 Count register while Timer1 is enabled.

Option 1 is the legacy Timer1 Write mode, TWDIS bit = 0. To determine when it is safe to write to the TMR1 Count register, it is recommended to poll the TWIP bit. When TWIP = 0, it is safe to perform the next write operation to the TMR1 Count register. When TWIP = 1, the previous write operation to the TMR1 Count register is still being synchronized and any additional write operations should wait until TWIP = 0.

Option 2 is the new Synchronized Timer1 Write mode, TWDIS bit = 1. A write to the TMR1 Count register can be performed at any time. However, if the previous write operation to the TMR1 Count register is still being synchronized, any additional write operations are ignored.

Note: A write to the TMR1 Count register must be performed prior to configuring Timer1 for Asynchronous mode if the proper procedure to check TWIP and TWDIS is not followed.

23.4.4.2 Asynchronous Clock Counter Considerations

The asynchronous TMR1 value is synchronized before it is read. This allows the user to read a clean timer value; however, this also means the read data is the value of TMR1 from two int_sys_clk cycles prior to the read. This is also the value stored into the TMR1 register when the module is turned off.

When the timer is configured in asynchronous mode, the act of setting the T1CON.ON bit does not take effect until two rising edges of the external clock input have occurred.

When the timer is configured in Asynchronous mode and the timer is running, a write to TMR1 must be synchronized for two asynchronous timer clock pulses before the contents of the timer register are updated with the timer value. The timer will not increment until two timer clocks have transpired. Two clock cycles are required to synchronize the write data into the clock domain of the asynchronous timer.

When the timer is configured for an asynchronous external clock, the interrupt latency is one asynchronous timer clock period plus 2 sys_clk.

23.4.4.3 Asynchronous External Clock Counter Initialization Steps

The following steps must be performed to configure the timer for 32-bit Asynchronous Counter mode:

1. Clear the TON Control bit (T1CON[15] = 0) to disable the timer.
2. If available on the device, set the external clock source using the TECS[1:0] bits (T1CON[9:8]).
3. Set the TCS Control bit (T1CON[1] = 1) to enable external clock selection.
4. Clear the TSYNC Control bit (T1CON[2] = 0) to disable clock synchronization.
5. Select the desired prescaler.
6. Load/clear the Timer1 register, TMR1.
7. If using period match, load the Timer1 Period register, PR1, with the desired 32-bit match value.
8. If interrupts are used:
   a. Clear the T1IF Interrupt Flag bit in the IFSx register.
   b. Configure the interrupt priority levels in the IPCx register.
   c. Set the T1IE Interrupt Enable bit in the IECx register.
9. Set the TON Control bit (T1CON[15] = 1) to enable the timer.

Example 23-4. 32-bit Asynchronous Counter Mode Code

```
/*    32-bit Asynchronous Counter Mode Example */
T1CON = 0x0;                    // Stops the Timer1 and resets the control
register
TMR1 = 0x0;                     // Clear timer register
T1CON = 0x00000002;             // Set prescaler 1:1, external clock,
asynchronous mode
PR1 = 0xFFFFFFFF;               // Load period register
T1CONbits.TON = 1;              // Start timer
```

23.4.5 Timer Prescalers

Timer1 timers provide input clock (peripheral clock or external clock) prescale options of 1:1, 1:8, 1:64 and 1:256, which can be selected by using the TCKPS[1:0] bits (T1CON[5:4]).

The prescaler counter is cleared when any of the following occurs:

- A write to the TMR1 register
- Disabling the Timer1 TON bit (T1CON[15]) = 0
- Any device Reset

23.4.5.1 Period Match and Slow Clock

When using the prescaler, the user software can change the value of the period register faster than one timer period.

Figure 23-9 shows that when the counter is operating with an 8x prescale and PRx is written before the end of the prescaler rollover, a period match is not detected and an interrupt is not generated. In this case, the timer will continue running until a compare match occurs between TMRx and the

new value *M*. If *M* is lower than the original period of *N*, the timer will run all the way to 0xFFFF_FFFF and rollover to 0x0000_0000.

Figure 23-7. Synchronous Clock with Prescale with Write to PRx Following Match (TxCON.TCS = 0, TxCON.TCKPS[1:0] =01)

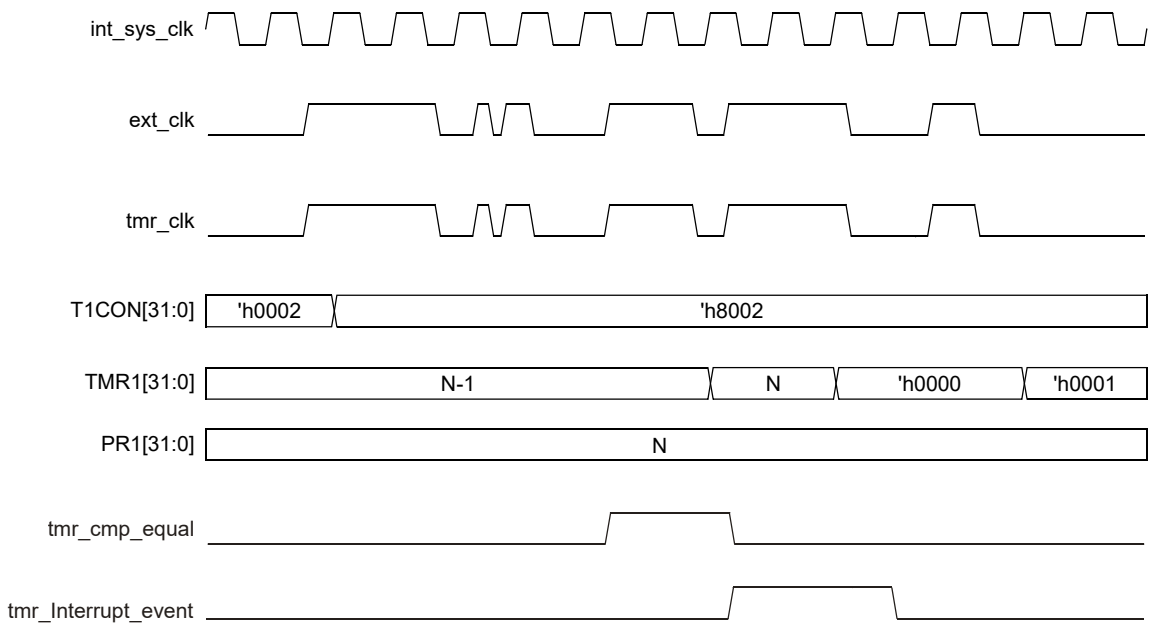


Figure 23-8. Synchronized External Clock with 8:1 Prescale Timing Diagram

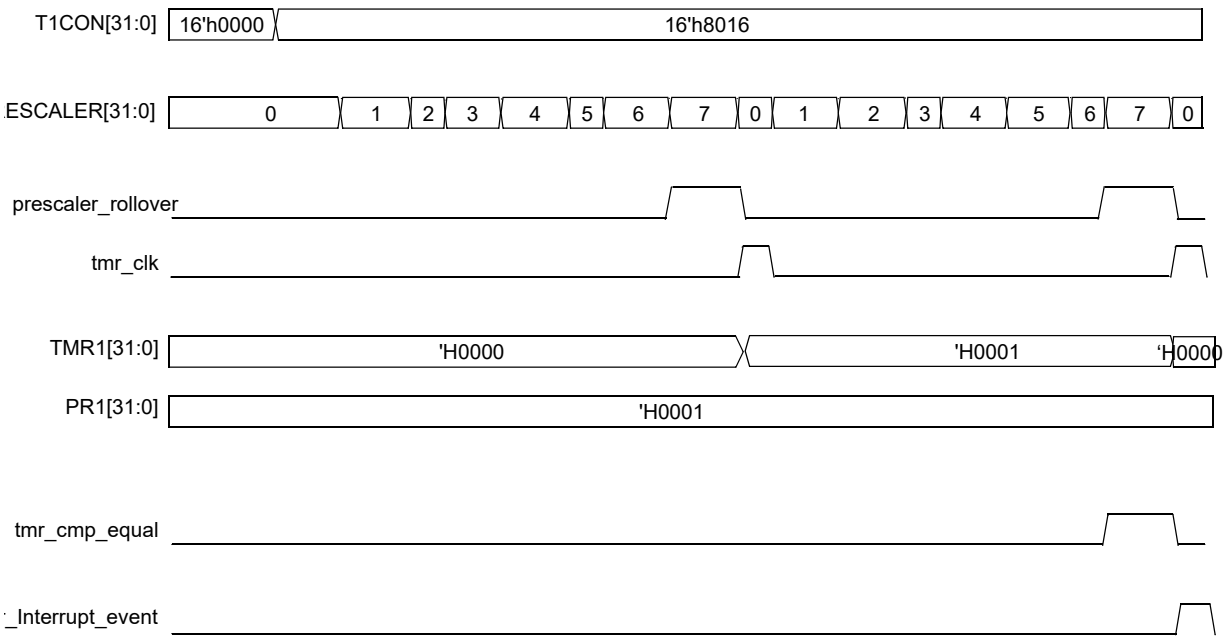
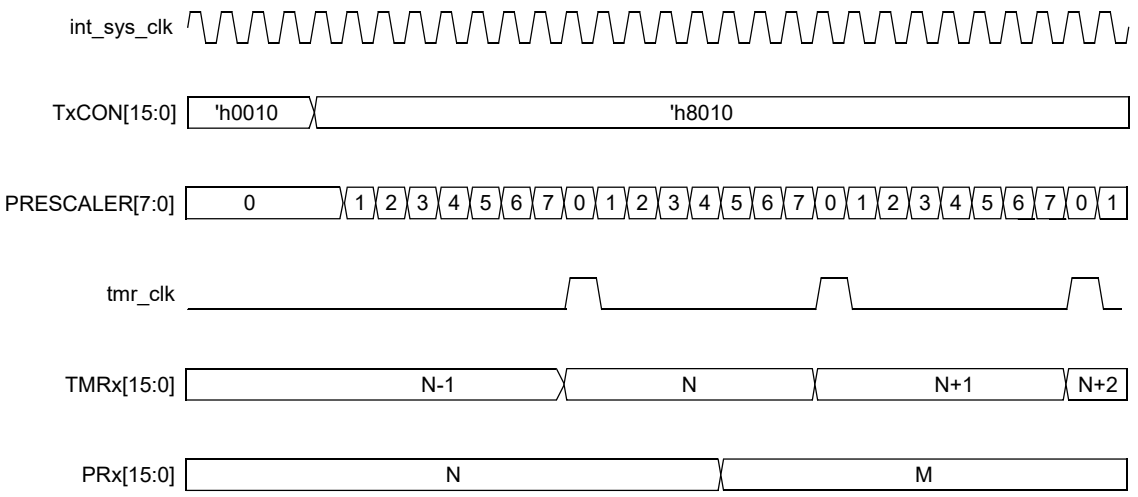


Figure 23-9. Synchronous Clock with Prescale with Write to PRx Following Match (TxCON.TCS = 0, TxCON.TCKPS[1:0] =01)

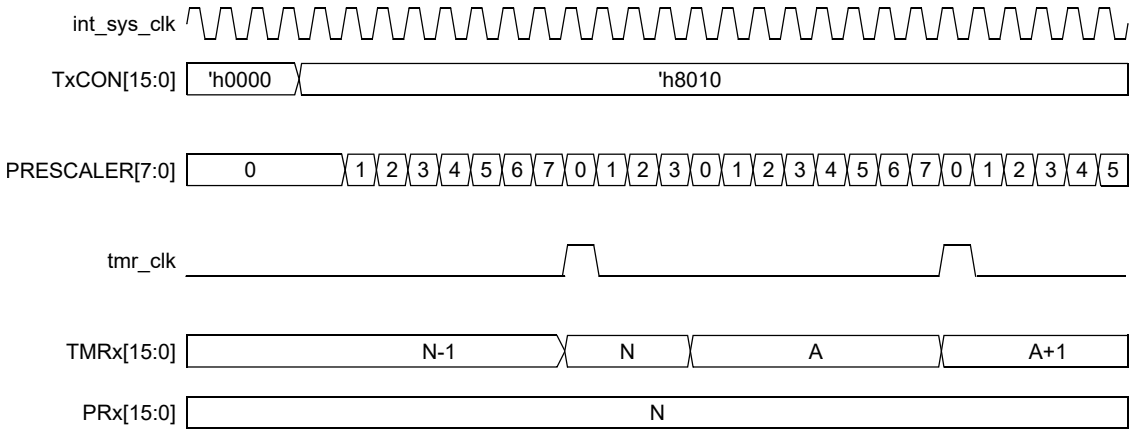


23.4.5.2 Timer Update and Slow Clock

When using the prescaler, the user software can change the value of the timer register faster than one timer period.

Figure 23-10 shows that when the counter is operating with an 8x prescale and TMRx is written before the end of the prescaler rollover, a period match is not detected and an interrupt is not generated. The user software can write a new value into the TMRx register before the timer resets to zero. In this case, the module will not assert the match signal and will continue counting with the new timer value on the next clock edge.

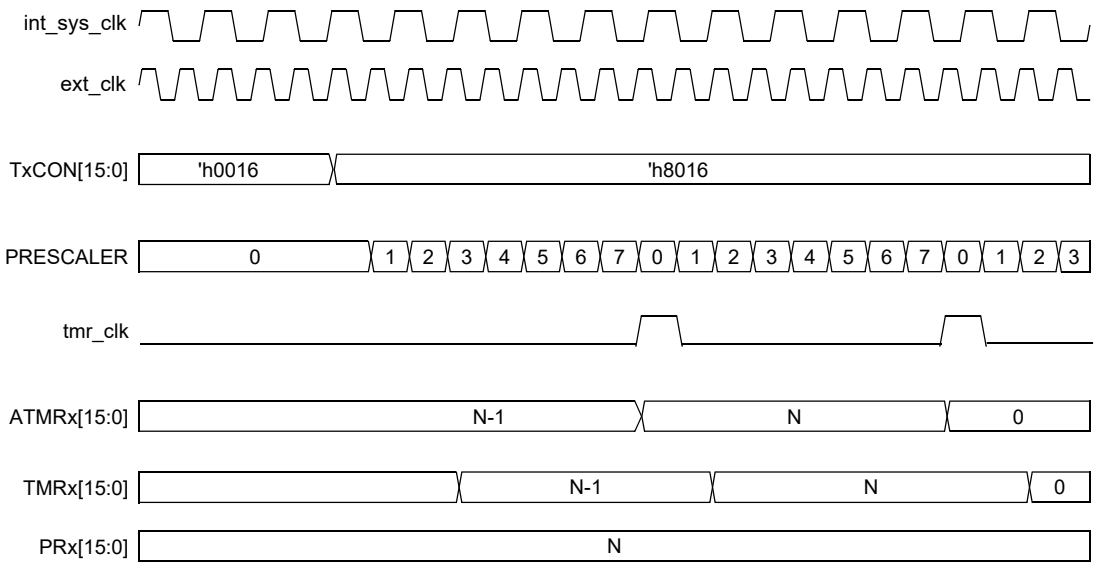
Figure 23-10. Synchronous Clock with Prescale with Write to TMRx Following Match (TxCON.TCS = 0, TxCON.TCKPS[1:0] = 01)



23.4.5.3 Prescaler with External Clock

In all modes, the prescaler enable signal is generated based on the int_sys_clk domain TxCON.TCS and TxCON.TGATE bit settings; therefore, these bits should not be modified while the timer is enabled to avoid clock domain crossing issues (refer to Figure 23-11).

Figure 23-11. Asynchronous Prescale External Clock Mode Timing Diagram



When a synchronized clock is chosen, whether the source is internal or external, the prescaler Reset logic will use the int_sys_clk. The prescaler Reset signal is generated by writes to TMRx using int_sys_clk.

When an asynchronous clock is chosen, the prescaler Reset logic will be generated from the external clock asynchronous domain. The prescaler Reset event will be generated by the timer write event, which is synchronous to the external clock domain.

23.4.6 Writing to T1CON, TMR1 and PR1 Registers

The Timer1 module is disabled and powered off when the TON bit (T1CON[15]) = 0, thus providing maximum power savings.

The timer register can be written while the module is operating. The bus write always has priority over the timer increment.

If 0xFFFFFFFF is written into the timer register, the next timer count clock after this write will cause the timer to roll over to 0x00000000.

To prevent unpredictable timer behavior, it is recommended that the timer be disabled before writing to any of the T1CON register bits or the timer input clock prescaler. Attempting to set the TON bit = 1 and writing to any T1CON register bits in the same instruction may cause an erroneous timer operation.

The PR1 Period register can be written to while the module is operating. However, to prevent unintended period matches, writing to the PR1 Period register while the timer is enabled (TON bit = 1) is not recommended.

The user must write the TxCON register to establish the operating mode prior to any updates to the TMRx register. The user is allowed to write the TMRx register while the timer is running. Writes to TMRx while the timer is running require the following synchronization sequence to be completed:

- Three timer domain clocks: two to sync the written data and TMRWIP bit and one to perform the

write into TMRx.

- An additional two system clocks after the write completes are needed to sync TMRx to the system domain

If the user attempts to write the timer again while the current write is awaiting synchronization, the value written to the timer can be corrupted.

The TMR1 Count register is not reset to zero when the module is disabled.

For read write operation on timer in external asynchronous mode refer to 23.4.4. Asynchronous Clock Counter Mode.

Two bits in the T1CON register can be used to ensure that writes during timer operation will not cause the timer value to be corrupted. The T1CON.TMWDIS bit, when set, will prevent a write to the timer and period registers when a previous write to the timer is awaiting synchronization into the asynchronous timer clock domain. The T1CON.TMWIP bit indicates when write synchronization is complete and it is safe to write another value to the timer.

In Asynchronous mode, writes are synchronized once the high byte is written, at which point the T1CON.PRWIP bit is set.

The T1CON.TMWDIS bit, when set, will prevent a write to the timer or period registers when a previous write to that register is awaiting synchronization into the asynchronous timer clock domain. The T1CON.TMWIP and T1CON.PRWIP bits indicate when write synchronization is complete and it is safe to write another value to the timer.

In synchronous mode, writes have immediate effect upon write to registers PR1 and TMR1. T1CON Status bits (TMWIP & PRWIP) will not update in synchronous mode. The T1CON.TMWDIS bit has no effect in synchronous mode.

Note: If a write to TMR is completed (and TMWIP cleared) on the same cycle where TMR = PR, an interrupt will still be generated and the timer will still read as '0' on the subsequent cycle due to the timing of the synchronization logic. Customer code that writes to the asynchronous timer while it is running should be capable of ignoring the unintended interrupt if necessary.

23.5 Interrupts

The Timer1 module has the ability to generate an interrupt on a period match or falling edge of the external gate signal, depending on the operating mode.

The T1IF bit is set when one of the following conditions is true:

- When the TMR1 count matches the respective PR1 register and the Timer1 module is not operating in Gated Time Accumulation mode
- When the falling edge of the gate signal is detected while the Timer1 is operating in Gated Time Accumulation mode

The T1IF bit must be cleared in software.

The Timer1 module is enabled as a source of interrupt via the Timer1 Interrupt Enable bit, T1IE. The Timer1 Interrupt Priority Level bits, T1IP[2:0], define the priority group to which the interrupt source will be assigned.

Note: A special case occurs when the PR1 register is loaded with '0' and the timer is enabled. An interrupt is not generated for this configuration. A Falling Edge Gate signal will not wake up the device from Sleep or Idle.

23.5.1 Interrupt Configuration

The Timer1 module has a dedicated Interrupt Flag bit (T1IF) and a corresponding \linterrupt Enable/ Mask bit (T1IE).

The T1IF bit is set when the timer count matches the respective period register and the timer module is not operational in Gated Time Accumulation mode. This bit is also set if the falling edge of the gate signal is detected when the timer is operating in Gated Time Accumulation mode. The T1IF

bit is set, regardless of the state of the corresponding T1IE bit. If required, the T1IF bit can be polled by software.

The T1IE bit is used to define the behavior of the interrupt controller when the T1IF bit is set. When the T1IE bit is clear, the interrupt controller does not generate a CPU interrupt for the event. If the T1IE bit is set, the interrupt controller generates an interrupt to the CPU when the T1IF bit is set (subject to the interrupt priority).

It is the responsibility of the user's software routine that services a particular interrupt to clear the appropriate interrupt flag bit before the service routine is complete.

The priority of the Timer1 module can be set with the T1IP[2:0] bits. This priority defines the priority group to which the interrupt source will be assigned. The priority groups range from a value of seven (the highest priority) to a value of 0 (which does not generate an interrupt). An interrupt being serviced will be preempted by an interrupt in a higher priority group.

After an enabled interrupt is generated, the CPU will jump to the vector assigned to that interrupt. The vector number for the interrupt is the same as the natural order number. The CPU will then begin executing code at the vector address. The user's code at this vector address should perform any application-specific operations and clear the T1IF interrupt flag and then exit.

23.6 Operation in Power-Saving Modes

23.6.1 Timer Operation in Sleep Mode

As the device enters Sleep mode, the system clock (SYSCLK) and peripheral clock are disabled.

The Timer1 timer can operate asynchronously from an external clock source. Therefore, the Timer1 module can continue to operate during Sleep mode.

To operate in Sleep mode, the Timer1 module is configured as follows:

- Timer1 module is enabled, TON bit (T1CON[15]) = 1
- Timer1 clock source is selected as external, TCS bit (T1CON[1]) = 1
- TSYNC bit (T1CON[2]) is set to a logic '0' (Asynchronous Counter mode enabled)

When these conditions are met, Timer1 continues to count and detect period matches when the device is in Sleep mode. When a match between the timer and the period register occurs, the T1IF status bit is set. If the T1IE bit is set, and its priority is greater than the current CPU priority, the device wakes from Sleep or Idle mode and executes the Timer1 Interrupt Service Routine.

If the assigned priority level of the Timer1 interrupt is less than or equal to the current CPU priority level, the CPU is not awakened and the device enters Idle mode.

23.6.2 Timer Operation in Idle Mode

When the device enters Idle mode, the system clock sources remain functional and the CPU stops executing code. The Timer1 module can optionally continue to operate in Idle mode.

The setting of the SIDL bit (T1CON[13]) determines whether the Timer1 module stops in Idle mode or continues to operate normally. If SIDL = 0, the Timer1 Module continues operation in Idle mode. If SIDL = 1, the Timer1 module stops in Idle mode.

23.7 Effects of Various Resets

23.7.1 Device Reset

All Timer1 registers are forced to their Reset states on a device Reset.

23.7.2 Power-on Reset (POR)

All Timer1 registers are forced to their Reset states on a Power-on Reset (POR).

23.7.3 Watchdog Timer Reset

All Timer1 registers are forced to their Reset states on a Watchdog Timer Reset.

24. Single-Output Capture/Compare/PWM/Timer Modules (SCCP)

This document describes the features and operation of the Single-Output CCP module. The module has been designed with the following goals in mind:

- Combine timebase, input capture, compare and PWM functions into a single peripheral
- Provide better edge resolution in PWM mode
- Provide the required functionality in the PWM mode to support a selected range of motor control, power supply and lighting applications

The following features are covered in this section:

- General Purpose Timer
- Input Capture
- Output Compare/PWM

24.1 Device-Specific Information

Table 24-1. SCCP Summary Table

Module Instances	Peripheral Bus Speed	Clock Source
4	Standard	See Table 24-2

Table 24-2. CLKSEL Time Base Clock Select bits

Value	Description
111	T1CK
110-010	Reserved
001	CLKGEN12
000	Standard Speed Peripheral Clock

Table 24-3. ICS Input Capture Source Select bits

Value	Description
111	CLC4 Out
110	CLC3 Out
101	CLC2 Out
100	CLC1 Out
011	CMP3 Out
010	CMP2 Out
001	CMP1 Out
000	SCCP Input Capture x (ICx) pin (PPS)

24.2 Architectural Overview

Select dsPIC33A family devices include one or more Capture/Compare/PWM/Timer (CCP) modules. The multipurpose timer module provides the functionality of the comparable Input Capture, Output Compare and General Purpose Timer peripherals found in all other devices.

CCP modules can operate in one of three major modes:

- **Time Base –** The module generates internal signals, triggers and interrupt events only based on a 16-bit or 32-bit count value. All associated I/O functions are disabled. The time base clock may be gated using one of the auto-shutdown/gating signal sources.
- **Input Capture –** The value of the 16-bit or 32-bit time base is written to a buffer on certain edge events received from a device input pin. Input Capture signal sources may be gated using one of the auto-shutdown/gating signal sources.

- **Output Compare –** A device output pin is set, reset, toggled or pulsed based on register values compared to the 16-bit or 32-bit time base. Register values may be buffered or non-buffered. The output compare signal can be steered to multiple output pins.

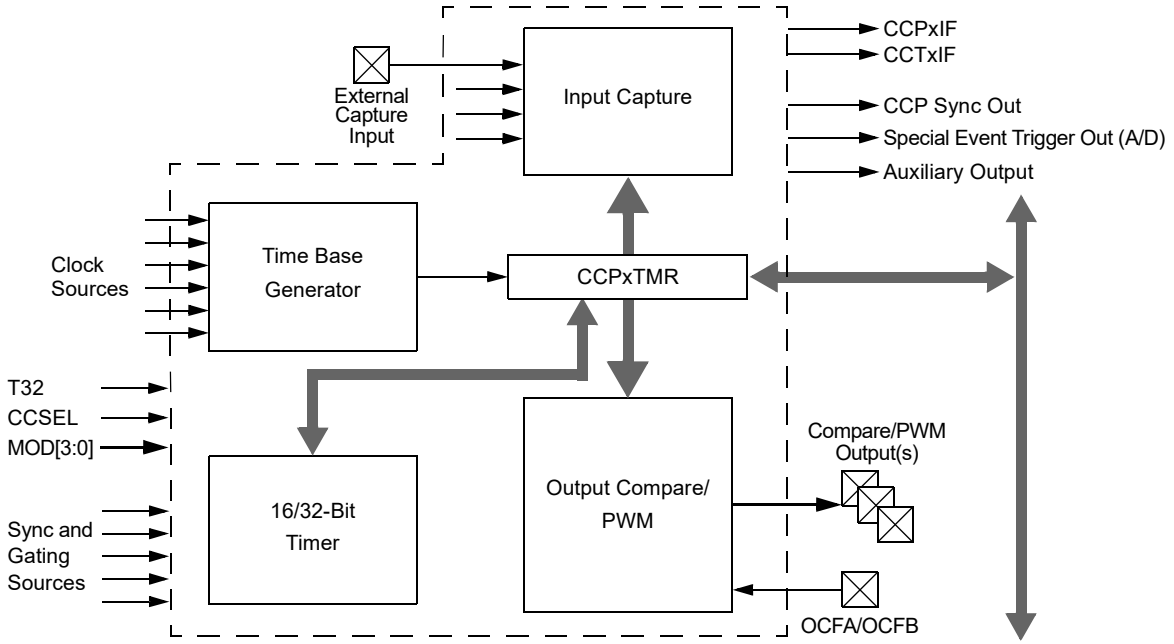
The SCCP module include these features:

- User-Selectable Clock Inputs, Including System Clock and External Clock Input Pins
- Input Clock Prescaler for Time Base
- Output Postscaler for Module Interrupt Events or Triggers
- Synchronization Output Signal for Coordinating Other SCCP Modules with User-Configurable Alternate and Auxiliary Source Options
- Fully Asynchronous Operation in All Modes and in Low-Power Operation
- Special Output Trigger for A/D Conversions
- 16-bit and 32-bit General Purpose Timer Modes with Optional Gated Operation for Simple Time Measurements
- Capture Modes:
  - Backward compatible with previous Input Capture peripherals of the dsPIC33/PIC24 families
  - 16-bit or 32-bit Capture of Time Base on external event
  - Up to four-level deep FIFO Capture buffer
  - Capture source input multiplexer
  - Gated Capture operation to reduce noise-induced false Captures
- Output Compare/PWM Modes:
  - Backward compatible with previous Output Compare peripherals of the dsPIC33/PIC24 families
  - Single Edge and Dual Edge Compare modes
  - External Input mode

The CCP module can be operated only in one of the three major modes (Capture, Compare or Timer) at any time. The other modes are not available unless the module is reconfigured.

A conceptual block diagram for the module is shown in Figure 24-1. All three modes use the Time Base Generator and the common Timer register (CCPxTMR). Other shared hardware components, such as comparators and buffer registers, are activated and used as a particular mode requires.

Figure 24-1. CCP Conceptual Block Diagram



24.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1B00	CCP1CON1	31:24	OPSSRC	RTRGEN			OPS[3:0]			
		23:16	TRIGEN	ONESHOT	ALTSYNC	SYNC[4:0]				
		15:8	ON		SIDL	SLPEN	TMRSYNC	CLKSEL[2:0]		
		7:0	TMRPS[1:0]		T32	CCSEL	MOD[3:0]			
0x1B04	CCP1CON2	31:24	OENSYNC							OCAEN
		23:16	ICGSM[1:0]			AUXOUT[1:0]		ICS[2:0]		
		15:8	PWMRSEN	ASDGM		SSDG				
		7:0	ASDG[7:0]							
0x1B08	CCP1CON3	31:24	OETRIG		OSCNT[2:0]					
		23:16			POLACE		PSSACE[1:0]			
		15:8								
		7:0								
0x1B0C	CCP1STAT	31:24								
		23:16				PRLWIP	TMRHWIP	TMRLWIP	RBWIP	RAWIP
		15:8						ICGARM		
		7:0	CCPTRIG	TRSET	TRCLR	ASEVT	SCEVT	ICDIS	ICOV	ICBNE
0x1B10	CCP1TMR	31:24	TMR[31:24]							
		23:16	TMR[23:16]							
		15:8	TMR[15:8]							
		7:0	TMR[7:0]							
0x1B14	CCP1PR	31:24	PR[31:24]							
		23:16	PR[23:16]							
		15:8	PR[15:8]							
		7:0	PR[7:0]							
0x1B18	CCP1RA	31:24								
		23:16								
		15:8	CMPA[15:8]							
		7:0	CMPA[7:0]							
0x1B1C	CCP1RB	31:24								
		23:16								
		15:8	CMPB[15:8]							
		7:0	CMPB[7:0]							
0x1B20	CCP1BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x1B24 ... 0x1B2F	Reserved									
0x1B30	CCP2CON1	31:24	OPSSRC	RTRGEN			OPS[3:0]			
		23:16	TRIGEN	ONESHOT	ALTSYNC	SYNC[4:0]				
		15:8	ON		SIDL	SLPEN	TMRSYNC	CLKSEL[2:0]		
		7:0	TMRPS[1:0]		T32	CCSEL	MOD[3:0]			
0x1B34	CCP2CON2	31:24	OENSYNC							OCAEN
		23:16	ICGSM[1:0]			AUXOUT[1:0]		ICS[2:0]		
		15:8	PWMRSEN	ASDGM		SSDG				
		7:0	ASDG[7:0]							
0x1B38	CCP2CON3	31:24	OETRIG		OSCNT[2:0]					
		23:16			POLACE		PSSACE[1:0]			
		15:8								
		7:0								
0x1B3C	CCP2STAT	31:24								
		23:16				PRLWIP	TMRHWIP	TMRLWIP	RBWIP	RAWIP
		15:8						ICGARM		
		7:0	CCPTRIG	TRSET	TRCLR	ASEVT	SCEVT	ICDIS	ICOV	ICBNE
0x1B40	CCP2TMR	31:24	TMR[31:24]							
		23:16	TMR[23:16]							
		15:8	TMR[15:8]							
		7:0	TMR[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1B44	CCP2PR	31:24	PR[31:24]							
		23:16	PR[23:16]							
		15:8	PR[15:8]							
		7:0	PR[7:0]							
0x1B48	CCP2RA	31:24								
		23:16								
		15:8	CMPA[15:8]							
		7:0	CMPA[7:0]							
0x1B4C	CCP2RB	31:24								
		23:16								
		15:8	CMPB[15:8]							
		7:0	CMPB[7:0]							
0x1B50	CCP2BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x1B54 ... 0x1B5F	Reserved									
0x1B60	CCP3CON1	31:24	OPSSRC	RTRGEN			OPS[3:0]			
		23:16	TRIGEN	ONESHOT	ALTSYNC	SYNC[4:0]				
		15:8	ON		SIDL	SLPEN	TMRSYNC	CLKSEL[2:0]		
		7:0	TMRPS[1:0]		T32	CCSEL	MOD[3:0]			
0x1B64	CCP3CON2	31:24	OENSYNC							OCAEN
		23:16	ICGSM[1:0]			AUXOUT[1:0]		ICS[2:0]		
		15:8	PWMRSEN	ASDGM		SSDG				
		7:0	ASDG[7:0]							
0x1B68	CCP3CON3	31:24	OETRIG		OSCNT[2:0]					
		23:16			POLACE		PSSACE[1:0]			
		15:8								
		7:0								
0x1B6C	CCP3STAT	31:24								
		23:16				PRLWIP	TMRHWIP	TMRLWIP	RBWIP	RAWIP
		15:8						ICGARM		
		7:0	CCPTRIG	TRSET	TRCLR	ASEVT	SCEVT	ICDIS	ICOV	ICBNE
0x1B70	CCP3TMR	31:24	TMR[31:24]							
		23:16	TMR[23:16]							
		15:8	TMR[15:8]							
		7:0	TMR[7:0]							
0x1B74	CCP3PR	31:24	PR[31:24]							
		23:16	PR[23:16]							
		15:8	PR[15:8]							
		7:0	PR[7:0]							
0x1B78	CCP3RA	31:24								
		23:16								
		15:8	CMPA[15:8]							
		7:0	CMPA[7:0]							
0x1B7C	CCP3RB	31:24								
		23:16								
		15:8	CMPB[15:8]							
		7:0	CMPB[7:0]							
0x1B80	CCP3BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x1B84 ... 0x1B8F	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1B90	CCP4CON1	31:24	OPSSRC	RTRGEN			OPS[3:0]			
		23:16	TRIGEN	ONESHOT	ALTSYNC	SYNC[4:0]				
		15:8	ON		SIDL	SLPEN	TMRSYNC	CLKSEL[2:0]		
		7:0	TMRPS[1:0]		T32	CCSEL	MOD[3:0]			
0x1B94	CCP4CON2	31:24	OENSYNC							OCAEN
		23:16	ICGSM[1:0]			AUXOUT[1:0]		ICS[2:0]		
		15:8	PWMRSEN	ASDGM		SSDG				
		7:0	ASDG[7:0]							
0x1B98	CCP4CON3	31:24	OETRIG		OSCNT[2:0]					
		23:16			POLACE		PSSACE[1:0]			
		15:8								
		7:0								
0x1B9C	CCP4STAT	31:24								
		23:16				PRLWIP	TMRHWIP	TMRLWIP	RBWIP	RAWIP
		15:8						ICGARM		
		7:0	CCPTRIG	TRSET	TRCLR	ASEVT	SCEVT	ICDIS	ICOV	ICBNE
0x1BA0	CCP4TMR	31:24	TMR[31:24]							
		23:16	TMR[23:16]							
		15:8	TMR[15:8]							
		7:0	TMR[7:0]							
0x1BA4	CCP4PR	31:24	PR[31:24]							
		23:16	PR[23:16]							
		15:8	PR[15:8]							
		7:0	PR[7:0]							
0x1BA8	CCP4RA	31:24								
		23:16								
		15:8	CMPA[15:8]							
		7:0	CMPA[7:0]							
0x1BAC	CCP4RB	31:24								
		23:16								
		15:8	CMPB[15:8]							
		7:0	CMPB[7:0]							
0x1BB0	CCP4BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							

24.3.1 CCPx Control Register 1

Name: CCPxCON1
Offset: 0x1B00, 0x1B30, 0x1B60, 0x1B90

Notes:

1. Control bit has no function in Input Capture modes.
2. Control bit has no function when TRIGEN = 0.
3. Values greater than '0011' will cause a FIFO buffer overflow in Input Capture mode.
4. See Table 24-11 for the definition of Sync inputs.
5. 32-bit operation is not available in Dual Edge Output Compare modes.

Bit	31	30	29	28	27	26	25	24
	OPSSRC	RTRGEN			OPS[3:0]			
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRIGEN	ONESHOT	ALTSYNC	SYNC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	SLPEN	TMRSYNC	CLKSEL[2:0]		
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TMRPS[1:0]		T32	CCSEL	MOD[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – OPSSRC Output Postscaler Source Select bit(1)

Value	Description
1	Output postscaler scales module trigger output events
0	Output postscaler scales time base interrupt events

Bit 30 – RTRGEN Retrigger Enable bit(2)

Value	Description
1	Time base can be retriggered when CCPTRIG (CCPxSTAT[7]) = 1
0	Time base may not be retriggered when CCPTRIG (CCPxSTAT[7]) = 1

Bits 27:24 – OPS[3:0] Time Base Interrupt/Input Capture Event Postscale Select bits(3)

Value	Description
1111	Interrupt CPU every 16th time base period match
1110	Interrupt CPU every 15th time base period match
. . .	
0100	Interrupt CPU every 5th time base period match
0011	Interrupt CPU every 4th time base period match or after four input capture events
0010	Interrupt CPU every 3rd time base period match or after three input capture events
0001	Interrupt CPU every 2nd time base period match or after two input capture events

Value	Description
0000	Interrupt CPU after each time base period match or each input capture event

Bit 23 – TRIGEN CCPx Trigger Enable bit

Value	Description
1	Trigger operation of time base is enabled
0	Trigger operation of time base is disabled

Bit 22 – ONESHOT One-Shot Mode Enable bit

Value	Description
1	One-Shot Trigger mode enabled; trigger duration set by the OSCNT[2:0] (CCPxCON3[30:28]) bits
0	One-Shot Trigger mode disabled

Bit 21 – ALTSYNC Synchronization Output Select bit

Value	Description
1	An alternate signal is used as the module synchronization output signal (see Table 24-9)
0	The module synchronization output signal is the time base Reset/rollover event

Bits 20:16 – SYNC[4:0] Capture/Compare/PWM Synchronization Source Select bits(4)
Refer to Table 24-11 for synchronization sources.

Value	Description
11111	Time base is in the Free-Running mode and rolls over at FFFF
11110	Time base is synchronized to source #30
. . .	
00001	Time base is synchronized to source #1
00000	Time base is self-synchronized and rolls over at FFFF or a match with the Period register

Bit 15 – ON Module Enable bit

Value	Description
1	Module is enabled with the operating mode specified by the MOD[3:0] control bits
0	Module is disabled

Bit 13 – SIDL Stop in Idle Mode bit

Value	Description
1	Discontinues module operation when device enters Idle mode
0	Continues module operation in Idle mode

Bit 12 – SLPEN Sleep Mode Enable bit

Value	Description
1	Module continues to operate in Sleep modes
0	Module does not operate in Sleep modes

Bit 11 – TMRSYNC Time Base Clock Synchronization bit

Value	Description
1	Module time base clock is synchronized to internal system clocks; timing restrictions apply
0	Module time base clock is not synchronized to internal system clocks

Bits 10:8 – CLKSEL[2:0] Time Base Clock Select bits
See Table 24-2.

Bits 7:6 – TMRPS[1:0] Capture/Compare/PWMx Time Base Prescale Select bits

Value	Description
11	1:64 prescaler
10	1:16 prescaler
01	1:4 prescaler
00	1:1 prescaler

Bit 5 – T32 32-Bit Time Base Select bit(1,5)

Value	Description
1	Uses 32-bit time base for selected Timer, Single Edge Output Compare or Input Capture function
0	Uses 16-bit time base for selected Timer, Single Edge Output Compare or Input Capture function

Bit 4 – CCSEL Capture/Compare Mode Select bit(1)

Value	Description
1	Module operates as an Input Capture peripheral
0	Module operates as an Output Compare peripheral

Bits 3:0 – MOD[3:0] CCP Mode Select bits(1)

Value	Description
CCSEL	1 (Input Capture modes)
1xxx	Reserved
0111	Reserved
0110	Reserved
0101	Capture every 16th rising edge
0100	Capture every 4th rising edge
0011	Capture every rising and falling edge
0010	Capture every falling edge
0001	Capture every rising edge
0000	Capture every rising and falling edge (Edge Detect mode)
CCSEL	0 (Output Compare modes)
1111	External Input mode, generator disabled; source selected by the ICS[2:0] bits
1110	Reserved
1101	Reserved
1100	Reserved
1011	Reserved
1010	Reserved
1001	Reserved
1000	Reserved
0111	Reserved
0110	Reserved
0101	Dual Edge Compare mode – Buffered
0100	Dual Edge Compare mode
0011	16-bit/32-bit Single Edge mode – toggle output on compare match
0010	16-bit/32-bit Single Edge mode – drive output low on compare match
0001	16-bit/32-bit Single Edge mode – drive output high on compare match
0000	16-bit/32-bit Timer mode – output functions disabled

24.3.2 CCPx Control Register 2

Name: CCPxCON2
Offset: 0x1B04, 0x1B34, 0x1B64, 0x1B94

Notes:

1. This bit has no effect in Timer modes, Output Compare modes or PWM modes. A write to the ICGARM (CCPxSTAT[10]) bit will re-arm the one-shot gating circuit when ICGSM = `01` or ICGSM = `10`.
2. This bit has no affect for Timer gating or Input Capture gating functions.

Bit	31	30	29	28	27	26	25	24
	OENSYNC							OCAEN
Access	R/W							R/W
Reset	0							0

Bit	23	22	21	20	19	18	17	16
	ICGSM[1:0]			AUXOUT[1:0]		ICS[2:0]		
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PWMRSEN	ASDGM		SSDG				
Access	R/W	R/W		R/W				
Reset	0	0		0				

Bit	7	6	5	4	3	2	1	0
	ASDG[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – OENSYNC Output Enable Synchronization bit

Value	Description
1	Update by output enable bits occurs on the next time base Reset or rollover
0	Update by output enable bits occurs immediately

Bit 24 – OCAEN Output Enable/Steering Control bits

Value	Description
1	OCx pin is controlled by the CCP module and produces an Output Compare or PWM signal
0	OCx pin is not controlled by the CCP module; the pin is available to the port logic or another peripheral multiplexed on the pin

Bits 23:22 – ICGSM[1:0] Input Capture Gating Source Mode Control bits(1)

Value	Description
11	Reserved
10	One-Shot mode; falling edge from gating source will disable future capture events (CDIS = 1)
01	One-Shot mode; rising edge from gating source will enable future capture events (CDIS = 0)
00	Level Sensitive mode; a high level from gating source will enable future capture events; a low level will disable future capture events

Bits 20:19 – AUXOUT[1:0] Auxiliary Output Signal Selection bits

Value	Description
11	Signal output depends on module operating mode (see Table 24-10)
10	Signal output depends on module operating mode (see Table 24-10)
01	Signal output depends on module operating mode (see Table 24-10)
00	No signal output on aux_out

Bits 18:16 – ICS[2:0] Input Capture Source Select bits
See Table 24-3.

Bit 15 – PWMRSEN CCPx Output Compare Restart Enable bit

Value	Description
1	ASEVT (CCPxSTAT[4]) bit clears automatically at the beginning of the next Output Compare period, after the shutdown input has ended
0	ASEVT (CCPxSTAT[4]) bit must be cleared in software to resume Output Compare activity on output pins

Bit 14 – ASDGM CCPx Auto-Shutdown/Gate Control bit(1)

Value	Description
1	Wait until next time base Reset or rollover for an Output Compare pin shutdown to occur
0	Output Compare pin shutdown event occurs immediately

Bit 12 – SSDG CCPx Software Shutdown/Gate Control bit

Value	Description
1	Manually force auto-shutdown, Timer clock gate or Input Capture signal gate event (setting of ASDGM bit still applies)
0	Normal module operation

Bits 7:0 – ASDG[7:0] CCPx Auto-Shutdown/Gating Source Enable bits
Refer to Table 24-8 for Auto-Shutdown and Gating Sources.

Value	Description
1	ASDG source *n* is enabled
0	ASDG source *n* is disabled

24.3.3 CCPx Control Register 3

Name: CCPxCON3
Offset: 0x1B08, 0x1B38, 0x1B68, 0x1B98

Note:
1. ONESHOT (CCPxCON1[22]) must be set for the OSCNT[2:0] bits to be effective.

Bit	31	30	29	28	27	26	25	24
	OETRIG	OSCNT[2:0]						
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bit	23	22	21	20	19	18	17	16
			POLACE		PSSACE[1:0]			
Access			R/W		R/W	R/W		
Reset			0		0	0		

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 31 – OETRIG Output Enable on Trigger Control bit

Value	Description
1	For Triggered mode (TRIGEN = 1), module does not drive enabled output pins until triggered
0	Normal output pin operation

Bits 30:28 – OSCNT[2:0] One-Shot Count bits(1)

Value	Description
111	Extend one-shot trigger event 7 time base count cycles (8 time base periods total)
110	Extend one-shot trigger event 6 time base count cycles (7 time base periods total)
101	Extend one-shot trigger event 5 time base count cycles (6 time base periods total)
100	Extend one-shot trigger event 4 time base count cycles (4 time base periods total)
011	Extend one-shot trigger event 3 time base count cycles (4 time base periods total)
010	Extend one-shot trigger event 2 time base count cycles (3 time base periods total)
001	Extend one-shot trigger event 1 time base count cycle (2 time base periods total)
000	Do not extend one-shot trigger event

Bit 21 – POLACE CCP Output Pin, OCxA, Polarity Control bit

Value	Description
1	Output pin polarity is active-low
0	Output pin polarity is active-high

Bits 19:18 – PSSACE[1:0] PWM Output Pin, OCxA, Shutdown State Control bits

Value	Description
11	Pins are driven active when a shutdown event occurs

Value	Description
10	Pins are driven inactive when a shutdown event occurs
0x	Pins are tri-stated when a shutdown event occurs

24.3.4 CCPx Status Register

Name: CCPxSTAT
Offset: 0x1B0C, 0x1B3C, 0x1B6C, 0x1B9C

Notes:

1. This is not a physical bit location and will always read as '0'. A write of '1' will initiate the hardware event.
2. This bit has no effect when CLKSEL[2:0] (CCPxCON1[10:8]) = 000 or TMRSYNC = 1.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
				PRLWIP	TMRHWIP	TMRLWIP	RBWIP	RAWIP
Access				R	R	R	R	R
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
						ICGARM		
Access						W		
Reset						0		

Bit	7	6	5	4	3	2	1	0
	CCPTRIG	TRSET	TRCLR	ASEVT	SCEVT	ICDIS	ICOV	ICBNE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 20 – PRLWIP CCPxPRL Write in Progress Status bit(2)

Value	Description
1	An update to the CCPxPRL register with the buffered contents is in progress
0	An update to the CCPxPRL register is not in progress

Bit 19 – TMRHWIP CCPxTMRH Write in Progress Status bit(2)

Value	Description
1	An update to the CCPxTMRH register with the buffered contents is in progress
0	An update to the CCPxTMRH register is not in progress

Bit 18 – TMRLWIP CCPxTMRL Write in Progress Status bit(2)

Value	Description
1	An update to the CCPxTMRL register with the buffered contents is in progress
0	An update to the CCPxTMRL register is not in progress

Bit 17 – RBWIP CCPxRB Write in Progress Status bit(2)

Value	Description
1	An update to the CCPxRB register with the buffered contents is in progress
0	An update to the CCPxRB register is not in progress

Bit 16 – RAWIP CCPxRA Write in Progress Status bit(2)

Value	Description
1	An update to the CCPxRA register with the buffered contents is in progress
0	An update to the CCPxRA register is not in progress

Bit 10 – ICGARM Input Capture Gate Arm bit(1)

A write of '1' to this location will arm the Input Capture gating logic for a one-shot gate event when ICGSM[1:0] = 01 or 10. Bit location reads as '0' (see Table 24-8).

Bit 7 – CCPTRIG CCPx Trigger Status bit

Value	Description
1	Timer has been triggered and is running
0	Timer has not been triggered and is held in Reset

Bit 6 – TRSET CCPx Trigger Set Request bit(1)

A write of '1' to this location will request a trigger of the time base when TRIGEN = 1. The bit will clear automatically after the trigger event has been generated, allowing a new trigger event to be requested.

Bit 5 – TRCLR CCPx Trigger Clear Request bit(1)

A write of '1' to this location will request a trigger cancellation when TRIGEN = 1 and CCPTRIG = 1. Bit clears automatically after the cancellation has completed, allowing a new cancellation to be requested.

Bit 4 – ASEVT CCPx Auto-Shutdown Event Status/Control bit

Value	Description
1	A shutdown event is in progress; CCP outputs are in the Shutdown state
0	CCP outputs operate normally

Bit 3 – SCEVT Single Edge Compare Event Status bit

Value	Description
1	A single edge Compare event has occurred
0	A single edge Compare event has not occurred

Bit 2 – ICDIS Input Capture Disable bit

Value	Description
1	Event on Input Capture pin does not generate a capture event
0	Event on Input Capture pin will generate a capture event

Bit 1 – ICOV Input Capture Buffer Overflow Status bit

Value	Description
1	The Input Capture FIFO buffer has overflowed
0	The Input Capture FIFO buffer has not overflowed

Bit 0 – ICBNE Input Capture Buffer Status bit

Value	Description
1	Input Capture buffer has data available
0	Input Capture buffer is empty

24.3.5 CCPx Time Base Register

Name: CCPxTMR
Offset: 0x1B10, 0x1B40, 0x1B70, 0x1BA0

Notes:

1. TMR[31:16] will be available when operating in a valid 32-bit Operating mode, or the dual 16-bit Time Base mode. TMR[31:16] will read as '0' when operating in all other modes.
2. All writes to CCPxTMR are buffered for atomic update operation. The CCPxTMR value is not updated until the uppermost byte of the timer is written. If the timer clock source is asynchronous, user software must monitor the status bits to ensure the prior write has completed before performing another write.
3. TMRL = TMR[15:0] and TMRH = TMR[31:16].

Bit	31	30	29	28	27	26	25	24
	TMR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TMR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TMR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TMR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – TMR[31:0] 32-bit Time Base Value bits(1,2,3)

24.3.6 CCPx Period Register

Name: CCPxPR
Offset: 0x1B14, 0x1B44, 0x1B74, 0x1BA4

Notes:

1. For Dual 16-bit Timer mode, the PR[31:16] bits set the count period for the second 16-bit Timer. For 32-bit Timer operation, the PR[31:0] bits set the count period for the single 32-bit Timer. On a device Reset, the module will reset to a Dual 16-bit Timer mode. The CCPxPR Reset value of FFFFFFFF provides the maximum count period for both timers. The PR[31:16] bits are not available in 16-bit Output Compare modes and will read as '0'. The PR[31:0] bits are not available in 32-bit Output Compare modes and will read as '0'.
2. PRL = PR[15:0] and PRH = PR[31:16].

Bit	31	30	29	28	27	26	25	24
	PR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PR[31:0] Period Register bits(1,2)

24.3.7 CCPx Primary Compare Register (Timer/Compare Modes Only)

Name: CCPxRA
Offset: 0x1B18, 0x1B48, 0x1B78, 0x1BA8

Note:

1. For operating the module in 32-bit modes, the CCPxRA register contains the lower 16 bits to be compared to the time base.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CMPA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CMPA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – CMPA[15:0] Compare Value bits(1)
The 16-bit value to be compared against the CCP time base.

24.3.8 CCPx Secondary Compare Register (Timer/Compare Modes Only)

Name: CCPxRB
Offset: 0x1B1C, 0x1B4C, 0x1B7C, 0x1BAC

Note:

1. For operating the module in 32-bit modes, the CCPxRB register contains the upper 16 bits to be compared to the time base. In certain 16-bit modes, the CCPxRB register sets the module output trigger time.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CMPB[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CMPB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – CMPB[15:0] Compare Value bits(1)
The 16-bit value to be compared against the CCP time base.

24.3.9 CCPx Capture Buffer Register (Compare Modes Only)

Name: CCPxBUF
Offset: 0x1B20, 0x1B50, 0x1B80, 0x1BB0

Notes:

1. The CCPxBUF[31:16] bits are only available in 32-bit Input Capture modes. BUF[31:16] will read as '0' when operating in a 16-bit Input Capture mode.
2. CCPxBUF[31:0] will read as '0' for all Timer, Output Compare and PWM operating modes.
3. BUFL = BUF[15:0] and BUFH = BUF[31:16].

Bit	31	30	29	28	27	26	25	24
	BUF[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BUF[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BUF[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BUF[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – BUF[31:0] Capture Buffer Value bits(1,2,3)
Indicates the oldest captured time base value in the FIFO.

24.4 Operation

24.4.1 Time Base Generator

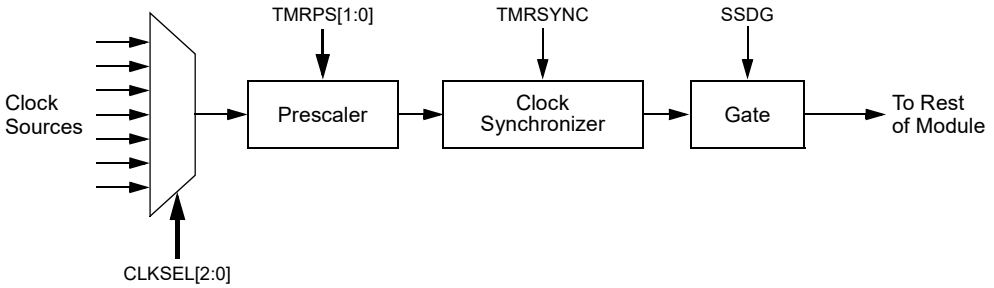
The Time Base Generator (TBG) provides a time base for the rest of the module using clock signals available on the microcontroller. This serves not only as the time base for the Timer modes, but also allows Input Capture and Output Compare pre-modes to operate without depending on another on-chip timer module.

Up to eight clock inputs are available to the clock generator, including the system clock (T_{CY}) and other on-chip oscillator sources. Depending on the device, external clock inputs may also be available. A prescaler divides the selected clock source to a suitable frequency for use by the module.

The TBG has the ability to synchronize its operation with the selected clock source, subject to input timing restrictions or the module's operating conditions. Setting the TMRSYNC bit (CCPxCON1[11]) enables synchronization of the time base with the clock input.

The TBG is shown in Figure 24-2.

Figure 24-2. Time Base Clock Generator



24.4.1.1 Gating Logic

The Time Base Generator incorporates a hardware gate that can disable the timer increment clock to the timer gate, which is available on Timer modes only.

Gating is controlled using the ASDG[7:0] control bits (CCPxCON2[7:0]) and the SSDG bit (CCPxCON2[12]). All of these bits are logically ORed together to generate a gating enable signal for the TBG.

Setting any one of the ASDGx bits enables its corresponding hardware trigger; any or all of the bits may be set to select multiple sources. The available sources for gating and auto-shutdown are device-dependent and typically include such sources as comparator outputs, I/O pins (including OCFA and OCFB for PWM operation), software control and so on. Any output signal from any of the enabled sources disables the TBG output. Events are generally level-sensitive and not edge-triggered.

The SSDG bit is simply a gating source that can be manipulated in software. Setting SSDG has the same effect as an input from any of the hardware sources.

The gating feature is described in the following sections:

- Timer Gating (see 24.4.2.3. Clock Gating for Timer Modes)
- Auto-Shutdown for Output Compare (see 24.4.4.4.4. Auto-Shutdown Control)
- Gated Input Capture (see 24.4.3.4.1. Input Capture Signal Gating)

Regardless of the operating mode, interrupt events are not generated by the CCP module based on the status of the gating inputs. If an interrupt is required for a gating event, the gating source itself must be used to generate the interrupt.

24.4.2 Timer Mode

When CCSEL = 0 and MOD[3:0] = 0000, the module functions as a timer. There are two basic Timer modes, selected by the T32 bit (CCPxCON1[5]); these are shown in Table 24-4. In either mode, the timer can operate as a free-running timer/counter, operate synchronously with other modules, or be triggered by other modules or external events.

Table 24-4. Timer Operating Modes

T32	Operating Mode
0	Dual Timer Mode (16-bit)
1	Timer Mode (32-bit)

24.4.2.1 Dual 16-Bit Timer Mode

Dual 16-Bit Timer mode is selected when T32 = 0. This mode is useful for the following functions:

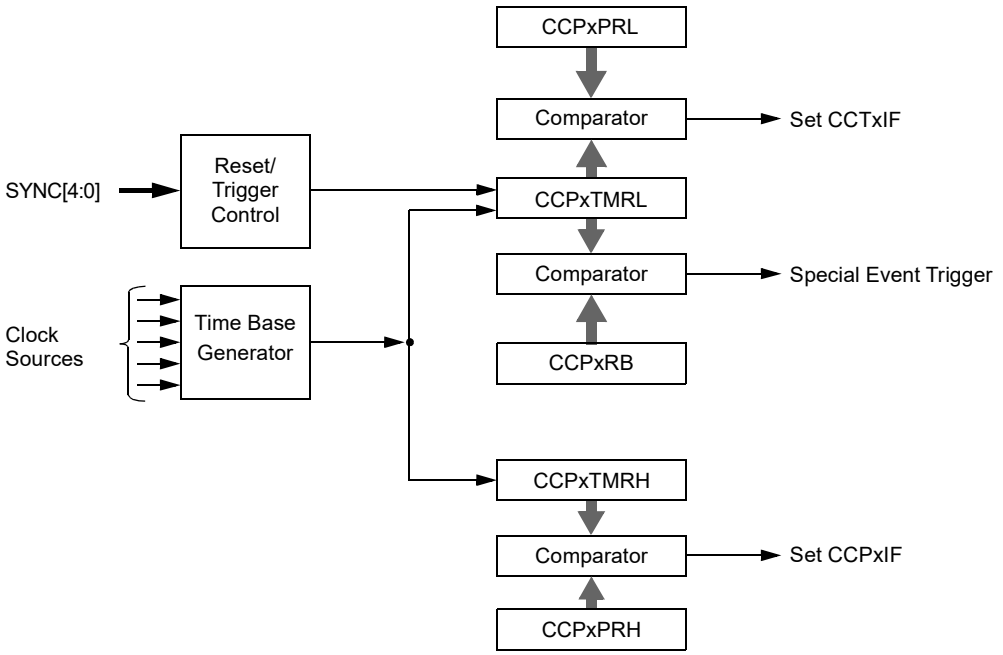
- CCPxTMR Periodic CPU Interrupts

- Master Time Base Function for Synchronizing Other CCP Modules
- Triggering Periodic A/D Conversion
- Periodic Wake from Sleep (if an Appropriate Clock Source is Available)

Note: The CCPxTMRH/L register bit locations may not be readable by the user if a high-speed asynchronous clock source is used to clock the time base. For a low-speed read, a double read can be done and the results compared.

Dual 16-Bit Timer mode provides a simple timer function with two independent 16-bit timer/counters, as shown in Figure 24-3. The primary timer, based on the lower word of the CCPxTMR, is fully functional and can interact with other modules on the device. It can generate the CCP Sync signals for use by other CCP modules. It can also use the SYNC[4:0] signal generated by other modules. The secondary timer, based on the upper word of CCPxTMR, has limited functionality. It is intended to be used only as a periodic interrupt source for scheduling CPU events. It does not generate an output trigger signal like the primary time base.

Figure 24-3. 16-Bit Dual Timer Mode



Both the primary and secondary timers use the same clock source from the TBG, as selected by CLKSEL[2:0]. The CCPxTMRH/L register bit locations provide user access to the two 16-bit time bases. Both timer register bit locations (CCPxTMRL and CCPxTMRH) increment at the same time based on the timer input; however, only the primary timer (CCPxTMRL) can use the timer Sync functionality. The secondary timer (CCPxTMRH) does not have timer Sync functionality.

The CCPxPRL register bit locations control the period for the primary 16-bit time base when SYNC[4:0] = 00000. When the module is configured to use an external synchronization source, the primary 16-bit time base is reset when the source selected by SYNC[4:0] is asserted. The module's Sync signal is generated whenever the time base rolls over or is reset to '0'.

The primary timer can generate the CCP interrupt when the value of CCPxTMRL resets to 0000h. When SYNC[4:0] = 00000, this occurs when CCPxTMRL matches CCPxPRL. If SYNC[4:0] is not '00000', CCPxTMRL resets and generates a CCT Interrupt Flag (CCTxIF) event whenever the signal selected by SYNC[4:0] is asserted.

The CCPxPRH register bit locations control the count period of the secondary 16-bit timer. The secondary timer does not support external synchronization and is not affected by the selected SYNC[4:0] input. The secondary time base begins counting when the CCPON bit (CCPxCON1[15]) is set. When a match occurs between the CCPxPRH register bit locations and the CCPxTMRH count value, the secondary 16-bit time base is reset and a timer rollover interrupt event (CCPxIF) is generated.

If either of the 16-bit timers is not used in the application, the timer can be disabled by writing 0000h to the corresponding period register. The timer is held in Reset, and no interrupts are generated as long as the period register's value is '0'. The CCPxPRH and CCPxPRL register bit locations are not buffered in this operating mode.

To use the module in Dual 16-Bit Timer mode:

1. Set CCSEL = 0 to select the Time Base/Output Compare mode of the module.
2. Set T32 = 0 to select the 16-bit time base operation.
3. Set MOD[3:0] = 0000 to select the Time Base mode.
4. Set SYNC[4:0] to the desired time base synchronization source:
   - Configure and enable the external source selected by SYNC[4:0] before enabling the timer.
   - If the timer is not using an external Sync source (SYNC[4:0] = 00000), or if the module is synchronizing to itself (the SYNC[4:0] bits select the module's own value as a Sync source), write the desired count period of the primary 16-bit time base to CCPxPRL.
5. If the secondary timer is also being used, write a non-zero value to CCPxPRH to specify the count period.
6. If the special A/D trigger is being used, set CCPxRB for the desired trigger output time.
7. Enable the module by setting the CCPON bit.
8. If an external synchronization source is selected in step four, configure and enable that source to allow the primary 16-bit time base to begin counting.

24.4.2.1.1 Special Event Trigger

In select devices, the Dual 16-Bit Timer mode can be used to generate a Special Event Trigger output signal. The primary timer can be used to start A/D conversions and trigger other peripheral events. The trigger period is set by the value of the CCPxRB register and must be less than the counter period, as defined by the CCPxPRL register bit locations.

24.4.2.2 32-Bit Timer Mode

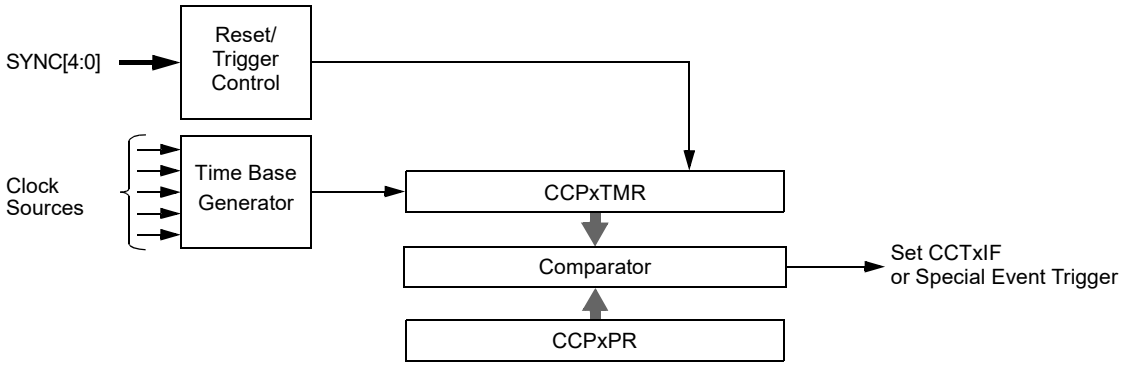
The 32-Bit Timer mode is selected when T32 = 1. In this mode, CCPxTMR will be used as single 32-bit register.

This mode provides a simple timer function when it is important to track long time periods. It is useful for the following functions:

- Periodic CPU Interrupts
- Synchronization and Trigger Generation for Other CCP Modules
- Periodic ADC Conversion Triggering
- Periodic Wake from Sleep (if an Appropriate Clock Source is Available)

No input or output functions are available from the CCP module in this operating mode.

Figure 24-4. 32-Bit Timer Mode



When external synchronization is not selected (SYNC[4:0] = 00000), the CCPxPR register sets the count period for the timer. A match between the CCPxTMR and the CCPxPR register also automatically generates the Sync output signal whenever the module is enabled (CCPON = 1).

To use the module in 32-Bit Timer mode:

1. Set CCSEL = 0 to select the Time Base/Output Compare mode of the module.
2. Set T32 = 1 to select the 32-bit time base operation.
3. Set MOD[3:0] = 0000 to select the Time Base mode.
4. Set SYNC[4:0] to the desired timer synchronization source:
   - Configure and enable the external source selected by SYNC[4:0] before enabling the timer.
   - If the timer is not using an external Sync source (SYNC[4:0] = 00000), or if the module is synchronizing to itself (SYNC[4:0] selects the module's own value as a Sync source), write the desired count period to CCPxPR.
5. Enable the module by setting the CCPON bit.

24.4.2.3 Clock Gating for Timer Modes

When operating in Timer mode, time base gating can be used to gate the timer's operation (see 24.4.1.1. Gating Logic for more information). This function provides a simple way to measure the time of an external event. Timer clock gating is enabled whenever one or more of the ASDG[7:0] bits (CCPxCON2[7:0]) is set or when the SSDG bit (CCPxCON2[12]) is set.

24.4.3 Input Capture Mode

When CCSEL = 1, the module is configured for Input Capture mode. This mode is used to capture a timer value from an independent timer base on the occurrence of an event on an input pin. This mode is useful in applications requiring frequency (time period) and pulse measurement.

Input Capture mode uses the CCPxTMR registers as a dedicated 16/32-bit synchronous, up counting timer used for event capture. This value is written to the FIFO buffer when a capture event occurs. The internal value may also be read with a synchronization delay from the CCPxTMR register.

Input Capture mode is the only major mode available when CCSEL is set. The T32 and the MOD[3:0] bits determine the various Capture modes, as shown in Table 24-5.

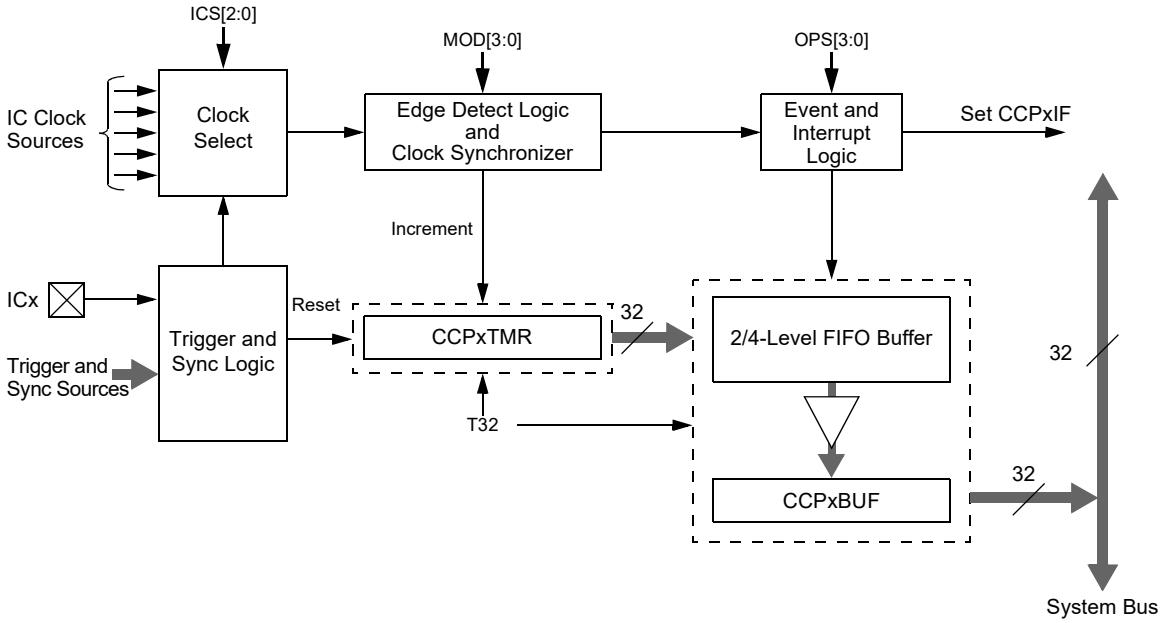
Figure 24-5 provides a simplified block diagram of the Input Capture mode.

Table 24-5. Capture Modes

T32 (CCPxCON1[15])	MOD[3:0] (CCPxCON1[3:0])	Operating Mode
0	0000	Edge Detect (16-bit capture)

.....continued		
T32 (CCPxCON1[15])	MOD[3:0] (CCPxCON1[3:0])	Operating Mode
1	0000	Edge Detect (32-bit capture)
0	0001	Every Rising (16-bit capture)
1	0001	Every Rising (32-bit capture)
0	0010	Every Falling (16-bit capture)
1	0010	Every Falling (32-bit capture)
0	0011	Every Rise/Fall (16-bit capture)
1	0011	Every Rise/Fall (32-bit capture)
0	0100	Every 4th Rising (16-bit capture)
1	0100	Every 4th Rising (32-bit capture)
0	0101	Every 16th Rising (16-bit capture)
1	0101	Every 16th Rising (32-bit capture)

Figure 24-5. Input Capture Block Diagram



24.4.3.1 Initialization

Since the module can be used for Input Capture/Output Compare/PWM, selecting the correct operation required should be the first task. The best practice is to clear all the associated control registers.

When the CCP module is reset or disabled (CCPON = 0):

- The ICOV and ICBNE status flags are cleared
- CCPxBUFH/L and their FIFO buffer are cleared
- CCPxTMRH/L are reset to zero
- The capture prescaler counter is reset to zero
- The capture event counter for interrupt generation is reset to zero

24.4.3.1.1 Mode Selection

As with Timer and Output Capture/PWM modes, the MOD[3:0] bits selects the Capture mode and prescaler options. To avoid inadvertent interrupts, always disable the module by clearing the CCPON bit when changing Capture modes. It is recommended to set the CCSEL bit and configure the MOD[3:0] bits in a single operation before enabling the module.

24.4.3.1.2 Timer Clock Source Selection

dsPIC33A family devices may have one or more Input Capture channels. Each channel can select between one of eight clock sources for its time base by using the CLKSEL[2:0] bits (CCPxCON1[10:8]), as described in 24.4.1. Time Base Generator. The module can be set to use the system clock source or clock from CLKGEN12, with Synchronization mode enabled, in the timer. The Input Capture pin (ICx) should be selected for Input Capture operation. It is recommended that the clock source be selected before enabling the module and not be changed during operation.

32-Bit Input Capture Support

The Input Capture modes have the ability to operate with a 32-bit time base. The 32-bit mode is selected by setting the T32 bit. All Input Capture functions are the same between 16-bit and 32-bit modes, with these changes in 32-bit operations:

- CCPxTMR is a 32-bit register
- CCPxBUF is a 32-bit register
- The FIFO buffer only has two levels available in 32-Bit Operating mode.

Example 24-1 shows a typical procedure for setting up Input Capture mode.

Example 24-1. Setup for Input Capture mode (Every Rising Edge)

```
CCP1CON1Lbits.CCSEL=1;     // Input capture mode
CCP1CON1bits.CLKSEL=0;    // Set the clock source (Tcy)
CCP1CON1bits.T32=0;       // 16-bit Dual Timer mode
CCP1CON1bits.MOD= 1;      // Capture ever rising edge of the event
CCP1CON2bits.ICSEL= 0;    // Capture rising edge on the Pin
CCP1CON1bits.IOPS=0;      // Interrupt on every input capture event
CCP1CON1bits.TMRPS=0;     // Set the clock pre-scaler (1:1)
CCP1CON1bits.CCPON=1;     // Enable CCP/input capture
```

Input Capture Source

The ICS[2:0] (CCPxCON2[18:16]) control bits select the input source that is used for the capture function.

I/O Pin Control

The capture module is input only and will not prevent other modules from driving its multiplexed pin. It is the user's responsibility to ensure no other modules are driving the capture pin.

24.4.3.2 Capture Event Modes

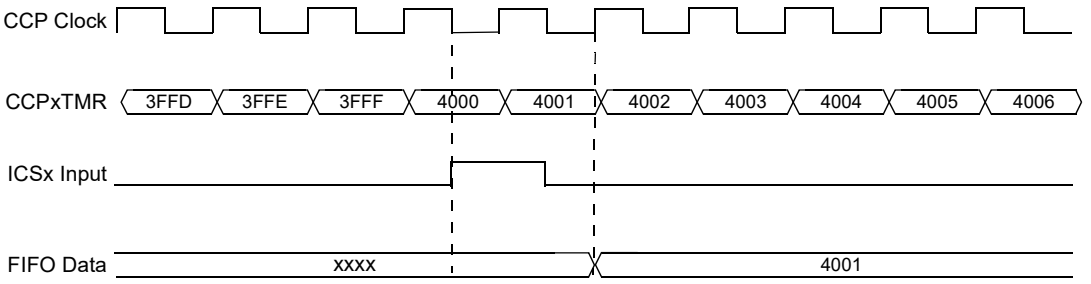
The module can capture a timer value on any of the following ICx pin transitions:

- Every rising edge (MOD[3:0] = 0001)
- Every falling edge (MOD[3:0] = 0010)
- Every rising and falling edge (MOD[3:0] = 0000, 0011)

Since the Input Capture pin is sampled on the falling edge of the timer clock; the capture pulse width must be greater than the timer clock period, plus some margin.

Because of internal synchronization requirements, the timer value captured will be up to 1.5 CCP clock cycles after the time of the actual capture edge event, as shown in Figure 24-6.

Figure 24-6. Input Capture Timing (Rising Edge)

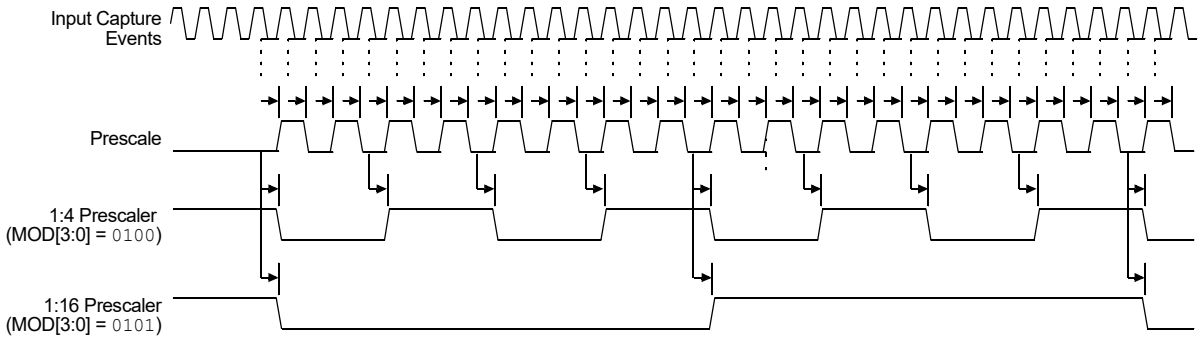


24.4.3.2.1 Input Capture Prescaler

Using the input prescaler, the Input Capture module can capture a timer value on every 4th edge (MOD[3:0] = 0100) or every 16th edge (MOD[3:0] = 0101) of the ICx input pin.

The capture pulse-width requirements are different than those for Simple Capture mode. Because of synchronization requirements inside, the timer value captured will be one to two timer clock cycles after the time of the edge capture, as shown in Figure 24-7.

Figure 24-7. Input Capture Prescaler



24.4.3.2.2 Edge Detect (Hall Sensor) Mode

Edge Detect mode (MOD[3:0] = 0000) operates the same as Capture Every Edge mode (MOD[3:0] = 0011), except that capture interrupt events do not stop when the Input Capture Buffer Overflow Status (ICOV) flag becomes set. This allows a continuous stream of capture events to trigger interrupt events without the need to continuously empty the FIFO.

24.4.3.2.3 Non-Capture Modes

When the module is in any operating mode that is not an Input Capture mode (CCM = 0), or the module is disabled (CCPON = 0), input capture functions are disabled and the module generates no Input Capture events or related interrupts. Reads of the Input Capture buffer will read as '0'.

24.4.3.2.4 Input Capture Buffer

The Input Capture FIFO buffer is up to four levels deep, depending on the Capture mode selected. For 16-bit timer captures, there are four levels in the FIFO (16-bit wide); for 32-bit timer captures, there are two levels (32-bit wide). The number of capture events required to generate a CPU interrupt can be selected by the user.

There are two status flags that provide status on the FIFO buffer. The ICBNE status bit (CCPxSTAT[0]) indicates that at least one capture event has occurred. The ICOV status bit (CCPxSTAT[1]) indicates that there have been more events than the buffer's current depth (four in 16-bit mode, two in

32-bit mode). These status flags operate the same for 16-bit capture operations and 32-bit capture operations.

While the CCP module is in Reset or not in Capture mode:

- The ICOV status flag is cleared
- The ICBNE status flag is cleared
- The FIFO is marked as empty
- A read of the FIFO buffer will return '0'

The ICBNE status flag is set on the first capture event and remains set until all capture events have been read from the FIFO. For example, if three capture events have occurred, then three reads of the Capture FIFO buffer are required before the ICBNE flag will be cleared. Each read of the FIFO buffer will allow the remaining word(s) to move to the next available top location of the FIFO.

In the event that the FIFO buffer is full with capture events and another capture event occurs prior to a read of the FIFO, an Overflow condition will occur and the ICOV bit becomes set. In addition, the capture event which caused the Overflow is not recorded, and subsequent capture events will not be placed into the FIFO until the Overflow condition is cleared by completely emptying the FIFO.

Overflow conditions cannot occur when the module is not in an Input Capture mode or when Edge Detect mode is enabled (MOD[3:0] = 0000).

Clearing of the Overflow condition can be accomplished in one of the following ways:

1. Disable the module by clearing the CCPON bit.
2. Read the Input Capture buffer until ICBNE = 0 (twice for 32-bit captures, four times for 16-bit captures).
3. Clear the ICOV bit in software. This effectively discards all previously stored data in the FIFO by resetting the data pointers to the beginning of the FIFO buffer. Clearing the ICOV in software also causes the ICBNE bit to be cleared automatically.
4. Perform a device Reset.

Upon clearing the Overflow condition, the ICBNE status flag is cleared, and to resume the Capture mode the user software must clear the ICOV status flag. If the module is disabled, and then re-enabled in Input Capture mode later, the FIFO buffer contents will be undefined and a read will yield indeterminate results.

In the event that a FIFO read is performed after the last read and no new capture event has been received, the FIFO read and write pointers will be pointing to the first buffer location of the FIFO. A read of the FIFO will return the value held in the first buffer location.

The FIFO pointer is adjusted whenever the most significant word of the buffer result is read by the CPU. This allows the results of a 32-bit Input Capture to be read by the 16-bit CPU.

24.4.3.3 Input Capture Interrupts

While in Input Capture mode, the module has the ability to generate an interrupt upon a capture event. A capture event is defined by writing a timer value to the FIFO.

The OPS[3:0] control bits (CCPxCON1[27:24]) select the interrupt postscaler, specifying the number of capture events that must occur before an interrupt is generated. Options range from an interrupt on every capture to every fourth capture. The first capture event is defined as the capture event occurring after a mode change from the Disabled state (CCPON = 0) or after ICBNE = 0.

On buffer overflow, the capture events cease and the interrupts stop unless OPS[3:0] = 0000 (interrupt on every capture). Clearing the FIFO by reading it also clears the internal interrupt counter and may affect when an interrupt is generated.

Applications often use the Input Capture pins as auxiliary external interrupt sources. In Edge Detect mode, interrupts occur regardless of FIFO overflow, as specified by OPS[3:0]. There is no need to

perform a dummy read on the Input Capture buffer to clear the event because the capture interrupt events will not stop when the FIFO Overflow (ICOV) flag becomes set. This allows a continuous stream of capture events to trigger interrupt events without the need to continuously empty the FIFO.

For example, assume that OPS[3:0] = 0001, specifying an interrupt on every second capture event. The following sequence of events will produce a single CCPxIF, as shown:

1. Turn on module; event count = 0.
2. Capture first event; FIFO contains one entry, event count = 1.
3. Read FIFO; FIFO is empty, event count = 0.
4. Capture second event; FIFO contains one entry, event count = 1.
5. Capture third event; FIFO contains two entries, event count = 2, set CCPxIF.
6. Clear interrupt count when interrupt is set (event count = 0).
7. Capture fourth event; FIFO contains three entries, event count = 1.
8. Read FIFO three times; FIFO is empty, interrupt count = 0.
9. Capture fifth event; FIFO contains one entry, event count = 1.
10. Read FIFO; FIFO is empty, event count = 0.

24.4.3.3.1 Timer Interrupts in Input Capture Modes

The module produces both timer interrupts (CCTxIF) as well as capture interrupts (CCPxIF) while operating in Input Capture mode. However, the timer interrupts only occur at the timer rollover, from FFFFh to 0000h, since there is no period register available to set the count period. If a shorter timer count period is desired, a second CCP module or external timer may be used to provide a synchronization source for the Input Capture time base.

24.4.3.4 Input Capture Operation with Synchronization and Triggering

By default, the CCP module in Input Capture mode operates with a free-running timer. The CCPxPR register is not available to set a different timer period in Input Capture mode. It is recommended to keep SYNC[4:0] configured as '11111' to maintain the free-running timer.

The timer will be held at 0000h under either of these conditions:

- Triggered operation is enabled (TRIGEN = 1) and a trigger event has not occurred (CCPTRIG = 0).
- An external Sync source has been selected (SYNC[4:0] has a value other than '11111'), which has not been enabled.

In either case, Input Capture input events will occur; however, a value of 0000h will always be captured in the FIFO. For these reasons, triggered operation and externally synchronized operation are not recommended.

24.4.3.4.1 Input Capture Signal Gating

The Input Capture source can optionally be gated by software or hardware to allow windowed capture measurements. This feature provides noise immunity in sensing applications.

The CDIS bit (CCPxSTAT[2]) provides the status of the input signal gating function. When the CDIS bit is cleared, capture events generated by the edge detect logic are allowed. When the CDIS bit is set, events from the edge detect logic are inhibited.

The time base gating logic is used for Input Capture signal gating (see 24.4.1.1. Gating Logic for more information). The ASDG[7:0] control bits (CCPxCON2[7:0]) select one or more input sources that are used to clear the CDIS status/control bit. The SSDG bit (CCPxCON2[12]) may also be used to manually gate Input Capture signals in software.

The behavior of the ASDGx sources and the SSDG bit depends on the Gating Source mode, which is selected using the ICGSM[1:0] control bits (CCPxCON2[23:22]). Three different options are available:

- When ICGSM[1:0] = 00, gating is level-sensitive. A low input level from the gating source disables subsequent capture events and the CDIS bit will be set to reflect this. A high input level enables subsequent capture events and the CDIS bit will be cleared to reflect this.
- When ICGSM[1:0] = 01, gating occurs with a rising edge of the gating source; the CDIS bit is cleared, disabling subsequent capture events. This is a One-Shot mode; subsequent edges from the gating source will have no effect.
- When ICGSM[1:0] = 10, gating occurs on the falling edge of the gating source; the CDIS bit is set, enabling subsequent capture events. This is a One-Shot mode; subsequent edges from the gating source will have no effect.

When ICGSM[1:0] = 01 or 10, the input capture gating logic operates in a One-Shot mode as described above. The user may arm the gating logic after a gating event by writing a '1' to the ICGARM (CCPxSTAT[10]) bit. This write to ICGARM has the effect of resetting the gate signal edge detection logic and also resets the CDIS status bit to the appropriate value. User software can determine the state of the one-shot logic by reading the CDIS status bit:

- When ICGSM[1:0] = 01 and a '1' is written to ICGARM, the gate signal edge detection logic is armed to look for a rising edge and the CDIS bit is set to disable input capture events until the rising edge occurs on the gate signal.
- When ICGSM[1:0] = 10 and a '1' is written to ICGARM, the gate signal edge detection logic is armed to look for a falling edge and the CDIS bit is cleared to enable input capture events until the next falling edge occurs on the gate signal.

Figure 24-8 shows the timing for gated capture events. Input events are sampled on the falling edge of the clock source. The example assumes that the Input Capture module is configured to capture every rising and falling edge (MOD[3:0] = 0011).

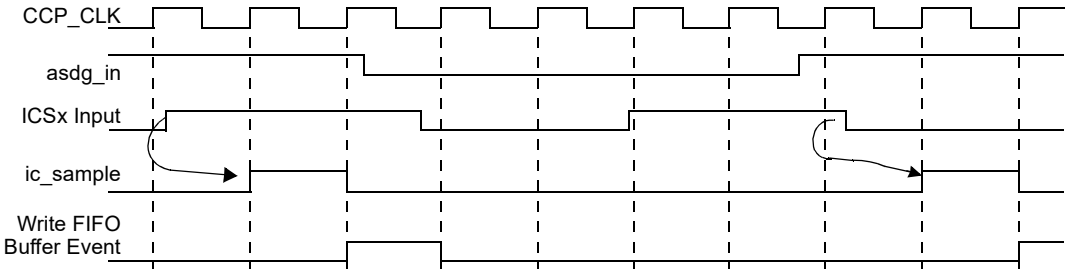
In the One-Shot modes, the edge detect logic is set to look for the appropriate edge event; the CDIS bit remains set or clear (depending on the mode) until that type of event occurs. The user may re-arm the gating logic after a gating event by rewriting ICGSM[1:0]. This act of writing to these bits (even if the same value) resets the gate signal edge detection logic and also resets the CDIS status bit to the appropriate value.

To use Input Capture gating:

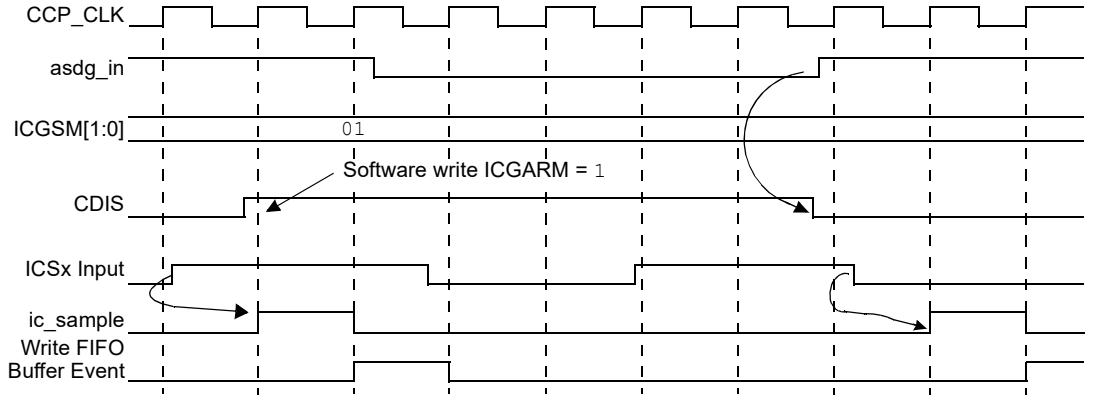
1. Select and configure the gating source.
2. Enable the appropriate gating signal source(s) using the ASDG[7:0] bits; alternatively, set or clear the SSDG bit during the event for software only control.
3. Select the Gating mode using ICGSM[1:0].
4. Configure the module for the desired Input Capture mode and input source using the MOD[3:0] and ICS[2:0] control bits. The module is now armed for a gate event.
5. The next valid rising or falling input signal edge (depending on Capture mode) after CDIS is cleared will trigger a capture event.

Figure 24-8. Gated Input Capture Examples

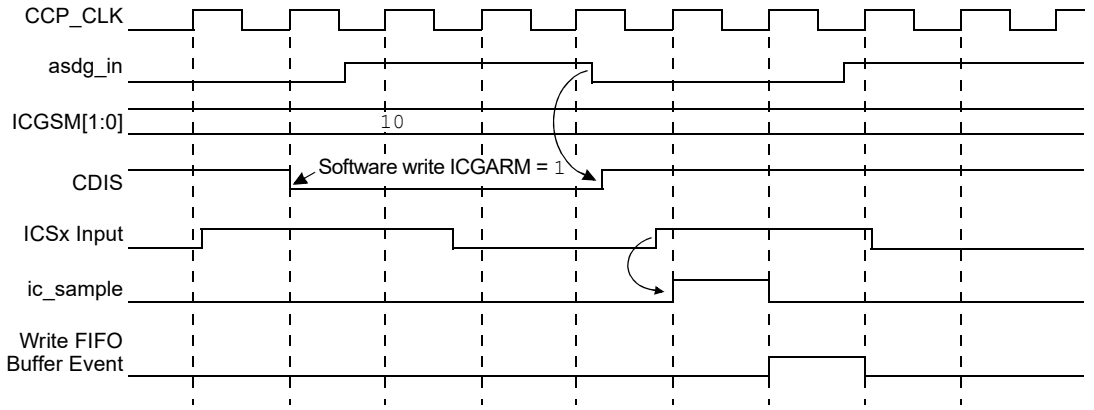
ICGSM[1:0] = 00 (Level-Sensitive)



ICGSM[1:0] = 01 (Rising Edge, One-Shot)



ICGSM[1:0] = 10 (Falling Edge, One-Shot)



24.4.4 Output Compare and PWM Modes

When CCSEL = 0 and the MOD[3:0] bits are any value other than '0000', the module operates in Output Compare mode.

Table 24-6 summarizes the various Output Compare modes.

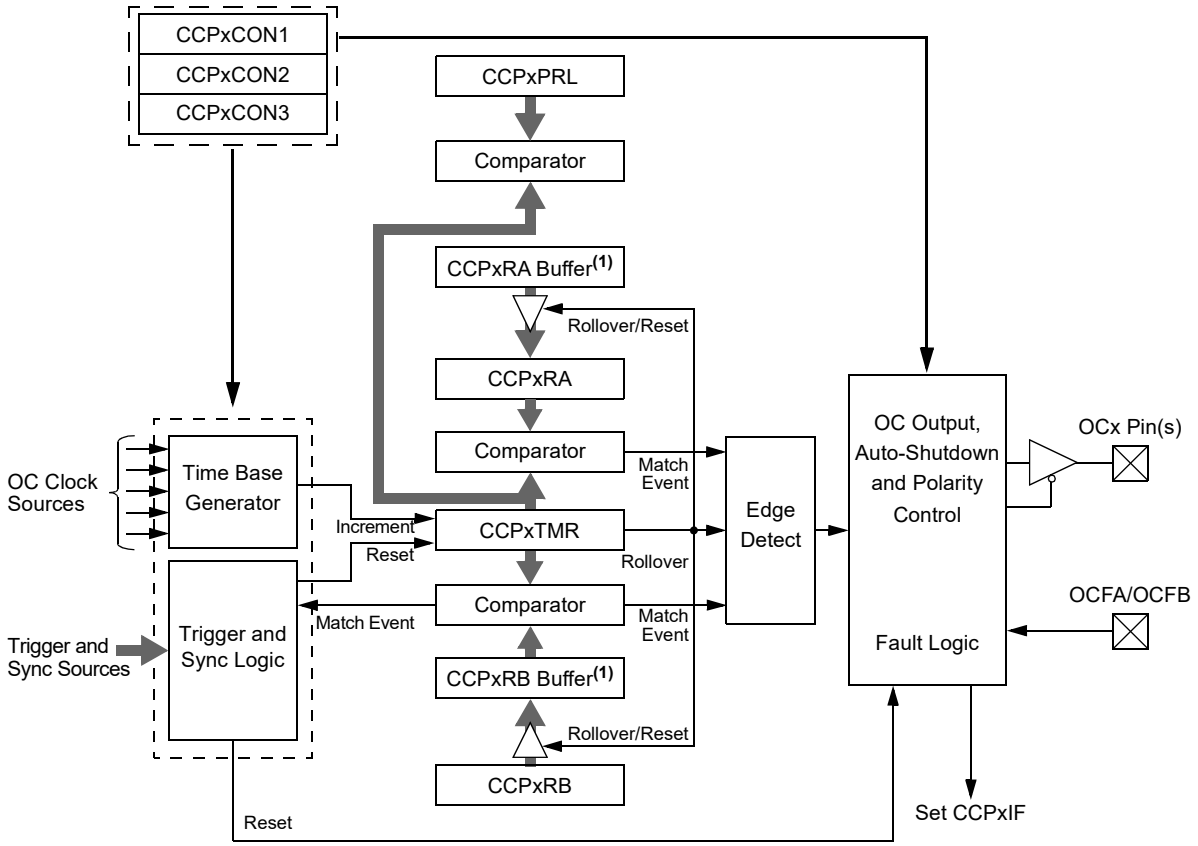
Table 24-6. Output Compare/PWM Modes

T32	MOD[3:0]	Operating Mode
0	0001	Output High on Compare (16-bit), Single Edge mode
1	0001	Output High on Compare (32-bit), Single Edge mode
0	0010	Output Low on Compare (16-bit), Single Edge mode
1	0010	Output Low on Compare (32-bit), Single Edge mode
0	0011	Output Toggle on Compare (16-bit), Single Edge mode
1	0011	Output Toggle on Compare (32-bit), Single Edge mode
0	0100	Dual Edge Compare (16-bit), Dual Edge mode
0	0101	Dual Edge Compare (16-bit buffered), PWM mode
0	0110	Reserved
0	0111	Reserved

The value of CCPxTMR is compared to one or two Compare registers, depending on its mode of operation. Output Compare mode can generate a single output transition or a train of output pulses and can generate interrupts on match-on-compare events. Figure 24-9 outlines the components used in Output Compare mode.

Like many previous dsPIC33/PIC24 modules, Output Compare mode can function as a PWM generator.

Figure 24-9. Output Compare Block Diagram



Note:

1. Buffered Output Compare and PWM modes only.

24.4.4.1 CCP Single Edge Output Compare Mode

When MOD[3:0] = `0001`, `0010` or `0011`, the selected Output Compare channel is configured for these Single Output Compare Match modes:

- Compare forces pin high (MOD[3:0] = `0001`)
- Compare forces pin low (MOD[3:0] = `0010`)
- Compare toggles pin (MOD[3:0] = `0011`)

In Single Compare mode, the CCPxRA register is used. The register is loaded with a value and is compared to the module Timer register. A CPU interrupt is generated on each compare event.

Single Edge Compare mode uses these Timer/Data registers:

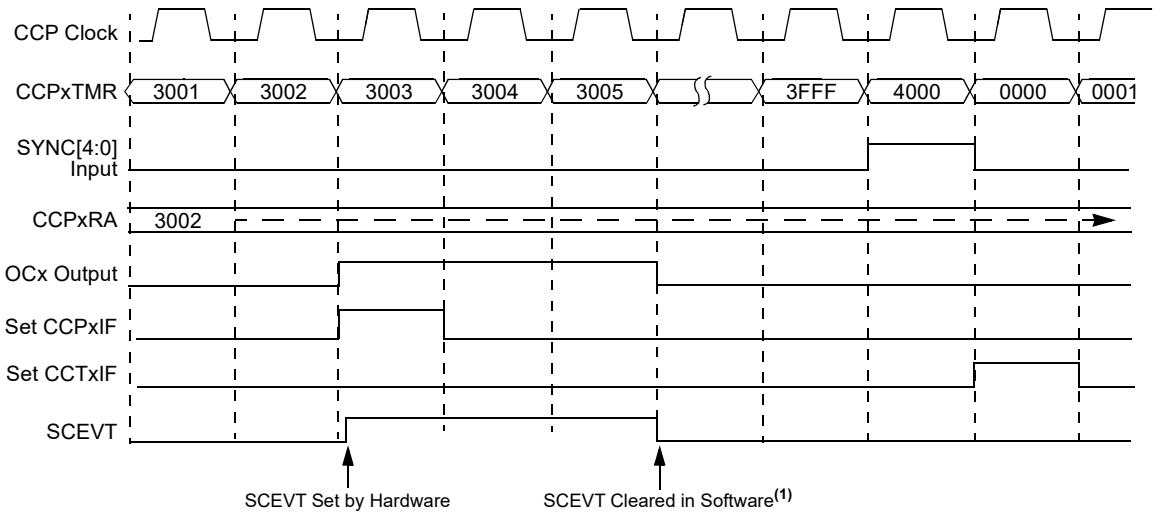
- CCPxTMRL as the Timer register (16-bit mode)
- CCPxTMR as the Timer register (32-bit mode)
- CCPxRA as the Compare Value register (16-bit mode)
- CCRxRB:CCPxRA as the Compare Value register (32-bit mode)
- CCPxPRL as the Timer Period register (16-bit mode only)

24.4.4.1.1 Single Edge Compare Mode (High Output)

In this mode (see Figure 24-10), the output pin is initially driven low and remains low until a match occurs between the timer and CCPxRA register. The key timing events to note are:

- The output pin is driven high, one clock period after a match occurs between the Timer and CCPxRA registers. The output pin remains high until a mode change has been made or the module is disabled.
- The timer counts up until it rolls over or until the selected SYNC[4:0] input is asserted (depending on the value of SYNC[4:0]) and then resets to 0000h on the next clock.
- The compare interrupt signal (to set CCPxIF) is asserted, and the output pin is driven high.
- The timer interrupt signal (to set CCTxIF) is asserted for one clock period on a Time Base Reset or rollover event.

Figure 24-10. Single Compare Mode (High Output)



Note:

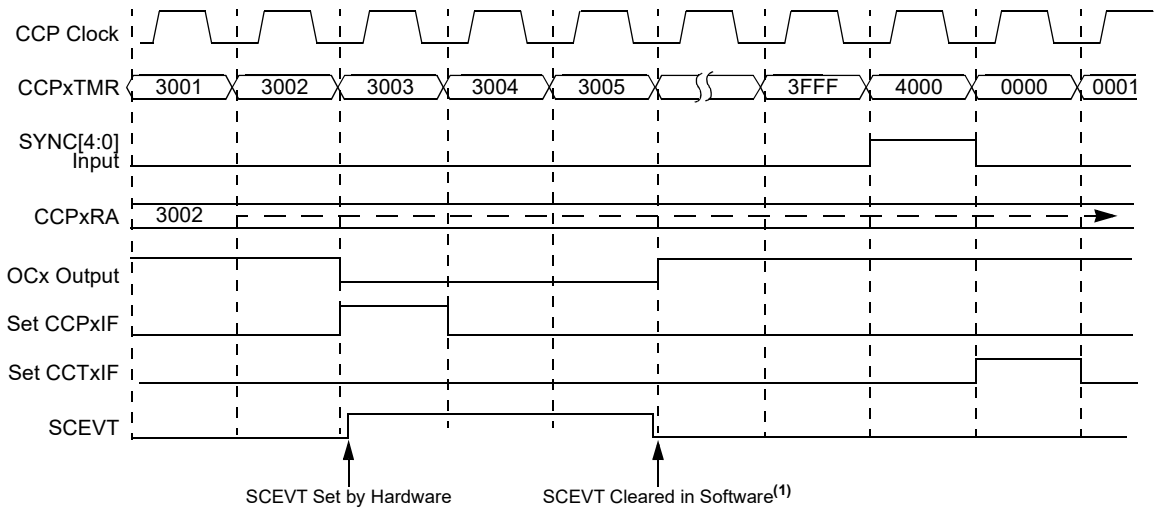
1. SCEVT has to be cleared to enable the next capture.

24.4.4.1.2 Single Compare Mode (Low Output)

Once the Compare mode has been enabled (Figure 24-11), the output pin will initially be driven high, and remain high until a match occurs between the timer and CCPxRA register. The key timing events to note are:

- The output pin is driven low one clock period after a match occurs between the timer and CCPxRA registers. The output pin remains low until a mode change has been made or the module is disabled.
- The timer counts up until it rolls over or until the selected SYNC[4:0] input is asserted, and then it resets to 0000h on the next clock.
- The compare interrupt signal (to set CCPxIF) is asserted and the output pin is driven low.
- The timer interrupt signal (to set CCTxIF) is asserted for one clock period on a Time Base Reset or rollover event.

Figure 24-11. Single Compare Mode (Low Output)



Note:

1. SCEVT has to be cleared to enable the next capture.

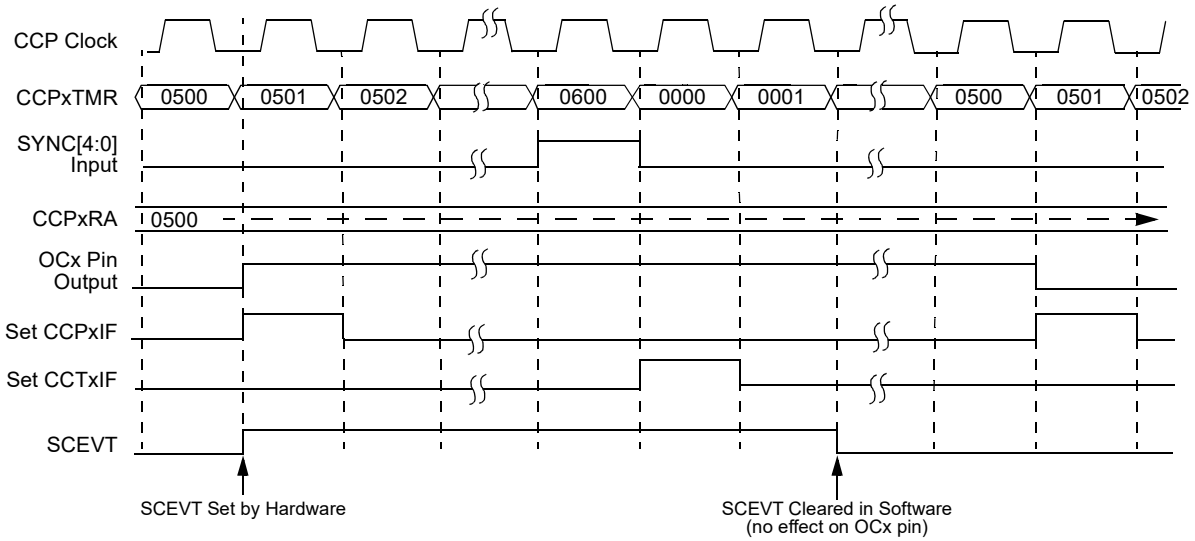
24.4.4.1.3 Single Compare Mode (Toggled Output)

Once this Compare mode has been enabled (Figure 24-12), the output pin is initially driven low and then toggled on each subsequent match event between the timer and CCPxRA register. The key timing events to note are:

- The state of the output pin is toggled one clock period after a match occurs between the timer and CCPxRA registers. The output pin remains at its new state until the next toggle event, until a mode change has been made or the module is disabled.
- The timer counts up until it rolls over, or until the selected SYNC[4:0] input is asserted, and then resets to 0000h on the next clock.
- The respective channel interrupt output (CCPxIF) is asserted when the output pin is toggled.
- The time base interrupt signal (CCTxIF) is generated on a Timer Reset or rollover event.

Note: The internal OCx pin output logic is set to a logic '0' on a device Reset; however, the initial output pin state for the Toggle mode can be reversed using the POLACE polarity control bit.

Figure 24-12. Single Compare Mode (Toggle Output)



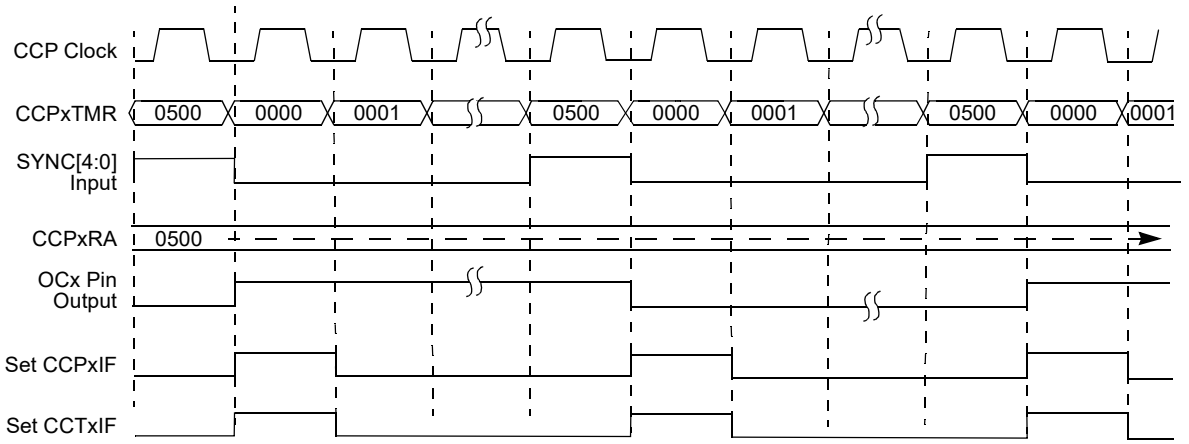
24.4.4.1.4 Special Cases of Single Compare Mode

In Single Edge Compare modes, there are several special cases to consider:

1. When the value of CCPxRA is greater than the timer period, the compare value will always be greater than the timer value. No compare event will ever occur, and the compare output will remain at the initial condition.
2. When the value of CCPxRA equals the timer period, the compare interval is the same as the timer period. Combined with Toggle mode, this can be used to generate a fixed frequency square wave (Figure 24-13).
3. When CCPxRA = 0000h, the timer is held in Reset, either by an asserted trigger source or when CCPTRIG = 0 is the triggered operation. The compare output will remain at the initial condition. The compare output will change once the selected trigger source is deasserted, allowing the timer to operate.
4. If CCPxRA is cleared after a compare event, the SYNC[4:0] signal is asserted and the compare output will remain at its previous state.
5. If, after a compare event, the CCPxRA register is modified to a value greater than the current timer value but less than the timer period, a second compare event will be generated within the count period. The SCEVT bit must also be cleared when MOD[3:0] is '0001' or '0010' to reset the output pin for the next compare event.

Note: For all special cases, 'timer period' can be defined as either a CCPxPR match or an event by the selected SYNC[4:0] source.

Figure 24-13. Single Compare Mode, Toggle Output, Timer Period = CCPxRA



24.4.4.1.5 Single Edge Output Compare Event Status

The SCEVT bit (CCPxSTAT[3]) indicates the status of a single edge compare event and allows the application to re-arm a single edge compare event without changing the module operating mode or resetting the module. It only functions during single edge compare events; in all other modes, the bit always reads as '0'.

When MOD[3:0] = 0001, the OCx pin is asserted high after a compare event and SCEVT is set to '1' by hardware. The application may clear SCEVT in software. Once the bit is cleared, the OCx pin is reset to a low output and the compare logic is reset to allow the next rising edge compare event.

When MOD[3:0] = 0010, the OCx pin is asserted low after a compare event and SCEVT is set to '1' by hardware. The application may clear SCEVT in software. Once the bit is cleared, the OCx pin is reset to a high output and the compare logic is reset to allow the next falling edge compare event.

When MOD[3:0] = 0011, the OCx pin is toggled after a compare event and SCEVT is set to '1'. The application may clear SCEVT in software, but the state of the OCx pin will not change when the bit is cleared. In this mode, SCEVT only provides event status information and does not affect the OCx pin.

When MOD[3:0] = 0001 or 0010, the application may set SCEVT to '1' to inhibit Single Edge Output Compare events. This feature is useful when it is desired to delay an edge event during a particular interval, for example. No changes will occur to the OCx pin during this time. When the SCEVT bit is cleared by software, the OCx output pin will be reset to the initial state, and a rising or falling edge will be generated when the next compare event occurs.

32-Bit Operation with Single Compare Mode

The previous examples all assume 16-bit single compare operations (T32 = 0). Single Edge Compare modes can also operate with a 32-bit time base, selected by setting T32 = 1. Operation in 32-bit mode is identical, except that the CCPxRB register is paired with CCPxRA to provide a 32-bit compare value for CCPxTMR. CCPxRA is used for the upper 16 bits of the compare value.

No period register is available to set the count period of CCPxTMR. If a count period less than FFFF FFFFh is desired, the module can be synchronized to an external source to set the count period.

24.4.4.2 Dual Edge Compare Mode

When MOD[3:0] = 0100, the Output Compare channel is configured to produce a continuous series of pulses. The parameters for the pulse train are determined by the CCPxRA, CCPxRB and CCPxPRL.

Dual Edge Compare mode is only available in 16-bit mode. The T32 bit has no affect.

Dual Edge Compare mode uses these timer/data registers:

- CCPxTMRL as the Timer register
- CCPxRA for the Rising Edge Value register
- CCPxRB for the Falling Edge Value register
- CCPxPRL for the Timer Period register

Figure 24-14 depicts the signal timing for Dual Edge Compare mode. The typical operation in this mode is as follows:

1. When Dual Edge Compare mode is enabled, the pin state is driven low. At some point, the timer is enabled (triggered) by a hardware or software event to start the count process.
2. Upon the first timer compare match with the Compare register, CCPxRA, the output pin will be driven high.
3. When the incrementing timer count matches the Compare register, CCPxRB, the second and trailing edge (high-to-low) of the pulse is driven onto the output pin. At this second compare, the Output Compare Interrupt Flag (CCPxIF) is generated.
4. The Timer Interrupt Flag (CCTxIF) is generated, along with the CCP Sync signal, when the timer rolls over (when SYNC[4:0] = 00000) or when an event is defined by SYNC[4:0].
5. The output pulses continue repeatedly until the mode is terminated by the application or a device Reset occurs.

This is the prototype case, where the Timer Period and the Match registers are all different values, ordered as (Timer Period > CCPxRB > CCPxRA). There are special cases, however, where the conditions differ, resulting in a specific type of output. The cases are listed in Table 24-7, and are described in the following sections.

Figure 24-14. Typical Dual Edge Compare Timing Sequence

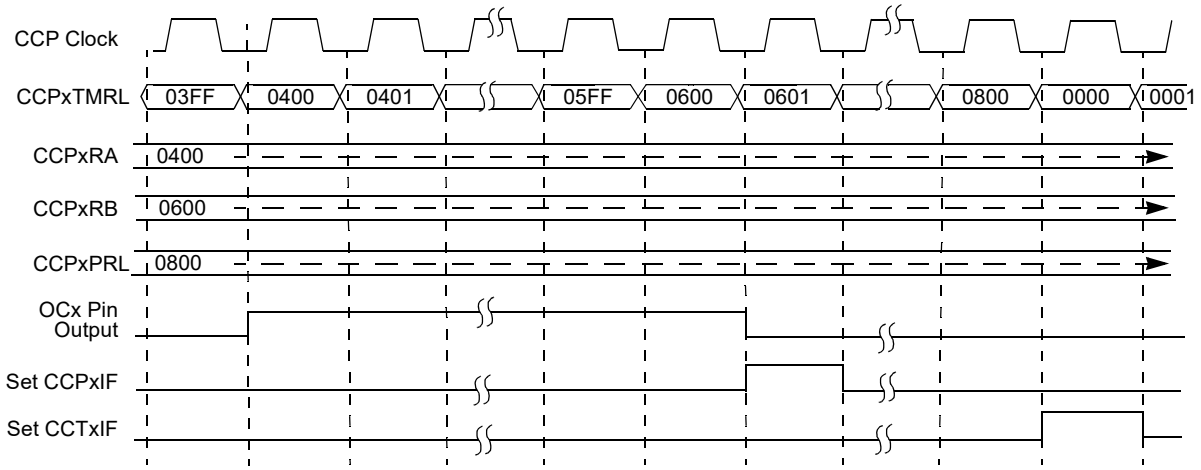


Table 24-7. Special Conditions for Dual Edge Compare Operations

Condition	Output	Output Compare Interrupt on Falling Edge of OCx Pin	Timer Interrupt When Timer Matches Period
CCPxRA = CCPxRB	No output, OCx pin remains low	None	Yes
CCPxRA = CCPxRB + 1	One pulse	Yes	Yes
Timer Period < CCPxRA(1)	No output	N/A(2)	Yes
Timer Period = CCPxRB(1)	OCx goes low at CCPxRB	Yes	Yes
Timer Period < CCPxRB(1)	Continuous high	None	Yes

.....continued

Condition	Output	Output Compare Interrupt on Falling Edge of OCx Pin	Timer Interrupt When Timer Matches Period
Timer Period = CCPxRB, CCPxRA = 0(1)	CCP goes high when TMR = 1 and goes low when CCPxRB matches timer	Yes	Yes
CCPxRA > CCPxRB	Pulse train	Yes	Yes

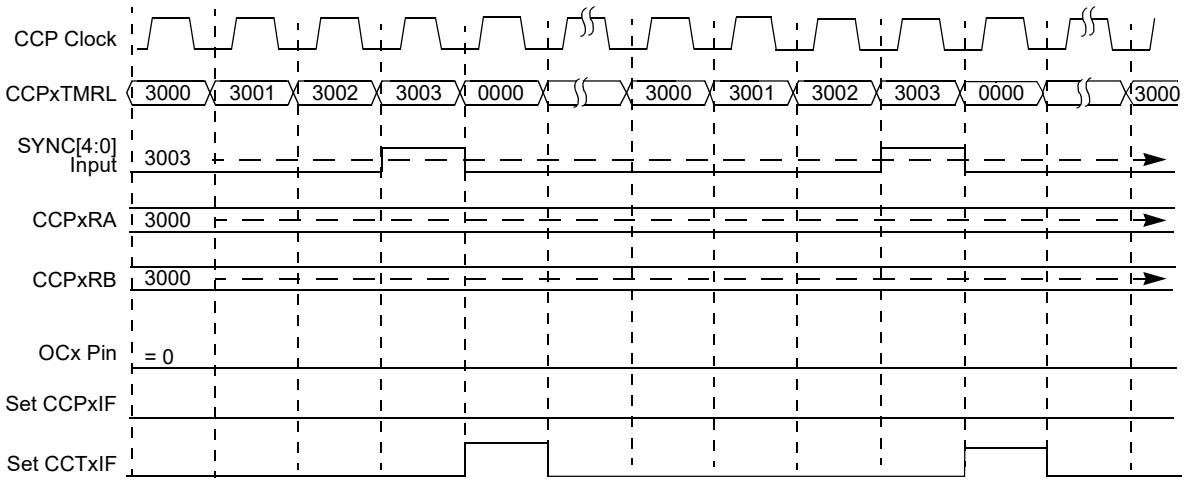
Notes:

1. Timer period is either the period register value or when the timer is reset by the input selected by SYNC[4:0].
2. If CCPxRB is also less than the timer period, an interrupt will be generated, even though there is no activity on the OCx pin.

24.4.4.2.1 CCPxRA = CCPxRB

If CCPxRA and CCPxRB have the same value, the output is initialized low and stays low; no pulses are generated and no Output Compare interrupt is generated (Figure 24-15). Put another way, the PWM duty cycle is 0. The Reset/clear-on-CCPxRB match logic overrides the set-on-CCPxRA match logic for a net result of no change in the output state.

Figure 24-15. Timing for Dual Edge Compare (CCPxRA = CCPxRB)



24.4.4.2.2 CCPxRB = CCPxRA + 1

When the value of CCPxRB is one greater than the value of CCPxRA and both registers are less than the period register, an output pulse that is one CCP clock cycle wide is generated.

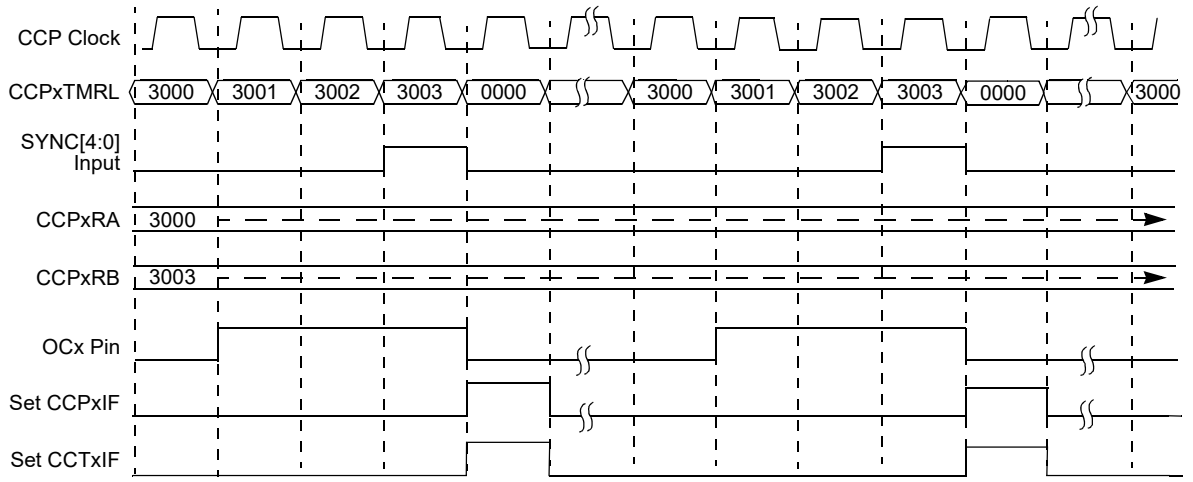
24.4.4.2.3 Timer Period < CCPxRA

When the value of CCPxRA is greater than the timer period, no output pulses are generated.

24.4.4.2.4 Timer Period = CCPxRB

The module will still generate the high-to-low transition when the value of CCPxRB equals the timer period. This is true whether the period is determined in Sync operation from an external source (as shown in Figure 24-16) or on a match with CCPxPRL.

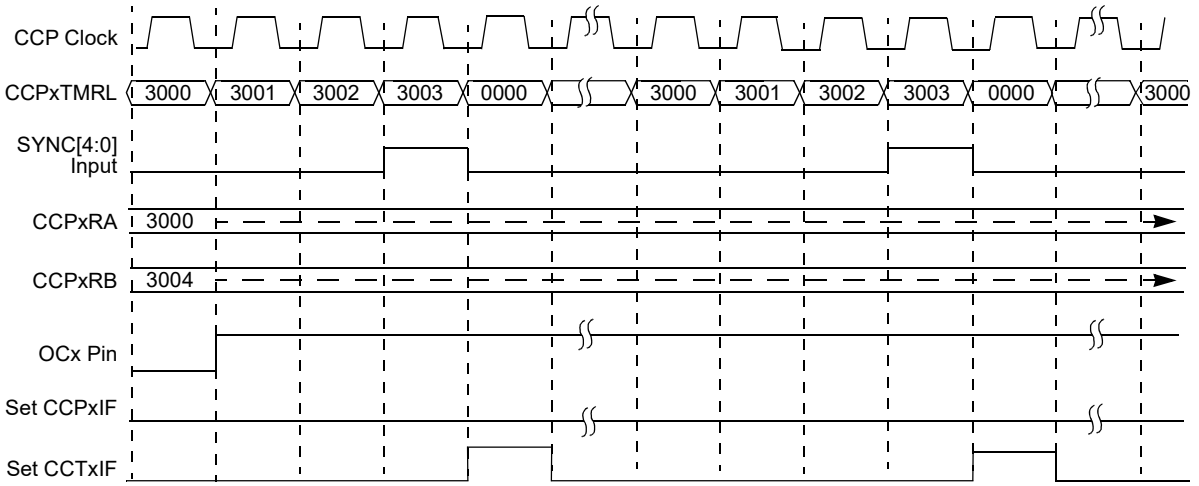
Figure 24-16. Timing for Dual Edge Compare (Timer Period = CCPxRB)



24.4.4.2.5 Timer Period < CCPxRB

If the value of the CCPxPRL is less than that of CCPxRB, but greater than CCPxRA, only one pin transition will be generated until the CCPxRB register contents are changed to a value less than or equal to CCPxPR. No Output Compare interrupt is generated (Figure 24-17). This condition allows the module to produce a 100% duty cycle output.

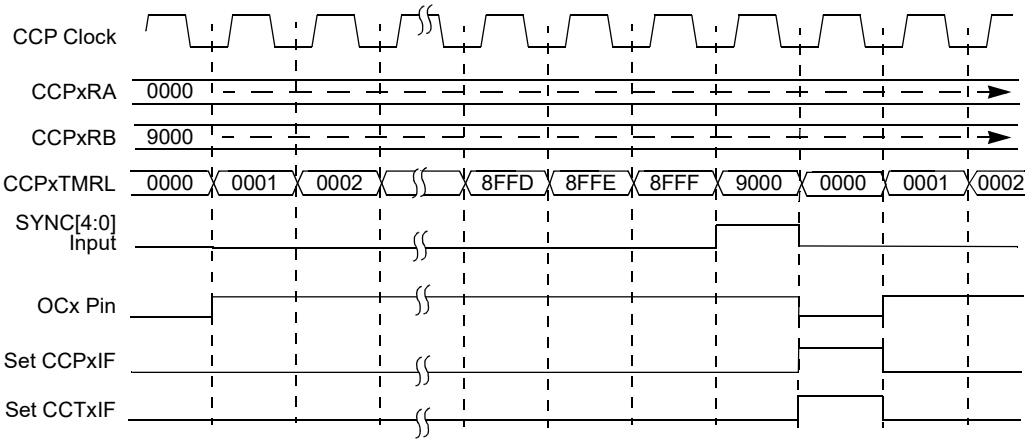
Figure 24-17. Timing for Dual Edge Compare (CCPxPR < CCPxRB)



24.4.4.2.6 CCPxTMRL = CCPxRB and CCPxRA = 0

In Sync operation, if CCPxRA is 0000h, the OCx output is asserted on the first clock after the Timer Reset (CCPxTMRL = 0001h). It remains asserted until the value of CCPxRB matches the timer period (when the input selected by SYNC[4:0] is asserted). At this point, the OCx output is deasserted and the CCPxIF is generated on the falling edge (Figure 24-18).

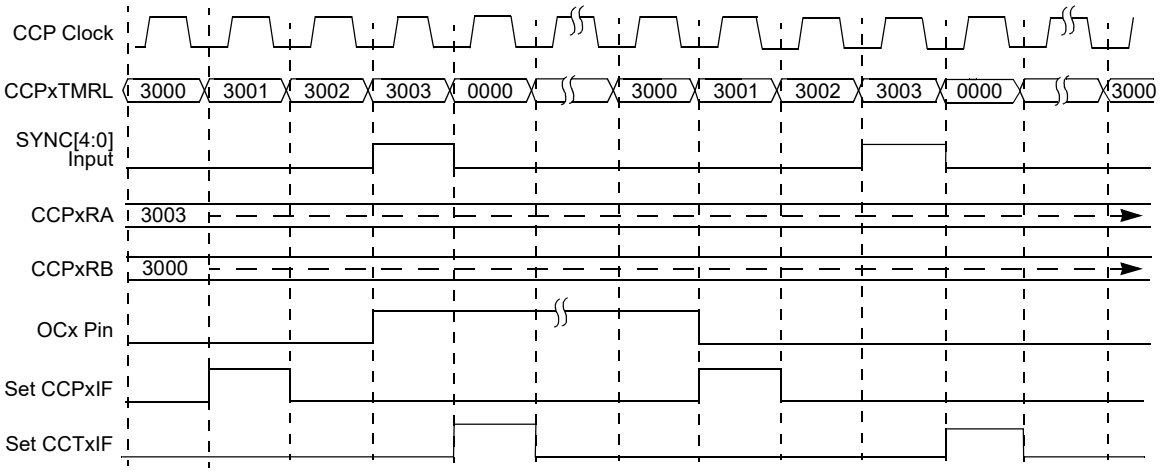
Figure 24-18. Timing for Dual Edge Compare (CCPxRA = 0000h, CCPxRB = Timer Period)



24.4.4.2.7 CCPxRA > CCPxRB

If CCPxRA > CCPxRB, a continuous train of pulses is generated. The timer counts up to the first match (CCPxTMRL = CCPxRA) and the first (rising) edge is generated. CCPxTMRL continues to count up, resetting when the Sync source selected by SYNC[4:0] is asserted. The timer then counts up to the second match (CCPxTMRL = CCPxRB), at which time the second (falling) edge of the signal is generated. The CCPxIF interrupt is generated on the falling edge of the output pulse. The sequence repeats until the module is disabled (Figure 24-19).

Figure 24-19. Timing for Dual Edge Compare (CCPxRA > CCPxRB)



Note: When operating in Dual Compare mode, the CCTxIF signal is asserted on a match between the CCPxRB register value and CCPxTMRL.

24.4.4.3 Dual Edge Buffered Compare (PWM) Mode

When MOD[3:0] = 0101, the module functions the same as in Dual Edge Compare mode, with the exception that CCPxRA and CCPxRB are double-buffered. In all other respects of output signal generation, operation is the same. Writes to the Data registers (CCPxRA and CCPxRB) are stored in holding buffers. The contents of the buffers are transferred to CCPxRA and CCPxRB on a Time Base Reset.

Dual Edge Buffered Compare mode is only available in 16-bit mode. The T32 bit has no affect.

Dual Edge Buffered Compare mode uses these Timer/Data registers:

- CCPxTMRL as the Timer register
- CCPxRA for the Rising Edge Value register of the next period
- CCPxRB for the Falling Edge Value register of the next period
- CCPxPRL for the Timer Period register

The Dual Edge Buffered Compare mode is used to create PWM signals. The buffering of the CCPxRA and CCPxRB registers allows the user to create glitch-free updates to the PWM signal edge times.

If edge-aligned PWM signals are desired, maintain CCPxRA with a value of 0000h. Using a non-zero value for CCPxRA creates PWM signals with arbitrary phase alignments.

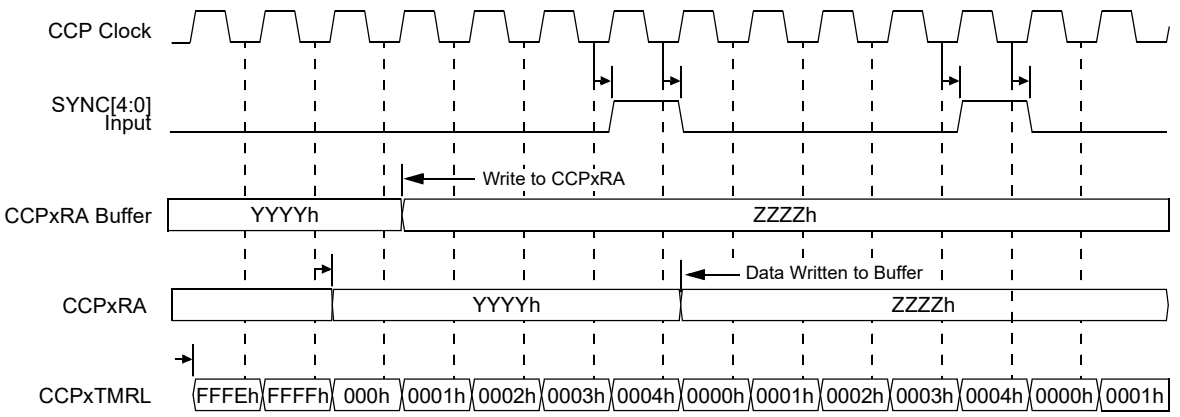
CCPxRA and CCPxRB are double-buffered. Data are written from the buffers to CCPxRA and CCPxRB under these conditions:

- When the timer is reset to 0000h on a Sync event (source selected by SYNC[4:0] is asserted)
- When the timer rolls over from FFFFh to 0000h
- When the module is disabled (CCPON = 0); any writes to CCPxRA and CCPxRB are immediately transferred to their Compare registers

Figure 24-20 shows the timing for writing to the buffer in Sync operation. CCPxRA and its buffer are shown; CCPxRB and its buffer operate in an identical manner. For output signal generation, refer to 24.4.4.2. Dual Edge Compare Mode.

The procedure for configuring the module for Dual Edge Buffered Compare mode is shown in Example 24-2.

Figure 24-20. Buffer Writes in Dual Edge Buffered Compare Mode



Example 24-2. Setup for Dual Edge Buffered Compare Mode

```
// Set CCP operating mode
CCP1CON1bits.CCSEL = 0;          // Set SCCP operating mode (OC mode)
CCP1CON1bits.MOD = 0b0101;       // Set mode (Buffered Dual-Compare/PWM mode)
//Configure SCCP Timebase
CCP1CON1bits.T32 = 0;            // Set timebase width (16-bit)
CCP1CON1bits.TMRSYNC = 0;        // Set timebase synchronization (Synchronized)
CCP1CON1bits.CLKSEL = 0b000;     // Set the clock source (Tcy)
CCP1CON1bits.TMRPS = 0b00;       // Set the clock pre-scaler (1:1)
CCP1CON1bits.TRIGEN = 0;         // Set Sync/Triggered mode (Synchronous)
CCP1CON1bits.SYNC = 0b00000;     // Select Sync/Trigger source (Self-sync)
//Configure SCCP output for PWM signal
CCP1CON2bits.OCAEN = 1;          // Enable desired output signals (OC1A)
CCP1CON3bits.OUTM = 0b000;       // Set advanced output modes (Standard output)
```

```
CCP1CON3bits.POLACE = 0;        // Configure output polarity (Active High)
CCP1TMR = 0x0000;               // Initialize timer prior to enable module.
CCP1PR = 0x0000FFFF;            // Configure timebase period
CCP1RA = 0x00001000;            // Set the rising edge compare value
CCP1RB = 0x00008000;            // Set the falling edge compare value
CCP1CON1bits.CCPON = 1;         // Turn on SCCP module
```

24.4.4.4 Output Control for Compare/PWM Modes

When the module operates in an Output Compare mode, the following blocks determine how the Output Compare signal is presented on the output pins:

- Auto-Shutdown Control Block
- Output Polarity Control Block

The auto-shutdown control block responds to asynchronous external inputs or software control, placing all output pins under control of the module into a predetermined state.

The output polarity control block determines the output polarity on each pin under control of the module. This block takes effect after all other control of the output pins.

24.4.4.4.1 Output Pin Enable (SCCP)

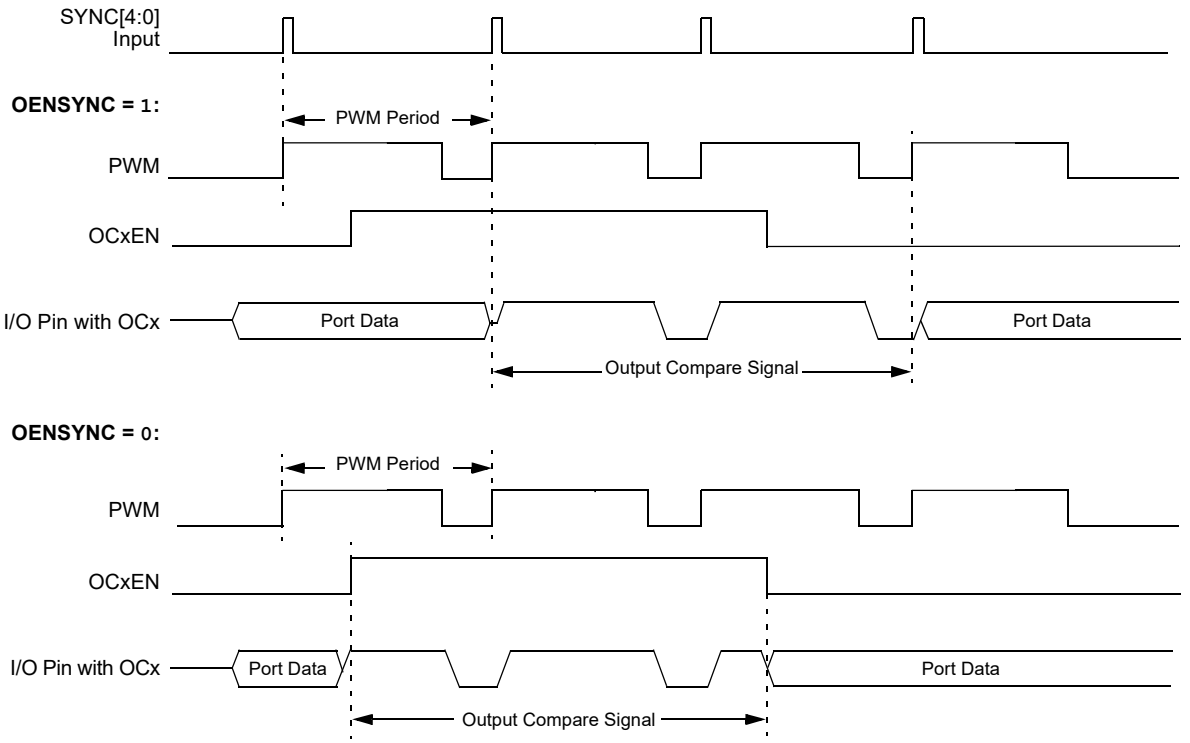
The SCCP module has only one output pin, OCxA, in Output Compare or PWM mode. The OCAEN bit (CCPxCON2[24]) is the only implemented control bit. It determines whether or not the module has control of the output pin. By default, this OCxA output is enabled on device Resets.

24.4.4.4.2 Output Enable Synchronization

Changes to the OCAEN control bit may be optionally synchronized to the timer period, allowing software changes to PWM settings be synchronized to the PWM period's boundaries. This prevents incomplete output pulses as a result of steering changes.

The OENSYNC control bit (CCPxCON2[31]) controls the synchronization of the PWM output to period boundaries. When OENSYNC = 1, changes to the OCAEN control bit take effect on a Timer Reset (i.e., the Sync input selected by SYNC[4:0] is asserted). When OENSYNC = 0, changes to the OCAEN control bit take effect immediately. Figure 24-21 shows the effect of the OENSYNC bit on the I/O pin shared by the OCx output.

Figure 24-21. OENSYNC Bit Operation



24.4.4.4.3 Output Enable on Trigger

The OETRIG control bit (CCPxCON3[31]) allows the user to select whether the CCP output pins are held in a high-impedance state or driven by the module before the timer is triggered. The OETRIG control bit only affects the module operation in Triggered mode (TRIGEN = 1).

The operation of the OETRIG bit function varies depending on the Output mode of the module.

The output pin is held in a high-impedance state when the timer is not triggered (CCPTRIG = 0).

24.4.4.4.4 Auto-Shutdown Control

The primary function of the auto-shutdown control logic is to place the module output pins in a safe state when driving external power circuitry. The auto-shutdown function can also be used to place the output pins of the CCP module in a specific state based on an external event.

Auto-shutdown control is implemented as part of the time base gating (see 24.4.1.1. Gating Logic). The user must select an input source for the auto-shutdown using the ASDG[7:0] control bits (CCPxCON2[7:0]). The available sources for auto-shutdown are device-dependent and typically include such sources as comparator outputs, I/O pins, software control (i.e., the SSDG bit) and so on. With the exception of the SSDG control bit, which is active-high, a low output from the shutdown source places the module OCA pins in the shutdown state. The auto-shutdown event is level-sensitive, not edge-triggered. The comparator output and other shutdown sources are not synchronized to the system clocks to provide an immediate response of the CCP module to the shutdown input signal.

When a shutdown occurs, the selected output states are placed onto the module port pins.

Table 24-8. Auto-Shutdown and Gating Sources

ASDG[7:0]	Auto-Shutdown/Gating Source
1xxx xxxx	OCFB

.....continued

ASDG[7:0]	Auto-Shutdown/Gating Source
x1xx xxxx	OCFA
xx1x xxxx	CLC1
xxx1 xxxx	ICMn
xxxx 1xxx	OCFD
xxxx x1xx	OCFC
xxxx xx1x	Comparator 2 Output
xxxx xxx1	Comparator 1 Output

Auto-Shutdown Pin State

The state of the output pins is controlled by the PSSA[1:0] control bits (CCPxCON3[19:18). The PSSAx bits affect the states of the OCxA output pin. These control bits let the user select whether the I/O pin is driven inactive, driven active or put into a high-impedance state.

Software Shutdown

The user application may invoke a shutdown event at any time by setting the SSDG control bit (CCPxCON2[12]). This bit behaves exactly like an external shutdown source, except that the polarity of the control bit is inverted. A shutdown event will be caused whenever the SSDG bit is set. The module output pins go to their programmed shutdown state and remain in that condition until the SSDG bit is cleared in software. The software shutdown feature may be used by itself or in parallel with an external source.

Notes:

1. The user may also need to clear the ASEVT status bit if automatic restarts are not enabled.
2. Any enabled shutdown source selected by the ASDGx bits and the SSDG software shutdown bit has priority over a software write to the ASEVT bit. The Fault condition cannot be exited unless all shutdown sources are inactive.

Auto-Shutdown Status

The ASEVT status bit (CCPxSTAT[4]) indicates the status of a shutdown event. If the ASEVT bit is cleared, the output pins associated with the CCP module will have normal activity.

If the ASEVT bit is set, then the output pins will be driven to their shutdown states or held in a high-impedance state. The ASEVT bit can also be used as a control bit to manually reset the shutdown condition, as described in Automatic Restart Enable.

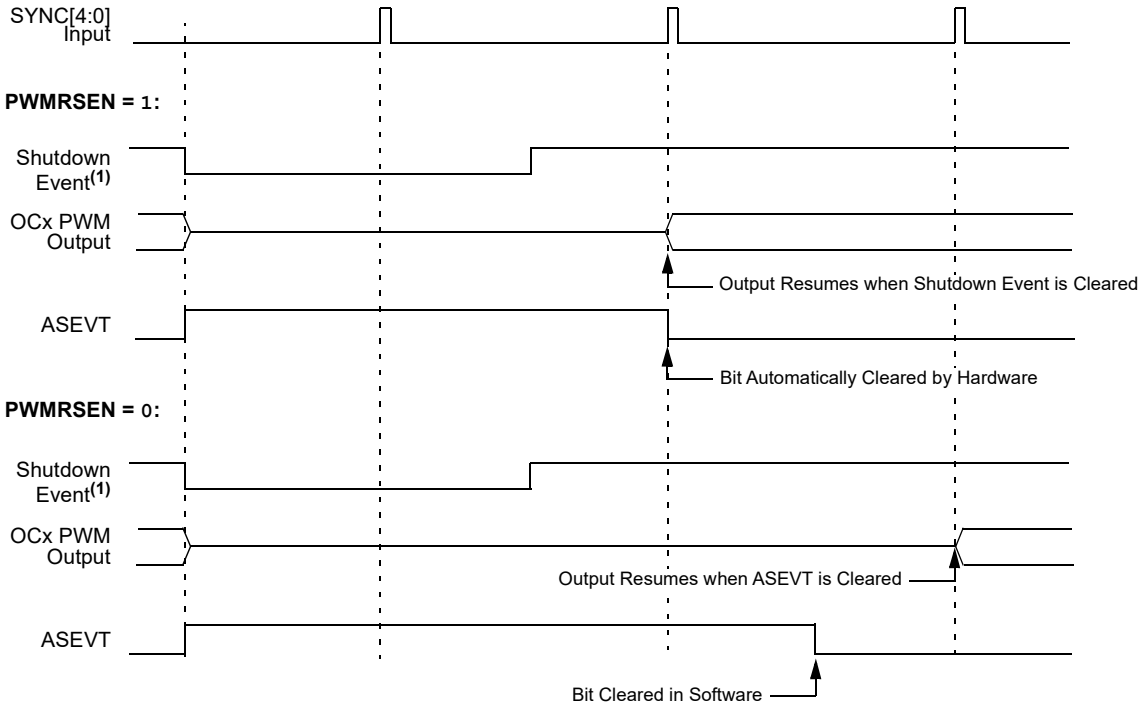
Automatic Restart Enable

The PWMRSEN bit (CCPxCON2[15]) controls how the shutdown state is ended. If PWMRSEN = 0, the module will wait until the ASEVT status bit is cleared in user software. Normal output pin activity can only be resumed when the ASEVT bit is cleared AND the external shutdown source signal is no longer present. If the external shutdown source signal is still active, the user cannot clear the ASEVT bit.

If PWMRSEN = 1, normal output pin activity will automatically resume when the external shutdown source signal is inactive and the next PWM period begins (i.e., when the Sync source selected by SYNC[4:0] is asserted). The ASEVT bit will automatically be cleared in hardware at this time. If the shutdown is still in effect at the time a new cycle begins, that entire cycle is suppressed, thus eliminating narrow, glitch pulses. The PWM outputs are then restarted on the next cycle.

If PWMRSEN = 0, once a shutdown condition occurs, the PWM remains Idle until manually restarted by the user. The module can be restarted by clearing the ASEVT bit in software.

Figure 24-22. Automatic Restart Enable



Note:

1. Any device-defined hardware shutdown event when the corresponding ASDG bit is set or when the SSDG bit is set.

Gated Auto-Shutdown Mode

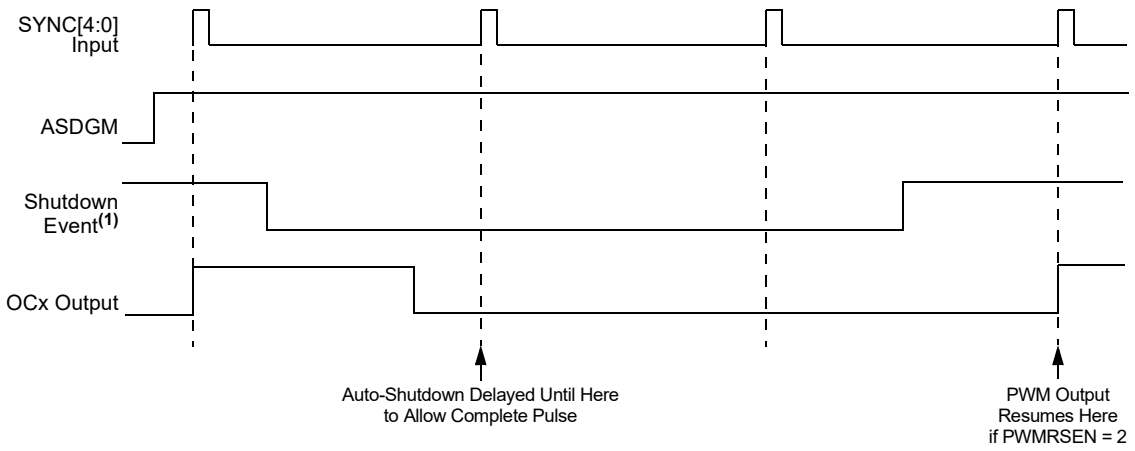
For some types of power control applications, it is useful to delay the effect of the auto-shutdown input. The ASDGM control bit (CCPxCON2[14]) enables gated shutdown operation. When ASDGM is set, the effect of an auto-shutdown signal input does not take place until the next PWM period boundary. This allows the PWM generator to produce the entire pulse that was programmed for the present cycle.

Pulses are terminated by the hardware beginning on the next cycle. Pulses will resume on the next PWM period after the shutdown event has ended.

Note: Pulses only resume automatically if PWMRSEN = 1. If PWMRSEN = 0, the ASEVT status bit must be cleared in software for pulses to resume.

The Gated mode allows the Automatic Shutdown mode to be used along with a comparator to implement a 'Pulse Skipping' or 'Gated Oscillator' Switch mode power supply. The inductor is charged for a fixed period of time with each pulse, putting a specific amount of energy into the supply. If the output voltage (current) is high enough, then the comparator gates the pulses.

Figure 24-23. Gated Auto-Shutdown Mode



Note:

1. Any device-defined hardware shutdown event when the corresponding ASDGx bit is set or when the SSDG bit is set.

Notes:

1. If automatic restarts are not enabled, the application may also need to clear the ASEVT bit.
2. Any enabled shutdown source selected by the ASDGx bits and the SSDG software shutdown bit takes priority over a software write to the ASEVT bit. The Fault condition cannot be exited unless all shutdown sources are inactive.

24.4.4.4.5 Output Polarity Control

The polarity of OCxA is controlled by the POLA control bit (CCPxCON3[21]).

The output polarity control is applied to the output signal after auto-shutdown logic. The polarity control bits are effective for all Output Compare and PWM modes of the module.

24.4.4.5 CCP External Input Source Mode

The External Input Source mode is selected when:

- CCM = 0
- MOD[3:0] = 1111
- T32 = x

The External Input Source mode is provided to bypass all Output Compare signal generation logic in the CCP module. This allows an external signal to be connected to the output control block to create complementary signals, provide Auto-shutdown control, and so on.

The module timebase operates in 16-bit mode for the External Input mode and timebase interrupts are generated. If the timebase is self-synchronized, the count period is set by the CCPxPR register. The CCPxRA and CCPxRB registers have no effect on the timebase operation for this mode.

The timebase remains operational in External Input mode so certain features of the output control logic can continue to operate. For example, these features must have a timebase period reference:

- The automatic restart feature of the auto-shutdown logic (PWMRSEN = 1) requires a timer synchronization signal to operate properly.
- The Auto-Shutdown Gated Mode (ASDGM = 1) requires a timebase synchronization signal.

If the timebase is not used for synchronization purposes in the External Input mode, then it can be used as a general purpose 16-bit timer to provide periodic CPU interrupts. The external signal input source can be one of the signals connected to the Input Capture source and is selected using the ICS[2:0] (CCPxCON2[18:16]) control bits.

The external signal will be synchronized to the CCP timebase clock source and therefore must meet certain timing requirements. Specifically, the high and low times for the external signal must be no less than one timer clock period.

Note: If the Input Capture pin (ICx) is located on the same pin as one of the Output Compare pins of the same module, and if the ICx pin is enabled as the External Input signal source, to avoid unexpected results, the user should not enable the ICx pin and the OCx pin simultaneously. To avoid this condition, the user should select an alternate input using the ICS[2:0] control bits.

24.4.5 Module Sync Outputs

By default, the CCP module generates a CCP Sync signal from the rollover of the CCPxTMR register. This signal is made available to all of the other CCP modules as a Sync source, or it is made available to trigger another peripheral. The CCP Sync signal is separate from the module's device-level interrupts or other outputs.

There may be circumstances where another event signal may serve as a better basis for the CCP Sync signal or where an additional event output, other than the selected signal, is required. The CCP modules include user configuration options to handle these situations.

24.4.5.1 Alternate Sync Out

The ALTSYNC control bit (CCPxCON1[21]) allows the user to substitute a different synchronization/trigger output in place of the timer rollover for the CCP Sync signal. When ALTSYNC = 0, the CCP Sync output is the default timer rollover signal in all operating modes. When ALTSYNC = 1, the synchronization signal depends on the specific operating mode. Table 24-9 lists the alternate outputs available.

Table 24-9. Alternate Sync Output Signals

ALTSYNC	CCSEL	MOD[3:0]	Output Signal
0	x	All	Standard (default) CCP Sync Output
1	0	0000	Special Event Trigger Output (Timer)
1	0	All except '0000'	Output Compare Interrupt Event (Compare)
1	1	All	Input Capture Event (Capture)

24.4.5.2 Auxiliary Output Signal

The CCP modules can also generate a secondary output that is different from the CCP Sync signal (or its alternate version if ALTSYNC is set). The auxiliary output is intended to allow other digital peripherals to access internal CCP module signals, such as:

- Time Base Synchronization
- Peripheral Trigger and Clock Inputs
- Signal Gating

The type of output signal is selected using the AUXOUT[1:0] control bits (CCPxCON2[20:19]) and is dependent on the module operating mode. More options are available for each mode than with the alternate Sync output, as shown in Table 24-10.

Table 24-10. Auxiliary Output Signals

AUXOUT[1:0]	CCSEL	MOD[3:0]	Output Signal
00	x	xxxx	Disabled (no output)
01	0	'0000' (Timer modes)	Time Base Period Reset or Rollover
10			Special Event Trigger Output
11			No Output
01	0	'0001' through '1111' (Output Compare modes)	Time Base Period Reset or Rollover
10			Output Compare Event Signal
11			Output Compare Signal
01	1	'xxxx' (Input Capture modes)	Time Base Period Reset or Rollover
10			Reflects the Value of the CDIS Bit
11			Input Capture Event Signal

24.4.6 Sync and Triggered Operation

Synchronized ("Sync") and Triggered mode operations can be thought of as Complementary modes that affect the operation of the CCPxTMR registers in most of the module's major operating modes. Both use the SYNC[4:0] bits (CCPxCON1[20:16]) to determine the input signal source. The difference is how that signal affects the timer.

In Sync mode operation, the timer counts freely when enabled by the CCPON bit and is reset to zero when the input, selected by SYNC[4:0], is asserted. The timer immediately begins to count again from zero unless it is held for some other reason. Sync operation is used whenever the TRIGEN bit (CCPxCON1[23]) is cleared.

In Triggered mode operation, the timer is held in Reset until the input selected by SYNC[4:0] is asserted; when this occurs, the timer starts counting and continues to count until the TRCLR bit (CCPxSTAT[5]) is set. Triggered operation is used whenever the TRIGEN bit is set.

Depending on the specific device, the SYNC[4:0] bits allow for the selection of up to 32 internal or external sources. Some implemented sources may be available for triggered operation but not for Sync operation. In addition, '11111' (free-running counter) is not valid for Sync operation.

Sync and trigger operations play a major role in the module's operation in Timer and Output Compare modes by allowing chained and synchronized operation of multiple modules.

Table 24-11. Synchronization Sources

SYNC[4:0]	Synchronization Source
11111	None; Timer with Auto-Rollover (FFFFh → 0000h)
11010-11110	Reserved
11001	Comparator 3 Output
11000	Comparator 2 Output
10111	Comparator 1 Output
10110	Reserved
10101	UART3 TX Edge Detect
10100	UART3 RX Edge Detect
10011	CLC4 Output
10010	CLC3 Output
10001	CLC2 Output
10000	CLC1 Output

.....continued

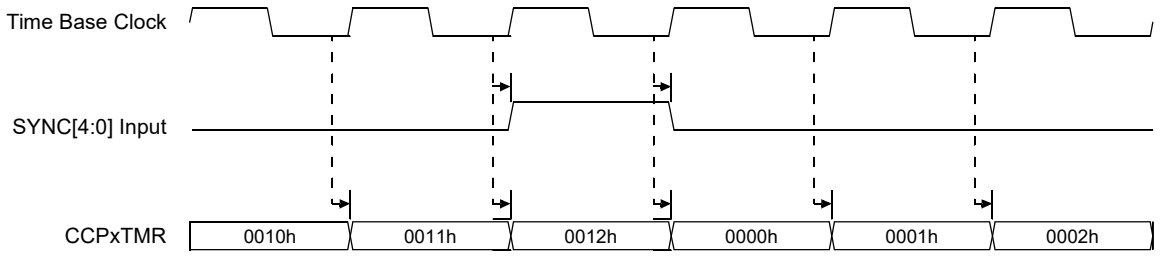
SYNC[4:0]	Synchronization Source
01111	UART2 TX Edge Detect
01110	UART2 RX Edge Detect
01101	UART1 TX Edge Detect
01100	UART1 RX Edge Detect
01011	INT2
01010	INT1
01001	INT0
00101-01000	Reserved
00100	Sync Output SCCP4
00011	Sync Output SCCP3
00010	Sync Output SCCP2
00001	Module's Own Timer Sync Out
00000	None; Timer with Rollover on CCPxPR Match or FFFFh

24.4.6.1 Timer Synchronized Operation

In Sync operation, the timer can be synchronized with other modules using the synchronization/trigger inputs selected by SYNC[4:0]. Basic operation is shown in Figure 24-24. Whenever the selected Sync input is asserted (high), the timer rolls over to 0000h on the next positive edge of the time base signal.

The timer functions in Synchronized operation when TRIGEN (CCPxCON1[23]) is cleared and the SYNC[4:0] bits have any value except '11111'. The CCPTRIG bit (CCPxSTAT[7]) has no function.

Figure 24-24. Timer Synchronized Operation



Selecting a SYNCx bits value of '00000' causes the module to run as a periodic timer with automatic rollover to 0000h when the Timer register matches the value of CCPxPR. A value of '11111' rolls over after an overflow from FFFFh.

For values of the SYNC[4:0] bits other than '00000' or '11111', the timer is reset when the input selected by the SYNCx bits is asserted.

The procedure for configuring the module for Synchronous operation is shown in Example 24-3.

Example 24-3. Setup for Synchronous Operation (16-Bit Dual Timer Mode)

```
CCP1CON1bits.TRIGEN = 0;     // Set Sync/Triggered mode (Synchronous Mode)
CCP1CON1bits.SYNC = 0;       // rolls over at FFFFh or match
                              // with period register (self sync)
CCP1CON1bits.T32 = 0;        // 16 bit dual timer mode
CCP1CON1bits.TMRSYNC = 0;    // Set timebase synchronization (Synchronized)
```

```
CCP1CON1bits.CLKSEL = 0;      // Set the clock source (Tcy)
CCP1CON1bits.TMRPS = 0;       // Set the clock pre-scaler (1:1)
CCP1PR = 0x00000FFF;          // 32 bit MCCP1 period register
CCP1CON1bits.CCPON = 1;       // Start the Timer
```

24.4.6.1.1 Synchronizing Multiple Modules

Each CCP module generates a CCP Sync output signal (see 24.4.5. Module Sync Outputs) that can be used to synchronize its operation with other modules. This signal is distinct from the module's interrupts or any other output signals. All of the CCP modules have access to each others' Sync signals through the SYNC[4:0] bits; this allows several modules to be chained together for more complex synchronized operations.

A simple example of Synchronized operation is shown in Figure 24-25. In this instance, CCP2 is being synchronized to CCP1. Each module has been configured to use the same clock source for their time bases. In addition, both modules use the CCP Sync signal from CCP1 as their Sync source inputs. CCP1PR now serves as the period register for both CCP1 and CCP2.

Figure 24-26 shows the timing relationship between the two modules. When a match between CCP1TMR and CCP1PR occurs, the Sync signal goes active. This causes the timers in both CCP1 and CCP2 to go to 0000h on the next positive timer input clock edge.

When synchronizing modules, there are two important things to keep in mind:

- All synchronized modules are to use the same clock source for their time bases.
- When initializing synchronized modules, the module being used as the synchronization source should be enabled last. This ensures that the timers of all synchronized modules are maintained in a Reset condition until the last module is initialized.

Figure 24-25. Example of Two Synchronized Timers (CCP2 Synchronized to CCP1)

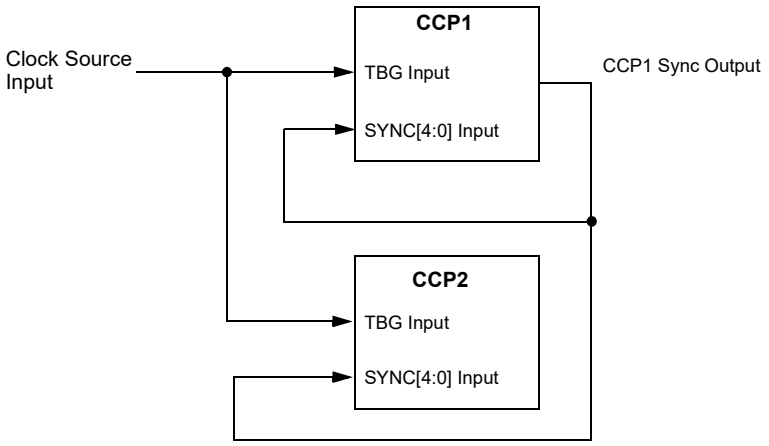
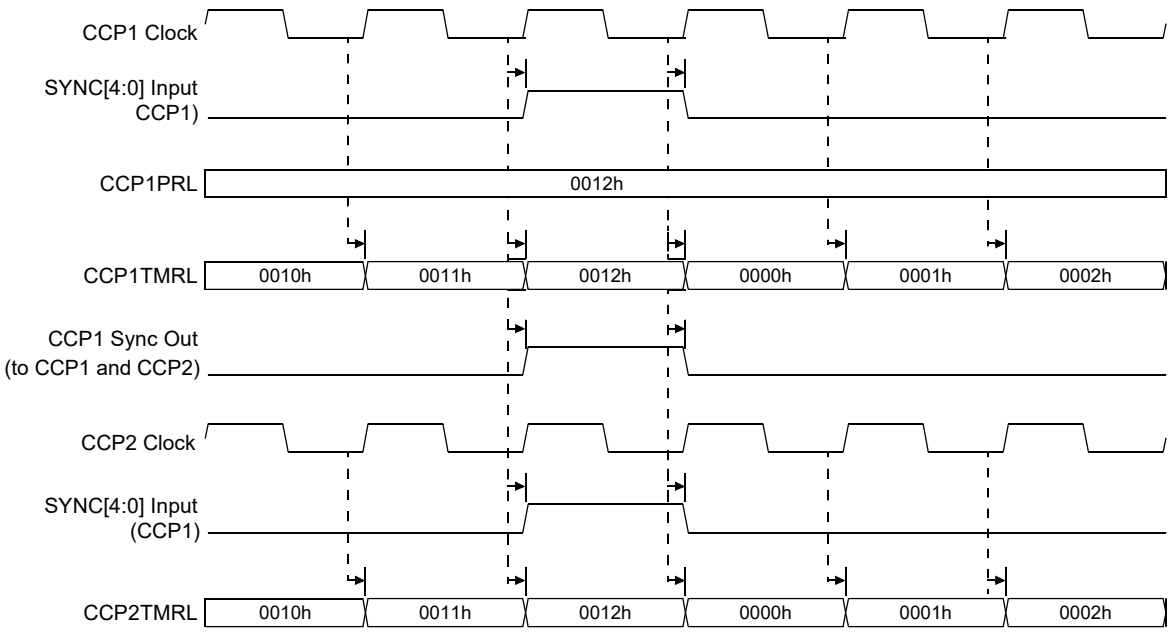


Figure 24-26. CCP1 and CCP2 Timers in Sync



24.4.6.2 Timer Triggered Operation

Triggered operation of the timer is enabled when TRIGEN = 1. Triggered mode operation is useful in creating time delays. A pulse or edge event can be generated after the delay, depending on the module operating mode.

When configured for triggered operation, the module timer is held in Reset until a trigger event with the source selected by SYNC[4:0] occurs. After the trigger event occurs, the timer begins to count. The timer increments on every positive clock of the time base signal.

If the timer is configured for 16-bit dual timer operation (T32 = 0), only the timer based on CCPxTMRL will function in triggered operation. The timer, based on CCPxTMRH, will operate as a free-running timer.

The CCPTRIG status bit (CCPxSTAT[7]) indicates whether the timer is held in Reset or released to count. When CCPTRIG = 0, the timer is being held in Reset; when CCPTRIG = 1, the timer has been released.

There are two types of trigger conditions when operating in Triggered mode: Hardware/Software and Software-Only. Hardware/software triggered operation is shown in Figure 24-27. When the module is enabled for a triggered response, the timer is held in Reset. It remains in this state until a trigger event is asserted for the SYNC[4:0] input, which sets the CCPTRIG bit within two clock cycles. The trigger signal determines only when the time base starts counting; the CCPxPR register sets the period for the timer. Unlike Sync operation, all trigger sources available through the SYNC[4:0] bits may be used for triggered operation.

CCPTRIG can be manually set at any time and the timer can be released from Reset by writing a '1' to the TRSET bit (CCPxSTAT[6]). The CCPTRIG bit can also be manually cleared in software by writing a '1' to the TRCLR bit (CCPxSTAT[5]).

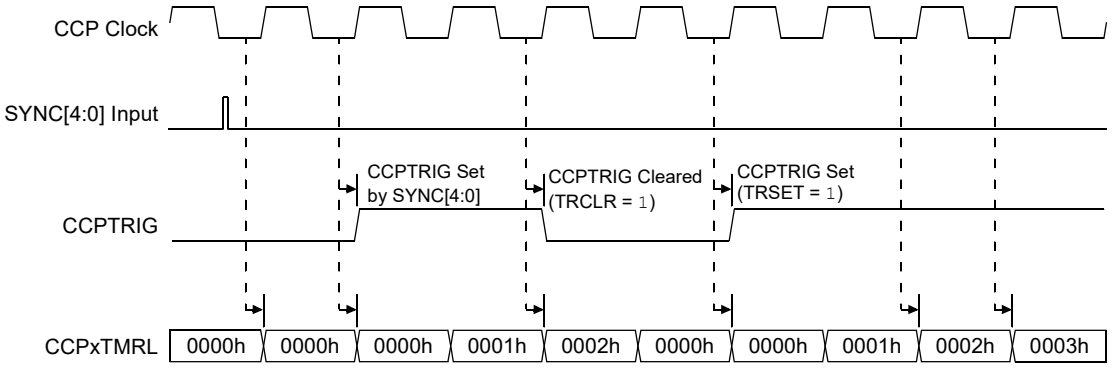
Software-Only operation is selected when SYNC[4:0] = 11111. In this configuration, the only way that the CCPTRIG bit can be set is by a software write to the TRSET bit. This selection effectively disables all external hardware trigger sources.

When the TRIGEN bit is cleared in software, the timer is reset to 0000h on the next timer clock rising edge and is ready for another SYNC[4:0].

Note: The TRSET and TRCLR bits are write-only bits which always read as '0'. Writing '0' to either location has no effect.

The procedure for configuring the module for triggered operation is shown in Example 24-4.

Figure 24-27. Timing for Triggered Operation (Hardware/Software Operations)



Example 24-4. Setup for Timer Triggered Operation (16-Bit Dual Timer Mode)

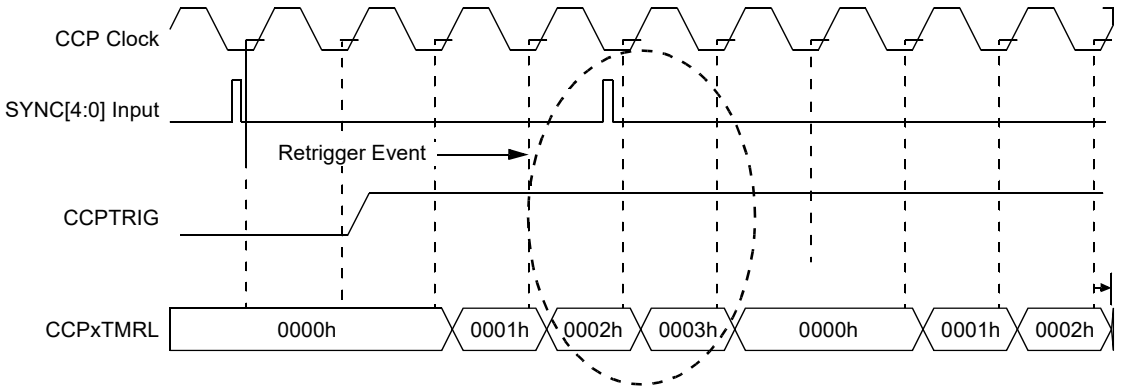
```
CCP1CON1bits.TRIGEN=1;        // Set Sync/Triggered mode (Triggered Mode)
CCP1CON1bits.SYNC = 0x08;     // INT0 as trigger (verify the device
                               // for Trigger source)
CCP1CON1bits.T32=0;           // 16 bit dual timer mode
CCP1CON1bits.TMRSYNC = 0;     // Set timebase synchronization (Synchronized)
CCP1CON1bits.CLKSEL = 0;      // Set the clock source (Tcy)
CCP1CON1bits.TMRPS = 0;       // Set the clock pre-scaler (1:1)
CCP1PR = 0x00000FFF;          // 32-bit MCCP1 period register
CCP1CON1bits.CCPON=1;         // Enable the Timer
```

24.4.6.2.1 Retrigger Operation

The RTRGEN bit (CCPxCON1[30]) allows the timer to be retriggered while the CCPTRIG bit remains set. When RTRGEN is set, a second trigger event occurring during trigger operation will cause the timer to reset and start counting again. Figure 24-28 shows how the timer restarts counting when the trigger comes again, before the timer overflow happens.

When RTRGEN = 1, multiple trigger pulses occurring within the same time base clock period will not be recognized and will be treated as a single trigger event. If trigger pulses are received on two adjacent timer clock periods, the time base will be held in Reset (0000h) for one additional clock period.

Figure 24-28. Retrigger Operation (RTRGEN = 1)



Note: In the example, if the CCPxPRL is 10h, the timer will reset before the 10h since the retrigger came before the timer reached 10h.

24.4.6.2.2 Triggered Operation with Asynchronous Clock

The module time base can operate from a variety of clock sources; these may or may not be synchronous to the system clock. In addition, the trigger source may be asynchronous to the module time base clock. To minimize glitches, the incoming trigger signal is latched and synchronized to the module's time base clock source by default.

When the time base clock source is asynchronous to the system clock, there will be a delay of up to two system clock cycles before the trigger state is reflected in the value of the CCPTRIG status bit.

When the time base clock source is asynchronous to the system clock, there will be a delay of up to two time base clock cycles before a trigger set or clear request from software affects the trigger state of the module.

24.4.6.2.3 Timer Rollover in Triggered Operation

When the module is configured for triggered operation, the signal source selected by SYNC[4:0] does not set the time base count period. The primary purpose of the trigger signal is to tell the timer when to start counting, not when to reset (as in Sync mode).

The timer rolls over to 0000h on the next clock after CCPxPRH/L matches CCPxTMRH/L or when CCPxTMRH/L reaches FFFFh (if CCPxPRH/L is not available in the specific operating mode of the module).

There are two values of SYNC[4:0] that are not allowed in triggered operation:

- '00000' (synchronous timer, external triggers are disabled)
- Any value that selects the module's own CCP Sync signal (a trigger must be external)

If the trigger source selected by SYNC[4:0] is initialized and enabled first, there is a chance that the timer will miss trigger events. Therefore, it is recommended that the timer be initialized and enabled before the trigger source.

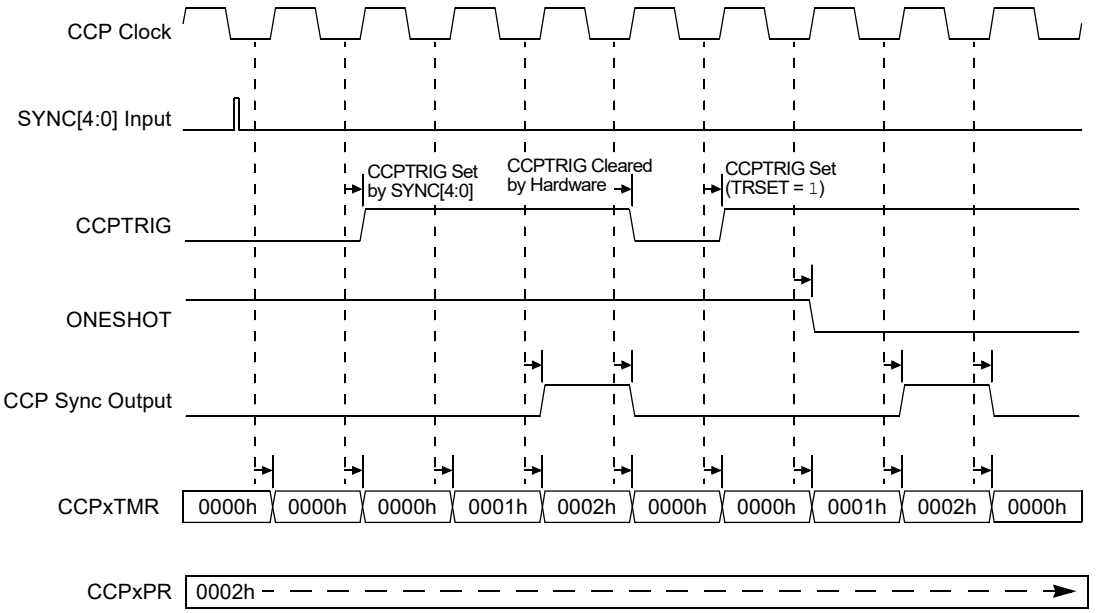
24.4.6.2.4 One-Shot Functionality

While in triggered operation, the timer can operate in a One-Shot mode. In this mode, the timer remains in Reset until a hardware or software trigger event occurs. This event sets the CCPTRIG bit and the timer begins to count. When the timer rolls over to 0000h, the CCPTRIG bit is cleared by hardware. This holds the timer in Reset until the next trigger event, creating a one-shot timer.

One-Shot mode is enabled by setting the ONESHOT bit (CCPxCON1[22]). The OSCNT[2:0] control bits (CCPxCON3[30:28]) allow a one-shot trigger event to be extended for more than one CCP timer clock cycle. This feature is useful, for example, when the module needs to create more than one pulse at a trigger event.

Note: Do not modify OSCNT[2:0] while the module is triggered (CCPTRIG = 1); unexpected results may occur.

Figure 24-29. Timing for One-Shot Functionality



24.4.7 Register Access

24.4.7.1 Time Base Access

The CCPxTMR register provides user access to the time base value in all operating modes of the module. It is readable and writable. The CCPxTMR register is 16 or 32 bits wide, depending on the operating mode and the value of the T32 (CCPxCON1[5]), CCSEL (CCPxCON1[4] and MOD[3:0] (CCPxCON1[3:0] control bits.

24.4.7.2 Input Capture Data Buffer Access

The CCPxBUFL register bits' location provides user access to a four-level deep FIFO used in Input Capture modes. During normal operation, the CCPxBUFL register bits location is read until ICBNE (CCPxSTAT[0]) = 0. The ICOV (CCPxSTAT[1]) status bit tells the user if the FIFO buffer size has been exceeded, therefore losing the most recently captured data.

When operating in a 32-bit Input Capture mode, a two-level FIFO buffer is available. The CCPxBUF register provides access to the FIFO. The ICBNE and ICOV status bits operate similar to 16-bit capture operations.

24.4.7.3 Compare Modes Data Write

Three data registers are provided for Output Compare modes:

- CCPxPR
- CCPxRA
- CCPxRB

CCPxPR provides a value for comparison to the module timer; the match is used to:

- Set the count period of the time base

- Generate CCP Sync outputs for other modules

CCPxRA provides a value for comparison to the module timer; the match is used:

- As a second compare register in Dual Compare mode to create all negative edges
- As the upper 16 bits of the compare value in 32-bit single edge compare operations

When the module operates with a 32-bit time base (T32 = 1), CCPxRB and CCPxRA are combined to form a single 32-bit Compare register. The CCPxRB value is compared to the upper 16 bits of the time base and the CCPxRA value is compared to the lower 16 bits. The CCPxRB/CCPxRA register pair performs these functions in 32-bit modes:

- Edge generation in Single Edge Output Compare modes
- Generates the ccp_trig_out signal

A Time Base Period register is not available in the 32-Bit Single Edge Output Compare mode. The CCP Sync output signal is generated only on a rollover from FFFFFFFF to 00000000.

24.4.7.4 CCPxPRL, CCPxRA and CCPxRB Write Status

In the operating modes, where the CCPxPRL, CCPxRA and CCPxRB are double-buffered, the PRLWIP (CCPxSTAT[20], RAWIP (CCPxSTAT[16]) and RBWIP (CCPxSTAT[17]) status bits indicate the buffer update status for these registers. The appropriate bit will set when one of these registers is written and will remain set until the buffered value takes effect, typically at the time base period boundary.

Note: It is not necessary to monitor these three bits if software always updates the registers at a specific time in the time base count period. This would occur, for example, when the time base period interrupt is used to schedule an update in the software. If, however, software is expected to write these registers at a random time with respect to the time base count period, then the appropriate status bit should be checked in software prior to performing the write.

Further writes to the CCPxPRL and/or CCPxRA/B registers should not be performed while the associated WIP status bit is set.

24.4.7.5 Data Register Manipulation with Asynchronous Clock Sources

Special consideration must be taken when certain data registers are manipulated using an asynchronous time base clock source. The time base clock source will be asynchronous to the system clock when TMRSYNC = 0 and CLKSEL[2:0] != 000.

24.4.7.5.1 CCPxTMR Reads with Asynchronous Clock Source

When the time base clock source is asynchronous to the system clock source, a read of CCPxTMR may produce unexpected results. This can be avoided by reading the CCPxTMR twice and comparing the read results, or the time base value can be captured using the Input Capture logic.

Note: A second CCP peripheral may be synchronized to the primary CCP and used in Input Capture mode, if needed.

24.4.7.5.2 CCPxTMR Writes with Asynchronous Clock Source

When an asynchronous clock source is used, writes to CCPxTMR are buffered and the write progress is indicated using the TMRHWIP (CCPxSTAT[19]) and TMRLWIP (CCPxSTAT[18]) status bits. The write to the CCPxTMR register is complete when the appropriate WIP status bit is cleared by hardware. Another write can be safely performed after the WIP is cleared.

Note: These status bits are also affected when the clock is synchronous. However, it is not necessary to monitor these bits since the write will occur on the next system clock cycle.

The timer WIP bits are set as follows:

- In Dual 16-Bit Timer mode, TMRHWIP is set upon a write that includes the highest byte of CCPxTMRH. TMRLWIP is set upon a write that includes the highest byte of CCPxTMRL.
- In modes that use a single 16-bit timer, TMRLWIP is set upon a write that includes the highest byte of CCPxTMRL.

- In modes that use a 32-bit timer, TMRLWIP and TMRHWIP are set upon a write that includes the highest byte of CCPxTMRH.

User software should not perform any writes to CCPxTMR while the appropriate WIP bit is set; otherwise, an unexpected timer value may result.

Note: The TMRHWIP and TMRLWIP bits get set automatically when the CCPON bit is set. This ensures any modifications to the CCPxTMR register while the macro is off will take effect. The user may monitor the WIP bits after the CCPON bit is set to determine when the CCPxTMR contents have been updated and the time base has begun counting.

24.5 Operation During Sleep and Idle Modes

24.5.1 Idle Mode

The behavior of the module in Idle mode is determined by the CCPSIDL bit (CCPxCON1[13]). If CCPSIDL is cleared, the module will continue to operate in Idle mode. If CCPSIDL is set, the module is disabled when the device enters Idle mode. If the module is performing an operation when Idle mode is invoked, in this case, the results will be similar to those with Sleep mode.

24.5.2 Sleep Modes

The behavior of the module in Sleep mode is determined by the CCPSLP bit (CCPxCON1[12]). If CCPSLP is set, the module will continue to operate during Sleep mode, assuming that the selected clock source remains available. The TMRSYNC bit must remain cleared for the module to operate in Sleep mode.

When CCPSLP is cleared and the device enters Sleep mode, the module is disabled. However, if CCPSLP is set and the module is configured for 16-Bit Edge Detect Input Capture mode (MOD[3:0] = `0000`, CCSEL = `1`), the module can generate an interrupt and wake up the device as long as the clock source remains active. In this configuration, the Input Capture pin can function like an external interrupt. The corresponding CCP interrupt must also be enabled (CCPxIE = `1`).

24.5.2.1 Triggered Operation and Sleep Mode

When the module is configured for triggered operation, a trigger signal received from an external source can also wake the module and its time base clock source. The module must request the time base clock source before triggered operation can begin. When the trigger is received from the external source, the CCP module will enable the selected clock source for the time base. When the clock source becomes available, the module will begin triggered operation.

If One-Shot Triggered mode is selected, the time base clock source will be disabled when the CCPTRIG status bit is cleared in hardware. The time base remains disabled until a new trigger signal is received.

The trigger signal can also be generated by an internal source that operates from a low-power clock source.

If Sleep mode operation is enabled, the module will continue to request the configured clock source when the device enters Sleep mode.

24.5.3 Interrupts

The CCP module has the ability to generate the interrupt on the period match input capture event or output compare event.

CCTxIF is set when the TMRx count matches the respective PRx register. CCPxIF is set whenever there is an input capture event or an output compare equal event. The CCTxIF and CCPxIF bits must be cleared in software.

The CCP module is enabled as a source of interrupt via the CCP Interrupt Enable bit, CCTxIE and CCPxIE. The CCP Interrupt Priority Level is defined by CCP Interrupt Priority bits CCTxIP[2:0] and CCPxIP[2:0].

Note: A special case occurs when the PRx register is loaded with '0' and the timer is enabled. An interrupt is not generated for this configuration.

24.5.3.1 Interrupt Configuration

The CCPx module has a dedicated Interrupt flag bit (CCT/PxIF) and a corresponding Interrupt Enable/Mask bit (CCT/PxIE).

The CCT/PxIE bit is used to define the behavior of the interrupt controller when the CCT/PxIF bit is set. When the CCT/PxIE bit is clear, the interrupt controller does not generate a CPU interrupt for the event. If the CCT/PxIE bit is set, the interrupt controller generates an interrupt to the CPU when the CCT/PxIF bit is set (subject to the interrupt priority).

The software routine that services a particular interrupt must clear the appropriate Interrupt Flag bit before the service routine is complete. The priority of the CCPx module can be set with the CCT/PxIP[2:0] bits. This priority defines the priority group to which the interrupt source will be assigned. The priority groups range from a value of seven (the highest priority) to a value of 0 (which does not generate an interrupt). An interrupt being serviced will be preempted by an interrupt in a higher priority group.

After an enabled interrupt is generated, the CPU will jump to the vector assigned to that interrupt. The vector number for the interrupt is the same as the natural order number. The CPU will then begin executing code at the vector address. Code at this vector address should perform any application-specific operations and clear the CCT/Px1IF interrupt flag and then exit.

24.6 Effects of a Reset

A device Reset forces all registers to their Reset state; this disables the module and returns it to its default configuration (16-bit timer). All buffer and address registers are initialized to 0000h and all status flags are reset.

By default, the pin associated with OCxA resets with the Output Compare function in control of the pin; however, this has no effect when the module is disabled.

25. Configurable Logic Cell (CLC)

The Configurable Logic Cell (CLC) module allows the user to specify combinations of signals as inputs to a logic function and to use the logic output to control other peripherals or I/O pins. This provides greater flexibility and potential in embedded designs since the CLC module can operate outside the limitations of software execution and supports a vast number of output designs.

High-level features include:

- General Purpose Logic Block
- 32 Input Sources, Including External Pins and Other Device Peripheral Outputs
- Combinational Logic Modes
- Sequential Logic Modes
- Edge Detection and Interrupt Generation When Processor is in Sleep
- Delay Generation
- Inversion Logic
- Signal Routing Between Otherwise Unconnected Peripherals

25.1 Device-Specific Information

Table 25-1. CLC Summary Table

CLC Module Instances	Inputs per Instance	CLC Outputs	Peripheral Bus Speed
4	8	4	Slow

Table 25-2. DS1 Data Selection MUX 1 Signal Selection bits (Main)

Value (binary)	Description
111	SCCP4 auxiliary output
110	SCCP2 auxiliary output
101	CLKGEN13
100	REFO1 clock output
011	INTRC/LPRC clock source
010	CLC3 out
001	Slow peripheral clock (system clock/4)
000	CLCINA I/O pin

Table 25-3. DS2 Data Selection MUX 2 Signal Selection bits (Main)

Value (binary)	Description
111	SCCP2 OCMP output
110	SCCP1 OCMP output
101	SCCP2 trigger output
100	SCCP1 trigger output
011	UART1 TX output
010	Comparator 1 output
001	CLC2 output
000	CLCINB I/O pin

Table 25-4. DS3 Data Selection MUX 3 Signal Selection bits (Main)

Value (binary)	Description
111	SCCP4 OCMP output

.....continued	
Value (binary)	**Description**
110	SCCP3 OCMP output
101	CLC4 out
100	UART1 RX input
011	SPI1 Output (SDOx)
010	Comparator 2 output
001	CLC1 output
000	CLCINC I/O pin

Table 25-5. DS4 Data Selection MUX 4 Signal Selection bits (Main)

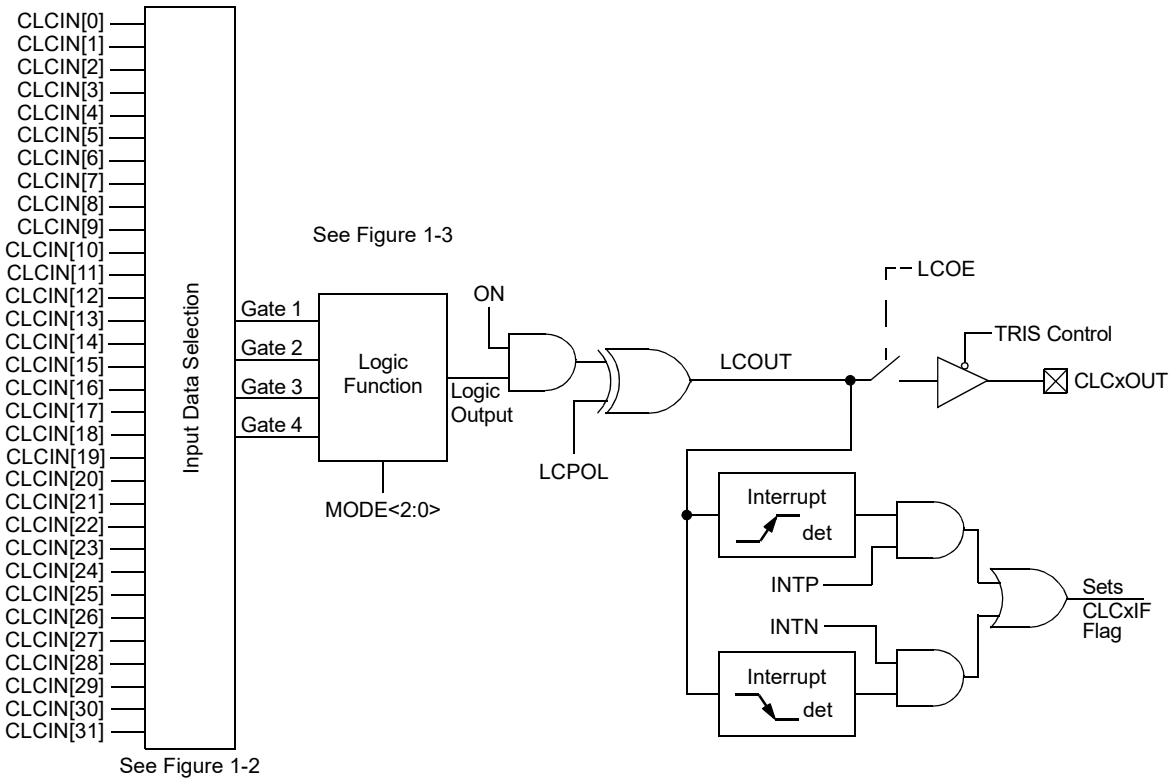
Value (binary)	Description
111	SCCP3 auxiliary out
110	SCCP1 auxiliary out
101	CLCIND I/O pin
100	Reserved
011	SPI1 Input (SDIx)(1)
010	Comparator 3 output
001	CLC2 output
000	PWM Event A

Note:
1. Valid only for the SPI with PPS selection.

25.2 Architecture

The CLC consists of four main sections, as shown in Figure 25-1. First, the input data selection MUXes route input signals to the four data gates, as shown in Figure 25-2. Each of the four data gates can then select any of the 32 input signals to pass along to the logic functions shown in Figure 25-3. The output of the logic function is then supplied to the internal logic and external pin and can generate interrupts. The output of a CLC module can be routed to the input of another CLC module to create more complex logic functions.

Figure 25-1. Configurable Logic Cell



Note: All control bits shown in this figure can be found in the CLCxCON register.

Figure 25-2. CLC Input Source Selection Diagram

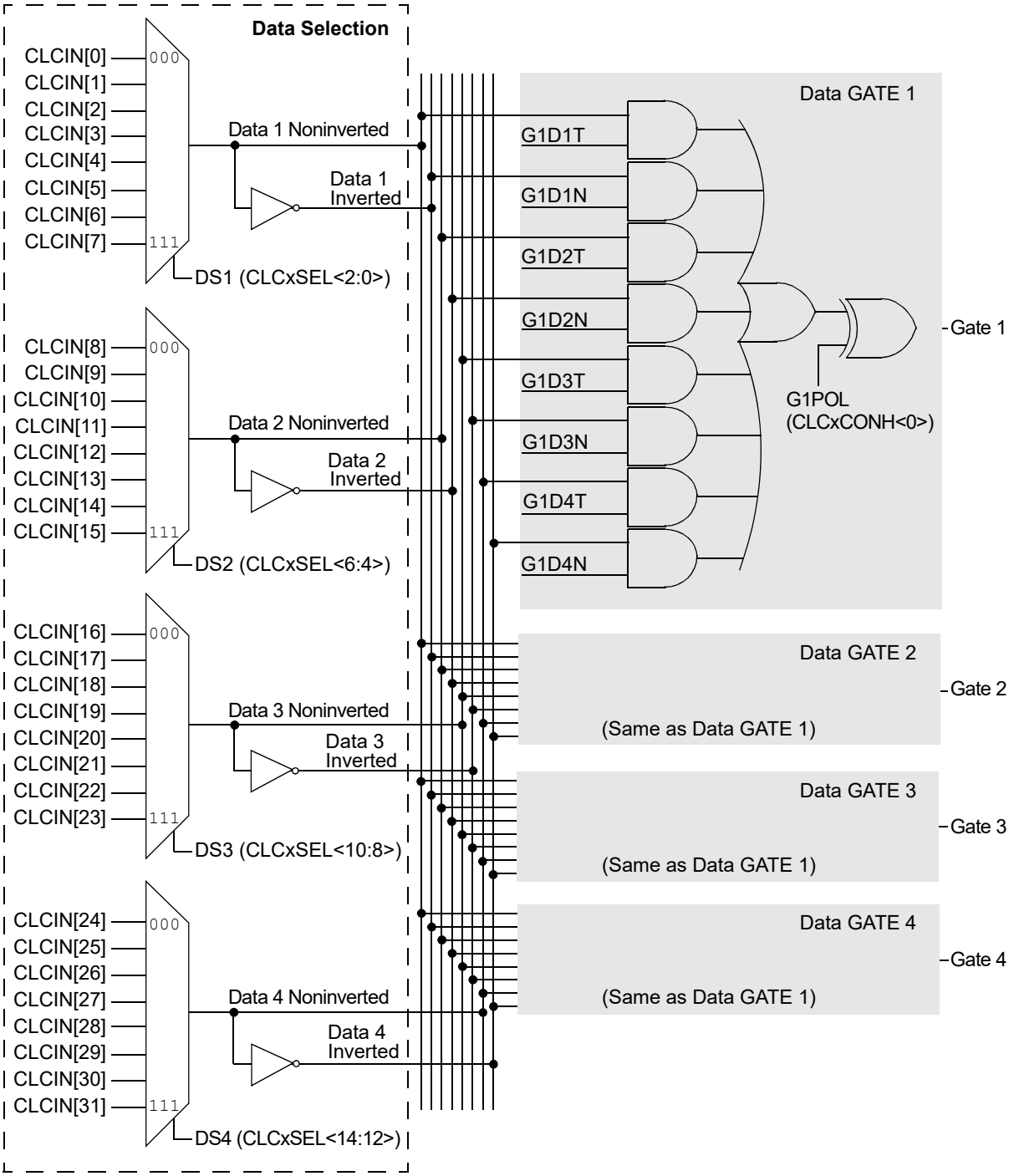
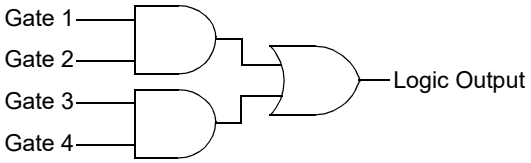


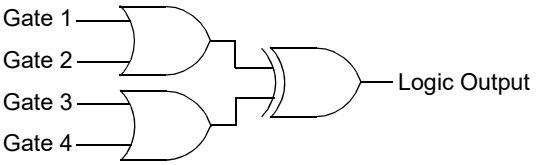
Figure 25-3. Logic Function Combinatorial Options

AND – OR



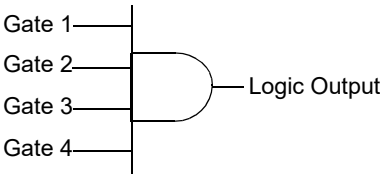
MODE<2:0> = 000

OR – XOR



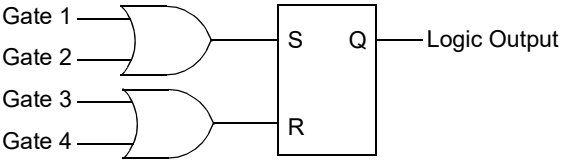
MODE<2:0> = 001

4-Input AND



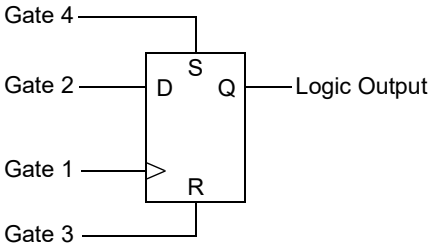
MODE<2:0> = 010

S-R Latch



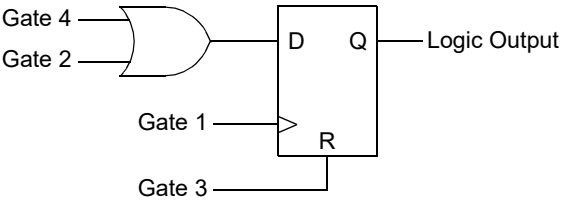
MODE<2:0> = 011

1-Input D Flip-Flop with S and R



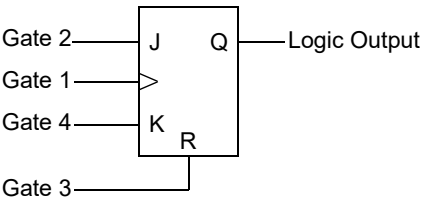
MODE<2:0> = 100

2-Input D Flip-Flop with R



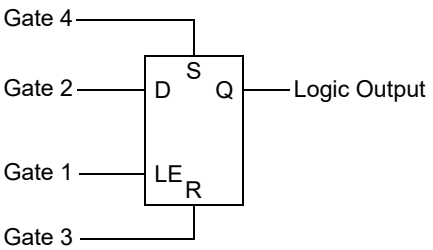
MODE<2:0> = 101

J-K Flip-Flop with R



MODE<2:0> = 110

1-Input Transparent Latch with S and R



MODE<2:0> = 111

25.3 CLC Control Registers

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3A60	CLC1CON	31:24								
		23:16					G4POL	G3POL	G2POL	G1POL
		15:8	ON				INTP	INTN		
		7:0	LCOE	LCOUT	LCPOL			MODE[2:0]		
0x3A64	CLC1SEL	31:24								
		23:16								
		15:8		DS4[2:0]				DS3[2:0]		
		7:0		DS2[2:0]				DS1[2:0]		
0x3A68	CLC1GLS	31:24	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
		23:16	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
		15:8	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
		7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x3A6C ... 0x3A6F	Reserved									
0x3A70	CLC2CON	31:24								
		23:16					G4POL	G3POL	G2POL	G1POL
		15:8	ON				INTP	INTN		
		7:0	LCOE	LCOUT	LCPOL			MODE[2:0]		
0x3A74	CLC2SEL	31:24								
		23:16								
		15:8		DS4[2:0]				DS3[2:0]		
		7:0		DS2[2:0]				DS1[2:0]		
0x3A78	CLC2GLS	31:24	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
		23:16	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
		15:8	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
		7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x3A7C ... 0x3A7F	Reserved									
0x3A80	CLC3CON	31:24								
		23:16					G4POL	G3POL	G2POL	G1POL
		15:8	ON				INTP	INTN		
		7:0	LCOE	LCOUT	LCPOL			MODE[2:0]		
0x3A84	CLC3SEL	31:24								
		23:16								
		15:8		DS4[2:0]				DS3[2:0]		
		7:0		DS2[2:0]				DS1[2:0]		
0x3A88	CLC3GLS	31:24	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
		23:16	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
		15:8	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
		7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x3A8C ... 0x3A8F	Reserved									
0x3A90	CLC4CON	31:24								
		23:16					G4POL	G3POL	G2POL	G1POL
		15:8	ON				INTP	INTN		
		7:0	LCOE	LCOUT	LCPOL			MODE[2:0]		
0x3A94	CLC4SEL	31:24								
		23:16								
		15:8		DS4[2:0]				DS3[2:0]		
		7:0		DS2[2:0]				DS1[2:0]		
0x3A98	CLC4GLS	31:24	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
		23:16	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
		15:8	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
		7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N

25.3.1 Configurable Logic Cell x Control Register

Name: CLCxCON
Offset: 0x3A60, 0x3A70, 0x3A80, 0x3A90

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					G4POL	G3POL	G2POL	G1POL
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON				INTP	INTN		
Access	R/W				R/W	R/W		
Reset	0				0	0		

Bit	7	6	5	4	3	2	1	0
	LCOE	LCOUT	LCPOL			MODE[2:0]		
Access	R/W	R	R			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bit 19 – G4POL Gate 4 Polarity Control bit

Value	Description
1	The output of Gate 4 logic is inverted when applied to the logic cell
0	The output of Gate 4 logic is not inverted

Bit 18 – G3POL Gate 3 Polarity Control bit

Value	Description
1	The output of Gate 3 logic is inverted when applied to the logic cell
0	The output of Gate 3 logic is not inverted

Bit 17 – G2POL Gate 2 Polarity Control bit

Value	Description
1	The output of Gate 2 logic is inverted when applied to the logic cell
0	The output of Gate 2 logic is not inverted

Bit 16 – G1POL Gate 1 Polarity Control bit

Value	Description
1	The output of Gate 1 logic is inverted when applied to the logic cell
0	The output of Gate 1 logic is not inverted

Bit 15 – ON Configurable Logic Cell Enable bit

Value	Description
1	Configurable Logic Cell is enabled and mixing input signals
0	Configurable Logic Cell is disabled and has logic zero outputs

Bit 11 – INTP Configurable Logic Cell Positive Edge Interrupt Enable bit

Value	Description
1	Interrupt will be generated when a rising edge occurs on LCOUT
0	Interrupt will not be generated

Bit 10 – INTN Configurable Logic Cell Negative Edge Interrupt Enable bit

Value	Description
1	Interrupt will be generated when a falling edge occurs on LCOUT
0	Interrupt will not be generated

Bit 7 – LCOE Configurable Logic Cell Port Enable bit

Value	Description
1	Configurable Logic Cell port pin output is enabled
0	Configurable Logic Cell port pin output is disabled

Bit 6 – LCOUT Configurable Logic Cell Data Output Status bit

Value	Description
1	Configurable Logic Cell output high
0	Configurable Logic Cell output low

Bit 5 – LCPOL Configurable Logic Cell Output Polarity Control bit

Value	Description
1	The output of the module is inverted
0	The output of the module is not inverted

Bits 2:0 – MODE[2:0] Configurable Logic Cell Mode bits

Value	Description
111	Cell is one-input transparent latch with S and R
110	Cell is J-K flip-flop with R
101	Cell is two-input D flip-flop with R
100	Cell is one-input D flip-flop with S and R
011	Cell is SR latch
010	Cell is four-input AND
001	Cell is OR-XOR
000	Cell is AND-OR

25.3.2 Configurable Logic Cell x Input MUX Select Register

Name: CLCxSEL
Offset: 0x3A64, 0x3A74, 0x3A84, 0x3A94

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
		DS4[2:0]				DS3[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit	7	6	5	4	3	2	1	0
		DS2[2:0]				DS1[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bits 14:12 – DS4[2:0] Data Selection MUX 4 Signal Selection bits (Main)
See Table 25-5.

Bits 10:8 – DS3[2:0] Data Selection MUX 3 Signal Selection bits (Main)
See Table 25-4.

Bits 6:4 – DS2[2:0] Data Selection MUX 2 Signal Selection bits (Main)
See Table 25-3.

Bits 2:0 – DS1[2:0] Data Selection MUX 1 Signal Selection bits (Main)
See Table 25-2.

25.3.3 Configurable Logic Cell x Source Enable Register

Name: CLCxGLS
Offset: 0x3A68, 0x3A78, 0x3A88, 0x3A98

Bit	31	30	29	28	27	26	25	24
	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – G4D4T Gate 4 Data 4 True Enable bit

Value	Description
1	The Data 4 (noninverted) signal is enabled for Gate 4
0	The Data 4 (noninverted) signal is disabled for Gate 4

Bit 30 – G4D4N Gate 4 Data 4 Negated Enable bit

Value	Description
1	The Data 4 (inverted) signal is enabled for Gate 4
0	The Data 4 (inverted) signal is disabled for Gate 4

Bit 29 – G4D3T Gate 4 Data 3 True Enable bit

Value	Description
1	The Data 3 (noninverted) signal is enabled for Gate 4
0	The Data 3 (noninverted) signal is disabled for Gate 4

Bit 28 – G4D3N Gate 4 Data 3 Negated Enable bit

Value	Description
1	The Data 3 (inverted) signal is enabled for Gate 4
0	The Data 3 (inverted) signal is disabled for Gate 4

Bit 27 – G4D2T Gate 4 Data 2 True Enable bit

Value	Description
1	The Data 2 (noninverted) signal is enabled for Gate 4
0	The Data 2 (noninverted) signal is disabled for Gate 4

Bit 26 – G4D2N Gate 4 Data 2 Negated Enable bit

Value	Description
1	The Data 2 (inverted) signal is enabled for Gate 4
0	The Data 2 (inverted) signal is disabled for Gate 4

Bit 25 – G4D1T Gate 4 Data 1 True Enable bit

Value	Description
1	The Data 1 (noninverted) signal is enabled for Gate 4
0	The Data 1 (noninverted) signal is disabled for Gate 4

Bit 24 – G4D1N Gate 4 Data 1 Negated Enable bit

Value	Description
1	The Data 1 (inverted) signal is enabled for Gate 4
0	The Data 1 (inverted) signal is disabled for Gate 4

Bit 23 – G3D4T Gate 3 Data 4 True Enable bit

Value	Description
1	The Data 4 (noninverted) signal is enabled for Gate 3
0	The Data 4 (noninverted) signal is disabled for Gate 3

Bit 22 – G3D4N Gate 3 Data 4 Negated Enable bit

Value	Description
1	The Data 4 (inverted) signal is enabled for Gate 3
0	The Data 4 (inverted) signal is disabled for Gate 3

Bit 21 – G3D3T Gate 3 Data 3 True Enable bit

Value	Description
1	The Data 3 (noninverted) signal is enabled for Gate 3
0	The Data 3 (noninverted) signal is disabled for Gate 3

Bit 20 – G3D3N Gate 3 Data 3 Negated Enable bit

Value	Description
1	The Data 3 (inverted) signal is enabled for Gate 3
0	The Data 3 (inverted) signal is disabled for Gate 3

Bit 19 – G3D2T Gate 3 Data 2 True Enable bit

Value	Description
1	The Data 2 (noninverted) signal is enabled for Gate 3
0	The Data 2 (noninverted) signal is disabled for Gate 3

Bit 18 – G3D2N Gate 3 Data 2 Negated Enable bit

Value	Description
1	The Data 2 (inverted) signal is enabled for Gate 3
0	The Data 2 (inverted) signal is disabled for Gate 3

Bit 17 – G3D1T Gate 3 Data 1 True Enable bit

Value	Description
1	The Data 1 (noninverted) signal is enabled for Gate 3
0	The Data 1 (noninverted) signal is disabled for Gate 3

Bit 16 – G3D1N Gate 3 Data 1 Negated Enable bit

Value	Description
1	The Data 1 (inverted) signal is enabled for Gate 3
0	The Data 1 (inverted) signal is disabled for Gate 3

Bit 15 – G2D4T Gate 2 Data 4 True Enable bit

Value	Description
1	The Data 4 (noninverted) signal is enabled for Gate 2
0	The Data 4 (noninverted) signal is disabled for Gate 2

Bit 14 – G2D4N Gate 2 Data 4 Negated Enable bit

Value	Description
1	The Data 4 (inverted) signal is enabled for Gate 2
0	The Data 4 (inverted) signal is disabled for Gate 2

Bit 13 – G2D3T Gate 2 Data 3 True Enable bit

Value	Description
1	The Data 3 (noninverted) signal is enabled for Gate 2
0	The Data 3 (noninverted) signal is disabled for Gate 2

Bit 12 – G2D3N Gate 2 Data 3 Negated Enable bit

Value	Description
1	The Data 3 (inverted) signal is enabled for Gate 2
0	The Data 3 (inverted) signal is disabled for Gate 2

Bit 11 – G2D2T Gate 2 Data 2 True Enable bit

Value	Description
1	The Data 2 (noninverted) signal is enabled for Gate 2
0	The Data 2 (noninverted) signal is disabled for Gate 2

Bit 10 – G2D2N Gate 2 Data 2 Negated Enable bit

Value	Description
1	The Data 2 (inverted) signal is enabled for Gate 2
0	The Data 2 (inverted) signal is disabled for Gate 2

Bit 9 – G2D1T Gate 2 Data 1 True Enable bit

Value	Description
1	The Data 1 (noninverted) signal is enabled for Gate 2
0	The Data 1 (noninverted) signal is disabled for Gate 2

Bit 8 – G2D1N Gate 2 Data 1 Negated Enable bit

Value	Description
1	The Data 1 (inverted) signal is enabled for Gate 2
0	The Data 1 (inverted) signal is disabled for Gate 2

Bit 7 – G1D4T Gate 1 Data 4 True Enable bit

Value	Description
1	The Data 4 (noninverted) signal is enabled for Gate 1
0	The Data 4 (noninverted) signal is disabled for Gate 1

Bit 6 – G1D4N Gate 1 Data 4 Negated Enable bit

Value	Description
1	The Data 4 (inverted) signal is enabled for Gate 1
0	The Data 4 (inverted) signal is disabled for Gate 1

Bit 5 – G1D3T Gate 1 Data 3 True Enable bit

Value	Description
1	The Data 3 (noninverted) signal is enabled for Gate 1
0	The Data 3 (noninverted) signal is disabled for Gate 1

Bit 4 – G1D3N Gate 1 Data 3 Negated Enable bit

Value	Description
1	The Data 3 (inverted) signal is enabled for Gate 1
0	The Data 3 (inverted) signal is disabled for Gate 1

Bit 3 – G1D2T Gate 1 Data 2 True Enable bit

Value	Description
1	The Data 2 (noninverted) signal is enabled for Gate 1
0	The Data 2 (noninverted) signal is disabled for Gate 1

Bit 2 – G1D2N Gate 1 Data 2 Negated Enable bit

Value	Description
1	The Data 2 (inverted) signal is enabled for Gate 1
0	The Data 2 (inverted) signal is disabled for Gate 1

Bit 1 – G1D1T Gate 1 Data 1 True Enable bit

Value	Description
1	The Data 1 (noninverted) signal is enabled for Gate 1
0	The Data 1 (noninverted) signal is disabled for Gate 1

Bit 0 – G1D1N Gate 1 Data 1 Negated Enable bit

Value	Description
1	The Data 1 (inverted) signal is enabled for Gate 1
0	The Data 1 (inverted) signal is disabled for Gate 1

25.4 Operation

The CLCx Control register (CLCxCON) is used to enable the module and interrupts, control the output enable bit, select output polarity and select the logic function. The CLCx Control register also allows the user to control the logic polarity of not only the cell output but also some intermediate variables.

The CLCx Input MUX Select register (CLCxSEL) allows the user to select one out of eight input signals for each of the four data selection multiplexers, pictured inside the dotted line in Figure 25-2. The output of each of the four data selection multiplexers is connected to the inputs of the logic function selected by the MODE[2:0] bits (CLCxCON[2:0]), see Figure 25-3.

The CLCx Source Enable register (CLCxGLS) allows the user to create any four variable boolean expressions from the four input data sources configured by CLCxSEL. Both the true and complementary values for each of the four signals, chosen by the CLCx Input MUX Select register (CLCxSEL), are available to the sum-of-products circuit pictured in the data gate in Figure 25-2.

25.4.1 Reset

When the ON bit is written to '0', the output of all state logic functions will be reset to '0'. A system Reset returns the CLCxCON, CLCxSEL and CLCxGLS registers to the default state and disables the module.

Asserting a device reset returns all bits in the module registers to the default state. The output of all logic functions is '0' after a reset; this includes both latch and flip-flop functions. When a device reset is asserted, ON (CLCxCON[15]) = 0, the state logic is reset and the output of the logic function is forced low.

25.4.2 CLC Setup

CLCxCON selects the logic function and determines and controls the I/O pins. This register also controls output signal polarity. The ON bit (CLCxCON[15]) must be set for the CLC to operate. All registers can be programmed while ON is clear.

The CLCxSEL (25.3.2. CLCxSEL) register controls which input signals are routed to the input bus of Figure 25-2. Both the True (T) and Negated (N) values are made available in the data bus.

The CLCxGLS (25.3.3. CLCxGLS) register selects which signals from the data bus are applied to the input OR gates. True and Negated inputs are separately enabled; do not enable both for the same signal.

The final polarity of the CLC module output is controlled by the LCPOL bit (CLCxCON[5]). The output is inverted when LCPOL = 1 and uninverted when LCPOL = 0. The GxPOL bits (CLCxCON[19:16]) control the polarity of the logic function inputs.

The INTP and INTN bits (CLCxCON[11:10]) enable interrupts on the rising and falling edge of the CLC output.

The LCOUT bit is read-only and reflects the status of the logic cell output. To output the CLCxOUT signal to an I/O pin, set the LCOE bit and configure the I/O as a digital output. The CLCxOUT signal is made available through Peripheral Pin Select (PPS) and will need to be configured.

25.4.3 Input Providers

Each logic cell in the CLC takes four inputs, one from each of the four data gates. Each data gate is connected to eight input sources. The data gate allows the selection between the inverted or noninverted polarity of each input source. Refer to 25.1. Device-Specific Information for input sources available for use with the CLC.

25.4.3.1 Source Multiplexers

The module has four input source multiplexers. Multiplexer inputs are selected by setting control bits in the CLCxSEL register to define the data source selected through each of the four data selection multiplexers. Each of the four data selection multiplexers feeds one of the four logic function input gates, as shown in Figure 25-3. The module has an internal data bus created from the output of each input source multiplexer (see Figure 25-2). The data bus has both True (T) and Negated (N) versions of each selected input source. Therefore, up to eight signals are available on the internal data bus to connect to the input gates of the logic function.

25.4.3.2 Logic Input Gates

Four logic input gates are used to route input sources from the data selection multiplexers into the four logic function inputs. The True and Negated forms of each input source signal are available for use by each logic gate. The input signal sources are enabled for use by each logic function input using the CLCxGLS register. There are up to eight signals that can be enabled for use by each logic function input. Any number of the eight signal sources may be enabled for each of the four logic function inputs. Each logic gate provides a logical OR of the input signals. The selected (True or Negated) signals are OR'd to form the gate output data. The logical NAND is obtained by changing the output polarity with the GxPOL bits. If the logical AND is required instead, select negated inputs

and invert the output polarity according to DeMorgan's theorem. If all inputs are negated and applied to a NOR, the result is identical to an AND operation. Written algebraically:

C = A AND B

is the same as:

C = NOT(NOT(A) OR NOT(B)).

Table 25-6 summarizes the basic functions that can be obtained by using the Gate Control bits. The table shows the use of all four input multiplexer sources, but the input gates can be configured to use less. If no inputs are selected (CLCxGLS = 0x00), the output will be zero or one, depending on the GxPOL bits.

Table 25-6. Example Logic Functions

CLCxGLS	GxPOL Bits	Function
0xAAAAAAAA	0	OR (D1, D2, D3, D4)
0xAAAAAAAA	1	NOR (D1, D2, D3, D4)
0x55555555	0	NAND (D1, D2, D3, D4)
0x55555555	1	AND (D1, D2, D3, D4)
0x22222222	1	NOT (D1)
0x00000000	0	Logic '0'
0x00000000	1	Logic '1'

If the output of a gate must be zero or one, the recommended method is to set all of the bits related to that gate in CLCxGLS to zero and use the Gate Polarity bit, GxPOL, to set the desired level.

25.4.3.3 Logic Function

There are eight available logic functions, including:

- AND-OR
- OR-XOR
- AND
- S-R Latch
- D Flip-Flop with Set and Reset
- D Flip-Flop with Reset
- J-K Flip-Flop with Reset
- Transparent Latch with Set and Reset

Logic functions are shown in Figure 25-3. Each logic function has four inputs and one output. The MODE[2:0] bits (CLCxCON[2:0]) set the functional behavior of the logic cell. There are four combinatorial options and four state options. Three of the state options define Input Gate 1 as a rising edge clock with the traditional meanings of D and J-K flip-flops. The fourth state option, MODE[2:0] bits = 111, is a transparent latch; Q follows D when Latch Enable (LE) is true; Q holds state when LE is false. For options with both S (Set) and R (Reset) inputs, the output changes asynchronously to the clock when S or R is a logic '1'; R is dominant.

25.4.3.4 Software Inputs

The gate data input to the logic function can be directly controlled by software by setting all of the CLCxGLS bits associated with the logic gate to '0', and writing to the appropriate GxPOL bit (see Table 25-6). The gate output will be equal to the value of the GxPOL bit.

25.4.4 Output

LCOUT (CLCxCON[6]) is the logic cell output and is routed to the I/O port pin or to other modules within the device. In all cases, the signal value is taken after the LCPOL inverter. To observe this output on an I/O pin, the user will need to set LCOE (CLCxCON[7]).

25.4.5 Application Logic

The CLC provides both combinatorial and state (see Figure 25-3) logic function options. The outputs of the input gates are applied to the logic function. If CLCxGLS = 0x00, the function receives a logic '0' when the GxPOL bits (CLCxCON[19:16]) are clear or a logic '1' when the GxPOL bits are set.

25.4.5.1 Combinatorial Logic

The combinatorial functions (MODE[2:0] = 010, 001, 000) build on the AND/OR logic of the input gate. The four-input AND can provide an OR function by inverting the inputs and outputs using DeMorgan's theorem. Inverting the output of the XOR is the same as inverting one input (but not both).

The SR function (MODE[2:0] = 011) is not affected when ON (CLCxCON[15]) is cleared, as is the case with the State Logic register. The latch is Reset-dominant, meaning that the Reset signal takes precedent over any Set signal that may be present.

25.4.5.2 State Logic

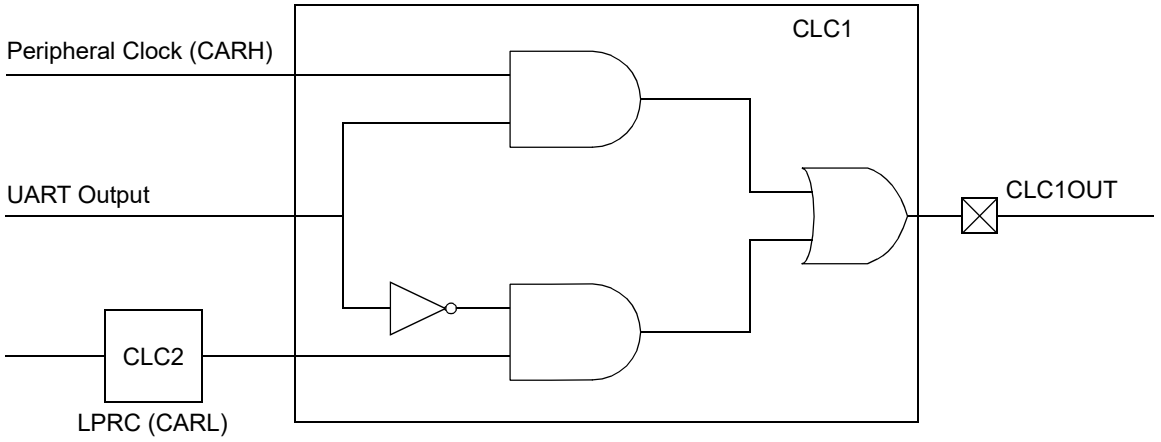
The state functions include both D and J-K flip-flops with asynchronous Set (S) and Reset (R). Input Gate 1 provides a rising edge clock. If a falling edge clock is required, Gate 1 can be inverted in the gate logic (G1POL). Input Gate 2, and sometimes also Gate 4, provide data to the register or latch input(s). When operating in Transparent Latch mode (MODE[2:0] = 111), the output, Q, follows D while LE is high and holds state while LE is low.

The various modes may or may not share state memory and switching modes may or may not change the state of the state variable. For all modes, the register is Reset-dominant.

25.5 CLC Application Example

Figure 25-4 depicts the configuration of CLC to generate Frequency Shift Key (FSK) modulation on a UART signal using CLC.

Figure 25-4. CLC Configuration for FSK Generation



Note: Inverter operation shown in the figure is done using the CLC register bits.

The peripherals required for this application are:

- CLC1 and CLC2
- UART data as modulator signal
- Peripheral clock to modulate logic '1'
- LPRC to modulate logic '0'

Note: UART, system clock and LPRC are used as examples for the modulator signal and carrier signal, respectively. However, it is possible to choose other sources as modulator and carrier signals.

The following is the application code for FSK modulation of the UART data.

Example 25-1. FSK Modulation of UART Data

```
/*Select input source for CLC1*/
CLC1SELbits.DS1 = 1;   //Peripheral clock as input source to modulate high
signal
CLC1SELbits.DS2 = 3;   //UART TX data as input source
CLC1SELbits.DS4 = 1;   //CLC2 output as input source

 //Configure Gates of CLC1
CLC1GLSbits.G2D2T = 1; // Gate 2 selects Data 2 (here UART TX data)
CLC1GLSbits.G1D1T = 1; // Gate 1 selects Data 1(here peripheral clock)
CLC1GLSbits.G3D2N = 1; // Gate 3 selects inverted Data 2 (here UART TX data)
CLC1GLSbits.G4D4T = 1; // Gate 4 selects Data 4 (here CLC2 output)

/*Select input source for CLC2*/
CLC2SELbits.DS1 = 3;   //LPRC as input source to modulate logic low signal

 //Configure Gates of CLC2
CLC2GLSbits.G1D1T = 1; // All gates select data 1 as input (here LPRC)
CLC2GLSbits.G2D1T = 1;
CLC2GLSbits.G3D1T = 1;
CLC2GLSbits.G4D1T = 1;

 //Configure CLC1 mode and output
CLC1CONbits.LCOE = 1;  // Enable CLC output on IO
CLC1CONbits.MODE = 0;  // Select AND-OR logic function

 //Configure CLC2 mode and output
CLC2CONbits.LCOE = 1;  // Enable CLC output on IO
CLC2CONbits.MODE = 2;  // Select AND logic function


/* UART configuration goes here*/

 //Turn on CLCs
CLC1CONbits.ON = 1;  // Enable CLC1
CLC2CONbits.ON = 1;  // Enable CLC2

/*Turn on UART here*/
```

25.6 CLC Interrupts

The CLC module has two types of interrupts that can be enabled: rising edge interrupt events and falling edge interrupt events. These events are enabled by the INTP (CLCxCON[11]) and INTN (CLCxCON[10]) control bits, respectively.

A valid occurrence of either interrupt will set the CLCx Interrupt Flag, CLCxIF. This will occur when the module is enabled (ON = 1) and either a rising edge output occurs when INTP = 1 or a falling edge event occurs when INTN = 1.

If the initial output state of the CLC logic is '1' and INTP = 1, an interrupt will be generated when ON is set to '1'. Likewise, an interrupt will be generated if the initial output state of the CLC is '0' and INTN = 1. These conditions must be detected and cleared in software. Similarly, a false interrupt could be generated if INTP or INTN is set while the CLC module is enabled.

The user should be sure to clear any spurious interrupt events that may occur in the initialization process of the CLC module.

If the CLCx Interrupt Enable bit, CLCxIE, is cleared, an interrupt will not be generated. However, the CLCxIF bit will still be set if an interrupt condition occurs. The user can clear the interrupt in the Interrupt Service Routine (ISR) by clearing CLCxIF. See the **10. Interrupt Controller** section of the data sheet for more information.

25.7 Operation in Power-Saving Modes

25.7.1 Sleep Mode

The CLC module is not affected by Sleep mode since it does not rely on system clock sources for operation. However, some input sources might be disabled during Sleep, so the function could be disrupted. If the source continues to operate, so will the module. Refer to the source peripheral's section for more information on its operation in Sleep mode.

25.7.2 Idle Mode

The CLC module is not affected by Idle mode since it does not rely on system clock sources for operation. However, some input sources might be disabled during Idle and the function could be disrupted. If the sources continues to operate, so will the module. Refer to the source peripheral's section for more information on its operation in Idle mode.

26. Peripheral Trigger Generator (PTG)

The dsPIC33AK128MC106 family Peripheral Trigger Generator (PTG) module is a user-programmable sequencer that is capable of generating complex trigger signal sequences to coordinate the operation of other peripherals. The PTG module is designed to interface with other modules, such as an Analog-to-Digital Converter (ADC), output compare and PWM modules, timers and interrupt controllers.

The PTG consists of the following key features:

- Behavior is Step Command Driven:
  - Step commands are eight bits wide
- Commands are Stored in a Step Queue:
  - Queue depth is up to 32 entries
  - Programmable Step execution time (Step delay)
- Supports the Command Sequence Loop:
  - Can be nested one-level deep
  - Conditional or unconditional loop
  - Two 16-bit loop counters
- Up to 16 Hardware Input Triggers:
  - Sensitive to either positive or negative edges, or a high or low level
- One Software Input Trigger
- Generates up to 32 Unique Output Trigger Signals
- Generates Two Types of Trigger Outputs:
  - Individual
  - Broadcast
- Strobed Output Port for Literal Data Values:
  - 5-bit literal write (literal part of a command)
  - 16-bit literal write (literal held in the PTGL0 register)
- Generates up to Ten Unique Interrupt Signals
- Two 16-Bit General Purpose Timers
- Flexible Self-Contained Watchdog Timer (WDT) to Set an Upper Limit to Trigger Wait Time
- Single-Step Command Capability in Debug mode
- Configurable Clock from Dedicated Clock Generator module
- Programmable Clock Divider

26.1 Device-Specific Information

Table 26-1. PTG Summary Table

PTG Module Instances	Input Trigger Sources from other Peripheral Modules	Output Triggers to other Peripheral Modules	Peripheral Bus Speed	Clock Source
1	10	5	Slow	CLKGEN10

Table 26-2. PTG Input Descriptions

PTG Input Number	PTG Input Description
PTG Trigger Input 0	Trigger Input from PWM1 ADC Trigger 2
PTG Trigger Input 1	Trigger Input from PWM2 ADC Trigger 2
PTG Trigger Input 2	Trigger Input from PWM3 ADC Trigger 2
PTG Trigger Input 3	Trigger Input from PWM4 ADC Trigger 2
PTG Trigger Input 4	Reserved
PTG Trigger Input 5	Reserved
PTG Trigger Input 6	Reserved
PTG Trigger Input 7	Trigger Input from SCCP4
PTG Trigger Input 8	Reserved
PTG Trigger Input 9	Trigger Input from Comparator 1
PTG Trigger Input 10	Trigger Input from Comparator 2
PTG Trigger Input 11	Trigger Input from Comparator 3
PTG Trigger Input 12	Trigger Input from CLC1
PTG Trigger Input 13	Trigger Input from ADC1 Ready Signal
PTG Trigger Input 14	Reserved
PTG Trigger Input 15	Trigger Input from INT2 PPS

Table 26-3. PTG Output Descriptions

PTG Output Number	PTG Output Description
PTGO0 to PTGO11	Reserved
PTGO12	ADC TRGSRC[30]
PTGO13 to PTGO23	Reserved
PTGO24	PPS Output 55
PTGO25	PPS Output 56
PTGO26	PPS Input 167
PTGO27	PPS Input 168
PTGO28 to PTGO31	Reserved

Table 26-4. PTG Trigger Interrupts

PTG Trigger Interrupts
_PTG0Interrupt
_PTG1Interrupt
_PTG2Interrupt
_PTG3Interrupt

Note: PTG strobe is not supported on the dsPIC33AK128MC106 device family.

26.2 Architectural Overview

The PTG module is a user-programmable sequencer for generating complex peripheral trigger sequences. The PTG module provides the ability to schedule complex peripheral operations, which would be difficult or impossible to achieve via a software solution.

The user writes 8-bit commands, called step commands, to the PTG Queue registers (PTGQUE0-PTGQUE7). Each 8-bit step command is made up of a command code and an option field. Table 26-5 shows the format and encoding of a step command. Based on the commands, the PTG can interact with other peripherals, such as the PWM, ADC, SCCP/MCCP and Peripheral Pin Select. See the device-specific data sheet for availability of peripherals.

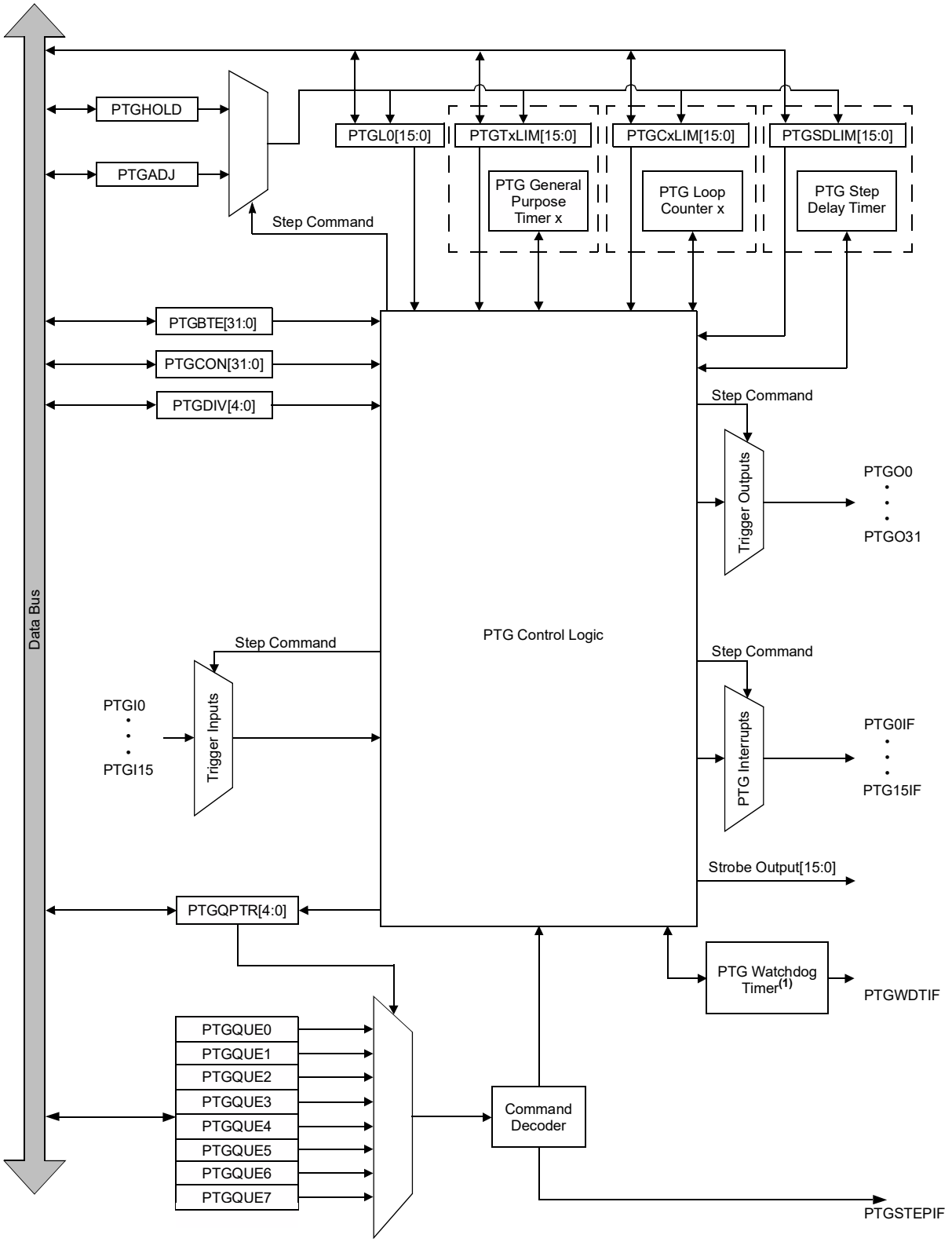
26.2.1 Step Commands and Format

The PTG operates using eleven 8-bit step commands to perform higher level tasks. There are four types of step commands:

- Input Event Control
- Control Functions
- Flow Control
- Output Generation

Combinations and sequences of these commands can be used to make decisions and take action without CPU intervention.

Figure 26-1. PTG Block Diagram



Note:

1. This is a dedicated Watchdog Timer for the PTG module and is independent of the device Watchdog Timer.

26.2.1.1 Input Event Control

There are two input event control commands, PTGWHI and PTGWLO, that wait for a high or low edge on one of the 16 PTGIx inputs. These commands are used in conjunction with a 4-bit OPTION field that specifies which PTGIx is used. Once the specified input transitions in the intended direction, the step queue is incremented and the next step command is evaluated. See 26.4.6.2. Wait for Trigger Input for additional information.

26.2.1.2 Control Functions

There are three commands for control functions, PTGCTRL, PTGADD and PTGCOPY. The PTGCTRL command controls the operation of the delay timers, software triggers and the strobe output. The PTGADD command is used to add the contents of the PTGHOLD register to other PTG registers, including the counters, timers, Step Delay and Literal register. The PTGCOPY command is used to copy the contents of the PTGHOLD register to other PTG registers, similar to the PTGADD command.

26.2.1.3 Flow Control

Flow control is accomplished using the three jump commands, PTGJMP, PTGJMPC0 and PTGJMPC1. These jump commands are three bits to allow a larger 5-bit OPTION to match that of the Queue Pointer, PTGQPTR. The PTGJMP command simply jumps to the specified queue location, whereas the PTGJMPCx commands are conditional jumps based on the comparison of the counters to the PTG Counter Limit registers (PTGCxLIM).

26.2.1.4 Output Generation

Output generation is achieved using the PTGTRIG, PTGIRQ and PTGSTRB commands. PTGTRIG is used to select and generate an output trigger (PTGOx). PTGTRIG also uses a 5-bit OPTION field to support the 32 PTGOs' selections. The PTGIRQ command is used to generate an interrupt request with the OPTION field specifying the interrupt (PTGxIF). See 26.4.8. PTG Module Outputs for additional information on triggers and interrupts. The PTGSTRB command is used to generate a strobe output, which outputs 16 bits of data to another peripheral, if implemented. See 26.4.8.3. Strobe Output for additional information.

Table 26-5 provides an overview of the PTG commands and Table 26-6 elaborates on the options available for each command. Example 26-1, Example 26-2 and Example 26-3 provide C code examples for command definitions and their options. Later examples in this FRM refer back to these examples.

Table 26-5. PTG Step Command Format and Description

Step Command Byte			
STEPx[7:0]			
CMD[3:0]		OPTION[3:0]	
bit 7	bit 4	bit 3	bit 0

Bits	Step Command	CMD[3:0]	Command Description
bits 7:4	PTGCTRL(1)	0000	Execute the control command as described by the OPTION[3:0] bits.
	PTGADD(1)	0001	Add contents of the PTGADJ register to the target register as described by the OPTION[3:0] bits.
	PTGCOPY(1)		Copy contents of the PTGHOLD register to the target register as described by the OPTION[3:0] bits.
	PTGSTRB	001x	Copy the values contained in the bits, CMD[0]:OPTION[3:0], to the strobe output bits[4:0].
	PTGWHI(2)	0100	Wait for a low-to-high edge input from a selected PTG trigger input as described by the OPTION[3:0] bits.
	PTGWLO(2)	0101	Wait for a high-to-low edge input from a selected PTG trigger input as described by the OPTION[3:0] bits.
	—	0110	Reserved; do not use.(1)
	PTGIRQ(1)	0111	Generate an individual interrupt request as described by the OPTION[3:0] bits.
	PTGTRIG(1)	100x	Generate an individual trigger output as described by the 5-bit field of CMD[0]:OPTION[3:0].
	PTGJMP	101x	Copy the values contained in the bits, CMD[0]:OPTION[3:0], to the PTGQPTR register and jump to that position in the step queue.
	PTGJMPC0	110x	PTGC0 = PTGC0LIM: Increment the PTGQPTR register.
			PTGC0 ≠ PTGC0LIM: Increment Counter 0 (PTGC0) and copy the values contained in the bits, CMD[0]:OPTION[3:0], to the PTGQPTR register and jump to that position in the step queue.
	PTGJMPC1	111x	PTGC1 = PTGC1LIM: Increment the PTGQPTR register.
			PTGC1 ≠ PTGC1LIM: Increment Counter 1 (PTGC1) and copy the values contained in the bits, CMD[0]:OPTION[3:0], to the PTGQPTR register and jump to that position in the step queue.

Notes:

1. Reserved commands or options will execute, but they do not have any effect (i.e., they execute as a NOP instruction).
2. Reserved input trigger options must not be used with PTGWHI/PTGWLO.

Example 26-1. PTG Command Definitions

```
/* PTG Commands */
#define PTGCTRL(x)      ((0b00000000)     | ((x) & 0b00001111))
#define PTGADD(x)       ((0b00010000)     | ((x) & 0b00000111))
#define PTGCOPY(x)      ((0b00011000)     | ((x) & 0b00000111))
#define PTGSTRB(x)      ((0b00100000)     | ((x) & 0b00011111))
#define PTGWHI(x)       ((0b01000000)     | ((x) & 0b00001111))
#define PTGWLO(x)       ((0b01010000)     | ((x) & 0b00001111))
#define PTGIRQ(x)       ((0b01110000)     | ((x) & 0b00001111))
#define PTGTRIG(x)      ((0b10000000)     | ((x) & 0b00011111))
#define PTGJMP(x)       ((0b10100000)     | ((x) & 0b00011111))
```

```
#define PTGJMPC0(x)     ((0b11000000)     | ((x) & 0b00011111))
#define PTGJMPC1(x)     ((0b11100000)     | ((x) & 0b00011111))
```

Table 26-6. PTG Command Options

Bits	Step Command	OPTION[3:0]	Command Description
bits 3:0	PTGCTRL(1)	0000	NOP
		0001	Reserved; do not use.
		0010	Disable Step Delay Timer (PTGSD).
		0011	Reserved; do not use.
		0100	Reserved; do not use.
		0101	Reserved; do not use.
		0110	Enable Step Delay Timer (PTGSD).
		0111	Reserved; do not use.
		1000	Start PTGT0 and wait for its value to match the PTGT0LIM register.
		1001	Start PTGT1 and wait for its value to match the PTGT1LIM register.
		1010	Wait for the software trigger (level, PTGSWT = 1).
		1011	Wait for the software trigger (positive edge, PTGSWT = 0 to 1).
		1100	Copy the PTGC0LIM register contents to the strobe output.
		1101	Copy the PTGC1LIM register contents to the strobe output.
		1110	Copy the PTGL0 register contents to the strobe output.
		1111	Generate the triggers indicated in the PTGBTE register.
	PTGADD(1)	0000	Add the PTGADJ register contents to the PTGC0LIM register.
		0001	Add the PTGADJ register contents to the PTGC1LIM register.
		0010	Add the PTGADJ register contents to the PTGT0LIM register.
		0011	Add the PTGADJ register contents to the PTGT1LIM register.
		0100	Add the PTGADJ register contents to the PTGSDLIM register.
		0101	Add the PTGADJ register contents to the PTGL0 register.
		0110	Reserved; do not use.
		0111	Reserved; do not use.

Notes:

1. All reserved options for this command will execute, but they do not have any effect (i.e., they execute as a NOP instruction).
2. Reserved input trigger options must not be used with PTGWHI/PTGWLO.

Bits	Step Command	OPTION[3:0]	Command Description
PTG Command Options (Continued)			

.....continued

Bits	Step Command	OPTION[3:0]	Command Description
bits 3:0	PTGCOPY(1)	1000	Copy the PTGHOLD register contents to the PTGC0LIM register.
		1001	Copy the PTGHOLD register contents to the PTGC1LIM register.
		1010	Copy the PTGHOLD register contents to the PTGT0LIM register.
		1011	Copy the PTGHOLD register contents to the PTGT1LIM register.
		1100	Copy the PTGHOLD register contents to the PTGSDLIM register.
		1101	Copy the PTGHOLD register contents to the PTGL0 register.
		1110	Reserved; do not use.
		1111	Reserved; do not use.
	PTGWHI(2) or PTGWLO(2)	0000	PTGI0 (see Table 26-2).
		...	...
		1111	PTGI15 (see Table 26-3).
	PTGIRQ(1)	0000	Generate PTG Interrupt 0
		...	...
		0011	Generate PTG Interrupt 4
	PTGTRIG	0000	PTGO0 (see Table 26-3).
		0001	PTGO1
		...	...
		1110	PTGO30
		1111	PTGO31 (see Table 26-3).

Notes:

1. All reserved options for this command will execute, but they do not have any effect (i.e., they execute as a NOP instruction).
2. Reserved input trigger options must not be used with PTGWHI/PTGWLO.

Example 26-2. `PTGCTRL` Options

```
// Used with PTGCTRL command
typedef    enum
{
           stepDelayDisable = 0b0010,
           ptgNop = 0b0000;
           stepDelayEnable = 0b0110,
           t0Wait = 0b1000,
           t1Wait = 0b1001,
           softTriggerLevelWait = 0b1010,
           softTriggerEdgeWait = 0b1011,
           c0Strobe = 0b1100,
           c1Strobe = 0b1101,
           l0Strobe = 0b1110,
           triggerGenerate = 0b1111,
} CTRL_T;
```

Example 26-3. Options for `PTGADD` and `PTGCOPY` Commands

```
// Used with PTGADD and PTGCOPY commands
typedef    enum
{
           c0Limit = 0b0000,
           c1Limit = 0b0001,
           t0Limit = 0b0010,
           t1Limit = 0b0011,
           stepDelay = 0b0100,
           literal0 = 0b0101,
} ADD_COPY_T;
```

26.3 PTG Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3500	PTGCON	31:24	Reserved[2:0]					PTGDIV[4:0]		
		23:16	PTGPWD[3:0]					PTGWDT[2:0]		
		15:8	ON	Reserved	SIDL	PTGTOGL		PTGSWT	PTGSSEN	PTGIVIS
		7:0	PTGSTRT	PTGWDTO	PTGBUSY				PTGITM[1:0]	
0x3504	PTGBTE	31:24	PTGBTE[31:24]							
		23:16	PTGBTE[23:16]							
		15:8	PTGBTE[15:8]							
		7:0	PTGBTE[7:0]							
0x3508	PTGHOLD	31:24								
		23:16								
		15:8	PTGHOLD[15:8]							
		7:0	PTGHOLD[7:0]							
0x350C	PTGT0LIM	31:24								
		23:16								
		15:8	PTGT0LIM[15:8]							
		7:0	PTGT0LIM[7:0]							
0x3510	PTGT1LIM	31:24								
		23:16								
		15:8	PTGT1LIM[15:8]							
		7:0	PTGT1LIM[7:0]							
0x3514	PTGSDLIM	31:24								
		23:16								
		15:8	PTGSDLIM[15:8]							
		7:0	PTGSDLIM[7:0]							
0x3518	PTGC0LIM	31:24								
		23:16								
		15:8	PTGC0LIM[15:8]							
		7:0	PTGC0LIM[7:0]							
0x351C	PTGC1LIM	31:24								
		23:16								
		15:8	PTGC1LIM[15:8]							
		7:0	PTGC1LIM[7:0]							
0x3520	PTGADJ	31:24								
		23:16								
		15:8	PTGADJ[15:8]							
		7:0	PTGADJ[7:0]							
0x3524	PTGL0	31:24								
		23:16								
		15:8	PTGL0[15:8]							
		7:0	PTGL0[7:0]							
0x3528	PTGQPTR	31:24								
		23:16								
		15:8								
		7:0				PTGQPTR[4:0]				
0x352C ... 0x352F	Reserved									
0x3530	PTGQUE0	31:24	STEP(4(0) + 3)[7:0]							
		23:16	STEP(4(0) + 2)[7:0]							
		15:8	STEP(4(0) + 1)[7:0]							
		7:0	STEP(4(0))[7:0]							
0x3534	PTGQUE1	31:24	STEP(4(1) + 3)[7:0]							
		23:16	STEP(4(1) + 2)[7:0]							
		15:8	STEP(4(1) + 1)[7:0]							
		7:0	STEP(4(1))[7:0]							
0x3538	PTGQUE2	31:24	STEP(4(2) + 3)[7:0]							
		23:16	STEP(4(2) + 2)[7:0]							
		15:8	STEP(4(2) + 1)[7:0]							
		7:0	STEP(4(2))[7:0]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x353C	PTGQUE3	31:24	STEP(4(3) + 3)[7:0]							
		23:16	STEP(4(3) + 2)[7:0]							
		15:8	STEP(4(3) + 1)[7:0]							
		7:0	STEP(4(3))[7:0]							
0x3540	PTGQUE4	31:24	STEP(4(4) + 3)[7:0]							
		23:16	STEP(4(4) + 2)[7:0]							
		15:8	STEP(4(4) + 1)[7:0]							
		7:0	STEP(4(4))[7:0]							
0x3544	PTGQUE5	31:24	STEP(4(5) + 3)[7:0]							
		23:16	STEP(4(5) + 2)[7:0]							
		15:8	STEP(4(5) + 1)[7:0]							
		7:0	STEP(4(5))[7:0]							
0x3548	PTGQUE6	31:24	STEP(4(6) + 3)[7:0]							
		23:16	STEP(4(6) + 2)[7:0]							
		15:8	STEP(4(6) + 1)[7:0]							
		7:0	STEP(4(6))[7:0]							
0x354C	PTGQUE7	31:24	STEP(4(7) + 3)[7:0]							
		23:16	STEP(4(7) + 2)[7:0]							
		15:8	STEP(4(7) + 1)[7:0]							
		7:0	STEP(4(7))[7:0]							

26.3.1 PTG Control Register

Name: PTGCON
Offset: 0x3500

Legend: R = Readable bit, W = Writable bit, HC = Hardware Clearable bit, HS = Hardware Settable bit, r = Reserved bit

Notes:

1. These bits are read-only when the module is executing step commands.
2. This bit is only used with the `PTGCTRL 0b1010/0b1011` step command software trigger options. Refer to 26.4.6.3. Wait for Software Trigger for more information.
3. The PTGSSEN bit may only be written during a debugging session. See 26.4.10. Single-Step Mode for more information.
4. These bits apply to the `PTGWHI` and `PTGWLO` commands only.

Bit	31	30	29	28	27	26	25	24
	Reserved[2:0]			PTGDIV[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PTGPWD[3:0]					PTGWDT[2:0]		
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bit	15	14	13	12	11	10	9	8
	ON	Reserved	SIDL	PTGTOGL		PTGSWT	PTGSSEN	PTGIVIS
Access	R/W	r	R/W	R/W		R/W/HC	R/W	R/W
Reset	0	0	0	0		0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGSTRT	PTGWDTO	PTGBUSY				PTGITM[1:0]	
Access	R/W/HC	R/W/HS	R/HS/HC				R/W	R/W
Reset	0	0	0				0	0

Bits 31:29 – Reserved[2:0] Reserved, maintain as 0.

Bits 28:24 – PTGDIV[4:0] PTG Module Clock Prescaler (Divider) bits(1)

Value	Description
11111	Divide by 32
11110	Divide by 31
. . .	
00001	Divide by 2
00000	Divide by 1

Bits 23:20 – PTGPWD[3:0] PTG Trigger Output Pulse-Width (in PTG clock cycles) bits(1)

Value	Description
1111	All trigger outputs are 16 PTG clock cycles wide
1110	All trigger outputs are 15 PTG clock cycles wide
. . .	
0001	All trigger outputs are 2 PTG clock cycles wide

Value	Description
0000	All trigger outputs are 1 PTG clock cycle wide

Bits 18:16 – PTGWDT[2:0] PTG Watchdog Timer Time-out Selection bits(1)

Value	Description
111	Watchdog Timer will time-out after 512 PTG clocks
110	Watchdog Timer will time-out after 256 PTG clocks
101	Watchdog Timer will time-out after 128 PTG clocks
100	Watchdog Timer will time-out after 64 PTG clocks
011	Watchdog Timer will time-out after 32 PTG clocks
010	Watchdog Timer will time-out after 16 PTG clocks
001	Watchdog Timer will time-out after 8 PTG clocks
000	Watchdog Timer is disabled

Bit 15 – ON PTG Enable bit

Value	Description
1	PTG is enabled
0	PTG is disabled

Bit 14 – Reserved Must be written as '0'

Bit 13 – SIDL PTG Stop in Idle Mode bit

Value	Description
1	Halts PTG operation when device is Idle
0	PTG operation continues when device is Idle

Bit 12 – PTGTOGL PTG Toggle Trigger Output bit(1)

Value	Description
1	Toggles state of TRIG output for each execution of PTGTRIG
0	Generates a single TRIG pulse for each execution of PTGTRIG

Bit 10 – PTGSWT PTG Software Trigger bit(2)

Value	Description
1	Asserts the PTG software trigger
0	Deasserts the PTG software trigger (Level-Sensitive mode)/cleared by hardware (Edge-Sensitive mode)

Bit 9 – PTGSSEN PTG Single-Step Enable bit(3)

If in Debug mode:
1 = Enables Single-Step mode
0 = Disables Single-Step mode
If not in Debug mode:
Writes have no effect; read as '0'.

Bit 8 – PTGIVIS PTG Internal Counter/Timer Visibility bit(1,4)

Value	Description
1	Reading the PTGSDLIM, PTGCxLIM or PTGTxLIM registers returns the current values of their corresponding internal Counter/Timer registers (PTGSD, PTGCx and PTGTx)
0	Reading the PTGSDLIM, PTGCxLIM or PTGTxLIM registers returns the value of these Limit registers

Bit 7 – PTGSTRT PTG Start Sequencer bit(3)

If not in Single-Step mode:
1 = Starts to sequentially execute the commands

0 = Stops executing the commands
If in Single-Step mode:
1 = Executes the next step command, then halts the sequencer
0 = Manually halts the sequencer/execution of signal command has completed (cleared by hardware)

Bit 6 – PTGWDTO PTG Watchdog Timer Time-out Status bit(1)

Value	Description
1	PTG Watchdog Timer has timed out
0	PTG Watchdog Timer has not timed out

Bit 5 – PTGBUSY PTG State Machine Busy bit

Value	Description
1	PTG is running on the selected clock source
0	PTG state machine is not running

Bits 1:0 – PTGITM[1:0] PTG Input Trigger Operation Selection bits(1,4)

Value	Description
11	Single-level detect with step delay not executed on exit of command, regardless of the PTGCTRL command (Mode 3)
10	Single-level detect with step delay executed on exit of command (Mode 2)
01	Continuous edge detect with step delay not executed on exit of command, regardless of the PTGCTRL command (Mode 1)
00	Continuous edge detect with step delay executed on exit of command (Mode 0)

26.3.2 PTG Broadcast Trigger Enable Register

Name: PTGBTE
Offset: 0x3504

Legend: R = Readable bit, W = Writable bit

Note:

1. These bits are read-only when the module is executing step commands.

Bit	31	30	29	28	27	26	25	24
	PTGBTE[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PTGBTE[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PTGBTE[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGBTE[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PTGBTE[31:0] PTG Broadcast Trigger Enable bits(1)

Value	Description
1	Generates PTG output trigger corresponding to bit number when the broadcast command is executed
0	Does not generate corresponding trigger when the broadcast command is executed

26.3.3 PTG Hold Register

Name: PTGHOLD
Offset: 0x3508

Legend: R = Readable bit, W = Writable bit

Note:

1. These bits are read-only when the module is executing step commands.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGHOLD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGHOLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGHOLD[15:0] PTG General Purpose Hold Register bits(1)
This register holds the user-supplied data to be copied to the PTGTxLIM, PTGCxLIM, PTGSDLIM or PTGL0 register using the `PTGCOPY` command.

26.3.4 PTG Timer0 Limit Register

Name: PTGT0LIM
Offset: 0x350C

Legend: R = Readable bit, W = Writable bit

Notes:

1. These bits are read-only when the module is executing step commands.
2. The value read from these register bits depends on the PTGIVIS bit (PTGCON[8]). Refer to 26.4.3. Control Register Access for more information.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGT0LIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGT0LIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGT0LIM[15:0] PTG Timer0 Limit Register bits(1,2)
General purpose Timer0 Limit register.

26.3.5 PTG Timer1 Limit Register

Name: PTGT1LIM
Offset: 0x3510

Legend: R = Readable bit, W = Writable bit, HC = Hardware Clearable bit, HS = Hardware Settable bit, r = Reserved bit

Notes:

1. These bits are read-only when the module is executing step commands.
2. The value read from these register bits depends on the PTGIVIS bit (PTGCON[8]). Refer to 26.4.3. Control Register Access for more information.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGT1LIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGT1LIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGT1LIM[15:0] PTG Timer1 Limit Register bits(1,2)
General purpose Timer1 Limit register.

26.3.6 PTG Step Delay Limit Register

Name: PTGSDLIM
Offset: 0x3514

Legend: R = Readable bit, W = Writable bit

Notes:

1. These bits are read-only when the module is executing step commands.
2. The value read from these register bits depends on the PTGIVIS bit (PTGCON[8]). Refer to 26.4.3. Control Register Access for more information.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGSDLIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGSDLIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGSDLIM[15:0] PTG Step Delay Limit Register bits(1,2)
This register holds a PTG step delay value representing the number of additional PTG clocks between the start of a step command and the completion of a step command.

26.3.7 PTG Counter 0 Limit Register

Name: PTGC0LIM
Offset: 0x3518

Legend: R = Readable bit, W = Writable bit

Notes:

1. These bits are read-only when the module is executing step commands.
2. The value read from these register bits depends on the PTGIVIS bit (PTGCON[8]). Refer to 26.4.3. Control Register Access for more information.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGC0LIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGC0LIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGC0LIM[15:0] PTG Counter 0 Limit Register bits(1,2)
This register is used to specify the loop count for the `PTGJMPC0` step command or as a Limit register for the General Purpose Counter 0.

26.3.8 PTG Counter 1 Limit Register

Name: PTGC1LIM
Offset: 0x351C

Legend: R = Readable bit, W = Writable bit

Notes:

1. These bits are read-only when the module is executing step commands.
2. The value read from these register bits depends on the PTGIVIS bit (PTGCON[8]). Refer to 26.4.3. Control Register Access for more information.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGC1LIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGC1LIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGC1LIM[15:0] PTG Counter 1 Limit Register bits(1,2)
This register is used to specify the loop count for the `PTGJMPC1` step command or as a Limit register for the General Purpose Counter 1.

26.3.9 PTG Adjust Register

Name: PTGADJ
Offset: 0x3520

Legend: R = Readable bit, W = Writable bit

Note:
1. These bits are read-only when the module is executing step commands.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGADJ[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGADJ[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGADJ[15:0] PTG Adjust Register bits(1)
This register holds the user-supplied data to be added to the PTGTxLIM, PTGCxLIM, PTGSDLIM or PTGL0 register using the `PTGADD` command.

26.3.10 PTG Literal 0 Register

Name: PTGL0
Offset: 0x3524

Legend: R = Readable bit, W = Writable bit

Note:

1. These bits are read-only when the module is executing step commands.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGL0[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGL0[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGL0[15:0] PTG Literal 0 Register bits(1)
This register holds a 16-bit value to be written by the strobe output using the `PTGCTRL` command.

26.3.11 PTG Step Queue Pointer Register

Name: PTGQPTR
Offset: 0x3528

Legend: R = Readable bit, W = Writable bit

Note:

1. These bits are read-only when the module is executing step commands.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				PTGQPTR[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 4:0 – PTGQPTR[4:0] PTG Step Queue Pointer Register bits(1)
This register points to the currently active step command in the step queue.

26.3.12 PTG Step Queue n Pointer Register (n = 0-7)

Name: PTGQUEn
Offset: 0x3530, 0x3534, 0x3538, 0x353C, 0x3540, 0x3544, 0x3548, 0x354C

Legend: R = Readable bit, W = Writable bit

Notes:
1. These bits are read-only when the module is executing step commands.
2. Refer to Table 26-5 for the step command encoding.

Bit	31	30	29	28	27	26	25	24
	STEP(4(n) + 3)[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	STEP(4(n) + 2)[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	STEP(4(n) + 1)[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	STEP(4(n))[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – STEP(4(n) + 3)[7:0] PTG Command Step 4n + 3 bits(1,2)
A queue location for storage of the STEP(4n + 3) command byte.

Bits 23:16 – STEP(4(n) + 2)[7:0] PTG Command Step 4n + 2 bits(1,2)
A queue location for storage of the STEP(4n + 2) command byte.

Bits 15:8 – STEP(4(n) + 1)[7:0] PTG Command Step 4n + 1 bits(1,2)
A queue location for storage of the STEP(4n + 1) command byte.

Bits 7:0 – STEP(4(n))[7:0] PTG Command Step 4n bits(1,2)
A queue location for storage of the STEP(4n) command byte.

26.4 Operation

26.4.1 Basic Operation

The user loads the step commands (8-bit values) into the PTG Queue registers. The commands define a sequence of events for generating the trigger output signals to the peripherals. The step commands can also be used to generate the interrupt requests to the processor.

The PTG module is enabled and clocked when the ON bit (PTGCON[15]) = 1. While the PTGSTRT bit (PTGCON[7]) = 0, the PTG module is in the Halt state.

Notes:

1. The control registers cannot be modified when the PTG module is in the Halt state.
2. The PTG module must be enabled (ON = 1) prior to attempting to set the PTGSTRT bit.
3. The user should not attempt to set the ON and PTGSTRT bits within the same data write cycle.

Subsequently, setting PTGSTRT = 1 will enable the module for Continuous mode execution of the step command queue. The PTG sequencer will start to read the step queue at the address held in the Queue Pointer (PTGQPTR). Each command byte is read, decoded and executed sequentially. The minimum duration of any step command is one PTG clock as explained in 26.4.2. PTG Clock Selection.

Step commands will execute sequentially until any of the following occurs:

- A PTGJMP, PTGJMPC0 or PTGJMPC1 (flow change) step command is executed.
- The user clears the PTGSTRT bit, stopping the PTG Sequencer. No further step commands are read/decoded and execution halts.
- The internal Watchdog Timer overflows, clearing the PTGSTRT bit and stopping the PTG sequencer. No further step commands are read/decoded and execution halts.
- The PTG module is disabled (ON = 0).

The step commands can also be made to wait on a condition, such as an input trigger edge, a software trigger or a timer match, before continuing execution. For more information, refer to 26.4.9. Stopping the Sequencer.

26.4.2 PTG Clock Selection

The PTG module has multiple clock options and has a selectable prescaler, which divides the PTG clock input from 1 to 32.

26.4.2.1 Clock Source Selection

The PTG clock source is determined by a dedicated device clock generator, external to the PTG. The assignment of clock generators is device-specific.

26.4.2.2 Clock Prescaler Selection

The PTGDIV[4:0] bits (PTGCON[28:24]) specify the prescaler value for the PTG clock generation logic. These bits can be written only when the PTG module is disabled (ON = 0).

Note: Any attempt to write to the PTGDIV[4:0] bits while ON = 1 will have no effect.

26.4.2.3 Module Enable Delay

Once the PTG module is enabled (ON = 1), there is a delay before the PTG starts to execute commands. This delay is expressed in Equation 26-1. The PTG clock period is the effective clock after the prescaler.

Equation 26-1. Enable Delay

T_{DLYEN} = 4 • PTG Clock Period (Maximum)

The user must ensure that no control bits are modified during the delay. Also, no external triggers may be asserted prior to the PTG state machine commencing execution; otherwise, the triggers will be missed.

26.4.3 Control Register Access

When the PTG module is enabled (ON = 1), writes are inhibited to the PTGDIV[4:0] bits in PTGCON. Other bits and control registers may be written to as long as PTGSTRT = 0. See 26.3.1. PTGCON for more information. When the PTG module is actively executing code (ON = 1 and PTGSTART = 1), only the ON, PTGSWT, PTGSIDL, and PTGSTRT bits in PTGCON are writeable. All other bits in PTGCON and other control registers are read-only.

When the PTG module is enabled (ON = 1), reads can be performed from any control register at any time; however, the data read from certain control registers depends on the PTGSTRT and PTGIVIS bits.

When the PTG module is disabled (ON = 0), all control registers can be read and written to as normal. The PTGIVIS bit has no effect when ON = 0; all Timer/Counter registers will read as their Limit register values.

26.4.3.1 Internal Control Register Visibility

For debugging purposes, some registers internal to the PTG can be read by the user during PTG operation. When PTGSTRT = 0, a read of any PTG Control register will return the value last written to it. However, when PTGSTRT is set to '1', the values stored in the PTGTxLIM, PTGCxLIM, PTGSDLIM, PTGL0 and PTGQPTR registers are transferred to internal registers, and the user-accessible registers become read-only. Subsequent reads of these registers, as long as PTGSTRT = 1, and for the Timer/Counter registers, PTGIVIS = 0, return the internal register values, which may be modified by step commands.

Note: In order to reliably read the internal registers while the PTG is running, the PTG clock source should be the same as the CPU.

26.4.3.2 Internal Timer/Counter Visibility

When PTGIVIS = 1 during PTG operation, reads of the PTGTxLIM, PTGCxLIM and PTGSDLIM registers return the values of internal counters used to implement the respective timer/counter functionality, instead of the Limit register values. This allows the user to monitor the operation of a timer/counter.

Note: In order to reliably read the internal registers while the PTG is running, the PTG clock source must be the same as the CPU instruction clock (F_{CY}).

26.4.4 Step Queue Pointer

The PTG Step Queue Pointer register (PTGQPTR) addresses the currently active step command in the step queue. While the PTG is not executing step commands (PTGSTRT = 0), any value can be written to PTGQPTR regardless of the state of the ON bit. Once the PTGSTRT bit is set, the PTG begins executing step commands at the queue location indicated by PTGQPTR, and the register becomes read-only. When the PTG is disabled, the PTGQPTR register is cleared once on transition of the ON bit from '1' to '0', after which PTGQPTR becomes writeable again.

The user can read the PTGQPTR register at any time. In the Disabled state (ON = 0) and Idle state (ON = 1 and PTGSTRT = 0), a read returns the index of the first step command to execute. In the Active state (ON = 1 and PTGSTRT = 1), a read returns the index of the currently executing step command. The PTGQPTR register is typically incremented during the first cycle of each command. The exceptions to this rule are:

- If the PTGJMP command is executed: The Step Queue Pointer is loaded with the target queue address.
- If the PTGJMPCx command is executed and PTGCx is less than PTGCxLIM: The Step Queue Pointer is loaded with the target queue address.
- If PTGQPTR points to the last step command in PTGQUEn: The Step Queue Pointer will roll over to '0'.

26.4.5 Command Looping Control

Two 16-bit loop counters are provided (PTGC0 and PTGC1) that can be used by the PTGJMPCx command as a block loop counter or delay generator.

Each loop counter consists of an incrementing counter (PTGCx) and a Limit register (PTGCxLIM). The Limit register value can be changed by writing directly to the register (when the module is disabled) or by the PTG sequencer (when the module is enabled). The data read from the Limit register

depends upon the state of the PTG Counter/Timer Visibility bit (PTGIVIS). The counters are cleared when the module is in the Reset state or when the PTG module is disabled (ON = 0).

26.4.5.1 Using the Loop Counter as a Block Loop Counter

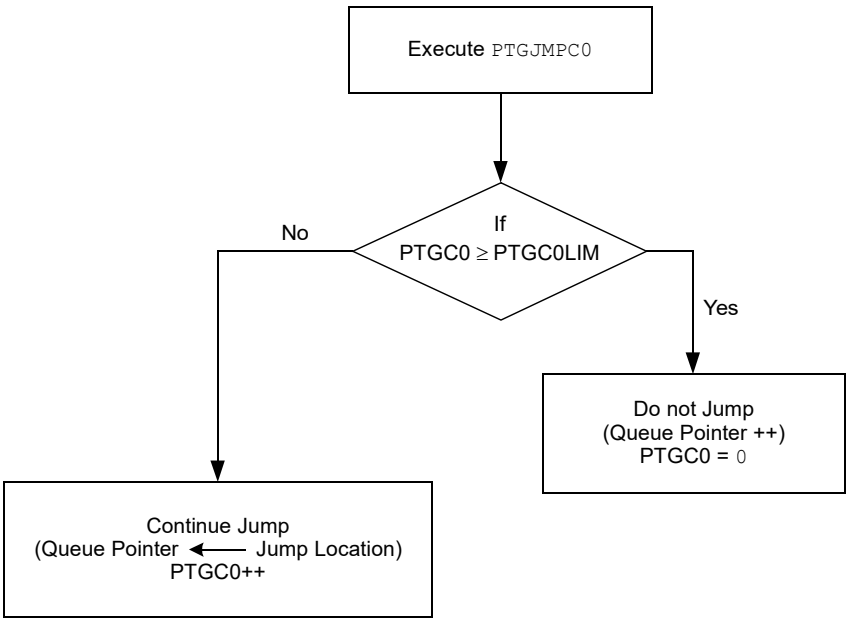
The PTGJMPCx (Jump Conditional) command uses one of the loop counters to keep track of the number of times the PTGJMPCx command is executed, and can therefore be used to create code block loops. This is useful in applications where a sequence of peripheral events needs to be repeated several times. The PTGJMPCx command allows the user to create code loops and use fewer step commands.

Each time the PTGJMPCx command is executed, the corresponding internal loop counter is compared to its limit value. If the loop counter has not reached the limit value, the jump location is loaded into the PTGQPTR register and the loop counter is incremented by one. The next command will be fetched from the new queue location. If the counter has reached the limit, the sequencer proceeds to the next command (i.e., increments the Queue Pointer). While preparing for the next PTGJMPCx command loop execution, the corresponding loop counter is cleared (see Figure 26-2).

Note: The loop counter value can be modified (via the PTGADD or PTGCOPY command) prior to execution of the first iteration of the command loop.

The provision for two separate loop counters and associated PTGJMPCx commands allows for the nested loops to be supported (one-level deep). There are no restrictions with regard to which PTGJMPCx command resides in the inner or outer loops.

Figure 26-2. Implementing Block Loop Diagram



26.4.6 Sequencer Operation

All commands are executed in a single cycle, except for the flow change commands and the commands that are waiting for an external input.

26.4.6.1 Step Command Duration

By default, each step command executes in one PTG clock period. There are several methods to slow the execution of the step commands:

- Wait for a trigger input

- Wait for a GP timer (PTGTxLIM)
- Insert a delay loop using the PTGJMP and PTGJMPCx commands
- Enable and insert a step delay (PTGSDLIM) after execution of each command

26.4.6.2 Wait for Trigger Input

The PTG module can support up to 16 independent trigger inputs. The user can specify a step command that waits for a positive or negative edge, or a high or low level, of the selected input signal to occur. The operating mode is selected by the PTGITM[1:0] bits in the PTGCON register.

The PTGWHI command looks for a positive edge or High state to occur on the selected trigger input. The PTGWLO command looks for a negative edge or Low state to occur on the selected trigger input. The PTG repeats the trigger input command (i.e., effectively waits) until the selected signal becomes valid before continuing the step command execution.

The minimum execution time to wait for a trigger is one PTG clock. There is no limit for the PTG wait for a trigger input other than that enforced by the Watchdog Timer. Refer to 26.4.7. PTG Watchdog Timer for more information.

The PTG module supports four input trigger command operating modes (Mode 0-Mode 3), which are selected by the PTGITM[1:0] bits in the PTGCON register.

Note: If the step delay is disabled, Mode 0 and Mode 1 are equivalent in operation and Mode 2 and Mode 3 are equivalent in operation.

26.4.6.2.1 Mode 0: PTGITM[1:0] = 0b00 (Continuous Edge Detect with Step Delay at Exit)

In this mode, the selected trigger input is continuously tested starting immediately after the PTGWHI or PTGWLO command is executed. When the trigger edge is detected, the command execution completes.

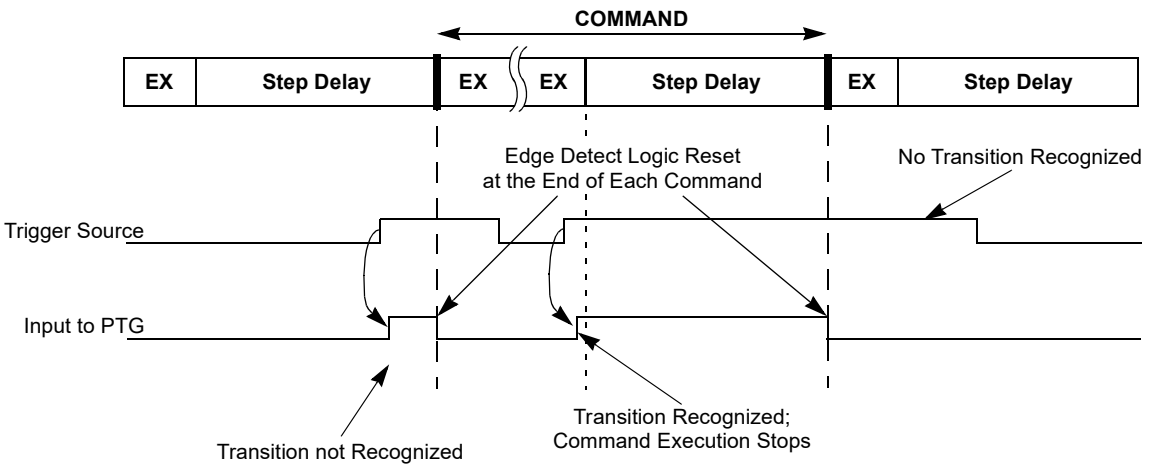
If the step delay counter is enabled, the step delay will be inserted (once) after the valid edge is detected and after the command execution.

If the step delay counter is not enabled, the command will complete after the valid edge is detected and execution of the subsequent command will commence immediately.

Note: The edge detect logic is reset after the command execution is complete (i.e., prior to any step delay associated with the command). For the edge to be detected, the edge should occur during the PTGWHI or PTGWLO command execution.

Figure 26-3 shows an example timing diagram of Mode 0 operation.

Figure 26-3. Operation of Edge-Sensitive Command with Exit Step Delay



26.4.6.2.2 Mode 1: PTGITM[1:0] = 0b01 (Continuous Edge Detect without Step Delay at Exit)

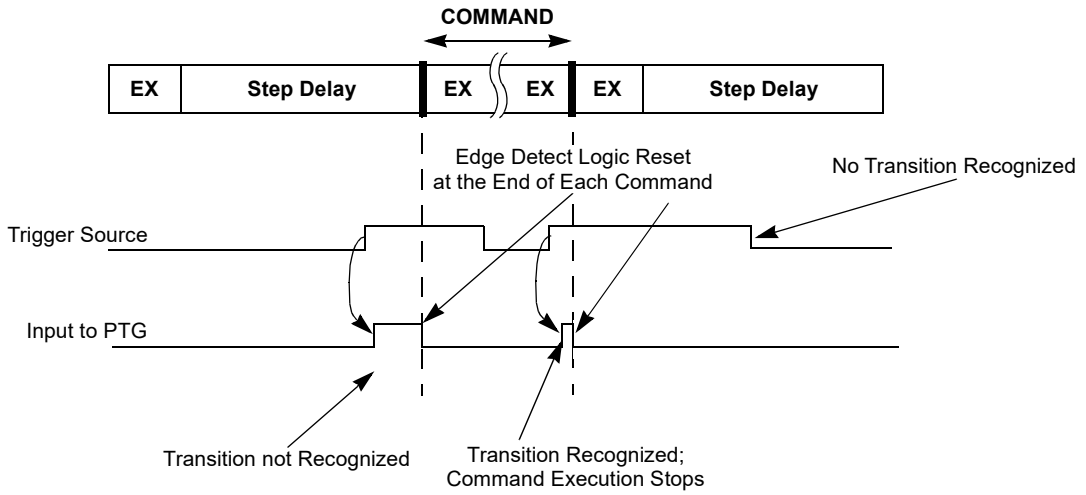
In this mode, the selected trigger input is continuously tested starting immediately after the PTGWHI or PTGWLO command is executed. When the trigger edge is detected, the command execution completes.

Regardless of whether the step delay counter is enabled or disabled, the step delay will not be inserted after the command execution has completed.

Note: The edge detect logic is reset after the command execution completes. To be detected, the edge may therefore occur during the PTGWHI or PTGWLO command execution.

Figure 26-4 shows an example timing diagram of Mode 1 operation.

Figure 26-4. Operation of Edge-Sensitive Command without Exit Step Delay



26.4.6.2.3 Mode 2: PTGITM[1:0] = 0b10 (Sampled Level Detect with Step Delay at Exit)

In this mode, the selected trigger input is sample tested for a valid level immediately after the PTGWHI or PTGWLO command is executed; the trigger input is tested (once per PTG clock).

If the trigger does not occur, and the step delay is enabled, the command waits for the step delay to expire before testing the trigger input again. When the trigger occurs, the command execution completes and the step delay is reinserted.

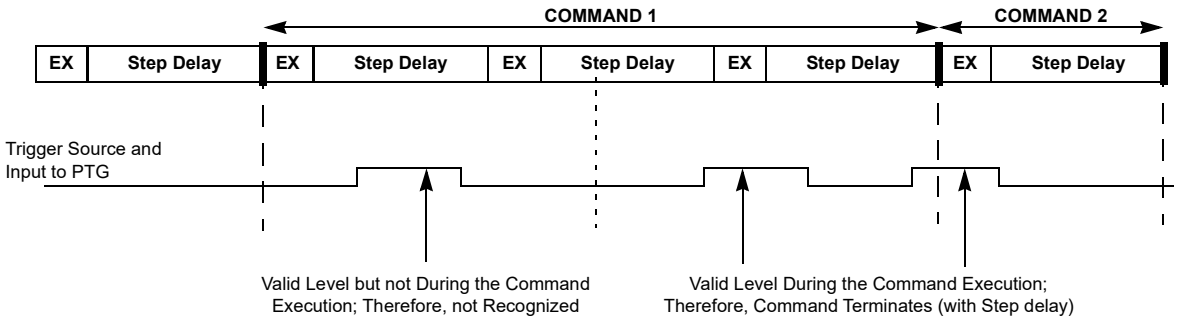
If the trigger does not occur and the step delay is disabled, the command immediately tests the trigger input again during the next PTG clock cycle. When the trigger occurs, the command execution completes and execution of the subsequent command will commence immediately.

Notes:

1. As this operating mode is level-sensitive, if the input trigger level is true at the start of execution of the PTGWHI or PTGWLO command, the input test will be instantly satisfied.
2. The input is not latched: therefore, it must be valid when the command executes in order to be recognized.

Figure 26-5 shows an example timing diagram of Mode 2 operation.

Figure 26-5. Operation of Level-Sensitive Command with Exit Step Delay



26.4.6.2.4 Mode 3: PTGITM[1:0] = 0b11 (Sampled Level Detect without Step Delay at Exit)

In this mode, the selected trigger input is sample tested for a valid level immediately after the PTGWHI or PTGWLO command is executed; the trigger input is tested (once per PTG clock).

If the trigger does not occur and the step delay is enabled, the command waits for the step delay to expire before testing the trigger input again. When the trigger is found to be true, the command execution completes and execution of the subsequent command will commence immediately. The step delay is not inserted.

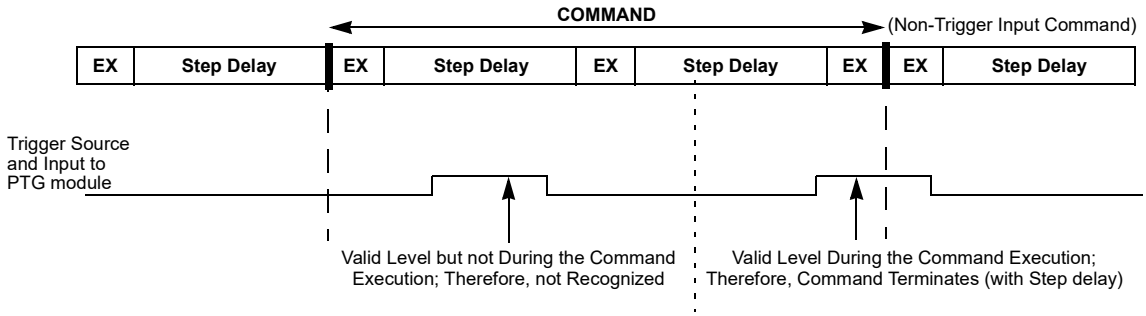
If the trigger does not occur and the step delay is disabled, the command immediately tests the trigger input again during the next PTG clock cycle. When the trigger occurs, the command execution completes and execution of the subsequent command will commence immediately.

Notes:

1. As this operating mode is level-sensitive, if the input trigger level is true at the start of execution of the PTGWHI or PTGWLO command, the input test will be instantly satisfied.
2. The input is not latched, therefore, it must be valid when the command executes in order to be recognized.

Figure 26-6 shows an example timing diagram of Mode 3 operation.

Figure 26-6. Operation of Level-Sensitive Command without Exit Step Delay



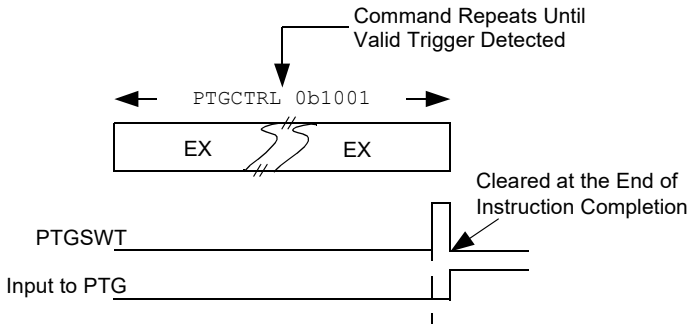
26.4.6.3 Wait for Software Trigger

The user can set either a 'PTGCTRL 0b1011' (edge-triggered) or 'PTGCTRL 0b1010' (level-triggered) command to wait for a software generated trigger. This trigger is generated by setting the PTGSWT bit (PTGCON[10]).

The 'PTGCTRL 0b1011' command is sensitive only to the PTGSWT bit transition from '0' to '1'. This transition must occur during the command execution; otherwise, the command will continue to wait. The PTGSWT bit is automatically cleared by hardware on completion of the 'PTGCTRL 0b1011' command execution, initializing the bit for the next software trigger command.

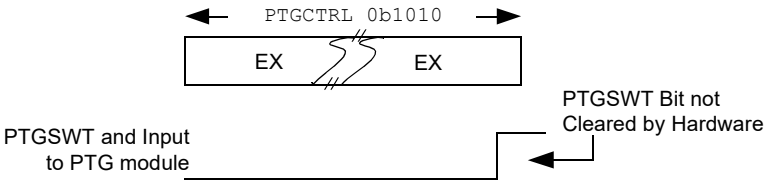
Figure 26-7 explains the operation of the wait for an edge-based software trigger.

Figure 26-7. Operation of Wait for Edge-Based Software Trigger



The 'PTGCTRL 0b1010' command is sensitive to the level of the PTGSWT bit. This command waits until PTGSWT = 1. It will complete immediately if PTGSWT = 1 upon entry to the command. The PTGSWT bit is not automatically cleared by the 'PTGCTRL 0b1010' command. If desired, the PTGSWT bit can be cleared by the user application on completion of the 'PTGCTRL 0b1010' command execution. Figure 26-8 explains the operation of the wait for the level-based software trigger.

Figure 26-8. Operation of Wait for Level-Based Software Trigger



Using the 'PTGCTRL 0b1010' or 'PTGCTRL 0b1011' step commands halts execution of further commands until the PTGSWT bit is set, allowing the user to coordinate activity between the PTG module and the application software.

26.4.6.4 Wait for GP Timer

The PTG has two internal dedicated 16-bit General Purpose (GP) timers (PTGT0 and PTGT1), which can be used by the sequencer to wait for a specified period. The step commands are available for loading, modifying or initializing the GP timers.

Each GP timer consists of an incrementing timer (PTGTx) and a Limit register (PTGTxLIM). The Limit register value can be changed by a CPU write (when the module is disabled) or by the PTG sequencer (when the module is enabled). Data read from the Limit register depend upon the state of the PTG Counter/Timer Visibility bit (PTGIVIS).

When running, the timers increment on the rising edge of the PTG clock, which is defined in the PTGCON register. The user can specify a wait operation using a GP timer by executing the appropriate 'PTGCTRL 0b1000' or 'PTGCTRL 0b1001' command (wait for selected GP timer).

When waiting for the GP timer, the command will wait until the value of the timer (Timer0 or Timer1) reaches its respective limit value (PTGT0LIM or PTGT1LIM). On reaching the limit value, the step command execution completes and the next command will start. The timer is also cleared for its next use. All timers are cleared when the device is in the Reset state or when the PTG module is disabled (ON = 0).

26.4.6.5 Step Command Delay

The Step Delay Timer (SDLY) is a convenient method to make each step command consume a specific amount of time. Normally, the user specifies a step delay equal to the duration of a peripheral function, such as the ADC conversion time. The step delay enables the user to generate the trigger output signals at a controlled rate, thereby avoiding overload on the target peripheral.

The PTGSDLIM register defines the additional time duration of each step command in terms of PTG clocks.

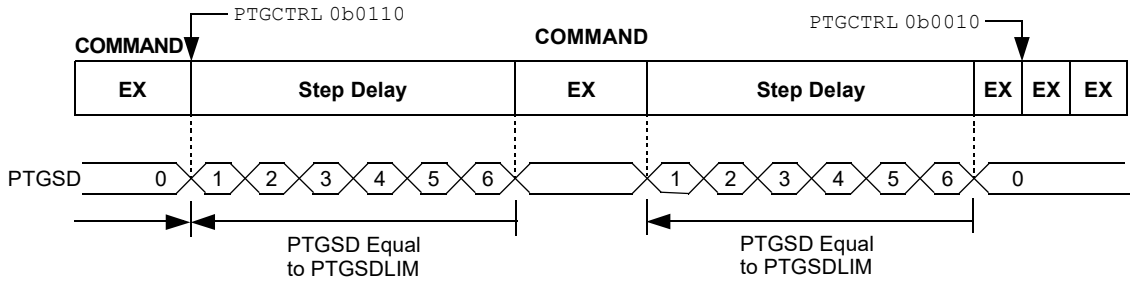
By default, the SDLY is disabled. The user can enable and disable the SDLY via the 'PTGCTRL 0b0110' or 'PTGCTRL 0b0010' command, which can be placed in the step queue.

When operating, the SDLY increments at the PTG clock rate. The PTGSDLIM register value is referred to as the step delay timer limit value. The step delay is inserted after each command is executed, so that all step commands are stalled until the PTGSD timer reaches its limit value. On reaching the limit value, the command execution completes and the next command starts execution. The timer is also cleared during execution of each command, so that it is ready for the next command execution.

Note: The PTGSDLIM register value of '0x0000' does not insert the additional PTG clocks when the step delay timer is enabled. The PTGSDLIM register value of '0x0001' inserts a single PTG step delay (one PTG clock) into every subsequent instruction after the step delay timer is enabled.

The trigger sources for the edge-sensitive commands ('PTGCTRL 0b1011' and PTGWHI/PTGWLO when operated in Edge-Sensitive mode) have additional logic to recognize the appropriate edge transition. The edge detection logic is reset at the end of each command to maintain the edge-sensitive nature of these input triggers. If an additional valid edge occurs during a step delay that has been inserted after the step command has executed, it will not be recognized by any subsequent step command. Figure 26-9 explains the operation of the step command delay. As shown, step delay is inserted immediately following the PTGCTRL 0b0110 command which enables step delay, and is not inserted after the PTGCTRL 0b0010 command which disables step delay; the change is immediate.

Figure 26-9. Operation of Step Command Delay



26.4.7 PTG Watchdog Timer

In some applications, a Watchdog Timer (WDT) may be required as the PTG can wait indefinitely for an external event when executing the following commands:

- Wait for hardware trigger positive edge or High state (PTGWHI)
- Wait for hardware trigger negative edge or Low state (PTGWLO)

The WDT is enabled and it starts counting when the command starts to execute. It is disabled when the command completes execution and prior to any step delay insertion. All other commands execute with the predefined cycle counts.

Note: The PTG Watchdog Timer is not enabled during execution of the '`PTGCTRL 0b1011`' or '`PTGCTRL 0b1010`' command. It is assumed that correct operation of the device will be monitored through other means.

26.4.7.1 Operation Overview

If an expected event fails to arrive before the WDT time-out period expires, the PTG module:

1. Aborts the (failing) command underway.
2. Halts the sequencer (PTGSTRT = `0`).
3. Sets PTGWDTO = `1`.
4. Issues a Watchdog Timer error interrupt to the processor.

The user can either use the Watchdog Timer error interrupt or periodically poll the PTGWDTO bit (PTGCON[6]) to determine that a WDT event has occurred.

26.4.7.2 Configuration

The WDT is configured by setting the PTGWDT[2:0] bits (PTGCON[18:16]). The WDT counts the PTG clocks as defined by the PTG input clock selection and the PTGDIV[4:0] bits in the PTGCON register. For more information, refer to 26.4.2. PTG Clock Selection. The WDT time-out count value is selected by using the PTGWDT[2:0] bits and is disabled when PTGWDT[2:0] = `0b000`.

Notes:

1. The WDT is disabled prior to insertion of any step delay; therefore, the user does not need to account for the Step delay when calculating a suitable WDT time-out value.
2. Some bits within the PTGCON register are read-only when PTGSTRT = `1` (sequencer executing commands). Refer to 26.3.1. PTGCON.

26.4.7.3 Watchdog Timer Event Recovery

If a WDT event occurs, the user has the option to take the necessary action to identify and fix the problem and then continue the step command sequence, or the user can simply restart the sequence.

To clear the PTGWDTO bit and to restart the PTG sequencer from the start of the step queue, disable (ON = `0`) and re-enable (ON = `1`) the PTG module and then restart execution (PTGSTRT = `1`).

Alternatively, as the sequencer is only halted (not reset), the user has the option to examine the PTGQPTR register to identify which step command was the source of the problem and then can take a corrective action. The offending command is aborted prior to the PTGQPTR update. Therefore, it will still address the failing command after the WDT event. After the PTGWDTO bit is cleared, the step queue can be restarted at the same command by setting PTGSTRT = `1`.

Note: The user should clear the PTGWDTO bit after a WDT event. Failing to clear the bit will not interfere with the subsequent module operation, but it will not be possible for the bit to poll any future WDT events.

26.4.8 PTG Module Outputs

The PTG module can generate trigger, interrupt and strobed data outputs by execution of specific step commands.

26.4.8.1 Trigger Outputs

The PTG module can generate up to 32 unique trigger output signals. There are two types of trigger output functions:

- Individual

- Broadcast

The PTG module can generate an individual output trigger on any one of the trigger outputs using the `PTGTRIG` command.

The individual trigger outputs may be assigned to various functions, such as ADC conversion triggers or PPS input/output signals. PPS signals can be routed to external pins or used as inputs to certain other peripherals; refer to the **11.2.2. Peripheral Pin Select (PPS) Overview** chapter for more information.

The broadcast trigger output feature is specified by the PTGBTE register. Each bit in the register corresponds to an associated individual trigger output. If a bit is set in the PTGBTE register, and a broadcast trigger step command ('`PTGCTRL 0b1111`') is executed, the corresponding individual trigger output is asserted. The broadcast trigger output enables the user to simultaneously generate a large number of trigger outputs with a single step command.

26.4.8.2 Interrupt Outputs

The PTG module can generate up to 16 unique interrupt request signals. These signals are useful for interacting with an application software to create more complex functions.

The PTG module can generate an individual interrupt pulse by using the `PTGIRQ` command.

26.4.8.3 Strobe Output

The strobe output of the PTG module can be used to output data from the PTG module. Typically, this output is connected to the ADC Channel Selection register, allowing the PTG to loop through ADC channels. The device-specific data sheet will indicate how the PTG strobe output is connected to other peripherals.

The '`PTGCTRL 0b1110`' command writes the PTGL0 register contents to the strobe output. The PTGL0 register can be modified by using the `PTGADD` and `PTGCOPY` commands.

The '`PTGCTRL 0b1100`' command writes the PTGC0LIM register contents to the strobe output. The '`PTGCTRL 0b1101`' command writes the PTGC1LIM register contents to the strobe output.

26.4.8.4 Output Timing

All triggers, interrupts and data strobe outputs are internally asserted by the PTG state machine when the corresponding step command execution starts (i.e., before any additional time specified by the step delay timer) on the rising edge of the PTG execution clock.

Note: The trigger and strobe output pulse generators operate independently of the PTG sequencer and will not stall command execution to allow a pulse to complete. If a trigger output is requested by a step command before a previous trigger pulse has completed, the previous pulse will be terminated and the new trigger output will commence as normal. If a strobed data command is executed before one PTG clock cycle after a previous data strobe has completed, the behavior is undefined.

In Pulse mode (PTGTOGL = `0`), the width of the trigger output signals is determined by the PTGPWD[3:0] bits (PTGCON[23:20]) and can be any value between 1 and 16 PTG clock cycles. The default value is one PTG clock cycle.

Refer to 26.4.8.4.2. TRIG Negation When PTGTOGL = 1 when operating in Toggle mode (PTGTOGL = `1`).

When globally controlled by the '`PTGCTRL 0b1111`' broadcast trigger command, the TRIG output pulse width is determined by the PTGPWD[3:0] bits (PTGCON[23:20]) and can be any value between 1 and 16 PTG clock cycles. The default value is one PTG clock cycle.

Note: The trigger generated by using the '`PTGCTRL 0b1111`' broadcast trigger command can only operate in Pulse mode (i.e., PTGTOGL = 'don't care').

26.4.8.4.1 TRIG Negation When PTGTOGL = 0

If generating an individual trigger output when the PTGTOGL bit (PTGCON[12]) = 0,or if generating a broadcast trigger output, the TRIG output(s) pulse width is determined by the PTGPWD[3:0] bits.

26.4.8.4.2 TRIG Negation When PTGTOGL = 1

If generating an individual trigger output when the PTGTOGL bit (PTGCON[12]) = 1, the TRIG outputs will remain set until the PTGTRIG command is executed again. On the start of the PTGTRIG command execution, the TRIG outputs are toggled at the beginning of the PTG execution clock.

Note: The PTGTOGL bit has no effect on the operation of the 'PTGCTRL 0b1111' multiple trigger (broadcast) generation command with the following exception:

- If a target trigger output is already in the logic '1' state, having been triggered individually while PTGTOGL is active, the 'PTGCTRL 0b1111' command will have no effect and the trigger output will remain at logic '1'.

26.4.9 Stopping the Sequencer

When the PTG module is disabled (ON = 0), the PTG clocks are disabled (except the trigger pulse counter), the sequencer stops execution and the module enters its lowest power state. The PTGSTRT, PTGSWT, PTGWDTO and PTGQPTR[4:0] bits are all reset. All other bits and registers are not modified. All of the control registers can be read or written when ON = 0.

When the ON bit is cleared, a command that is underway is immediately aborted if the command is waiting for any of the following actions:

- An input from another source
- A timer match
- The step delay to expire (for more information, refer to 26.4.6.5. Step Command Delay)

All other commands are allowed to complete before the PTG module is disabled.

When the PTG module is halted, all of the control registers remain in their present state. The PTG module can be halted by the user by clearing the PTGSTRT bit, or in the event of a Watchdog Timer time-out, which also clears the PTGSTRT bit. Refer to 26.4.7. PTG Watchdog Timer for more information.

Note: If a command has triggered the pulse width delay counter, the counter is synchronously reset with respect to the PTG clock, terminating the pulse (subject to a minimum pulse width of 1 PTG clock cycle).

26.4.10 Single-Step Mode

For debugging purposes, the PTG can be configured to run in Single-Step mode. In this mode, step commands are not executed continuously. Instead, one command is executed at a time and the PTG halts execution after each command. This allows the user to step through the step queue, similarly to stepping through CPU instructions while debugging.

26.4.10.1 Entering Single-Step Mode

Single-Step mode is entered by setting the PTGSSEN bit (PTGCON[9]) = 1. This bit may only be accessed from a debugging session and will read as '0' at all other times. For example, the following steps can be used to set PTGSSEN:

1. In MPLAB® X IDE, program the device to run in Debug mode.
2. Pause code execution at any point where PTGSTRT has not been set to '1'.
3. Add a variable watch for the PTGCON register.
4. Using the watch, set the PTGSSEN bit to '1'.

Note: The value of PTGSTRT must be '0' when setting the PTGSSEN bit or Single-Step mode will not be entered.

Note: Due to the Debug mode restriction, it is not possible to read PTGSSEN from application code to determine whether Single-Step mode has been entered as it will always read as '0'.

26.4.10.2 Executing a Single-Step Command

Once the PTG is in Single-Step mode, setting PTGSTRT = 1 will cause the PTG to execute the command pointed to by PTGQPTR and then halt. Once this step command has been executed, PTGSTRT is cleared by hardware and can be set again to execute another command. PTGQPTR will then point to the next command to execute; however, its value can be changed while PTGSTRT = 0, allowing single-stepping to take place at any queue location.

Note: PTGSTRT can be cleared manually during single-step execution, halting the sequencer. This is only effective when the currently executing command is waiting for input from an external source or a timer match, as all other commands will be allowed to complete.

26.4.10.3 PTG Step Interrupt

After each step command is executed in Single-Step mode, the PTG step interrupt will occur, if enabled. This notifies the application of the completion of the command.

Example 26-4. Single-Step Demonstration Program

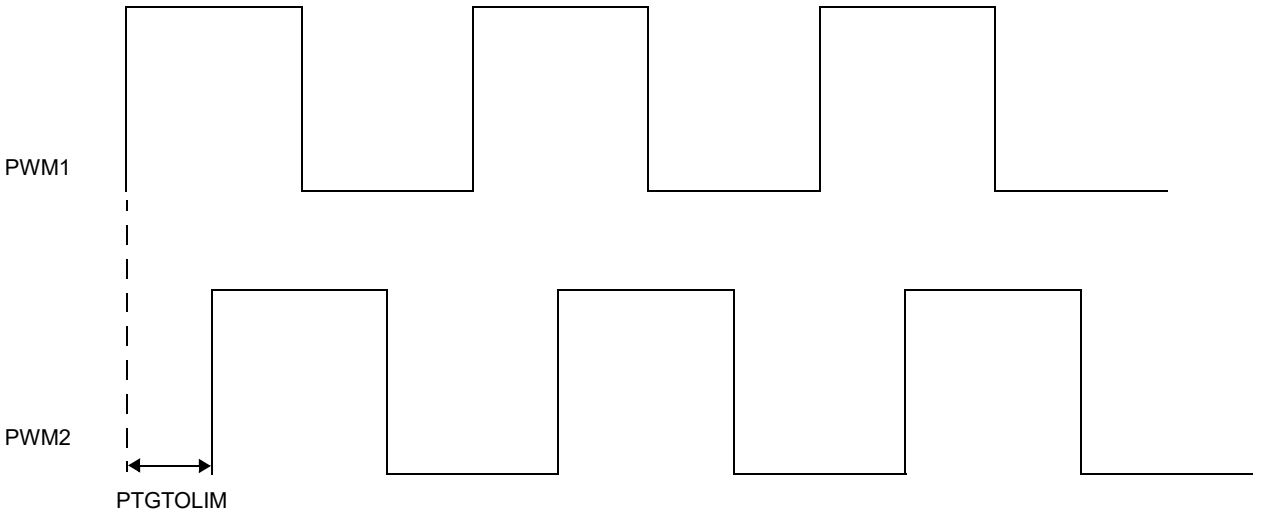
```
//This code shows a basic example of using PTG Single-Step Mode.
//Program must be run in Debug Mode.
//Once the PTGSSEN bit is set, press the push button (RE9) to set PTGSTRT to execute a
single command.
//An LED (RE0) will toggle on the completion of each command.

#include <xc.h>

void PTG_populate_queue() {
    //Set up commands in the PTG Step Queue
    PTGQUE0bits.STEP0 = 0;             //NOP
    PTGQUE0bits.STEP1 = 0b10100000;    //PTGJMP(0)
}

int main (void) {
    _TRISE9 = 1;                       //RE9 input connected to push button switch
    _Bool sw_latch = 0;                //Latch button input so one press makes one step

    _TRISE0 = 0;                       //RE0 output connected to external indicator
    _LATE0 = 0;                        //RE0 output low initially

    PTG_populate_queue();              //Set up step commands in PTG queue
    PTGCONbits.ON = 1;              //Place breakpoint on this line. When execution halts,
set PTGSSEN = 1 using the debugger, then continue.

    _PTGSTEPIE = 1;                    //Enable PTG Step Interrupt

    while(1) {
        if (!_RE9) {                   //Check if button is pressed (active low)
            if (!sw_latch) {
                _PTGSTRT = 1;     //Execute a single Step command
                sw_latch = 1;
            }
        }
        else {
            sw_latch = 0;             //Reset latch for next button press
        }
    }

    return 0;
}

void __attribute__ ((__interrupt__, no_auto_psv)) _PTGSTEPInterrupt(void) {
    _LATE0 = !_LATE0;                  //Toggle RE0 to show step has taken place
    _PTGSTEPIF = 0;
}
```

26.5 Application Examples

26.5.1 Generating Phase-Shifted Waveforms

Figure 26-10 shows an application example for generating phase-shifted PWM waveforms. In this example, PWM1 generates a waveform and the rising edge of this pulse is the trigger input to the PTG module. When the trigger from PWM1 is received, the PTG module waits on PTG Timer 0 using the PTGCTRL(0b1000) command, inserting a programmable delay and then outputting a PCI signal to PWM2. This signal is the synchronization source for PWM2, which is configured to output a single cycle with the same period and duty cycle as PWM1. As PWM2 is repeatedly triggered by the PTG, it outputs a phase-shifted version of PWM1, with the phase shift determined by the PTG Timer 0 delay.

Figure 26-10. Phase-Shifted Waveform Example Application



Example 26-5 shows code for generating a phase-shifted waveform.

Example 26-5. Generating Phase-Shifted Waveforms

```
#include "xc.h"
#include "ptg.h"     //Contains Examples 2-1, 2-2, and 2-3

void IO_initialize() {
    _PCI12R = 167;   //Connect PCI 12 to PTG trigger 26
    _TRISD2 = 0;     //PWM1H will be output on RD2
    _TRISD0 = 0;     //PWM2H will be output on RD0
}

void PWM1_initialize() {

    PG1CONbits.CLKSEL = 1; //Main PWM clock (undivided/unscaled) used for PWM2
    PG1CONbits.MSTEN = 1; //Broadcast status of update bit/EOC to other
generators
    PG1IOCONbits.PENH = 1; //PWM generator 1 controls PWM1H pin
    PG1EVTbits.ADTR2EN1 = 1; //PGA1TRIGA match controls PWM1 ADC Trigger 2

    PG1PER = 8000;         //Period 1ms for a 8MHz PWM clock
    PG1DC = 4000;          //50% duty cycle
    PG1PHASE = 0;          //0 phase offset
    PG1TRIGA = 0x0000;     //PWM ADC trigger 2 will happen at start of cycle

    PG1CONbits.ON = 1;     //Enable PWM Generator 1
}

void PWM2_initialize() {
    PG2CONbits.CLKSEL = 1; //Main PWM clock (undivided/unscaled) used for PWM2
    PG2CONbits.TRGMOD = 1; //PWM generator is re-triggerable
    PG2CONbits.SOCS = 0b1111; //PCI sync used for start of cycle
    PG2IOCONbits.PENH = 1; //PWM generator 2 controls PWM2H pin
    PG2PER = 8000; //Period 1ms for a 8MHz PWM clock
    PG2DC = 4000; //50% duty cycle
    PG2PHASE = 0; //0 phase offset
    PG2SPCIbits.PSS = 0b01100; //PCI12 as PWM2 sync source
    PG2CONbits.ON = 1; //Enable PWM Generator 2
}

void PTG_initialize() {
    PTGT0LIM = 1000; //0.125ms T0 delay
    PTGQUE0bits.STEP0 = PTGWHI(0); //Wait for high-to-low edge on trigger from
PWM1
    PTGQUE0bits.STEP1 = PTGCTRL(t0Wait); //Wait on T0
    PTGQUE0bits.STEP2 = PTGTRIG(26); //Trigger PWM2
    PTGQUE0bits.STEP3 = PTGJMP(0); //Restart sequence
    PTGCONbits.ON = 1; //Enable PTG
    PTGCONbits.PTGSTRT = 1; //Start executing commands
}

void clocks_initialize() {
    //Assuming 8MHz FRC is default clock selection:
    CLK5CONbits.ON = 1;     //8MHz for the Main PWM clock
    CLK10CONbits.ON = 1;    //8MHz for the PTG clock
}

int main(void) {
    clocks_initialize();
    IO_initialize();
    PWM1_initialize();
    PWM2_initialize();
    PTG_initialize();

    while (1);

    return 0;
}
```

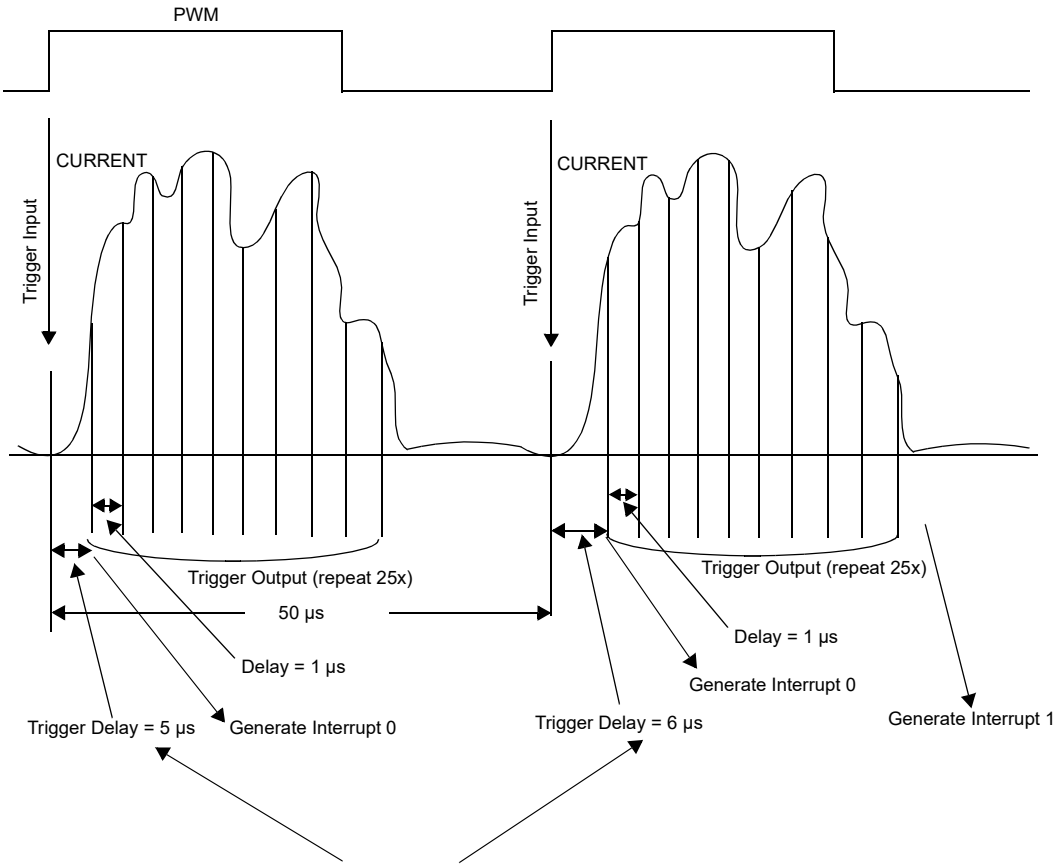
26.5.2 Interleaving Samples Over Multiple Cycles

Figure 26-11 shows the waveforms of an application where the power in a system needs to be accurately measured when the current load is highly dependent on temperature, voltage and user application. The current waveforms vary widely per usage, but over a few PWM cycles, the waveforms are relatively stable.

The goal of this example is to collect many current and voltage readings over several PWM cycles in an interleaved manner. The data are stored in the memory during acquisition and later processed (integrated) to yield an accurate power value.

This example shows a situation where it would not be practical or possible for software to accurately schedule the ADC samples.

Figure 26-11. Example Application – Average Power Calculation



Note: The trigger delay value is modified to make the subsequent sample triggers shift in time, thereby enabling the interleaving of the samples.

26.5.2.1 Interleaved Sampling Step Command Program

This section describes the step command programming for implementing the timing sequence shown in Figure 26-11.

The following assumptions are made:

1. Trigger Input 0 is connected to the PWM1 signal. The rising edge of this signal starts the sequence. PWM1 has a period of 50us.

2. Output Trigger 12 is connected to the ADC module. This signal gives command to the ADC module to begin a sample and conversion process.
3. Interrupt 0 is used to indicate to the processor that a subsequence has started (provides status).
4. Interrupt 1 is used to indicate to the processor that the entire sequence has completed.
5. The PTG clock source is 125 MHz.
6. The initial trigger delay is 5 µs.
7. The second trigger delay is 6 µs.
8. In each PWM cycle, the ADC will be triggered 25 times.
9. The basic sequence is executed twice.

Example 26-6. Interleaved Sampling Step Command Program

```
#include "xc.h"
#include "ptg.h"         //Contains Examples 2-1, 2-2, and 2-3

//Allocate buffers for ADC results. Size of 50 is based on 2 PWM cycles.
unsigned int voltage_buffer[50];
int voltage_buffer_index = 0;
unsigned int current_buffer[50];
int current_buffer_index = 0;
_Bool readings_ready = 0;

void clocks_initialize() {
    //To do: Set up CPU clock (Fcy) to run at 50 MHz.

    //Set up PLL1:
    //PLL1 has input frequency 8MHz (FRC)
    PLL1DIVbits.PLLPRE = 1;     //Reference input will be 8MHz, no division
    PLL1DIVbits.FBDIV = 125;    //Fvco = 8MHz * 125 = 1000MHz
    PLL1DIVbits.POSTDIV1 = 5;   //Divide Fcvo by 5
    PLL1DIVbits.POSTDIV2 = 4;   //Fpllo = Fvco / 5 / 4 = 50 MHz
    //Key assumption: CPU clock provided by PLL was inherently divided by 4
    //on 33C (Fcy = Fosc /2 and Fosc = Fpllo / 2 -> Fcy = Fpllo / 4), but,
    //these dividers are not present on 33A.
    //Need to verify this.

    //To do: Configure VCO 1 divider to divide by 4, rather than 8 (!)
    VCO1DIVbits.INTDIV = 4; //Divide Fvco by 8 (???)

    PLL1CONbits.ON = 1; //Enable PLL generator 1
    while(!PLL1CONbits.CLKRDY); //Wait for PLL1 clock to be ready

    CLK1CONbits.NOSC = 5; //Set PLL1 Fout as CPU clock source
    CLK1CONbits.OSWEN = 1; //Request clock switch
    while (CLK1CONbits.OSWEN); //Wait for switch to complete

    //To do: Configure CLKGEN5 to provide a 125MHz clock for the main PWM clock

    CLK5DIVbits.INTDIV = 2; //Divide input clock by 2

    CLK5CONbits.NOSC = 7; //Use divided PLL1 VCO (250MHz)
    CLK5CONbits.OSWEN = 1; //Request clock switch
    while (CLK5CONbits.OSWEN); //Wait for switch to complete

    //To do: Configure CLKGEN6 to provide a 250MHz input clock to the ADC

    CLK6CONbits.NOSC = 7; //Use divided PLL1 VCO (250MHz)
    CLK6CONbits.OSWEN = 1; //Request clock switch
    while (CLK6CONbits.OSWEN); //Wait for switch to complete

    //To do: Configure CLKGEN10 to provide a 125MHz clock for the PTG.

    CLK10DIVbits.INTDIV = 2; //Divide input clock by 2

    CLK10CONbits.NOSC = 7; //Use divided PLL1 VCO (250MHz)
    CLK10CONbits.OSWEN = 1; //Request clock switch
    while (CLK10CONbits.OSWEN); //Wait for switch to complete
}
```

```
void PTG_initialize() {

    //Enable PTG interrupts 0 and 1
    _PTG0IE = 1;
    _PTG1IE = 1;

    //Set up control registers
    PTGT0LIM = 625; //5us T0 delay
    PTGT1LIM = 125; //1us T1 delay
    PTGC0LIM = 24; //Repeat C0 loop 25 times
    PTGC1LIM = 1; //Repeat C1 loop once
    PTGHOLD = 625; //5us (used to restore T0 delay)
    PTGADJ = 125; //1us (added to T0 delay)
    PTGQPTR = 0; //Initialize step queue pointer

    //Outer loop
    PTGQUE0bits.STEP0 = PTGWHI(0); // Wait for positive edge trigger 0 (PWM1
ADC Trigger 2)
    PTGQUE0bits.STEP1 = PTGCTRL(t0Wait); // Start PTGT0, wait for time out
    PTGQUE0bits.STEP2 = PTGIRQ(0); // Generate IRQ 0
        // Inner loop
        PTGQUE0bits.STEP3 = PTGTRIG(12); // Generate output trigger 12 (ADC
conversion)
        PTGQUE1bits.STEP4 = PTGCTRL(t1Wait); // Start PTGT1, wait for time out
        PTGQUE1bits.STEP5 = PTGJMPC0(3); // Go to STEP3 if PTGC0 != PTGC0LIM,
increment PTGC0
        // End inner loop
    PTGQUE1bits.STEP6 = PTGADD(t0Limit); // Add PTGADJ to PTGT0LIM
    PTGQUE1bits.STEP7 = PTGJMPC1(0); // Jump to 0 PTGC1LIM times (once, making
2 iterations)
    // End outer loop

    PTGQUE2bits.STEP8 = PTGIRQ(1); // Generate IRQ 1
    PTGQUE2bits.STEP9 = PTGCOPY(t0Limit); // Copy PTGHOLD to PTGT0LIM (restore
original value)
    PTGQUE2bits.STEP10 = PTGJMP(0); // Jump to start of queue

    //Start PTG
    PTGCONbits.ON = 1;
    PTGCONbits.PTGSTRT = 1;
}
void PWM1_initialize() {
    _TRISD2 = 0; //Enable PWM1H output pin to observe cycle

    PG1CONbits.CLKSEL = 1; // //Main PWM clock (no dividing or scaling) used
for PWM1
    PG1IOCONbits.PENH = 1; // PWM generator 1 controls PWM1H pin (RD2)
    PG1EVTbits.ADTR2EN1 = 1; // Enable PG1TRIGA match as PWM1 ADC Trigger 2
source

    PG1PER = 5000; //Period of 50us
    PG1PHASE = 0; //0 phase offset
    PG1DC = 2500; //50% duty cycle

    PG1TRIGA = 0x0000; // ADC trigger 2 (PTG input) will happen at start of
cycle
    PG1CONbits.ON = 1; // Enable PWM Generator 1
}
void ADC_initialize() {
    //Note: These pin choices will most likely need to change. Review actual
pinout.

    //Use RA2 as input to be converted by ADC (voltage reading)
    _ANSA2 = 1;
    _TRISA2 = 1;
    //Use RA4 as input to be converted by ADC (current reading)
    _ANSA4 = 1;
    _TRISA4 = 1;

    //To do: Initialize and calibrate ADC
    AD1CONbits.ON = 1; //Is more setup needed first? If not this feels
backwards lol

    //Need to figure out: Does the ADC automatically calibrate itself on power
on? Or is more intervention needed?
    while(!AD1CONbits.ADRDY);
```

```
    //AD1P0 input corresponds to data channel 0 and is triggered by ADC
trigger 30
    AD1CHCON0bits.MODE = 0; //Single-sample mode
    AD1CHCON0bits.PINSEL = 0; //Positive input is AN1P0/RA2
    AD1CHCON0bits.NINSEL = 0; //Single-ended mode

    AD1CHCON0bits.TRG1SRC = 30; //Channel 0 triggered by ADC trigger 30 (PTG
trigger 12)

    //AD1P1 input corresponds to data channel 1 and is triggered by ADC
trigger 30
    AD1CHCON1bits.MODE = 0; //Single-sample mode
    AD1CHCON1bits.PINSEL = 1; //Positive input is AN1P1/RA4
    AD1CHCON1bits.NINSEL = 0; //Single-ended mode

    AD1CHCON1bits.TRG1SRC = 30; //Channel 1 ADC trigger 30 (PTG trigger 12)

    //Enable ADC Channel 0 and 1 interrupts
    _AD1AN0IE = 1;
    _AD1AN1IE = 1;
    //To do: Find any other interrupt enables (there were others on 33C)
}
void calculate_average_power() {
    //To do: Use the buffered ADC readings to calculate average power over the
last 2 PWM cycles.
}
int main(void) {

    clocks_initialize();
    PWM1_initialize();
    ADC_initialize();
    PTG_initialize();

    while(1) {
    //Wait for readings to be ready
        if (readings_ready) {
            //Perform power calculations and reset for next time.
            calculate_average_power();
            readings_ready = 0;
        }
    }
    return 0;
}

void __attribute__((__interrupt__, no_auto_psv)) _PTG0Interrupt() {
    _PTG0IF = 0;
    //Interrupt indicates a series of 25 ADC readings will now be collected.
}
void __attribute__((__interrupt__, no_auto_psv)) _PTG1Interrupt() {
    _PTG1IF = 0;
    //2 PWM cycles of ADC results have been collected, average power can be
calculated for those 2 cycles.
    readings_ready = 1;
    //Reset buffer indices here to prevent overrun.
    voltage_buffer_index = 0;
    current_buffer_index = 0;
}

void __attribute__((__interrupt__, no_auto_psv)) _AD1AN0Interrupt() {
    unsigned int voltage_result = AD1DATA0;
    _AD1AN0IF = 0;
    //Buffer ADC result for later use
    voltage_buffer[voltage_buffer_index++] = voltage_result;
}

void __attribute__((__interrupt__, no_auto_psv)) _AD1AN1Interrupt() {
    unsigned int current_result = AD1DATA1;
    _AD1AN1IF = 0;
    //Buffer ADC result for later use
    current_buffer[current_buffer_index++] = current_result;
}
```

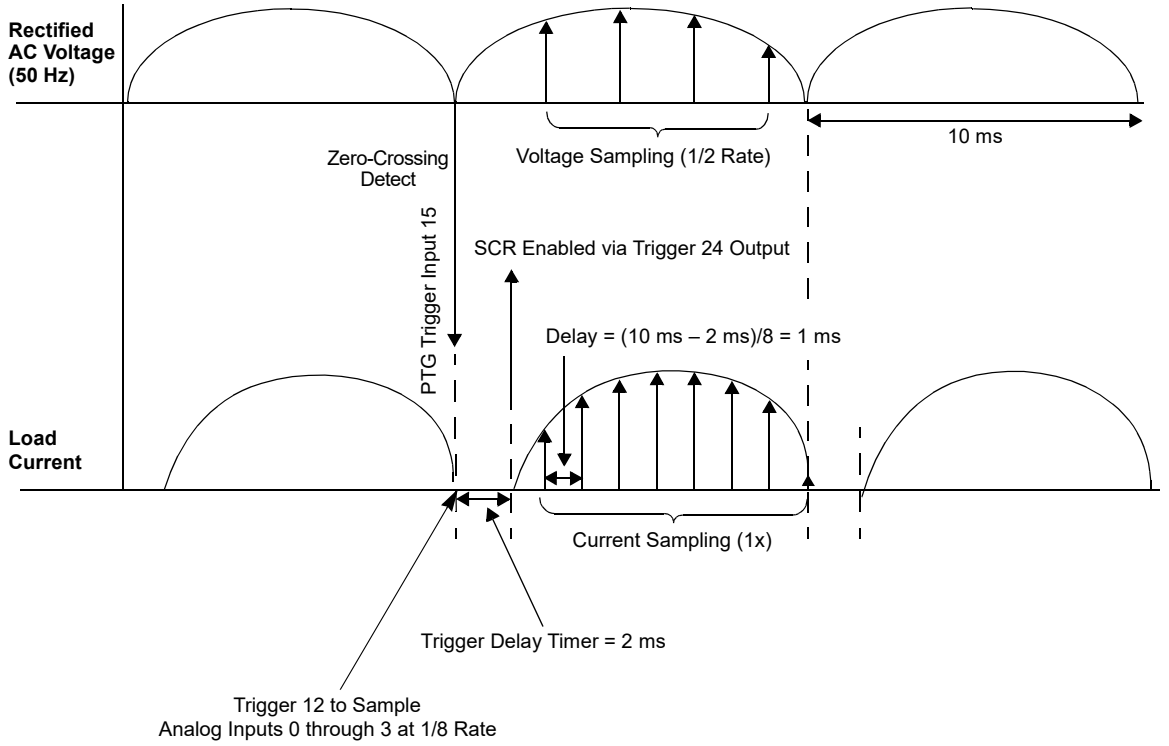
See Example 26-1 for PTG command definitions. See Example 26-2 and Example 26-3 for `PTGCTRL`, `PTGADD` and `PTGCOPY` command options.

26.5.3 Sampling at Multiple Rates

Figure 26-12 shows an application example of sampling an ADC input at a fast rate (1x rate), a second analog input at a slower rate (1/2 rate) and four other inputs at a 1/8 rate. The example is a motor control application using a Silicon Controlled Rectifier (SCR) that triggers at a specified time after the AC line zero-crossing.

While this example uses the simple binary sampling ratios, the PTG module can generate a very wide range of sample ratios to meet the requirements of an application. This example demonstrates coordination between the PTG, ADC, PWM and PPS in order to achieve the required sampling rates.

Figure 26-12. Example Application – Ratioed Sampling



26.5.3.1 Ratioed Sampling Step Command Program

This section describes the step command programming for implementing the timing sequence shown in Figure 26-12.

The following assumptions are made:

- Trigger Input 15 is connected to the zero-crossing detect. The rising edge of the Zero-Crossing Detect signal starts the sequence.
- Trigger Output 24 enables the SCR in the application circuit.
- The trigger delay from Trigger Input 15 to the generation of Trigger Output 24 is 2 ms.
- Trigger Output 26 is connected to PWM1's synchronization signal.
- PWM1 is configured to trigger the ADC to sample at 1/2x rate, and it is configured to trigger PWM2 which will sample the ADC at 1x rate.
- Trigger Output 12 is connected to the ADC to sample other data values (channels 0, 1, 2, and 3) once per cycle.

- The PTG clock is 8 MHz.

Example 26-7. Ratioed Sampling Step Command Program

```
/*
 * File:   main.c
 * Author: C68555
 *
 * Created on August 26, 2022, 10:00 AM
 */

#include "xc.h"
#include "ptg.h"                 //Contains Examples 2-1, 2-2, and 2-3

void clocks_initialize() {
    //Assumption: CLKGEN5 uses FRC @ MHz by default and does not need further
configuration.
    CLK5CONbits.ON = 1;

    //Configure CLKGEN6 to provide a 250MHz input clock to the ADC.

    //Set up PLL1
    PLL1DIVbits.PLLPRE = 1;     //Reference input will be 8MHz, no division
    PLL1DIVbits.FBDIV = 125;    //Fvco = 8MHz * 125 = 1000MHz
    PLL1DIVbits.POSTDIV1 = 5;   //Divide Fcvo by 4
    PLL1DIVbits.POSTDIV2 = 1;   //Fpllo = Fvco / 4 / 1 = 250 MHz

    PLL1CONbits.ON = 1;         //Enable PLL generator 1
    while(!PLL1CONbits.CLKRDY); //Wait for PLL1 clock to be ready

    CLK6CONbits.NOSC = 5;       //Set PLL1 Fout as ADC clock source
    CLK6CONbits.OSWEN = 1;      //Request clock switch
    while (CLK6CONbits.OSWEN);  //Wait for switch to complete

    //To do: Configure CLKGEN10 to provide an 8MHz clock for the PTG
    CLK10CONbits.ON = 1;
}

void PTG_initialize() {

    PTGT0LIM = 16000;          // 2 ms T0 delay
    PTGT1LIM = 8000;           // 1 ms T1 delay
    PTGC0LIM = 2;              // 3 iterations of the C0 loop

    //Other PTG registers are unused
    PTGQPTR = 0;

    PTGQUE0bits.STEP0 = PTGWHI(15);      // Wait for trigger from INT2 (zero
crossing detect)
    PTGQUE0bits.STEP1 = PTGTRIG(12);     // Take 1/8 rate samples using ADC
trigger 30
    PTGQUE0bits.STEP2 = PTGCTRL(t0Wait); // Wait 2ms
    PTGQUE0bits.STEP3 = PTGTRIG(24);     // Trigger PPS output 55 (SCR)
    PTGQUE1bits.STEP4 = PTGCTRL(t1Wait); // Wait1ms before starting the 1x and
1/2x triggers
    PTGQUE1bits.STEP5 = PTGTRIG(26);     // Trigger PWM1 to trigger
conversions at 1x and 1/2x rates
        //Start main loop
        PTGQUE1bits.STEP6 = PTGCTRL(t0Wait); // Wait2ms (pre-trigger delay)
for subsequent triggers
        PTGQUE1bits.STEP7 = PTGTRIG(26); // Trigger PWM1 to trigger ADC
conversions at 1x and 1/2x rates
        PTGQUE2bits.STEP8 = PTGJMPC0(6); // Jump to step 6 (twice, for 3
iterations total)
        //End main loop
    PTGQUE2bits.STEP9 = PTGJMP(0);

    //Start the PTG
    PTGCONbits.ON = 1;
    PTGCONbits.PTGSTRT = 1;
}
void ADC_initialize() {

    //Enable analog inputs AN1P0 - AN1P5
    _ANSA2 = 1;
    _ANSA4 = 1;
```

```
    _ANSA6 = 1;
    _ANSA5 = 1;
    _ANSB1 = 1;
    _ANSB3 = 1;

    //To do: Configure and initialize ADC (should be same code for Example 2)
    AD1CONbits.ON = 1; //Is more setup needed first? If not this feels
backwards lol

    //Need to figure out: Does the ADC automatically calibrate itself on power
on? Or is more intervention needed?
    while(!AD1CONbits.ADRDY);

    //To do: Connect ADC inputs to core 1 channels 0 - 5
    AD1CHCON0bits.MODE = 0; //Single-sample mode
    AD1CHCON0bits.PINSEL = 0; //Positive input is AN1P0/RA2
    AD1CHCON0bits.NINSEL = 0; //Single-ended mode

    AD1CHCON1bits.MODE = 0; //Single-sample mode
    AD1CHCON1bits.PINSEL = 1; //Positive input is AN1P1/RA4
    AD1CHCON1bits.NINSEL = 0; //Single-ended mode

    AD1CHCON2bits.MODE = 0; //Single-sample mode
    AD1CHCON2bits.PINSEL = 2; //Positive input is AN1P2/RA6
    AD1CHCON2bits.NINSEL = 0; //Single-ended mode

    AD1CHCON3bits.MODE = 0; //Single-sample mode
    AD1CHCON3bits.PINSEL = 3; //Positive input is AN1P3/RA5
    AD1CHCON3bits.NINSEL = 0; //Single-ended mode

    AD1CHCON4bits.MODE = 0; //Single-sample mode
    AD1CHCON4bits.PINSEL = 4; //Positive input is AN1P4/RB1
    AD1CHCON4bits.NINSEL = 0; //Single-ended mode

    AD1CHCON5bits.MODE = 0; //Single-sample mode
    AD1CHCON5bits.PINSEL = 5; //Positive input is AN1P5/RB3
    AD1CHCON5bits.NINSEL = 0; //Single-ended mode

    //Configure channel triggers
    AD1CHCON0bits.TRG1SRC = 30; //Channel 0 triggered by ADC trigger 30 (PTG
trigger 12)
    AD1CHCON1bits.TRG1SRC = 30; //Channel 1 triggered by ADC trigger 30 (PTG
trigger 12)
    AD1CHCON2bits.TRG1SRC = 30; //Channel 2 triggered by ADC trigger 30 (PTG
trigger 12)
    AD1CHCON3bits.TRG1SRC = 30; //Channel 3 triggered by ADC trigger 30 (PTG
trigger 12)
    AD1CHCON4bits.TRG1SRC = 5; //Channel 4 triggered by PWM1 ADC Trigger 2
    AD1CHCON5bits.TRG1SRC = 7; //Channel 5 triggered by PWM2 ADC Trigger 2

    //Enable interrupts for ADC core 1 channels 0 - 5
    _AD1AN0IE = 1;
    _AD1AN1IE = 1;
    _AD1AN2IE = 1;
    _AD1AN3IE = 1;
    _AD1AN4IE = 1;
    _AD1AN5IE = 1;

}
void PWM1_initialize() {
    PG1CONbits.CLKSEL = 1; //Main PWM clock (no dividing or scaling) used for
PWM1
    PG1CONbits.SOCS = 0b1111; //PCI sync used for start of cycle
    PG1SPCIbits.PSS = 0b01100; //PCI12 as PWM1 sync source
    PG1CONbits.TRGMOD = 1; //PWM generator is re-triggerable
    PG1IOCONbits.PENH = 0; //PWM generator 1 does not control PWM1H pin
    PG1EVTbits.ADTR2EN1 = 1; //Enable PGA1TRIGA match as ADC trigger 2 source
    PG1EVTbits.ADTR1EN1 = 1; //Enable PGA1TRIGA match as ADC trigger 1 source
    PG1EVTbits.PGTRGSEL = 1; //Use TRIGA compare as PWM generator trigger

    PG1PER = 16000; //PWM1 period is 2ms
    PG1DC = 8000; //50% duty cycle
    PG1PHASE = 0; //0 phase offset

    PG1TRIGA = 0; //ADC trigger at 0

    PG1CONbits.ON = 1; //Enable PWM Generator 1
}
```

```
void PWM2_initialize() {
    PG2CONbits.CLKSEL = 1; //Main PWM clock (no dividing or scaling) used for
PWM2
    PG2CONbits.TRGCNT = 1; //PWM2 completes two cycles once triggered
    PG2CONbits.TRGMOD = 1; //PWM2 is re-triggerable
    PG2CONbits.SOCS = 0b0001; //PWM start of cycle from PG1, selected by
PGTRGSEL
    PG2IOCONbits.PENH = 0; //PWM generator 2 does not control PWM2H pin
    PG2EVTbits.ADTR2EN1 = 1; //Enable PGA1TRIGA match as ADC trigger 2 source

    PG2PER = 8000; //PWM2 period is 1ms
    PG2DC = 4000; //50% duty cycle
    PG2PHASE = 0; //0 phase offset

    PG1TRIGA = 0; //ADC trigger at 0

    PG2CONbits.ON = 1; //Enable PWM Generator 2
}
void IO_initialize() {
    //Note: This has been partially adapted from 33C, but is incomplete;
output indicators are based on 33C, need to double check other mappings.

    //Configure INT2 to use RC2 (zero-crossing detect input)
    _TRISC2 = 1;
    _INT2R = 34;
    //Configure PTG trigger 24 to use RC3 (SCR output)
    _TRISC3 = 0;
    _RP65R = 55;
    //Connect PCI12 to PTG Trigger 26, PPS input #167 (RPI166)
    _PCI12R = 167;

    //Output indicators for ADC interrupts
    _TRISD4 = 0;
    _TRISD5 = 0;
    _TRISD6 = 0;
    _TRISD10 = 0;
    _TRISD11 = 0;
    _TRISD12 = 0;

}
int main(void) {
    clocks_initialize();
    IO_initialize();
    ADC_initialize();
    PWM1_initialize();
    PWM2_initialize();
    PTG_initialize();

    while (1);

    return 0;
}
//Note: Need to update this interrupt for 33A (function name, flag name, etc.)
void __attribute__((__interrupt__, no_auto_psv)) _AD1AN0Interrupt() {
    unsigned int adc_result = AD1DATA0; //Reading the result clears the
interrupt
    _AD1AN0IF = 0;
    asm volatile("BTG LATD, #4"); //Inline assembly to accomplish single-cycle
bit toggle
}

void __attribute__((__interrupt__, no_auto_psv)) _AD1AN1Interrupt() {
    unsigned int adc_result = AD1DATA1; //Reading the result clears the
interrupt
    _AD1AN1IF = 0;
    asm volatile("BTG LATD, #5"); //Inline assembly to accomplish single-cycle
bit toggle
}
void __attribute__((__interrupt__, no_auto_psv)) _AD1AN2Interrupt() {
    unsigned int adc_result = AD1DATA2; //Reading the result clears the
interrupt
    _AD1AN2IF = 0;
    asm volatile("BTG LATD, #6"); //Inline assembly to accomplish single-cycle
bit toggle
}
void __attribute__((__interrupt__, no_auto_psv)) _AD1AN3Interrupt() {
    unsigned int adc_result = AD1DATA3; //Reading the result clears the
interrupt
```

```
    _AD1AN3IF = 0;
    asm volatile("BTG LATD, #10"); //Inline assembly to accomplish single-
cycle bit toggle
}
void __attribute__((__interrupt__, no_auto_psv)) _AD1AN4Interrupt() {
    unsigned int adc_result = AD1DATA4; //Reading the result clears the
interrupt
    _AD1AN4IF = 0;
    asm volatile("BTG LATD, #11"); //Inline assembly to accomplish single-
cycle bit toggle
}
void __attribute__((__interrupt__, no_auto_psv)) _AD1AN5Interrupt() {
    unsigned int adc_result = AD1DATA5; //Reading the result clears the
interrupt
    _AD1AN5IF = 0;
    asm volatile("BTG LATD, #12"); //Inline assembly to accomplish single-
cycle bit toggle
}
```

26.6 Interrupts

The PTG generates three types of interrupts: the individual interrupt requests, the PTG Step interrupt and the PTG Watchdog Timer time-out interrupt.

26.6.1 PTG Individual Interrupt Requests

The PTG individual interrupt requests are generated using the `PTGIRQ` step command, with the parameter indicating which of several interrupts are supported. See Table 26-6 for specific information. Refer to the **"Interrupt Controller"** section for more information.

26.6.2 PTG Step Interrupt

The PTG Step interrupt is generated in Single-Step mode (PTGSSEN = `1`) each time a step command has completed execution. This can be used for debugging a PTG step command sequence. See 26.4.10. Single-Step Mode for more information about Single-Step mode.

26.6.3 PTG Watchdog Timer (WDT) Interrupt

The PTG WDT interrupt occurs when the WDT times out while waiting for an input trigger using the `PTGWLO/PTGWHI` step commands. This indicates that an expected trigger was missed, allowing the application to take corrective action.

Refer to 26.4.7. PTG Watchdog Timer for more information about the WDT.

26.7 Power-Saving Modes

The PTG module supports three power-saving modes:

- Disabled: the PTG module is not clocked in this mode
- Idle: the processor core and selected peripherals are shut down
- Sleep: the entire device is shut down

26.7.1 Disabled Mode

When ON = `1`, the module is considered in an Active mode and is fully powered and functional. When ON = `0`, the module is turned off. The PTG clock portions of the circuit are disabled for maximum current savings. Only the control registers remain functional for reading and writing to allow the software to change the module's operational mode. The module sequencer is reset.

26.7.2 Idle Mode

To continue full module operation while the device is in Idle mode, the PTGSIDL bit must be cleared prior to entry into Idle mode. If PTGSIDL = `1`, the PTG module will behave the same way in Idle mode as it does in Sleep mode.

26.7.3 Sleep Mode

If the device enters Sleep mode while the PTG module is enabled (ON = 1), the module will be suspended in its current state until clock execution resumes. This situation should be avoided as it might result in unexpected operation. It is recommended that all peripherals be shut down in an orderly manner prior to entering Sleep mode.

27. 32-Bit Programmable Cyclic Redundancy Check (CRC) Generator

The Programmable Cyclic Redundancy Check (CRC) module is a software-configurable CRC generator. The module provides a hardware implemented method of quickly generating checksums for various communication and security applications. The CRC engine calculates the CRC checksum without CPU intervention; moreover, it is much faster than the software implementation.

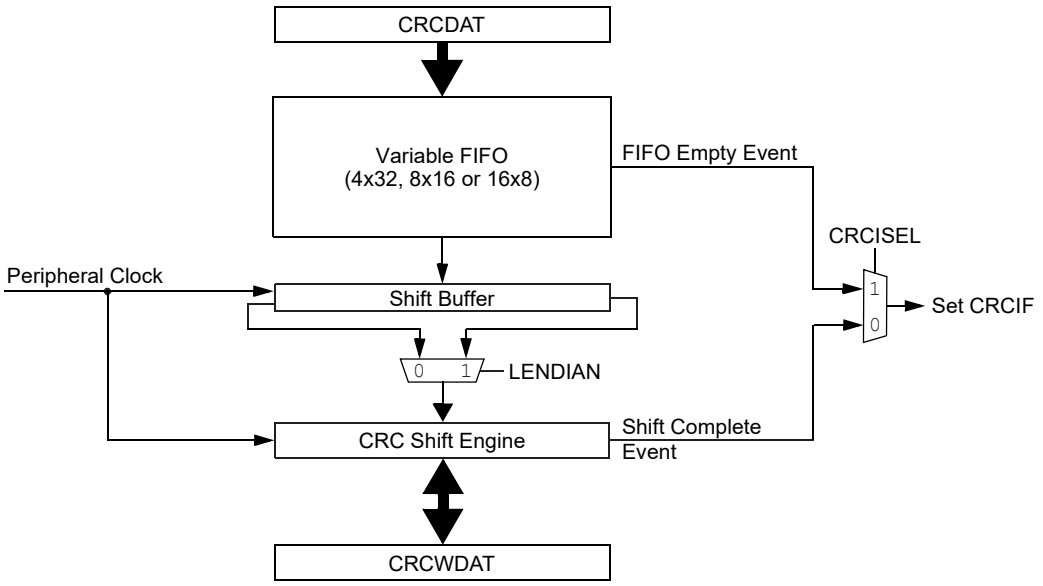
The programmable CRC generator provides the following features:

- User-Programmable CRC Polynomial Equation, up to 32 bits
- Programmable Shift Direction (Little or Big-Endian)
- Independent Data and Polynomial Lengths
- Configurable Interrupt Output
- Data FIFO

27.1 Architectural Overview

The programmable CRC generator module can be divided into two parts: the control logic and the CRC engine. The control logic incorporates a register interface, data FIFO, an interrupt generator and a CRC engine interface. The CRC engine incorporates a CRC calculator, which is implemented using a serial shifter with XOR function. A simplified block diagram is shown in Figure 27-1.

Figure 27-1. Simplified Block Diagram of the Programmable CRC Generator



The checksum is a unique number associated with a message or a particular block of data containing several bytes. Whether it is a data packet for communication or a block of data stored in memory, a checksum, helps to validate it before processing. The simplest way to calculate a checksum is to add together all the data bytes present in the message. However, this method of checksum calculation fails badly when the message is modified by inverting or swapping groups of bytes. Also, it fails when null bytes are added anywhere in the message.

The Cyclic Redundancy Checksum (CRC) is a more complicated but robust error-checking algorithm. The main idea behind the CRC algorithm is to treat a message as a binary bit stream and divide it by a fixed binary number. The remainder from this division is considered to be the checksum. Like in division, the CRC calculation is also an iterative process. The only difference is that these operations are done on modulo arithmetic, based on mod 2. For example, division is replaced with

the XOR operation (i.e., subtraction without carry). The CRC algorithm uses the term polynomial to perform all of its calculations. The divisor, dividend and remainder that are represented by numbers are termed as polynomials with binary coefficients. For example, the number 25h (`11001`) is represented as:

$(1 * x^4) + (1 * x^3) + (0 * x^2) + (0 * x^1) + (1 * x^0)$ or $x^4 + x^3 + x^0$

In order to perform the CRC calculation, a suitable divisor is first selected. This divisor is called the generator polynomial. Since CRC is used to detect errors, a generator polynomial of a suitable length needs to be chosen for a given application, as each polynomial has different error detection capabilities. Some polynomials are widely used for many applications, but the error detecting capabilities of any particular polynomial are beyond the scope of this reference section.

The CRC algorithm is straightforward to implement in software. However, it requires considerable CPU bandwidth to implement the basic requirements, such as shift, bit test and XOR. Moreover, CRC calculation is an iterative process, and additional software overhead for data transfer instructions puts an enormous burden on the MIPS requirement of a microcontroller. In contrast, the software-configurable CRC hardware module facilitates a fast CRC checksum calculation with minimal software overhead.

27.2 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x02C8	CRCCON	31:24				DWIDTH[4:0]				
		23:16				PLEN[4:0]				
		15:8	ON		SIDL	VWORD[4:0]				
		7:0	CRCFULL	CRCEMPTY	CRCISEL	CRCGO	LENDIAN	MOD		
0x02CC	CRCXOR	31:24	X[30:23]							
		23:16	X[22:15]							
		15:8	X[14:7]							
		7:0	X[6:0]							X
0x02D0	CRCDAT	31:24	DATA[31:24]							
		23:16	DATA[23:16]							
		15:8	DATA[15:8]							
		7:0	DATA[7:0]							
0x02D4	CRCWDAT	31:24	CRCWDAT[31:24]							
		23:16	CRCWDAT[23:16]							
		15:8	CRCWDAT[15:8]							
		7:0	CRCWDAT[7:0]							

27.2.1 CRC Control Register

Name: CRCCON
Offset: 0x2C8

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
				DWIDTH[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
				PLEN[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	VWORD[4:0]				
Access	R/W		R/W	R	R	R	R	R
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CRCFULL	CRCEMPTY	CRCISEL	CRCGO	LENDIAN	MOD		
Access	R	R	R/W	R/W	R/W	R/W		
Reset	0	1	0	0	0	0		

Bits 28:24 – DWIDTH[4:0] Data Word Width Configuration bits
Configures the width of the data word (data word width - 1)

Bits 20:16 – PLEN[4:0] Polynomial Length Configuration bits
Configures the length of the polynomial (polynomial length - 1)

Bit 15 – ON CRC Enable bit

Value	Description
1	Enables module
0	Disables module. All state machines, pointers and CRCSHFT/CRCDAT reset. Other SFRs are not reset.

Bit 13 – SIDL CRC Stop in Idle Mode bit

Value	Description
1	Discontinue module operation when device enters Idle mode
0	Continue module operation in Idle mode

Bits 12:8 – VWORD[4:0] Valid Word Pointer Value bits
Indicates the number of valid words in the FIFO

- Has a maximum value of 16 when DWIDTH[4:0] ≦ 7 (data words 8-bit wide or less)
- Has a maximum value of 8 when DWIDTH[4:0] ≦ 15 (data words from 9-bit to 16-bit wide)
- Has a maximum value of 4 when DWIDTH[4:0] ≦ 31 (data words from 17-bit to 32-bit wide)

Bit 7 – CRCFULL FIFO Full bit

Value	Description
1	FIFO is full

Value	Description
0	FIFO is not full

Bit 6 – CRCEMPTY FIFO Empty bit

Value	Description
1	FIFO empty
0	FIFO not empty

Bit 5 – CRCISEL CRC Interrupt Selection bit

Value	Description
1	Interrupt on FIFO empty; final word of data still shifting through CRC
0	Interrupt on shift complete and results ready

Bit 4 – CRCGO Start CRC bit

Value	Description
1	Start CRC serial shifter; clearing the bit aborts shifting
0	CRC serial shifter turned off

Bit 3 – LENDIAN Little Endian Enable bit

Value	Description
1	Data word is shifted into the CRC starting with the LSb (little endian)
0	Data word is shifted into the CRC starting with the MSb (big endian)

Bit 2 – MOD Accumulator Mode bit

Value	Description
1	Accumulator configured for $x^{16} + x^{12} + x^5 + 1$ (when MOD = 1)
0	Accumulator configured for $x^{16} + x^{12} + x^5 + 1$ (when MOD = 0)

27.2.2 CRC XOR Register

Name: CRCXOR
Offset: 0x2CC

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	X[30:23]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	X[22:15]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	X[14:7]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	X[6:0]							X
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

Bits 31:1 – X[30:0] XOR of Polynomial Term X^N Enable bits

Bit 0 – X XOR of Polynomial Term X^N Enable bits
Note: Bitfield is read only.

27.2.3 CRC Data Register

Name: CRCDAT
Offset: 0x2D0

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 31:0 – DATA[31:0] CRC Input Data bits
Writing to this register fills the FIFO. Reading from this register returns '0'

27.2.4 CRC Shift Register

Name: CRCWDAT
Offset: 0x2D4

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	CRCWDAT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CRCWDAT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCWDAT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CRCWDAT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CRCWDAT[31:0] CRC Shift Register bits
Writing to this register writes to the CRC shifter register through the CRC write bus. Reading from this register reads the CRC read bus

27.3 CRC Operation

27.3.1 Polynomial Interface

The CRC module can be programmed for CRC polynomials of up to the 32nd order using up to 32 bits. Polynomial length, which reflects the highest exponent in the equation, is selected by the PLEN[4:0] bits (CRCCON[20:16]). The CRCXOR registers control which exponent terms are included in the equation. Setting a particular bit includes that exponent term in the equation functionally; this includes an XOR operation on the corresponding bit in the CRC engine. Clearing the bit disables the XOR.

For example, consider two CRC polynomials, one a 16-bit equation and the other a 32-bit equation. To program these polynomials into the CRC generator, set the register bits as shown in Table 27-1.

$x^{16} + x^{12} + x^5 + 1$ and

and

$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Table 27-1. CRC Setup Examples for 16 and 32-Bit Polynomials

CRC Control Bits	Bit Values	
	16-Bit Polynomial	32-Bit Polynomial
PLEN[4:0]	01111	11111
X[31:16]	0000 0000 0000 0000	0000 0100 1100 0001

.....continued		
CRC Control Bits	Bit Values	
	16-Bit Polynomial	32-Bit Polynomial
X[15:1]	0001 0000 0010 0001	0001 1101 1011 0111

Note that the appropriate positions are set to '1' to indicate that they are used in the equation (e.g., X26 and X23). The Most Significant bit (MSb) of the polynomial does not affect the calculation and can be set to any value.

27.3.2 Data Shift Direction

The LENDIAN bit (CRCCON[3]) is used to control the shift direction. By default, the CRC module will shift data through the engine, MSb first (LENDIAN = 0). Setting LENDIAN to '1' causes the CRC module to shift data, LSb first. This setting allows better integration with various communication schemes and removes the overhead of reversing the bit order in software. Note that this only changes the direction the data is shifted into the engine. The result of the CRC calculation will still be a normal CRC result, not a reverse CRC result.

dsPIC33A devices are little-endian. When the CRC module is configured for the big-endian (LENDIAN = 0), the input data bytes and words must be swapped in the application code before loading them into the data FIFO (CRCDAT registers).

27.3.3 Data FIFO

The module incorporates a FIFO that works with a variable data width. The data width is defined by the DWIDTH[4:0] bits (CRCCON[28:24]). It can be configured to any value between 1 and 32 bits. The logic associated with the FIFO contains a 5-bit counter, VWORD[4:0] bits (CRCCON[12:8]).

The value in the VWORD[4:0] bits indicates the number of unprocessed data elements in the FIFO. The FIFO is:

- 16-word deep when DWIDTH[4:0] ≤ 7 (data words, 8-bit wide or less)
- 8-word deep when DWIDTH[4:0] ≤ 15 (data words from 9 to 16-bit wide)
- 4-word deep when DWIDTH[4:0] ≤ 31 (data words from 17 to 32-bit wide)

The data for the CRC calculation must be written into the FIFO using the CRCDAT registers. Reading the CRCDAT registers always returns zero. To accommodate the MSb first shift method (LENDIAN = 0), byte and word swapping must be done in software when filling the FIFO.

Note: Ensure that the new data is not written into the CRCDAT registers when the CRCFUL bit is set; if the new data is written, it will be ignored.

When all shifts are done (i.e., the FIFO is empty and the CRC shift engine is Idle), it is possible to change the FIFO width (DWIDTH[4:0] bits) without any information loss or CRC result damage.

With a data width of eight bits or less, the FIFO increments on a write to the lower byte of the CRCDAT register (a byte access to the CRCDAT register must be used). The smallest data element that can be written into the FIFO is one byte.

For example, if DWIDT[20:16] is five, then the size of the data is DWIDT[20:16] + 1 or six. The data is written as a whole byte; the two unused upper bits are ignored. Once the data byte is written into the CRCDAT register, the value of the VWORD[4:0] bits (CRCCON[12:8]) increments by one.

With data widths more than 8 bits and less than or equal to 16 bits, the FIFO increments on a write to the CRCDAT register (16-bit word access to the CRCDAT register must be used). Unused upper data bits are ignored. The value of the VWORD[4:0] bits is incremented for every write to the CRCDAT register.

When the data width is greater than 16 bits, any write to the CRCDAT register increments the VWORD[4:0] bits by one. Writing the lower word into the CRCDAT register must be done before writing the upper word into the CRCDAT register. Unused upper data bits are ignored.

27.3.4 CRC ENGINE

27.3.4.1 Generic CRC Engine

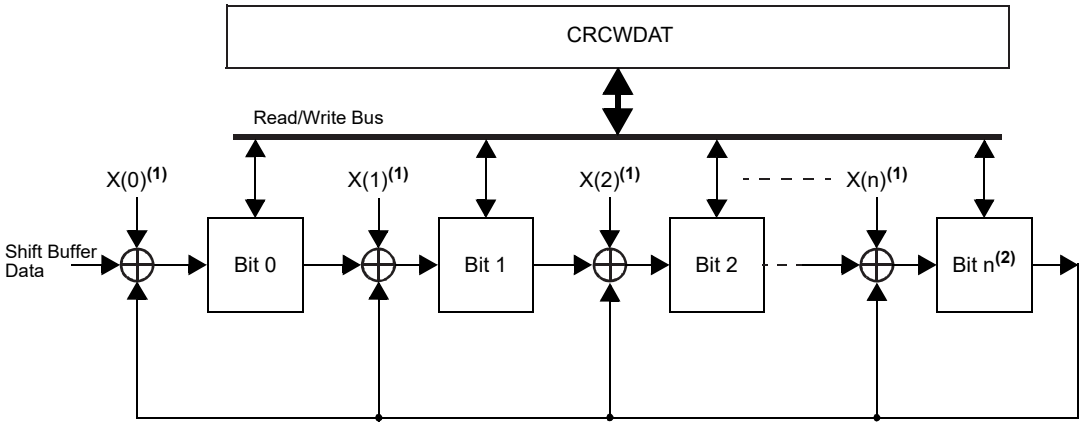
The CRC engine is a serial shifting CRC calculator that is configurable through multiplexer settings. The engine can also be configured to where shift buffer data is introduced using the MOD bit (CRCCON[2]). A simplified diagram of the CRC shift engine is shown in Figure 27-2.

The CRC algorithm uses a simplified form of arithmetic process using the XOR operation instead of binary division. The coefficients of the generator polynomial are programmed with the CRCXOR registers. Writing a '1' into a location enables XORing of that element in the polynomial. The length of the polynomial is programmed using the PLEN[4:0] bits in the CRCCON register (CRCCON[20:16]). The value of PLEN[4:0] signals the length of the polynomial and switches a multiplexer to indicate the tap from which the feedback originated.

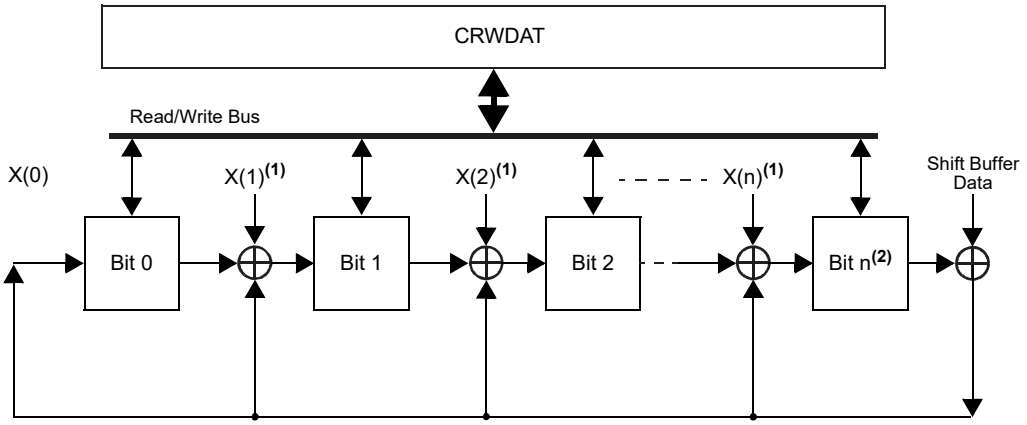
The result of the CRC calculation is obtained by reading the CRCWDAT registers.

Figure 27-2. CRC Shift Engine Detail

Legacy Mode (MOD bit = 0)



Alternate Mode (MOD bit = 1)



Notes:

1. Each XOR stage of the shift engine is programmable.
2. Polynomial Length n is determined by (PLEN[4:0] + 1).

27.3.4.2 CRC Engine Interface

27.3.4.3 FIFO to CRC Shift Engine

To start moving the data from the FIFO to the CRC shift buffer, the CRCGO bit (CRCCON[4]) must be set. The serial shifter starts shifting data from the shift buffer to the CRC shift engine, starting from the MSb first for LENDIAN = 0 and LSb first for LENDIAN = 1, when CRCGO = 1 and the value of VWORD[4:0] is greater than zero. If the CRCFUL bit was set earlier, then it is cleared when the VWORDx bits decrement by one. The VWORD[4:0] bits decrement by one when a FIFO location is moved to the shift buffer. The serial shifter continues shifting until the VWORD[4:0] bits reach zero; at this point, the CRCEMPTY bit becomes set to indicate that the FIFO is empty. If the CRCGO bit is cleared during a CRC calculation, then the CRC shift engine will stop calculating until the CRCGO bit is set.

The application can write into the FIFO while the shift operation is in progress. The CRCFULL bit should be monitored. If the CRCFULL bit is not set, another word can be written into the FIFO. At least one instruction cycle must pass after a write to the CRCDAT registers before a read of the valid value of the VWORD[4:0] bits.

When the VWORD[4:0] bits reach the maximum value for the configured value of the DWIDTH[4:0] bits, the CRCFULL bit becomes set. When the VWORD[4:0] bits reach zero, the CRCEMPTY bit becomes set. The FIFO is emptied and the VWORD[4:0] bits are set to '00000' whenever the ON bit is '0'.

27.3.4.4 Number of Clock Cycles to Shift Data

The data from FIFO goes to the shift buffer. It takes two peripheral clock cycles to start moving the data words from FIFO to the shift buffer. The data from the shift buffer is then shifted to the CRC shift engine. It takes (DWIDTH[4:0] + 1) clock cycles to completely move the data from the shift buffer to the CRC shift engine. For example, if DWIDTH[4:0] = 5, then the data length is six bits (DWIDTH[4:0] + 1) and six cycles are required to shift the data. In this case, only six bits of a byte are shifted out. The two MSbs of each byte are don't care bits. Similarly, for a 12-bit polynomial selection, the Most Significant four bits of each word are ignored.

27.3.4.5 CRC Initial Value

The access to the CRC shift engine is provided through the CRCWDAT registers. These registers can be loaded with a desired CRC initial value prior to the start of the calculations. The form of this initial value depends on the operating mode selected by the MOD bit (CRCCON[2]).

In Alternate mode (MOD bit = 1, not available on all devices), the CRC initial value must be in direct form.

In Legacy mode (MOD bit = 0), the CRC initial value must be in non-direct form. The non-direct initial value is a value for which the CRC calculation gives the desired direct CRC initial value. For example, if the application uses CRC-32 polynomial, 0x04C11DB7, and must start the calculations from the CRC direct initial value, 0xFFFFFFFF, then the non-direct value, 0x46AF6449, must be loaded in the CRCWDAT registers (the CRC of this non-direct value, 0x46AF6449, is 0xFFFFFFFF). When the non-direct initial value is written into the shift engine using the CRCWDAT registers, it will be converted by the CRC module to the direct initial value after (PLEN[4:0] + 1) peripheral clock cycles.

Note: The write to the CRCWDAT registers clears/resets the shift buffer.

Usually, the CRC calculation starts from the same initial value every time. In this case, the non-direct initial value can be found just once and then can be defined as a constant in the application code.

Note: The CRC non-direct initial value of zero is zero.

Example 27-1 shows a possible software routine to get the non-direct initial value from the direct initial value.

The CRC module can be used to get the non-direct initial value. To do this:

1. Enable the CRC module (ON = 1) and shifts (CRCGO = 1).
2. Shift the polynomial value right by one.
3. Reverse the bit order of the shifted polynomial value.
4. Write this result in the CRCXOR registers.
5. Set the data width and polynomial length (DWIDTH[4:0] and PLEN[4:0] bits) to the polynomial order (length).
6. Reverse the bit order of the desired direct initial value.
7. Write the reversed initial value in the CRCWDAT registers.
8. Write a dummy data to the CRCDAT registers and wait two peripheral clock cycles to move the data from the FIFO to the shift buffer and (PLEN[4:0] + 1) peripheral clock cycles to shift out the result.
   Alternatively, clear the CRC Interrupt Selection bit (CRCISEL = 0) to get the interrupt when shifts from the shift buffer are done, clear the CRC interrupt flag, write dummy data in the CRCDAT registers and wait for the CRC interrupt flag to set.
9. Read the value from the CRCWDAT registers.
10. Reverse the bit order of the read result; it will give the final non-direct initial value.

Example 27-2 shows one way to implement this procedure.

To continue calculations of the full data message in the applications where the intermediate CRC sums must be read in the middle of the calculations, the non-direct value must be calculated and set to the CRCWDAT registers again. In this case, the CRC direct initial value will be an intermediate CRC result read.

Example 27-1. Software Routine to Calculate the Non-Direct Initial Value

```
unsigned int CalculateNonDirectSeed(□
unsigned int seed,                        // direct CRC initial value
unsigned int polynomial,                  // polynomial
unsigned char polynomialOrder)            // polynomial order
{
unsigned char lsb;
unsigned char i;
unsigned int msbmask;

    msbmask = ((unsigned int)1)<<(polynomialOrder-1);
    for (i=0; i<polynomialOrder; i++) {
        lsb = seed & 1;
        if (lsb) seed ^= polynomial;
        seed >>= 1;
        if (lsb) seed |= msbmask;
}
return seed;                              // return the non-direct CRC initial
value}
```

Example 27-2. Calculating the Non-Direct Initial Value (MOD bit = 0)

```
unsigned int CalculateNonDirectSeed(unsigned int seed,  // direct CRC initial value
          unsigned int polynomial,                      // polynomial
unsigned char polynomialOrder)                           // polynomial order (valid
values are□
                                                         // 8, 16, 32 bits)
{
    CRCCON1 = 0;
    CRCCON2 = 0;
    CRCCON1bits.CRCEN= 1;                                // enable CRC
```

```
    CRCCON1bits.CRCISEL= 0;                                     // interrupt when all shifts
are done
    CRCCON2bits.DWIDTH= polynomialOrder-1;                      // data width
    CRCCON2bits.PLEN= polynomialOrder-1;                        // polynomial length
    CRCCON1bits.CRCGO= 1;                                       // start CRC calculation

    polynomial >>= 1;                                           // shift the polynomial right
    polynomial = ReverseBitOrder(polynomial, polynomialOrder);  // reverse bits order of
the polynomial

    CRCXOR = polynomial;                                        // set the reversed polynomial
    seed = ReverseBitOrder(seed, polynomialOrder);              // reverse bits order of the
seed value
    CRCWDAT = seed;


    _CRCIF = 0;                                                 // clear interrupt flag
    switch(polynomialOrder)                                     // load dummy data to shift
out the
                                                                // seed result
    {
    case 8:
       *((unsigned char*)&CRCDAT) = 0;                          // load byte
        while(!_CRCIF);                                         // wait until shifts are done
        seed = CRCWDAT&0x00ff;                                  // read reversed seed
    case 16: CRCDAT = 0;                                        // load short
        while(!_CRCIF);                                         // wait until shifts are done
        seed = CRCWDAT;                                         // read reversed seed
        break;
    case 32:
    // load long
    CRCDAT = 0;
    while(!_CRCIF);                                             // wait for shifts are done
    seed = CRCWDAT;                                             // read reversed seed
    break;
  default:
     ;
    }

    seed = ReverseBitOrder(seed, polynomialOrder);              // reverse the bit order to
get the
                                                                // non-direct seed
    return seed;                                                // return the non-direct CRC
initial value}
// WHERE THE FUNCTION TO REVERSE THE BIT ORDER CAN BE

unsigned int ReverseBitOrder(unsigned int data,                 // input data
              unsigned char numberOfBits)                       // width of the input data,
                                                                // valid values are 8,16,32
bits
{
unsignedintmaskin = 0;
unsignedintmaskout = 0;
unsignedintresult = 0;
unsignedchari;

    switch(numberOfBits)
    {
        case 8:
            maskin = 0x80;
            maskout = 0x01;
            break;
        case 16:
            maskin = 0x8000;
            maskout = 0x0001;
            break;
        case 32:
            maskin = 0x80000000;
            maskout = 0x00000001;
            break;
        default:
            ;
    }
    for(i=0; i<numberOfBits; i++)
    {
        if(data&maskin){
            result |= maskout;
        }
            maskin >>= 1;
            maskout <<= 1;
        }
        return result;
}
```

27.3.4.6 CRC Result

Reading the result of a CRC calculation depends on the selected operating mode.

In Alternate mode (MOD bit = 1 not available on all devices), the result is available in the CRCWDAT registers when all the data in the CRC FIFO buffer has been processed. Submitting dummy data to generate extra cycles is not required.

In Legacy mode (MOD bit = 0), the CRC module requires (PLEN[4:0] + 1) extra peripheral clock cycles to finish the calculations. To generate these additional cycles, the dummy data, with the width equal to the polynomial order (length), must be loaded into the CRCDAT registers. After the shifts are finished, the final CRC result can be read from the CRCWDAT registers.

To get the final CRC result after all data are loaded into the CRC module follow the procedure below..

If the data width (DWIDTH[4:0]) is more than the polynomial length (PLEN[4:0]):

1. Wait for the data FIFO to empty (CRCEMPTY bit is set).
2. Wait (DWIDTH[4:0] + 1) clock cycles to make sure that shifts from the shift buffer are finished.
3. Change the data width to the polynomial length (DWIDTH[4:0] = PLEN[4:0]).
4. Write one dummy data word to the CRCDAT registers.
5. Wait two peripheral clock cycles to move the data from the FIFO to the shift buffer, plus (PLEN[4:0] + 1) clock cycles to shift out the result.

   Alternatively, clear the CRC Interrupt Selection bit (CRCISEL = 0) to get the interrupt when all shifts are done. Clear the CRC interrupt flag. Write dummy data in the CRCDAT registers and wait until the CRC interrupt flag is set.
6. Read the final CRC result from the CRCWDAT registers.
7. Restore the data width (DWIDTH[4:0] bits) for further calculations (optional).

   If the data width (DWIDTH[4:0]) is equal to, or less than, the polynomial length (PLEN[4:0]), the procedure to get the result can be different.
8. Clear the CRC Interrupt Selection bit (CRCISEL = 0) to get the interrupt when all shifts are done.
9. Suspend the calculation by setting CRCGO = 0.
10. Clear the CRC interrupt flag.
11. Write the dummy data with the total data length equal to the polynomial length in the CRCDAT registers.
12. Resume the calculation by setting CRCGO = 1.
13. Wait until the CRC interrupt flag is set.
14. Read the final CRC result from the CRCWDAT registers.

    When the CRC result is achieved, the CRC non-direct initial value should be written again into the CRCWDAT registers to clear/reset the shift buffer from the previously loaded dummy data to start a new calculation. Example 27-3 shows the steps described above for the polynomial orders of 8, 16 and 32 bits.

Example 27-3. Routine to Get the Final CRC Result in Legacy Mode (MOD bit = 0)

```
unsigned int GetCRC(unsigned char polynomialOrder,   // valid values are 8,16,32
unsigned char currentDataWidth)                          // valid values are 8,16,32
{
unsignedintcrc = 0;

   while(!CRCCON1bits.CRCMPT);                           // wait until data FIFO is empty

   asmvolatile("repeat %0\n nop" : : "r"(currentDataWidth>>1)); // wait until previous
data
                                                         // shifts are done
CRCCON2bits.DWIDTH = polynomialOrder-1;                  // set data width to polynomial
length
CRCCON1bits.CRCISEL = 0;                                 // interrupt when all shifts are
done
```

```
_CRCIF = 0;                                             // clear interrupt flag

switch(polynomialOrder)
   {
      case 8:                                           // polynomial length is 8 bits
         *((unsigned char*)&CRCDAT) = 0;                // load byte
         while(!_CRCIF);                                // wait until shifts are done
         crc = CRCWDATL&0x00ff;                         // get crc
         break;
      case 16:                                          // polynomial length is 16 bits
         CRCDAT = 0;                                    // load short
         while(!_CRCIF);                                // wait until shifts are done
         crc = CRCWDAT;                                 // get crc
         break;
      case 32:                                          // polynomial length is 32 bits
         CRCDAT = 0;
         while(!_CRCIF);                                // wait until shifts are done
         crc = CRCWDAT;                                 // get crc
         break;
      default:
         ;
   }
   CRCCON2bits.DWIDTH = currentDataWidth-1;             // restore data width for
further
                                                        // calculations
   return crc;                                          // return the final CRC value
}
```

27.3.5 Interrupts

The module generates an interrupt that is configurable by the user for either of the two conditions. If CRCISEL is '1', an interrupt is generated when the VWORD[4:0] bits make a transition from a value of '1' to '0'. If CRCISEL is '0', an interrupt will be generated when the FIFO is empty and shifts from the shift buffer are finished.

27.4 Application Examples

The CRC is a robust error checking algorithm in digital communication for messages containing several bytes or words. After calculation, the checksum is appended to the message and transmitted to the receiving station. The receiver calculates the checksum with the received message to verify the data integrity.

27.4.1 Variations

The 32-bit programmable CRC module can be programmed to shift out either the MSb or LSb first. MSb first is a popular implementation as employed in XMODEM protocol. In one of the variations (CCITT protocol) for CRC calculation, the LSb is shifted out first. Discussions on all the variations are beyond the scope of this document, but several variations of CRC can be implemented using this module.

The choice of the polynomial length and the polynomial itself are application dependent. Polynomial lengths of 5, 7, 8, 10, 12, 16 and 32 are normally used in various standard implementations. The following sections explain the recommended step-by-step procedure for CRC calculation. Users can decide whether zeros, or any other values, need to be appended to the message stream. Depending on the application, the user may decide whether any value needs to be appended at all.

27.4.2 Typical Operations

To use the module for a typical CRC calculation:

1. Set the CRCEN bit to enable the module.
2. Configure the module for the desired operation:
   a. Program the desired polynomial using the CRCXOR registers and the PLEN<4:0> bits.
   b. Configure the data width and shift direction using the DWIDTH<4:0> and LENDIAN bits.
3. Set the CRCGO bit to start the calculations.
4. Set the desired CRC initial value in the CRCWDAT registers as described in 27.3.4.5. CRC Initial Value.

5. Load all data into the FIFO by writing to the CRCDAT registers as space becomes available (the CRCFUL bit must be zero before the next data loading).
6. Wait until the data FIFO is empty (CRCMPT bit is set).
7. Read the CRC result as described in 27.3.4.6. CRC Result.

Example 27-4 through Example 27-7 show typical code for different combinations of polynomial length, data width, shift direction and CRC Engine modes.

Example 27-4. CRC-SMBus (8-Bit Polynomial with 32-Bit Data, Big-Endian, MOD bit = 1)

```
// This macro is used to swap bytes for big endian
#define            Swap(x) __extension__({ \
unsigned            int __x = (x), __v; \
__asm__ ("wsbh %0,%1;\n\t" \
"rotr %0,16" \
: "=d" (__v) \
: "d" (__x)); \
__v; \
})
// ASCII            bytes "12345678"
volatile            unsigned char __attribute__((aligned(4))) message[] =
{'1','2','3','4','5','6','7','8'};
volatile            unsigned char crcResultCRCSMBUS = 0;
int            main    (void)
{
unsigned            int*    pointer;
unsigned            short      length;
unsigned            int     data;
// standard CRC-SMBUS
#define            CRCSMBUS_POLYNOMIAL ((unsigned int)0x00000007)
#define            CRCSMBUS_SEED_VALUE ((unsigned int)0x00000000)            // direct
initial value
CRCCON = 0;
CRCCONbits.MOD = 1;                                                  // alternate mode
CRCCONbits.ON  = 1;                                                  // enable CRC
CRCCONbits.LENDIAN = 0;                                              // big endian
CRCCONbits.CRCISEL = 0;                                              // interrupt when all
shifts are done
CRCCONbits.DWIDTH  = 32-1;                                              // 32-bit data width
CRCCONbits.PLEN    = 8-1;                                               // 8-bit polynomial order
CRCXOR  = CRCSMBUS_POLYNOMIAL;                                          // set polynomial
CRCWDAT = CRCSMBUS_SEED_VALUE;                                          // set initial value
CRCCONbits.CRCGO   = 1;                                                 // start CRC calculation
pointer = (unsigned int*)message;
length  = sizeof(message)/sizeof(unsigned int);
while(1)
{
while(CRCCONbits.CRCFUL);                                               // wait if FIFO is full
data = *pointer++;                                                   // load from little endian
data = Swap(data);                                                   // swap bytes for big endian
length--;
if(length == 0)
{
break;
}
CRCDAT = data;                                                  // 32-bit word access to FIFO
}
CRCCONbits.CRCGO = 0;                                                   // suspend CRC calculation
IFS0CLR = _IFS0_CRCIF_MASK;                                             // clear the interrupt
flag
CRCDAT  = data;                                                 // write last data into FIFO
CRCCONbits.CRCGO = 1;                                                   // resume CRC calculation
while(!IFS0bits.CRCIF);                                                 // wait until shifts are
done
crcResultCRCSMBUS = (unsigned char)CRCWDAT&0x00ff;                                  // get CRC
result (must be 0xC7)
while(1);
return 1;
}
```

Example 27-5. CRC-16 (16-Bit Data with 16-Bit Polynomial, Little-Endian, MOD bit = 1)

```
// ASCII bytes "87654321"
volatile            unsigned        short message[] = {0x3738,0x3536,0x3334,0x3132};
volatile            unsigned        short crcResultCRC16 = 0;
int            main        (void)
{
unsigned            short*        pointer;
unsigned            short        length;
```

```
unsigned            short        data;
// standard CRC-16
#define             CRC16_POLYNOMIAL ((unsigned int)0x00008005)
#define             CRC16_SEED_VALUE ((unsigned int)0x00000000)              // direct
initial value
CRCCON = 0;
CRCCONbits.MOD = 1;                                                 // alternate mode
CRCCONbits.ON  = 1;                                                 // enable CRC
CRCCONbits.CRCISEL = 0;                                                // interrupt when all
shifts are done
CRCCONbits.LENDIAN = 1;                                                // little endian
CRCCONbits.DWIDTH  = 16-1;                                             // 16-bit data width
CRCCONbits.PLEN    = 16-1;                                             // 16-bit polynomial
order
CRCXOR  = CRC16_POLYNOMIAL;                                               // set polynomial
CRCWDAT = CRC16_SEED_VALUE;                                               // set initial value
CRCCONbits.CRCGO = 1;                                                  // start CRC calculation
pointer = (unsigned short*)message;
length = sizeof(message)/sizeof(unsigned short);
while(1)
{
while(CRCCONbits.CRCFUL);                                              // wait if FIFO is full
data = *pointer++;                                                  // load data
length--;
if(length == 0)
{
break;
}
*((unsigned short*)&CRCDAT) = data;                                          // 16-bit word
access to FIFO
}
CRCCONbits.CRCGO = 0;                                                // suspend CRC calculation
IFS0CLR = _IFS0_CRCIF_MASK;                                             // clear the interrupt
flag
*((unsigned short*)&CRCDAT) = data;                                          // write last data
into FIFO
CRCCONbits.CRCGO = 1;                                                // resume CRC calculation
while(!IFS0bits.CRCIF);                                              // wait until shifts are
done
crcResultCRC16 = (unsigned short)CRCWDAT;                                  // get CRC result
(must be 0xE716)
while(1);
return 1;
}
```

Example 27-6. CRC-32 (32-Bit Polynomial with 32-Bit Data, Little-Endian, MOD bit = 1)

```
// ASCII bytes "12345678"
volatile            unsigned        char __attribute__((aligned(4))) message[] =
{'1','2','3','4','5','6','7','8'};
// function to reverse the bit order (OPTIONAL)
unsigned            int         ReverseBitOrder(unsigned int data);
volatile            unsigned        int crcResultCRC32 = 0;
int           main(void)
{
unsigned            int*        pointer;
unsigned            short        length;
// standard CRC-32
#define             CRC32_POLYNOMIAL ((unsigned int)0x04C11DB7)
#define             CRC32_SEED_VALUE ((unsigned int)0xFFFFFFFF)             // direct
initial value
CRCCON = 0;
CRCCONbits.MOD = 1;                                                 // alternate mode
CRCCONbits.ON  = 1;                                                 // enable CRC
CRCCONbits.CRCISEL = 0;                                                // interrupt when all
shifts are done
CRCCONbits.LENDIAN = 1;                                                // little endian
CRCCONbits.DWIDTH  = 32-1;                                             // 32-bit data width
CRCCONbits.PLEN    = 32-1;                                             // 32-bit polynomial
order
CRCXOR  = CRC32_POLYNOMIAL;                                               // set polynomial
CRCWDAT = CRC32_SEED_VALUE;                                               // set initial value
CRCCONbits.CRCGO = 1;                                                  // start CRC calculation
pointer = (unsigned int*)message;
length  = sizeof(message)/sizeof(unsigned int);
while(1)
{
while(CRCCONbits.CRCFUL);                                              // wait if FIFO is full
length--;
if(length == 0)
{
break;
}
CRCDAT = *pointer++;                                                // 32-bit word access to
FIFO
```

```
}
CRCCONbits.CRCGO = 0;                                          // suspend CRC calculation
IFS0CLR = _IFS0_CRCIF_MASK;                                       // clear the interrupt
flag
CRCDAT  = *pointer;                                            // write last data into FIFO
CRCCONbits.CRCGO = 1;                                          // resume CRC calculation
while(!IFS0bits.CRCIF);                                        // wait until shifts are
done
crcResultCRC32 = CRCWDAT;                                         // get the final CRC
result
// OPTIONAL reverse CRC value bit order and invert (must be 0x9AE0DAAF)
crcResultCRC32 = ~ReverseBitOrder(crcResultCRC32);
while(1);
return 1;
}
unsigned            int ReverseBitOrder(unsigned int data)
{
unsigned            int         maskin;
unsigned            int         maskout;
unsigned            int         result = 0;
unsigned            char i;
maskin  = 0x80000000;
maskout = 0x00000001;
for(i=0; i<32; i++)
{
if(data&maskin){
result |= maskout;
}
maskin >>= 1;
maskout <<= 1;
}
return            result;
}
```

Example 27-7. Data Width Switching (32-Bit Polynomial, Little-Endian, MOD bit = 1)

```
// ASCII bytes "12345678"
volatile            unsigned        int message1[] = {0x34333231,0x38373635};
// ASCII bytes "123"
volatile            unsigned        char message2[] = {'1','2','3'};
volatile            unsigned        int crcResultCRC32 = 0;
int            main(void)
{
unsigned            char*        pointer8;
unsigned            int*         pointer32;
unsigned            short        length;
#define            CRC32_POLYNOMIAL ((unsigned int)0x04C11DB7)
#define            CRC32_SEED_VALUE ((unsigned int)0xFFFFFFFF)    // direct initial value
CRCCON = 0;
CRCCONbits.MOD = 1;                                       // alternate mode
CRCCONbits.ON  = 1;                                       // enable CRC
CRCCONbits.CRCISEL = 0;                                   // interrupt when all shifts are
done
CRCCONbits.LENDIAN = 1;                                   // little endian
CRCCONbits.DWIDTH  = 32-1;                                   // 32-bit data width
CRCCONbits.PLEN    = 32-1;                                   // 32-bit polynomial order
CRCXOR  = CRC32_POLYNOMIAL;                                  // set polynomial
CRCWDAT = CRC32_SEED_VALUE;                                  // set initial value
CRCCONbits.CRCGO = 1;                                     // start CRC calculation
pointer32 = (unsigned int*)message1;
length    = sizeof(message1)/sizeof(unsigned int);
while(1)
{
while(CRCCONbits.CRCFUL);                                    // wait if FIFO is full
length--;
if(length == 0)
{
break;
}
CRCDAT = *pointer32++;                                    // 32-bit word access to FIFO
}
CRCCONbits.CRCGO = 0;                                     // suspend CRC calculation
IFS0CLR = _IFS0_CRCIF_MASK;                                  // clear the interrupt flag
CRCDAT  = *pointer32;                                     // write last 32-bit data into FIFO
CRCCONbits.CRCGO = 1;                                     // resume CRC calculation
while(!IFS0bits.CRCIF);                                   // wait until shifts are done
CRCCONbits.DWIDTH = 8-1;                                     // switch the data width to 8-bit
pointer8 = (unsigned char*)message2;                            // calculate CRC
length   = sizeof(message2)/sizeof(unsigned char);
while(length--)
{
while(CRCCONbits.CRCFUL);                                    // wait if FIFO is full
length--;
if(length == 0)
{
```

```
break;
}
*((unsigned char*)&CRCDAT) = *pointer8++;                       // byte access to FIFO
}
CRCCONbits.CRCGO = 0;                                   // suspend CRC calculation
IFS0CLR = _IFS0_CRCIF_MASK;                                // clear the interrupt flag
*((unsigned char*)&CRCDAT) = *pointer8;                          // write last 8-bit data
into FIFO
CRCCONbits.CRCGO = 1;                                   // resume CRC calculation
while(!IFS0bits.CRCIF);                                 // wait until shifts are done
crcResultCRC32 = CRCWDAT;                                  // get the final CRC result
(must be 0xE092727E)
while(1);
return 1;
}
```

27.5 CRC Operation in Power Saving Modes

27.5.1 Sleep Mode

If Sleep mode is entered while the module is operating, the module is suspended in its current state until clock execution resumes.

27.5.2 Idle Mode

To continue full module operation in Idle mode, the SIDL bit must be cleared prior to entry into the mode.

If SIDL = 1, the module behaves the same way as it does in Sleep mode; pending interrupt events will be passed on, even though the module clocks are not available.

28. Current Bias Generator (CBG)

The Current Bias Generator (CBG) module is a set of constant current sources that can be used for many purposes including pull up detection and offset correction. The module is made up of four independent 10 µA outputs and four independent selectable sources ranging from 0-200 µA. The two classes of current source share a physical output pin on the device.

28.1 Device-Specific Information

Output availability for each source variant is shown in Table 28-2 and specific output availability for selectable sources is shown in Table 28-4.

Table 28-1. CBG Summary Table

Module Instances	Number of Outputs	Peripheral Bus Speed
1	4	Slow

Table 28-2. Selectable and Fixed Current Source Availability

Source	Quantity	Type	Range
I10ENx (IBIAS)	4	Fixed	10 µA
ISELOUTx (ISRC)	4	Selectable	30-200 µA

Table 28-3. Channel Per Package Availability

Package Type	ISRCx	IBIASx
28-pin (SSOP and VQFN)	0,1,2	0,1,2
36-pin	0,1,2	0,1,2
48-pin	0,1,2,3	0,1,2,3
64-pin	0,1,2,3	0,1,2,3

Table 28-4. Selectable Source Options

Selectable Source Options
30
50
100
100
120
150
200

28.2 Architectural Overview

28.2.1 Module Description

The CBG module consists of two classes of current sources: four 10 µA sources and four selectable sources ranging from 30-200 µA. Both sets of current sources are routed to the same output pin.

Both sets of current sources can be used in general current sourcing applications to generate voltages using an external resistor or to provide biasing to external circuitry or sensors.

One intended use of the selectable current source is to generate an offset voltage to shift a positive external signal to be within the input range of the internal analog peripherals, such as the ADC. Shifting the input voltage maintains the dynamic range of the AC component of the input signal but removes the offset voltage. An external resistor (refer to Figure 28-2) is used in conjunction with the current source to develop the offset voltage.

The external resistors are large due to the small generated currents. This large resistor value protects the device input circuitry by limiting the current injected into the device when the current source is not enabled.

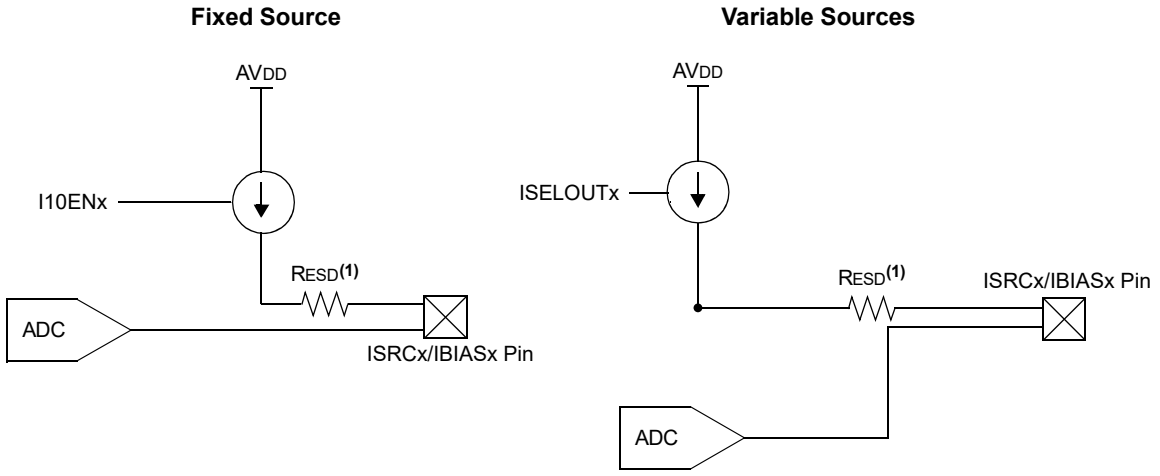
Both classes of current sources can be externally paralleled by connecting the output pins together to increase current.

The large resistors used to create the voltage offset may exceed the ADC input impedance specification. To meet the ADC input requirements, one or more of the following may be required:

- Increase in sampling time
- Use of an internal amplifier, such as an op amp or PGA
- Use of a small capacitor on the input pin if the input signal does not change quickly
- Use of an AC bypass capacitor

A high-level diagram of the CBG circuitry is shown in Figure 28-1.

Figure 28-1. Current Bias Generator Sources



Note:

1. R_{ESD} is typically 300 Ohms.

28.3 Current Bias Generator Control Register

Name: IBIASCON
Offset: 0x3AA4

Legend:R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	Reserved[11:4]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	Reserved[3:0]							Reserved
Access	R	R	R	R				R
Reset	0	0	0	0				0

Bit	15	14	13	12	11	10	9	8
	I10EN3	I10EN2	ISELOUT3[2:0]			ISELOUT2[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	I10EN1	I10EN0	ISELOUT1[2:0]			ISELOUT0[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:20 – Reserved[11:0]

Bit 16 – Reserved

Bit 15 – I10EN3 Enable for the ISRC3 Current Source bit

Value	Description
1	Current source is enabled
0	Current source is disabled

Bit 14 – I10EN2 Enable for the ISRC2 Current Source bit

Value	Description
1	Current source is enabled
0	Current source is disabled

Bits 13:11 – ISELOUT3[2:0] Current Output Selection for IBIAS3 bits

Value	Description
111	200 µA
110	150 µA
101	120 µA
100	100 µA
011	100 µA
010	50 µA
001	30 µA
000	0 µA

Bits 10:8 – ISELOUT2[2:0] Current Output Selection for IBIAS2 bits

Value	Description
111	200 µA
110	150 µA
101	120 µA
100	100 µA
011	100 µA
010	50 µA
001	30 µA
000	0 µA

Bit 7 – I10EN1 Enable for the ISRC1 Current Source bit

Value	Description
1	Current source is enabled
0	Current source is disabled

Bit 6 – I10EN0 Enable for the ISRC0 Current Source bit

Value	Description
1	Current source is enabled
0	Current source is disabled

Bits 5:3 – ISELOUT1[2:0] Current Output Selection for IBIAS1 bit

Value	Description
111	200 µA
110	150 µA
101	120 µA
100	100 µA
011	100 µA
010	50 µA
001	30 µA
000	0 µA

Bits 2:0 – ISELOUT0[2:0] Current Output Selection for IBIAS0 bit

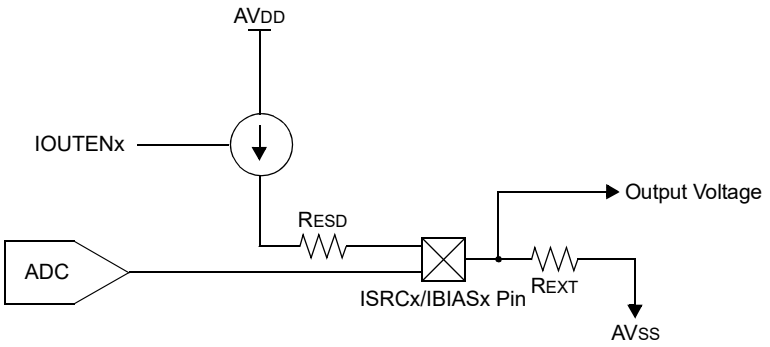
Value	Description
111	200 µA
110	150 µA
101	120 µA
100	100 µA
011	100 µA
010	50 µA
001	30 µA
000	0 µA

28.4 Operation

28.4.1 Basic Operation of the 10 µA Source

The primary application of this source is to generate current to create an external voltage. This voltage can then be measured with the internal ADC or used to bias external circuitry. This class of source can only supply (source) current. To generate an external voltage, an external resistor is connected between the current source pin and AV_{SS} (refer to Figure 28-2). The current flow generates a voltage across the R_EXT resistor (refer to Equation 28-1 and Example 28-1). Multiple sources can be paralleled, as needed, to increase current.

Figure 28-2. 10 µA Current Source



Equation 28-1. Equation for Determining the Value of R_EXT

$V(R_{EXT}) = 10\ \mu A \times R_{EXT}$

$V_{(REXT)}$ should not exceed AV_{DD} – 0.5V typical (see 28.4.3. Operating Range).

Example 28-1. Enabling a 10 µA Source

R_{EXT} = 10 kOhms, AV_{DD} = 3.3V V_{PIN} = 10k × 10 µA = 100 mv V_{REXT} << 3.3V – .5V, and therefore, meets the $V(R_{EXT})$ requirement

```
// User code to enable a 10 ua source
BIASCONbits.I10EN0 = 1;                // enable 10ua source channel 0
```

28.4.1.1 Device Pin ESD Configuration

Devices have a single ESD resistor on each pin (refer to Figure 28-2 and Figure 28-4).

28.4.2 Basic Operation of the 200 µA Source

The primary application of the 200 µA current source is to remove the DC offset so that the signal to be measured is within the ADC module's input range. Figure 28-3 shows a typical signal to be measured: an AC signal with a DC offset. This class of current source can be used to create a positive shift with an external resistor. The following examples show the basic configurations for shifting the input voltage and the required calculations. The equations in Equation 28-2 are used for positive voltage shift calculations.

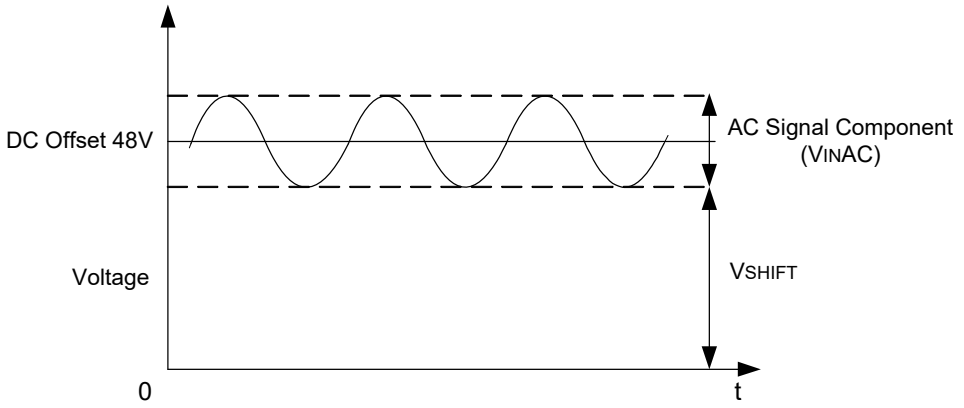
Equation 28-2. Equation for Determining the Value of R_{SHIFT}

$$R_{SHIFT} = \frac{V_{SHIFT}}{I\_SEL\ \mu A}$$

$$V_{SHIFT} = V_{INDC} - \left(\frac{V_{INAC}}{2}\right)$$

Note: $V(R_{SHIFT})$ should not exceed AV_{DD} – 0.7V typical (see 28.4.3. Operating Range).

Figure 28-3. AC Signal Component with a DC Offset



28.4.3 Operating Range

The maximum voltage that can be developed across a resistor driven by a current source depends on AV_{DD} and the other voltage sources in the circuit.

When the resistor is connected to AV_{SS}, such as seen in Figure 28-2, the maximum voltage that can be developed is approximately AV_{DD}. However, when the developed voltage is greater than the current source's internal threshold, the output current is reduced. To prevent this, the maximum developed voltage across $R_{ESD} + R_{EXT}$ should be limited to $AV_{DD} – 0.5V$ (typical), with respect to the output current value.

28.4.4 ADC Input Considerations

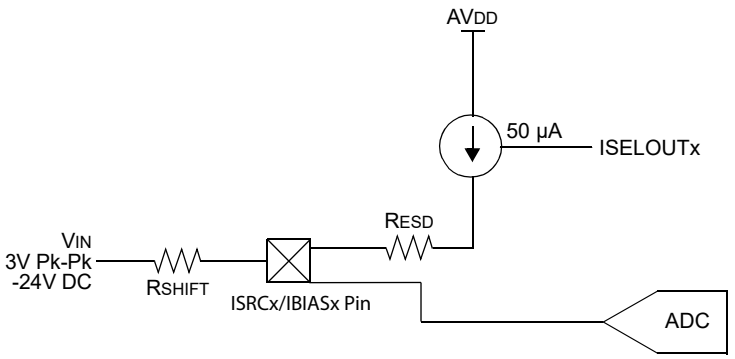
The input impedance for the ADC determines the required change time, specified in ADC clocks or Tad. The impedance consists of the following internal resistances: the ADC channel select switch as well as any external resistance. The large external resistor values required to generate offsets may violate the device's ADC input specifications. This may require the use of an internal amplifier op amp or PGA to isolate the ADC from the large resistance.

28.5 Application Examples

28.5.1 Voltage Shifting for a Positive Input Voltage

To shift a positive input voltage, a single CBG source is used. The source is used to generate a negative voltage to offset the input signal. Figure 28-4, Equation 28-3 and Example 28-2 show the calculations and configuration for this application.

Figure 28-4. Single-Ended Positive Voltage Shift



Equation 28-3. VSHIFT Calculations

VINMIN = – 24V – (3V/2) = – 25.5V

VINMAX = – 24V + (3V/2) = – 22.5V

VSHIFT = |VINMIN|

RSHIFT = 25.5V/50 µA = 510 kOhm

Shift with standard value resistor is *511k × 50 µA = 25.55V*

Input range = (VINMAX – VSHIFT) - (VINMIN – VSHIFT), (49.5V – 46.5V) - (46.5V – 46.5V) = 3V

Example 28-2. Single-Ended Positive Voltage Shift with 50 µA Selection

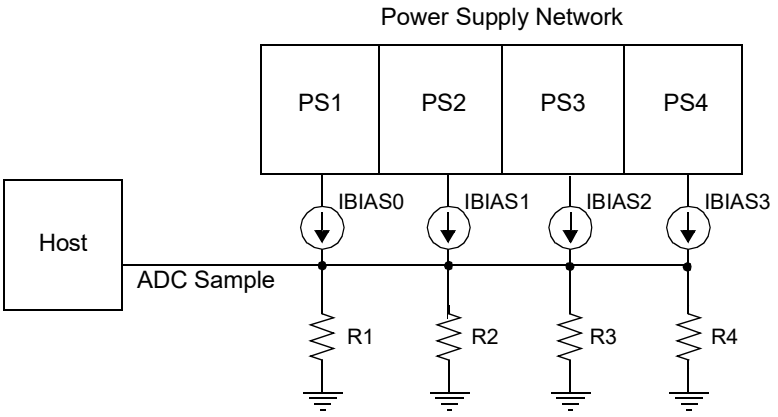
3V p-p Signal with a -24V Offset:

```
// sample code to enable 50 uA current sink.
BIASCONbits.ISELOUT1 = 0b010;    // set the 50 uA source on channel 1
```

28.5.2 Instrument Identification Assignment

Another example application of the CBG module is utilizing the 10 µA ISRC output channels to identify and assign an ID for multiple power supplies in a rack-mounted configuration. Typically, rack mounted power supplies use resistors mounted on a backplane to assign IDs to each power supply in the rack. The 10 µA ISRC pins feed these resistors to generate a voltage that is read by the ADC. The ADC value is translated via software into an identifier for each back plane slot.

Figure 28-5. Power Supply Network Identifier



28.6 Interrupts

The current source modules do not generate interrupts.

28.7 Operating in Power-Saving Modes

Both classes of current sources continue to operate in power save modes.

28.8 Effects of a Reset

A Reset forces module registers to their initial Reset values, disabling the current sources.

29. Operational Amplifier

An operational amplifier is a type of analog circuit that is designed to amplify the difference between the inputs non-inverting (positive) and inverting (negative) inputs. Op amps can be connected in Inverting and Non Inverting Applications and are commonly used to buffer or amplify signals, such as the signals from sensors, current shunts and resistive dividers. The op amp modules implemented in dsPIC33A devices provide the following features:

- 100 MHz Gain Bandwidth
- High-Power and Low-Power Operating Modes
- Complementary P and N Channel Differential Pairs on the Input
- User Adjustable Offset

29.1 Device-Specific Information

Table 29-1. Op Amp Summary Table

Op Amp Module Instances	Clock Input	Peripheral Bus Speed
Up to 3	None	Slow

Table 29-2. Op Amp Availability by Device Package

Package	Op Amp Availability
64-Pin	OA1, OA2, OA3
48-Pin	OA1, OA2, OA3
36-Pin	OA1, OA2, OA3
28-Pin	OA1, OA2

Note: The calibration registers FOPAMPLP and FOPAMPHP are placed at 0x007F20C0 and 0x007F20D0, respectively. The user can read this register to determine the factory calibrated trim settings.

29.2 Architectural Overview

Op amps allow the user to do signal conditioning without dedicated external circuitry. Op amps in the device support Unity Gain mode without any external connection. The user can adjust the gain of the op amp by using external resistors and disabling Unity Gain mode. The op amp has an offset trim feature to reduce the effects of input offsets. The input signal and output signal of the op amp can be monitored by enabling an internal connection to ADC.

While not in use, the user can disable the op amp by clearing the AMPEN bit to '0'. This reduces the power consumption.

29.3 Op Amp Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3AB0	AMP1CON1	31:24								
		23:16								
		15:8	AMPEN	HPEN	UGE	DIFFCON[1:0]			IMONEN	OMONEN
		7:0								
0x3AB4	AMP1CON2	31:24	POFFSETHP[4:0]							
		23:16	NOFFSETHP[4:0]							
		15:8	POFFSETLP[4:0]							
		7:0	NOFFSETLP[4:0]							
0x3AB8	AMP2CON1	31:24								
		23:16								
		15:8	AMPEN	HPEN	UGE	DIFFCON[1:0]			IMONEN	OMONEN
		7:0								
0x3ABC	AMP2CON2	31:24	POFFSETHP[4:0]							
		23:16	NOFFSETHP[4:0]							
		15:8	POFFSETLP[4:0]							
		7:0	NOFFSETLP[4:0]							
0x3AC0	AMP3CON1	31:24								
		23:16								
		15:8	AMPEN	HPEN	UGE	DIFFCON[1:0]			IMONEN	OMONEN
		7:0								
0x3AC4	AMP3CON2	31:24	POFFSETHP[4:0]							
		23:16	NOFFSETHP[4:0]							
		15:8	POFFSETLP[4:0]							
		7:0	NOFFSETLP[4:0]							

29.3.1 AMPx Control Register 1

Name: AMPxCON1
Offset: 0x3AB0, 0x3AB8, 0x3AC0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	AMPEN	HPEN	UGE	DIFFCON[1:0]			IMONEN	OMONEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 15 – AMPEN Op Amp Enable/On bit

Value	Description
1	Enables op amp module
0	Disables op amp module

Bit 14 – HPEN High-Power Enable bit

Value	Description
1	Enables Op Amp High-Power (high bandwidth) mode
0	Disables Op Amp High-Power mode

Bit 13 – UGE Unity Gain Buffer Enable bit

Value	Description
1	Enables Unity Gain mode
0	Disables Unity Gain mode

Bits 12:11 – DIFFCON[1:0] Differential Input Mode Control bits

Value	Description
11	Reserved, do not use
10	Turn NMOS differential input pair off
01	Turn PMOS differential input pair off
00	Use both NMOS and PMOS differential input pair

Bit 9 – IMONEN Enable Input Monitor bit

Value	Description
1	Enables positive input to ADC
0	Disables positive input to ADC

Bit 8 – OMONEN Enable Output Monitor bit

Value	Description
1	Enables output to ADC
0	Disables output to ADC

29.3.2 AMPx Control Register 2

Name: AMPxCON2
Offset: 0x3AB4, 0x3ABC, 0x3AC4

Notes:

1. Unit voltage = trim step voltage 3 mV
2. When the op amp is configured for gain = 1, and the non-inverting input is set to ~V_{DD}/2, positive values of the offset correction (5'b00001 - 5'b01111) reduce the voltage measured on the op amp output terminal. Negative values of the offset correction (5'b11110- 5'b10000) increase the voltage measured on the op amp output terminal.
3. If the op amp offset is +6 mV before correction, using an offset correction code of 00010 will change the output voltage by -6 mV, canceling the op amp offset.

Bit	31	30	29	28	27	26	25	24
				POFFSETHP[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
				NOFFSETHP[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				POFFSETLP[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				NOFFSETLP[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 28:24 – POFFSETHP[4:0] Offset Correction for PMOS Differential Input Pair (High-Power mode) bits
See descriptions for bits[4:0].

Bits 20:16 – NOFFSETHP[4:0] Offset Correction for NMOS Differential Input Pair (High-Power mode) bits
See descriptions for bits[4:0].

Bits 12:8 – POFFSETLP[4:0] Offset Correction for PMOS Differential Input Pair (Low-Power mode) bits
See descriptions for bits[4:0].

Bits 4:0 – NOFFSETLP[4:0] Offset Correction for NMOS Differential Input Pair (Low-Power mode) bits(1)

Value	Description
01111	Reduce output voltage by 15 unit voltage
. . .	
00111	Reduce output voltage by 7 unit voltage
00110	Reduce output voltage by 6 unit voltage
00101	Reduce output voltage by 5 unit voltage
00100	Reduce output voltage by 4 unit voltage
00011	Reduce output voltage by 3 unit voltage
00010	Reduce output voltage by 2 unit voltage

Value	Description
00001	Reduce output voltage by 1 unit voltage
00000	No correction from production calibration value
11111	No correction from production calibration value
11110	Increase output voltage by 1 unit voltage
11101	Increase output voltage by 2 unit voltage
11100	Increase output voltage by 3 unit voltage
11011	Increase output voltage by 4 unit voltage
11010	Increase output voltage by 5 unit voltage
11001	Increase output voltage by 6 unit voltage
11000	Increase output voltage by 7 unit voltage
. . .	
10000	Increase output voltage by 15 unit voltage

29.4 Operations

29.4.1 Enabling the Op Amp

The op amp has a per instance enable signal. This is controlled by the AMPEN bit in the AMPxCON1 register. Upon Reset, AMPEN is set to '0'. This keeps all of the op amp instances disabled upon Reset. To enable instance x of the op amp, set the AMPEN bit to '1' in the AMPxCON1 register.

29.4.2 Gain Settings

29.4.2.1 Unity Gain Setup

The op amp supports unity gain operation without any external connections. To configure the op amp in unity gain configuration, set the UGE bit to '1' in the AMPxCON1 register. In this mode, the op amp output is internally shorted to an inverting input and the resulting gain will be one.

Example 29-1. Op Amp Unity Gain Buffer

```
AMP1CON1bits.AMPEN = 1;     //Enable OP-AMP 1
AMP1CON1bits.UGE = 1;       //Enable unity gain mode
```

29.4.2.2 Gain Control Using External Resistor

The op amp supports non-unity gain configurations using an external resistor. To configure the op amp in this gain configuration, set the UGE bit to '0' in the AMPxCON1 register.

Example 29-2. Op Amp Fixed Gain Amplifier with External Resistor

```
AMP1CON1bits.AMPEN = 1;     //Enable OP-AMP 1
AMP1CON1bits.UGE = 0;       //Disable unity gain mode
```

29.4.3 Power Modes

The Op Amp Power mode is controlled using the HPEN bit in the AMPxCON1 register. When this bit is set, the op amp works in High-Power mode. When this bit is cleared, the op amp enters Low-Power mode. In Low-Power mode, the current consumption and bandwidth of the op amp are reduced.

29.4.4 Differential Input Mode

Differential Input Mode of OPAMP can be controlled by the DIFFCON bit in the AMPxCON1 register. The DIFFCON[1:0] bit field enables the user to disable operation of the PMOS and/or the NMOS input differential input pair of transistors in the amplifier. This choice of operation allows the customer to trade input voltage range for improved INL and offset voltage operation.

29.4.5 Enable Input Monitor and Output Monitor

This functionality of OPAMP is controlled by the IMONEN and OMONEN bits in the AMPxCON1 register.

The IMONEN control bit connects the non-inverting amplifier input to an ADC input. This allows the user to measure the op amp input voltage with the ADC.

Similarly, the bit connects the amplifier output to an input on the ADC. This allows the user to read the op amp output voltage on the ADC.

29.4.6 Input Offset Trim

It is possible for the user to override factory calibration values and perform user calibration for input referred voltage offset errors. The AMPxCON2 register is used for input offset which allows an adjustment to be added or subtracted from the factory offset trim value. The register allows the user to increase or decrease the offset trim up to the maximum or minimum value of trim.

The op amp has provisions for the user to measure and adjust the offset. The offset is measured by connecting the DAC input to the op amp input. The IMONEN and OMONEN control bits allow the non-inverting input and op amp output to be connected to the ADC inputs to do input offset calibration. Note that the user must select the appropriate analog input using the ADC controller and set the IMONEN or OMONEN bit to route these op amp signals to the ADC.

The Op amp has a high bandwidth/power mode and a low bandwidth/power operating mode. The user can trim the input offset error for one or both modes, depending on the application. The complementary P and N channel differential pairs on the input allows full rail-to-rail input voltages to be applied to the amplifier inputs. Each differential pair should be trimmed independently to avoid interactions during the calibration procedure. Therefore, there are four sets of trim adjustment bit fields to account for the two input differential pairs and two power modes described above:

- NOFFSETLP[4:0]
- POFFSETLP[4:0]
- NOFFSETHP[4:0]
- POFFSETHP[4:0]

The procedure for calibrating the input offset error assumes that the user will use minimal external components and will measure voltage offsets using the ADC available on the device:

1. Configure the op amp for Unity Gain mode using the internal feedback connection.
2. Connect the non-inverting input of the op amp to a midscale voltage reference of $V_{DD}/2$.
3. Select either high power (high bandwidth) or low power (low bandwidth) operational mode.
4. Disable the P channel differential input pair of the op amp and enable the N-channel differential pair.
5. Measure the input voltage and output voltage of the op amp using an external meter or the internal ADC.
6. Subtract the input voltage measurement from the output voltage measurement value to determine the amount of offset error.
7. Raise or lower the value in the NOFFSETxx register and repeat steps 5 and 6 until the offset error is nulled out.
8. Enable the P channel differential pair of the op amp and disable the N channel differential pair.
9. Repeat steps 5-6 and adjust the POFFSETxx register to null out the offset error.
10. Change the op amp power mode (step 3) and repeat steps 4-9 to trim the op amp offset voltage in the second power mode, using the appropriate POFFSETxx and NOFFSETxx trim registers.

11. Enable both the P channel and N channel differential pair to return the op amp to normal operation with full input voltage range.
12. Save the values of the POFFSETxx and NOFFSETxx SFRs in non-volatile memory for later retrieval and use by the application.

29.4.7 Interrupts

None.

29.4.8 Power-Saving Modes

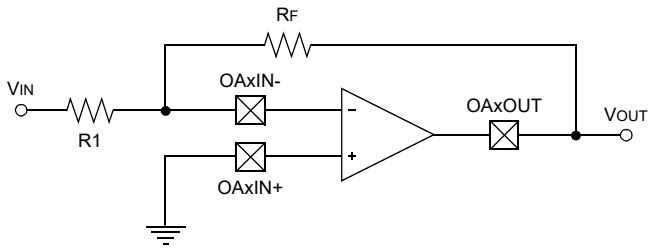
Op amp does not operate in Sleep and Idle mode.

29.5 Op Amp Application Examples

29.5.1 Op Amp Inverting Configuration

In this configuration, an external resistor, R_F, is connected between the OAxOUT pin and OAxIN- pin to provide feedback. An input signal is applied to the OAxIN- pin through resistor R1. The gain in this mode is $-R_F/R1$.

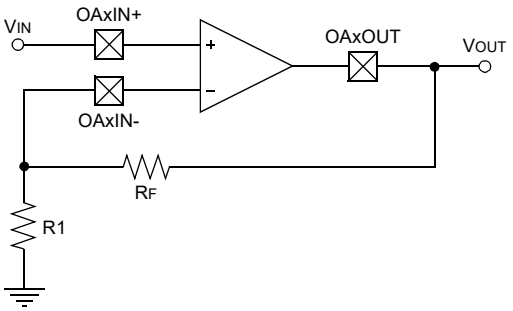
Figure 29-1. Op Amp Inverting Configuration



29.5.2 Op Amp Noninverting Configuration

In this configuration, an external resistor, R_F, is connected between the OAxOUT pin and OAxIN- pin to provide feedback. R1 is connected between OAxIN- and ground. An input signal is applied to the OAxIN+ pin. The gain in this mode is $1 + (R_F/R1)$.

Figure 29-2. Op Amp Noninverting Configuration



30. Watchdog Timer (WDT)

The dsPIC33A device family offers several built-in capabilities that allow user applications to select the best balance of performance and low-power consumption.

The WDT can be used to detect system software malfunctions by resetting the device if the WDT is not cleared periodically in software. The WDT can be configured in Window mode or Non-Window mode. Various WDT time-out periods can be selected using the WDT postscaler. The WDT can also be used to wake the device from Sleep or Idle mode (Power Save mode).

The Watchdog Timer has the following features:

- Independent Run and Sleep Mode Counters
- May Use Alternate Clock Sources and Postscalers for Run Mode Counter
- Up to 32 Configurable Time-Out Periods
- Independent 5-bit Postscalers for Run and Sleep Mode Counters
- Hardware and Software Enable
- Windowed WDT Option - Four Window Sizes Controlled by Software Configuration Bits

30.1 Device-Specific Information

Table 30-1. WDT Summary Table

Clock Source	Peripheral Bus Speed
Slow speed peripheral clock	Slow

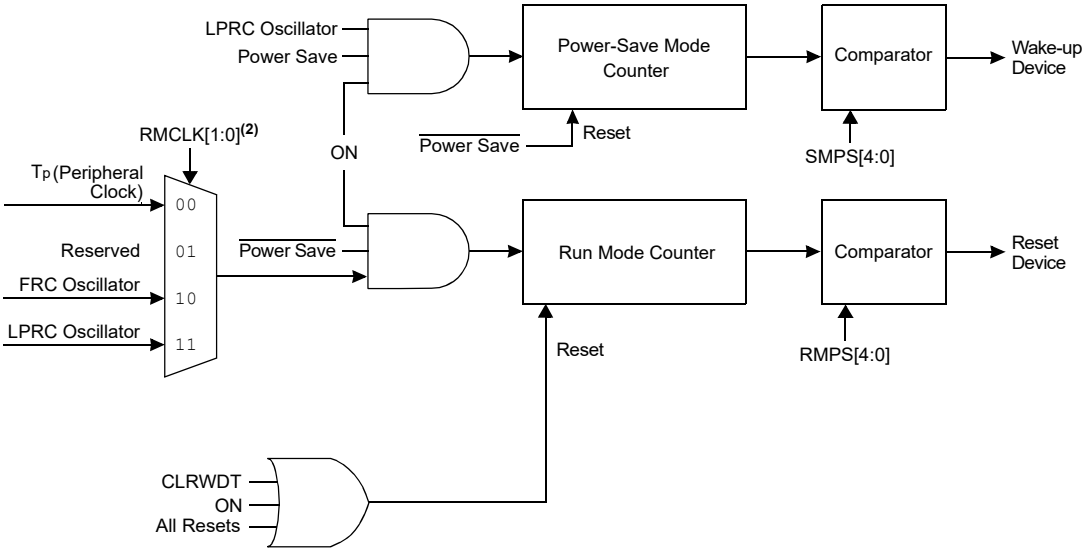
30.2 Architectural Overview

The WDT is a free-running timer with a configurable postscaler. The counter is clocked with an external reference clock until the counter value exceeds the selected WDT period. If enabled, the WDT will continue to operate even if the main processor clock (e.g., the crystal oscillator) fails. The WDT, when enabled, operates from the internal Low-Power RC (LPRC) oscillator clock source or user selectable clock source in Run mode. Refer to Figure 30-1 for a block diagram of the WDT.

The WDT uses separate internal counters for use in Run mode and Sleep/Idle modes. One counter operates only in Run mode; the count value of this counter is frozen when the device is in Sleep or Idle modes. The second counter operates only in Sleep and Idle modes; it is reset when entering Sleep or Idle. In either case, this provides the following benefits:

- A different WDT clock source can be used in Run mode.
- A different postscaler value may be used in Run mode vs. Sleep/Idle modes.
- The Run mode WDT count is preserved while in Sleep or Idle modes, which makes it easier to manage the WDT while in windowed mode.

Figure 30-1. Watchdog Timer Block Diagram



Notes:

1. WDT Reset behavior following a specific clock switch event is device-dependent.
2. The available clock sources are device-dependent.

30.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x32C8	WDTCON	31:24								
		23:16								
		15:8	ON	WINSIZE[1:0]		RMPS[4:0]				
		7:0	RMCLK[1:0]		SMPS[4:0]					WINDIS

30.3.1 Watchdog Timer Control Register

Name: WDTCON
Offset: 0x32C8

Legend: cfg = Configurable at Reset

Notes:

1. All these bits reflect the value of FWDT Configuration bits at Reset and can be set or cleared by software.
2. When ON = 1, writes to the WINSIZE[1:0], RMPS[4:0], RMCLK[1:0], SMPS[4:0], WINDIS are disabled.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON	WINSIZE[1:0]		RMPS[4:0]				
Access	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg
Reset								

Bit	7	6	5	4	3	2	1	0
	RMCLK[1:0]		SMPS[4:0]					WINDIS
Access	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg
Reset								

Bit 15 – ON On bit(2)

Value	Description
01	Enables the WDT
00	Disable and reset WDT

Bits 14:13 – WINSIZE[1:0] Size of Watchdog Window

Value	Description
11	Window size is 25% (Timer Count > 11xxx...xxxxx for timer to be cleared)
10	Window size is 37.5% (Timer Count > 101xx...xxxxx for timer to be cleared)
01	Window size is 50% (Timer Count > 1xxxx...xxxxx for timer to be cleared)
00	Window size is 75% (Timer Count > 01xxx...xxxxx for timer to be cleared)

Bits 12:8 – RMPS[4:0] Configures the postscaler value for Run Mode Counter
Refer to Table 30-2 for the encoding of this bit field.

Bits 7:6 – RMCLK[1:0] Watchdog Timer Run Mode Counter Clock Selection bits

Value	Description
11	LPRC Oscillator
10	BFRC Oscillator
01	Reserved

Value	Description
00	$F_{OSC}/4$

Bits 5:1 – SMPS[4:0] Configures the postscaler value for Sleep Mode Counter
Refer to Table 30-2 for the encoding of this bit field.

Bit 0 – WINDIS Watchdog Window Mode Disable bit

Value	Description
1	Disables Window mode
0	Enables Window mode

30.4 Watchdog Timer Operation

30.4.1 Watchdog Clock Inputs

The Sleep Mode Counter always uses the 32 kHz clock source, which is typically connected to a LPRC oscillator. The Run Mode Counter will use the clock source selected by the RMCLK[1:0] in (WDTCON[7:6]).

Note: The LPRC oscillator is automatically enabled whenever it is being used as a WDT clock source and the WDT is enabled.

30.4.2 Postscaler

A variable postscaler divides the WDT prescaler output and allows for a wide range of time-out periods. User software may select the postscaler setting for the WDT in Run and Sleep mode. The postscaler is controlled by the RMPS[4:0] bits (WDTCON[12:8]) in Run Mode and SMPS[4:0] bits (WDTCON[5:1]) in Sleep/Idle Mode, which allows the selection of 32 settings, from 1:1 to 1:2147483648. Using the postscaler, time-out periods ranging from 1 ms to 24 days can be achieved.

The RMPS[4:0] selects the postscaler value for the Run Mode Counter. The RMPS[4:0] bits can only be changed when the ON(WDTCON[15]) bit is cleared.

The SMPS[4:0] selects the postscaler value for the Sleep Mode Counter. The SMPS bits can only be changed when the ON bit is cleared.

The WDT module time-out period is directly related to the frequency of the WDT clock source. The nominal frequency of the clock source is device dependent. The frequency may vary as a function of the device operating voltage and temperature.

The available clock sources for Run mode are device dependent.

30.4.3 Watchdog Time-out Period Selection

The WDT time-out period is selected by postscaler dividers.

The WDT time-out period can be calculated using Equation 30-1 and WDT time-out period with window option can be calculated using Equation 30-2. The WDT period configurations for WDT time-out period are provided in Table 30-2.

Equation 30-1. Watchdog Timer Time-out Period

$$T_{WTO} = (N1) \times (N2) \times (T_{CLK})$$

Where:

$N1$ = 32

$N2$ = Postscaler divider ratio (see Table 30-2)

T_{CLK} = WDT clock source

Table 30-2. WDT Period Configurations

RMPS[4:0]or SMPS[4:0]	Post-scale Ratio	Time-out Period		
		32 kHz	8 MHz	50 MHz
00000	1	1 ms	4 µs	.64 µs
00001	2	2 ms	8 µs	1.28 µs
00010	4	4 ms	16 µs	2.56 µs
00011	8	8 ms	32 µs	5.12 µs
00100	16	16 ms	64 µs	10.24 µs
00101	32	32 ms	128 µs	20.48 µs
00110	64	64 ms	256 µs	40.95 µs
00111	128	128 ms	512 µs	81.9 µs
01000	256	256 ms	1.02 ms	163.85 µs
01001	512	512 ms	2.05 ms	327.7 µs
01010	1024	1.024s	4.1 ms	0.655 ms
01011	2048	2.048s	8.2 ms	1.31 ms
01100	4096	4.096s	16.4 ms	2.62 ms
01101	8192	8.192s	32.8ms	5.25 ms
01110	16384	16.384s	65.5 ms	10.485 ms
01111	32768	32.768s	131.1 ms	20.95 ms
10000	65536	00:01:05	262.1 ms	41.95 ms
10001	131072	00:02:11	524.3 ms	83.85 ms
10010	262144	00:04:22	1.05s	167.75 ms
10011	524288	00:08:44	2.1s	335.55 ms
10100	1048576	00:17:28	4.2s	0.67s
10101	2097152	00:34:57	8.4 s	1.34s
10110	4194304	01:09:54	16.8s	2.685s
10111	8388608	02:19:48	33.5s	5.35s
11000	16777216	04:39:37	00:01:07	10.75s
11001	33554432	09:19:14	00:02:14	21.45s
11010	67108864	18:38:28	00:04:28	00:01:25
11011	134217728	1day, 13:16:57	00:08:56	00:02:51
11100	268435456	3days, 02:33:55	00:17:53	00:05:43
11101	536870912	6days, 05:07:50	00:35:47	00:11:27
11110	1073741824	12days, 10:15:41	01:11:35	00:22:53
11111	2147483648	24days, 20:31:23	02:23:10	00:45:46

30.4.4 Enabling and Disabling the Watchdog Timer

The Reset value of each bit in the WDTCON SFR can be determined by the FWDT configuration register to allow the WDT to be either disabled or enabled and preconfigured at device start-up.

The Watchdog Timer is enabled or disabled by the WDT Enable ON(WDTCON[15]) bit. When the ON bit is set, the WDT is enabled and writes to the WINSIZE [1:0], RMPS [4:0], RMCLK [1:0], SMPS [4:0] and WINDIS are disabled. Additionally, writes to the WDTCON register can also be locked using PACCON2[WDWTCONWR] to avoid accidental WDT writes. Refer to 8.6. Peripheral Access Controller (PAC) for more information.

If the ON bit is cleared in the WDTCON register, the WDT is disabled and reset and WDTCON SFR modification is allowed.

The ON bit in WDTCON mirrors the ON bit in the FWDT Configuration register on device Reset. The ON bit allows the user application to enable the WDT for critical code segments and disable the WDT during non-critical segments for maximum power savings.

The WDT flag bit, WDTO (RCON[4]), is not cleared automatically after a WDT timeout. To detect subsequent WDT events, the flag must be cleared in software.

Note: The Run mode WDT is expected to stall when the device is performing any Flash operation while executing the code from the same Flash partition, and it is expected that it will resume its Run mode WDT counter once the Flash operation is complete.

30.4.5 Watchdog Timer Window

The Watchdog Timer has an optional Windowed mode enabled the WINDIS = 0 bit (WDTCON[0]). In the Windowed mode (WINDIS= 0), the WDT must be cleared only within the allowed window interval of the Watchdog time-out period, as shown in Figure 30-2. There is also an option to select a WDT window where the WDT should be cleared based on the Watchdog Window Select bits (WINSIZE[1:0]) in WDTCON setting. The bit settings are as follows:

- 11 = WDT Allowed Window is 25% of the WDT period
- 10 = WDT Allowed Window is 37.5% of the WDT period
- 01 = WDT Allowed Window is 50% of the WDT period
- 00 = WDT Allowed Window is 75% of the WDT period

If the Watchdog Timer is cleared before the allowed window, a system Reset is generated immediately.

The Windowed mode is useful for resetting the device during unexpectedly quick or slow execution of a critical portion of the code.

Equation 30-2. Watchdog Timer Time-out Period with Window Option

$T_{DW} = T_{WTO} - T_{AW}$

$T_{AW} = (N1) \times (N2) \times (T_{CLK}) \times (WINSIZE[1:0])$

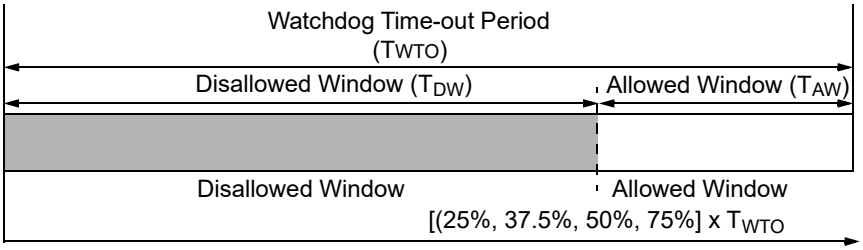
Where:

$N1$ = 32

$N2$ = Postscaler divider ratio (see Table 30-2)

T_{CLK} = WDT clock source

Figure 30-2. Windowed Watchdog Timer



30.5 Watchdog Timer Reset

The Watchdog Timer is reset and the Run mode WDT counter is cleared by any of the following circumstances:

- Any device Reset
- Execution of WDT_CLEAR Instruction
- Disabling WDT by clearing the ON bit of WDTCON register.
- Execution of a `DEBUG` command

The Sleep mode WDT counter is reset upon entry into Sleep/Idle.

Note: The Run mode WDT is not reset when the device enters a Power-Saving mode.

30.5.1 WDT Time-out in Run Mode

When the WDT times out in Run mode, a device Reset/NMI is generated. Firmware can determine if the cause of the Reset was the WDT time-out in Run mode by testing the WDTO bit in RCON. WDT can be controlled to raise a device reset or WDT Generic Trap on WDT time-out by modifying WDTRSTEN (FWDT[16]) bit.

30.5.2 WDT Time-out in Sleep/Idle Mode

When the WDT module times out in Power Save mode, it wakes the device and the WDT Run mode resumes counting.

WDT Sleep/Idle mode event can also trigger a WDT interrupt on wake up from Sleep/Idle mode if IEC0[WDTIE] is enabled.

If the WDT Interrupt Enable bit, WDTIE, is cleared, an interrupt will not be generated on WDT Wakeup event. However, the WDTIF bit will still be set if an interrupt condition occurs. The user can clear the interrupt in the Interrupt Service Routine (ISR) by clearing WDTIF. See the **"Interrupt Controller"** chapter for more information.

30.5.3 Wake from Power Save Mode by a Non-WDT Event

When the device is awakened from a Power-Saving mode by a non-WDT NMI interrupt, the Power-Saving mode WDT is held in Reset and the WDT Run mode continues counting from the pre-power save count value.

30.6 Operation of Watchdog Timer in Sleep/Idle Modes

30.6.1 WDT Operation in Power-Saving Modes

The WDT, if enabled, will continue operation in Sleep mode or Idle mode and can be used to wake the device. This allows the device to remain in Sleep or Idle mode until the WDT expires or another interrupt wakes the device. If the device does not re-enter Sleep or Idle mode following a wake-up, the WDT must be disabled or periodically serviced to prevent a WDT Run mode NMI.

30.6.2 WDT Operation in Sleep Mode

The WDT module may be used to wake the device from Sleep mode. When entering Sleep mode, the WDT Run mode counter stops counting and the Power Save mode WDT begins counting from the Reset state until it times out or the device is woken up by an interrupt. When the WDT times out in Sleep mode, the device wakes up, resumes code execution and resumes the Run mode WDT.

30.6.3 WDT Operation in Idle Mode

The WDT module may be used to wake the device from Idle mode. When entering Idle mode, the WDT Run mode counter stops counting and the Power Save mode WDT begins counting from the Reset state until it times out or the device is woken up by an interrupt. Once the device wakes up, either by sleep WDT time-out or by any interrupt, the device resumes the code execution and the Run mode WDT timer.

30.6.4 Time Delays During Wake-up

There will be a time delay between the WDT event in Sleep and the beginning of code execution. The duration of this delay consists of the start-up time for the oscillator in use. Unlike a wake-up from Sleep mode, there are no time delays associated with wake-up from Idle mode. The system clock is running during Idle mode; therefore, no start-up delays are required at wake-up.

30.7 WDT Generic Trap

WDT Event occurs due to counter overflow or an attempt to clear the WDT in a windowed interval. The FWDT[RSTEN] bit in configuration fuse can be set to enable a device reset or be cleared to raise a Generic trap in case of a WDT event. When the FWDT[RSTEN] bit is cleared, the WDT Event results in (INTCON5[WDT]) WDT Generic Trap. In this case, the WDT counter resets and starts counting again soon after the WDT event has occurred.

30.8 WDT Sample Configuration

30.8.1 Run WDT Configuration

Example 30-1 shows a code example to configure Run WDT for 1.024 sec with LPRC as its clock source.

Example 30-1. Run WDT Configuration

```
//code example to configure Run WDT for 1.024sec
int main()
{

WDTCONbits.RMCLK = 3;          // LPRC as Run WDT Clock
WDTCONbits.RMPS = 10;          // Run Postscaler 1024
WDTCONbits.ON = 1;             // Enable WDT

}
```

30.8.2 Sleep WDT Configuration

Example 30-2 shows a code example to configure Sleep WDT for 1.024 sec with LPRC as its default clock source.

Example 30-2. Sleep WDT Configuration Example

```
//code example to configure Sleep WDT for 1.024 sec
int main()
{

WDTCONbits.SMPS = 10;          // Sleep Postscaler 1024
WDTCONbits.ON = 1;             // Enable WDT
```

```
}
```

30.8.3 Configuring WDT Via Config Fuse

Example 30-3 shows a code example to configure Run and Sleep WDT for 1.024 sec with LPRC as Run WDT Clock Source via Config Fuse.

Example 30-3. WDT Configuration Example

```
//code example to configure Run and Sleep WDT for 1.024 sec via config
//fuse
// FWDT
#pragma config FWDT_WINEN = ON          // Watchdog Timer Window Enable bit
(Watchdog Timer operates in Window mode)
#pragma config FWDT_SWDTMPS = PS1024    // Sleep Mode Watchdog Timer Post
Scaler select bits (1:1024)
#pragma config FWDT_RCLKSEL = BPRC256   // Watchdog Timer Clock select bits
(WDT Run Mode uses BFRC:256)
#pragma config FWDT_RWDTPS = PS1024     // Run Mode Watchdog Timer Post Scaler
select bits (1:1024)
#pragma config FWDT_WDTWIN = WIN25      // Watchdog Timer Window Size Select
bits (WDT Window is 25% of WDT period)
#pragma config FWDT_WDTEN = SW          // Watchdog Timer Enable bit (WDT is
controlled by software, use WDTCON.ON bit)
#pragma config FWDT_WDTRSTEN = ON       // Watchdog Timer Reset Enable bit
(WDT event generates a reset)
```

30.8.4 Clearing Watchdog Timer

Example 30-4 shows a code example to clear Run WDT Counter.

Notes:

1. If enabled, Run WDT should be periodically serviced (must be disabled within timeout) to prevent WDT Run mode NMI/Reset.
2. Processor takes a few cycles to execute `ClrWdt` instruction; if this is not taken into account, Run WDT will not be cleared reliably.
3. When Window mode is enabled, use caution when clearing WDT because this can only be done in the allowed Window. Clearing WDT in a disallowed window causes Run WDT NMI/Reset.

Example 30-4. Clear WDT after 300ms

```
//code example to clear Run Counter after 300ms with WDT timeout as 1.024sec
 int main()
{
WDTCONbits.RMPS = 10;     // Run Postscaler 1024
 WDTCONbits.ON = 1;       // Enable WDT
__delay_ms(300);          // 300ms delay
ClrWdt();                 // Clears Run Counter after 300ms delay

}
```

31. Deadman Timer (DMT)

The Deadman Timer (DMT) module is designed to enable users to monitor the health of their application software by requiring periodic timer interrupts within a user-specified timing window. The DMT module is a synchronous counter that, when enabled, counts instruction fetches and is able to cause a soft trap if the DMT counter is not cleared within a set number of instructions. The DMT is typically connected to the system clock that drives the processor (T_{CY}). The user specifies the timer time-out value and a mask value that specifies the range of the window, which is the range of counts that is not considered for the comparison event.

Some of the key features of this module are:

- Software Enabled
- User-Configurable Time-out Period or Instruction Count
- Two Instruction Sequences to Clear Timer
- 32-Bit Configurable Window to Clear Timer

Table 31-1. DMT Summary

DMT Module Instances	DMT Output	Clock Source	Peripheral Bus Speed
1	Soft Trap	Slow Speed Peripheral Clock	Slow

31.1 Architectural Overview

The primary function of the Deadman Timer (DMT) is to interrupt the processor in the event of a software malfunction. The DMT, which works on the system clock, is a free-running instruction fetch timer, which is clocked whenever an instruction fetch occurs, until a count match occurs. Instructions are not fetched when the processor is in Sleep mode.

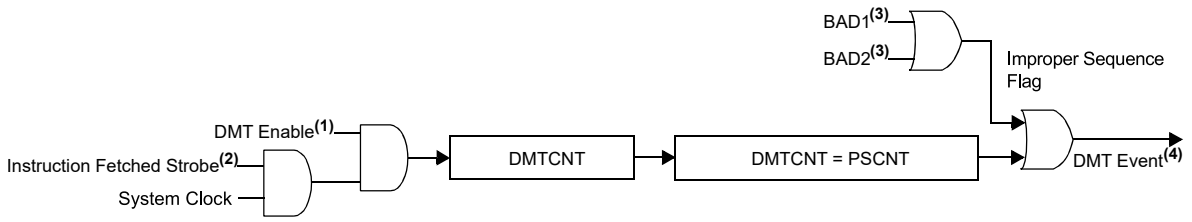
DMT can be enabled in the Configuration fuse or by software in the DMTCON register by setting the ON bit.

The DMT module consists of a 32-bit counter, the read-only DMTCNT register, and a time-out count match value as specified by the PSCNT register. Whenever the count match occurs, a DMT event will occur, which is a soft trap.

A DMT is typically used in mission-critical and safety-critical applications where any single failure of the software functionality and sequencing must be detected.

Figure 31-1 shows a block diagram of the Deadman Timer module.

Figure 31-1. Deadman Timer Block Diagram



Notes:

1. The DMT can be enabled via Special Function Register (SFR), DMTCON.
2. The DMT is clocked whenever the instructions are fetched by the processor using a system clock. For example, after executing a `GOTO` instruction (which uses four instruction cycles), the DMT counter will be incremented only once.
3. BAD1 and BAD2 are the improper sequence flags.
4. A DMT event is a non-maskable soft trap.

31.2 Deadman Timer Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3A00	DMTCON	31:24								
		23:16								
		15:8	ON							
		7:0								
0x3A04	DMTPRECLR	31:24								
		23:16								
		15:8	STEP1[7:0]							
		7:0								
0x3A08	DMTCLR	31:24								
		23:16								
		15:8								
		7:0	STEP2[7:0]							
0x3A0C	DMTSTAT	31:24								
		23:16								
		15:8								
		7:0	BAD1	BAD2	DMTEVENT					WINOPN
0x3A10	DMTCNT	31:24	COUNTER[31:24]							
		23:16	COUNTER[23:16]							
		15:8	COUNTER[15:8]							
		7:0	COUNTER[7:0]							
0x3A14	PSCNT	31:24	PSCNT[31:24]							
		23:16	PSCNT[23:16]							
		15:8	PSCNT[15:8]							
		7:0	PSCNT[7:0]							
0x3A18	PSINTV	31:24	PSINTV[31:24]							
		23:16	PSINTV[23:16]							
		15:8	PSINTV[15:8]							
		7:0	PSINTV[7:0]							
0x3A1C	PPPC	31:24								
		23:16								
		15:8	NMISTEP1[7:0]							
		7:0								
0x3A20	PPC	31:24								
		23:16								
		15:8								
		7:0	NMISTEP2[7:0]							

31.2.1 Deadman Timer Control Register

Name: DMTCON
Offset: 0x3A00
Property: Read-only

Legend: S = Settable bit

Note:

1. The ON bit is a write once bit, and writes to the PSCNT and PSINTV registers are not enabled when ON = 1.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON							
Access	R/S							
Reset	0							

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 15 – ON On bit(1)

Value	Description
1	Enables the Deadman Timer
0	The DMT is reset at a RESET instruction

31.2.2 Deadman Timer Preclear Register

Name: DMTPRECLR
Offset: 0x3A04
Property: R/W

Notes:

1. STEP1[15:8] (DMTPRECLR[15:8]) bits are cleared when a DMT Reset event occurs.
2. STEP1 bits are also cleared if the STEP2 (DMTCLR[7:0]) bits are loaded with the correct value in the correct sequence.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	STEP1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 15:8 – STEP1[7:0] Preclear Enable bits(1,2)

Value	Description
01000000b	Enables the Deadman Timer preclear
All other write patterns	Sets the BAD1 flag

31.2.3 Deadman Timer Clear Register

Name: DMTCLR
Offset: 0x3A08
Property: R/W

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	STEP2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – STEP2[7:0] Clear Enable bits

Value	Description
00001000b	Clears STEP1 (DMTPRECLR[15:8]), STEP2 (DMTCLR[7:0]), and the Deadman Timer (STEP2) if, and only if, preceded by correct loading of the Preclear Enable bits (STEP1) in the correct sequence. The write to the STEP2 bits field may be verified by reading DMTCNT and observing the counter being reset.
All other write patterns	Sets BAD2 Flag; the value in the STEP1 bits will remain unchanged and the new value being written to the STEP2 bits will be captured.

31.2.4 Deadman Timer Status Register

Name: DMTSTAT
Offset: 0x3A0C
Property: R/W

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	BAD1	BAD2	DMTEVENT					WINOPN
Access	R	R	R					R
Reset	0	0	0					0

Bit 7 – BAD1 BAD STEP1 Value Detect bit
This bit is cleared by a Reset or a successful post NMI clear sequence bit.

Value	Description
1	Incorrect STEP1[7:0] value was detected
0	Incorrect STEP1[7:0] value was not detected

Bit 6 – BAD2 BAD STEP2 Value Detect bit
This bit is cleared by a Reset or a successful post NMI clear sequence bit.

Value	Description
1	Incorrect STEP2[7:0] value was detected
0	Incorrect STEP2[7:0] value was not detected

Bit 5 – DMTEVENT Deadman Timer Event bit
This bit is cleared by a Reset or a successful post NMI clear sequence bit.

Value	Description
1	DMT counter expired, or a BAD1 or BAD2 step occurred
0	No errors detected

Bit 0 – WINOPN Deadman Timer Clear Window bit

Value	Description
1	Deadman Timer clear window is open
0	Deadman Timer clear window is not open

31.2.5 Deadman Timer Count Register

Name: DMTCNT
Offset: 0x3A10
Property: R/W

Bit	31	30	29	28	27	26	25	24
	COUNTER[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	COUNTER[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	COUNTER[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	COUNTER[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – COUNTER[31:0] Read Current Contents of DMT Counter bits

31.2.6 Deadman Timer Counter Limit Register

Name: PSCNT
Offset: 0x3A14
Property: R/W

Bit	31	30	29	28	27	26	25	24
	PSCNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PSCNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PSCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PSCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PSCNT[31:0] DMT Instruction Count Limit Value bits
This register sets the maximum value of the DMT counter. If the counter counts past this limit value, a DMT soft trap is generated.
This register can be written only when the ON bit is zero.

31.2.7 DMT Interval Status Register

Name: PSINTV
Offset: 0x3A18
Property: R/W

Bit	31	30	29	28	27	26	25	24
	PSINTV[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PSINTV[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PSINTV[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PSINTV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PSINTV[31:0] DMT Window Interval bits
The DMT Interval Status register specifies when the DMT counter “Clear” operation may be performed. The DMTCNT bits must be greater than the PSINTV bits before the Clear operation can be initiated. This register can be written only when the ON bit is zero.

31.2.8 DMT NMI Preclear Register

Name: PPPC
Offset: 0x3A1C
Property: R/W

Note:

1. Bits[15:8] are cleared when a DMT Reset event occurs. NMISTEP1 bits are also cleared if NMISTEP2 (PPC[7:0]) bits are loaded with the correct value in the correct sequence.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	NMISTEP1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 15:8 – NMISTEP1[7:0] NMI Post-Processing Preclear Enable bits(1)

Value	Description
01000001b	Enables the NMI preclear (NMI_STEP1)
All other write patterns	This register remains unchanged and the instruction writing to it is considered to be unsuccessful

31.2.9 DMT NMI Clear Register

Name: PPC
Offset: 0x3A20
Property: R/W

Note:

1. Bits[7:0] are also cleared when a DMT Reset event occurs. NMISTEP1 and NMISTEP2 are also cleared if NMISTEP2 is loaded with the correct value during the Pre-DMT Event Clear sequence.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	NMISTEP2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – NMISTEP2[7:0] NMI Clear DMT Event Enable bits(1)

Value	Description
10001000b	If write pattern has been preceded by correct execution of PPPC NMI preclear instruction, the DMT event is cleared
All other write patterns	This register remains unchanged and the instruction writing to it is considered to be unsuccessful

31.3 DMT Operation

31.3.1 Enabling and Disabling the DMT Module

The DMT module can be enabled through software by writing to the DMTCON register. Software can enable the DMT by setting the ON (DMTCON[15]) bit in the Deadman Timer Control register. The ON bit is a write once bit, and disabling the DMT in software is not possible.

31.3.2 DMT Count Selection

The Deadman Timer count is set by the PSCNT register bits. The current DMT count value can be obtained by reading the DMTCNT register.

The PSCNT[31:0] bits allow the software to read the maximum count selected for the Deadman Timer. Whenever the DMT event occurs, the user can always compare to see whether the current counter value in the DMTCNT register is equal to the value of the PSCNT register, which holds the maximum count value.

The PSINTV[31:0] bits allow the software to read the DMT window interval value. So whenever the DMT current counter value in DMTCNT reaches the value of PSINTV, the window interval opens so that the user can insert the clear sequence to the STEP2 bits, which causes the DMT to reset.

The initial DMT state has the timer cleared; the DMTPRECLR and BAD1/BAD2 flags are cleared. When reset, the DMT clears the DMTSTAT register. When activated, the DMT begins to count whenever an instruction is fetched while clocks are running.

31.3.3 DMT Count Windowed Interval

The DMT module has a Windowed Operation mode. The PSINTV register will set the window interval value. In Windowed mode, software can clear the DMT only when the counter is in its final window before a count match occurs. That is, if the DMT counter value (DMTCNT) is greater than or equal to the value written to the window interval value (PSINTV), then only the clear sequence can be inserted into the DMT module.

The “window interval” is open when: (COUNTER) ≥ PSINTV.

If the DMT is cleared before the allowed window, a Deadman Timer soft trap is immediately generated.

Note: Care should be exercised when selecting the “window interval”, such that the interval does not begin after the time-out instruction count specified by PSCNT[31:0].

31.3.4 Resetting the DMT

The DMT can be reset in three ways:

1. System Reset.
2. Writing an ordered sequence to the DMTPRECLR and DMTCLR registers. Refer to 31.3.4.1. Clearing Pre-DMT Event.
3. Following an expired DMT count exception, to clear the DMT event, two instructions must be executed in sequence: the PPPC followed by the PPC. Refer to 31.3.4.2. Clearing Post-DMT Event.

31.3.4.1 Clearing Pre-DMT Event

Clearing the DMT counter value requires a special sequence of operations:

1. The STEP1[15:8] bits in the DMTPRECLR register must be written as ‘01000000’ (0x40).
2. The STEP2[7:0] bits in the DMTCLR register must be written as ‘00001000’ (0x08). This can only be done if preceded by Step 1 and the DMT is in the open window interval (WINOPN = 1).

Once these values are written, the DMT counter will be cleared to zero. The DMTPRECLR, DMTCLR and DMTSTAT registers’ values will also be cleared to zero. Refer to the pseudocode in Example 31-1 for the correct sequence of operation. This is the normal use of the module while clearing the DMT.

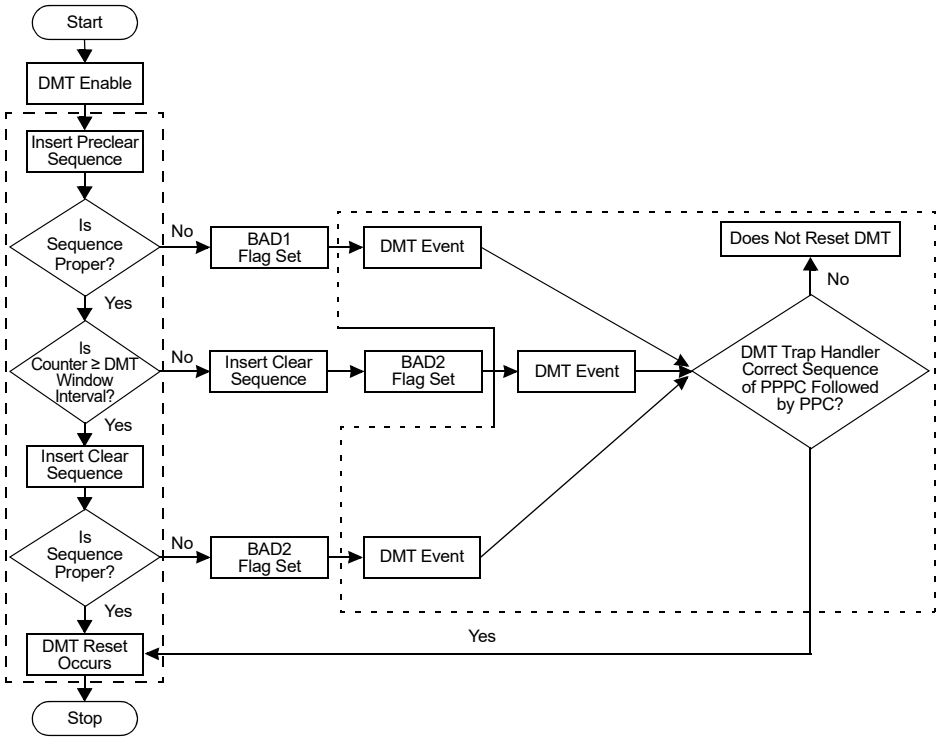
If any value other than 0x40 is written to the STEP1 bits, the BAD1 bit in the DMTSTAT register will be set and it causes a DMT event to occur. Any value other than 0x08 written to the STEP2 bits will cause the BAD2 bit to be set in the DMTSTAT register. Also, if Step 2 is not preceded by Step 1, or Step 2 is not carried out in the open window interval, it causes the BAD2 flag to be set. Immediately, a DMT event will occur.

Example 31-1. Clearing DMT Before DMT Count Match Occurs

```
//Assume PSCNT = 0x0000FFFF and PSINTV = 0x000000FF
//When DMTCNT >= PSINTV WINOPN gets set

DMTPRECLR=0b01000000;            // Valid DMTPRECLR Value
while(DMTSTAT & 0x0001!=0x0001); //Wait till DMT Clear Window is open
```

```
                                        i.e WINOPN=1
DMTCLR=0b00001000;                    // Valid DMTCLR Value
```

Figure 31-2. Flowchart for Resetting the DMT



31.3.4.2 Clearing Post-DMT Event

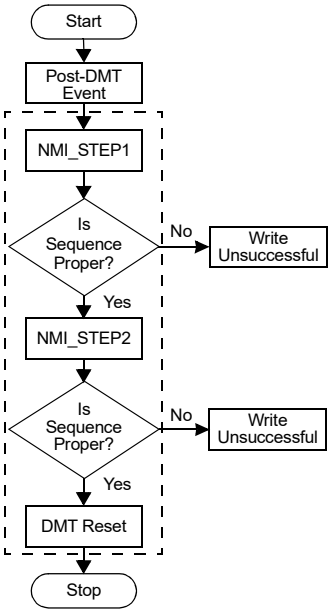
The DMT NMI will cause the processor to execute a trap handler (ISR) that will decide what to do. The user may decide to reset the DMT through the Post-Clearing (PPPC followed by PPC) mechanism, or the user can execute a `RESET` instruction to reset the system in the trap handler.

Clearing the DMT counter post-expiry requires a special sequence of operations:

1. The NMI_STEP1[7:0] bits in the PPPC register must be written as '`01000000b`'.
2. The NMI_STEP2[7:0] bits in the PPC register must be written as '`10001000b`'. This can only be done if preceded by NMI_STEP1.

If any value other than '`01000000b`' is written to the NMI_STEP1 bits, this register remains unchanged and the instruction writing to it is considered to be unsuccessful. Any value other than '`10001000b`' written to the NMI_STEP2 bits renders the operation unsuccessful.

Figure 31-3. Flowchart for Clearing Post-DMT Event



31.3.5 DMT Generic Trap

A DMT Event due to counter overflow or a BAD1/BAD2 event results in a DMT Generic Trap. This being a persistent event, DMT needs to be reset using the PPPC/PPC registers within the ISR before exiting the trap routine.

31.3.6 DMT Operation in Power-Saving Modes

As the DMT module is only incremented by instruction fetches, the count value will not change when the core is inactive. The DMT module remains inactive in Sleep and Idle modes. As soon as the device wakes up from Sleep or Idle, the DMT counter again starts incrementing.

32. Power-Saving Modes

The dsPIC33AK128MC106 family devices provide the ability to manage power consumption by selectively managing clocking to the CPU and the peripherals. In general, a lower clock frequency and a reduction in the number of peripherals being clocked constitutes lower consumed power.

This section describes the power-saving modes implemented in dsPIC33A devices. The dsPIC33A device family offers a number of built-in capabilities that allow user applications to select the best balance of performance and low-power consumption.

The power-saving module supports these features:

- Instruction-Based Power-Saving Modes
- Peripheral Module Disable (PMD)

Combinations of these methods can be used to selectively tailor an application's power consumption while still maintaining critical application features, such as timing-sensitive communications.

32.1 Architectural Overview

Reducing the system clock frequency results in power saving that is roughly proportional to the frequency reduction. The dsPIC33A devices provide an on-the-fly clock switching feature that allows the user application to optimize power consumption by dynamically changing the system clock frequency.

There are two ways to reduce power consumption in dsPIC33A devices:

1. Instruction-based power-saving modes which includes Sleep and Idle.
   - **Sleep Mode:** In Sleep mode, the CPU, the system clock source and the peripherals that operate on the system clock source are disabled. This is the lowest power mode for the device. Optionally, the peripherals can operate in Sleep mode using specific clock sources. Please refer to the individual peripheral chapter for descriptions on behavior in Sleep mode. The SLEEP status flag bit in the Reset Control register (RCON[4]) is set when the device enters Sleep mode.
   - **Idle Mode:** In Idle mode, the CPU is disabled, but the system clock source continues to operate. The peripherals continue to operate but can optionally be disabled. The IDLE status flag bit in the Reset Control register (RCON[3]) is set when the device enters Idle mode.

   The SLEEP and IDLE status bits are cleared on Power-on Reset (POR) and Brown-out Reset (BOR). These bits can also be cleared in software. For more information on Resets, refer to the **"Resets"** section.
2. The peripherals can be selectively disabled using the Peripheral Module Disable (PMD) bit of the corresponding peripheral.

Notes:

1. `SLEEP_MODE` and `IDLE_MODE` are constants defined in the assembler include file for the selected device.
2. Sleep mode does not change the state of the I/O pins.
3. Execution of the `PWRSAV` instruction is ignored while any of the NVM operations are in progress.

32.2 Power-Saving Control Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3198	RCON	31:24								
		23:16			VREG3R	VREG2R	VREG1R			
		15:8							CM	
		7:0	EXTR	SWR		WDTO	SLEEP	IDLE	BOR	POR
0x319C ... 0x3A43	Reserved									
0x3A44	PMD1	31:24						SPI3MD	SPI2MD	SPI1MD
		23:16						U3MD	U2MD	U1MD
		15:8					T1MD			
		7:0					CCP4MD	CCP3MD	CCP2MD	CCP1MD
0x3A48	PMD2	31:24					CLC4MD	CLC3MD	CLC2MD	CLC1MD
		23:16								DACMD
		15:8								DMAMD
		7:0			SENT2MD	SENT1MD			I2C2MD	I2C1MD
0x3A4C	PMD3	31:24								
		23:16							ADC2MD	ADC1MD
		15:8								OPAMPMD
		7:0				BISS1MD				QEI1MD
0x3A50	PMD4	31:24								
		23:16					CM4MD	CM3MD	CM2MD	CM1MD
		15:8						PWMMD	IOIM4MD	IOIM3MD
		7:0	IOIM2MD	IOIM1MD				DMTMD	CRCMD	PTGMD

32.2.1 Reset Control Register

Name: RCON
Offset: 0x3198
Property: R/W

Legend: R = Readable bit, C = Clearable bit, HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			VREG3R	VREG2R	VREG1R			
Access			R/C/HS	R/C/HS	R/C/HS			
Reset			0	0	0			

Bit	15	14	13	12	11	10	9	8
							CM	
Access							R/C/HS	
Reset							0	

Bit	7	6	5	4	3	2	1	0
	EXTR	SWR		WDTO	SLEEP	IDLE	BOR	POR
Access	R/C/HS	R/C/HS		R/C/HS	R/C/HS	R/C/HS	R/C/HS	R/C/HS
Reset	0	0		0	0	0	1	1

Bit 21 – VREG3R Voltage Domain 3 Regulation Loss Flag bit

Value	Description
1	Voltage Domain 3 lost voltage regulation
0	Voltage Regulation in Domain 3 has not been lost

Bit 20 – VREG2R Voltage Domain 2 Regulation Loss Flag bit

Value	Description
1	Voltage Domain 2 lost voltage regulation
0	Voltage Regulation in Domain 2 has not been lost

Bit 19 – VREG1R Voltage Domain 1 Regulation Loss Flag bit

Value	Description
1	Voltage Domain 1 lost voltage regulation
0	Voltage Regulation in Domain 1 has not been lost

Bit 9 – CM Configuration Mismatch Flag bit

Value	Description
1	A Configuration Mismatch Reset has occurred
0	A Configuration Mismatch Reset has not occurred

Bit 7 – EXTR External Reset ($\overline{MCLR}$) Pin bit

Value	Description
1	A Master Clear (pin) Reset has occurred
0	A Master Clear (pin) Reset has not occurred

Bit 6 – SWR Software RESET (Instruction) Flag bit

Value	Description
1	A RESET instruction has been executed
0	A RESET instruction has not been executed

Bit 4 – WDTO Watchdog Timer Time-out Flag bit

Value	Description
1	Device reset has occurred due to WDT time-out
0	WDT time-out has not occurred

Bit 3 – SLEEP Wake-up from Sleep Flag bit

Value	Description
1	Device has been in Sleep mode
0	Device has not been in Sleep mode

Bit 2 – IDLE Wake-up from Idle Flag bit

Value	Description
1	Device has been in Idle mode
0	Device has not been in Idle mode

Bit 1 – BOR Brown-out Reset Flag bit

Value	Description
1	A Brown-out Reset has occurred
0	A Brown-out Reset has not occurred. Set by hardware at detection of a BOR event.

Bit 0 – POR Power-on Reset Flag bit

Value	Description
1	A Power-on Reset has occurred
0	A Power-on Reset has not occurred. Set by hardware at detection of a POR even

32.2.2 Peripheral Module Disable 1 Register

Name: PMD1
Offset: 0x3A44

Bit	31	30	29	28	27	26	25	24
						SPI3MD	SPI2MD	SPI1MD
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	23	22	21	20	19	18	17	16
						U3MD	U2MD	U1MD
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
					T1MD			
Access					R/W			
Reset					0			

Bit	7	6	5	4	3	2	1	0
					CCP4MD	CCP3MD	CCP2MD	CCP1MD
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 26 – SPI3MD SPI3 Module Disable bit

Value	Description
1	SPI3 module is disabled
0	SPI3 module is enabled

Bit 25 – SPI2MD SPI2 Module Disable bit

Value	Description
1	SPI2 module is disabled
0	SPI2 module is enabled

Bit 24 – SPI1MD SPI1 Module Disable bit

Value	Description
1	SPI1 module is disabled
0	SPI1 module is enabled

Bit 18 – U3MD UART3 Module Disable bit

Value	Description
1	UART3 module is disabled
0	UART3 module is enabled

Bit 17 – U2MD UART2 Module Disable bit

Value	Description
1	UART2 module is disabled
0	UART2 module is enabled

Bit 16 – U1MD UART1 Module Disable bit

Value	Description
1	UART1 module is disabled
0	UART1 module is enabled

Bit 11 – T1MD Timer 1 Module Disable bit

Value	Description
1	Timer 1 module is disabled
0	Timer 1 module is enabled

Bit 3 – CCP4MD CCP 4 Module Disable bit

Value	Description
1	CCP 4 module is disabled
0	CCP 4 module is enabled

Bit 2 – CCP3MD CCP 3 Module Disable bit

Value	Description
1	CCP 3 module is disabled
0	CCP 3 module is enabled

Bit 1 – CCP2MD CCP 2 Module Disable bit

Value	Description
1	CCP 2 module is disabled
0	CCP 2 module is enabled

Bit 0 – CCP1MD CCP 1 Module Disable bit

Value	Description
1	CCP 1 module is disabled
0	CCP 1 module is enabled

32.2.3 Peripheral Module Disable 2 Register

Name: PMD2
Offset: 0x3A48

Bit	31	30	29	28	27	26	25	24
					CLC4MD	CLC3MD	CLC2MD	CLC1MD
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
								DACMD
Access								R/W
Reset								0

Bit	15	14	13	12	11	10	9	8
								DMAMD
Access								R/W
Reset								0

Bit	7	6	5	4	3	2	1	0
			SENT2MD	SENT1MD			I2C2MD	I2C1MD
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bit 27 – CLC4MD CLC 4 Module Disable bit

Value	Description
1	CLC 4 module is disabled
0	CLC 4 module is enabled

Bit 26 – CLC3MD CLC 3 Module Disable bit

Value	Description
1	CLC 3 module is disabled
0	CLC 3 module is enabled

Bit 25 – CLC2MD CLC 2 Module Disable bit

Value	Description
1	CLC 2 module is disabled
0	CLC 2 module is enabled

Bit 24 – CLC1MD CLC 1 Module Disable bit

Value	Description
1	CLC 1 module is disabled
0	CLC 1 module is enabled

Bit 16 – DACMD Comparator 1 Module Disable bit

Value	Description
1	CMP1 Module is disabled
0	CMP1 Module is enabled

Bit 8 – DMAMD DMA Module Disable bit

Value	Description
1	DMA module is disabled
0	DMA module is enabled

Bit 5 – SENT2MD SENT 2 Module Disable bit

Value	Description
1	SENT 2 module is disabled
0	SENT 2 module is enabled

Bit 4 – SENT1MD SENT 1 Module Disable bit

Value	Description
1	SENT 1 module is disabled
0	SENT 1 module is enabled

Bit 1 – I2C2MD I^2C 2 Module Disable bit

Value	Description
1	I^2C 2 module is disabled
0	I^2C 2 module is enabled

Bit 0 – I2C1MD I^2C 1 Module Disable bit

Value	Description
1	I^2C 1 module is disabled
0	I^2C 1 module is enabled

32.2.4 Peripheral Module Disable 3 Register

Name: PMD3
Offset: 0x3A4C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
							ADC2MD	ADC1MD
Access							R/W	R/W
Reset							0	0

Bit	15	14	13	12	11	10	9	8
								OPAMPMD
Access								R/W
Reset								0

Bit	7	6	5	4	3	2	1	0
				BISS1MD				QEI1MD
Access				R/W				R/W
Reset				0				0

Bit 17 – ADC2MD ADC 2 Module Disable bit

Value	Description
1	ADC 2 module is disabled
0	ADC 2 module is enabled

Bit 16 – ADC1MD ADC 1 Module Disable bit

Value	Description
1	ADC 1 module is disabled
0	ADC 1 module is enabled

Bit 8 – OPAMPMD Op Amp 1 Module Disable bit

Value	Description
1	OPA1 module is disabled
0	OPA1 module is enabled

Bit 4 – BISS1MD BiSS 1 Module Disable bit

Value	Description
1	BiSS 1 module is disabled
0	BiSS 1 module is enabled

Bit 0 – QEI1MD QEI 1 Module Disable bit

Value	Description
1	QEI 1 module is disabled
0	QEI 1 module is enabled

32.2.5 Peripheral Module Disable 4 Register

Name: PMD4
Offset: 0x3A50

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					CM4MD	CM3MD	CM2MD	CM1MD
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
						PWMMD	IOIM4MD	IOIM3MD
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
	IOIM2MD	IOIM1MD				DMTMD	CRCMD	PTGMD
Access	R/W	R/W				R/W	R/W	R/W
Reset	0	0				0	0	0

Bit 19 – CM4MD Clock Monitor 4 Disable bit

Value	Description
1	Clock monitor 4 module is disabled
0	Clock monitor 4 module is enabled

Bit 18 – CM3MD Clock Monitor 3 Disable bit

Value	Description
1	Clock monitor 3 module is disabled
0	Clock monitor 3 module is enabled

Bit 17 – CM2MD Clock Monitor 2 Disable bit

Value	Description
1	Clock monitor 2 module is disabled
0	Clock monitor 2 module is enabled

Bit 16 – CM1MD Clock Monitor 1 Disable bit

Value	Description
1	Clock monitor 1 module is disabled
0	Clock monitor 1 module is enabled

Bit 10 – PWMMD PWM Module Disable bit

Value	Description
1	PWM module is disabled
0	PWM module is enabled

Bit 9 – IOIM4MD IO Integrity Monitor 4 Disable bit

Value	Description
1	IO integrity monitor 4 module is disabled
0	IO integrity monitor 4 module is enabled

Bit 8 – IOIM3MD IO Integrity Monitor 3 Disable bit

Value	Description
1	IO integrity monitor 3 module is disabled
0	IO integrity monitor 3 module is enabled

Bit 7 – IOIM2MD IO Integrity Monitor 2 Disable bit

Value	Description
1	IO integrity monitor 2 module is disabled
0	IO integrity monitor 2 module is enabled

Bit 6 – IOIM1MD IO Integrity Monitor 1 Disable bit

Value	Description
1	IO integrity monitor 1 module is disabled
0	IO integrity monitor 1 module is enabled

Bit 2 – DMTMD DMT Module Disable bit

Value	Description
1	CLC6 module is disabled
0	CLC6 module is enabled

Bit 1 – CRCMD CRC Module Disable bit

Value	Description
1	CRC module is disabled
0	CRC module is enabled

Bit 0 – PTGMD PTG Module Disable bit

Value	Description
1	PTG module is disabled
0	PTG module is enabled

32.3 Power-Saving Operations

32.3.1 Instruction-Based Power-Saving Modes

The dsPIC33A family of devices can operate in two instruction-based power-saving modes: Sleep and Idle. These modes can be entered by executing a special `PWRSAV` instruction. Sleep mode stops clock operation and halts all code execution. Idle mode halts the CPU and code execution but allows peripheral modules to continue operation. The assembler syntax of the `PWRSAV` instruction is shown in Example 32-1 and Example 32-2.

Note: SLEEP_MODE and IDLE_MODE are constants defined in the assembler include file for the selected device. Sleep and Idle modes can be exited as a result of an enabled interrupt, WDT time-out or a device Reset. When the device exits these modes, it is said to “wake-up.”

Example 32-1. `PWRSAV` Instruction Syntax in Assembly

```
PWRSAV #SLEEP_MODE    ; Put the device into Sleep mode
PWRSAV #IDLE_MODE     ; Put the device into Idle mode
```

Example 32-2. `PWRSAV` Instruction Syntax in C Language

```
Sleep()    // Put the device into Sleep mode
Idle ()    // Put the device into Idle mode
```

32.3.1.1 Interrupts Coincident with Power Save Instructions

Any interrupt that coincides with the execution of a `PWRSAV` instruction is held off until entry into Sleep or Idle mode has completed. If the interrupt is a wake-up event, it will then wake up the device and execute.

32.3.1.2 Sleep Mode

Sleep mode is the lowest Current Consumption state. The CPU and most peripherals are halted. Select peripherals can continue to operate in Sleep mode and can be used to wake the device from Sleep. See the individual peripheral module sections for descriptions of behavior in Sleep mode.

32.3.1.2.1 Sleep Mode Entry

1. The Primary Oscillator (POSC) and Auxuliary Oscillator are disabled.
2. The Secondary Oscillator (SOSC) continues to run if the Secondary Oscillator Enable bit (SOSC_EN) in the Oscillator Control register (OSCCTRL[3]) is set. For details, refer to the **"Oscillator Module"** section of the data sheet.
3. The WDT, clock source and LPRC Oscillator continue to run if the Watchdog Timer is enabled. The WDT, if enabled, is automatically cleared prior to entering Sleep mode. For details, see the **"Watchdog Timer"** section of the data sheet.
4. Any attempt to wake up the device while going into Sleep results in erratic behavior.
5. If any peripheral requests the FRC clock in Sleep mode, then FRC will remain active. Also, the FRC Oscillator can be operated in Low-Power mode. This is controlled by the FRCLPWR[1:0] bits in the Oscillator Configuration register (OSCCFG[17:16]). For details, refer to the **"Oscillator Module"** section.
6. The peripherals operating with the system clock are disabled. Optionally, the peripherals can operate in Sleep mode using specific clock sources.
7. The Fail-Safe Clock Monitor (FSCM) does not operate during Sleep mode because the system clock is disabled.

To minimize current consumption in Sleep mode, do the following:

- Ensure that I/O pins do not drive resistive loads
- Ensure that I/O pins configured as inputs are not floating
- Disable the SOSC
- Disable the WDT

32.3.1.2.2 Sleep Mode Exit

When the device exits Sleep mode, the CPU starts executing instructions within eight system clock cycles. System clock and peripheral clock are re-enabled. The device restarts with the current clock source as indicated by the Current Clock Source Selection bits (COSC[2:0]) in the Oscillator Control register (OSCCTRL[2:0]). Device is now in Normal Operation mode.

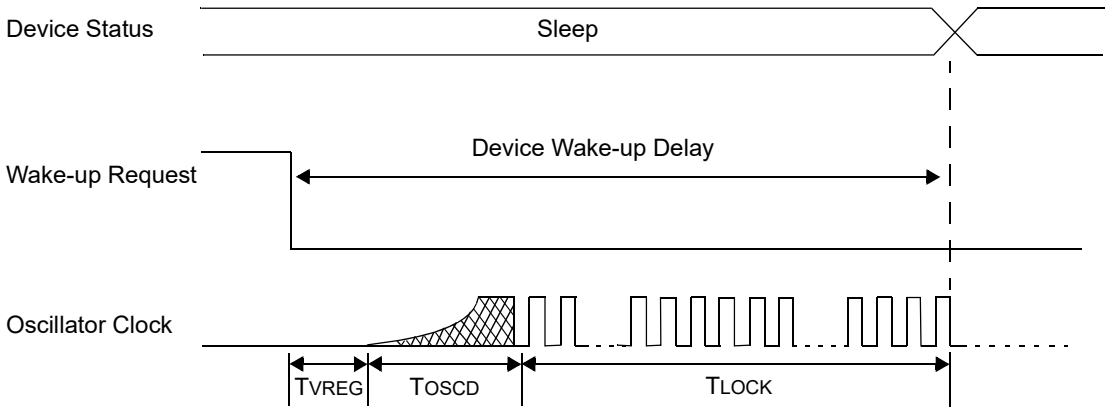
32.3.1.2.3 Delay on Wake-up from Sleep

Figure 32-1 shows the wake-up delay from Sleep mode. This delay consists of the voltage regulator delay and the oscillator delay:

- **Voltage Regulator Delay:** This is the time delay for the voltage regulator to transition from the Standby state to the Active state. This delay is required only if Standby mode is enabled for the voltage regulator.

- **Oscillator Delay:** The time delay for the clock to be ready for various clock sources is shown in Table 32-1. For details, refer to the **12. Oscillator and Clocking Module** section.

Figure 32-1. Wake-up Delay from Sleep Mode



Where: T_{VREG} = Voltage Regulator Standby to Active Mode Transition Time
T_{OSCD} = Oscillator Start-up Delay
T_{LOCK} = PLL Lock Time if PLL is Enabled

Table 32-1. Oscillator Delay(1,2)

Oscillator Source	Oscillator Start-up Delay	PLL Lock Time	Total Delay
FRC, FRCDIV16, FRCDIVN	T_{OSCD}	—	T_{OSCD}
FRCPLL	T_{OSCD}	T_{LOCK}	$T_{OSCD} + T_{LOCK}$
XT	T_{OSCD}	—	T_{OSCD}
HS	T_{OSCD}	—	T_{OSCD}
EC	—	—	—
XTPLL	T_{OSCD}	T_{LOCK}	$T_{OSCD} + T_{LOCK}$
HSPLL	T_{OSCD}	T_{LOCK}	$T_{OSCD} + T_{LOCK}$
ECPLL	—	T_{LOCK}	T_{LOCK}
SOSC	T_{OSCD}	—	T_{OSCD}
LPRC	T_{OSCD}	—	T_{OSCD}

Notes:
1. T_{OSCD} = Oscillator start-up delay. Crystal oscillator start-up time varies with crystal characteristics, load capacitance, etc.
2. T_{LOCK} = PLL lock time if PLL is enabled.

Note: Refer to the **37. Electrical Characteristics** section for T_{VREG} and T_{LOCK} specifications and also for the T_{OSCD} specifications when using the internal FRC or internal LPRC Oscillator.

32.3.1.3 Idle Mode

Idle mode has the following characteristics:

- The CPU stops executing instructions.
- The system clock source remains active. The clock generators, controlled by the CLKGENx register, can be put into Idle mode by setting the corresponding SIDL bit. Note that the SIDL bit is unimplemented in the CLKGEN1 register. For more details, refer to the **12. Oscillator and Clocking Module** section.

- The WDT is automatically cleared.
- If the WDT or FSCM is enabled, the LPRC remains active.
- The peripheral modules, by default, continue to operate normally from the system clock source.
- Peripherals can optionally be shut down using their Stop-in-Idle (SIDL) control bit, which is located in bit position 13 of the control register for most peripheral modules. The generic bit field name format is “xxxSIDL” (where “xxx” is the mnemonic name of the peripheral device). For more details, refer to the respective peripheral sections in the data sheet.
- The FRC Oscillator can be operated in Low-Power mode. This is controlled by the FRCLPWR[1:0] bits in the Oscillator Configuration register (OSCCFG[17:16]). For details, refer to the **12. Oscillator and Clocking Module** section.
- Flash can be put into Power-Down mode to reduce the power consumption. This is controlled by the NVMPIDL bit in the NVM Control register (NVMCON[12]). For details, refer to the **6. Flash Program Memory** section.

When the device exits Idle mode, the CPU starts executing instructions within eight system clock cycles.

32.3.1.4 Wake-up from Sleep and Idle

Sleep and Idle modes exit on the following events:

- An enabled interrupt event
- A WDT time-out
- Reset from any source (Power-on Reset, Brown-out Reset and $\overline{\text{MCLR}}$)

32.3.1.4.1 Wake-up on Interrupt

An enabled interrupt event wakes up the device from Sleep or Idle mode and then the following occurs:

- If the assigned priority for the interrupt is less than or equal to the current CPU priority, the device wakes up and continues code execution from the instruction following the `PWRSAV` instruction that initiated Sleep/Idle mode.
- If the assigned priority level for the interrupt source is greater than the current CPU priority, the device wakes up and the CPU exception process begins. Code execution continues from the first instruction of the ISR.

32.3.1.4.2 Wake-up on WDT Time-out

If enabled, the Watchdog Timer continues to run during Sleep mode or Idle mode. When the WDT time-out occurs, the device wakes up and code execution continues from where the `PWRSAV` instruction was executed.

The Watchdog Timer Time-out Flag bit (WDTO) in the Reset Control register (RCON[4]) is set to indicate that the wake-up event is due to a WDT time-out.

32.3.1.4.3 Wake-up on Reset

A Reset from any source (Power-on Reset, Brown-out Reset and $\overline{\text{MCLR}}$) causes the device to exit Sleep or Idle mode and begin executing from the Reset vector.

32.3.2 Peripheral Module Disable (PMD)

All peripheral modules (except for I/O ports) in dsPIC33A devices have a control bit that can be selectively disabled to reduce power consumption. These bits, known as the Peripheral Module Disable (PMD) bits, are generically named “xxxMD” (where “xxx” is the mnemonic version of the module’s name). These bits are located in the PMDx Special Function Registers (SFRs). The PMD bit must be set (= `1`) to disable the module. The PMD bit completely shuts down the peripheral, effectively powering down all circuits and removing all clock sources. By default all peripherals are not disabled by PMD.

32.3.3 Clock Frequency and Clock Switching

The dsPIC33AK128MC106 family devices allow a wide range of clock frequencies to be selected under application control. If the system clock configuration is not locked, users can choose low-power or high-precision oscillators by setting NOSCx bits (OSCCON[10:8]) and setting the OSCSWEN bit, as well as addressing any CLKxDIV changes required.. The process of changing a system clock during operation, as well as limitations to the process, are discussed in more detail in 12. Oscillator and Clocking Module.

33. JTAG Interface

The dsPIC33AK128MC106 family of devices support Boundary Scan through a JTAG interface. However, programming is not supported through the JTAG Interface.

34. In-Circuit Debugger

When a supported emulator/debugging tool, such as the MPLAB® ICD5, is selected as a debugger, the in-circuit debugging functionality is enabled. This function allows simple debugging functions when used with MPLAB X IDE. Debugging functionality is controlled through the PGCx (Emulation/Debug Clock) and PGDx (Emulation/Debug Data) pin functions.

Any of the three pairs of debugging clock/data pins can be used:

- PGC1 and PGD1
- PGC2 and PGD2
- PGC3 and PGD3

To use the in-circuit debugger function of the device, the design must implement ICSP connections to $\overline{MCLR}$, V_{DD}, V_{SS} and the PGCx/PGDx pin pair.

35. Instruction Set Summary

Note: This data sheet summarizes the features of the dsPIC33AK128MC106 family of devices. It is not intended to be a comprehensive reference source. To complement the information in this data sheet, refer to the *"dsPIC33A Programmer's Reference Manual"*, which is available from the Microchip website (www.microchip.com).

The instruction set is highly orthogonal and is grouped into five basic categories:

- Word or byte-oriented operations
- Bit-oriented operations
- Literal operations
- DSP operations
- Control operations

Table 35-1 lists the general symbols used in describing the instructions.

The dsPIC33 instruction set summary in Table 35-2 lists all the instructions, along with the status flags affected by each instruction.

Most word or byte-oriented W register instructions (including barrel shift instructions) have three operands:

- The first source operand, which is typically a register 'Wb' without any address modifier
- The second source operand, which is typically a register 'Ws' with or without an address modifier
- The destination of the result, which is typically a register 'Wd' with or without an address modifier

However, word or byte-oriented file register instructions have two operands:

- The file register specified by the value 'f'
- The destination, which could be either the file register 'f' or any W register

Most bit-oriented instructions (including simple rotate/shift instructions) have two operands:

- The W register (with or without an address modifier) or file register (specified by the value of 'Ws' or 'f')
- The bit in the W register or file register (specified by a literal value or indirectly by the contents of register 'Wb')

The literal instructions that involve data movement can use some of the following operands:

- A literal value to be loaded into a W register (specified by 'k')
- The W register or file register where the literal value is to be loaded (specified by 'Wb' or 'f')

However, literal instructions that involve arithmetic or logical operations use some of the following operands:

- The first source operand, which is a register 'Wb' without any address modifier
- The second source operand, which is a literal value
- The destination of the result (only if not the same as the first source operand), which is typically a register 'Wd' with or without an address modifier

The MAC class of DSP instructions can use some of the following operands:

- The accumulator (A or B) to be used (required operand)
- The W registers to be used as the two operands
- The X and Y address space prefetch operations

- The X and Y address space prefetch destinations
- The accumulator write-back destination

The other DSP instructions do not involve any multiplication and can include:

- The accumulator to be used (required)
- The source or destination operand (designated as Wso or Wdo, respectively) with or without an address modifier
- The amount of shift specified by a W register 'Wn' or a literal value

The control instructions can use a program memory address.

Most single-word instructions are executed in a single instruction cycle unless a conditional test is true, the Program Counter is changed as a result of the instruction. In these cases, the execution takes multiple instruction cycles with the additional instruction cycle(s) executed as a `NOP`. Certain instructions that involve skipping over the subsequent instruction require extra cycles if the skip is performed, depending on whether the instruction being skipped is a single-word or two-word instruction.

Note: For more details on the instruction set, refer to the *"dsPIC33A Programmer's Reference Manual (DS70005540)"* (www.microchip.com/).

Table 35-1. Symbols Used in Opcode Descriptions

Symbol[1]	Description
`{ }`	Optional field or operation
`[text]`	The location addressed by text
`(text)`	The contents of text
`#text`	The literal defined by text
`{label:}`	Optional label name
`[n:m]`	Register bit field
`.l`	32-bit Long Word mode selection
`.b`	8-bit Byte mode selection
`.sl`	24-bit (literal) Word mode selection
`.V`	Destination data value select (MAXABW, MINABW and FLIMW)
`.w`	16-bit Word mode selection (default)
`AWB`	Accumulator write back destination address register
`bit3`	3-bit bit selection field (used in byte addressed instructions) (0:7)
`bit4`	4-bit bit selection field (used in word addressed instructions) (0:15)
C, N, OV, Z	ALU status bits: Carry, Digit Carry, Negative, Overflow, Zero
d	File register destination (W0, none)
`Expr`	Absolute address, label or expression (resolved by the linker)
`f`	File register address (0x0000:0xFFFF) or (0x00000:0xFFFFF) (addressable space varies depending upon instruction class)
`Fd`[2]	One of 32 FPU destination data registers (F0:F31) (Register Direct)
`Fs`[2]	One of 32 FPU source data registers (F0:F31) (Register Direct)
`FSR`, `FSRH`, `FCR`, `FEAR 1`[2]	FPU special (control & status) coprocessor registers (Register Direct)
`label`	Translates to a literal representing the location of the label name
`lit1`	1-bit unsigned literal (0:1)
`lit3`	3-bit unsigned literal (0:7)
`lit5`	5-bit unsigned literal (0:31)
`lit6`	6-bit unsigned literal (0:63)

.....continued

Symbol(1)	Description
lit8	8-bit unsigned literal (0:255)
lit16	16-bit unsigned literal (0:65535)
lit24	24-bit unsigned literal (0:1677215; LSB must be 0 if an address)
lit32	32-bit unsigned literal (0:4294967295)
none	Field does not require an entry and may be blank
OA, OB, SA, SB	DSC status bits: ACCA Overflow, ACCB Overflow, ACCA Saturate, ACCB Saturate
PC	Program Counter
Rdo	Destination Working register
Rnd	Instruction rounding mode [E, Z, P, N]
Rso	Source Working register
Slit6	Signed 6-bit literal (-32:31)
Slit7	Signed 7-bit literal (-64:63)
Slit8	Signed 8-bit literal (-128:127)
Slit20	Signed 20-bit literal (-524288:524287)
SR	Status Register
text1 ∈ {text2, {text3,…}	text1 must be in the set of text2, text3, ...
v	Selects MULxxx operand data types
Wb	Base Working register
Wd	Destination Working register
Wm	One of 16 Working registers (W0:W15)
Wn	Both source and destination Working register (W0:W15)
Wnd	One of 16 destination Working registers
Wns	One of 16 source Working registers
Wm * Wm	Multiplicand and Multiplier Working register for Square instructions
Wm * Wn	Multiplicand and Multiplier W register for DSP instructions
Ws	Source Working register
Wx	X data space fetch address register for DSP instructions
Wy	Y data space fetch address register for DSP instructions

Notes:

1. The range of each symbol is instruction-dependent.
2. Only applicable when the FPU coprocessor is present.
3. Read and Read-Modify-Write (RMW) operations on non-CPU Special Function Registers require additional cycles when updating SFRs belonging to peripherals on different peripheral clock speeds. The bus speed should be taken into consideration on time sensitive writes.

Table 35-2. Instruction Set Overview

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
1	ABS	ABS Fs, Fd	Absolute value of Fs (FPU Instr)	1	1	SUBO

.....continued

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
2	ADD	ADD A	Add Accumulators	0.5	1	OA, SA, OB, SB
		ADD Rso,#Slit6, A	16-bit Signed Add to Accumulator	1	1	OA, SA, OB, SB
		ADD f,Wn	f = f + Wn	1	1	C, N, OV, Z
		ADD f,Wn,Wn	Wn = f + Wn	1	1	C, N, OV, Z
		ADD.l #lit5,Wn	Wn = Wn + lit5	0.5	1	C, N, OV, Z
		ADD #lit16,Wn	Wn = Wn + lit16	0.5/1	1	C, N, OV, Z
		ADD Wb,Ws,Wd	Wd = Wb + Ws	1	1	C, N, OV, Z
		ADD Wb,#lit7,Wd	Wd = Wb + lit7 (literal zero-extended)	1	1	C, N, OV, Z
		ADD Fb, Fs, Fd	Fd = Fb + Fs (FPU Instr)			SUBO, INX, UDF, OVF, INVAL
3	ADDC	ADDC f,Wn	f = f + Wn + (C)	1	1	C, N, OV, Z
		ADDC f,Wn,Wn	Wn = f + Wn + (C)	1	1	C, N, OV, Z
		ADDC #lit16,Wn	Wn = Wn + lit16 + (C)	1	1	C, N, OV, Z
		ADDC Wb,Ws,Wd	Wd = Wb + Ws + (C)	0.5/1	1	C, N, OV, Z
		ADDC Wb,#lit7,Wd	Wd = Wb + lit7 + (C)(literal zero-extended)	1	1	C, N, OV, Z
4	AND	AND f,Wn	f = f .AND. Wn	1	1	N, Z
		AND f,Wn,Wn	W0 = f .AND. Wn	1	1	N, Z
		AND #lit16,Wn	Wn = Wn .AND. lit16	1	1	N, Z
		AND Wb,Ws,Wd	Wd = Wb .AND. Ws	0.5/1	1	N, Z
		AND Wb,#lit7,Wd	Wd = Wb .AND. Lit7 (literal zero-extended)	1	1	N, Z
		AND1 Wb,#lit7,Wd	Wd = Wb .AND. Lit7 (literal zero-extended)	1	1	N, Z
		AND #lit16, FSR	FSR = FSR AND lit16 (FPU instr)	1	1	SUBO, HUGI, INX, UDF, OVF, DIV0, INVAL
		AND #lit16, FCR	FCR = FCR AND lit16 (FPU instr)	1	1	None
		AND #lit16, FEAR	FEAR = FEAR AND lit16 (FPU instr)	1	1	None
5	ASR	ASR f	f = Arithmetic Right Shift f by 1	1	1	N, Z
		ASR f,Wn	Wn = Arithmetic Right Shift f by 1	1	1	N, Z
		ASR Ws,Wd	Wd = Arithmetic Right Shift Ws by 1	0.5/1	1	N, Z
		ASR Ws,Wb,Wd	Wnd = Arithmetic Right Shift Ws by Wb	0.5/1	1	N, Z
		ASR Ws,lit5,Wd	Wnd = Arithmetic Right Shift Ws by lit5	0.5/1	1	N, Z
6	ASRM	ASRM Ws, #lit5, Wnd	Wnd = Arithmetic Right Shift Ws by lit5, then logically OR with next lsw	1	2	N, Z
		ASRM Ws, Wb, Wnd	Wnd = Arithmetic Right Shift Ws by Wb, then logically OR with next lsw	1	2	N, Z
7	BCLR	BCLR.b f,bit3	Bit Clear f	1	1	None
		BCLR Ws,bit4	Bit Clear Ws	0.5/1	1	None
8	BFEXT	BFEXT bit4,wid5,Ws,Wb	Bit Field Extract from Ws to Wb	1	1	None
		BFEXT bit4,wid5,f,Wb	Bit Field Extract from f to Wb	2	2	None
9	BFINS	BFINS bit4,wid5,Wb,Ws	Bit Field Insert from Wb into Ws	1	1	None
		BFINS bit4,wid5,Wb,f	Bit Field Insert from Wb into f	2	2	None
		BFINS bit4,wid5,#lit8,Ws	Bit Field Insert lit8 into Ws	2	2	None
10	BOOTSWP	BOOTSWP	Swap the Active and Inactive Program Flash Space	1	2	None

.....continued

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
11	BRA	BRA Label	Branch Unconditionally	1	1	None
		BRA Wn	Computed Branch	1	2	None
		BRA C,Label	Branch if Carry	1	1(2/3)	None
		BRA GE,Label	Branch if greater than or equal	1	1(2/3)	None
		BRA GEU,Label	Branch if unsigned greater than or equal	1	1(2/3)	None
		BRA GT,Label	Branch if greater than	1	1(2/3)	None
		BRA GTU,Label	Branch if unsigned greater than	1	1(2/3)	None
		BRA LE,Label	Branch if less than or equal	1	1(2/3)	None
		BRA LEU,Label	Branch if unsigned less than or equal	1	1(2/3)	None
		BRA LT,Label	Branch if less than	1	1(2/3)	None
		BRA LTU,Label	Branch if unsigned less than	1	1(2/3)	None
		BRA N,Label	Branch if Negative	1	1(2/3)	None
		BRA NC,Label	Branch if Not Carry	1	1(2/3)	None
		BRA NN,Label	Branch if Not Negative	1	1(2/3)	None
		BRA NOV,Label	Branch if Not Overflow	1	1(2/3)	None
		BRA NZ,Label	Branch if Not Zero	1	1(2/3)	None
		BRA Z,Label	Branch if Zero	1	1(2/3)	None
		BRA OA,Label	Branch if accumulator A overflow	1	1(2/3)	None
		BRA OB,Label	Branch if accumulator B overflow	1	1(2/3)	None
		BRA OV,Label	Branch if Overflow	1	1(2/3)	None
		BRA SA,Label	Branch if accumulator A saturated	1	1(2/3)	None
		BRA SB,Label	Branch if accumulator B saturated	1	1(2/3)	None
12	BREAK	BREAK	Stop User Code Execution	1	1	None
13	BSET	BSET.b f,bit3	Bit Set f	1	1	None
		BSET Ws,bit4	Bit Set Ws	0.5/1	1	None
14	BSW	BSW.C Ws,Wb	Write C or Z bit to Ws<Wb>	1	1	None
		BSW.Z Ws,Wb	Write C or Z bit to Ws<Wb>	0.5/1	1	None
15	BTG	BTG.b f,bit3	Bit Toggle f	1	1	None
		BTG Ws,bit4	Bit Toggle Ws	0.5/1	1	None
16	BTSC		Not supported			N/A
17	BTSS		Not supported			N/A
18	BTST	BTST.b f,bit3	Bit Test f	1	1	Z
		BTST.C Ws,bit4	Bit Test Ws to C	0.5/1	1	C, Z
		BTST.Z Ws,bit4	Bit Test Ws to Z	1	1	Z
		BTST.C Ws,Wb	Bit Test Ws<Wb> to C	0.5/1	1	C, Z
		BTST.Z Ws,Wb	Bit Test Ws<Wb> to Z	1	1	Z
19	BTSTS	BTSTS.b f,bit3	Bit Test then Set f	1	1	Z
		BTSTS.C Ws,bit4	Bit Test Ws to C then Set	0.5/1	1	C, Z
		BTSTS.Z Ws,bit4	Bit Test Ws to Z then Set	1	1	Z
20	CALL	CALL Label	Call subroutine (label > ~ 16MB)	1	1	None
			Call subroutine (label < ~ 16MB)	2	2	None
		CALL Wns	Call indirect subroutine at address [W11]	1	2	None
21	CLR	CLR f	f = 0x0000 0000	1	1	None
		CLR Wd	Wd = 0x0000 0000	1	1	None
		CLR A	Clear Accumulator	0.5	1	None
21	CLRWDT	CLRWDT	Clear Watchdog Timer	0.5	1	WDTO, Sleep
22	COM	COM f	f = f	1	1	N, Z
		COM f,Wd	Wd = f	1	1	N, Z
		COM Ws,Wd	Wd = Ws	0.5/1	1	N, Z

.....continued

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
23	CP	CP f,Ws	Compare f with Ws	1	1	C, N, OV, Z
		CP Ws,#lit13	Compare Ws with lit13 (literal zeroextended)	1	1	C, N, OV, Z
		CP Wb,#lit16	Compare Wb with lit16 (literal zeroextended)	1	1	C, N, OV, Z
		CP Wb, Ws	Compare Wb with Ws	0.5/1	1	C, N, OV, Z
24	CP0	CP0 f	Compare f with 0x0000 0000	1	1	C, N, OV, Z
		CP0 Ws	Compare Ws with 0x0000 0000 (substitute CPLS Ws ,#0)	1	1	C, N, OV, Z
25	CPB	CPB f,Ws	Compare f with Ws, with borrow	1	1	C, N, OV, Z
		CP Wb,#lit13	Compare Wb with lit13, with borrow (literal zero-extended)	1	1	C, N, OV, Z
		CP Wb,#lit16	Compare Wb with lit16, with borrow (literal zero-extended)	1	1	C, N, OV, Z
		CPB Wb,Ws	Compare Borrow Wb with Ws	0.5/1	1	C, N, OV, Z
	CPSEQ		Not supported			N/A
	CPBEQ		Not supported			N/A
	CPSGT		Not supported			N/A
	CPBGT		Not supported			N/A
	CPSNE		Not supported			N/A
26	CTXTSWP	CTXTSWP #lit3	Swap to CPU register context defined by lit3	0.5	2	None
		CTXTSWP Wn	Swap to CPU register context defined in Wn[2:0]	1	2	None
27	DAW		Not supported			N/A
28	DEC	DEC f	f = f -1	1	1	C, N, OV, Z
		DEC f,Wd	W5 = f -1	1	1	C, N, OV, Z
		DEC Ws,Wd	Wd = Ws - 1	1	1	C, N, OV, Z
29	DEC2	DEC2 f	f = f -2	1	1	C, N, OV, Z
		DEC2 f,Wd	W5 = f -2	1	1	C, N, OV, Z
		DEC2 Ws,Wd	Wd = Ws - 2	1	1	C, N, OV, Z
30	DISI		Not supported			N/A
31	DISICTL	DISICTL #lit3 {,Wd}	Disable interrupts at IPL <= lit3 Optionally save prior IPL threshold to Wd	1	1	None
		DISICTL Wns {,Wd}	Disable interrupts at IPL <= Wns[2:0] Optionally save prior IPL threshold to Wd	1	1	None
32	DIVF	DIVF Wm/Wn	Interruptible Signed 16/16 or 32/16 Fractional Divide	1	1	C, N, OV, Z
33	DIVFL	DIVFL Wm/Wn	Interruptible Signed 32/32 Fractional Divide	1	1	C, N, OV, Z
34	DIVS	DIVS.w Wm/Wn	Interruptible Signed 16/16-bit Integer Divide	1	1	C, N, OV, Z
		DIVS.l Wm/Wn	Interruptible Signed 32/16-bit Integer Divide	1	1	C, N, OV, Z
35	DIVSL	DIVSL Wm/Wn	Interruptible Signed 32/32 Integer Divide	1	1	C, N, OV, Z
36	DIVU	DIVU.w Wm/Wn	Interruptible Unsigned 16/16-bit Integer Divide	1	1	C, N, OV, Z
		DIVU.l Wm/Wn	Interruptible Unsigned 32/16-bit Integer Divide	1	1	C, N, OV, Z
37	DIVUL	DIVUL Wm/Wn	Interruptible Unsigned 32/32 Integer Divide	1	1	C, N, OV, Z
38	DIV2		Not supported			N/A
39	DTB	DTB Wn,Label	Decrement Wn, then branch if not zero	1	1(2/3)	None

.....continued

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
40	DO		Not supported			N/A
41	ED	ED Wxp * Wyp, A, AWB	Euclidean Distance	1	2	OA, SA, OB, SB
42	EDAC	EDAC Wxp * Wyp, A, AWB	Euclidean Distance Accumulate	1	2	OA, SA, OB, SB
43	EXCH	EXCH Wns,Wnd	Swap Wns with Wnd	1	2	None
44	FBCL	FBCL Ws,Wnd	Find Bit Change from Left (MSb) Side	1	1	C
45	FF1L	FF1L Ws,Wnd	Find First One from Left (MSb) Side	1	1	C
46	FF1R	FF1R Ws,Wnd	Find First One from Right (LSb) Side	1	1	C
47	FLIM	FLIM Wb, Ws	Force Data (Upper and Lower) Range Limit without Limit Excess Result	1	2	N, Z, OV
		FLIM Wb, Ws, Wd	Force Data (Upper and Lower) Range Limit with Limit Excess Flag (Wd=-1)	1	2	N, Z, OV
		FLIM.V Wb, Ws, Wd	Force Data (Upper and Lower) Range Limit with Limit Excess Result	1	2	N, Z, OV
48	GOTO	GOTO Label	Goto address (address < ~ 16MB)	1	2	None
		GOTO Label	(label < ~ 16MB)	2	2	None
		GOTO Wn	Go to indirect address at [W11]	1	2	None
49	INC	INC f	f = f + 1	1	1	C, N, OV, Z
		INC f,Wd	W5 = f + 1	1	1	C, N, OV, Z
		INC Ws,Wd	Wd = Ws + 1	1	1	C, N, OV, Z
50	INC2	INC2 f	f = f + 2	1	1	C, N, OV, Z
		INC2 f,Wd	W5 = f + 2	1	1	C, N, OV, Z
		INC2 Ws,Wd	Wd = Ws +2	1	1	C, N, OV, Z
51	IOR	IOR f,Wn	f = f .IOR. Wn	1	1	N, Z
		IOR f,Wn,Wn	Wn = f .IOR. Wn	1	1	N, Z
		IOR #lit16,Wn	Wn = Wn .IOR. lit16	1	1	N, Z
		IOR Wb,Ws,Wd	Wd = Wb .IOR. Ws	0.5/1	1	N, Z
		IOR Wb,#lit7,Wd	Wd = Wb .IOR. lit7	1	1	N, Z
52	LAC	LAC Rso,#Slit6, A	Load Accumulator (16/32-bit), literal shift	1	1	OA, SA, OB, SB
	LLAC	LLAC.l Rso,#Slit6, A	Load Lower (lsw of) Accumulator (32-bit), literal shift	1	1	OA, SA, OB, SB
	LAUC	LUAC.l Rso,#Slit6, A	Load Upper (LSB) of Accumulator (32-bit), literal shift	1	1	OA, SA, OB, SB
53	LNK	LNK #lit16	Link frame pointer	1	1	None
		LNK #lit7	Link frame pointer (literal < 128)	0.5	1	None
54	LSR	LSR f	f = Logical Right Shift f by 1	1	1	C,N,Z
		LSR f,Wd	Wd = Logical Right Shift f by 1	1	1	C,N,Z
		LSR Ws,Wd	Wd = Logical Right Shift Ws by 1	0.5/1	1	C,N,Z
		LSR Ws,Wb,Wd	Wnd = Logical Right Shift Ws by Wns	0.5/1	1	C,N,Z
		LSR Ws,#lit5,Wd	Wnd = Logical Right Shift Ws by lit5	0.5/1	1	N,Z
		LSRM Ws,#lit5, Wnd	Wnd = Logical Right Shift Ws by lit5, then logically OR with next lsw	1	2	N,Z
		LSRM Ws, Wb, Wnd	Wnd = Logical Right Shift Ws by Wb, then logically OR with next lsw	1	2	N,Z
55	MAC	MAC Wxp * Wyp, A, AWB	Multiply and Accumulate	1	1	OA, SA, OB, SB
56	MAX	MAX Wb, Ws	Force Data Maximum Range Limit	1	1	OA, SA, OB, SB
		MAX A	Force Data Maximum Range Limit	0.5	1	N, Z, OV
		MAX.V A, Rdo	Force Data Maximum Range Limit with Result	1	2	N, Z, OV
57	MIN	MIN Wb, Ws	Force Data Minimum Range Limit	1	1	N, Z, OV
		MIN A	Force Data Minimum Range Limit	0.5	1	N, Z, OV
		MIN.V A, Rdo	Force Data Minimum Range Limit with Result	1	2	N, Z, OV

.....continued

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
58	MOV	MOV Rso,Rdo	Move Ws to Wd	0.5/1	1	None
		MOV.l #lit32,Wnd	Move 32-bit unsigned literal to Wnd	2	2	None
		MOV.sl #lit24,Wnd	Move 24-bit unsigned literal to Wnd; 0 extend to 32-bits	1	1	None
		MOV.w #lit16,Wnd	Move 16-bit unsigned literal to Wnd; 0 extend to 32-bits	1	1	None
		MOV.bz #lit8,Wnd	Move 8-bit unsigned literal to Wnd; 0 extend to 32-bits	1	1	None
		MOV.l [W15-#lit7], Wnd [W14+#slit7], Wnd	Move from system stack with literal offset to Wnd using SP or FP	0.5	1	None
		MOV.l Wns, [W15-#lit7] Wns, [W14+#slit7]	Move from Wns to system stack with literal off-set using SP or FP	0.5	1	None
		MOV.l f,Wnd	Move f to Wnd (Word or Long Word)(f < ~1MB)	1	1	None
		MOV.w f,Wnd	Move f to Wnd (Word or Long Word)(f > ~1MB)	2	2	None
		MOV.b f,Wnd	Move f to Wnd (Byte)	1	1	None
		MOV.l Wns,f	Move Wns to f (Word or Long Word)(f < ~1MB)	1	1	None
		MOV.w Wns,f	Move Wns to f (Word or Long Word)(f > ~1MB)	2	2	None
		MOV.b Wns,f	Move Wns to f (Byte)	1	1	None
		MOV [Wns+#Slit12],Wnd	Move [Wns+Slit12] to Wnd	1	1	None
		MOV Wns, [Wnd+#Slit12]	Move Wns to [Wnd+Slit12]	1	1	None
		MOVIF.l CC, Wb, Wns, Wd	If SR.Z=1 Move W1 to [W15++] Else Move W2 to [W15++]	1	1	None
		MOVIF.w CC, Wb, Wns, Wd	If SR.Z=1 Move W1 to [W15++] Else Move W2 to [W15++]	1	1	None
		MOVIF.bz CC, Wb, Wns, Wd	If SR.Z=1 Move W1 to [W15++] Else Move W2 to [W15++]	1	1	None
		MOVIF.b CC, Wb, Wns, Wd	If SR.Z=1 Move W1 to [W15++] Else Move W2 to [W15++]	1	1	None
		MOVR.l	Move Ws to Wd with destination Bit Reversed	1	1	None
		MOVR.w	Move Ws to Wd with destination Bit Reversed	1	1	None
		MOVS.l #slit16, Wd	Move signed extended 16-bit literal to Wd	1	1	None
		MOVS.w #slit16, Wd	Move 16-bit literal to Wd; sign extend to 32-bits if register direct mode.	1	1	None
		MOVS.b #slit8, Wnd	Move 8-bit literal to Wd; no extension.	1	1	None
59	MOVPAG		Not supported			N/A
60	MOVSAC		Not supported			N/A
61	MPY	MPY Wxp * Wyp, A, AWB	Multiply Wm by Wn to Accumulator	1	1	OA, SA, OB, SB
62	MPYN	MPYN Wxp * Wyp, A, AWB	(negative)(Multiply Wm by Wn) to Accumulator	1	1	OA, SA, OB, SB
63	MSC	MSC Wxp * Wyp, A, AWB	Multiply and Subtract from Accumulator	1	1	OA, SA, OB, SB

.....continued

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
65	MUL	MUL f, Wn	W2 = f * Wn	1	1	None
		MULISS Wb,Ws,A	Integer: Acc(A or B) = signed(Wb) * signed(Ws)	1	1	None
		MULFSS Wb,Ws,A	Fractional: Acc(A or B) = signed(Wb) * signed(Ws)	1	1	None
		MULISU Wb,Ws,A	Integer: Acc(A or B) = signed(Wb) * unsigned(Ws)	1	1	None
		MULFSU Wb,Ws,A	Fractional: Acc(A or B) = signed(Wb) * unsigned(Ws)	1	1	None
		MULIUS Wb,Ws,A	Integer: Acc(A or B) = unsigned(Wb) * signed(Ws)	1	1	None
		MULFUS Wb,Ws,A	Fractional: Acc(A or B) = unsigned(Wb) * signed(Ws)	1	1	None
		MULIUU Wb,Ws,A	Integer: Acc(A or B) = unsigned(Wb) * unsigned(Ws)	1	1	None
		MULFUU Wb,Ws,A	Fractional: Acc(A or B) = unsigned(Wb) * unsigned(Ws)	1	1	None
		MULISS Wb,#slit8,A	Integer: Acc(A or B) = signed(Wb) * signed(slit8)	1	1	None
		MULFSS Wb,#slit8,A	Integer: Acc(A or B) = signed(Wb) * signed(slit8)	1	1	None
		MULISU Wb,#lit8,A	Integer: Acc(A or B) = signed(Wb) * unsigned(lit8)	1	1	None
		MULFSU Wb,#lit8,A	Integer: Acc(A or B) = signed(Wb) * unsigned(lit8)	1	1	None
		MULIUS Wb,#slit8,A	Integer: Acc(A or B) = signed(Wb) * signed(slit8)	1	1	None
		MULFUS Wb,#slit8,A	Integer: Acc(A or B) = signed(Wb) * signed(slit8)	1	1	None
		MULIUU Wb,#lit8,A	Integer: Acc(A or B) = signed(Wb) * unsigned(lit8)	1	1	None
		MULFUU Wb,#lit8,A	Integer: Acc(A or B) = signed(Wb) * unsigned(lit8)	1	1	None
		MULSS Wb,Ws,Wnd	{Wd}=signed(Wb) * signed(Ws)	0.5/1	1	None
		MULSU Wb,Ws,Wnd	{Wd}=signed(Wb) * unsigned(Ws)	0.5/1	1	None
		MULUS Wb,Ws,Wnd	{Wd}=unsigned(Wb) * signed(Ws)	0.5/1	1	None
		MULUU Wb,Ws,Wnd	{Wd}=unsigned(Wb) * unsigned(Ws)	0.5/1	1	None
		MULSU Wb,#lit8,Wnd	{Wd}=signed(Wb) * unsigned(lit8)	1	1	None
		MULUU Wb,#lit8,Wnd	{Wd}=unsigned(Wb) * unsigned(lit8)	1	1	None
		MULSS Wb,#slit8,Wnd	{Wd}=signed(Wb) * signed(slit8)	1	1	None
		MULUS Wb,#slit8,Wnd	{Wd}=unsigned(Wb) * signed(slit8)	1	1	None
66	NEG	NEG A	Negate Accumulator	1	1	OA, SA, OB, SB
		NEG f	f = f +1	1	1	OA, SA, OB, SB
		NEG f,Wd	Wd = f + 1	1	1	OA, SA, OB, SB
		NEG Ws,Wd	Wd = Ws + 1	1	1	OA, SA, OB, SB
67	NEOP	NEOP	None executable NOP (16-bit instruction pad)	0.5	0	None
68	NOP	NOP	No Operation	1	1	None
		NOPR	No Operation	1	1	None
69	NORM	NORM A, Rdo	Normalize Accumulator	1	1	N,OV,Z
70	POP	POP f	Pop f from top of stack (TOS)	1	1	None
		POP {[--Ws],} Wnd	Pop Wnd Register from system stack.	0.5	1	None
		POP Fd	Pop Fd Register from system stack.	0.5	1	None
71	PUSH	PUSH f	Push f to top of stack (TOS)	1	1	None
		PUSH Wns, {[Wd++]}	Push Wns Register to system stack	0.5	1	None
		PUSH Fs	Push Fs Register to system stack	0.5	1	None

.....continued

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
72	PWRSAV	PWRSAV mode	Go into standby mode	0.5	2	WDTO, Sleep
	RCALL	RCALL Label	Relative Call	1	1	None
		RCALL Wns	Computed Call	1	2	None
73	REPEAT	REPEAT #lit15	Repeat Next Instruction lit15+1 times	1	1	RA (if lit15 > 0)
		REPEAT #lit5	Repeat Next Instruction lit5+1 times	0.5	1	RA (if lit5 > 0)
		REPEAT Wn	Repeat Next Instruction (Wn)+1 times	1	1	RA (if Wn > 0)
74	RESET	RESET	Software Device Reset	1	1	None
75	RETFIE	RETFIE	Return from interrupt enable	1	4	None
76	RETLW lit16,Wn	RETLW #lit16,Wn	Return from Subroutine with literal in Wn	1	3	None
77	RETURN	RETURN	Return from Subroutine	0.5	3	None
78	RLC	RLC f	f = Rotate Left through Carry f	1	1	C, N, Z
		RLC f,Wd	Wd = Rotate Left through Carry f	1	1	C, N, Z
		RLC Ws,Wd	Wd = Rotate Left through Carry Ws	0.5/1	1	C, N, Z
79	RLNC	RLNC f	f = Rotate Left (No Carry) f	1	1	N, Z
		RLNC f,Wd	Wd = Rotate Left (No Carry) f	1	1	N, Z
		RLNC Ws,Wd	Wd = Rotate Left (No Carry) Ws	0.5/1	1	N, Z
80	RRC	RRC f	f = Rotate Right through Carry f	1	1	C, N, Z
		RRC f,Wd	Wd = Rotate Right through Carry f	1	1	C, N, Z
		RRC Ws,Wd	Wd = Rotate Right through Carry Ws	0.5/1	1	C, N, Z
81	RRNC	RRNC f	f = Rotate Right (No Carry) f	1	1	N, Z
		RRNC f,Wd	Wd = Rotate Right (No Carry) f	1	1	N, Z
		RRNC Ws,Wd	Wd = Rotate Right (No Carry) Ws	0.5/1	1	N, Z
82	SAC	SAC A,#Slit6,Rdo	Store Accumulator (16/32-bit)	1	1	None
		SACR A,#Slit6,Rdo	Store Rounded Accumulator (16/32-bit), literal shift	1	1	None
		SACRW A,Ws,Rdo	Store Rounded Accumulator (16/32-bit), Wb shift	1	1	None
83	SE	SE Rso,Wnd	Wd = sign-extended Ws	0.5/1	1	C, N, Z
84	SETM	SETM f	f = 0xFFFF FFFF	1	1	None
		SETM Wd	Wd = 0xFFFF FFFF	1	1	None
85	SFTAC	SFTAC A,Wn	Arithmetic Shift by (Wn) Accumulator	1	1	OA, SA, OB, SB
		SFTAC A,#Slit7	Arithmetic Shift by Slit7 Accumulator	1	1	OA, SA, OB, SB
86	SL	SL f	f = Left Shift f by 1	1	1	C, N, Z
		SL f,Wd	Wd = Left Shift f by 1	1	1	C, N, Z
		SL Ws,Wd	Wd = Left Shift Ws by 1	0.5/1	1	C, N, Z
		SL Ws,Wb,Wnd	Wnd = Left Shift Wb by Wns	0.5/1	1	C, N, Z
		SL Ws,#lit5,Wnd	Wnd = Left Shift Ws by lit5	0.5/1	1	C, N, Z
	SLM	SLM Ws, #lit5, Wnd	Wnd = Left Shift Wb by lit5, then logically OR with next msw	1	2	Z
		SLM Ws, Wb, Wnd	Wnd = Left Shift Wb by Wb, then logically OR with next msw	1	2	Z
87	SLAC	SLAC.l A,#Slit6,Rdo	Store Lower (lsw of) Accumulator (32-bit), literal shift	1	1	None
88	SUAC	SUAC.l A,#Slit6,Rdo	Store sign extended Upper (MSB) Accumulator (32- bit), literal shift	1	1	None
89	SUB	SUB A	Subtract Accumulators	0.5	1	OA, SA, OB, SB
		SUB Rso,#Slit6, A	16-bit Signed Subtract from Accumulator	1	1	OA, SA, OB, SB
		SUB f,Wn	f = f - Wn	1	1	C, N, OV, Z
		SUB f,Wn,Wn	Wn = f - Wn	1	1	C, N, OV, Z
		SUB.l #lit5,Wn	Wn = Wn - lit5	0.5	1	C, N, OV, Z
		SUB #lit16,Wn	Wn = Wn - lit16	1	1	C, N, OV, Z
		SUB Wb,Ws,Wd	Wd = Wb - Ws	0.5/1	1	C, N, OV, Z
		SUB Ws,#lit7,Wd	Wd = Ws - lit7 (literal zero-extended)	1	1	C, N, OV, Z

.....continued

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
90	SUBB	SUBB f,Wn	f = f - Wn - (C)	1	1	C, N, OV, Z
		SUBB f,Wn,Wn	Wn = f - Wn - (C)	1	1	C, N, OV, Z
		SUBB #lit16,Wn	Wn = Wn - lit16 - (C)	1	1	C, N, OV, Z
		SUBB Wb,Ws,Wd	Wd = Wb - Ws - (C)	0.5/1	1	C, N, OV, Z
		SUBB Ws,#lit7,Wd	Wd = Ws - lit7 - (literal zero-extended)	1	1	C, N, OV, Z
91	SUBR	SUBR f,Wn	f = Wn - f	1	1	C, N, OV, Z
		SUBR f,Wn,Wn	Wn = Wn - f	1	1	C, N, OV, Z
		SUBR Wb,Ws,Wd	Wd = Ws - Wb	0.5/1	1	C, N, OV, Z
		SUBR Ws,#lit7,Wd	Wd = lit7 - Ws (literal zero-extended)	0.5/1	1	C, N, OV, Z
92	SUBBR	SUBBR f,Wn	f = Wn - f - (C)	1	1	C, N, OV, Z
		SUBBR f,Wn,Wn	Wn = Wn -f - (C)	1	1	C, N, OV, Z
		SUBBR Wb,Ws,Wd	Wd = Ws - Wb - (C)	0.5/1	1	C, N, OV, Z
		SUBBR Ws,#lit7,Wd	Wd = lit7 - Ws - (C) (literal zero-extended)	1	1	C, N, OV, Z
93	SWAP	SWAP Wn	Wn = Word or byte swap Wn	1	1	None
94	SQR	SQR Wxp, A, AWB	Square to Accumulator	1	1	OA, SA, OB, SB
95	SQRAC	SQRAC Wxp, A, AWB	Square and Accumulate	1	1	OA, SA, OB, SB
96	SQRN	SQRN Wxp, A, AWB	Negated Square to Accumulator	1	1	OA, SA, OB, SB
97	SQRSC	SQRSC Wxp, A, AWB	Square and Subtract from Accumulator	1	1	OA, SA, OB, SB
98	TBLRDH		Not supported			N/A
99	TBLRDL		Not supported			N/A
100	TBLWTH		Not supported			N/A
101	TBLWTL		Not supported			N/A
102	TST	TST f	Test f	1	1	N, Z
		TST f,Wnd	Test f and move f to Wnd	1	1	N, Z
103	ULNK	ULNK	Unlink frame pointer	1	1	None
104	XOR	XOR f,Wn	f = f .XOR. Wn	1	1	N, Z
		XOR f,Wn,Wn	Wn = f .XOR. Wn	1	1	N, Z
		XOR lit16,Wn	Wn = Wn .XOR. lit16	1	1	N, Z
		XOR Wb,Ws,Wd	Wd = Wb .XOR. Ws	0.5/1	1	N, Z
		XOR Wb,lit7,Wd	Wd = Wb .XOR. Lit7 (literal zero-extended)	1	1	N, Z
105	ZE	ZE Rso,Wnd	Wd = Zero-extend Ws	0.5/1	1	C, N, Z

Notes:

1. Read and Read-Modify-Write (e.g., bit operations and logical operations) on non-CPU SFRs incur an additional instruction cycle.
2. For dsPIC33AK128MC106 devices, the divide instructions must be preceded with a "`REPEAT #5`" instruction, such that they are executed six consecutive times.

36. Development Support

Move a design from concept to production in record time with Microchip's award-winning development tools. Microchip tools work together to provide state of the art debugging for any project with easy-to-use Graphical User Interfaces (GUIs) in our free MPLAB® X and Atmel Studio Integrated Development Environments (IDEs) and our code generation tools. Providing the ultimate ease-of-use experience, Microchip's line of programmers, debuggers and emulators work seamlessly with our software tools. Microchip development boards help evaluate the best silicon device for an application, while our line of third party tools round out our comprehensive development tool solutions.

Microchip's MPLAB X and Atmel Studio ecosystems provide a variety of embedded design tools to consider that support multiple devices, such as PIC® MCUs, AVR® MCUs, SAM MCUs and dsPIC® DSCs. MPLAB X tools are compatible with Windows®, Linux® and Mac® operating systems while Atmel Studio tools are compatible with Windows.

Go to the following website for more information and details:

www.microchip.com/development-tools/

37. Electrical Characteristics

This section provides an overview of the dsPIC33AK128MC106 family electrical characteristics. Additional information will be provided in future revisions of this document as it becomes available.

Absolute maximum ratings for the dsPIC33AK128MC106 family are listed below. Exposure to these maximum rating conditions for extended periods may affect device reliability. Functional operation of the device at these or any other conditions above the parameters indicated in the operation listings of this specification is not implied.

Table 37-1. Absolute Maximum Ratings(1)

Ambient temperature under bias	-40°C to +125°C
Storage temperature	-65°C to +150°C
Voltage on V_{DD} with respect to V_{SS}	-0.3V to +4.0V
Voltage on AV_{DD} with respect to AV_{SS}	-0.3V to +4.0V
Voltage on any pin that is not 5V tolerant with respect to V_{SS}(3)	-0.3V to (V_{DD} + 0.3V)
Voltage on any 5V tolerant pin with respect to V_{SS} when $V_{DD} \geq 3.0V$(3)	-0.3V to +5.5V
Voltage on any 5V tolerant pin with respect to V_{SS} when $V_{DD} < 3.0V$(3)	-0.3V to +3.6V
Maximum current out of V_{SS} pin	200 mA
Maximum current into V_{DD} pin(2)	200 mA
Maximum current sunk/sourced by any 4x I/O pin	15 mA
Maximum current sunk/sourced by any 8x I/O pin	25 mA
Maximum current sunk by a group of I/Os between two V_{SS} pins(4)	80 mA
Maximum current sourced by a group of I/Os between two V_{DD} pins(4)	80 mA
Maximum current sunk by all ports(2)	200 mA

Notes:

1. Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.
2. Maximum allowable current is a function of device maximum power dissipation (see 37.1. DC Characteristics.
3. See the **Pin Diagrams** section for the 5V tolerant pins.
4. Not applicable to AV_{DD} and AV_{SS} pins.

37.1 DC Characteristics

Table 37-2. Operating MIPs vs. Voltage

Characteristic	V_{DD} Range (in Volts)	Temperature Range (in °C)	Maximum MIPS dsPIC33AK128MC106 Family
—	3.0V to 3.6V	-40°C to +85°C	200
	3.0V to 3.6V	-40°C to +125°C	200
	3.0V to 3.6V	-40°C to +150°C	200

Table 37-3. Thermal Operating Conditions

Rating	Symbol	Min.	Typ.	Max.	Unit
Industrial Temperature Devices Operating Junction Temperature Range Operating Ambient Temperature Range	 T_J T_A	 -40 -40	 — —	 +125 +85	 °C °C
Extended Temperature Devices Operating Junction Temperature Range Operating Ambient Temperature Range	 T_J T_A	 -40 -40	 — —	 +140 +125	 °C °C
Power Dissipation: Internal Chip Power Dissipation: $P_{INT} = V_{DD} \times (I_{DD} - \Sigma I_{OH})$	P_D	$P_{INT} + P_{I/O}$			w
I/O Pin Power Dissipation: $I/O = \Sigma (\{V_{DD} - V_{OH}\} \times I_{OH}) + \Sigma (V_{OL} \times I_{OL})$					
Maximum Allowed Power Dissipation	P_{DMAX}	$(T_J - T_A)/\theta_{JA}$			W

Table 37-4. Thermal Packaging Characteristics(1)

Characteristic	Symbol	Typ.	Max.	Unit
Package Thermal Resistance, 64-Pin TQFP 10x10x1 mm	θ_{JA}	45.7	—	°C/W
Package Thermal Resistance, 64-Pin VQFN 9x9x0.9 mm	θ_{JA}	23.0	—	°C/W
Package Thermal Resistance, 48-Pin TQFP 7x7 mm	θ_{JA}	62.76	—	°C/W
Package Thermal Resistance, 48-Pin VQFN 6x6 mm	θ_{JA}	33.7	—	°C/W
Package Thermal Resistance, 36 VQFN 5x5 mm	θ_{JA}	40.48	—	°C/W
Package Thermal Resistance, 28-Pin VQFN 4x4 mm	θ_{JA}	32.5	—	°C/W
Package Thermal Resistance, 28-Pin SSOP	θ_{JA}	52.48	—	°C/W

Note:

1. Junction to ambient thermal resistance, Theta-$_{JA}$ (θ_{JA}) numbers are achieved by package simulations.

Table 37-5. Operating Voltage Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial** **-40°C ≤ T_A ≤ +125°C for Extended**							
Param No.	**Symbol**	**Characteristic**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
Operating Voltage							
DC10	V_{DD}	Supply Voltage(1)	3.0	—	3.6	V	
DC11	AV_{DD}	Supply Voltage	Greater of: V_{DD} – 0.3 or 3.0	—	Lesser of: V_{DD} + 0.3 or 3.6	V	The difference between AV_{DD} supply and V_{DD} supply must not exceed ±300 mV at all times, including during device power-up.
DC16	V_{POR}	V_{DD} Start Voltage to Ensure Internal Power-on Reset Signal(2)	—	—	V_{SS}	V	
DC17	SV_{DD}	V_{DD} Rise Rate to Ensure Internal Power-on Reset Signal(2)	0.03	—	—	V/ms	0V-3V in 100 ms

Notes:

1. Device is functional at $V_{BORMIN} < V_{DD} < V_{DDMIN}$. Analog specifications cannot be guaranteed.
2. Parameters are characterized but not tested.

.....continued

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	**Symbol**	**Characteristic**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
BO10	V_{BOR}	BOR Event on V_{DD} Transition High-to-Low(2)	2.84	2.94	2.99	V	

Notes:

1. Device is functional at $V_{BORMIN} < V_{DD} < V_{DDMIN}$. Analog specifications cannot be guaranteed.
2. Parameters are characterized but not tested.

Table 37-6. DC Characteristics: Operating Current (I_{DD})

DC Characteristics	Operating Currrent		Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended			
Parameter No.	Typ.(1)	Max.	Units	Conditions		
Operating Current (I_{DD})(2)						
DC21	7	17.5	mA	-40°C	3.3V	20 MIPS (N = 1, N2 = 7, N3 = 4, M = 70, F_{VCO} = 560 MHz, F_{PLLO} = 20 MHz)
	9	21.5	mA	+25°C		
	30	72	mA	+85°C		
	38	82	mA	+125°C		
DC22	10	21.5	mA	-40°C	3.3V	50 MIPS (N = 1, N2 = 4, N3 = 3, M = 75, F_{VCO} = 600 MHz, F_{PLLO} = 50 MHz)
	12	25.5	mA	+25°C		
	33	75	mA	+85°C		
	41	85	mA	+125°C		
DC23	14	27	mA	-40°C	3.3V	100 MIPS (N = 1, N2 = 3, N3 = 2, M = 75, F_{VCO} = 600 MHz, F_{PLLO} = 100 MHz)
	17	31	mA	+25°C		
	37	80	mA	+85°C		
	45	90	mA	+125°C		
DC24	19	34	mA	-40°C	3.3V	150 MIPS (N = 1, N2 = 2, N3 = 2, M = 75, F_{VCO} = 600 MHz, F_{PLLO} = 150 MHz)
	21	38	mA	+25°C		
	42	85	mA	+85°C		
	50	95	mA	+125°C		
DC25	23	41	mA	-40°C	3.3V	200 MIPS (N = 1, N2 = 3, N3 = 1, M = 75, F_{VCO} = 600 MHz, F_{PLLO} = 200 MHz)
	26	45	mA	+25°C		
	47	89	mA	+85°C		
	55	99	mA	+125°C		

Notes:

1. Data in the "Typ." column are for design guidance only and are not tested.
2. I_{DD} is primarily a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption. Base Run current (I_{DD}) is measured as follows:
   - Oscillator is switched to EC+PLL mode in software
   - OSC1 pin is driven with external 8 MHz square wave with levels from 0.3V to V_{DD} – 0.3V
   - OSC2 is configured as an I/O
   - Watchdog Timer is disabled
   - All I/O pins (except OSC1) are configured as outputs and driving low
   - No peripheral modules are operating or being clocked (defined PMDx bits are all '1's)
   - JTAG is disabled
   - `NOP` instructions are executed in `while(1)` loop

Table 37-7. Idle Current (I_{IDLE})[2]

DC Characteristics	Idle		Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended			
Parameter No.	Typ.[1]	Max.	Units	Conditions		
DC41	6	12.5	mA	-40°C	3.3V	20 MIPS (N = 1, N2 = 7, N3 = 4, M = 70, F_{VCO} = 560 MHz, F_{PLLO} = 20 MHz)
	8	16.5	mA	+25°C		
	28	70	mA	+85°C		
	36	80	mA	+125°C		
DC42	8	16.5	mA	-40°C	3.3V	50 MIPS (N = 1, N2 = 4, N3 = 3, M = 75, F_{VCO} = 600 MHz, F_{PLLO} = 50 MHz)
	10	20.5	mA	+25°C		
	30	75	mA	+85°C		
	38	85	mA	+125°C		
DC43	9	22	mA	-40°C	3.3V	100 MIPS (N = 1, N2 = 3, N3 = 2, M = 75, F_{VCO} = 600 MHz, F_{PLLO} = 100 MHz)
	11	26	mA	+25°C		
	32	77	mA	+85°C		
	40	87	mA	+125°C		
DC44	11	29	mA	-40°C	3.3V	150 MIPS (N = 1, N2 = 2, N3 = 2, M = 75, F_{VCO} = 600 MHz, F_{PLLO} = 150 MHz)
	13	33	mA	+25°C		
	34	80	mA	+85°C		
	42	90	mA	+125°C		
DC45	13	36	mA	-40°C	3.3V	200 MIPS (N = 1, N2 = 3, N3 = 1, M = 75, F_{VCO} = 600 MHz, F_{PLLO} = 200 MHz)
	15	40	mA	+25°C		
	36	85	mA	+85°C		
	44	95	mA	+125°C		

Notes:

1. Data in the “Typ.” column are for design guidance only and are not tested.
2. Base Idle current (I_{IDLE}) is measured as follows:
   - Oscillator is switched to EC+PLL mode in software
   - OSC1 pin is driven with external 8 MHz square wave with levels from 0.3V to V_{DD} – 0.3V
   - OSC2 is configured as an I/O
   - FSCM is disabled
   - Watchdog Timer is disabled
   - All I/O pins (except OSC1) are configured as outputs and are driving low
   - No peripheral modules are operating or being clocked (defined PMDx bits are all ‘1’s)
   - JTAG is disabled
   - Flash in standby with NVMPIDL = 1

Table 37-8. DC Characteristics: Power-Down Current (I_{PD})

DC Charateristics	Sleep		Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended		
Parameter No.	Typ.(1)	Max.	Units	Conditions	
Power-Down Current (I_{PD})(2)					
DC60	2	4.2	mA	-40°C	3.3V
	4	10	mA	+25°C	
	24	61	mA	+85°C	
	31	73.1	mA	+125°C	

Notes:

1. Data in the "Typ." column are for design guidance only and are not tested.
2. I_{PD} (Sleep) current is measured as follows:
   - CPU core in sleep, oscillator is configured in EC mode and External Clock is active; OSCI is driven with external square wave from rail-to-rail (EC clock overshoot/undershoot < 250 mV required)
   - CLKO is configured as an I/O input pin
   - All I/O pins are configured as output low
   - $\overline{MCLR}$ = V_{DD}, WDT and FSCM are disabled
   - All peripheral modules are disabled (PMDx bits are all set)
   - JTAG is disabled

Table 37-9. DC Characteristics: Watchdog Timer Delta Current (ΔI_{WDT})(1)

DC Characteristics	Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended			
Parameter No.	Typ.	Units	Conditions	
DC61	5	µA	-40°C	3.3V
	5	µA	+25°C	
	7	µA	+85°C	
	10	µA	+125°C	

Note:

1. ΔI_{WDT} is the additional current consumed when the module is enabled including the LPRC/BFRC clock source current. Parameters are characterized but not tested during manufacturing.

Table 37-10. DC Characteristics: PWM Delta Current[1]

DC Characteristics	Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended				
Parameter No.	Typ.	Max.	Units	Conditions	
DC100	2.3	4	mA	-40°C, 3.3V	PWM output 500 kHz. PWM input 400 Mhz. First instance of PWM generator (PGx).
	2	3.9	mA	+25°C, 3.3V	
	1.8	3,8	mA	+85°C, 3.3V	
	1.8	3.8	mA	+125°C, 3.3V	
DC101	0.8	3.3	mA	-40°C, 3.3V	PWM output 500 kHz. PWM input 400 Mhz. Additional instance of PWM Generator (PGx).
	0.66	3.2	mA	+25°C, 3.3V	
	0.6	3.1	mA	+85°C, 3.3V	
	0.6	3.1	mA	+125°C, 3.3V	

Note:

1. Does not include PLL or CLKGEN currents. PWM in standard resolution mode. Both PWMxH and PWMxL pins driven. Parameters characterized but not tested in manufacturing.

   Multiple instance PWM current = DC100 + n(DC101) where n is the number of PWM Generators (PGx).

Table 37-11. DC Characteristics: CLKGEN Delta Current[1]

DC Characteristics	Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended				
Parameter No.	Typ	Max	Units	Conditions	
DC150	3	5	mA	-40°C, 3.3V	Input frequency 800 MHz. Output frequency 400 MHz.
	3	5	mA	+25°C, 3.3V	
	3	5	mA	+85°C, 3.3V	
	3	5	mA	+125°C, 3.3V	
DC151	2	5	mA	-40°C, 3.3V	Input frequency 400 MHz. Output frequency 400 MHz.
	2	5	mA	+25°C, 3.3V	
	2	5	mA	+85°C, 3.3V	
	2	5	mA	+125°C, 3.3V	
DC152	0.6	2	mA	-40°C, 3.3V	Input frequency 200 MHz. Output frequency 200 MHz.
	0.6	2	mA	+25°C, 3.3V	
	0.6	2	mA	+85°C, 3.3V	
	0.6	2	mA	+125°C, 3.3V	
DC153	0.06	1.6	mA	-40°C, 3.3V	Input frequency 32.768 kHz. Output frequency 32.768 kHz.
	0.06	1.6	mA	+25°C, 3.3V	
	0.06	1.6	mA	+85°C, 3.3V	
	0.06	1.6	mA	+125°C, 3.3V	

Note:

1. Parameters characterized but not tested in manufacturing.

Table 37-12. DC Characteristics: PLL Delta Current[1]

DC Characteristics	Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended				
Parameter No.	Typ.	Max.	Units	Conditions[1]	
DC110	6.4	7.5	mA	-40°C, 3.3V	Input frequency 8 MHz. Output frequency 800 MHz. M = 100, N = 1, N2 = 1, F_{VCO} = 800 MHz
	6.5	7.6	mA	+25°C, 3.3V	
	6.6	7.7	mA	+85°C, 3.3V	
	6.7	7.8	mA	+125°C, 3.3V	
DC111	4.4	4.8	mA	-40°C, 3.3V	Input frequency 8 MHz. Output frequency 320 MHz. M = 40, N = 1, N2 = 1, F_{VCO} = 640 MHz
	4.5	4.9	mA	+25°C, 3.3V	
	4.6	5	mA	+85°C, 3.3V	
	4.7	5.1	mA	+125°C, 3.3V	
DC112	3.9	4.2	mA	-40°C, 3.3V	Input frequency 8 MHz. Output frequency 200 MHz. M = 75, N = 3, N2 = 1, F_{VCO} = 600 MHz
	4	4.3	mA	+25°C, 3.3V	
	4.1	4.4	mA	+85°C, 3.3V	
	4.1	4.5	mA	+125°C, 3.3V	

Note:

1. Values do not include CLKGEN current. Parameters characterized but not tested in manufacturing.

Table 37-13. DC Characteristics: ADC Δ Current[1]

DC Charateristics	Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended				
Parameter No.	Typ.	Max.	Units	Conditions	
DC120	3.4	3.6	mA	-40°C, 3.3V	Input frequency 320 MHz. ADC clock 80 MHz. T_{AD} = 12.5 nS. Continuous conversion of single channel of single ADC core. First instance of ADC.
	3.6	3.8	mA	+25°C, 3.3V	
	3.8	4	mA	+85°C, 3.3V	
	3.9	4.2	mA	+125°C, 3.3V	
DC121	3.7	4.1	mA	-40°C, 3.3V	Input frequency 320 MHz. ADC clock 80 MHz. T_{AD} = 12.5 nS. Continuous conversion of single channel of single ADC core. Additional instances of ADC.
	3.8	4.2	mA	+25°C, 3.3V	
	3.9	4.3	mA	+85°C, 3.3V	
	4	4.4	mA	+125°C, 3.3V	

Note:

1. Parameters are characterized but not tested in manufacturing. Does not include PLL or CLKGEN currents. Multiple instance ADC current = DC120 + n(DC121) where n is the number of ADC instances.

Table 37-14. DC Characteristics: Comparator + DAC Delta Current[1]

DC Characteristics	Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended				
Parameter No.	Typ.	Max.	Units	Conditions	
DC130	1.6	2	mA	-40°C, 3.3V	Input frequency 400 Mhz First instance of DAC + Comparator.
	1.7	2.2	mA	+25°C, 3.3V	
	1.9	2.6	mA	+85°C, 3.3V	
	2.1	2.8	mA	+125°C, 3.3V	
DC131	0.8	1.1	mA	-40°C, 3.3V	Input frequency 400 Mhz Additional instance of DAC + Comparator.
	0.8	1.1	mA	+25°C, 3.3V	
	0.9	1.2	mA	+85°C, 3.3V	
	0.9	1.2	mA	+125°C, 3.3V	

Note:

1. Parameters are characterized but not tested in manufacturing. Does not include PLL or CLKGEN currents. DAC output pin driven to $V_{DD}/2$. CMP input at V_{SS}. Multiple instance DAC+CMP current = DC130 + n(DC131) where n is the number of DAC+CMP instances.

Table 37-15. Op Amp Delta Current[1]

Parameter No.	Typ.	Max.	Units	Conditions	
DC140	1.7	2.2	mA	-40°C, 3.3V	High-power mode first instance of Op Amp
	2.2	2.8	mA	+25°C, 3.3V	
	2.6	3.5	mA	+85°C, 3.3V	
	3.1	3.8	mA	+125°C, 3.3V	
DC141	1.7	2.4	mA	-40°C, 3.3V	High-power mode additional instance of Op Amp
	2.3	3.1	mA	+25°C, 3.3V	
	2.7	3.8	mA	+85°C, 3.3V	
	3.2	4.1	mA	+125°C, 3.3V	
DC142	0.13	0.2	mA	-40°C, 3.3V	Low-power mode first instance of Op Amp
	0.15	0.3	mA	+25°C, 3.3V	
	0.18	0.3	mA	+85°C, 3.3V	
	0.2	0.3	mA	+125°C, 3.3V	
DC143	0.14	0.2	mA	-40°C, 3.3V	Low-power mode additional instance of OpAmp
	0.18	0.3	mA	+25°C, 3.3V	
	0.19	0.3	mA	+85°C, 3.3V	
	0.22	0.3	mA	+125°C, 3.3V	

Note:

1. Parameters are characterized but not tested in manufacturing. Does not include PLL or CLKGEN currents. Output pin enabled. Inverting input at $V_{DD}/2$. Multiple instance Op Amp current = DC140 + n(DC141) where n is the number of Op Amp instances.

Table 37-16. I/O Pin Input Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
DI10	V_{IL}	Input Low Voltage					
		Any I/O Pin and $\overline{MCLR}$	V_{SS}	—	0.2 V_{DD}	V	
		I/O Pins with SDAx, SCLx	V_{SS}	—	0.3 V_{DD}	V	SMBus disabled
		I/O Pins with SDAx, SCLx(4)	V_{SS}	—	0.8	V	SMBus enabled
		I/O Pins with SDAx, SCLx(4)	V_{SS}	—	0.8	V	SMBus 3.0
DI20	V_{IH}	Input High Voltage					
		I/O Pins Not 5V Tolerant(2)	0.8 V_{DD}	—	V_{DD}	V	
		5V Tolerant I/O Pins and $\overline{MCLR}$(2)	0.8 V_{DD}	—	5.5	V	
		5V Tolerant I/O Pins with SDAx, SCLx(2)	0.8 V_{DD}	—	5.5	V	SMBus disabled
		5V Tolerant I/O Pins with SDAx, SCLx(2,4)	2.1	—	5.5	V	SMBus enabled
		5V Tolerant I/O Pins with SDAx, SCLx(2,4)	1.35	—	5.5	V	SMBus 3.0
		I/O Pins with SDAx, SCLx Not 5V Tolerant(2)	0.8 V_{DD}	—	V_{DD}	V	SMBus disabled
		I/O Pins with SDAx, SCLx Not 5V Tolerant(2,4)	2.1	—	V_{DD}	V	SMBus enabled
		I/O Pins with SDAx, SCLx Not 5V Tolerant(2,4)	1.35	—	V_{DD}	V	SMBus 3.0
DI30	I_{CNPU}	Input Change Notification Pull-up Current(1,3)	-175	-360	-545	µA	V_{DD} = 3.3V, V_{PIN} = V_{SS}
DI31	I_{CNPD}	Input Change Notification Pull-Down Current(3)	250	320	380	µA	V_{DD} = 3.3V, V_{PIN} = V_{DD}

Notes:

1. Negative current is defined as current sourced by the pin.
2. See the **Pin Diagrams** section for the 5V tolerant I/O pins.
3. All parameters are characterized but not tested during manufacturing.
4. Parameters are not characterized or tested in manufacturing.

Table 37-17. I/O Pin Input Leakage Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended						
Param No.	Symbol	Characteristic	Min.	Max.	Units	Conditions
DI50	I_{IL}	Input Leakage Current(1)				
		I/O Pins 5V Tolerant(2)	-1.5	7.5	µA	V_{PIN} = V_{SS} or V_{DD}
		I/O Pins Not 5V Tolerant(2)	-1	1	µA	
		$\overline{MCLR}$	-1	7.5	µA	
		OSCI	-1	1	µA	MS and HS modes

Notes:

1. Negative current is defined as current sourced by the pin.
2. See the **Pin Diagrams** section for the 5V tolerant I/O pins.

Table 37-18. I/O Pin Input Injection Current Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended						
Param No.	Symbol	Characteristic	Min.	Max.	Units	Conditions
DI60a	I_{ICL}	Input Low Injection Current	0	-5[1,4]	mA	All pins
DI60b	I_{ICH}	Input High Injection Current	0	+5[2,3,4]	mA	All pins, except all 5V tolerant pins
DI60c	ΣI_{ICT}	Total Input Injection Current (sum of all I/O and control pins)[5]	-20	20	mA	Absolute instantaneous sum of all ± input injection currents from all I/O pins (| I_{ICL} | + | I_{ICH} |) ≤ ΣI_{ICT}

Notes:

1. V_{IL} Source < (V_{SS} – 0.3).
2. V_{IH} Source > (V_{DD} + 0.3) for non-5V tolerant pins only.
3. 5V tolerant pins do not have an internal high-side diode to V_{DD} and therefore cannot tolerate any "positive" input injection current.
4. Injection currents can affect the ADC results.
5. Any number and/or combination of I/O pins, not excluded under I_{ICL} or I_{ICH} conditions, are permitted in the sum.

Table 37-19. I/O Pin Output Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
DO10	V_{OL}	Output Low Voltage 4x Sink Driver Pins	—	—	0.4	V	V_{DD} = 3.3V, I_{OL} < 5 mA
		Output Low Voltage 8x Sink Driver Pins[1]	—	—	0.4	V	V_{DD} = 3.3V, I_{OL} < 15 mA
DO20	V_{OH}	Output High Voltage 4x Source Driver Pins	2.7	—	—	V	V_{DD}= 3.3V, I_{OH} > -5 mA
		Output High Voltage 8x Source Driver Pins[1]	2.7	—	—	V	V_{DD} = 3.3V, I_{OH} > -10 mA

Note:

1. See Pinout Diagrams to identify pin current capability of either 4x or 8x.

Table 37-20. Program Memory Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended						
Param No.	Symbol	Characteristic	Min.	Max.	Units	Conditions
		Program Flash Memory				
D130	E_P	Cell Endurance	10,000	—	E/W	-40°C to +125°C
D131	V_{PR}	V_{DD} for Read	3.0	3.6	V	
D132b	V_{PEW}	V_{DD} for Self-Timed Write	3.0	3.6	V	
D134	T_{RETD}	Characteristic Retention	20	—	Year	Provided no other specifications are violated, -40°C to +125°C

Note: Programming time is non-deterministic and not specified. Refer to 6.3. Operation for more information.

Table 37-21. ESD and Latch-Up Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended					
Param No.	**Characteristic**	**Tolerance**	**Units**	**Condition**	**Specification**
CDM1	Captive Discharge Model	750	V	Corner pins, all packages except SSOP	AEC-Q100-11-C1
		700		Corner pins, 28 SSOP	
		500		Non-corner pins	
LU1	Latch-Up	105	mA		AEC-Q100-004
		1.5x Max V_{DD}	V		
HBM1	Human Body Model	2	kV		AEC-Q100-002E

37.2 AC Characteristics and Timing Parameters

Figure 37-1. Load Condition for Device Timing Specifications

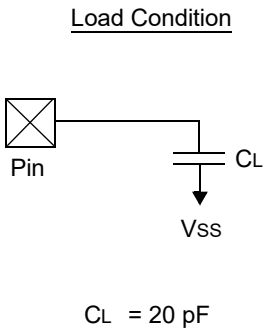


Table 37-22. Capacitive Loading Requirements on Output Pins

Param No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
DO50	C_{OSCO}	OSCO Pin	—	—	20	pF	In XT and HS modes, when External Clock is used to drive OSCI
DO56	C_{IO}	All I/O Pins and OSCO	—	—	20	pF	EC mode
DO58	C_B	SCLx, SDAx	—	—	400	pF	In I^2C mode

Figure 37-2. External Clock Timing

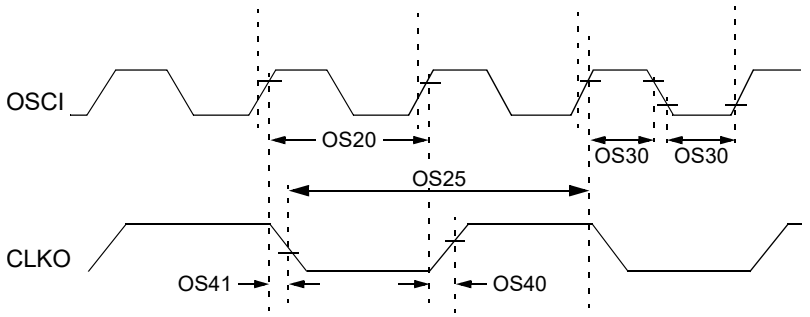


Table 37-23. External Clock Timing Requirements

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Sym	Characteristic	Min.	Typ.(1)	Max.	Units	Conditions
OS10	F_{IN}	External CLKI Frequency (External Clocks allowed only in EC mode)	DC	—	64	MHz	EC
		Oscillator Crystal Frequency	3.5	—	10	MHz	XT
			10	—	32	MHz	HS
OS11	F_{REFI}	REFI External Clock Input	—	—	50	MHz	
OS20	T_{OSC}	T_{OSC} = 1/F_{OSC}	15.6	—	—	ns	
OS25	T_{CY}	Instruction Cycle Time	5	—	DC	ns	
OS30	$T_{OS}L$, $T_{OS}H$	External Clock in (OSCI) High or Low Time	0.45 x T_{OSC}	—	0.55 x T_{OSC}	ns	EC
OS40	$T_{CK}R$	CLKO Rise Time	—	—	—	ns	Refer to DO31
OS41	$T_{CK}F$	CLKO Fall Time	—	—	—	ns	Refer to DO32
OS42	G_M	External Oscillator Transconductance(3)	2.8	—	3.7	mA/V	PXTALCFG[1:0] = 00, PXTALCFG[2] = 0
			4.5	—	6.2	mA/V	PXTALCFG[1:0] = 00, PXTALCFG[2] = 1
			4.5	—	6.2	mA/V	PXTALCFG[1:0] = 01, PXTALCFG[2] = 0
			7	—	10.5	mA/V	PXTALCFG[1:0] = 01, PXTALCFG[2] = 1
			5.8	—	8.5	mA/V	PXTALCFG[1:0] = 10, PXTALCFG[2] = 0
			8.9	—	14.1	mA/V	PXTALCFG[1:0] = 10, PXTALCFG[2] = 1
			7	—	10.5	mA/V	PXTALCFG[1:0] = 11, PXTALCFG[2] = 0
			10.7	—	17.3	mA/V	PXTALCFG[1:0] = 11, PXTALCFG[2] = 1

Notes:

1. Data in "Typ." column are at 3.3V, +25°C unless otherwise stated.
2. Measurements are taken in EC mode. The CLKO signal is measured on the OSCO pin.
3. Parameters characterized but not tested in manufacturing.

Table 37-24. PLLn Timing Specifications[1]

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.[1]	Max.	Units	Conditions
OS50	F_{PLLI}	PLL Voltage Controlled Oscillator (VCO) Input Frequency Range	5	—	64	MHz	
OS51	F_{VCO}	On-Chip VCO System Frequency	500	—	1600	MHz	
OS52	M	Feedback Divider Value	16	—	400	N/A	
OS53	T_{LOCK}	PLL Start-up Time (Lock Time) [2]	—	125	188	µs	F_{PLLI} = 8 MHz
OS54	DCLK	CLKO Stability (Jitter)	—	—	1	ps	F_{VCO} = 1600 MHz, F_{OUT} = 400 MHz
OS55	F_{OUT}	PLL PostDiv Output	—	—	800	MHz	
OS56	F_{OUTDIV}	PLL VCO FractDiv Output	—	—	800	MHz	
OS57	F_{INDIV}	PLL FracDiv Input,	500	—	800	MHz	

Notes:

1. Parameters for design guidance only and not tested in manufacturing.
2. Parameters characterized but not tested in manufacturing.

Table 37-25. Peripheral Input Clock Timing Specifications[1]

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended					
Param No.	Characteristic	Min.	Max.	Units	Conditions
F_{INMAX}	CLKGEN	0	800	MHz	CLKGEN input
	PWM	0	400	MHz	
	ADC	32	320	MHz	
	DAC	400	500	MHz	
	DMT	—	200	MHz	
	MCCP	0	200	MHz	System clock
	SPI	0	320	MHz	
	UART	0	320	MHz	
	PTG	0	400	MHz	From PWM clock
		0	320	MHz	From ADC clock
	CLC	0	200	MHz	System clock
	I2C	0	100	MHz	Standard speed clock
	SENT	0	100	MHz	Standard speed clock
	BISS	0	320	MHz	
	CRC	0	200	MHz	Fast speed clock
	QEI	0	100	MHz	Standard speed clock
	QEI	0	100	MHz	Standard speed clock
	PPS	0	50	MHz	Slow speed clock
	GPIO	0	200	MHz	Fast speed clock for PORTx and LATx registers
		0	50	MHz	Slow Speed clock, all other GPIO registers
	JTAG	0	20	MHz	TCK max input clock

Note:
1. Parameters are for design guidance only and are not tested.

Table 37-26. Internal FRC Accuracy

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended					
Param No.	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
Internal FRC Accuracy @ FRC Frequency = 8 MHz(1)					
F20	FRC	-1.5	1.5	%	-40°C ≤ T_A ≤ 85°C
		-2	2	%	+85°C ≤ T_A ≤ +125°C
F21	BFRC	-2	2	%	-40°C ≤ T_A ≤ +125°C
F22	BFRC/244-LPRC	-3	3	%	-40°C ≤ T_A ≤ +125°C

Note:

1. Frequency is calibrated at +25°C and 3.3V. TUNx bits can be used to compensate for temperature drift.

Figure 37-3. I/O Timing Characteristics

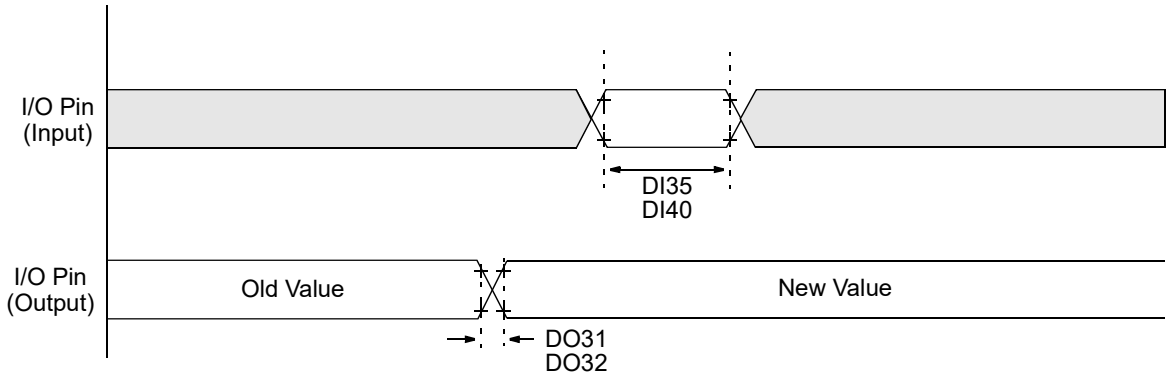


Table 37-27. I/O Timing Requirements

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	**Symbol**	**Characteristic**	**Min.**	**Typ.(1)**	**Max.**	**Units**	**Conditions**
DO31	$T_{IO}R$	Port Output Rise Time(2)	—	3.2	4	ns	4x output pins
			—	2.3	3.1	ns	8x output pins
DO32	$T_{IO}F$	Port Output Fall Time(2)	—	2.5	3	ns	4x output pins
			—	1.7	2.4	ns	8x output pins
DO33	F_{REFO}	REFO Output Frequency	—	—	30	MHz	
DI35	T_{INP}	INTx Pin High or Low Time (input)	6	—	—	ns	
DI40	T_{RBP}	CNx High or Low Time (input)	2	—	—	T_{CY}	

Notes:

1. Data in "Typ." column are at 3.3V, +25°C unless otherwise stated.
2. Parameters characterized but not tested in manufacturing.

Figure 37-4. BOR and Master Clear Reset Timing Characteristics

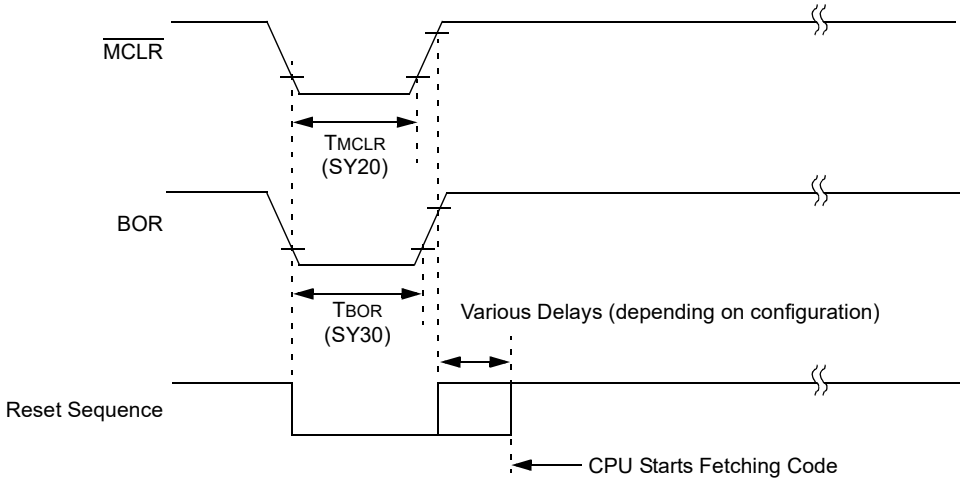


Table 37-28. Reset and Watchdog Timer Timing Requirements

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	**Symbol**	**Characteristic(1)**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
SY00	T_{PU}	Power-up Period	—	80	—	µs	BOR trip point to first instruction on FRC
SY13	T_{IOZ}	I/O High-Impedance from $\overline{MCLR}$ Low or Watchdog Timer Reset	—	1	—	µs	
SY20	T_{MCLR}	$\overline{MCLR}$ Pulse Width (low)	8	—	—	µs	
SY35	T_{FSCM}	Fail-Safe Clock Monitor Delay	—	—	2	µs	Clock fail to BFRC ready
SY40	F_{MIN}	Fail-Safe Clock Monitor Minimum Frequency	4	—	—	MHz	

Note:
1. Parameters characterized but not tested in manufacturing.

Figure 37-5. High-Speed PWMx Module Fault Timing Characteristics

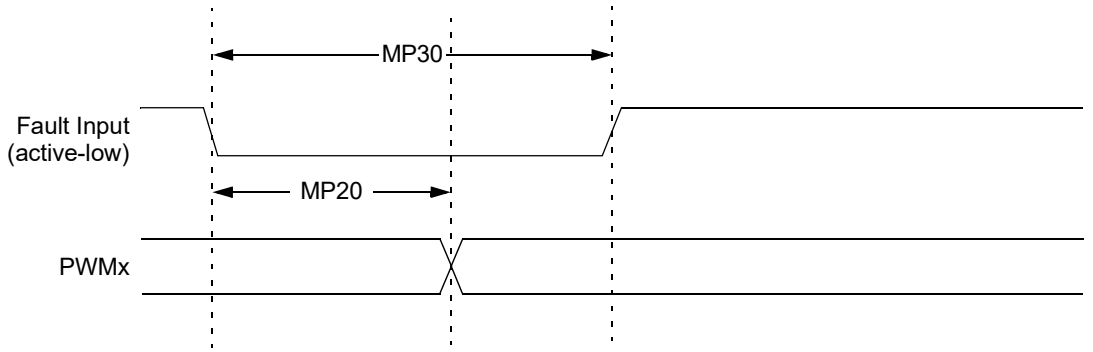


Figure 37-6. High-Speed PWMx Module Timing Characteristics

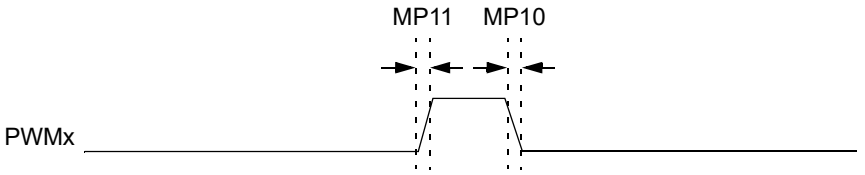


Table 37-29. High-Speed PWMx Module Timing Requirements

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	**Symbol**	**Characteristic**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
MP10	T_{FPWM}	PWMx Output Fall Time	—	—	—	ns	See Parameter DO32
MP11	T_{RPWM}	PWMx Output Rise Time	—	—	—	ns	See Parameter DO31
MP20	T_{FD}	Fault Input ↓ to PWMx I/O Change(1)	—	15	—	ns	PCI Inputs 19 through 22, level triggered
MP30	T_{FH}	Fault Input Pulse Width(1)	4	—	—	ns	

Note:
1. These parameters are characterized but not tested in manufacturing.

Figure 37-7. QEI Module Index Pulse Timing Characteristics

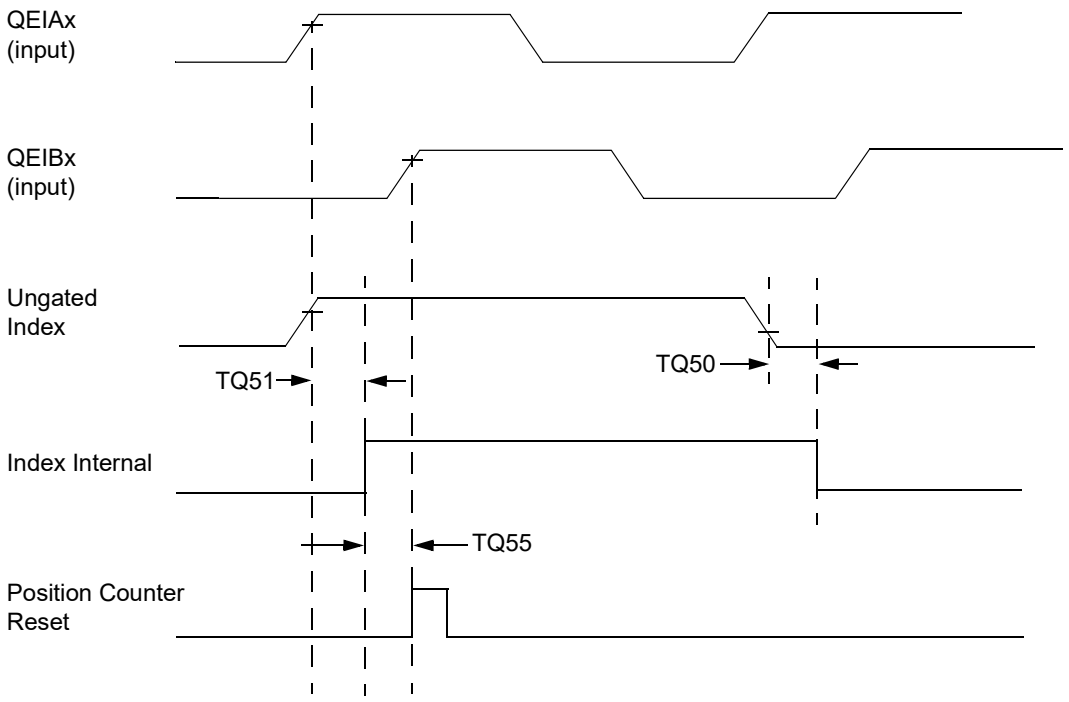


Table 37-30. QEI Index Pulse Timing Requirements

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended						
Param No.	Symbol	Characteristic[1]	Min.	Max	Units	Conditions
TQ50	TqIL	Filter Time to Recognize Low with Digital Filter[2]	3 * N * T_{CY}	—	ns	N = 1, 2, 4, 16, 32, 64, 128 and 256
TQ51	TqiH	Filter Time to Recognize High with Digital Filter[2]	3 * N * T_{CY}	—	ns	N = 1, 2, 4, 16, 32, 64, 128 and 256
TQ55	Tqidxr	Index Pulse Recognized to Position Counter Reset (ungated index)	3 T_{CY}	—	ns	

Notes:

1. These parameters are characterized but not tested in manufacturing.
2. Alignment of index pulses to QEIA and QEIB is shown for position counter Reset timing only. Shown for forward direction only (QEIA leads QEIB). Same timing applies for reverse direction (QEIA lags QEIB), but index pulse recognition occurs on the falling edge.

Table 37-31. SPIx Maximum Data/Clock Rate Summary

SPI Host Transmit Only (Half-Duplex)	SPI Host Transmit/ Receive (Full-Duplex)	SPI Client Transmit/ Receive (Full-Duplex)	CKE	Maximum Data Rate (MHz)	Condition
Figure 37-8 Figure 37-9	—	—	0	25	Using PPS
				50	Dedicated Pin
Figure 37-9 Table 37-32	—	—	1	25	Using PPS
				50	Dedicated Pin
—	Figure 37-10 Table 37-33	—	0	25	Using PPS
				50	Dedicated Pin
—	Figure 37-11 Table 37-34	—	1	25	Using PPS
				50	Dedicated Pin
—	—	Figure 37-12 Table 37-35	0	25	Using PPS
				50	Dedicated Pin
—	—	Figure 37-13 Table 37-36	1	25	Using PPS
				50	Dedicated Pin

Figure 37-8. SPIx Host Mode (Half-Duplex, Transmit Only, CKE = 0) Timing Characteristics

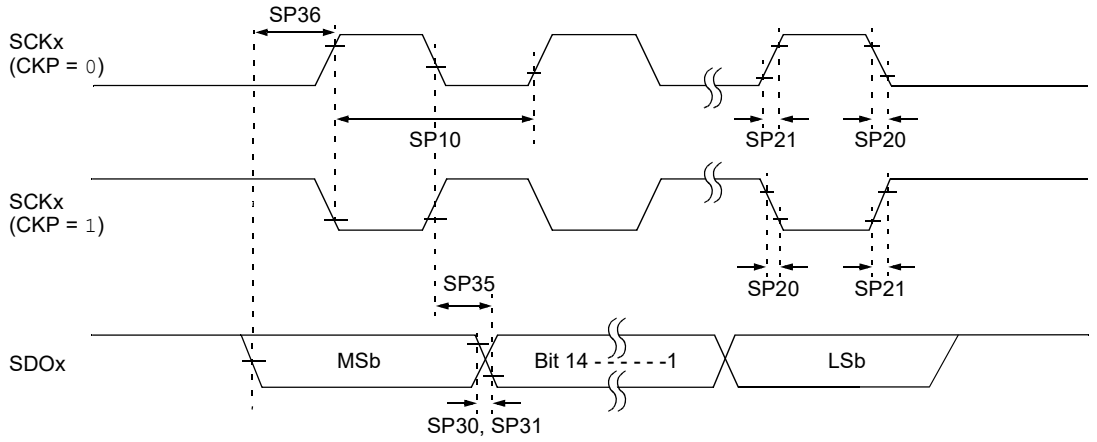


Figure 37-9. SPIx Host Mode (Half-Duplex, Transmit Only, CKE = 1) Timing Characteristics

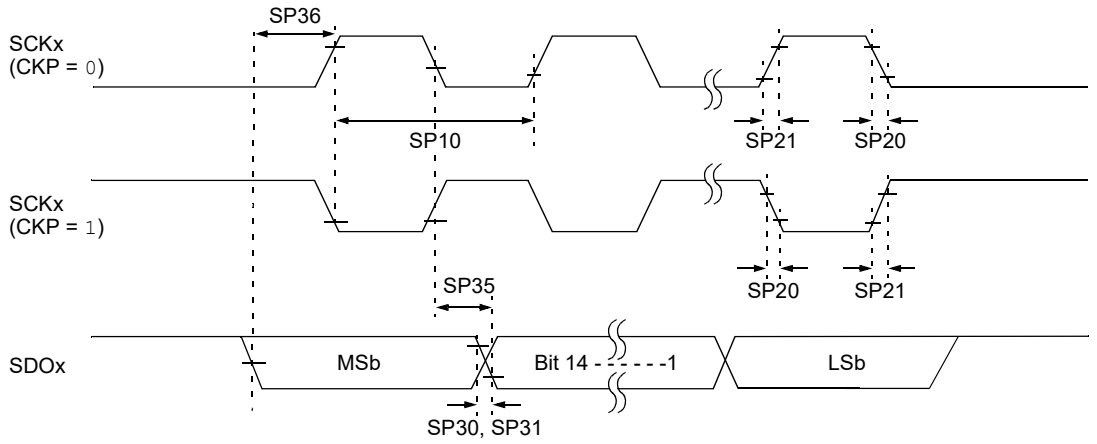


Table 37-32. SPIx Host Mode (Half-Duplex, Transmit Only) Timing Requirements(1)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.(2)	Max.	Units	Conditions
SP10	$F_{SC}P$	Maximum SCKx Frequency	—	—	25	MHz	Using PPS pins
			—	—	50	MHz	SPI2 dedicated pins
SP20	$T_{SC}F$	SCKx Output Fall Time(3)	—	—	5	ns	Using PPS pins
			—	—	2.5		SPI2 dedicated pins
SP21	$T_{SC}R$	SCKx Output Rise Time(3)	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP30	$T_{DO}F$(3)	SDOx Data Output Fall Time	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP31	$T_{DO}R$(3)	SDOx Data Output Rise Time	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP35	$T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$	SDOx Data Output Valid After SCKx Edge	—	—	7	ns	Using PPS pins
			2.7	—	4.4	ns	SPI dedicated pins
SP36	$T_{DI}V2_{SC}H$, $T_{DI}V2_{SC}L$	SDOx Data Output Setup to First SCKx Edge	10	—	—	ns	Using PPS pins
			1	—	2	ns	SPI2 dedicated pins

Notes:

1. These parameters are characterized but not tested in manufacturing.
2. Data in "Typ." column are at 3.3V, +25°C unless otherwise stated.
3. Assumes 30 pF load on all SPIx pins.

Figure 37-10. SPIx Host Mode (Full-Duplex, CKE = 1, CKP = x, SMP = 1) Timing Characteristics

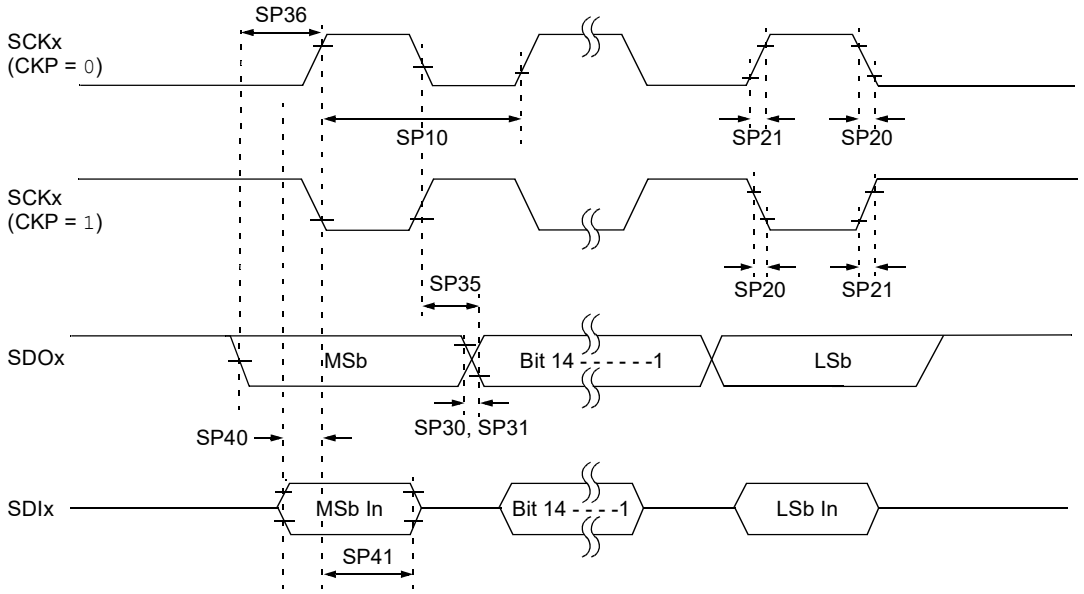


Table 37-33. SPIx Host Mode (Full-Duplex, CKE = 1, CKP = X, SMP = 1) Timing Requirements(1)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.(2)	Max.	Units	Conditions
SP10	$F_{SC}P$	Maximum SCKx Frequency	—	—	25	MHz	Using PPS pins
			—	—	50	MHz	SPI2 dedicated pins
SP20	$T_{SC}F$	SCKx Output Fall Time(3)	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP21	$T_{SC}R$	SCKx Output Rise Time(3)	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP30	$T_{DO}F$	SDOx Data Output Fall Time(3)	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP31	$T_{DO}R$	SDOx Data Output Rise Time(3)	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP35	$T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$	SDOx Data Output Valid After SCKx Edge	—	—	7	ns	Using PPS pins
			2.7	—	4.4	ns	SPI2 dedicated pins

Notes:

1. These parameters are characterized but not tested in manufacturing.
2. Data in "Typ." column are at 3.3V, +25°C unless otherwise stated.
3. Assumes 30 pF load on all SPIx pins.

.....continued

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.(2)	Max.	Units	Conditions
SP36	$T_{DO}V2_{SC}$, $T_{DO}V2_{SC}L$	SDOx Data Output Setup to First SCKx Edge	10	—	—	ns	Using PPS pins
			1	—	2	ns	SPI2 dedicated pins
SP40	$T_{DI}V2_{SC}H$, $T_{DI}V2_{SC}L$	Setup Time of SDIx Data Input to SCKx Edge	—	—	17	ns	Using PPS pins
			0.1	—	0.6	ns	SPI2 dedicated pins
SP41	$T_{SC}H2_{DI}L$, $T_{SC}L2_{DI}L$	Hold Time of SDIx Data Input to SCKx Edge	2	—	—	ns	Using PPS pins
			2.5	—	3.3	ns	SPI2 dedicated pins

Notes:

1. These parameters are characterized but not tested in manufacturing.
2. Data in "Typ." column are at 3.3V, +25°C unless otherwise stated.
3. Assumes 30 pF load on all SPIx pins.

Figure 37-11. SPIx Host Mode (Full-Duplex, CKE = 0, CKP = x, SMP = 1) Timing Characteristics

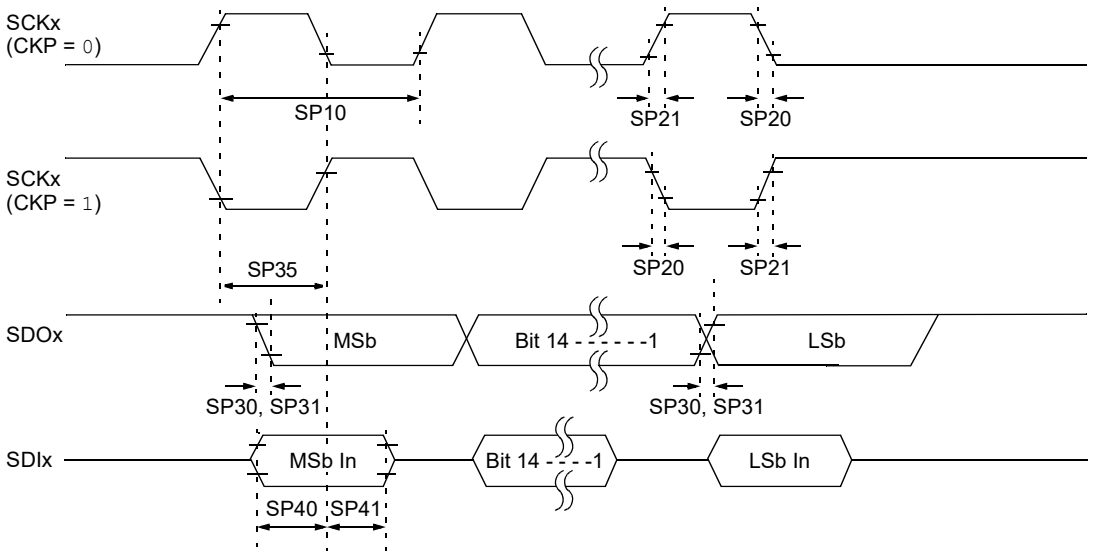


Table 37-34. SPIx Host Mode (Full-Duplex, CKE = 0, CKP = x, SMP = 1) Timing Requirements[1]

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.[2]	Max.	Units	Conditions
SP10	$F_{SC}P$	Maximum SCKx Frequency	—	—	25	MHz	Using PPS pins
			—	—	50	MHz	SPI2 dedicated pins
SP20	$T_{SC}F$	SCKx Output Fall Time[3]	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP21	$T_{SC}R$	SCKx Output Rise Time[3]	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP30	$T_{DO}F$	SDOx Data Output Fall Time[3]	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP31	$T_{DO}R$	SDOx Data Output Rise Time[3]	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP35	$T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$	SDOx Data Output Valid After SCKx Edge	—	—	7	ns	Using PPS pins
			2.7	—	4.4	ns	SPI2 dedicated pins
SP40	$TdiV2_{SC}H$, $TdiV2_{SC}L$	Setup Time of SDIx Data Input to SCKx Edge	—	—	17	ns	Using PPS pins
			0.1	—	0.6	ns	SPI2 dedicated pins
SP41	$T_{SC}H2_{DI}L$, $T_{SC}L2_{DI}L$	Hold Time of SDIx Data Input to SCKx Edge	2	—	—	ns	Using PPS pins
			2.5	—	3.3	ns	SPI2 dedicated pins

Notes:

1. These parameters are characterized but not tested in manufacturing.
2. Data in "Typ." column are at 3.3V, +25°C unless otherwise stated.
3. Assumes 30 pF load on all SPIx pins.

Figure 37-12. SPIx Client Mode (Full-Duplex, CKE = 0, CKP = x, SMP = 0) Timing Characteristics

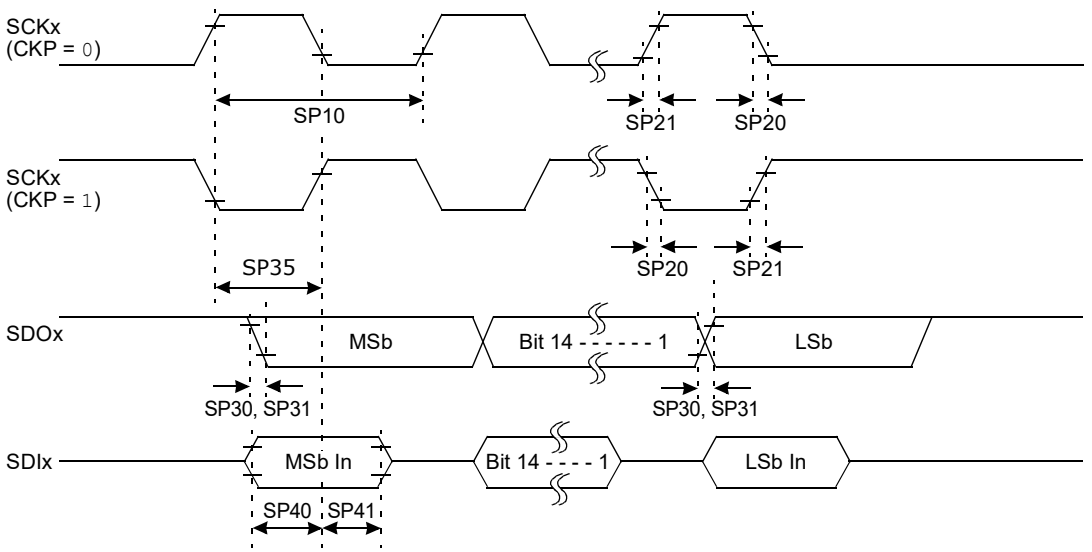


Table 37-35. SPIx Client Mode (Full-Duplex, CKE = 0, CKP = x, SMP = 0) Timing Requirements[1]

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Char.	Min.	Typ.[2]	Max.	Units	Conditions
SP10	$F_{SC}P$	Maximum SCKx Input Frequency	—	—	25	MHz	Using PPS pins
			—	—	50	MHz	SPI2 dedicated pins
SP72	$T_{SC}F$	SCKx Input Fall Time[3]	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP73	$T_{SC}R$	SCKx Input Rise Time[3]	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP30	$T_{DO}F$	SDOx Data Output Fall Time[3]	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP31	$T_{DO}R$	SDOx Data Output Rise Time[3]	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP35	$T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$	SDOx Data Output Valid After SCKx Edge	—	—	7	ns	Using PPS pins
			2.7	—	4.4	ns	SPI2 dedicated pins
SP40	$T_{DI}V2_{SC}H$, $T_{DI}V2_{SC}L$	Setup Time of SDIx Data Input to SCKx Edge	—	—	17	ns	Using PPS pins
			1	—	1	ns	SPI2 dedicated pins
SP41	$T_{SC}H2_{DI}L$, $T_{SC}L2_{DI}L$	Hold Time of SDIx Data Input to SCKx Edge	2	—	—	ns	Using PPS pins
			2.5	—	3.3	ns	SPI2 dedicated pins
SP50	$T_{SS}L2_{SC}H$, $T_{SS}L2_{SC}L$	$\overline{SSx}$ ↓ to SCKx ↑ or SCKx ↓ Input	90	—	—	ns	
SP51	$T_{SS}H2_{DO}Z$	$\overline{SSx}$ ↑ to SDOx Output High-Impedance	5	—	20	ns	
SP52	$T_{SC}H2_{SS}H$, $T_{SC}L2_{SS}H$	$\overline{SSx}$ ↑ After SCKx Edge[3]	1.5 T_{CY} + 40	—	—	ns	

Notes:

1. These parameters are characterized but not tested in manufacturing.
2. Data in "Typ." column are at 3.3V, +25°C unless otherwise stated.
3. Assumes 30 pF load on all SPIx pins.

Figure 37-13. SPIx Client Mode (Full-Duplex, CKE = 1, CKP = x, SMP = 0) Timing Characteristics

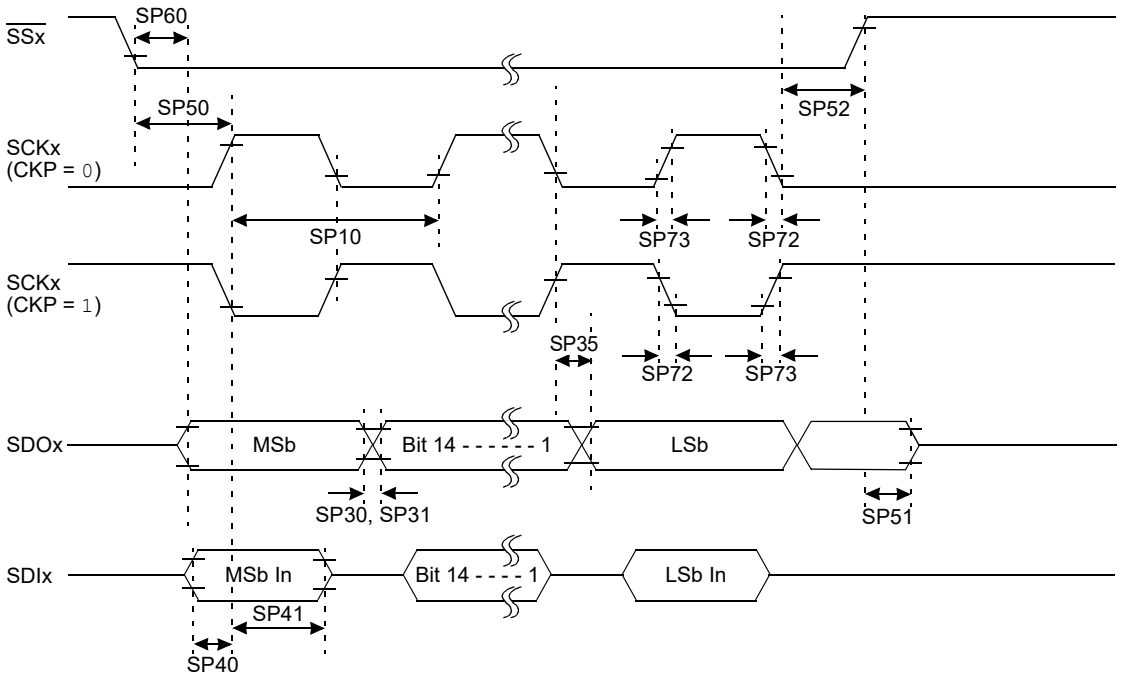


Table 37-36. SPIx Client Mode (Full-Duplex, CKE = 1, CKP = x, SMP = 0) Timing Requirements(1)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.(2)	Max.	Units	Conditions
SP10	$F_{SC}P$	Maximum SCKx Input Frequency	—	—	25	MHz	Using PPS pins
			—	—	50	MHz	SPI2 dedicated pins
SP72	$T_{SC}F$	SCKx Input Fall Time(3)	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP73	$T_{SC}R$	SCKx Input Rise Time(3)	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP30	$T_{DO}F$	SDOx Data Output Fall Time(3)	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins

Notes:

1. These parameters are characterized but not tested in manufacturing.
2. Data in "Typ." column are at 3.3V, +25°C unless otherwise stated.
3. Assumes 30 pF load on all SPIx pins.

.....continued

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.(2)	Max.	Units	Conditions
SP31	$T_{DO}R$	SDOx Data Output Rise Time(3)	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP35	$T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$	SDOx Data Output Valid After SCKx Edge	—	—	7	ns	Using PPS pins
			2.7	—	4.4	ns	SPI dedicated pins
SP40	$T_{DI}V2_{SC}H$, $T_{DI}V2_{SC}L$	Setup Time of SDIx Data Input to SCKx Edge	—	—	17	ns	Using PPS pins
			0.1	—	0.6	ns	SPI2 dedicated pins
SP41	$T_{SC}H2diL$, $T_{SC}L2_{DI}L$	Hold Time of SDIx Data Input to SCKx Edge	2	—	—	ns	Using PPS pins
			2.5	—	3.3	ns	SPI2 dedicated pins
SP50	$T_{SS}L2_{SC}H$, TssL2scL	$\overline{SSx}$ ↓ to SCKx ↑ or SCKx ↓ Input	90	—	—	ns	
SP51	$T_{SS}H2doZ$	$\overline{SSx}$ ↑ to SDOx Output High-Impedance(3)	5	—	20	ns	
SP52	$T_{SC}H2_{SS}H$, $T_{SC}L2_{SS}H$	$\overline{SSx}$ ↑ After SCKx Edge(3)	1.5 T_{CY} + 40	—	—	ns	
SP60	$T_{SS}L2_{DO}V$	SDOx Data Output Valid After $\overline{SSx}$ Edge	—	—	50	ns	

Notes:

1. These parameters are characterized but not tested in manufacturing.
2. Data in "Typ." column are at 3.3V, +25°C unless otherwise stated.
3. Assumes 30 pF load on all SPIx pins.

Figure 37-14. I2Cx Bus Start/Stop Bits Timing Characteristics (Host Mode)

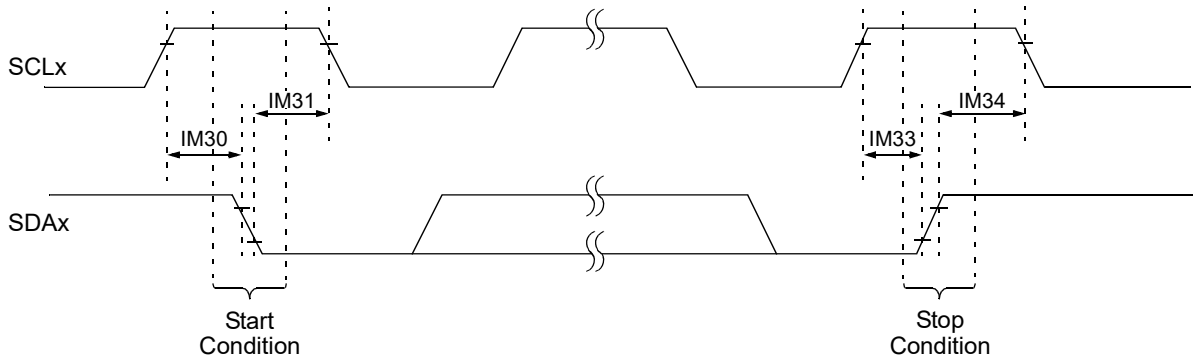


Figure 37-15. I2Cx Bus Data Timing Characteristics (Host Mode)

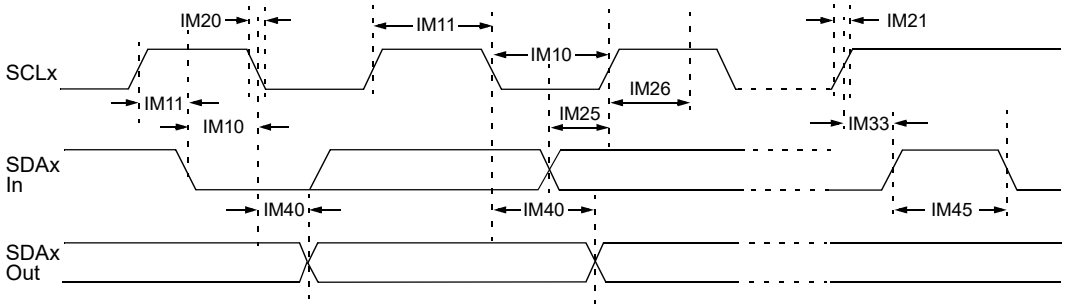


Table 37-37. I2Cx Bus Data Timing Requirements (Host Mode)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	**Symbol**	**Characteristic(4)**		**Min.(1)**	**Max.**	**Units**	**Conditions**
IM10	Tlo:scl	Clock Low Time	100 kHz mode	T_{CY} (BRG + 1)	—	µs	
			400 kHz mode	T_{CY} (BRG + 1)	—	µs	
			1 MHz mode(2)	T_{CY} (BRG + 1)	—	µs	
IM11	Thi:scl	Clock High Time	100 kHz mode	T_{CY} (BRG + 1)	—	µs	
			400 kHz mode	T_{CY} (BRG + 1)	—	µs	
			1 MHz mode(2)	T_{CY} (BRG + 1)	—	µs	

Notes:

1. BRG is the value of the I^2C Baud Rate Generator.
2. Maximum Pin Capacitance = 10 pF for all I2Cx pins (for 1 MHz mode only).
3. Typical value for this parameter is 130 ns.
4. These parameters are characterized but not tested in manufacturing.

.....continued

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic(4)		Min.(1)	Max.	Units	Conditions
IM20	Tf:scl	SDAx and SCLx Fall Time	100 kHz mode	—	300	ns	Cb is specified to be from 10 to 400 pF
			400 kHz mode	20 x (V_{DD}/5.5V)	300	ns	
			1 MHz mode(2)	—	120	ns	
IM21	Tr:scl	SDAx and SCLx Rise Time	100 kHz mode	—	1000	ns	Cb is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1 Cb	300	ns	
			1 MHz mode(2)	—	120	ns	
IM25	Tsu:dat	Data Input Setup Time	100 kHz mode	250	—	ns	
			400 kHz mode	100	—	ns	
			1 MHz mode(2)	50	—	ns	
IM26	Thd:dat	Data Input Hold Time	100 kHz mode	0	—	µs	
			400 kHz mode	0	0.9	µs	
			1 MHz mode(2)	0	0.3	µs	
IM30	Tsu:sta	Start Condition Setup Time	100 kHz mode	T_{CY} (BRG + 1)	—	µs	Only relevant for Repeated Start condition
			400 kHz mode	T_{CY} (BRG + 1)	—	µs	
			1 MHz mode(2)	T_{CY} (BRG + 1)	—	µs	
IM31	Thd:sta	Start Condition Hold Time	100 kHz mode	T_{CY} (BRG + 1)	—	µs	After this period, the first clock pulse is generated
			400 kHz mode	T_{CY} (BRG + 1)	—	µs	
			1 MHz mode(2)	T_{CY} (BRG + 1)	—	µs	
IM33	Tsu:sto	Stop Condition Setup Time	100 kHz mode	T_{CY} (BRG + 1)	—	µs	
			400 kHz mode	T_{CY} (BRG + 1)	—	µs	
			1 MHz mode(2)	T_{CY} (BRG + 1)	—	µs	
IM34	Thd:sto	Stop Condition Hold Time	100 kHz mode	T_{CY} (BRG + 1)	—	µs	
			400 kHz mode	T_{CY} (BRG + 1)	—	µs	
			1 MHz mode(2)	T_{CY} (BRG + 1)	—	µs	
IM40	Taa:scl	Output Valid from Clock	100 kHz mode	—	3450	ns	
			400 kHz mode	—	900	ns	
			1 MHz mode(2)	—	450	ns	

Notes:

1. BRG is the value of the I^2C Baud Rate Generator.
2. Maximum Pin Capacitance = 10 pF for all I2Cx pins (for 1 MHz mode only).
3. Typical value for this parameter is 130 ns.
4. These parameters are characterized but not tested in manufacturing.

.....continued

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic[4]		Min.[1]	Max.	Units	Conditions
IM45	Tbf:sda	Bus Free Time	100 kHz mode	4.7	—	µs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	µs	
			1 MHz mode[2]	0.5	—	µs	
IM50	Cb	Bus Capacitive Loading		—	400	pF	
IM51	Tpgd	Pulse Gobbler Delay[3]		65	390	ns	

Notes:

1. BRG is the value of the I^2C Baud Rate Generator.
2. Maximum Pin Capacitance = 10 pF for all I2Cx pins (for 1 MHz mode only).
3. Typical value for this parameter is 130 ns.
4. These parameters are characterized but not tested in manufacturing.

Figure 37-16. I2Cx Bus Start/Stop Bits Timing Characteristics (Client Mode)

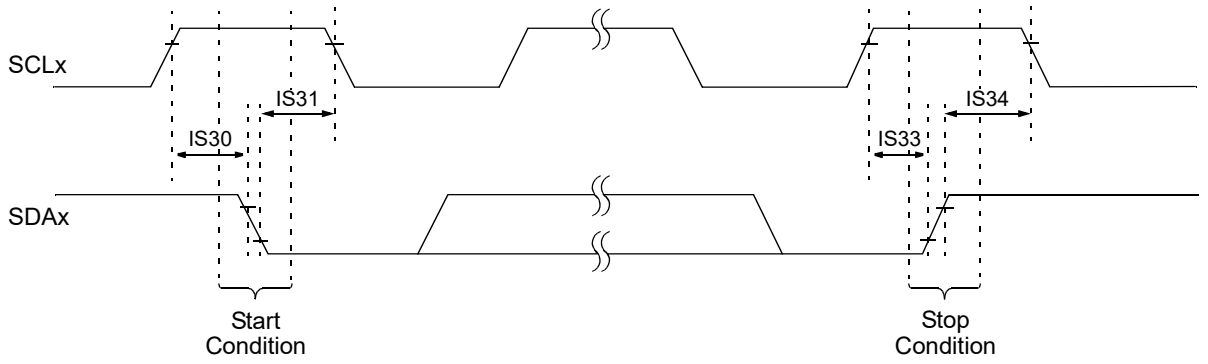


Figure 37-17. I2Cx Bus Data Timing Characteristics (Client Mode)

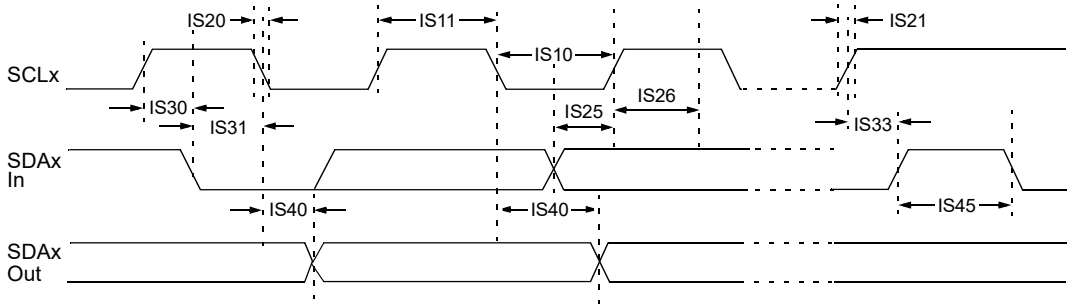


Table 37-38. I2Cx Bus Data Timing Requirements (Client Mode)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	**Symbol**	**Characteristic(3)**		**Min.**	**Max.**	**Units**	**Conditions**
IS10	Tlo:scl	Clock Low Time	100 kHz mode	4.7	—	µs	
			400 kHz mode	1.3	—	µs	
			1 MHz mode(1)	0.5	—	µs	
IS11	Thi:scl	Clock High Time	100 kHz mode	4.0	—	µs	Device must operate at a minimum of 1.5 MHz
			400 kHz mode	0.6	—	µs	Device must operate at a minimum of 10 MHz
			1 MHz mode(1)	0.28	—	µs	
IS20	Tf:scl	SDAx and SCLx Fall Time	100 kHz mode	—	300	ns	Cb is specified to be from 10 to 400 pF
			400 kHz mode	20 x (V_{DD}/5.5V)	300	ns	
			1 MHz mode(1)	20 x (V_{DD}/5.5V)	120	ns	
IS21	Tr:scl	SDAx and SCLx Rise Time	100 kHz mode	20 + 0.1 Cb	1000	ns	Cb is specified to be from 10 to 400 pF
			400 kHz mode	—	300	ns	
			1 MHz mode(1)	—	120	ns	
IS25	Tsu:dat	Data Input Setup Time	100 kHz mode	250	—	ns	
			400 kHz mode	100	—	ns	
			1 MHz mode(1)	50	—	ns	
IS26	Thd:dat	Data Input Hold Time	100 kHz mode	0	—	µs	
			400 kHz mode	0	0.9	µs	
			1 MHz mode(1)	0	0.3	µs	
IS30	Tsu:sta	Start Condition Setup Time	100 kHz mode	4.7	—	µs	Only relevant for Repeated Start condition
			400 kHz mode	0.6	—	µs	
			1 MHz mode(1)	0.26	—	µs	
IS31	Thd:sta	Start Condition Hold Time	100 kHz mode	4.0	—	µs	After this period, the first clock pulse is generated
			400 kHz mode	0.6	—	µs	
			1 MHz mode(1)	0.26	—	µs	
IS33	Tsu:sto	Stop Condition Setup Time	100 kHz mode	4	—	µs	
			400 kHz mode	0.6	—	µs	
			1 MHz mode(1)	0.26	—	µs	

Notes:

1. Maximum Pin Capacitance = 10 pF for all I2Cx pins (for 1 MHz mode only).
2. Typical value for this parameter is 130 ns.
3. These parameters are characterized but not tested in manufacturing.

.....continued

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic[3]		Min.	Max.	Units	Conditions
IS34	Thd:sto	Stop Condition Hold Time	100 kHz mode	> 0	—	µs	
			400 kHz mode	> 0	—	µs	
			1 MHz mode[1]	> 0		µs	
IS40	Taa:scl	Output Valid from Clock	100 kHz mode	0	3540	ns	
			400 kHz mode	0	900	ns	
			1 MHz mode[1]	0	400	ns	
IS45	Tbf:sda	Bus Free Time	100 kHz mode	4.7	—	µs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	µs	
			1 MHz mode[1]	0.5	—	µs	
IS50	C_B	Bus Capacitive Loading		—	400	pF	
IS51	T_{PGD}	Pulse Gobbler Delay[2]		65	390	ns	

Notes:

1. Maximum Pin Capacitance = 10 pF for all I2Cx pins (for 1 MHz mode only).
2. Typical value for this parameter is 130 ns.
3. These parameters are characterized but not tested in manufacturing.

Figure 37-18. UARTx Module I/O Timing Characteristics

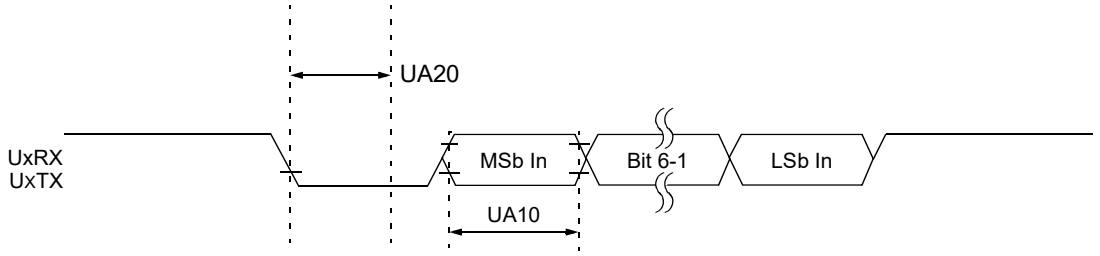


Table 37-39. UARTx Module I/O Timing Requirements[1]

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤+85°C for Industrial Operating temperature -40°C ≤ T_A ≤ +125°C							
Param No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
UA10	T_{BAUD}	UARTx Baud Time	12.5	—	—	ns	
UA11	F_{BAUD}	UARTx Baud Frequency	—	—	20	Mbps	BRGS = 0, 16x divide
			—	—	50	Mbps	BRGS = 1, 4x divide or CLKMOD = 1, (fractional)
UA20	T_{SBPM}	Start Bit Pulse Width to Trigger UARTx Wake-up	50	—	—	ns	

Note:

1. Parameters characterized but not tested in manufacturing.

Table 37-40. ADC Module Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	**Symbol**	**Characteristics**	**Min.**	**Typical**	**Max.**	**Units**	**Conditions**
Analog Input							
AD12	$V_{INH} - V_{INL}$	Full-Scale Input Span	AV_{SS}	—	AV_{DD}	V	
AD14	V_{IN}	Absolute Input Voltage	AV_{SS} – 0.3	—	AV_{DD} + 0.3	V	
AD17	R_{IN}	Recommended Impedance of Analog Voltage Source(1)	—	100	—	Ω	For minimum sampling time
AD60	C_{HOLD}	Hold Capacitor Capacitance(2)	—	1	—	pF	
AD61	C_{PIN}	Pin Capacitance(2)	—	4	—	pF	Shared core
AD62	R_{IC}	Input resistance(2)	—	1000	—	Ω	Includes R_{SS}
AD66	V_{BG}	Internal Voltage Reference Source	0.784	0.8	0.816	V	From 3/4 band gap buffer
AD67	V_{REF}	Internal 15/16 V_{DD} Reference(2)	-15	5	15	Lsb	
AD68	T_{REF}	Internal 15/16 V_{DD}Reference Sampling Time(2)	32	—	—	ns	
ADC Accuracy							
AD20	N_R	Resolution	12 data bits			bits	
AD21	INL	Integral Nonlinearity	-3	—	3	LSb	V_{DD} = 3.3V, AV_{DD} = 3.3V Gain error uncompensated
AD22	DNL	Differential Nonlinearity	-1	—	2	LSb	
AD23	GERR	Gain Error	-130	—	50	LSb	
AD24	OERR	Offset Error	—	5	—	LSb	
AD23a	GERR	Gain Error(2)	-15	—	5	LSb	V_{DD} = 3.3V, AV_{DD} = 3.3V Gain error compensated
AD25	—	Monotonicity	—	—	—	—	**Guaranteed**
Dynamic Performance							
AD34	ENOB	Effective Number of Bits(2,3)	—	10.5	—	bits	
AD50	T_{AD}	ADC Clock Period	12.5	—	125	ns	T_{AD} = FIN/4
AD51	F_{TP}	Throughput Rate(1,4)	—	—	40	Msps	

Notes:

1. Parameters not characterized or tested in manufacturing.
2. Parameters characterized but not tested in manufacturing.
3. Characterized with a 1 kHz sine wave.
4. Throughput includes 1.5 T_{AD} conversion time.
5. Data in the “Typ” column are at 3.3V, +25°C. Parameters are for design guidance only and are not tested.

Table 37-41. Die Temperature Diode Specifications(1)

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	**Symbol**	**Characteristic**	**Min.**	**Typ.**	**Max.**	**Units**	**Comments**
TD01	T_{COEFF}	Temperature Coefficient(1)	—	1.5	—	mV/C	

Note:

1. Parameters are for design guidance only and are not tested.

.....continued

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Comments
TD02	T_{SAMPLE}	Sampling Time	0.3125	—	—	µS	

Note:

1. Parameters are for design guidance only and are not tested.

Table 37-42. High-Speed Analog Comparator Module Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Comments
CM10	V_{IOFF}	Input Offset Voltage	-35	—	+35	mV	All packages except 28 SSOP
			-70	—	+70	mV	For 28 SSOP package only
CM11	V_{ICM}	Input Common-Mode Voltage Range(1)	AV_{SS}	—	AV_{DD}	V	
CM13	CMRR	Common-Mode Rejection Ratio(1)	60	—	—	dB	
CM14	T_{RESP}	Large Signal Response(1)	—	5	—	ns	V+ input step of 100 mV while V- input is held at $AV_{DD}/2$
CM15	V_{HYST}	Input Hysteresis(1)	15	30	45	mV	Depends on HYSSEL[1:0]

Note:

1. Parameters are characterized but not tested in manufacturing.

Table 37-43. DACx Module Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Comments
DA02	CV_{RES}	Resolution		12		bits	
DA03	INL	Integral Nonlinearity Error(2)	-25	—	35	LSB	
DA04	DNL	Differential Nonlinearity Error(2)	-5	—	5	LSB	
DA05	EOFF	Offset Error(2)	-10	—	20	LSB	Internal node at comparator input
DA06	EG	Gain Error(2)	-10	—	50	LSB	Internal node at comparator input

Notes:

1. Parameters are for design guidance only and are not tested in manufacturing.
2. DAC output codes from 5% to 95%. DAC operational at values <5% or >95%, but may not meet specifications listed.

.....continued

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Comments
DA07	T_{SET}	Settling Time(1)	600	750	2000	ns	Output with 1% of desired output voltage with a 5-95% or 95-5% step
DA08	V_{OUT}	Voltage Output Range	0.165	—	3.135	V	V_{DD} = 3.3V
DA09	T_{TR}	Transition Time(1)	340	—	—	ns	
DA10	T_{SS}	Steady-State Time(1)	550	—	—	ns	

Notes:

1. Parameters are for design guidance only and are not tested in manufacturing.
2. DAC output codes from 5% to 95%. DAC operational at values <5% or >95%, but may not meet specifications listed.

Table 37-44. DACx Output (DACOUTx Pins) Specifications(1)

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Comments
DA11	R_{LOAD}	Resistive Output Load Impedance	10K	—	—	Ohm	
DA11a	C_{LOAD}	Output Load Capacitance	—	—	30	pF	Including output pin capacitance
DA12	I_{OUT}	Output Current Drive Strength	-3	—	3	mA	Sink and source

Note:

1. Parameters are for design guidance only and are not tested in manufacturing.

Table 37-45. Current Bias Generator Specifications(1)

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
CC03	I10SRC	10 µA Source Current	8	—	12	µA	ISRCx pin
CC04	ISELOUT	0-200 µA Selectable Output Source Current	—	±15	—	%	IBIASx pin

Note:

1. Parameters are for design guidance only and are not tested in manufacturing.

Table 37-46. Operational Amplifier Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Sym	Characteristic	Min	Typ	Max	Units	Comments
OA01	GBWP	Gain Bandwidth Product(1)	—	50	—	MHz	Low-Power mode
			—	100	—	MHz	High-Power mode
OA02	SR	Slew Rate(1)	—	10	—	V/µs	Low-Power mode. Measured from 0.5 to 2.5 Volts with a step change in input voltage
			—	100	—	V/µs	High-Power mode. Measured from 0.5 to 2.5 Volts with a step change in input voltage
OA03	V_{IOFF}	Input Offset Voltage	-3(3)	-1/+1	+3(3)	mV	Unity gain configuration, High-Power mode
OA04	V_{IBC}	Input Bias Current(2)	—	—	—	nA	
OA05	V_{ICM}	Common-Mode Input Voltage Range(1)	AV_{SS}	—	AV_{DD}	V	
OA07	CMRR	Common-Mode Rejection Ratio(1)	—	80	—	dB	
OA08	PSRR	Power Supply Rejection Ratio(1)	—	60	—	dB	At 10 kHz
OA09	V_{OR}	Output Voltage Range(1)	AV_{SS}	—	AV_{DD}	mV	0.5V input overdrive, no output loading
OA11	C_{LOAD}	Output Load Capacitance(1)	—	—	30	pF	Including output pin capacitance
OA12	I_{OUT}	Output Current Drive Strength(1)	—	10	—	mA	Sink and source
OA13	P_{MARGIN}	Phase Margin(1)	65	—	—	degree	Unity gain
OA14	G_{MARGIN}	Gain Margin(1)	20	—	—	dB	Unity gain
OA15	OLG	Open-Loop Gain(1)	80	90	—	dB	

.....continued

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Sym	Characteristic	Min	Typ	Max	Units	Comments

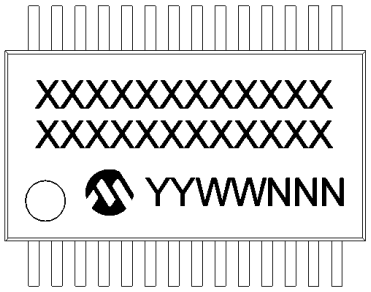
Notes:

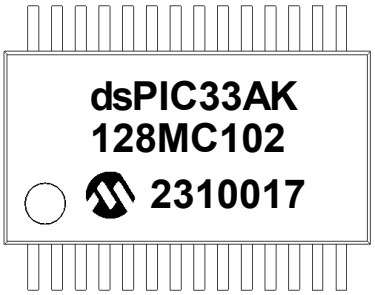
1. These parameters are not characterized or tested in manufacturing.
2. The op amps use CMOS input circuitry with negligible input bias current. The maximum “effective bias current” is the I/O pin leakage specified by electrical Parameter DI50.
3. Parameters are characterized but not tested in manufacturing.

38. Packaging Information

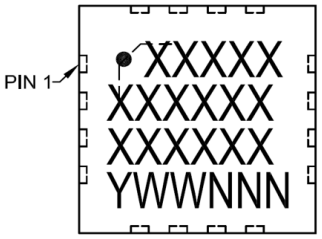
38.1 Package Marking Information

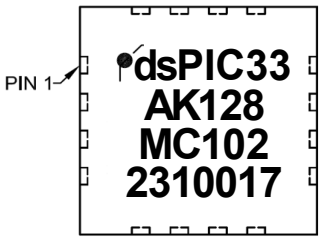
28-Lead SSOP (5.30 mm)



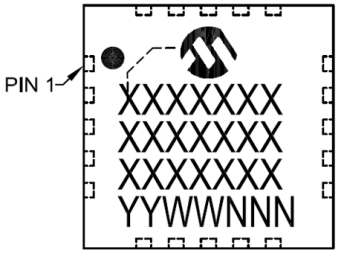


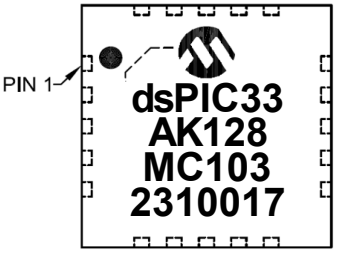
28-Lead vQFN (4x4 mm)





36-Lead vQFN (5x5 mm)





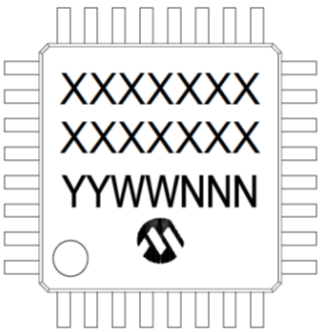
Legend:	XX...X	Customer-specific information
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

48-Lead vQFN (6x6 mm)

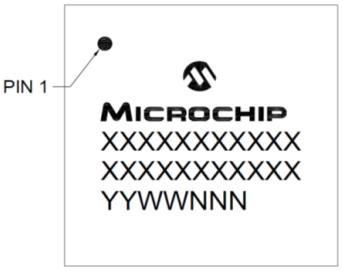
Example

48-Lead TQFP (7x7 mm)

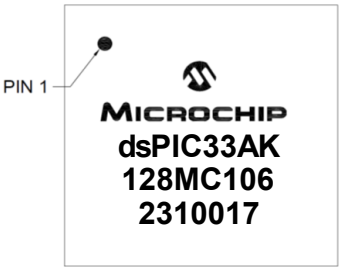


Example

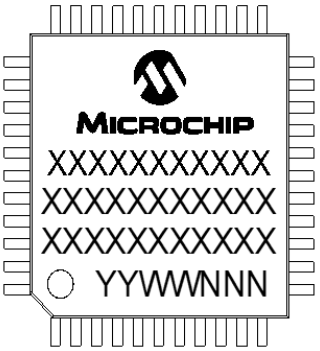
64-Lead vQFN (9x9 mm)



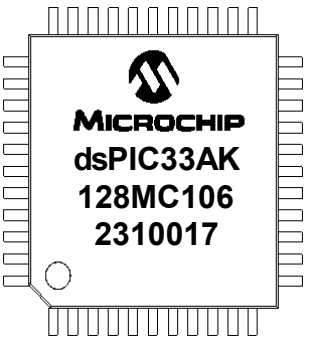
Example



64-Lead TQFP (10x10 mm)



Example



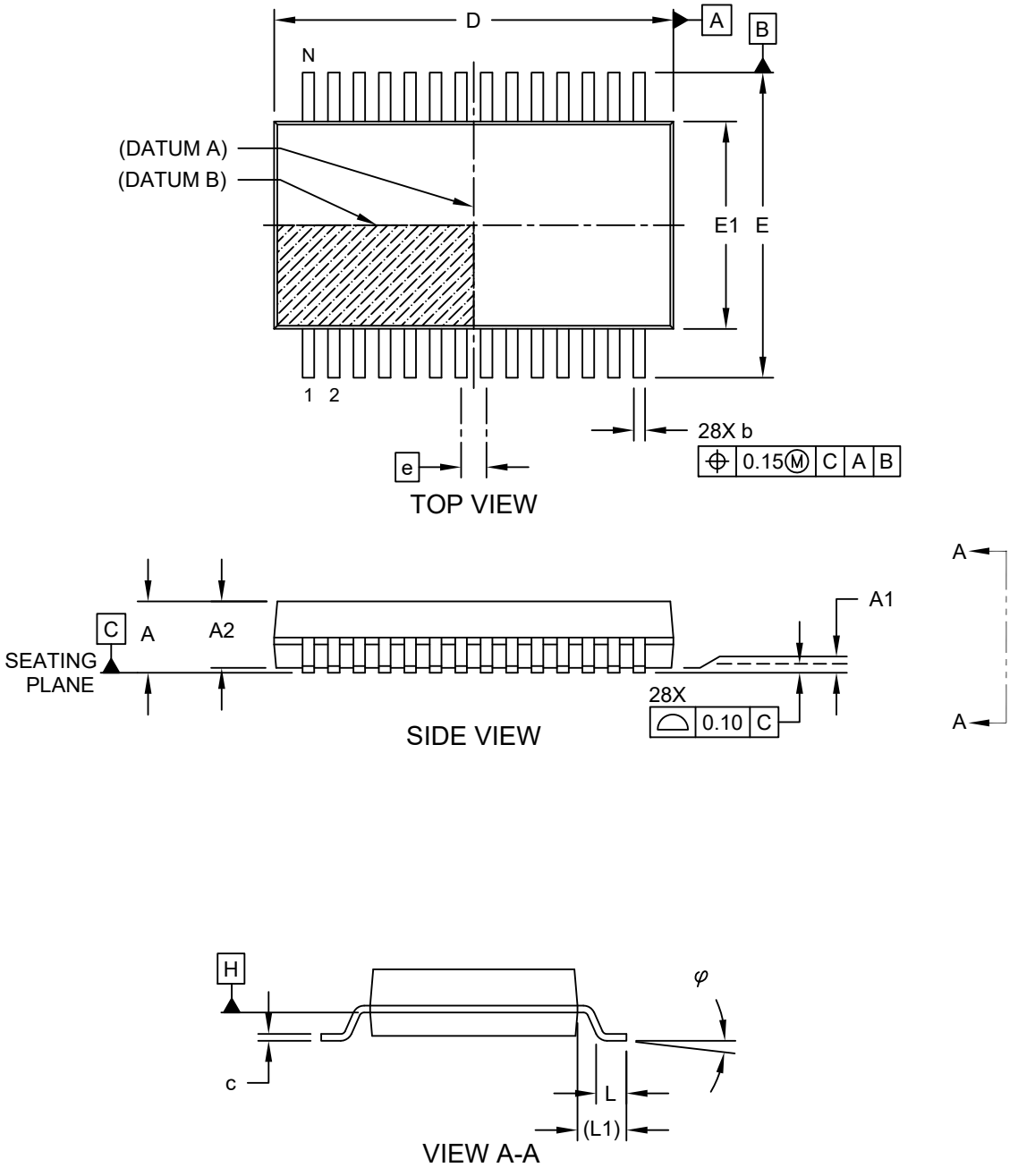
Legend:	XX...X	Customer-specific information
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

38.2 Package Details

28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

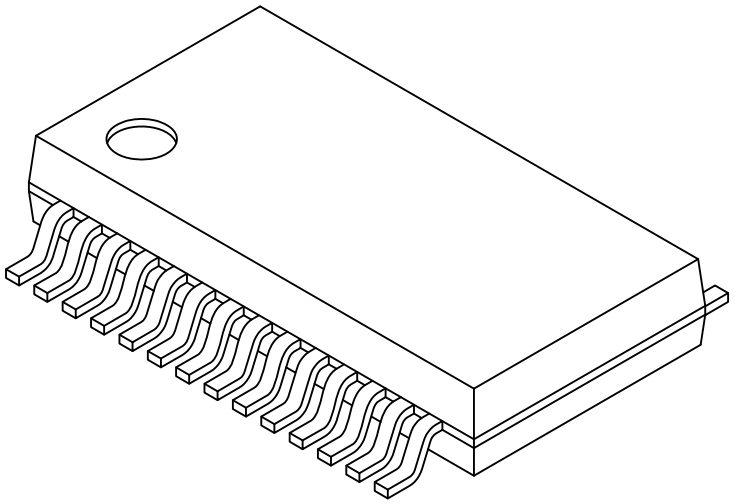
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-073 Rev C Sheet 1 of 2

28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		0.65 BSC	
Overall Height	A	-	-	2.00
Molded Package Thickness	A2	1.65	1.75	1.85
Standoff	A1	0.05	-	-
Overall Width	E	7.40	7.80	8.20
Molded Package Width	E1	5.00	5.30	5.60
Overall Length	D	9.90	10.20	10.50
Foot Length	L	0.55	0.75	0.95
Footprint	L1		1.25 REF	
Lead Thickness	c	0.09	-	0.25
Foot Angle	φ	0°	4°	8°
Lead Width	b	0.22	-	0.38

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M

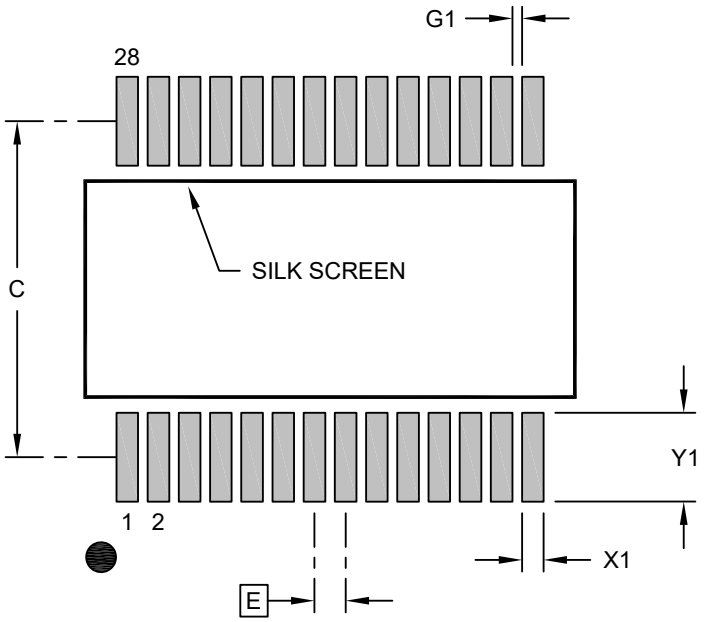
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-073 Rev C Sheet 2 of 2

28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E	0.65 BSC		
Contact Pad Spacing	C		7.00	
Contact Pad Width (X28)	X1			0.45
Contact Pad Length (X28)	Y1			1.85
Contact Pad to Center Pad (X26)	G1	0.20		

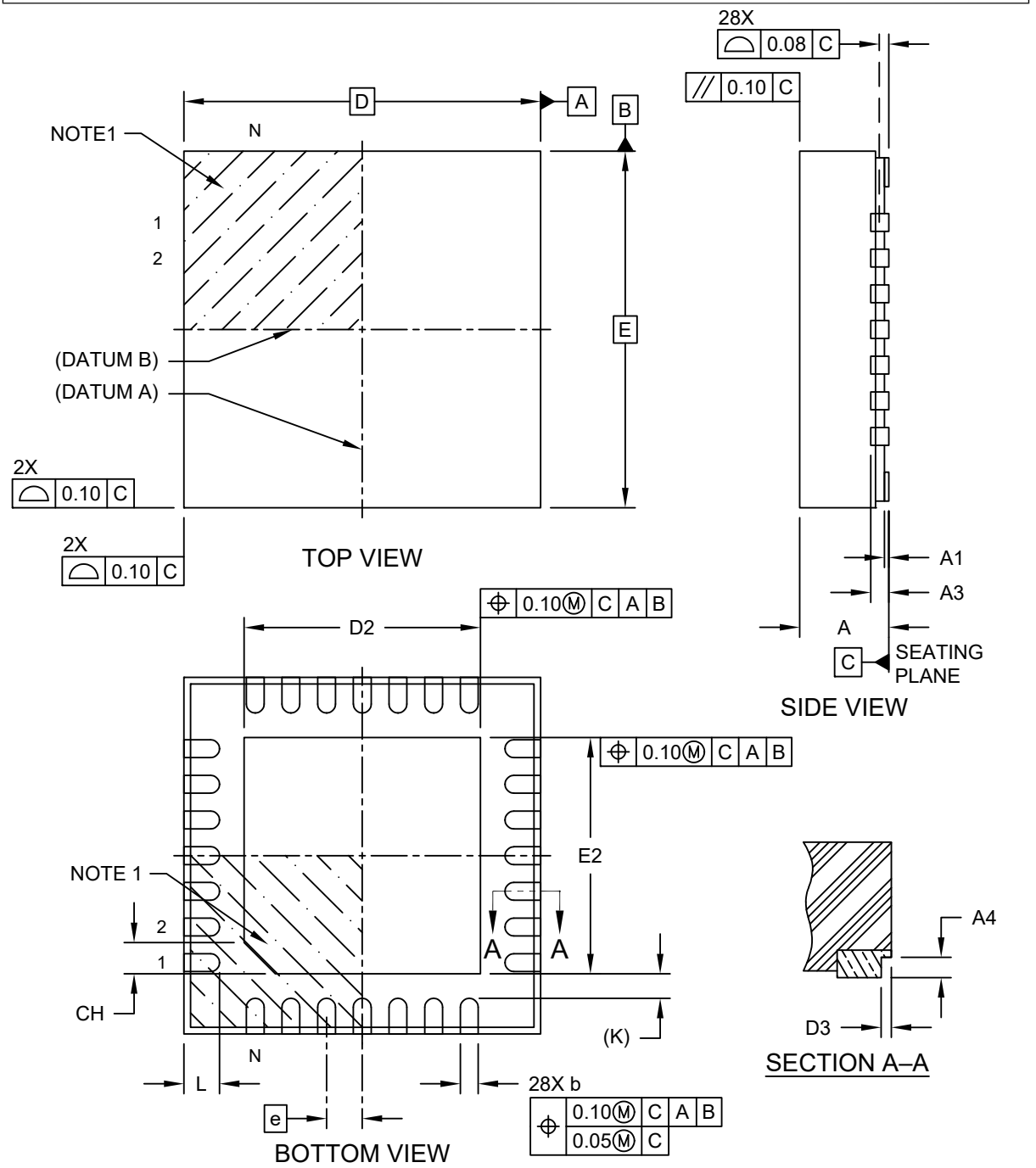
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2073 Rev B

28-Lead Very Thin Plastic Quad Flat, No Lead Package (3LW) – 4x4x1.0 mm Body [VQFN] With 2.65 mm Exposed Pad and Stepped Wettable Flanks

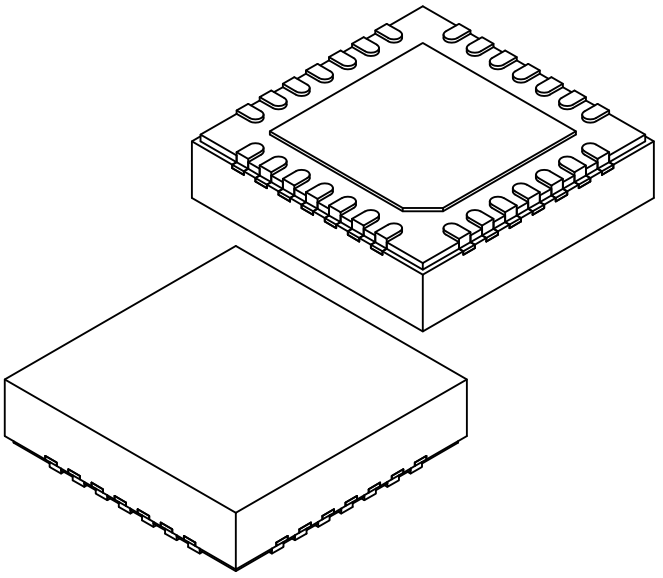
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-565-3LW Rev B Sheet 1 of 2

28-Lead Very Thin Plastic Quad Flat, No Lead Package (3LW) – 4x4x1.0 mm Body [VQFN] With 2.65 mm Exposed Pad and Stepped Wettable Flanks

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		28	
Pitch	e		0.40 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.203 REF	
Overall Length	D		4.00 BSC	
Exposed Pad Length	D2	2.55	2.65	2.75
Overall Width	E		4.00 BSC	
Exposed Pad Width	E2	2.55	2.65	2.75
Exposed Pad Index Chamfer	CH		0.35 REF	
Terminal Width	b	0.15	0.20	0.25
Terminal Length	L	0.30	0.40	0.50
Terminal-to-Exposed-Pad	K		0.275 REF	
Wettable Flank Step Cut Length	D3	–	–	0.085
Wettable Flank Step Cut Height	A4	0.10	–	0.19

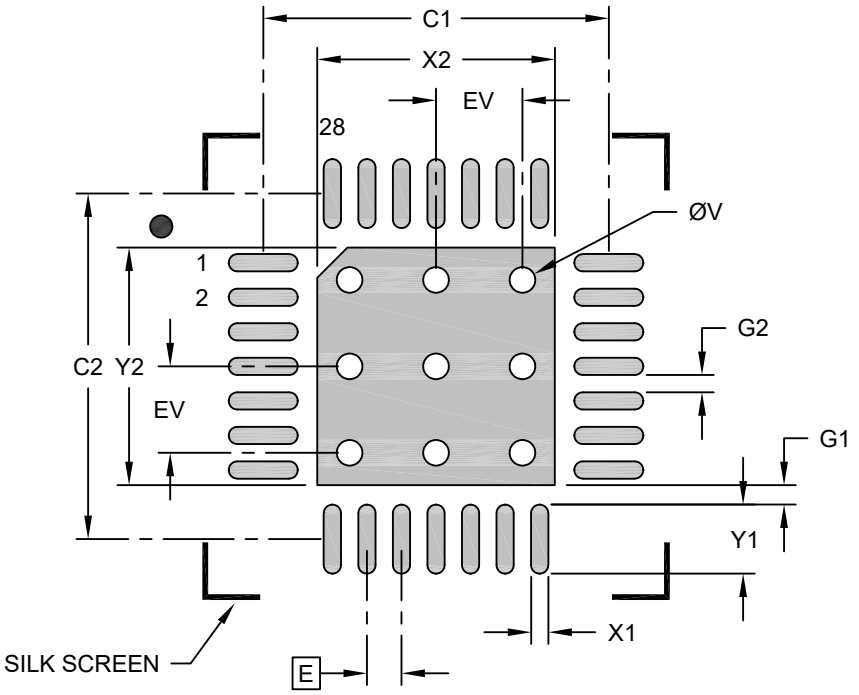
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-565-3LW Rev B Sheet 2 of 2

28-Lead Very Thin Plastic Quad Flat, No Lead Package (3LW) – 4x4x1.0 mm Body [VQFN] With 2.65 mm Exposed Pad and Stepped Wettable Flanks

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.40 BSC	
Center Pad Width	X2			2.75
Center Pad Length	Y2			2.75
Contact Pad Spacing	C1		4.00	
Contact Pad Spacing	C2		4.00	
Contact Pad Width (X28)	X1			0.20
Contact Pad Length (X28)	Y1			0.80
Contact Pad to Center Pad (X28)	G1	0.23		
Contact Pad to Contact Pad (X24)	G2	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

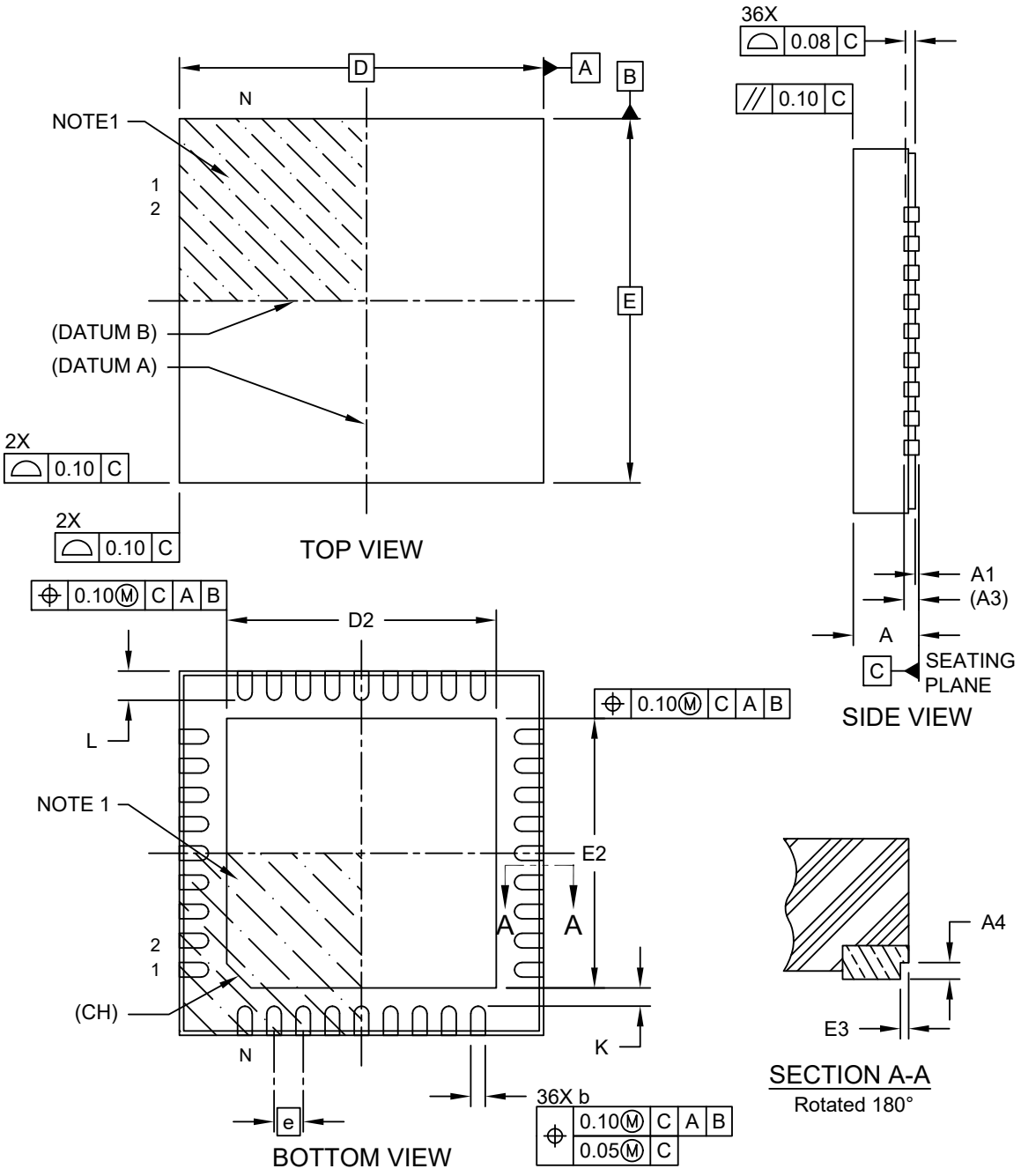
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2565-3LW Rev B

36-Lead Very Thin Plastic Quad Flat, No Lead Package (4AW) - 5x5x0.9 mm Body [VQFN] With 3.7x3.7 mm Exposed Pad and Step Cut Wettable Flanks

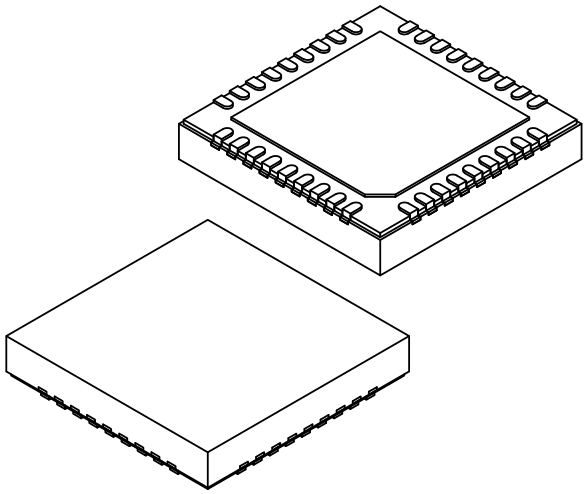
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-579-4AW Rev B Sheet 1 of 2

36-Lead Very Thin Plastic Quad Flat, No Lead Package (4AW) - 5x5x0.9 mm Body [VQFN] With 3.7x3.7 mm Exposed Pad and Step Cut Wettable Flanks

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		36	
Pitch	e		0.40 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.203 REF	
Overall Length	D		5.00 BSC	
Exposed Pad Length	D2	3.60	3.70	3.80
Overall Width	E		5.00 BSC	
Exposed Pad Width	E2	3.60	3.70	3.80
Exposed Pad Chamfer	CH		0.35 REF	
Terminal Width	b	0.15	0.20	0.25
Terminal Length	L	0.30	0.40	0.50
Terminal-to-Exposed-Pad	K	0.25	-	-
Wettable Flank Step Cut Width	E3	0.035	-	0.085
Wettable Flank Step Cut Depth	A4	0.100	-	0.190

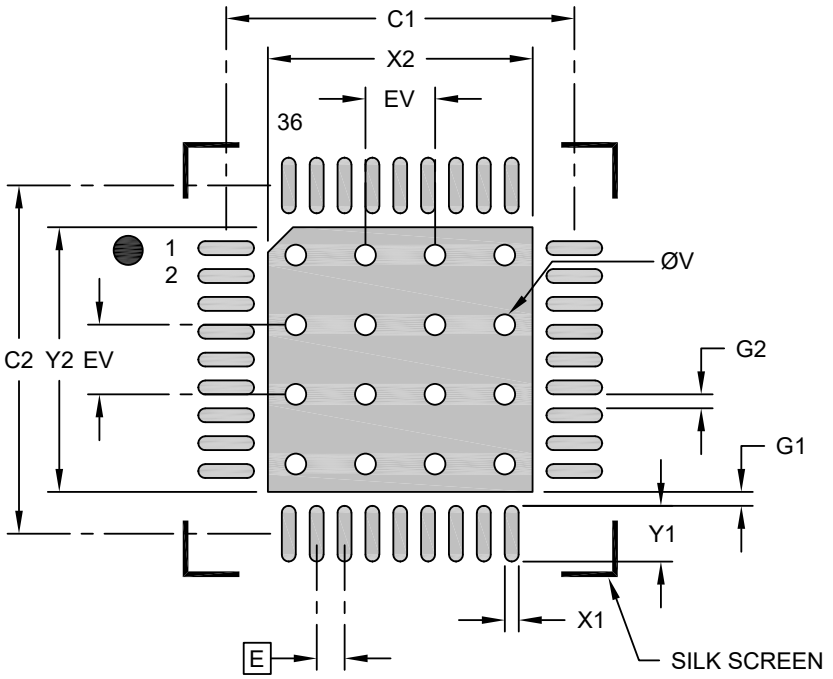
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-579-4AW Rev B Sheet 2 of 2

36-Lead Very Thin Plastic Quad Flat, No Lead Package (4AW) - 5x5x0.9 mm Body [VQFN] With 3.7x3.7 mm Exposed Pad and Step Cut Wettable Flanks

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.40 BSC	
Center Pad Width	X2			3.80
Center Pad Length	Y2			3.80
Contact Pad Spacing	C1		5.00	
Contact Pad Spacing	C2		5.00	
Contact Pad Width (X36)	X1			0.20
Contact Pad Length (X36)	Y1			0.80
Contact Pad to Center Pad (X36)	G1	0.20		
Contact Pad to Center Pad (X32)	G2	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

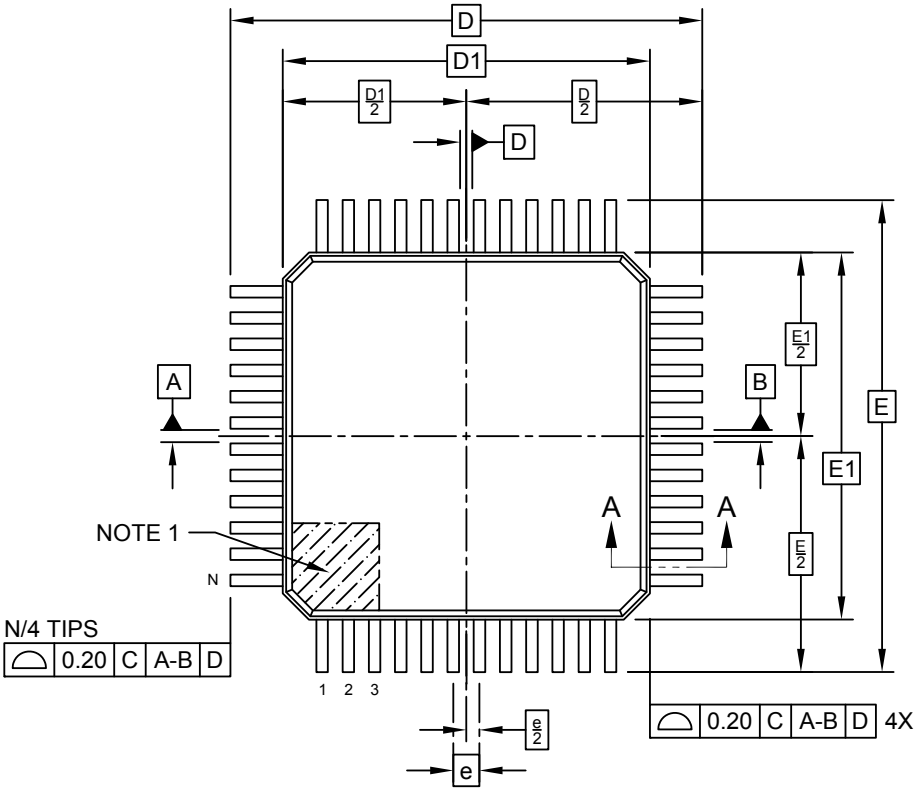
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

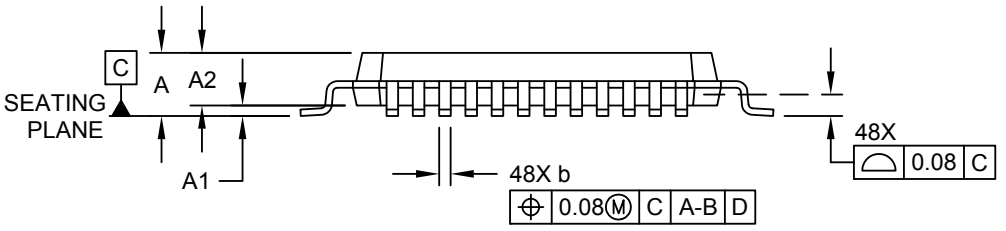
Microchip Technology Drawing C04-2579-4AW Rev B

48-Lead Plastic Thin Quad Flatpack (Y8X) - 7x7x1.0 mm Body [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW

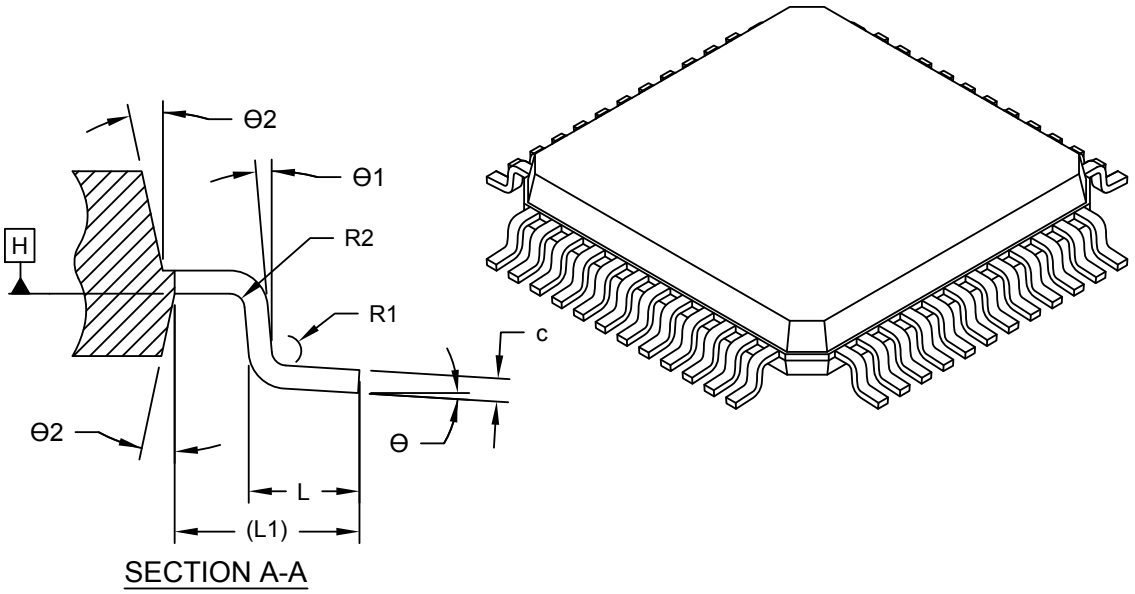


SIDE VIEW

Microchip Technology Drawing C04-300-Y8X Rev D Sheet 1 of 2

48-Lead Plastic Thin Quad Flatpack (Y8X) - 7x7x1.0 mm Body [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		48	
Pitch	e		0.50 BSC	
Overall Height	A	-	-	1.20
Standoff	A1	0.05	-	0.15
Molded Package Thickness	A2	0.95	1.00	1.05
Overall Length	D		9.00 BSC	
Molded Package Length	D1		7.00 BSC	
Overall Width	E		9.00 BSC	
Molded Package Width	E1		7.00 BSC	
Terminal Width	b	0.17	0.22	0.27
Terminal Thickness	c	0.09	-	0.16
Terminal Length	L	0.45	0.60	0.75
Footprint	L1		1.00 REF	
Lead Bend Radius	R1	0.08	-	-
Lead Bend Radius	R2	0.08	-	0.20
Foot Angle	Θ	0°	3.5°	7°
Lead Angle	Θ1	0°	-	-
Mold Draft Angle	Θ2	11°	12°	13°

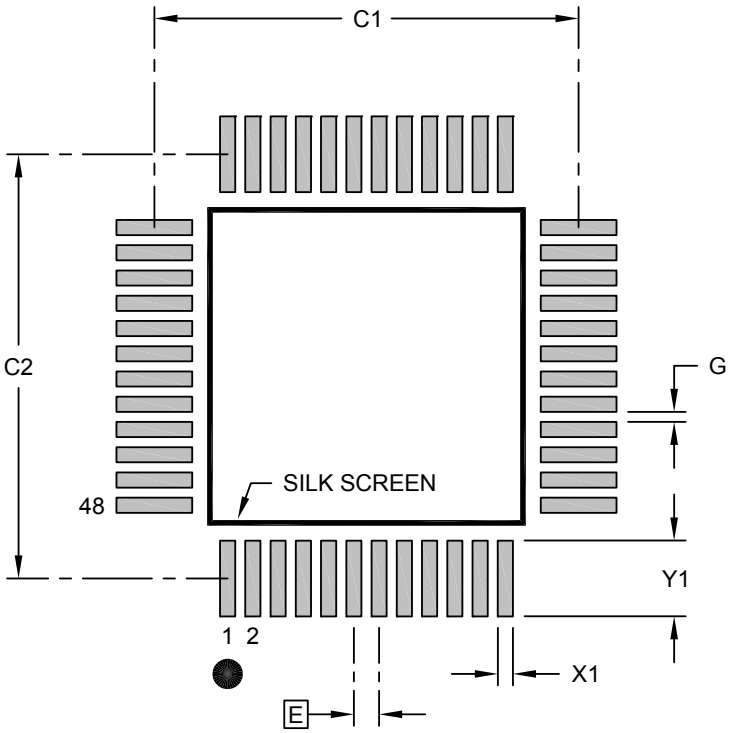
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-300-Y8X Rev D Sheet 2 of 2

48-Lead Plastic Thin Quad Flatpack (Y8X) - 7x7x1.0 mm Body [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Contact Pad Spacing	C1		8.40	
Contact Pad Spacing	C2		8.40	
Contact Pad Width (X48)	X1			0.30
Contact Pad Length (X48)	Y1			1.50
Distance Between Pads	G	0.20		

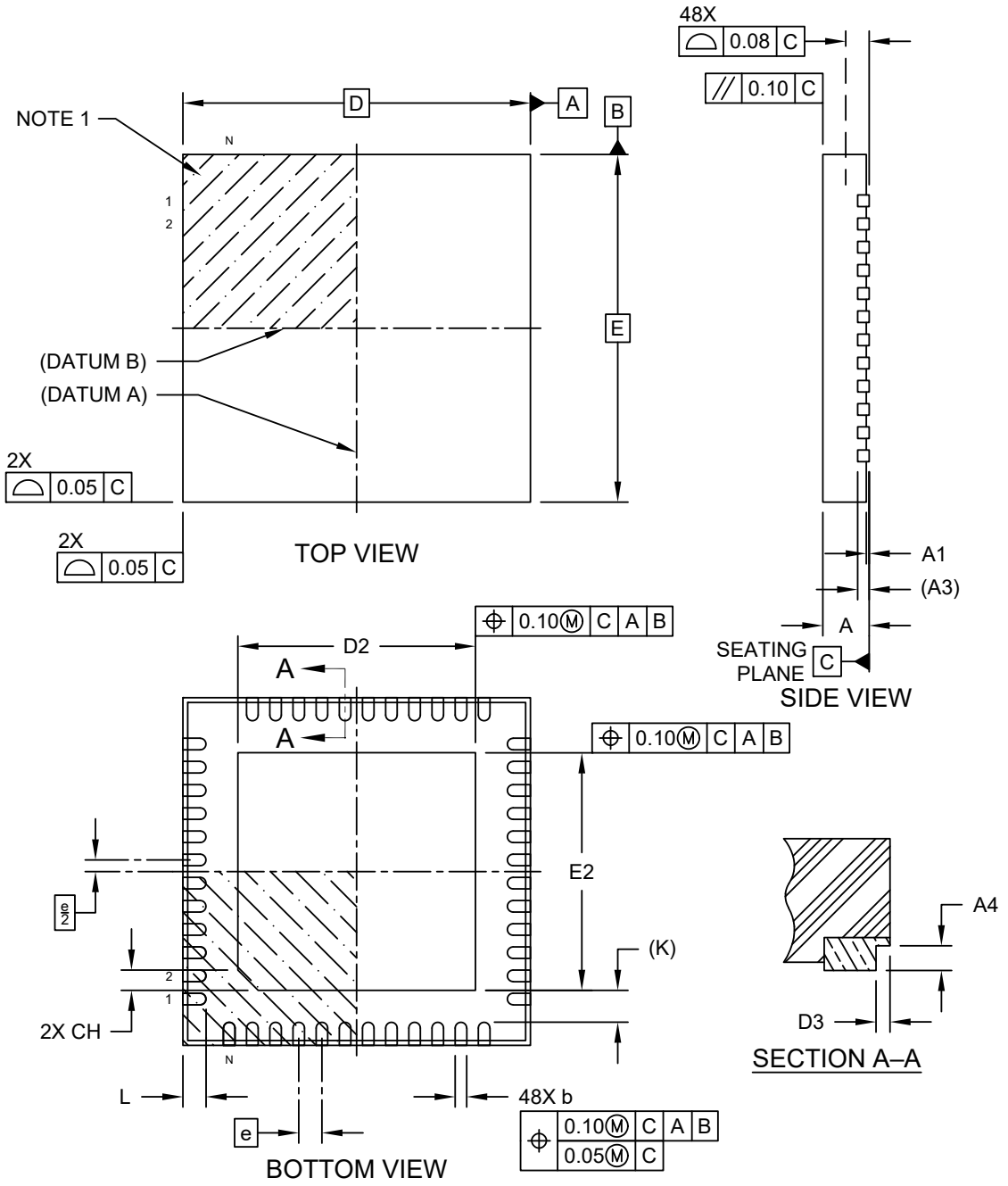
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2300-Y8X Rev D

48-Lead Very Thin Plastic Quad Flat, No Lead Package (6MX) - 6x6 mm Body [VQFN] With 4.1x4.1 mm Exposed Pad and Stepped Wettable Flanks

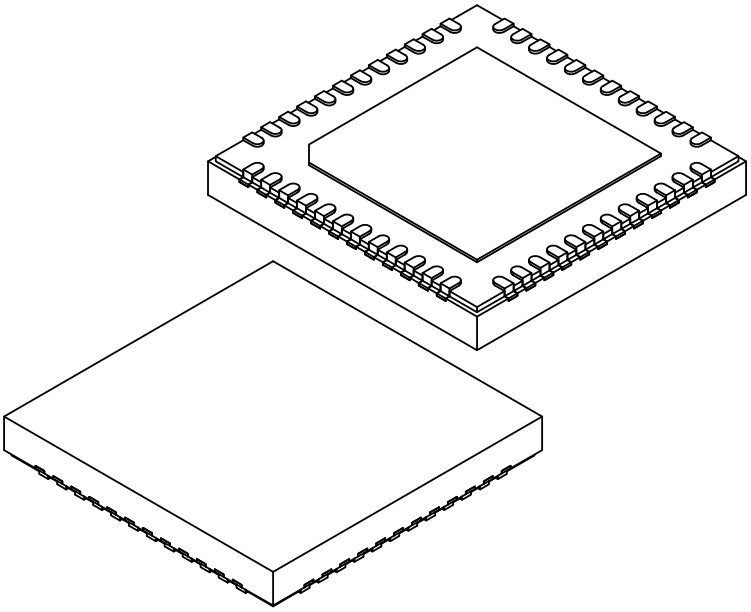
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-504-6MX Rev B Sheet 1 of 2

48-Lead Very Thin Plastic Quad Flat, No Lead Package (6MX) - 6x6 mm Body [VQFN] With 4.1x4.1 mm Exposed Pad and Stepped Wettable Flanks

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		48	
Pitch	e		0.40 BSC	
Overall Height	A	0.80	0.85	0.90
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.20 REF	
Overall Length	D		6.00 BSC	
Exposed Pad Length	D2	4.00	4.10	4.20
Overall Width	E		6.00 BSC	
Exposed Pad Width	E2	4.00	4.10	4.20
Exposed Pad Corner Chamfer	CH		0.35 REF	
Terminal Width	b	0.15	0.20	0.25
Terminal Length	L	0.30	0.40	0.50
Terminal-to-Exposed-Pad	K		0.55 REF	
Wettable Flank Step Length	D3	-	-	0.085
Wettable Flank Step Height	A4	0.10	-	0.19

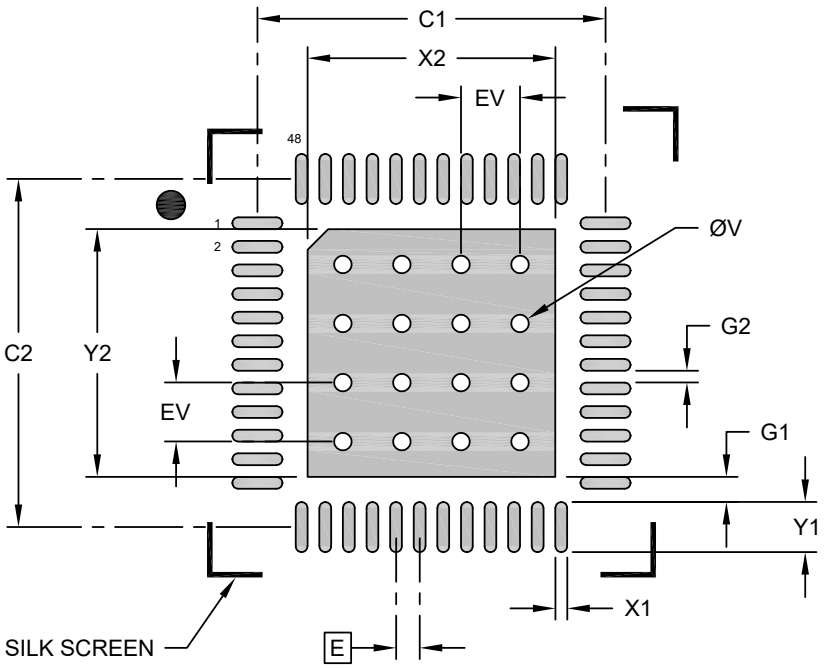
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-504-6MX Rev B Sheet 2 of 2

48-Lead Very Thin Plastic Quad Flat, No Lead Package (6MX) - 6x6 mm Body [VQFN] With 4.1x4.1 mm Exposed Pad and Stepped Wettable Flanks

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E	0.40 BSC		
Optional Center Pad Width	X2			4.20
Optional Center Pad Length	Y2			4.20
Contact Pad Spacing	C1		5.90	
Contact Pad Spacing	C2		5.90	
Contact Pad Width (X48)	X1			0.20
Contact Pad Length (X48)	Y1			0.85
Contact Pad to Center Pad (X48)	G1	0.20		
Contact Pad to Contact Pad (X44)	G2	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

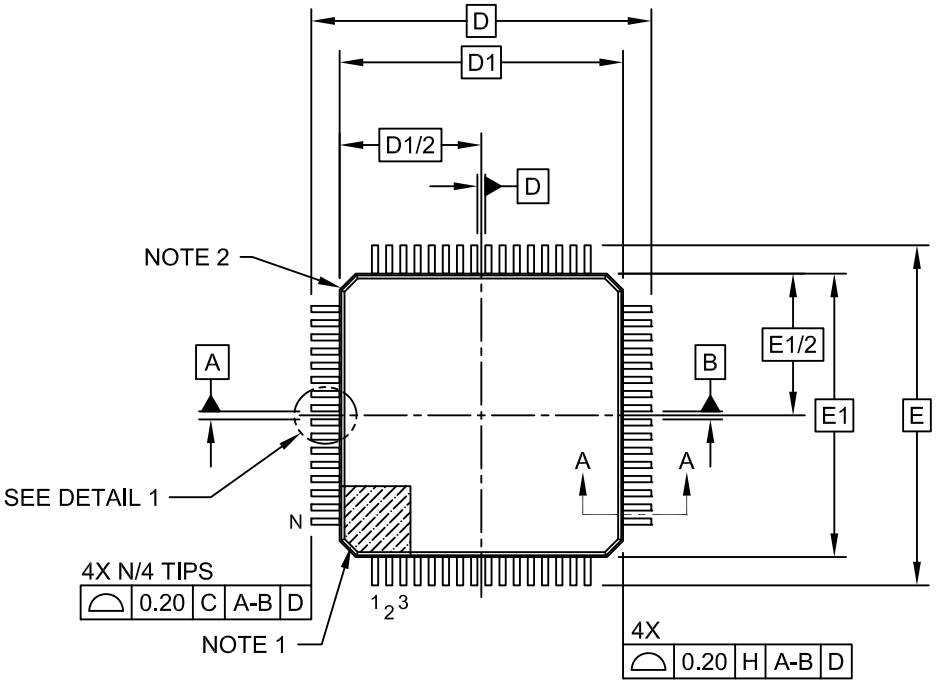
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

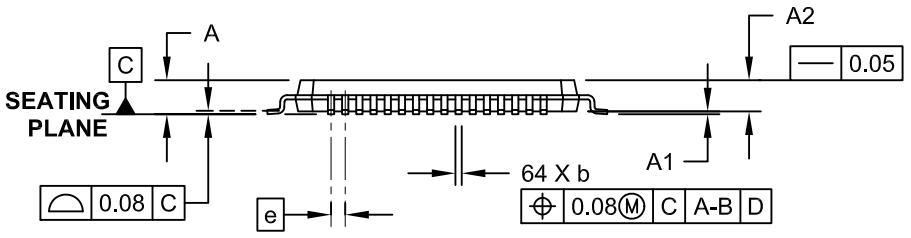
Microchip Technology Drawing C04-2504-6MX Rev B

64-Lead Plastic Thin Quad Flatpack (V2X)-10x10x1 mm Body, 2.00 mm Footprint [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW

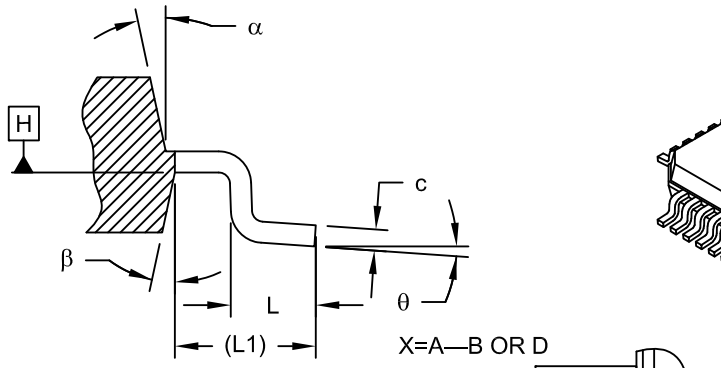


SIDE VIEW

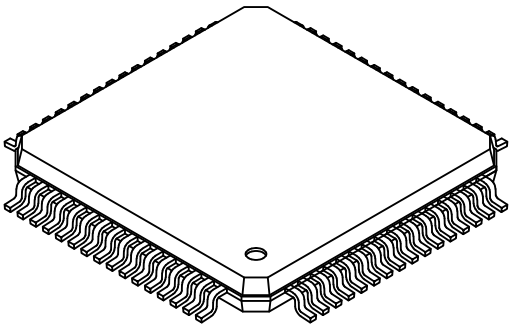
Microchip Technology Drawing C04-085-V2X Rev E Sheet 1 of 2

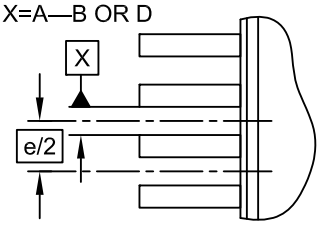
64-Lead Plastic Thin Quad Flatpack (V2X)-10x10x1 mm Body, 2.00 mm Footprint [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



SECTION A-A





DETAIL 1

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Leads	N		64	
Lead Pitch	e		0.50 BSC	
Overall Height	A	-	-	1.20
Molded Package Thickness	A2	0.95	1.00	1.05
Standoff	A1	0.05	-	0.15
Foot Length	L	0.45	0.60	0.75
Footprint	L1		1.00 REF	
Foot Angle	θ	0°	3.5°	7°
Overall Width	E		12.00 BSC	
Overall Length	D		12.00 BSC	
Molded Package Width	E1		10.00 BSC	
Molded Package Length	D1		10.00 BSC	
Lead Thickness	c	0.09	-	0.20
Lead Width	b	0.17	0.22	0.27
Mold Draft Angle Top	α	11°	12°	13°
Mold Draft Angle Bottom	β	11°	12°	13°

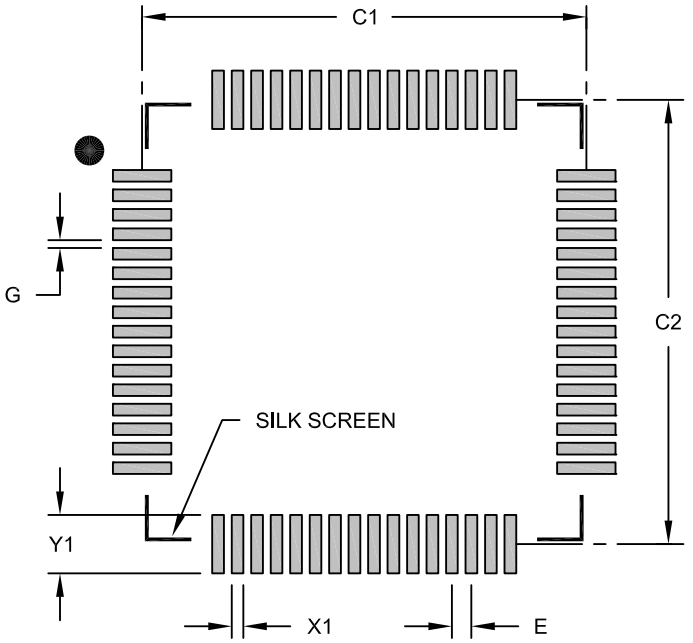
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-085-V2X Rev E Sheet 2 of 2

64-Lead Plastic Thin Quad Flatpack (V2X) 10x10x1 mm Body, 2.00 mm Footprint [TQFP]



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E	0.50 BSC		
Contact Pad Spacing	C1		11.40	
Contact Pad Spacing	C2		11.40	
Contact Pad Width (X64)	X1			0.30
Contact Pad Length (X64)	Y1			1.50
Distance Between Pads	G	0.20		

Notes:

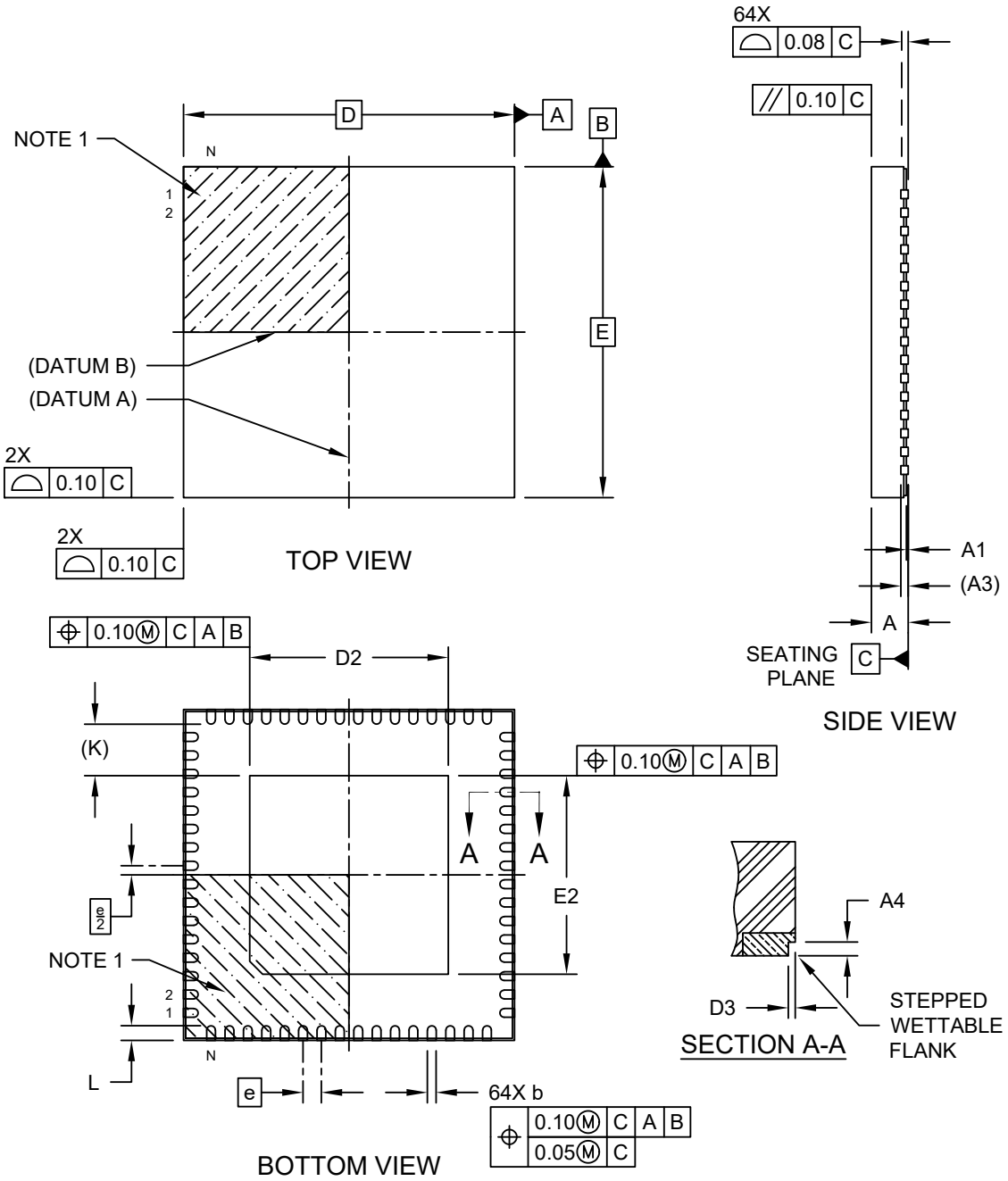
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Erawing C04-2085-V2X Rev E

64-Lead Very Thin Plastic Quad Flat, No Lead Package (5LX) - 9x9x1.0 mm Body [VQFN] With 5.4 mm Exposed Pad and Stepped Wettable Flanks

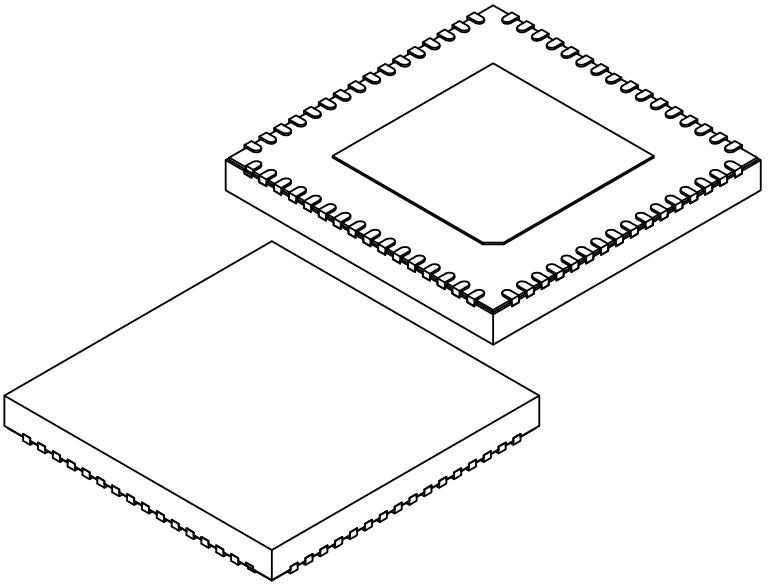
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-483-5LX Rev F Sheet 1 of 2

64-Lead Very Thin Plastic Quad Flat, No Lead Package (5LX) - 9x9x1.0 mm Body [VQFN] With 5.4 mm Exposed Pad and Stepped Wettable Flanks

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		64	
Pitch	e		0.50 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.203 REF	
Overall Length	D		9.00 BSC	
Exposed Pad Length	D2	5.30	5.40	5.50
Overall Width	E		9.00 BSC	
Exposed Pad Width	E2	5.30	5.40	5.50
Terminal Width	b	0.20	0.25	0.30
Terminal Length	L	0.30	0.40	0.50
Terminal-to-Exposed-Pad	K		1.40 REF	
Wettable Flank Step Length	D3	0.035	0.060	0.085
Wettable Flank Step Height	A4	0.10	-	0.19

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M

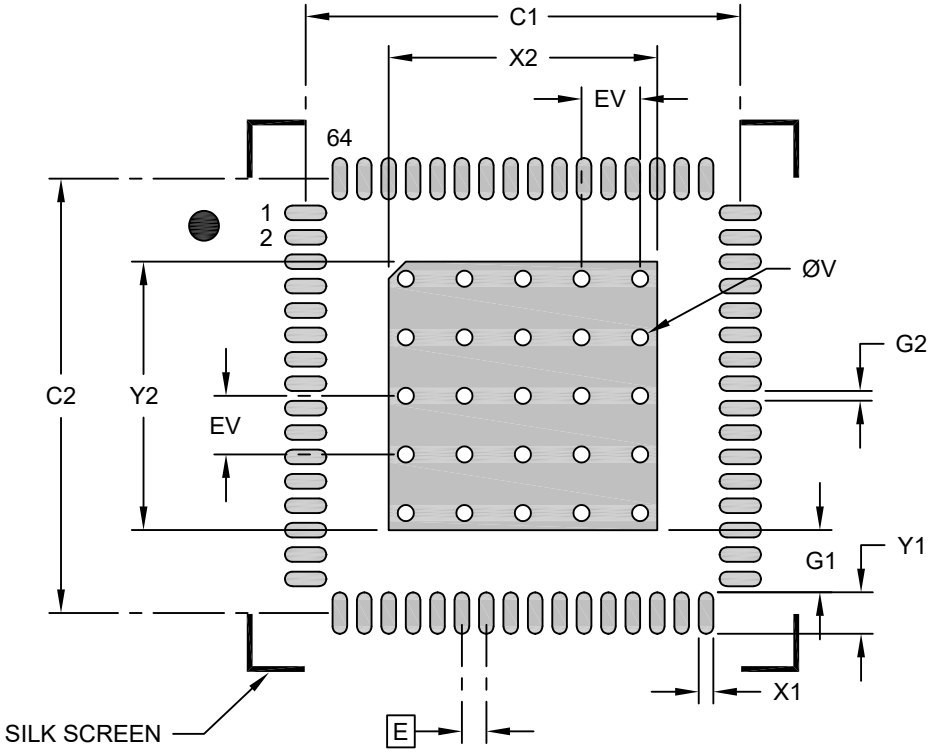
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-483-5LX Rev F Sheet 2 of 2

64-Lead Very Thin Plastic Quad Flat, No Lead Package (5LX) - 9x9x1.0 mm Body [VQFN] With 5.4 mm Exposed Pad and Stepped Wettable Flanks

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Optional Center Pad Width	X2			5.50
Optional Center Pad Length	Y2			5.50
Contact Pad Spacing	C1		8.90	
Contact Pad Spacing	C2		8.90	
Contact Pad Width (X64)	X1			0.30
Contact Pad Length (X64)	Y1			0.85
Contact Pad to Center Pad (X64)	G1	1.28		
Contact Pad to Contact Pad (X60)	G2	0.20		
Thermal Via Diameter	V		0.33	
Thermal Via Pitch	EV		1.20	

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2483-5LX Rev F

39. Revision History

Revision A (November 2023)

This is the initial version of the document.

Revision B (July 2024)

This revision incorporates the following updates:

- Sections:
  - Updated **Operating Conditions**, **High-Performance dsPIC33A DSP/CISC CPU**, **Controller Features**, **High-Speed PWM**, **Two High-Speed Analog-to-Digital Converters**, **Other Analog Features**, **Peripheral Features**, **dsPIC33AK128MC106 Product Family**, **1. Device Overview**, **2.1 Basic Connection**, **2.2 Decoupling Capacitors**, **2.4 ICSP Pins**, **3.1 Architectural Overview**, **3.3.15.1.2 Interrupting a `REPEAT` Loop**, **3.3.6 Exception Processing**, **3.3.16 Data Space Address Generation Units (AGUs)**, **3.3.7.1 SR: CPU STATUS Register**, **3.3.9 Alternate Working Register Arrays**, **3.3.10.1 Software Stack Examples**, **3.3.10.3 Stack Pointer Overflow**, **3.3.12.1 Data Accumulators**, **3.3.12.2 Multiplier**, **3.3.12.3 Data Accumulator Adder/Subtractor**, **3.3.12.3.3 Data Space Write Saturation**, **3.3.12.3.4 Accumulator Write Back**, **3.3.12.5 Barrel Shifter**, **3.3.12.7 DSP Engine Trap Events**, **3.3.13 Divide Support**, **3.3.17 MAC Instructions**, **3.3.19 Address Register Dependencies**, **3.3.12.2.1 DSP Multiply Instructions**, **3.3.12.3 Data Accumulator Adder/Subtractor**, **3.3.12.3.4 Accumulator Write Back**, **3.3.15 Loop Constructs**, **3.3.15.1 `REPEAT` Loop Construct**, **3.3.15.1.1 REPEAT Operation**, **3.3.16 Data Space Address Generation Units (AGUs)**, **3.4 Prefetch Buffer Unit (PBU)**, **3.4.3.2.1 Cache Invalidation when Writing to Flash**, **4.1.1 Address Space**, **4.2.3 Data Memory Organization**, **4.2.4 Bus Matrix**, **4.4.5 Bus Error**, **5.4.6.2 Performing Fault Injection**, **5.4.7.1 BIST at Start-up**, **6. Flash Program Memory**, **6.3.2.1.2 LOCK bit**, **User Page Erase Sequence**, **User Row Programming Sequence**, **6.3.3 Flash ECCUser Word Programming Sequence**, **7. Configuration Bits**, **7.1 Configuration Register Summary**, **8. Security Module**, **8.3.2 User Configuration A (UCA)**, **8.3.2 User Configuration B (UCB)**, **9.3.2 Power-On Reset (POR)**, **9.3.6 Brown-Out Reset (BOR)**, **9.3.7 On-Chip Voltage Regulators**, **10. Interrupt Controller**, **10.3.1 Remappable Interrupt Table**, **10.3.2 Collapsed Interrupt Vector**, **10.7.1 CPU Priority**, **10.9.1.1 Bus Error Traps**, **11. I/O Ports with Edge Detect**, **11.4.12.2.1 Clock Selection**, **11.4.13 Boundary Scan Cell Connections**, **11.7.3 I/O Integrity Module Operations in Sleep Mode**, **12. Oscillator and Clocking Module**, **12.1 Device-Specific Information**, **12.2 Architectural Overview**, **12.4.1 Clock Generators (CLKGEN)**, **12.4.1.1 Fail-Safe Clock Monitor (FSCM)**, **12.4.2 Phase-Locked Loop (PLL)**, **12.4.2.1 Input Clock Limitation at Start-up for PLL Mode**, **12.4.2.2 PLL Lock Status**, **12.4.2.3.1 Setup for Using PLL with the Primary Oscillator (POSC)**, **12.4.2.3.2 Setup for Using PLL with 8 MHz Internal FRC**, **12.4.3 Primary Oscillator (POSC)**, **12.4.3.2 Primary Oscillator Pin Functionality**, **12.4.4 Internal Fast RC (FRC) Oscillator**, **12.4.5 BFRC Oscillator**, **15.4.6 Internal Low-Power RC (LPRC) Oscillator**, **12.4.7 Clock Monitor**, **12.4.7.1 Clock Monitor Overview**, **12.4.7.2 Detection on Monitored Clock**, **Reference Clock Failure**, **12.4.8 Reference Clock Output (REFOx)**, **13. Direct Memory Access (DMA) Controller**, **13.4.12 Bit Manipulation**, **14. PWM**, **14.3.4.1.1 Master Clocking**, **14.3.4.2.1 PWM Generator Clocking**, **14.3.4.3.3 Shared Clocking**, **Frequency Scaling**, **15. High-Speed, 12-Bit Low Latency ADC**, **15.4.3.1 Sampling Time Requirements**, **15.4.4 Conversions**, **15.4.14.1.1 Error Compensation Coefficient Calculation**, **16. High-Speed Analog Comparator with Slope Compensation DAC**, **17. Quadrature Encoder Interface (QEI)**, **17.4.8 Interval Timer**, **17.6 Interrupts**, **18. Universal Asynchronous Receive Transmitter (UART)**, **18.4.1 Clocking and Baud Rate Configuration**, **18.4.1.1 Legacy Mode**, **18.4.1.2 Fractional Division Mode**, **18.4.1.4 Auto-Baud Feature**, **18.4.2.1 Asynchronous Mode**, **18.4.2.1.1 Asynchronous Transmit**, **Setup for UART Transmit**, **Half-Duplex Transmit**, **18.4.2.1.2 Asynchronous Receive**, **Setup for UART Receive**, **Receive Errors and Events**, **18.4.2.1.6 Address Detect**, **Address Detect Receive**, **XON/XOFF**, **Wait Time**,

Post-ATR Initialization, **18.6.1 Interrupt Watermarks**, **18.7.1 Sleep**, **20. Inter-Integrated Circuit (I^2C)**, **21.4.1 Transmit Mode**, **21.4.1.2 CRC Calculation**, **21.4.1.6 Asynchronous Transmitter Mode**, **21.4.1.7 Synchronous Transmitter Mode**, **21.4.2 Receive Mode**, **21.4.2.4 Receive Setup Procedure**, **21.4.2.6 Short PWM Code (SPC) Support**, **21.7.1 Sleep Mode**, **21.7.2 Idle Mode**, **23.4.5 Timer Prescalers**, **23.6.1 Timer Operation in Sleep Mode**, **26.5.1 Generating Phase-Shifted Waveforms**, **28.1 Device-Specific Information**, **28.2 Architectural Overview**, **28.4.1.1 Device Pin ESD Configuration**, **28.4.4 ADC Input Considerations**, **30. Watchdog Timer**, **30.1 Device-Specific Information**, **30.2 Architectural Overview**, **32.3.1.3 Idle Mode**, **32.3.1 Instruction-Based Power-Saving Modes**, **32.3.1.2.1 Sleep Mode Entry**, **32.3.1.2.3 Delay on Wake-up from Sleep**, **34. In-Circuit Debugger** and **Product Identification System**.

  - Added **4.1.2 Multiple Speed Peripheral Bus and Clock Overview**, **12.2.1 Peripheral Clock Divider**, **12.4.6 Internal Low-Power RC (LPRC) Oscillator**, **13.4.1 Data Transfer Options**, **13.4.7.1 Peripheral to Memory (Receive)**, **13.4.11 Ping-Pong**, **13.4.12.1.1 Channel Chaining15. High-Speed, 12-Bit Low Latency ADC**, **15.2 Architectural Overview**, **15.4.3 Sampling and Conversion Timing**, **15.4.6 Calibration**. **15.4.14.1 Software Gain Error Compensation**, **15.4.14.1.1 Error Compensation Coefficient Calculation**, **15.4.14.1.2 Application of Error Compensation Coefficient**, **I2S Audio Host Mode of Operation**, **15.2.2 Band Gap Reference**, **16.4.4. Calibration** and **38.1 Package Marking Information**.
  - Removed **One Instruction Word, One Instruction Cycle**, **One Instruction Word, Two Instruction Cycles**, **One Instruction Word, Two or Four Instruction Cycles (Program Flow Changes)**, **Two Instruction Words, Four Instruction Cycles – `GOTO` or `CALL`, Address Register Dependencies**. **X Address Generation Unit**, **Y Address Generation Unit**, **Read After Write Dependency Rules**, **Instruction Stall Cycles, Instruction Stall Cycles and Interrupts**, **Instruction Stall Cycles and Flow Change Instructions**, **Instruction Stalls and `REPEAT` Loops**, **Data Space Arbiter Stalls**, **Coprocessor Interface**, **Coprocessor Instructions**, **Coprocessor Data Interface**, **Coprocessor Register Context**, **CPU Accessing Coprocessor Status**,

- Tables:
  - Updated **Table 1. dsPIC33AK128MC106 Family**, **Table 2. 28-Pin SSOP Complete Pin Function Descriptions**, **Table 3. 28-Pin VQFN Complete Pin Function Descriptions**, **Table 4. 36-Pin VQFN Complete Pin Function Descriptions**, **Table 5. 48-Pin VQFN, TQFP Complete Pin Function Descriptions**, **Table 6. 64-Pin VQFN, TQFP Complete Pin Function Descriptions**, **Table 7. Pinout I/O Descriptions**, **Table 3-2. Programmer's Model Register Descriptions**, **Table 3-4. DSP Instructions that Use the Multiplier**, **Table 3-3. dsPIC33A Data Ranges**, **Table 4-1. Peripheral Bus Speed to Peripheral Mapping**, **Table 4-4. Target to Bus Error Index Mapping**, **4.4.5.3 Target Bus Error Handling**, **Table 7-2. dsPIC33AK128MC106 Configuration Addresses**, **Table 10-1. Interrupt Vector Details**, **Table 11-9. PPS Availability by Package**, **Table 11-13. Output Selection for Remappable Pins (RPn)**, **Table 11-10. Selectable Input Sources (Maps Input to Function)**, **Table 12-1. Oscillator Summary**, **Table 12-2. Clock Generator Input and Destination**, **Table 12-3. Clock Generator Input Clock Selections**, **Table 12-4. PLL Input Clock Selections**, **Table 12-5. Clock Monitor CNTSEL/WINSEL Input Clock Selections**, **Table 12-7. Primary Oscillator Clock Source Options**, **Table 12-8 Clock Pin Function Selection**, **Table 13-1. DMA Summary Table**, **Table 13-2. DMA Channel Trigger Sources**, **Table 14-1. PWM Summary Table**, **Table 15-1. ADC Summary**, **Table 15-2. ADC External Input Availability**, **Table 15-3. TRG1SRC Trigger Source Selection Bits**, **Table 15-4. TRG2SRC Trigger Source Selection Bits**, **Table 15-5 FPDMDAC Calibration Register**, **Table 16-1. DAC Summary**, **Table 17-1. QEI Summary**, **Table 18-1. UART Summary**, **Table 18-2. Baud Clock Source Selection bits**, **Table 19-1. SPI Summary Table**, **Table 19-2. MCLKEN Host Clock Enable bit**, **Table 19-3. Device FIFO Depth**, **Table 20-1. I^2C Summary Table**, **Table 21-1. SENT Summary Table**, **Table 22-1. BiSS Summary Table**, **Table 22-2. CLKSEL Macro Baud Clock Selection bit**, **Table 22-1. BiSS Summary TableTable 23-2. Timer1**

Clock Source Select bit, **Table 24-2. CLKSEL Time Base Clock Select bits**, **Table 26-1. PTG Summary Table**, **Table 28-1. CBG Summary Table**, **Table 29-1. Op Amp Summary Table**, **Table 29-2. Op Amp Availability by Device Package**, **Table 30-2. WDT Period Configurations**, **Table 31-1. DMT Summary**, **Table 32-1. Oscillator Delay**, **Table 35-1. Symbols Used in Opcode Descriptions**, **Table 35-2. Instruction Set Overview**, **Table 37-1. Absolute Maximum Ratings**, **Table 37-4. Thermal Packaging Characteristics**, **Table 37-5. Operating Voltage Specifications**, **Table 37-6. DC Characteristics: Operating Current (I_{DD})**, **Table 37-7. Idle Current (I_{IDLE})**, **Table 37-8. DC Characteristics: Power-Down Current (IPD)**, **Table 37-9. DC Characteristics: Watchdog Timer Delta Current (ΔIWDT)**, **Table 37-10. DC Characteristics: PWM Delta Current**, **Table 37-11. DC Characteristics: CLKGEN Delta Current**, **Table 37-12. DC Characteristics: PLL Delta Current**, **Table 37-13. DC Characteristics: ADC Δ Current**, **Table 37-14. DC Characteristics: Comparator + DAC Delta Current**, **Table 37-15. Op Amp Delta Current**, **Table 37-16. I/O Pin Input Specifications**, **Table 37-17. I/O Pin Input Leakage Specifications**, **Table 37-19. I/O Pin Output Specifications**, **Table 37-20. Program Memory Specifications**, **Table 37-21. Capacitive Loading Requirements on Output Pins**, **Table 37-22. External Clock Timing Requirements**, **Table 37-23. PLLn Timing Specifications**, **Table 37-24. Peripheral Input Clock Timing Specifications**, **Table 37-25. Internal FRC Accuracy**, **Table 37-26. I/O Timing Requirements**, **Table 37-27. Reset, Watchdog Timer Timing Requirements**, **Table 37-28. High-Speed PWMx Module Timing Requirements**, **Table 37-29. QEI Index Pulse Timing Requirements**, **Table 37-30. SPIx Maximum Data/Clock Rate Summary**, **Table 37-31. SPIx Host Mode (Half-Duplex, Transmit Only) Timing Requirements**, **Table 37-32. SPIx Host Mode (Full-Duplex, CKE = 1, CKP = X, SMP = 1)**, **Table 37-33. SPIx Host Mode (Full-Duplex, CKE = 0, CKP = x, SMP = 1) Timing Requirements**, **Table 37-34. SPIx Client Mode (Full-Duplex, CKE = 0, CKP = x, SMP = 0) Timing Requirements**, **Table 37-35. SPIx Client Mode (Full-Duplex, CKE = 1, CKP = x, SMP = 0) Timing Requirements**, **Table 37-36. I2Cx Bus Data Timing Requirements (Host Mode)**, **Table 37-38. UARTx Module I/O Timing Requirements**, **Table 37-39. ADC Module Specifications**, **Table 37-40. Die Temperature Diode Specifications**, **Table 37-41. High-Speed Analog Comparator Module Specifications**, **Table 37-42. DACx Module Specifications**, **Table 37-43. DACx Output (DACOUTx Pins) Specifications**, **Table 37-44. Current Bias Generator Specifications** and **Table 37-45. Operational Amplifier Specifications**.

  - Added **Table 7-3. dsPIC33AK128MC106 Backup Configuration Addresses**, **Table 7-4. Device ID and Revision Addresses**, **Table 7-5. Family Device Identifier**, **Table 8.3.5. Security Configuration Words**, **Table 14-2. MCLKSEL PWM Master Clock Selection**, **Table 15-4. TRG2SRC Trigger Source Selection Bits**, **Table 24-1. CCP Summary Table**, **Table 28-1. CBG Summary Table** and **Table 30-1. WDT Summary Table**.
  - Removed **Device Reset to Code Execution Start Time**, **CLKSEL DAC Clock Source Select bits**, **Doze Current (IDOZE)** and **Electrical Characteristics: BOR**.

- Registers:
  - Updated **3.2.9 Y AGU Modulo Addressing Start Address Register**, **3.2.17 CPU STATUS Register**, **4.3.4 BMX Error Status Register for Initiator**, **4.3.5 BMX Error Status Register for Initiator**, **4.3.6 BMX Error Status Register for Initiator**, **4.3.7 BMX Error Status Register for CPU**, **4.3.8 BMX Error Status Register for Initiator**, **4.3.9 BMX Error Status Register for Initiator x**, **5.3.3 BMX ECC RAM Fault Pointer Register**, **6.2.1 Nonvolatile Memory (NVM) Control Register**, **6.2.2 NVM Address Register**, **6.2.3 NVM Write Data 0 Register**, **6.2.4 NVM Write Data 1 Register**, **6.2.5 NVM Write Data 2 Register**, **6.2.6 NVM Write Data 3 Register**, **6.2.7 NVM Source Data Address Register**, **6.2.8 NVM ECC Control Register**, **6.2.9 NVM ECC Status Register**, **6.2.10 NVM ECC Fault Injection Pointer Register**, **6.2.11 NVM ECC Fault Injection Address Register**, **6.2.12 NVM ECC Error Address Register**, **6.2.13 NVM ECC Error Data 0 Register**, **6.2.14 NVM ECC Error Data 1 Register**, **6.2.15 NVM ECC Error Data 2 Register**, **6.2.16 NVM ECC Error Data 3 Register**, **6.2.17 NVM ECC Value Register**, **6.2.18 NVM ECC Syndrome Register**, **6.2.19 NVM CRC**

Control Register, **6.2.20 NVM CRC Start Address Register**, **6.2.21 NVM CRC End Address Register**, **6.2.22 NVM CRC Seed Register**, **6.2.23 NVM CRC Output Data Register**, **7.1.3 FDEVOPT Configuration Register**, **7.1.6 FPRxST Configuration Register**, **7.1.7 FPRxEND Configuration Register**, **7.1.8 FIRT Configuration Register**, **7.1.10 FPED Configuration Register**, **7.1.11 FEPUCB Configuration Register**, **7.1.12 FWPUCB Configuration Register**, **7.2.2 Device ID Register**, **9.2.1 Reset Control Register**, **10.4.3 Interrupt Control Register 3**, **10.4.5 Interrupt Control Register 5**, **10.4.6 Interrupt Control and Status Register**, **10.4.7 Interrupt Vector Base Address Register**, **10.4.9 Interrupt Request Flags Register 0**, **10.4.10 Interrupt Request Flags Register 1**, **10.4.11 Interrupt Request Flags Register 2**, **10.4.13 Interrupt Request Flags Register 4**, **10.4.14 Interrupt Request Flags Register 5**, **10.4.15 Interrupt Request Flags Register 6**, **10.4.16 Interrupt Request Flags Register 7**, **10.4.18 Interrupt Enable Register 0**, **10.4.19 Interrupt Enable Register 1**, **10.4.20 Interrupt Enable Register 2**, **10.4.22 Interrupt Enable Register 4**, **10.4.23 Interrupt Enable Register 5**, **10.4.24 Interrupt Enable Register 6**, **10.4.25 Interrupt Enable Register 7**, **10.4.26 Interrupt Enable Register 8**, **10.4.27 Interrupt Priority Register 0**, **10.4.28 Interrupt Priority Register 1**, **10.4.29 Interrupt Priority Register 2**, **10.4.30 Interrupt Priority Register 3**, **10.4.33 Interrupt Priority Register 6**, **10.4.35 Interrupt Priority Register 8**, **10.4.37 Interrupt Priority Register 10**, **10.4.38 Interrupt Priority Register 11**, **10.4.39 Interrupt Priority Register 12**, **10.4.43 Interrupt Priority Register 16**, **10.4.45 Interrupt Priority Register 18**, **10.4.47 Interrupt Priority Register 20**, **11.3.24 Peripheral Pin Select Input Register 14**, **12.3.13 User Clock Diagnostics Control Register**, **12.3.2 Oscillator Configuration Register**, **12.3.5 Clock Generator Control Register**, **12.3.10 PLL Divider Register**, **12.3.12 Reset Control Register**, **12.3.14 Clock Monitor Control Register**, **12.3.15 Clock Monitor Control Register**, **12.3.16 Clock Monitor Prescaler Register**, **13.3.1 DMA Module Control Register**, **13.3.2 DMA Data Buffer Register**, **13.3.5 DMA Channel x Control Register**, **13.3.10 DMA Channel x Count Register**, **14.3.2.1 PWM Clock Control Register**, **14.3.3.1 PWM Generator x Control Register**, **14.3.3.13 PWM Generator x Period Register**, **14.3.3.16 PWM Generator x Trigger C Register**, **15.3.1 ADC n Control Register**, **15.3.2 ADC n Test Mode Data Register**, **15.3.3 ADC n Result Ready Flags Register**, **15.3.4 ADC n Comparators Status Register**, **15.3.5 ADC n Software Triggers Request Register**, **15.3.6 ADC 1 Channel x Control Register**, **15.3.7 ADC 1 Channel x Result Register**, **15.3.8 ADC 1 Channel x Counter Register**, **15.3.9 ADC 1 Channel x Low Compare Register**, **15.3.10 ADC 1 Channel x High Compare Register**, **15.3.11 ADC 1 Channel x Secondary Accumulator Register**, **16.3.1 DAC Control 1 Register**, **16.3.2 DAC Control 2 Register**, **17.3.3 QEI Status Register**, **18.3.1 UARTx Configuration Register**, **18.3.2 UARTx Status Register**, **18.3.6 UARTx Timing Parameter A Register**, **18.3.10 UARTx Interrupt Register**, **21.3.1 SENTx Control Register 1**, **21.3.4 SENTx Status Register**, **21.3.6 SENTx Data Register**, **22.3.1 BiSS Single Cycle Data Register**, **22.3.2 BiSS Single Cycle Data Register**, **22.3.3 BiSS 1 Register Data Register**, **22.3.4 BiSS Client Configuration Register**, **27.2.21 CRC Control Register**, **26.3.1 PTG Control Register**, **27.2.1 CRC Control Register**, **27.2.2 CRC XOR Register**, **27.2.3 CRC Data Register**, **27.2.4 CRC Shift Register**, **32.2.1 Reset Control Register**, **32.2.3 Peripheral Module Disable 2 Register**, **32.2.4 Peripheral Module Disable 3 Register** and **32.2.3 Peripheral Module Disable 2 Register**.

  - Added **7.1.9 FSECDBG Configuration Register**, **15.3.12 ADC 2 Channel x Control Register**, **15.3.13 ADC 2 Channel x Result Register**, **15.3.14 ADC 2 Channel x Counter Register**, **15.3.15 ADC 2 Channel x Low Compare Register**, **15.3.16 ADC 2 Channel x High Compare Register** and **15.3.17 ADC 2 Channel x Secondary Accumulator Register**.
  - Removed **NVM Address Register (Duplicate)** and **NVM ECC Fault Injection Address Register (Duplicate)**.

- Figures
  - Updated **Figure 1. 28-Pin SSOP**, **Figure 2. 28-Pin VQFN**, **Figure 3. 36-Pin VQFN**, **Figure 4. 48-Pin VQFN, TQFP**, **Figure 5. 64-Pin VQFN, TQFP**, **Figure 1-1. dsPIC33AK128MC106 Family Block Diagram**, **Figure 2-1. Recommended Minimum Connection**, **2.3 Master**

Clear (MCLR) Pin, **2.4 ICSP Pins**, **Figure 3-12. Conventional and Convergent Rounding ModesFigure 3-14. Modulo Addressing Operation Example**, **Figure 4-1. Memory Map for dsPIC33AK128MC106**, **Figure 5-3. BIST Flowchart**, **Figure 8-2. PAC Locking Behavior**, **Figure 9-1. System Reset Block Diagram**, **Figure 12-1. Oscillator Module Block Diagram**, **Figure 12-2. Clock Generator**, **Figure 12-4. PLL Block Diagram**, **Figure 12-9. Clock Monitor Architecture**, **Figure 12-13. Frequency Measurement Function – CPU**, **Figure 12-5. Crystal or Ceramic Resonator Operation (XT or HS Oscillator Mode)**, **Figure 12-8. OSCO Pin for Clock Output (in EC Mode)**, **Figure 12-10. Monitor Function**, **Figure 13-6. Ping-Pong Support Connection**, **Figure 13-7. Ping-Pong – Transmit**, **Figure 13-8. Ping-Pong Transmit in Steady State**, **Figure 13-9. Ping-Pong – Receive**, **Figure 13-10. Ping-Pong Receive Operation in Steady State**, **Figure 14-4. PWM Generator Clocking**, **Figure 15-2. ADC Input Model**, **Figure 15-1. ADC Module Block Diagram**, **Figure 16-1. High-Speed Analog Comparator Module Block Diagram**, **Figure 17-2. Quadrature Encoder Interface (QEI) Module Block Diagram**, **Figure 19-1. SPIx Module Block Diagram**, **Figure 18-1. Simplified UARTx Block Diagram**, **Figure 18-8. Asynchronous Transmitter Block Diagram**, **Figure 18-9. Asynchronous Receiver Block Diagram**, **Figure 18-15. Smart Card Subsystem**, **Figure 22-7. Clock Tree**, **Figure 23-1. Timer1 Block Diagram**, **Figure 23-2. Timer1 Clock Input Logic**, **Figure 21-1. SENTx Module Block Diagram**, **Figure 21-3. SENTx Transmit Mode Block Diagram**, **Figure 21-4. SENTx Data Transmission, Asynchronous Mode**, **Figure 21-6. SENTx Data Transmission, Synchronous Mode**, **Figure 21-7. SENTx Receive Mode Block Diagram**, **Figure 30-1. Watchdog Timer Block Diagram**, **Figure 28-1. Current Bias Generator Sources**, **Figure 28-2. 10 µA Current Source**, **Figure 28-4. Single-Ended Positive Voltage Shift**, **Figure 32-1. Wake-up Delay from Sleep Mode**, **Figure 37-1. Load Conditions for Device Timing Specifications**, **Figure 37-2. External Clock Timing**, **Figure 37-7. QEI Module Index Pulse Timing Characteristics**, **Figure 37-11. SPIx Host Mode (Full-Duplex, CKE = 0, CKP = x, SMP = 1) Timing Characteristics**, **Figure 37-12. SPIx Client Mode (Full-Duplex, CKE = 0, CKP = x, SMP = 0) Timing Characteristics** and **Figure 37-13. SPIx Client Mode (Full-Duplex, CKE = 1, CKP = x, SMP = 0) Timing Characteristics**, **Table 37-17. I/O Pin Input Leakage Specifications**, **Table 37-27. Reset and Watchdog Timer Timing Requirements**, **Table 37-32. SPIx Host Mode (Full-Duplex, CKE = 1, CKP = X, SMP = 1)**, **Table 37-39. ADC Module Specifications**and **Table 37-42. DACx Module Specifications**.

  - Added **Figure 12-3. Peripheral Clock Divider Block Diagram**
  - Removed **REPEAT [W] Instruction Flow** and **Connecting DAC Output to the Op Amp Internally**.

- Examples:
  - Updated **Example 3-2. File Register Instructions**, **Example 12-4. Code Example for Using PLL with 8 MHz Internal FRC**, **Example 13-1. Typical Code Sequence for a One-Shot Transfer**, **Example 13-2. Typical Code Sequence for a Repeated One-Shot Transfer**, **Example 13-3. Typical Code Sequence for a Continuous Transfer**, **Example 14-3. Typical Code Sequence for a Repeated Continuous Transfer**, **Example 13-5. Code for Fixed to Block Continuous Transfer (Peripheral to Memory)**, **Example 14-1. Six-Step PWM Scheme 1 Code**, **Example 14-2. Six-Step PWM Scheme 2 Code**, **Example 14-3. Six-Step PWM Scheme 3 Code**, **Example 14-4. Three-Phase Sinusoidal PMSM/ACIM Motor Control Code**, **Example 14-5. Complementary PWM Output Mode**, E**xample 14-6. Cycle-by-Cycle Current Limit Mode**, **Example 14-7. External Period Reset Mode**, **Example 18-1. UART1 Transmission with Interrupts**, **Example 18-2. UART1 Reception with Interrupts**, **Example 18-3. Address Detect Transmission**, **Example 18-4. Address Detect Reception**, **Example 21-2. SENT1 Asynchronous Transmission Code**, **Example 21-3. SENT1 Synchronous Transmission Code**, **Example 21-4. SENT1 Reception Code**, **Example 21-5. Short PWM Code (SPC) Support**, **Example 21-6. SENT Reception (SPC Pulse Transmission)**, **Example 23-2. Synchronous External Counter Example Code** and **Example 23-4. 32-bit Asynchronous Counter Mode Code**, **Example 21-2. SENT1**

Asynchronous Transmission Code, **Example 21-3. SENT1 Synchronous Transmission Code**, **Example 21-4. SENT1 Reception Code**, **Example 21-5. SENT Transmission (SPC Pulse Reception)** and **Example 21-6. SENT Reception (SPC Pulse Transmission)**.

- Equations:
  - Updated **Equation 16-2. Determining the SLPxDAT Value**, **Equation 14-1. Leading-Edge Blanking Period**, **Equation 15-2. One Reference Voltage Gain Error Compensation Coefficient Calculation** and **Equation 15-3. Two Reference Voltages Gain Error Compensation Coefficient and Offset Error Calculation**.

Minor grammatical corrections and formatting changes throughout the document.

Microchip Information

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

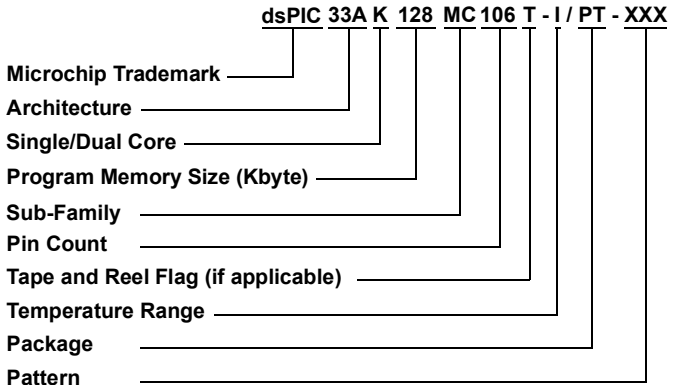
- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.



Device:	dsPIC33AK32MC102, dsPIC33AK32MC103, dsPIC33AK32MC105, dsPIC33AK32MC106, dsPIC33AK64MC102, dsPIC33AK64MC103, dsPIC33AK64MC105, dsPIC33AK64MC106, dsPIC33AK128MC102, dsPIC33AK128MC103, dsPIC33AK128MC105, dsPIC33AK128MC106	
Architecture:	33A	= 32-bit Digital Signal Controller (DSC)
Single/Dual Core:	K	= Single-Core Device
Sub-Family:	MC	= Motor Control and General Purpose Real-Time Control
Pin Count:	102	= 28
	103	= 36
	105	= 48
	106	= 64
Tape & Reel Option:	Blank	= Tube or Tray
	T	= Tape & Reel
Temperature Range:	I	= -40°C to +85°C (Industrial)
	E	= -40°C to +125°C (Extended)
	H	= -40°C to +150°C (High Temp)
Package:	28 SS	= SSOP (N2X)
	28 M7	= vQFN (3LW)
	36 M7	= vQFN (4AW)
	48 M7	= vQFN (6MX)
	48 PT	= TQFP (Y8X)
	64 M7	= vQFN (5LX)
	64 PT	= TQFP (V2X)

Examples:

- dsPIC33AK128MC106-I/PT: dsPIC33, 32-bit Core, 128-Kbyte Program Memory, 64-Pin, Industrial temperature, TQFP package.

Notes:

1. Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.
2. Small form-factor packaging options may be available. Please check www.microchip.com/packaging for small-form factor package availability, or contact your local Sales Office.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP’S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, TimeCesium, TimeHub, TimePictra, TimeProvider, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

ISBN: 978-1-6683-4942-7

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com	**Australia - Sydney** Tel: 61-2-9868-6733	**India - Bangalore** Tel: 91-80-3090-4444	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393
Atlanta Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455	**China - Beijing** Tel: 86-10-8569-7000	**India - New Delhi** Tel: 91-11-4160-8631	**Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829
Austin, TX Tel: 512-257-3370	**China - Chengdu** Tel: 86-28-8665-5511	**India - Pune** Tel: 91-20-4121-0141	**Finland - Espoo** Tel: 358-9-4520-820
Boston Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088	**China - Chongqing** Tel: 86-23-8980-9588	**Japan - Osaka** Tel: 81-6-6152-7160	**France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79
Chicago Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075	**China - Dongguan** Tel: 86-769-8702-9880	**Japan - Tokyo** Tel: 81-3-6880- 3770	**Germany - Garching** Tel: 49-8931-9700
Dallas Addison, TX Tel: 972-818-7423 Fax: 972-818-2924	**China - Guangzhou** Tel: 86-20-8755-8029	**Korea - Daegu** Tel: 82-53-744-4301	**Germany - Haan** Tel: 49-2129-3766400
Detroit Novi, MI Tel: 248-848-4000	**China - Hangzhou** Tel: 86-571-8792-8115	**Korea - Seoul** Tel: 82-2-554-7200	**Germany - Heilbronn** Tel: 49-7131-72400
Houston, TX Tel: 281-894-5983	**China - Hong Kong SAR** Tel: 852-2943-5100	**Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906	**Germany - Karlsruhe** Tel: 49-721-625370
Indianapolis Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380	**China - Nanjing** Tel: 86-25-8473-2460	**Malaysia - Penang** Tel: 60-4-227-8870	**Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44
Los Angeles Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800	**China - Qingdao** Tel: 86-532-8502-7355	**Philippines - Manila** Tel: 63-2-634-9065	**Germany - Rosenheim** Tel: 49-8031-354-560
Raleigh, NC Tel: 919-844-7510	**China - Shanghai** Tel: 86-21-3326-8000	**Singapore** Tel: 65-6334-8870	**Israel - Hod Hasharon** Tel: 972-9-775-5100
New York, NY Tel: 631-435-6000	**China - Shenyang** Tel: 86-24-2334-2829	**Taiwan - Hsin Chu** Tel: 886-3-577-8366	**Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781
San Jose, CA Tel: 408-735-9110 Tel: 408-436-4270	**China - Shenzhen** Tel: 86-755-8864-2200	**Taiwan - Kaohsiung** Tel: 886-7-213-7830	**Italy - Padova** Tel: 39-049-7625286
Canada - Toronto Tel: 905-695-1980 Fax: 905-695-2078	**China - Suzhou** Tel: 86-186-6233-1526	**Taiwan - Taipei** Tel: 886-2-2508-8600	**Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340
	China - Wuhan Tel: 86-27-5980-5300	**Thailand - Bangkok** Tel: 66-2-694-1351	**Norway - Trondheim** Tel: 47-72884388
	China - Xian Tel: 86-29-8833-7252	**Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Poland - Warsaw** Tel: 48-22-3325737
	China - Xiamen Tel: 86-592-2388138		**Romania - Bucharest** Tel: 40-21-407-87-50
	China - Zhuhai Tel: 86-756-3210040		**Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91
			Sweden - Gothenberg Tel: 46-31-704-60-40
			Sweden - Stockholm Tel: 46-8-5090-4654
			UK - Wokingham Tel: 44-118-921-5800 Fax: 44-118-921-5820