PIC32MX Family
Data Sheet

64/100-Pin General Purpose,
32-Bit Flash Microcontrollers

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

PIC32MX FAMILY

64/100-Pin General Purpose, 32-Bit Flash Microcontrollers

High-Performance RISC CPU:

- MIPS32® M4K™ 32-Bit Core with 5-Stage Pipeline
- Single-Cycle Multiply and High-Performance Divide Unit
- MIPS16e™ Mode for Up to 40% Smaller Code Size
- User and Kernel Modes to Enable Robust Embedded System
- Two 32-Bit Core Register Files to Reduce Interrupt Latency
- Prefetch Cache Module to Speed Execution from Flash

Special Microcontroller Features:

- Operating Voltage Range of 2.5V to 3.6V
- 32-512K Flash and 8-32K Data Memory
- Additional 12 KB of Boot Flash Memory
- Pin-Compatible with most PIC24/dsPIC® Devices
- Multiple Power Management Modes
- Multiple Interrupt Vectors with Individually Programmable Priority
- Fail-Safe Clock Monitor Mode
- Configurable Watchdog Timer with On-Chip, Low-Power RC Oscillator for Reliable Operation
- Two Programming and Debugging Interfaces:
  - 2-wire interface with unintrusive access and real-time data exchange with application
  - 4-wire MIPS standard enhanced JTAG interface
- Unintrusive Hardware-Based Instruction Trace
- IEEE Std 1149.2 Compatible (JTAG) Boundary Scan

Analog Features:

- Up to 16-Channel 10-Bit Analog-to-Digital Converter:
  - 400 ksps conversion rate
  - Conversion available during Sleep, Idle
- Two Analog Comparators

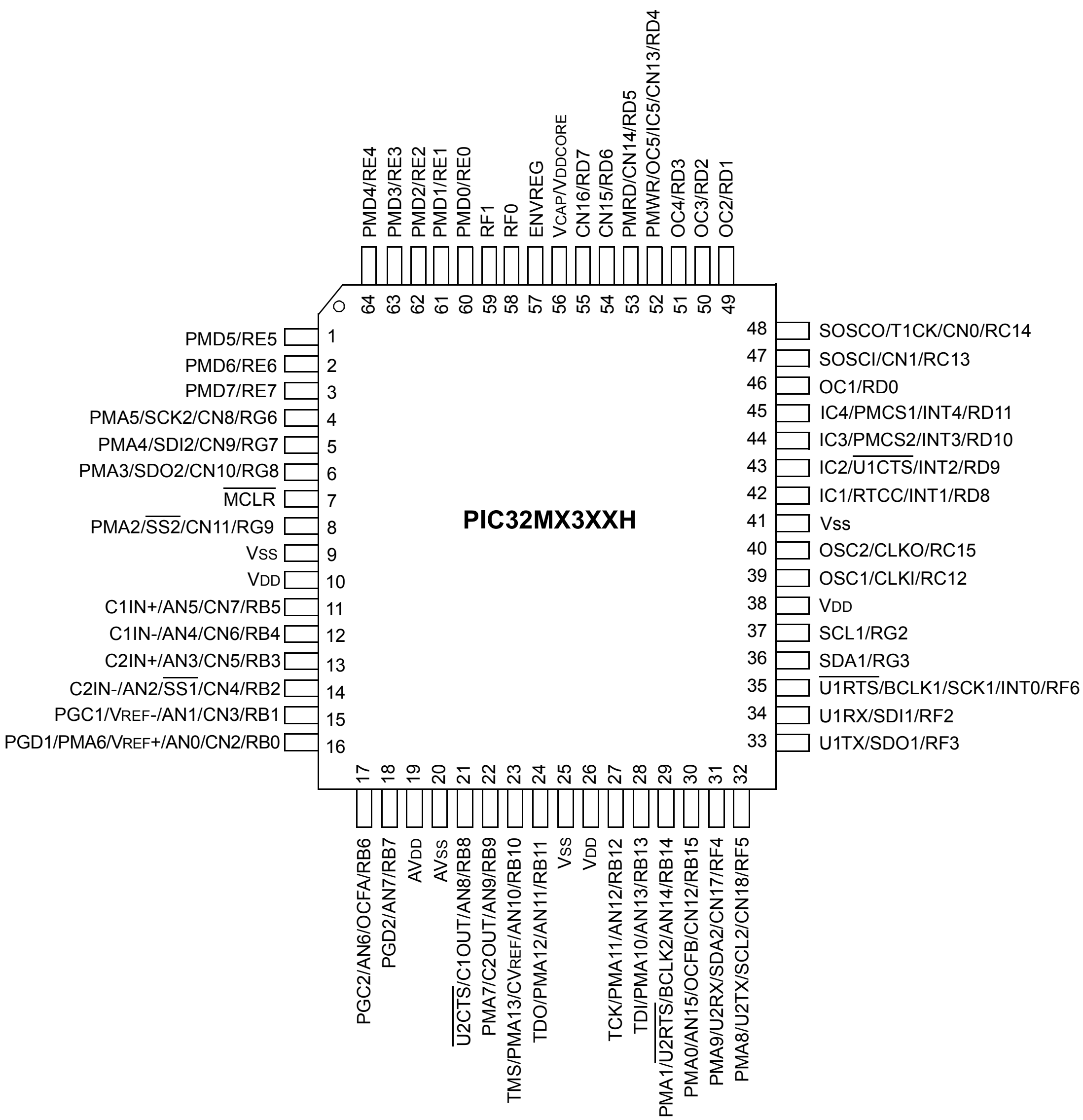
Peripheral Features:

- Atomic SET, CLEAR and INVERT Operation on Select Peripheral Registers
- Up to 4-Channel Hardware DMA Controller with Automatic Data Size Detection
- Two I^2C™ Modules
- Two UART Modules with:
  - RS-232, RS-485 and LIN 1.2 support
  - IrDA® with on-chip hardware encoder and decoder
- Parallel Master and Slave Port (PMP/PSP) with 8-Bit and 16-Bit Data and Up to 16 Address Lines
- Hardware Real-Time Clock/Calendar (RTCC)
- Five 16-Bit Timers/Counters (two 16-bit pairs combine to create two 32-bit timers)
- Five Capture Inputs
- Five Compare/PWM Outputs
- Five External Interrupt pins
- 5V Tolerant Input Pins
- 8 mA Sink/Source on Select I/O Pins
- Configurable Open-Drain Output on Digital I/O Pins

General Purpose												
Device	Pins	Program/ Data Memory (KB)	Timers/ Capture/ Compare	DMA Channels	VREG	Prefetch Cache	Trace	EUART/ SPI/ I^2C™	10-Bit A/D (ch)	Comparators	PMP/PSP	JTAG
PIC32MX300F032H	64	32/8	5/5/5	0	Yes	No	No	2/2/2	16	2	Yes	Yes
PIC32MX320F064H	64	64/16	5/5/5	0	Yes	Yes	No	2/2/2	16	2	Yes	Yes
PIC32MX320F128H	64	128/16	5/5/5	0	Yes	Yes	No	2/2/2	16	2	Yes	Yes
PIC32MX340F256H	64	256/32	5/5/5	4	Yes	Yes	No	2/2/2	16	2	Yes	Yes
PIC32MX320F128L	100	128/16	5/5/5	0	Yes	Yes	No	2/2/2	16	2	Yes	Yes
PIC32MX360F256L	100	256/32	5/5/5	4	Yes	Yes	Yes	2/2/2	16	2	Yes	Yes
PIC32MX360F512L	100	512/32	5/5/5	4	Yes	Yes	Yes	2/2/2	16	2	Yes	Yes

Pin Diagram (64-Pin General Purpose)

64-Pin TQFP (General Purpose)



Pin Diagram (100-Pin General Purpose)

100-Pin TQFP (General Purpose)

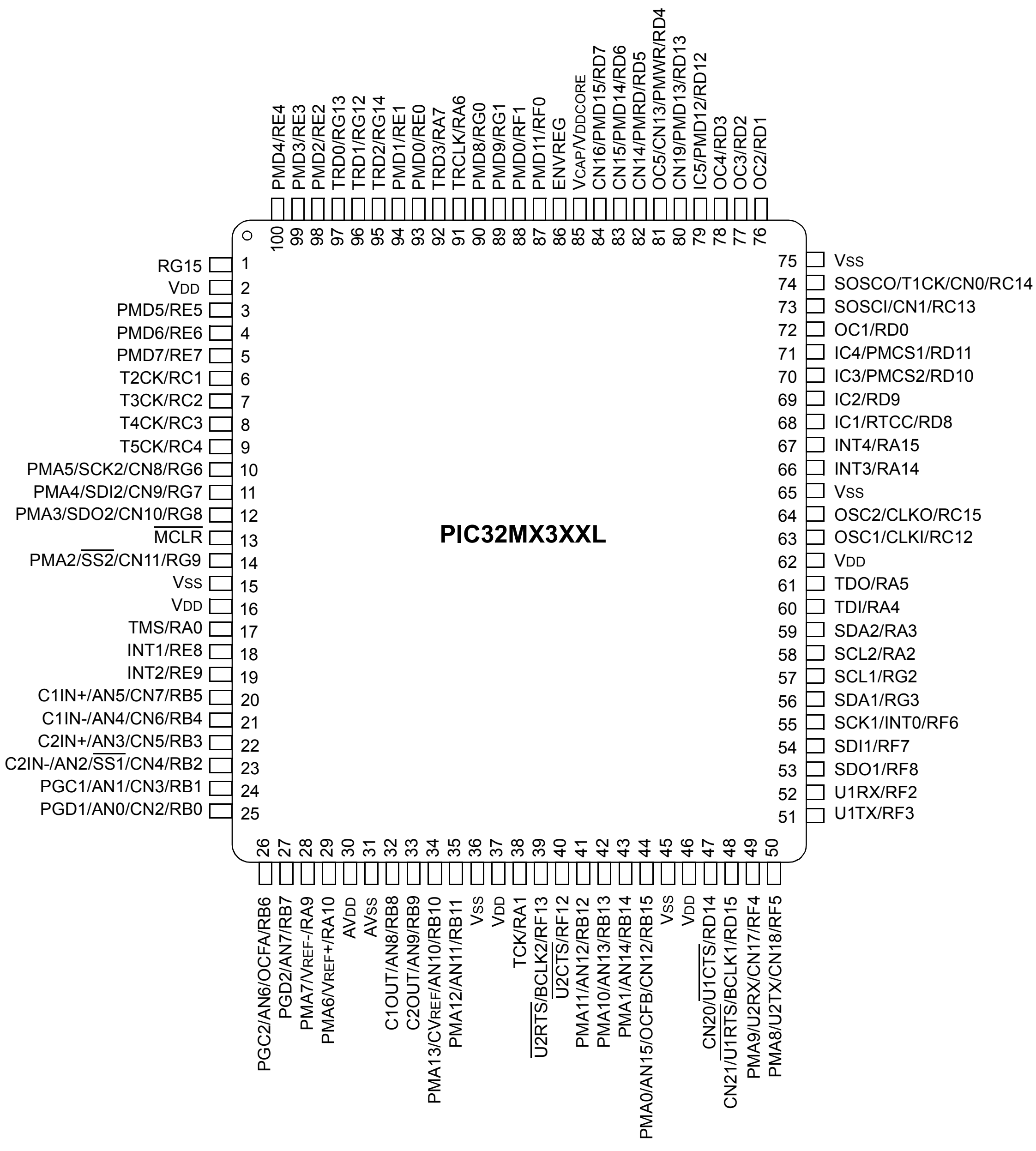


Table of Contents

 Advance Information

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** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. 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 Web site at:

http://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., DS30000A is version A of document DS30000).

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 Web site; http://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 web site at **www.microchip.com** to receive the most current information on all of our products.

NOTES:

PIC32MX FAMILY

64/100-Pin General Purpose, 32-Bit Flash Microcontrollers

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC32MX300F032H
- PIC32MX320F064H
- PIC32MX320F128H
- PIC32MX320F128L
- PIC32MX340F256H
- PIC32MX360F256L
- PIC32MX360F512L

This family introduces a new line of Microchip devices: a 32-bit RISC microcontroller family with a broad peripheral feature set and enhanced computational performance. The PIC32MX Family offers a new migration option for those high-performance applications which may be outgrowing their 16-bit platforms.

1.1 Easy Migration

The PIC32MX Family was designed to provide an easy migration path as the application needs change.

The consistent pinout scheme used throughout the entire family aids in migrating to the next larger device. This is true when moving between devices with the same pin count, or even jumping from 64-pin to 100-pin devices.

The PIC32MX Family is pin compatible with Microchip PIC24FJ128GA010 devices.

1.2 Core Features

1.2.1 32-BIT RISC ARCHITECTURE

Central to all PIC32MX Family devices is the 32-bit MIPS32 M4K CPU core, offering a wide range of features, such as:

- Up to 1.5 DMIPS/MHz
- 32-bit Address and Data paths
- 32-bit Linear (program space) addressing
- (2) thirty-two element 32-bit core register files
- Single-cycle multiply and high-performance divide unit for 32-bit integer math
- 16 and 32-bit instructions, optimized for high-level languages, such as ‘C’

1.3 Power-Saving Technology

All of the devices in the PIC32MX Family incorporate a range of features that can significantly reduce power consumption during operation. Key features include:

- **On-the-Fly Clock Switching:** The device clock can be changed under software control to any of the four clock sources during operation.
- **Instruction-Based Power-Saving Modes:** The microcontroller can suspend all operations, or selectively shut down its core while leaving its peripherals active, with a single instruction in software.

1.4 Communications

The PIC32MX Family incorporates a range of serial communication peripherals to handle a range of application requirements. All devices are equipped with two independent UARTs with built-in IrDA encoder/decoders. There are also two independent SPI modules, and two independent I^2C modules that support both Master and Slave modes of operation.

1.5 10-Bit A/D Converter

The A/D Converter features 400+ ksps maximum sample rate. This configurable module incorporates a user-selectable scan list and auto-convert functions to allow acquisitions without processor intervention. Multiple A/D trigger sources are user-selectable: timer event, external pin, manual and auto-convert.

1.6 External Interface

A Parallel Master Port Parallel Slave Port enables 8/16-bit parallel data communications in Master mode with up to 16 address lines; 8-bit Slave modes are also supported.

1.7 Real-Time Clock/Calendar

This module implements a full-featured clock and calendar with alarm functions in hardware, freeing up timer resources and program memory space for use of the core application.

1.8 Oscillator Options and Features

All of the devices in the PIC32MX Family offer four different oscillator options, allowing users a range of choices in developing application hardware. These include:

- A Primary Oscillator (POSC) with two External Crystal modes using crystals or ceramic resonators.
- Two External Clock modes with selectable peripheral bus clock output.
- A Fast Internal Oscillator (FRC) with a nominal 8 MHz output.
- On-board postscalers and/or PLL to provide clock speeds ranging from 31 kHz to maximum specified frequency.
- A Secondary Oscillator (SOSC) designed to operate with an external 32.768 kHz crystal. This oscillator can also be used with Timer1 and the integrated RTCC.
- An Internal Low-Power RC oscillator (LPRC) having a fixed 31 kHz output, which provides a low-power option for timing-insensitive applications.

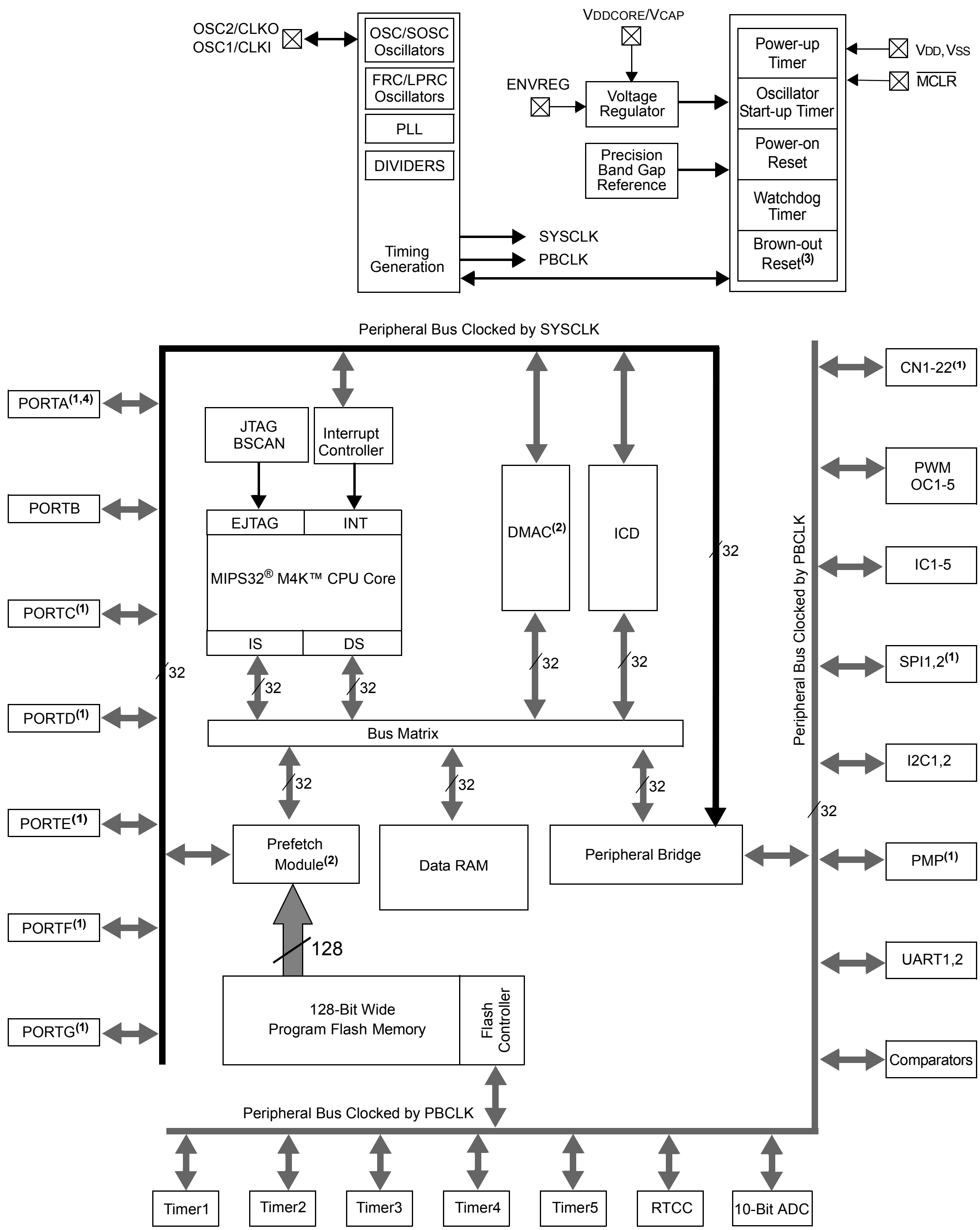
The oscillator block also provides a stable reference source for the user-controlled Fail-Safe Clock Monitor. This option constantly monitors the main clock source against a reference signal provided by the internal oscillator and enables the controller to switch to the internal oscillator, allowing for continued low-speed operation or a safe application shutdown.

1.9 Device Features, Block Diagrams and Pinout Tables

TABLE 1-1: DEVICE FEATURES FOR THE PIC32MX3XXFXXX GENERAL PURPOSE FAMILY

Features	PIC32MX300F032H	PIC32MX320F064H	PIC32MX320F128H	PIC32MX340F256H	PIC32MX320F128L	PIC32MX360F256L	PIC32MX360F512L
Operating Frequency	DC – 20 MHz	DC – 72 MHz					
Program Memory (Bytes)	32K	64K	128K	256K	128K	256K	512K
Data Memory (Bytes)	8K	16K	16K	32K	16K	32K	32K
Interrupt Sources/Vectors	95 / 63						
I/O Ports	Ports B, C, D, E, F, G				Ports A, B, C, D, E, F, G		
Total I/O Pins	53				85		
DMA Channels	0			4	0	4	
Timers:							
Total number (16-bit)	5						
32-bit (paired 16-bit)	2						
32-bit core timer	1						
Input Capture Channels	5						
Output Compare/PWM Channels	5						
Input Change Interrupt Notification	19				22		
Serial Communications:							
Enhanced UART	2						
SPI (3-wire/4-wire)	2						
I^2C™	2						
Parallel Communications (PMP/PSP)	Yes						
JTAG Boundary Scan	Yes						
JTAG Debug and Program	Yes						
ICSP™ 2-Wire Debug and Program	Yes						
Instruction Trace	No					Yes	
Hardware Break Points	6 Instruction, 2 Data						
10-Bit Analog-to-Digital Module (input channels)	16						
Analog Comparators	2						
Internal LDO	Yes						
Resets (and delays)	POR, BOR, $\overline{MCLR}$, WDT, SWR (Software Reset), CM (Configuration Bit Mismatch) (PWRT, OST, PLL Lock)						
Instruction Support	MIPS32® Enhanced Architecture (Release 2) MIPS16e™ Code Compression						
Packages	64-pin TQFP				100-pin TQFP		

FIGURE 1-1: PIC32MX FAMILY BLOCK DIAGRAM (GENERAL PURPOSE)



Note 1: Not all pins or features are implemented on all device pinout configurations. See Table 1-2 for I/O port pin descriptions.

2: Some features are not available on certain devices.

3: BOR functionality is provided when the on-board voltage regulator is enabled.

4: PORTA is not present on 64-pin devices

 Advance Information

TABLE 1-2: PIC32MX FAMILY PINOUT DESCRIPTIONS – GENERAL PURPOSE

Function	Pin Number		I/O	Input Buffer	Description
	64-pin	100-pin			
AN0	16	25	I	ANA	A/D Analog Inputs.
AN1	15	24	I	ANA	
AN2	14	23	I	ANA	
AN3	13	22	I	ANA	
AN4	12	21	I	ANA	
AN5	11	20	I	ANA	
AN6	17	26	I	ANA	
AN7	18	27	I	ANA	
AN8	21	32	I	ANA	
AN9	22	33	I	ANA	
AN10	23	34	I	ANA	
AN11	24	35	I	ANA	
AN12	27	41	I	ANA	
AN13	28	42	I	ANA	
AN14	29	43	I	ANA	
AN15	30	44	I	ANA	
AVDD	19	30	P	—	Positive Supply for Analog Modules.
AVSS	20	31	P	—	Ground Reference for Analog Modules.
BCLK1	35	48	O	—	UART1 IrDA® Baud Clock.
BCLK2	29	39	O	—	UART2 IrDA Baud Clock.
C1IN-	12	21	I	ANA	Comparator 1 Negative Input.
C1IN+	11	20	I	ANA	Comparator 1 Positive Input.
C1OUT	21	32	O	—	Comparator 1 Output.
C2IN-	14	23	I	ANA	Comparator 2 Negative Input.
C2IN+	13	22	I	ANA	Comparator 2 Positive Input.
C2OUT	22	33	O	—	Comparator 2 Output.
CLKI	39	63	I	ANA	Main Clock Input Connection.
CLKO	40	64	O	—	System Clock Output.
CN0	48	74	I	ST	Interrupt-on-Change Inputs.
CN1	47	73	I	ST	
CN2	16	25	I	ST	
CN3	15	24	I	ST	
CN4	14	23	I	ST	
CN5	13	22	I	ST	
CN6	12	21	I	ST	
CN7	11	20	I	ST	
CN8	4	10	I	ST	
CN9	5	11	I	ST	
CN10	6	12	I	ST	
CN11	8	14	I	ST	
CN12	30	44	I	ST	
CN13	52	81	I	ST	
CN14	53	82	I	ST	
CN15	54	83	I	ST	
CN16	55	84	I	ST	
CN17	31	49	I	ST	

Legend: TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; I^2C™ = I^2C/SMBus input buffer

TABLE 1-2: PIC32MX FAMILY PINOUT DESCRIPTIONS – GENERAL PURPOSE (CONTINUED)

Function	Pin Number		I/O	Input Buffer	Description
	64-pin	100-pin			
CN18	32	50	I	ST	Interrupt-on-Change Inputs.
CN19	—	80	I	ST	
CN20	—	47	I	ST	
CN21	—	48	I	ST	
CVREF	23	34	O	ANA	Comparator Voltage Reference Output.
ENVREG	57	86	I	ST	Enable for On-Chip Voltage Regulator.
IC1	42	68	I	ST	Input Capture Inputs.
IC2	43	69	I	ST	
IC3	44	70	I	ST	
IC4	45	71	I	ST	
IC5	52	79	I	ST	
INT0	35	55	I	ST	External Interrupt Inputs.
INT1	42	18	I	ST	
INT2	43	19	I	ST	
INT3	44	66	I	ST	
INT4	45	67	I	ST	
$\overline{\text{MCLR}}$	7	13	I	ST	Master Clear (Device Reset) Input. Bring this line low to cause a Reset.
OC1	46	72	O	—	Output Compare/PWM Outputs.
OC2	49	76	O	—	
OC3	50	77	O	—	
OC4	51	78	O	—	
OC5	52	81	O	—	
OCFA	17	26	I	ST	Output Compare Fault A Input.
OCFB	30	44	I	ST	Output Compare Fault B Input.
OSC1	39	63	I	ANA	Main Oscillator Input Connection.
OSC2	40	64	O	ANA	Main Oscillator Output Connection.
PGC1	15	24	I/O	ST	In-Circuit Debugger and ICSP™ Programming Clock
PGD1	16	25	I/O	ST	In-Circuit Debugger and ICSP Programming Data.
PGC2	17	26	I/O	ST	In-Circuit Debugger and ICSP™ Programming Clock.
PGD2	18	27	I/O	ST	In-Circuit Debugger and ICSP Programming Data.
PMA0/ PMALL	30	44	I/O	ST	Parallel Master Port Address Bit 0 Input (Buffered Slave modes) and Output (Master modes). Parallel Master Port Address Latch Enable low-byte (Multiplexed Master modes).
PMA1/ PMALH	29	43	I/O	ST	Parallel Master Port Address Bit 1 Input (Buffered Slave modes) and Output (Master modes). Parallel Master Port Address Latch Enable high-byte (Multiplexed Master modes).

Legend: TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; I^2C™ = I^2C/SMBus input buffer

TABLE 1-2: PIC32MX FAMILY PINOUT DESCRIPTIONS – GENERAL PURPOSE (CONTINUED)

Function	Pin Number		I/O	Input Buffer	Description
	64-pin	100-pin			
PMA2	8	14	O	—	Parallel Master Port Address (Demultiplexed Master modes).
PMA3	6	12	O	—	
PMA4	5	11	O	—	
PMA5	4	10	O	—	
PMA6	16	29	O	—	
PMA7	22	28	O	—	
PMA8	32	50	O	—	
PMA9	31	49	O	—	
PMA10	28	42	O	—	
PMA11	27	41	O	—	
PMA12	24	35	O	—	
PMA13	23	34	O	—	
PMCS1/ PMA14	45	71	O	—	Parallel Master Port Chip Select 1 Strobe/Address bit 14.
PMCS2/ PMA15	44	70	O	—	Parallel Master Port Chip Select 2 Strobe/Address bit 15.
PMD0	60	93	I/O	ST/TTL	Parallel Master Port Data (Demultiplexed Master mode) or Address/ Data (Multiplexed Master modes).
PMD1	61	94	I/O	ST/TTL	
PMD2	62	98	I/O	ST/TTL	
PMD3	63	99	I/O	ST/TTL	
PMD4	64	100	I/O	ST/TTL	
PMD5	1	3	I/O	ST/TTL	
PMD6	2	4	I/O	ST/TTL	
PMD7	3	5	I/O	ST/TTL	
PMD8	—	90	I/O	ST/TTL	
PMD9	—	89	I/O	ST/TTL	
PMD10	—	88	I/O	ST/TTL	
PMD11	—	87	I/O	ST/TTL	
PMD12	—	79	I/O	ST/TTL	
PMD13	—	80	I/O	ST/TTL	
PMD14	—	83	I/O	ST/TTL	
PMD15	—	84	I/O	ST/TTL	
PMRD/ PMRD/$\overline{\text{PMWR}}$	53	82	O	—	Parallel Master Port Read Strobe (Master Mode 2) Parallel Master Port Read/Write Strobe (Master Mode 1).
PMWR/ PMENB	52	81	O	—	Parallel Master Port Write Strobe (Master Mode 2) Parallel Master Port Enable Strobe (Master Mode 1).

Legend: TTL = TTL input buffer; ANA = Analog level input/output; ST = Schmitt Trigger input buffer; I^2C™ = I^2C/SMBus input buffer

TABLE 1-2: PIC32MX FAMILY PINOUT DESCRIPTIONS – GENERAL PURPOSE (CONTINUED)

Function	Pin Number		I/O	Input Buffer	Description
	64-pin	100-pin			
RA0	—	17	I/O	ST	PORTA Digital I/O.
RA1	—	38	I/O	ST	
RA2	—	58	I/O	ST	
RA3	—	59	I/O	ST	
RA4	—	60	I/O	ST	
RA5	—	61	I/O	ST	
RA6	—	91	I/O	ST	
RA7	—	92	I/O	ST	
RA9	—	28	I/O	ST	
RA10	—	29	I/O	ST	
RA14	—	66	I/O	ST	
RA15	—	67	I/O	ST	
RB0	16	25	I/O	ST	PORTB Digital I/O.
RB1	15	24	I/O	ST	
RB2	14	23	I/O	ST	
RB3	13	22	I/O	ST	
RB4	12	21	I/O	ST	
RB5	11	20	I/O	ST	
RB6	17	26	I/O	ST	
RB7	18	27	I/O	ST	
RB8	21	32	I/O	ST	
RB9	22	33	I/O	ST	
RB10	23	34	I/O	ST	
RB11	24	35	I/O	ST	
RB12	27	41	I/O	ST	
RB13	28	42	I/O	ST	
RB14	29	43	I/O	ST	
RB15	30	44	I/O	ST	
RC1	—	6	I/O	ST	PORTC Digital I/O.
RC2	—	7	I/O	ST	
RC3	—	8	I/O	ST	
RC4	—	9	I/O	ST	
RC12	39	63	I/O	ST	
RC13	47	73	I/O	ST	
RC14	48	74	I/O	ST	
RC15	40	64	I/O	ST	

Legend: TTL = TTL input buffer; ST = Schmitt Trigger input buffer
ANA = Analog level input/output; I^2C™ = I^2C/SMBus input buffer

 Advance Information

TABLE 1-2: PIC32MX FAMILY PINOUT DESCRIPTIONS – GENERAL PURPOSE (CONTINUED)

Function	Pin Number		I/O	Input Buffer	Description
	64-pin	100-pin			
RD0	46	72	I/O	ST	PORTD Digital I/O.
RD1	49	76	I/O	ST	
RD2	50	77	I/O	ST	
RD3	51	78	I/O	ST	
RD4	52	81	I/O	ST	
RD5	53	82	I/O	ST	
RD6	54	83	I/O	ST	
RD7	55	84	I/O	ST	
RD8	42	68	I/O	ST	
RD9	43	69	I/O	ST	
RD10	44	70	I/O	ST	
RD11	45	71	I/O	ST	
RD12	—	79	I/O	ST	
RD13	—	80	I/O	ST	
RD14	—	47	I/O	ST	
RD15	—	48	I/O	ST	
RE0	60	93	I/O	ST	PORTE Digital I/O.
RE1	61	94	I/O	ST	
RE2	62	98	I/O	ST	
RE3	63	99	I/O	ST	
RE4	64	100	I/O	ST	
RE5	1	3	I/O	ST	
RE6	2	4	I/O	ST	
RE7	3	5	I/O	ST	
RE8	—	18	I/O	ST	
RE9	—	19	I/O	ST	
RF0	58	87	I/O	ST	PORTF Digital I/O.
RF1	59	88	I/O	ST	
RF2	34	52	I/O	ST	
RF3	33	51	I/O	ST	
RF4	31	49	I/O	ST	
RF5	32	50	I/O	ST	
RF6	35	55	I/O	ST	
RF7	—	54	I/O	ST	
RF8	—	53	I/O	ST	
RF12	—	40	I/O	ST	
RF13	—	39	I/O	ST	

Legend: TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; I^2C™ = I^2C/SMBus input buffer

TABLE 1-2: PIC32MX FAMILY PINOUT DESCRIPTIONS – GENERAL PURPOSE (CONTINUED)

Function	Pin Number		I/O	Input Buffer	Description
	64-pin	100-pin			
RG0	—	90	I/O	ST	PORTG Digital I/O.
RG1	—	89	I/O	ST	
RG2	37	57	I/O	ST	
RG3	36	56	I/O	ST	
RG6	4	10	I/O	ST	
RG7	5	11	I/O	ST	
RG8	6	12	I/O	ST	
RG9	8	14	I/O	ST	
RG12	—	96	I/O	ST	
RG13	—	97	I/O	ST	
RG14	—	95	I/O	ST	
RG15	—	1	I/O	ST	
RTCC	42	68	O	—	Real-Time Clock Alarm Output.
SCK1	35	55	O	—	SPI1 Serial Clock Output.
SCK2	4	10	I/O	ST	SPI2 Serial Clock Output.
SCL1	37	57	I/O	I^2C	I2C1 Synchronous Serial Clock Input/Output.
SCL2	32	58	I/O	I^2C	I2C2 Synchronous Serial Clock Input/Output.
SDA1	36	56	I/O	I^2C	I2C1 Data Input/Output.
SDA2	31	59	I/O	I^2C	I2C2 Data Input/Output.
SDI1	34	54	I	ST	SPI1 Serial Data Input.
SDI2	5	11	I	ST	SPI2 Serial Data Input.
SDO1	33	53	O	—	SPI1 Serial Data Output.
SDO2	6	12	O	—	SPI2 Serial Data Output.
SOSCI	47	73	I	ANA	Secondary Oscillator/Timer1 Clock Input.
SOSCO	48	74	O	ANA	Secondary Oscillator/Timer1 Clock Output.
$\overline{SS1}$	14	23	I/O	ST	Slave Select Input/Frame Select Output (SPI1).
$\overline{SS2}$	8	14	I/O	ST	Slave Select Input/Frame Select Output (SPI2).
T1CK	48	74	I	ST	Timer1 Clock.
T2CK	—	6	I	ST	Timer2 External Clock Input.
T3CK	—	7	I	ST	Timer3 External Clock Input.
T4CK	—	8	I	ST	Timer4 External Clock Input.
T5CK	—	9	I	ST	Timer5 External Clock Input.
TCK	27	38	I	ST	JTAG Test Clock/Programming Clock Input.
TDI	28	60	I	ST	JTAG Test Data/Programming Data Input.
TDO	24	61	O	—	JTAG Test Data Output.
TMS	23	17	I	ST	JTAG Test Mode Select Input.
TRCLK	—	91	O	—	Trace Clock.
TRD0	—	97	O	—	Trace Data Bit 0.
TRD1	—	96	O	—	Trace Data Bit 1.
TRD2	—	95	O	—	Trace Data Bit 2.
TRD3	—	92	O	—	Trace Data Bit 3.

Legend: TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; I^2C™ = I^2C/SMBus input buffer

Advance Information

TABLE 1-2: PIC32MX FAMILY PINOUT DESCRIPTIONS – GENERAL PURPOSE (CONTINUED)

Function	Pin Number		I/O	Input Buffer	Description
	64-pin	100-pin			
$\overline{U1CTS}$	43	47	I	ST	UART1 Clear to Send Input.
$\overline{U1RTS}$	35	48	O	—	UART1 Request to Send Output.
U1RX	34	52	I	ST	UART1 Receive.
U1TX	33	51	O	DIG	UART1 Transmit Output.
$\overline{U2CTS}$	21	40	I	ST	UART2 Clear to Send Input.
$\overline{U2RTS}$	29	39	O	—	UART2 Request to Send Output.
U2RX	31	49	I	ST	UART 2 Receive Input.
U2TX	32	50	O	—	UART2 Transmit Output.
VDD	10, 26, 38	2, 16, 37, 46, 62	P	—	Positive Supply for Peripheral Digital Logic and I/O pins.
VDDCAP	56	85	P	—	External Filter Capacitor Connection (regulator enabled).
VDDCORE	56	85	P	—	Positive Supply for Microcontroller Core Logic (regulator disabled).
VREF-	15	28	I	ANA	A/D and Comparator Reference Voltage (Low) Input.
VREF+	16	29	I	ANA	A/D and Comparator Reference Voltage (High) Input.
VSS	9, 25, 41	15, 36, 45, 65, 75	P	—	Ground Reference for Logic and I/O pins.

Legend: TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; I^2C™ = I^2C/SMBus input buffer

NOTES:

2.0 PIC32MX MCU

Note:	This data sheet summarizes the features of the PIC32MX of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

The MCU module is the heart of the PIC32MX processor. The MCU fetches instructions, decodes each instruction, fetches source operands, executes each instruction, and writes the results of instruction execution to the proper destinations.

2.1 Features

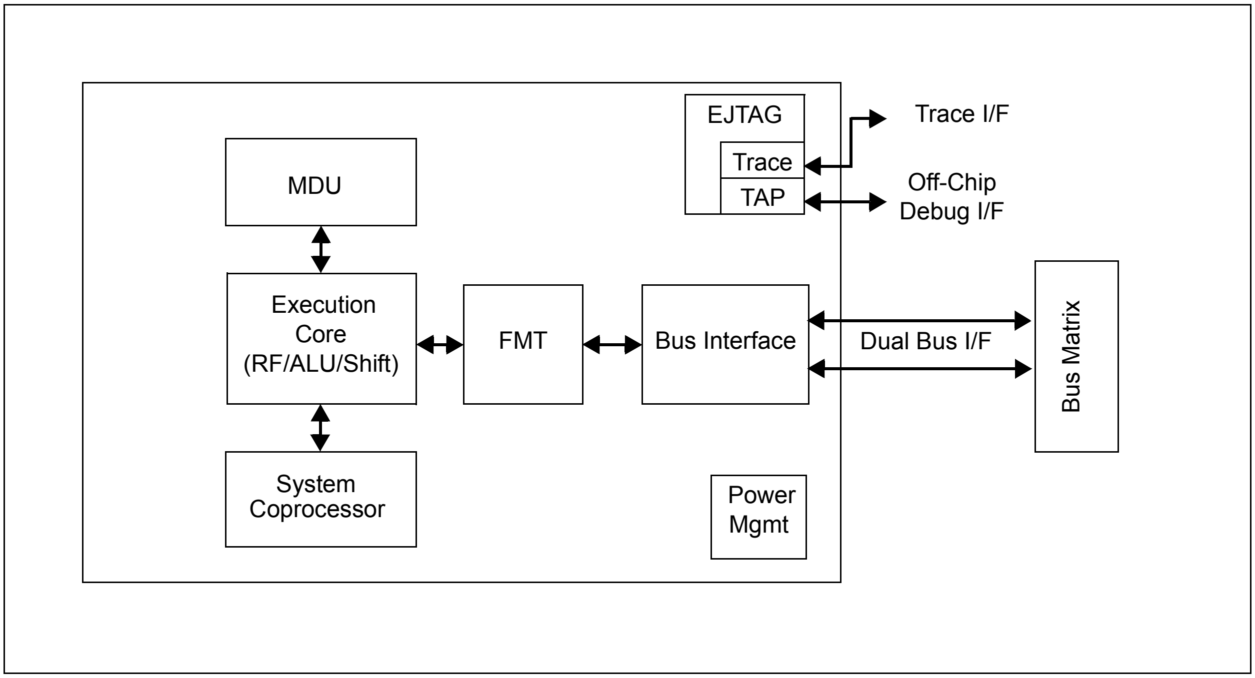
- 5-stage pipeline
- 32-bit Address and Data Paths
- MIPS32 Enhanced Architecture (Release 2)
  - Multiply-Accumulate and Multiply-Subtract Instructions
  - Targeted Multiply Instruction
  - Zero/One Detect Instructions
  - Wait Instruction
  - Conditional Move Instructions (`MOVN`, `MOVZ`)
  - Vectored interrupts
  - Programmable exception vector base
  - Atomic interrupt enable/disable
  - GPR shadow registers to minimize latency for interrupt handlers
  - Bit field manipulation instructions
- MIPS16e™ Code Compression
  - 16 bit encodings of 32 bit instructions to improve code density
  - Special PC-relative instructions for efficient loading of addresses and constants
  - SAVE & RESTORE macro instructions for setting up and tearing down stack frames within subroutines
  - Improved support for handling 8 and 16 bit data types
- Simple Fixed Mapping Translation (FMT) mechanism
- Simple Dual Bus Interface
  - Independent 32-bit address and data busses
  - Transactions can be aborted to improve interrupt latency
- Autonomous Multiply/Divide Unit
  - Maximum issue rate of one 32x16 multiply per clock
  - Maximum issue rate of one 32x32 multiply every other clock
  - Early-in iterative divide. Minimum 11 and maximum 34 clock latency (dividend (*rs*) sign extension-dependent)
- Power Control
  - Minimum frequency: 0 MHz
  - Low-Power mode (triggered by WAIT instruction)
  - Extensive use of local gated clocks
- EJTAG Debug and Instruction Trace
  - Support for single stepping
  - Virtual instruction and data address/value breakpoints
  - PC tracing w/ trace compression

2.2 Architecture Overview

The PIC32MX core contains several logic blocks working together in parallel, providing an efficient high performance computing engine. The blocks included with the PIC32MX core are as follows:

- Execution Unit
- Multiply/Divide Unit (MDU)
- System Control Coprocessor (CP0)
- Fixed Mapping Translation (FMT)
- Dual Internal Bus interfaces
- Power Management
- MIPS16e support
- Enhanced JTAG (EJTAG) Controller

FIGURE 2-1: MCU BLOCK DIAGRAM



 Advance Information

2.2.1 EXECUTION UNIT

The PIC32MX core execution unit implements a load/store architecture with single-cycle ALU operations (logical, shift, add, subtract) and an autonomous multiply/divide unit. The PIC32MX core contains thirty-two 32-bit general-purpose registers used for integer operations and address calculation. One additional register file shadow set (containing thirty-two registers) is added to minimize context switching overhead during interrupt/exception processing. The register file consists of two read ports and one write port and is fully bypassed to minimize operation latency in the pipeline.

The execution unit includes:

- 32-bit adder used for calculating the data address
- Address unit for calculating the next instruction address
- Logic for branch determination and branch target address calculation
- Load aligner
- Bypass multiplexers used to avoid stalls when executing instructions streams where data producing instructions are followed closely by consumers of their results
- Leading Zero/One detect unit for implementing the CLZ and CLO instructions
- Arithmetic Logic Unit (ALU) for performing bitwise logical operations
- Shifter & Store Aligner

2.2.2 MULTIPLY/DIVIDE UNIT (MDU)

The PIC32MX core includes a multiply/divide unit (MDU) that contains a separate pipeline for multiply and divide operations. This pipeline operates in parallel with the integer unit (IU) pipeline and does not stall when the IU pipeline stalls. This allows MDU operations to be partially masked by system stalls and/or other integer unit instructions.

The high-performance MDU consists of a 32x16 booth recoded multiplier, result/accumulation registers (HI and LO), a divide state machine, and the necessary multiplexers and control logic. The first number shown (‘32’ of 32x16) represents the *rs* operand. The second number (‘16’ of 32x16) represents the *rt* operand. The PIC32MX core only checks the value of the latter (*rt)* operand to determine how many times the operation must pass through the multiplier. The 16x16 and 32x16 operations pass through the multiplier once. A 32x32 operation passes through the multiplier twice.

The MDU supports execution of one 16x16 or 32x16 multiply operation every clock cycle; 32x32 multiply operations can be issued every other clock cycle. Appropriate interlocks are implemented to stall the issuance of back-to-back 32x32 multiply operations. The multiply operand size is automatically determined by logic built into the MDU.

Divide operations are implemented with a simple 1 bit per clock iterative algorithm. An early-in detection checks the sign extension of the dividend (*rs*) operand. If rs is 8 bits wide, 23 iterations are skipped. For a 16-bit-wide rs, 15 iterations are skipped, and for a 24-bit-wide rs, 7 iterations are skipped. Any attempt to issue a subsequent MDU instruction while a divide is still active causes an IU pipeline stall until the divide operation is completed.

Table 2-1 lists the repeat rate (peak issue rate of cycles until the operation can be reissued) and latency (number of cycles until a result is available) for the PIC32MX core multiply and divide instructions. The approximate latency and repeat rates are listed in terms of pipeline clocks.

TABLE 2-1: PIC32MX CORE HIGH-PERFORMANCE INTEGER MULTIPLY/DIVIDE UNIT LATENCIES AND REPEAT RATES

Opcode	Operand Size (mul *rt*) (div *rs*)	Latency	Repeat Rate
MULT/MULTU, MADD/MADDU, MSUB/MSUBU	16 bits	1	1
	32 bits	2	2
MUL	16 bits	2	1
	32 bits	3	2
DIV/DIVU	8 bits	12	11
	16 bits	19	18
	24 bits	26	25
	32 bits	33	32

The MIPS architecture defines that the result of a multiply or divide operation be placed in the HI and LO registers. Using the Move-From-HI (`MFHI`) and Move-From-LO (`MFLO`) instructions, these values can be transferred to the general-purpose register file.

In addition to the HI/LO targeted operations, the MIPS32 architecture also defines a multiply instruction, MUL, which places the least significant results in the primary register file instead of the HI/LO register pair. By avoiding the explicit `MFLO` instruction, required when using the LO register, and by supporting multiple destination registers, the throughput of multiply-intensive operations is increased.

Two other instructions, multiply-add (MADD) and multiply-subtract (MSUB), are used to perform the multiply-accumulate and multiply-subtract operations. The MADD instruction multiplies two numbers and then adds the product to the current contents of the HI and LO registers. Similarly, the MSUB instruction multiplies two operands and then subtracts the product from the HI and LO registers. The MADD and MSUB operations are commonly used in DSP algorithms.

2.2.3 SYSTEM CONTROL COPROCESSOR (CP0)

In the MIPS architecture, CP0 is responsible for the virtual-to-physical address translation, the exception control system, the processor's diagnostics capability, the operating modes (kernel, user, and debug), and whether interrupts are enabled or disabled. Configuration information, such as presence of options like MIPS16e, is also available by accessing the CP0 registers, listed in Table 2-2.

TABLE 2-2: COPROCESSOR 0 REGISTERS

Register Number	Register Name	Function
0-6	Reserved	Reserved in the PIC32MX core
7	HWREna	Enables access via the RDHWR instruction to selected hardware registers
8	BadVAddr(1)	Reports the address for the most recent address-related exception
9	Count(1)	Processor cycle count
10	Reserved	Reserved in the PIC32MX core
11	Compare(1)	Timer interrupt control
12	Status(1)	Processor status and control
12	IntCtl(1)	Interrupt system status and control
12	SRSCtl(1)	Shadow register set status and control
12	SRSMap(1)	Provides mapping from vectored interrupt to a shadow set
13	Cause(1)	Cause of last general exception
14	EPC(1)	Program counter at last exception
15	PRId	Processor identification and revision
15	EBASE	Exception vector base register
16	Config	Configuration register
16	Config1	Configuration register 1
16	Config2	Configuration register 2
16	Config3	Configuration register 3
17-22	Reserved	Reserved in the PIC32MX core
23	Debug(2)	Debug control and exception status
24	DEPC(2)	Program counter at last debug exception.
25-29	Reserved	Reserved in the PIC32MX core.
30	ErrorEPC(1)	Program counter at last error.
31	DESAVE(2)	Debug handler scratchpad register.

Note 1: Registers used in exception processing.

2: Registers used during debug.

Coprocessor 0 also contains the logic for identifying and managing exceptions. Exceptions can be caused by a variety of sources, including alignment errors in data, external events, or program errors. Table 2-3 shows the exception types in order of priority.

TABLE 2-3: PIC32MX CORE EXCEPTION TYPES

Exception	Description
Reset	Assertion MCLR or a Power-On Reset (POR)
DSS	EJTAG Debug Single Step.
DINT	EJTAG Debug Interrupt. Caused by the assertion of the external *EJ_DINT* input, or by setting the EjtagBrk bit in the ECR register.
NMI	Assertion of *NMI* signal.
Interrupt	Assertion of unmasked hardware or software interrupt signal.
DIB	EJTAG debug hardware instruction break matched.
AdEL	Fetch address alignment error. Fetch reference to protected address.
IBE	Instruction fetch bus error.
DBp	EJTAG Breakpoint (execution of SDBBP instruction).
Sys	Execution of SYSCALL instruction.
Bp	Execution of BREAK instruction.
RI	Execution of a Reserved Instruction.
CpU	Execution of a coprocessor instruction for a coprocessor that is not enabled.
CEU	Execution of a CorExtend instruction when CorExtend is not enabled.
Ov	Execution of an arithmetic instruction that overflowed.
Tr	Execution of a trap (when trap condition is true).
DDBL / DDBS	EJTAG Data Address Break (address only) or EJTAG Data Value Break on Store (address + value).
AdEL	Load address alignment error. Load reference to protected address.
AdES	Store address alignment error. Store to protected address.
DBE	Load or store bus error.
DDBL	EJTAG data hardware breakpoint matched in load data compare.

2.2.4 INTERRUPT HANDLING

The PIC32MX core includes support for peripheral interrupts, two software interrupts, and a timer interrupt.

The PIC32MX MCU uses the MIPS External Interrupt Controller (EIC) mode, which redefines the way in which interrupts are handled to provide full support for an external interrupt controller handling prioritization and vectoring of interrupts. This presence of this mode denoted by the VEIC bit in the *Config3* register. On the PIC32MX core, the VEIC bit is always set to `1` to indicate the presence of an external interrupt controller.

Note:	Although EIC mode is designated as "External", the interrupt controller is on-chip.

The interrupt controller specifies which shadow set should be used upon entry to a particular vector. The shadow registers further improve interrupt latency by avoiding the need to save context when invoking an interrupt handler.

2.2.5 GPR SHADOW REGISTERS

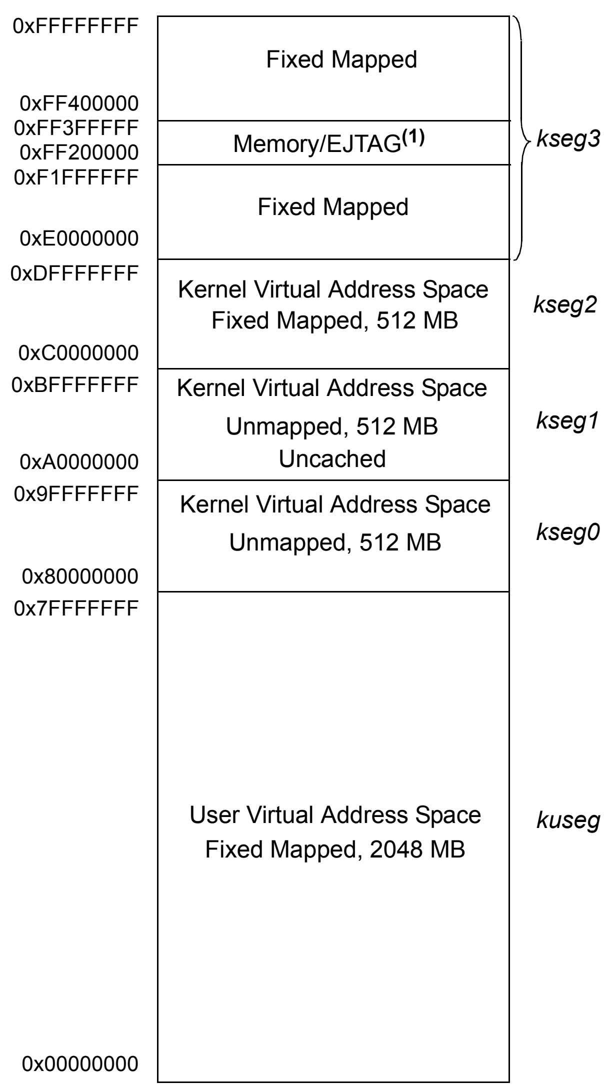
Release 2 of the MIPS32 Architecture optionally removes the need to save and restore GPRs on entry to high priority interrupts or exceptions, and to provide specified processor modes with the same capability. This is done by introducing multiple copies of the GPRs, called *shadow sets*, and allowing privileged software to associate a shadow set with entry to kernel mode via an interrupt vector or exception. The normal GPRs are logically considered shadow set zero.

The PIC32MX core implements two sets of registers, the normal GPRs, and one shadow set. This is indicated by the $SRSCtl_{HSS}$ field.

2.3 Modes of Operation

The PIC32MX core supports three modes of operation: user mode, kernel mode, and debug mode. User mode is most often used for applications programs. Kernel mode is typically used for handling exceptions and operating system kernel functions, including CP0 management and I/O device accesses. An additional Debug mode is used during system bring-up and software development. Refer to the EJTAG specification for more information on debug mode.

FIGURE 2-2: PIC32MX CORE VIRTUAL ADDRESS MAP

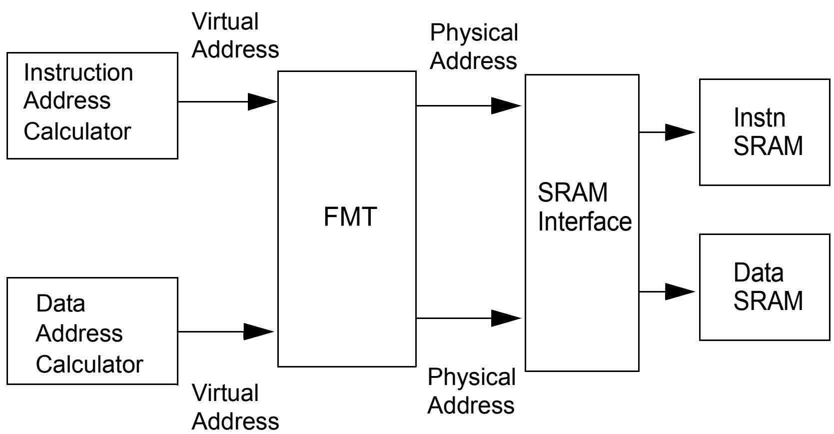


Note 1: This space is mapped to memory in user or kernel mode, and by the EJTAG module in Debug mode.

2.3.1 FIXED MAPPING TRANSLATION

The PIC32MX core provides a simple Fixed Mapping Translation (FMT) mechanism that is smaller and simpler than a full Translation Lookaside Buffer (TLB) found in other MIPS cores. Like a TLB, the FMT performs virtual-to-physical address translation and provides attributes for the different segments. Those segments that are unmapped in a TLB implementation (kseg0 and kseg1) are translated identically by the FMT. Figure 2-3 shows how the FMT is implemented in the PIC32MX core.

FIGURE 2-3: ADDRESS TRANSLATION DURING MEMORY ACCESS



In general, the FMT also determines the cacheability of each segment. These attributes are controlled via bits in the Config register. Table 2-4 shows the encoding for the K23 (bits 30:28), KU (bits 27:25), and K0 (bits 2:0) fields of the Config register. The PIC32MX core passes these Config fields to the Prefetch Cache module to determine cacheability of Program Memory Flash accesses. Table 2-5 shows how the cacheability of the virtual address segments is controlled by these fields.

TABLE 2-4: CACHE COHERENCY ATTRIBUTES

Config Register Fields K23, KU, and K0	Cache Coherency Attribute
2	Uncached.
3	Cacheable

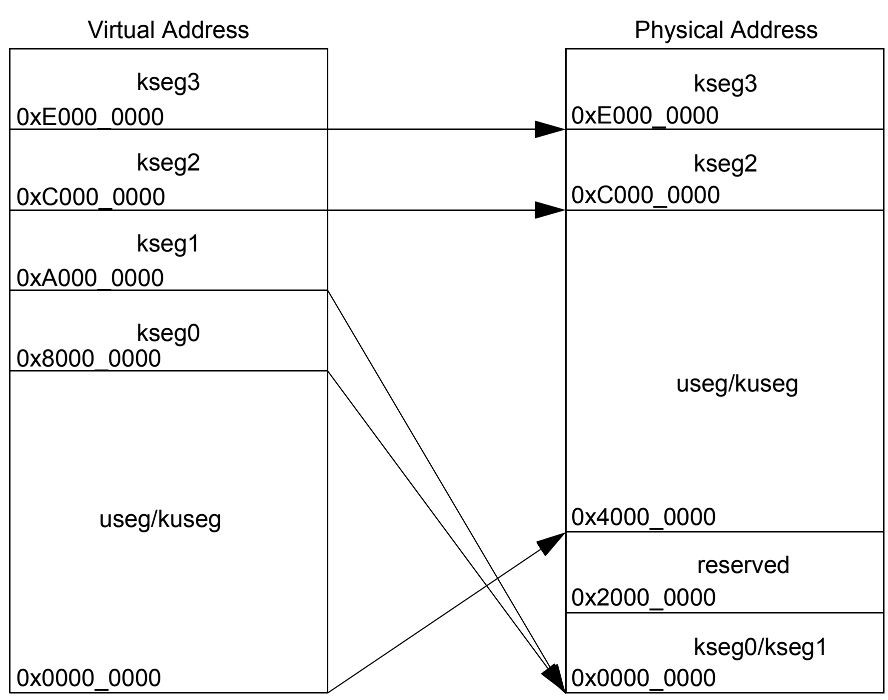
In the PIC32MX core, no translation exceptions are taken, although address errors are still possible.

TABLE 2-5: CACHEABILITY OF SEGMENTS WITH FIXED MAPPING TRANSLATION

Segment	Virtual Address Range	Cacheability
useg/kuseg	0x0000_0000-0x7FFF_FFFF	Controlled by the KU field (bits 27:25) of the Config register. See Figure 2-4 for mapping. This segment is always uncached when ERL = 1.
kseg0	0x8000_0000- 0x9FFF_FFFF	Controlled by the K0 field (bits 2:0) of the Config register. See Figure 2-4 for mapping.
kseg1	0xA000_0000-0xBFFF_FFFF	Always uncacheable.
kseg2	0xC000_0000-0xDFFF_FFFF	Controlled by the K23 field (bits 30:28) of the Config register. See Figure 2-4 for mapping.
kseg3	0xE000_0000-0xFFFF_FFFF	Controlled by the K23 field (bits 30:28) of the Config register. See Figure 2-4 for mapping.

The FMT performs a simple translation to map from virtual addresses to physical addresses. This mapping is shown in Figure 2-4.

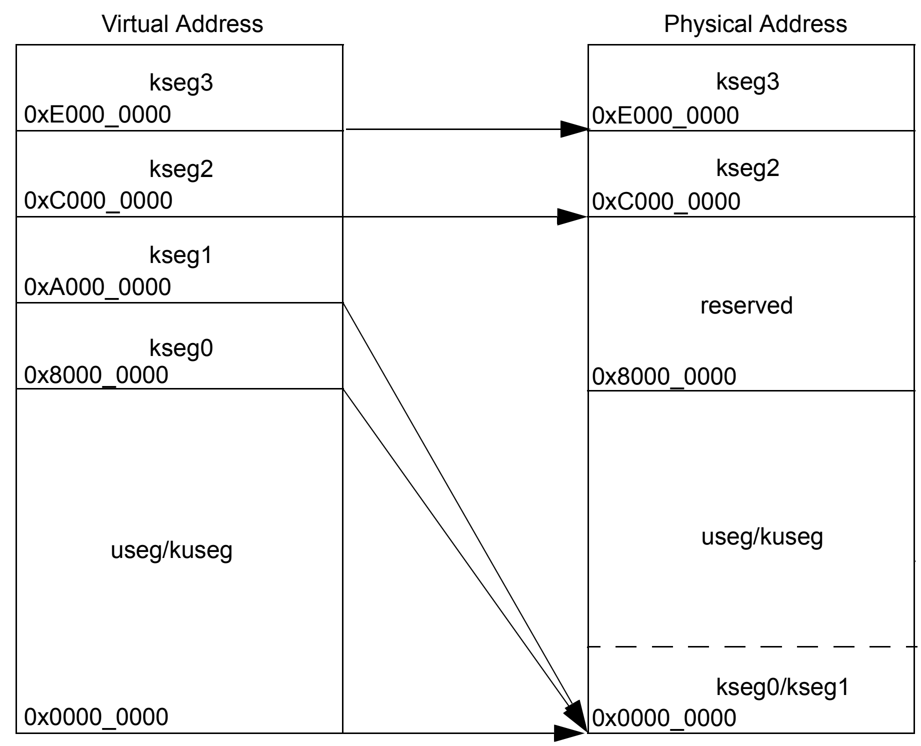
FIGURE 2-4: FMT MEMORY MAP (ERL = 0) IN THE PIC32MX CORE



 Advance Information

When ERL = 1, useg and kuseg become unmapped (virtual address is identical to the physical address) and uncached. This behavior is the same as if there was a TLB. This mapping is shown in Figure 2-5.

FIGURE 2-5: PIC32MX CORE FMT MEMORY MAP (ERL = 1)



2.3.2 DUAL INTERNAL BUS INTERFACES

The SRAM interface includes dual instruction and data interfaces.

The dual interface enables independent connection to instruction and data devices. It yields the highest performance, since the pipeline can generate simultaneous I and D requests which are then serviced in parallel.

The internal buses are connected to the Bus Matrix unit, which is a switch fabric that provides this parallel operation.

2.3.3 MIPS16E EXECUTION

When the core is operating in MIPS16e mode, instruction fetches only require 16-bits of data to be returned. For improved efficiency, however, the core will fetch 32-bits of instruction data whenever the address is word-aligned. Thus for sequential MIPS16e code, fetches only occur for every other instruction, resulting in better performance and reduced system power.

2.4 Power Management

The PIC32MX core offers a number of power management features, including low-power design, active power management, and power-down modes of operation. The core is a static design that supports slowing or halting the clocks, which reduces system power consumption during idle periods.

2.4.1 INSTRUCTION-CONTROLLED POWER MANAGEMENT

The mechanism for invoking power-down mode is through execution of the WAIT instruction. For more information on power management, see **11.0 "Power Saving"**.

2.4.2 LOCAL CLOCK GATING

The majority of the power consumed by the PIC32MX core is in the clock tree and clocking registers. The PIC32MX uses extensive use of local gated-clocks to reduce this dynamic power consumption.

2.5 EJTAG Debug Support

The PIC32MX core provides for an Enhanced JTAG (EJTAG) interface for use in the software debug of application and kernel code. In addition to standard user mode and kernel modes of operation, the PIC32MX core provides a Debug mode that is entered after a debug exception (derived from a hardware breakpoint, single-step exception, etc.) is taken and continues until a debug exception return (DERET) instruction is executed. During this time, the processor executes the debug exception handler routine.

The EJTAG interface operates through the Test Access Port (TAP), a serial communication port used for transferring test data in and out of the PIC32MX core. In addition to the standard JTAG instructions, special instructions defined in the EJTAG specification define what registers are selected and how they are used.

2.5.1 DEBUG REGISTERS

Three debug registers (DEBUG, DEPC, and DESAVE) have been added to the MIPS Coprocessor 0 (CP0) register set. The DEBUG register shows the cause of the debug exception and is used for setting up single-step operations. The DEPC, or Debug Exception Program Counter, register holds the address on which the debug exception was taken. This is used to resume program execution after the debug operation finishes. Finally, the DESAVE, or Debug Exception Save, register enables the saving of general-purpose registers used during execution of the debug exception handler.

To exit debug mode, a Debug Exception Return (DERET) instruction is executed. When this instruction is executed, the system exits debug mode, allowing normal execution of application and system code to resume.

2.5.2 EJTAG HARDWARE BREAKPOINTS

There are several types of simple hardware breakpoints defined in the EJTAG specification. These stop the normal operation of the MCU and force the system into debug mode. There are two types of simple hardware breakpoints implemented in the PIC32MX core: Instruction breakpoints and Data breakpoints.

The PIC32MX core has two data and six instruction breakpoints

Instruction breaks occur on instruction fetch operations, and the break is set on the virtual address. A mask can be applied to the virtual address to set breakpoints on a range of instructions.

Data breakpoints occur on load/store transactions. Breakpoints are set on virtual address values, similar to the Instruction breakpoint. Data breakpoints can be set on a load, a store, or both. Data breakpoints can also be set based on the value of the load/store operation. Finally, masks can be applied to both the virtual address and the load/store value.

2.5.3 INSTRUCTION TRACING

The PIC32MX core includes Trace support for real-time tracing of instruction addresses. The trace information is collected in an off-chip memory, for post-capture processing by trace regeneration software.

Off-chip trace memory is accessed through a special trace probe that consists of 4 data pins plus a clock.

2.6 MCU Initialization

Software is required to initialize the following parts of the device after a reset event.

2.6.1 GENERAL-PURPOSE REGISTERS

The MCU register file powers up in an unknown state with the exception of r0 which is always 0. Initializing the rest of the register file is not required for proper operation of hardware. Depending on the software environment however, several registers may need to be initialized. Some of these are:

- SP - Stack Pointer
- GP - Global Pointer
- FP - Frame Pointer

2.6.2 COPROCESSOR 0 STATE

Miscellaneous CP0 states need to be initialized prior to leaving the boot code. There are various exceptions which are blocked by ERL = 1 or EXL = 1 and which are not cleared by Reset. These can be cleared to avoid taking spurious exceptions when leaving the boot code.

TABLE 2-6: CP0 INITIALIZATION

CP0 Register	Action
Cause	WP (Watch Pending), SW0/1 (Software Interrupts) should be cleared.
Config	Typically, the K0, KU and K23 fields should be set to the desired Cache Coherency Algorithm (CCA) value prior to accessing the corresponding memory regions. But in the M4K core, all CCA values are treated identically, so the hardware reset value of these fields need not be modified.
Count[(1)]	Should be set to a known value if Timer Interrupts are used.
Compare[(1)]	Should be set to a known value if Timer Interrupts are used. The write to compare will also clear any pending Timer Interrupts (Thus, Count should be set before Compare to avoid any unexpected interrupts).
Status	Desired state of the device should be set.
Other CP0 state	Other registers should be written before they are read. Some registers are not explicitly write-able, and are only updated as a by-product of instruction execution or a taken exception. Uninitialized bits should be masked off after reading these registers.

Note 1: When the Count register is equal to the Compare register a timer interrupt is signaled. There is a mask bit in the interrupt controller to disable passing this interrupt to the MCU if desired.

2.7 I/O Pin Configuration

The MCU module has EJTAG pins that may be configured as user-available I/O pins. If EJTAG is used for debug, it is important to make sure that software does not clear DDPCON<JTAGEN>.

NOTES:

3.0 INSTRUCTION SET

The PIC32MX family instruction set complies with the MIPS32 Release 2 instruction set architecture. The PIC32MX does not support the following features:

- CoreExtend instructions
- Coprocessor 1 instructions
- Coprocessor 2 instructions

Table 3-1 provides a summary of instructions implemented by the PIC32MX family core.

TABLE 3-1: PIC32MX FAMILY INSTRUCTION SET

<table>
<tr><th>Instruction</th><th>Description</th><th>Function</th></tr>
<tr><td>ADD</td><td>Integer Add</td><td>Rd = Rs + Rt</td></tr>
<tr><td>ADDI</td><td>Integer Add Immediate</td><td>Rt = Rs + Immed</td></tr>
<tr><td>ADDIU</td><td>Unsigned Integer Add Immediate</td><td>Rt = Rs $+_U$ Immed</td></tr>
<tr><td>ADDIUPC</td><td>Unsigned Integer Add Immediate to PC
(MIPS16e™ only)</td><td>Rt = PC $+_u$ Immed</td></tr>
<tr><td>ADDU</td><td>Unsigned Integer Add</td><td>Rd = Rs $+_U$ Rt</td></tr>
<tr><td>AND</td><td>Logical AND</td><td>Rd = Rs & Rt</td></tr>
<tr><td>ANDI</td><td>Logical AND Immediate</td><td>Rt = Rs & (0_{16} || Immed)</td></tr>
<tr><td>B</td><td>Unconditional Branch
(Assembler idiom for: BEQ r0, r0, offset)</td><td>PC += (int)offset</td></tr>
<tr><td>BAL</td><td>Branch and Link
(Assembler idiom for: BGEZAL r0, offset)</td><td>GPR[31> = PC + 8
PC += (int)offset</td></tr>
<tr><td>BEQ</td><td>Branch On Equal</td><td>if Rs == Rt
  PC += (int)offset</td></tr>
<tr><td>BEQL</td><td>Branch On Equal Likely</td><td>if Rs == Rt
  PC += (int)offset
else
  Ignore Next Instruction</td></tr>
<tr><td>BGEZ</td><td>Branch on Greater Than or Equal To Zero</td><td>if !Rs[31>
  PC += (int)offset</td></tr>
<tr><td>BGEZAL</td><td>Branch on Greater Than or Equal To Zero And Link</td><td>GPR[31> = PC + 8
if !Rs[31>
  PC += (int)offset</td></tr>
<tr><td>BGEZALL</td><td>Branch on Greater Than or Equal To Zero And Link Likely</td><td>GPR[31> = PC + 8
if !Rs[31>
  PC += (int)offset
else
  Ignore Next Instruction</td></tr>
<tr><td>BGEZL</td><td>Branch on Greater Than or Equal To Zero Likely</td><td>if !Rs[31>
  PC += (int)offset
else
  Ignore Next Instruction</td></tr>
<tr><td>BGTZ</td><td>Branch on Greater Than Zero</td><td>if !Rs[31> && Rs != 0
  PC += (int)offset</td></tr>
<tr><td>BGTZL</td><td>Branch on Greater Than Zero Likely</td><td>if !Rs[31> && Rs != 0
  PC += (int)offset
else
  Ignore Next Instruction</td></tr>
<tr><td>BLEZ</td><td>Branch on Less Than or Equal to Zero</td><td>if Rs[31> || Rs == 0
  PC += (int)offset</td></tr>
</table>

TABLE 3-1: PIC32MX FAMILY INSTRUCTION SET (CONTINUED)

<table>
<tr><th>Instruction</th><th>Description</th><th>Function</th></tr>
<tr><td>BLEZL</td><td>Branch on Less Than or Equal to Zero Likely</td><td>if Rs[31> || Rs == 0
PC += (int)offset
else
Ignore Next Instruction</td></tr>
<tr><td>BLTZ</td><td>Branch on Less Than Zero</td><td>if Rs[31>
PC += (int)offset</td></tr>
<tr><td>BLTZAL</td><td>Branch on Less Than Zero And Link</td><td>GPR[31> = PC + 8
if Rs[31>
PC += (int)offset</td></tr>
<tr><td>BLTZALL</td><td>Branch on Less Than Zero And Link Likely</td><td>GPR[31> = PC + 8
if Rs[31>
PC += (int)offset
else
Ignore Next Instruction</td></tr>
<tr><td>BLTZL</td><td>Branch on Less Than Zero Likely</td><td>if Rs[31>
PC += (int)offset
else
Ignore Next Instruction</td></tr>
<tr><td>BNE</td><td>Branch on Not Equal</td><td>if Rs != Rt
PC += (int)offset</td></tr>
<tr><td>BNEL</td><td>Branch on Not Equal Likely</td><td>if Rs != Rt
PC += (int)offset
else
Ignore Next Instruction</td></tr>
<tr><td>BREAK</td><td>Breakpoint</td><td>Break Exception</td></tr>
<tr><td>CLO</td><td>Count Leading Ones</td><td>Rd = NumLeadingOnes(Rs)</td></tr>
<tr><td>CLZ</td><td>Count Leading Zeroes</td><td>Rd = NumLeadingZeroes(Rs)</td></tr>
<tr><td>COP0</td><td>Coprocessor 0 Operation</td><td>See Software User's Manual</td></tr>
<tr><td>DERET</td><td>Return from Debug Exception</td><td>PC = DEPC
Exit Debug Mode</td></tr>
<tr><td>DI</td><td>Atomically Disable Interrupts</td><td>Rt = Status; Status$_{IE}$ = 0</td></tr>
<tr><td>DIV</td><td>Divide</td><td>LO = (int)Rs / (int)Rt
HI = (int)Rs % (int)Rt</td></tr>
<tr><td>DIVU</td><td>Unsigned Divide</td><td>LO = (uns)Rs / (uns)Rt
HI = (uns)Rs % (uns)Rt</td></tr>
<tr><td>EHB</td><td>Execution Hazard Barrier</td><td>Stop instruction execution until execution hazards are cleared</td></tr>
<tr><td>EI</td><td>Atomically Enable Interrupts</td><td>Rt = Status; Status$_{IE}$ = 1</td></tr>
<tr><td>ERET</td><td>Return from Exception</td><td>if SR[2>
PC = ErrorEPC
else
PC = EPC
SR[1> = 0
SR[2> = 0
LL = 0</td></tr>
<tr><td>EXT</td><td>Extract Bit Field</td><td>Rt = ExtractField(Rs, pos, size)</td></tr>
<tr><td>INS</td><td>Insert Bit Field</td><td>Rt = InsertField(Rs, Rt, pos, size)</td></tr>
<tr><td>J</td><td>Unconditional Jump</td><td>PC = PC[31:28> || offset<<2</td></tr>
</table>

TABLE 3-1: PIC32MX FAMILY INSTRUCTION SET (CONTINUED)

<table>
<tr><th>Instruction</th><th>Description</th><th>Function</th></tr>
<tr><td>JAL</td><td>Jump and Link</td><td>GPR[31> = PC + 8
PC = PC[31:28> || offset<<2</td></tr>
<tr><td>JALR</td><td>Jump and Link Register</td><td>Rd = PC + 8
PC = Rs</td></tr>
<tr><td>JALR.HB</td><td>Jump and Link Register with Hazard Barrier</td><td>Like JALR, but also clears execution and instruction hazards</td></tr>
<tr><td>JALRC</td><td>Jump and Link Register Compact – do not execute instruction in jump delay slot (MIPS16e™ only)</td><td>Rd = PC + 2
PC = Rs</td></tr>
<tr><td>JR</td><td>Jump Register</td><td>PC = Rs</td></tr>
<tr><td>JR.HB</td><td>Jump Register with Hazard Barrier</td><td>Like JR, but also clears execution and instruction hazards</td></tr>
<tr><td>JRC</td><td>Jump Register Compact – do not execute instruction in jump delay slot (MIPS16e only)</td><td>PC = Rs</td></tr>
<tr><td>LB</td><td>Load Byte</td><td>Rt = (byte)Mem[Rs+offset></td></tr>
<tr><td>LBU</td><td>Unsigned Load Byte</td><td>Rt = (ubyte))Mem[Rs+offset></td></tr>
<tr><td>LH</td><td>Load Halfword</td><td>Rt = (half)Mem[Rs+offset></td></tr>
<tr><td>LHU</td><td>Unsigned Load Halfword</td><td>Rt = (uhalf)Mem[Rs+offset></td></tr>
<tr><td>LL</td><td>Load Linked Word</td><td>Rt = Mem[Rs+offset>
LL = 1
LLAdr = Rs + offset</td></tr>
<tr><td>LUI</td><td>Load Upper Immediate</td><td>Rt = immediate << 16</td></tr>
<tr><td>LW</td><td>Load Word</td><td>Rt = Mem[Rs+offset></td></tr>
<tr><td>LWPC</td><td>Load Word, PC relative</td><td>Rt = Mem[PC+offset></td></tr>
<tr><td>LWL</td><td>Load Word Left</td><td>See Architecture Reference Manual</td></tr>
<tr><td>LWR</td><td>Load Word Right</td><td>See Architecture Reference Manual</td></tr>
<tr><td>MADD</td><td>Multiply-Add</td><td>HI | LO += (int)Rs * (int)Rt</td></tr>
<tr><td>MADDU</td><td>Multiply-Add Unsigned</td><td>HI | LO += (uns)Rs * (uns)Rt</td></tr>
<tr><td>MFC0</td><td>Move From Coprocessor 0</td><td>Rt = CPR[0, Rd, sel></td></tr>
<tr><td>MFHI</td><td>Move From HI</td><td>Rd = HI</td></tr>
<tr><td>MFLO</td><td>Move From LO</td><td>Rd = LO</td></tr>
<tr><td>MOVN</td><td>Move Conditional on Not Zero</td><td>if Rt ¼ 0 then
Rd = Rs</td></tr>
<tr><td>MOVZ</td><td>Move Conditional on Zero</td><td>if Rt = 0 then
Rd = Rs</td></tr>
<tr><td>MSUB</td><td>Multiply-Subtract</td><td>HI | LO -= (int)Rs * (int)Rt</td></tr>
<tr><td>MSUBU</td><td>Multiply-Subtract Unsigned</td><td>HI | LO -= (uns)Rs * (uns)Rt</td></tr>
<tr><td>MTC0</td><td>Move To Coprocessor 0</td><td>CPR[0, n, Sel> = Rt</td></tr>
<tr><td>MTHI</td><td>Move To HI</td><td>HI = Rs</td></tr>
<tr><td>MTLO</td><td>Move To LO</td><td>LO = Rs</td></tr>
<tr><td>MUL</td><td>Multiply with register write</td><td>HI | LO =Unpredictable
Rd = ((int)Rs * (int)Rt)$_{31..0}$</td></tr>
<tr><td>MULT</td><td>Integer Multiply</td><td>HI | LO = (int)Rs * (int)Rd</td></tr>
<tr><td>MULTU</td><td>Unsigned Multiply</td><td>HI | LO = (uns)Rs * (uns)Rd</td></tr>
<tr><td>NOP</td><td>No Operation
(Assembler idiom for: SLL r0, r0, r0)</td><td></td></tr>
<tr><td>NOR</td><td>Logical NOR</td><td>Rd = ~(Rs | Rt)</td></tr>
<tr><td>OR</td><td>Logical OR</td><td>Rd = Rs | Rt</td></tr>
</table>

TABLE 3-1: PIC32MX FAMILY INSTRUCTION SET (CONTINUED)

Instruction	Description	Function
ORI	Logical OR Immediate	Rt = Rs | Immed
RDHWR	Read Hardware Register	Allows unprivileged access to registers enabled by HWREna register
RDPGPR	Read GPR from Previous Shadow Set	Rt = SGPR[SRSCtl$_{PSS}$, Rd>
RESTORE	Restore registers and deallocate stack frame (MIPS16e™ only)	See Architecture Reference Manual
ROTR	Rotate Word Right	Rd = $Rt_{sa-1..0}$ || $Rt_{31..sa}$
ROTRV	Rotate Word Right Variable	Rd = $Rt_{Rs-1..0}$ || $Rt_{31..Rs}$
SAVE	Save registers and allocate stack frame (MIPS16e only)	See Architecture Reference Manual
SB	Store Byte	(byte)Mem[Rs+offset> = Rt
SC	Store Conditional Word	if LL = 1 mem[Rs+offset> = Rt Rt = LL
SDBBP	Software Debug Break Point	Trap to SW Debug Handler
SEB	Sign-Extend Byte	Rd = (byte)Rs
SEH	Sign-Extend Half	Rd = (half)Rs
SH	Store Half	(half)Mem[Rs+offset> = Rt
SLL	Shift Left Logical	Rd = Rt << sa
SLLV	Shift Left Logical Variable	Rd = Rt << Rs[4:0>

SLT	Set on Less Than	if (int)Rs < (int)Rt
		  Rd = 1
		else
		  Rd = 0

SLTI	Set on Less Than Immediate	if (int)Rs < (int)Immed
		  Rt = 1
		else
		  Rt = 0

SLTIU	Set on Less Than Immediate Unsigned	if (uns)Rs < (uns)Immed
		  Rt = 1
		else
		  Rt = 0

SLTU	Set on Less Than Unsigned	if (uns)Rs < (uns)Immed
		  Rd = 1
		else
		  Rd = 0

SRA	Shift Right Arithmetic	Rd = (int)Rt >> sa
SRAV	Shift Right Arithmetic Variable	Rd = (int)Rt >> Rs[4:0>
SRL	Shift Right Logical	Rd = (uns)Rt >> sa
SRLV	Shift Right Logical Variable	Rd = (uns)Rt >> Rs[4:0>
SSNOP	Superscalar Inhibit No Operation	NOP
SUB	Integer Subtract	Rt = (int)Rs - (int)Rd
SUBU	Unsigned Subtract	Rt = (uns)Rs - (uns)Rd
SW	Store Word	Mem[Rs+offset> = Rt
SWL	Store Word Left	See Architecture Reference Manual
SWR	Store Word Right	See Architecture Reference Manual
SYNC	Synchronize	See Software User's Manual
SYSCALL	System Call	SystemCallException

TABLE 3-1: PIC32MX FAMILY INSTRUCTION SET (CONTINUED)

Instruction	Description	Function
TEQ	Trap if Equal	if Rs == Rt TrapException
TEQI	Trap if Equal Immediate	if Rs == (int)Immed TrapException
TGE	Trap if Greater Than or Equal	if (int)Rs >= (int)Rt TrapException
TGEI	Trap if Greater Than or Equal Immediate	if (int)Rs >= (int)Immed TrapException
TGEIU	Trap if Greater Than or Equal Immediate Unsigned	if (uns)Rs >= (uns)Immed TrapException
TGEU	Trap if Greater Than or Equal Unsigned	if (uns)Rs >= (uns)Rt TrapException
TLT	Trap if Less Than	if (int)Rs < (int)Rt TrapException
TLTI	Trap if Less Than Immediate	if (int)Rs < (int)Immed TrapException
TLTIU	Trap if Less Than Immediate Unsigned	if (uns)Rs < (uns)Immed TrapException
TLTU	Trap if Less Than Unsigned	if (uns)Rs < (uns)Rt TrapException
TNE	Trap if Not Equal	if Rs != Rt TrapException
TNEI	Trap if Not Equal Immediate	if Rs != (int)Immed TrapException
WAIT	Wait for Interrupts	Stall until interrupt occurs
WRPGPR	Write to GPR in Previous Shadow Set	$SGPR[SRSCtl_{PSS}$, Rd> = Rt
WSBH	Word Swap Bytes Within Halfwords	Rd = $Rt_{23..16}$ || $Rt_{31..24}$ || $Rt_{7..0}$ || $Rt_{15..8}$
XOR	Exclusive OR	Rd = Rs ^ Rt
XORI	Exclusive OR Immediate	Rt = Rs ^ (uns)Immed
ZEB	Zero-extend byte (MIPS16e™ only)	Rt = (ubyte) Rs
ZEH	Zero-extend half (MIPS16e only)	Rt = (uhalf) Rs

NOTES:

4.0 PREFETCH CACHE

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

The Prefetch cache increases performance for applications executing out of the cacheable program flash memory region by implementing instruction caching, data caching, and instruction prefetching.

4.1 Features

- 16 Fully Associative Lockable Cache Lines
- 16-byte Cache Lines
- Up to 4 Cache Lines allocated to Data
- 2 Cache Lines with Address Mask to hold repeated instructions
- Pseudo LRU replacement policy
- All Cache Lines are software writable
- 16-byte parallel memory fetch
- Predictive Instruction Prefetch

FIGURE 4-1: PREFETCH MODULE BLOCK DIAGRAM

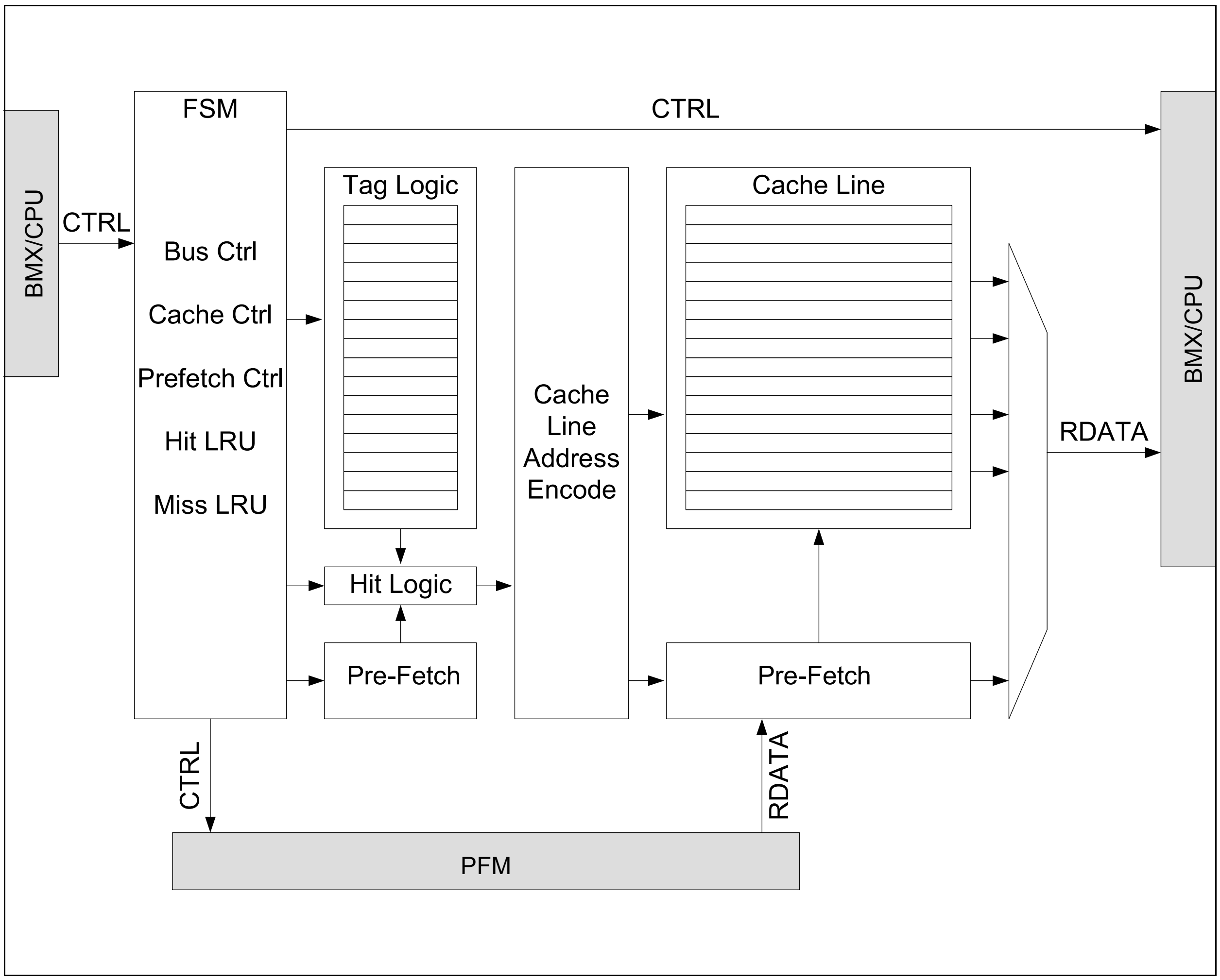


TABLE 4-1: PREFETCH SFR SUMMARY

Virtual Address	Name		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
BF88_4000	CHECON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	CHECOH
		15:8	—	—	—	—	—	—	DCSZ<1:0>	
		7:0	—	—	PREFEN<1:0>		—	PFMWS<2:0>		
BF88_4004	CHECONCLR	31:0	Clears selected bits in CHECON, read yields undefined value							
BF88_4008	CHECONSET	31:0	Sets selected bits in CHECON, read yields undefined value							
BF88_400C	CHECONINV	31:0	Inverts selected bits in CHECON, read yields undefined value							
BF88_4010	CHEACC	31:24	CHEWEN	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	—	—	—	—	CHEIDX<3:0>			
BF88_4014	CHEACCCLR	31:0	Clears selected bits in CHEACC, read yields undefined value							
BF88_4018	CHEACCSET	31:0	Sets selected bits in CHEACC, read yields undefined value							
BF88_401C	CHEACCINV	31:0	Inverts selected bits CHEACC, read yields undefined value							
BF88_4020	CHETAG	31:24	LTAGBOOT	—	—	—	—	—	—	—
		23:16	LTAG<23:16>							
		15:8	LTAG<15:8>							
		7:0	LTAG<7:4>				LVALID	LLOCK	LTYPE	—
BF88_4024	CHETAGCLR	31:0	Clears selected bits in CHETAG, read yields undefined value							
BF88_4028	CHETAGSET	31:0	Sets selected bits in CHETAG, read yields undefined value							
BF88_402C	CHETAGINV	31:0	Inverts selected bits CHETAG, read yields undefined value							
BF88_4030	CHEMSK	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	LMASK<15:8>							
		7:0	LMASK<7:5>			—	—	—	—	—
BF88_4034	CHEMSKCLR	31:0	Clears selected bits in CHEMSK, read yields undefined value							
BF88_4038	CHEMSKSET	31:0	Sets selected bits in CHEMSK, read yields undefined value							
BF88_403C	CHEMSKINV	31:0	Inverts selected bits CHEMSK, read yields undefined value							
BF88_4040	CHEW0	31:24	CHEW0<31:24>							
		23:16	CHEW0<23:16>							
		15:8	CHEW0<15:8>							
		7:0	CHEW0<7:0>							
BF88_4050	CHEW1	31:24	CHEW1<31:24>							
		23:16	CHEW1<23:16>							
		15:8	CHEW1<15:8>							
		7:0	CHEW1<7:0>							
BF88_4060	CHEW2	31:24	CHEW2<31:24>							
		23:16	CHEW2<23:16>							
		15:8	CHEW2<15:8>							
		7:0	CHEW2<7:0>							
BF88_4070	CHEW3	31:24	CHEW3<31:24>							
		23:16	CHEW3<23:16>							
		15:8	CHEW3<15:8>							
		7:0	CHEW3<7:0>							
BF88_4080	CHELRU	31:24	—	—	—	—	—	—	—	CHELRU<24>
		23:16	CHELRU<23:16>							
		15:8	CHELRU<15:8>							
		7:0	CHELRU<7:0>>							
BF88_4090	CHEHIT	31:24	CHEHIT<31:24>							
		23:16	CHEHIT<23:16>							
		15:8	CHEHIT<15:8>							
		7:0	CHENIT<7:0>							

TABLE 4-1: PREFETCH SFR SUMMARY (CONTINUED)

Virtual Address	Name		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
BF88_40A0	CHEMIS	31:24	CHEMIS<31:24>							
		23:16	CHEMIS<23:16>							
		15:8	CHEMIS<15:8>							
		7:0	CHEMIS<7:0>							
BF88_40C0	PFABT	31:24	PFABT<31:24>							
		23:16	PFABT<23:16>							
		15:8	PFABT<15:8>							
		7:0	PFABT<7:0>							

4.2 Prefetch Registers

REGISTER 4-1: CHECON: CACHE CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	CHECOH
bit 23							bit 16

U-0	U-0	r-0	r-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	DCSZ<1:0>	
bit 15							bit 8

U-0	U-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
—	—	PREFEN<1:0>		—	PFMWS<2:0>		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-17 **Unimplemented:** Read as ‘0’

bit 16 **CHECOH:** Cache Coherency setting on a PFM Program Cycle bit
1 = Invalidate all data and instruction lines
0 = Invalidate all data lnes and instruction lines that are not locked

bit 15-14 **Unimplemented:** Read as ‘0’

bit 13-12 **Reserved:** Must be written with zeros

bit 11-10 **Unimplemented:** Read as ‘0’

bit 9-8 **DCSZ<1:0>:** Data Cache Size in Lines bits
11 = Enable data caching with a size of 4 Lines
10 = Enable data caching with a size of 2 Lines
01 = Enable data caching with a size of 1 Line
00 = Disable data caching
Changing this field causes all lines to be re-initialized to the “invalid” state.

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5-4 **PREFEN<1:0>:** Predictive Prefetch Cache Enable bits
11 = Enable predictive prefetch cache for both cacheable and non-cacheable regions
10 = Enable predictive prefetch cache for non-cacheable regions only
01 = Enable predictive prefetch cache for cacheable regions only
00 = Disable predictive prefetch cache

bit 3 **Unimplemented:** Read as ‘0’

REGISTER 4-1: CHECON: CACHE CONTROL REGISTER (CONTINUED)

bit 2-0 **PFMWS<2:0>:** PFM Access Time Defined in terms of SYSLK Wait states bits

111 = Seven Wait states
110 = Six Wait states
101 = Five Wait state
100 = Four Wait states
011 = Three Wait states
010 = Two Wait states
001 = One Wait state
000 = Zero Wait states

REGISTER 4-2: CHEACC: CACHE ACCESS

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
CHEWEN	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	CHEIDX<3:0>			
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31 **CHEWEN:** Cache Access Enable bits for registers CHETAG, CHEMSK, CHEW0, CHEW1, CHEW2, and CHEW3

1 = The cache line selected by CHEIDX is writable
0 = The cache line selected by CHEIDX is not writable

bit 30-4 **Unimplemented:** Read as ‘0’

bit 3-0 **CHEIDX<3:0>:** Cache Line Index bits

The value selects the cache line for reading or writing.

REGISTER 4-3: CHETAG[(1)]: CACHE TAG REGISTER

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
LTAGBOOT	—	—	—	—	—	—	—
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
LTAG<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
LTAG<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-0	R/W-0	R/W-1	r-0
LTAG<7:4>				LVALID	LLOCK	LTYPE	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31 **LTAGBOOT:** Line TAG Address Boot
1 = The line is in the 0x1D000000 (physical) area of memory
0 = The line is in the 0x1FC00000 (physical) area of memory

bit 30-24 **Unimplemented:** Read as ‘0’

bit 23-4 **LTAG<23:4>:** Line TAG Address bits
LTAG bits are compared against physical address <23:4> to determine a hit. Because its address range and position of Flash in kernel space and user space, the LTAG Flash address is identical for virtual addresses, (system) physical addresses, and Flash physical addresses.

bit 3 **LVALID:** Line Valid bit
1 = The line is valid and is compared to the physical address for hit detection
0 = The line is not valid and is not compared to the physical address for hit detection

bit 2 **LLOCK:** Line Lock bit
1 = The line is locked and will not be replaced
0 = The line is not locked and can be replaced

bit 1 **LTYPE:** Line Type bit
1 = The line caches instruction words
0 = The line caches data words

bit 0 **Reserved:**

Note 1: The TAG and Status of the Line pointed to by CHEIDX (CHEACC<3:0>).

REGISTER 4-4: CHEMSK[1]: CACHE TAG MASK REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
LMASK<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
LMASK<7:5>			—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-5 **LMASK<15:5>:** Line Mask bits

1 = Enables mask logic to force a match on the corresponding bit position in LTAG (CHETAG<23:4>) and the physical address.

0 = Only writeable for values of CHEIDX (CHEACC<3:0>) equal to OxOA and OxOB. Disables mask logic.

bit 4-0 **Unimplemented:** Read as ‘0’

Note 1: The TAG Mask of the Line pointed to by CHEIDX (CHEACC<3:07>).

REGISTER 4-5: CHEW0: CACHE WORD 0

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW0<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW0<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW0<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW0<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **CHEW0<31:0>:** Word 0 of the cache line selected by CHEACC.CHEIDX
Readable only if the device is not code-protected.

REGISTER 4-6: CHEW1: CACHE WORD 1

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW1<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW1<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW1<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW1<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **CHEW1<31:0>:** Word 1 of the cache line selected by CHEACC.CHEIDX
Readable only if the device is not code-protected.

 Advance Information

REGISTER 4-7: CHEW2 CACHE WORD 2

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW2<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW2<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW2<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW2<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **CHEW2<31:0>:** Word 2 of the cache line selected by CHEACC.CHEIDX
Readable only if the device is not code-protected.

REGISTER 4-8: CHEW3(1): CACHE WORD 3

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW3<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW3<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW3<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW3<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **CHEW3<31:0>:** Word 3 of the cache line selected by CHEACC.CHEIDX
Readable only if the device is not code-protected.

Note 1: This register is a window into the cache data array and is readable only if the device is not code-protected.

REGISTER 4-9: CHELRU: CACHE LRU REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R-0
—	—	—	—	—	—	—	CHELRU<24>
bit 31							bit 24
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHELRU<23-16>							
bit 23							bit 16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHELRU<15-8>							
bit 15							bit 8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHELRU<7-0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-25 **Unimplemented:** Read as ‘0’

bit 24-0 **CHELRU<24:0>:** Cache Least Recently Used State Encoding bits
CHELRU indicates the Pseudo-LRU state of the cache.

REGISTER 4-10: CHEHIT: CACHE HIT STATISTICS REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEHIT<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEHIT<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEHIT<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEHIT<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **CHEHIT<31:0>:** Cache Hit Count bits

Incremented each time the processor issues an instruction fetch or load that hits the prefetch cache from a cacheable region. Non-cacheable accesses do not modify this value.

REGISTER 4-11: CHEMIS: CACHE MISS STATISTICS REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEMIS<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEMIS<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEMIS<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEMIS<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **CHEMIS<31:0>:** Cache Miss Count bits

Incremented each time the processor issues an instruction fetch from a cacheable region that misses the prefetch cache. Non-cacheable accesses do not modify this value.

REGISTER 4-12: PFABT: PREFETCH CACHE ABORT STATISTICS REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
PFABT<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
PFABT<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
PFABT<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
PFABT<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **PFABT<31:0>:** Prefab Abort Count bits

Incremented each time an automatic prefetch cache is aborted due to a non-sequential instruction fetch, load or store.

 Advance Information

4.3 Prefetch Configuration

The CHECON register controls the configurations available for instruction and data caching of Program Flash Memory.

In addition to normal instruction caching, the prefetch cache has the ability to cache lines specifically for Flash Memory data.

The CHECON.DCSZ field controls the number of lines allocated to program data caching. Table 4-2 shows the cache line relationship for values of DCSZ. The data caching capability is for read only data such as constants, parameters, table data, etc., that are not modified.

TABLE 4-2: PROGRAM DATA CACHE

DCSZ<1:0>	Lines Allocated to Program Data
00	None
01	Cache Line Number 15
10	Cache Line Number 14 and 15
11	Cache Line Number 12 through 15

The CHECON.PREFEN field controls predictive prefetching, which allows the prefetch module to speculatively fetch the next 16-byte aligned set of instructions.

The prefetch module loads data into the data array only on accesses to cacheable regions (CCA bits = 3).

EXAMPLE 4-1: EXAMPLE CODE: INITIALIZATION CODE FOR PREFETCH MODULE

```
/* Prefetch Cache Initialization */

    tmp = _CP0_GET_CONFIG(); // read CONFIG register
    tmp |= 1; // kseg0 cacheable
    _CP0_SET_CONFIG(tmp); // write CONFIG register

    CHECON = (1<<4) | 3; // 3 wait-states,
                         // Prefetching enabled for cached memory
```

4.3.1 LINE LOCKING

Each line in the cache can be locked to hold its contents. A line is locked if both LVALID=1 and LLOCK=1. If LVALID=0 and LLOCK=1, the prefetch module issues a preload request (see below). Locking cache lines may reduce the performance of general program flow. However, if one or two functions calls consume a significant percent of overall processing, locking their address can provide improved performance.

Though any number of lines can be locked, the cache works most efficiently when locking either 1 or 4 lines. If locking 4 lines, choose lines whose line number divide by 4 have the same quotient. This locks an entire LRU group which benefits the LRU algorithm. For example, lines 8, 9, A, and B each have a quotient of 2 when divided by 4.

If cache lines are manually filled, it is recommended that the following sequence be used.

1. Choose a cache line to fill.
2. Set the Lock and Valid bits of the cache line by writing to CHETAG.
3. Write to each word of the cache line by writing to CHEW0, CHEW1, CHEW2, and CHEW3.

EXAMPLE 4-2: EXAMPLE CODE: LOCKING A LINE IN PREFETCH MODULE

```
#define LOCKED_LINE_NUM 3

/* lock first line of func1() in cache */

CHEACC = (1<<31) | LOCKED_LINE_NUM;
tmp = (unsigned long)func1;
ltagboot = (tmp & 0x00c00000) ? 0 : 1; // 0x9fc????? or 0x9d0?????
CHETAG = (ltagboot<<31) | (tmp & 0x0007fff0) | 6; // locked and invalid
```

4.3.2 PRELOAD BEHAVIOR

Application code can direct the prefetch module to preform a preload of a cache line and lock it with instructions or data from the flash. The Preload function uses the CHEACC.CHEIDX register field to select the cache line into which the load is directed. Setting CHEACC.CHEWEN to a '1' enables writes to the CHETAG register.

Writing CHETAG.LVALID = 0 and CHETAG.LLOCK = 1 causes a preload request to the prefetch module. The controller acknowledges the request in the cycle after the write and if possible stops any outstanding flash access and stalls any CPU load from the cache or Flash.

When it has finished or stalled the previous transaction, it initiates a flash read to fetch the instructions or data requested using the address in CHETAG.LTAG. After the programmed number of wait states as defined by CHECON.PFMWS, the controller updates the data array with the values read from flash. On the update it sets CHETAG.LVALID = 1. The LRU state of the line is not affected.

Once the controller finishes updating the cache, it allows CPU requests to complete. If this request misses the cache, the controller initiates a flash read which incurs the full flash access time.

4.3.3 ADDRESS MASK

Cache lines 10 and 11 allow masking of the CPU address and tag address to force a match on corresponding bits. The CHEMSK.LMASK field is set up to compliment the interrupt vector spacing field in the CPU. This feature allows boot code to lock the first four instruction of a vector in the cache. If all vectors contain identical instructions in their first four locations, then setting the CHEMSK.LMASK to match the vector spacing and the LTAG to match the vector base address causes all the vector addresses to hit the cache. The prefetch module responds with zero wait states and immediately initiates a fetch of the next set of four instruction for the requesting vector if prefetch is enabled.

Using CHEMSK.LMASK is restricted to aligned address ranges. Its size allows for a max range of 32KB and a minimum spacing of 32B. Using the two lines, in conjunction provides the ability to have different ranges and different spacing.

Setting up the address mask such that more than one line will match an address causes undefined results. Therefore, it is highly recommended to set up masking before entering cacheable code.

EXAMPLE 4-3: EXAMPLE CODE: DUPLICATION OF CODE USING MASK REGISTERS

```
#define INT_LINE_NUM 10

    CHEACC = (1<<31) | INT_LINE_NUM;
    tmp = (unsigned long)intbase;
    ltagboot = (tmp & 0x00c00000) ? 0 : 1; // 0x9fc????? or 0x9d0?????
CHETAG = (ltagboot<<31) | (tmp & 0x0007fff0) | 6; // locked and invalid
CHEMSK = 0xe0; // first 4 instructions of intbase() replicated 8 times on 32-byte boundaries
```


4.3.4 PREDICTIVE PREFETCH BEHAVIOR

When configured for predictive prefetch on cacheable addresses, the module predicts the next line address and returns it into the pseudo LRU line of the cache. If enabled, the prefetch function starts predicting based on the first CPU instruction fetch. When the first line is placed in the cache, the module simply increments the address to the next 16-byte aligned address and starts a flash access. When running linear code (i.e. no jumps), the flash returns the next set of instructions into the prefetch buffer on or before all instructions can be executed from the previous line.

If at any time during a predicted flash access, a new CPU address does not match the predicted one, the flash access will be changed to the correct address. This behavior does not cause the CPU access to take any longer than without prediction.

If an access that misses the cache hits the prefetch buffer, the instructions are placed in the pseudo LRU line along with its address tag. The pseudo LRU value is marked as the most recently used line and other lines are updated accordingly. If an access misses both the cache and the prefetch buffer, the access passes to the flash and those returning instructions are placed in the pseudo LRU line.

When configured for predictive prefetch on non-cacheable addresses, the controller only uses the prefetch buffer. The LRU cache line is not updated for hits or fills so the cache remains intact. For linear code, enabling predictive prefetch for non-cacheable addresses allows the CPU to fetch instructions in zero wait states.

It is not useful to use non-cacheable predictive prefetching when accesses to the flash are set for zero wait states. The controller holds prefetched instructions on the output of the flash for up to 3 clock cycles (while the CPU is fetching from the buffer). This consumes more power without any benefit for zero wait state flash accesses.

Predictive data prefetching is not supported. However, a data access in the middle of a predictive instruction fetch causes the prefetch controller to stop the flash access for the instruction fetch and to start the data load from flash. The predictive prefetch does not resume, but instead waits for another instruction fetch. At which time, it either fills the buffer because of a miss, or starts a prefetch because of a hit.

4.3.5 COHERENCY SUPPORT

It is not possible to execute out of cache while programming the flash memory. The flash controller stalls the cache during the programming sequence. Therefore, user code that initiates a programming sequence must not be located in a cacheable address region.

If CHECON.CHECOH = 1, then coherency is strictly supported by invalidating, unlocking, and clearing masks for all lines whenever the Flash Program Memory is written or programmed.

If CHECON.CHECOH = 0, then only lines that are not locked are forced invalid. Lines that are locked are retained.

4.4 Prefetch Module Interrupts and Exceptions

The prefetch module does not generate any interrupts.

Exceptions can occur if cache lines are marked as valid manually by writing to individual CHETAG registers then executing code that hits one of these lines containing invalid instructions. Also manually placing data into an un-locked cache line may cause a coherency problem from an eviction due to a cache miss in the middle of the loading algorithm.

4.4.1 I/O PIN CONFIGURATION

The prefetch module does not use any external pins.

NOTES:

5.0 DIRECT MEMORY ACCESS (DMA) CONTROLLER

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

The PIC32MX Direct Memory Access (DMA) controller is a bus master module useful for data transfers between different devices without the CPU intervention. The source and destination of a DMA transfer can be any of the memory mapped modules existent in the PIC32MX (such as Peripheral Bus (PBUS) devices: SPI, UART, I^{2}C™, etc.) or memory itself.

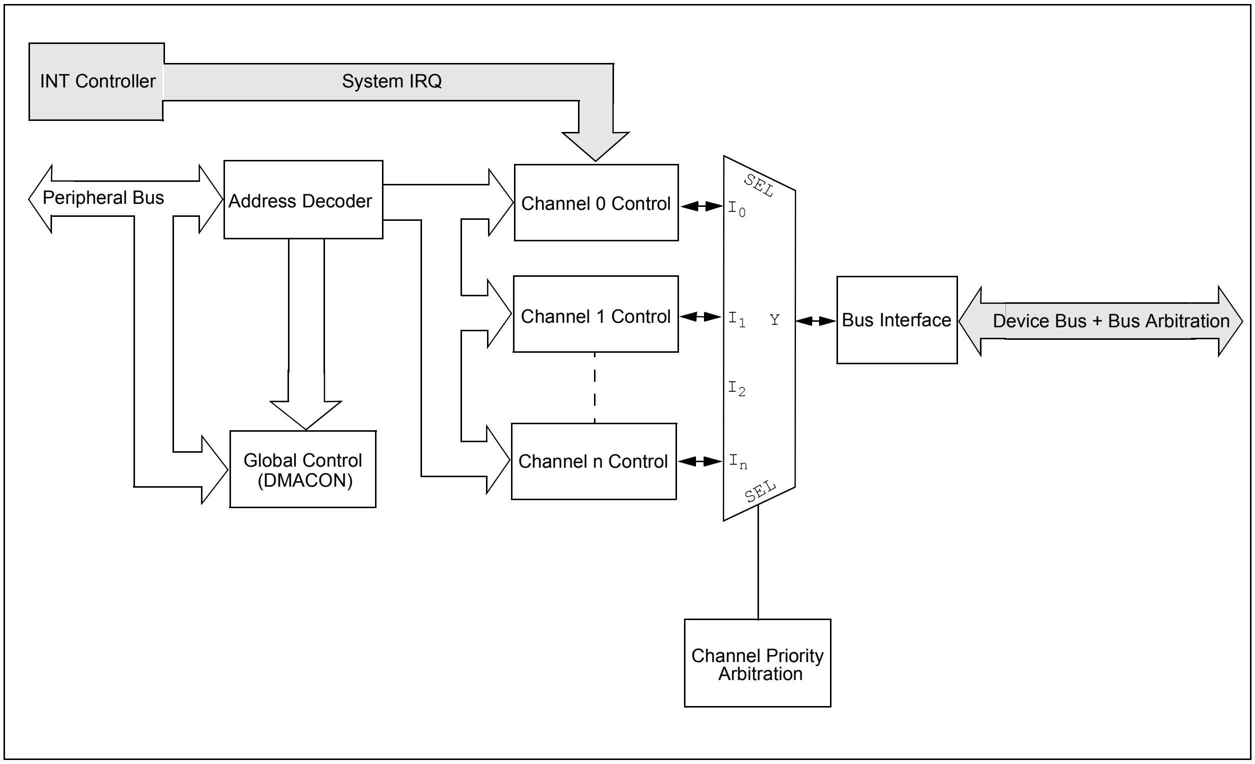
Following are some of the key features of the DMA controller module:

- Four Identical Channels, each featuring:
  - Auto-Increment Source and Destination Address registers
  - Source and Destination Pointers
- Automatic Word-Size Detection:
  - Transfer granularity down to byte level
  - Bytes need not be word-aligned at source and destination
- Fixed Priority Channel Arbitration
- Flexible DMA Channel Operating modes:
  - Manual (software) or automatic (interrupt) DMA requests
  - One-Shot or Auto-Repeat Block Transfer modes
  - Channel-to-channel chaining
- Flexible DMA Requests:
  - A DMA request can be selected from any of the peripheral interrupt sources
  - Each channel can select any (appropriate) observable interrupt as its DMA request source
  - A DMA transfer abort can be selected from any of the peripheral interrupt sources
  - Pattern (data) match transfer termination
- Multiple DMA Channel Status Interrupts:
  - DMA channel block transfer complete
  - Source empty or half empty
  - Destination full or half full
  - DMA transfer aborted due to an external event
  - Invalid DMA address generated
- DMA Debug Support Features:
  - Most recent address accessed by a DMA channel
  - Most recent DMA channel to transfer data
- CRC Generation Module:
  - CRC module can be assigned to any of the available channels
  - CRC module is highly configurable
- Extended Addressing mode:
  - Extended Addressing mode allows large memory to memory copies

TABLE 5-1: DMA CONTROLLER FEATURES

Available DMA Modes	Transfer Length	Unaligned Transfers	Different Source and Destination Sizes	Memory to Memory Transfers	Memory to Peripheral Transfers	Channel Auto-Enable	Events Start/Stop	Pattern Match Detection	Channel Chaining	CRC Calculation
Normal Addressing Mode	<= 256B	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Extended Addressing Mode	<= 64 KB	No	No	Yes	No	Yes	Yes	Yes	Yes	Yes

FIGURE 5-1: DMA CONTROLLER BLOCK DIAGRAM



5.1 DMA Controller Registers

TABLE 5-2: DMA GLOBAL SFR SUMMARY

Virtual Address	Name		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
BF88_3000	DMACON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	SUSPEND	—	—	—	—
		7:0	—	—	—	—	—	—		—
BF88_3004	DMACONCLR	31:0	Write clears selected bits in DMACON, read yields undefined value							
BF88_3008	DMACONSET	31:0	Write sets selected bits in DMACON, read yields undefined value							
BF88_300C	DMACONINV	31:0	Write inverts selected bits in DMACON, read yields undefined value							
BF88_3010	DMASTAT	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	—	—	—	—	RDWR	—	DMACH<1:0>	
BF88_3020	DMAADDR	31:24	DMAADDR<31:24>							
		23:16	DMAADDR<23:16>							
		15:8	DMAADDR<15:8>							
		7:0	DMAADDR<7:0>							

TABLE 5-3: DMA CRC SFR SUMMARY

Virtual Address	Name		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
BF88_3030	DCRCCON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	PLEN<4:0>				
		7:0	CRCEN	CRCAPP	—	—	—	—	CRCCH<1:0>	
BF88_3034	DCRCCONCLR	31:0	Write clears selected bits in DCRCCON, read yields undefined value							
BF88_3038	DCRCCONSET	31:0	Write sets selected bits in DCRCCON, read yields undefined value							
BF88_303C	DCRCCONINV	31:0	Write inverts selected bits in DCRCCON, read yields undefined value							
BF88_3040	DCRCDATA	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	DCRCDATA<15:8>							
		7:0	DCRCDATA<7:0>							
BF88_3044	DCRCDATACLR	31:0	Write clears selected bits in DCRCDATA, read yields undefined value							
BF88_3048	DCRCDATASET	31:0	Write sets selected bits in DCRCDATA, read yields undefined value							
BF88_304C	DCRCDATAINV	31:0	Write inverts selected bits in DCRCDATA, read yields undefined value							
BF88_3050	DCRCXOR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	DCRCXOR<15:8>							
		7:0	DCRCXOR<7:0>							
BF88_3054	DCRCXORCLR	31:0	Write clears selected bits in DCRCXOR, read yields undefined value							
BF88_3058	DCRCXORSET	31:0	Write sets selected bits in DCRCXOR, read yields undefined value							
BF88_305C	DCRCXORINV	31:0	Write inverts selected bits in DCRCXOR, read yields undefined value							

TABLE 5-4: DMA CHANNEL 0 SFR SUMMARY

Virtual Address(1)	Name		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
BF88_3060	DCH0CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	CHCHNS
		7:0	CHEN	CHAED	CHCHN	CHAEN	CHXM	CHEDET	CHPRI<1:0>	
BF88_3064	DCH0CONCLR	31:0	Write clears selected bits in DCH0CON, read yields undefined value							
BF88_3068	DCH0CONSET	31:0	Write sets selected bits in DCH0CON, read yields undefined value							
BF88_306C	DCH0CONINV	31:0	Write inverts selected bits in DCH0CON, read yields undefined value							
BF88_3070	DCH0ECON	31:24	—	—	—	—	—	—	—	—
		23:16	CHAIRQ<7:0>							
		15:8	CHSIRQ<7:0>							
		7:0	CFORCE	CABORT	PATEN	SIRQEN	AIRQEN	—	—	—
BF88_3074	DCH0ECONCLR	31:0	Write clears selected bits in DCH0ECON, read yields undefined value							
BF88_3078	DCH0ECONSET	31:0	Write sets selected bits in DCH0ECON, read yields undefined value							
BF88_307C	DCH0ECONINV	31:0	Write inverts selected bits in DCH0ECON, read yields undefined value							
BF88_3080	DCH0INT	31:24	—	—	—	—	—	—	—	—
		23:16	CHSDIE	CHSHIE	CHDDIE	CHDHIE	CHBCIE	CHCCIE	CHTAIE	CHERIE
		15:8	—	—	—	—	—	—	—	—
		7:0	CHSDIF	CHSHIF	CHDDIF	CHDHIF	CHBCIF	CHCCIF	CHTAIF	CHERIF
BF88_3084	DCH0INTCLR	31:0	Write clears selected bits in DCH0INT, read yields undefined value							
BF88_3088	DCH0INTSET	31:0	Write sets selected bits in DCH0INT, read yields undefined value							
BF88_308C	DCH0INTINV	31:0	Write inverts selected bits in DCH0INT, read yields undefined value							
BF88_3090	DCH0SSA	31:24	CHSSA<31:24>							
		23:16	CHSSA<23:16>							
		15:8	CHSSA<15:8>							
		7:0	CHSSA<7:0>							
BF88_3094	DCH0SSACLR	31:0	Write clears selected bits in DCH0SSA, read yields undefined value							
BF88_3098	DCH0SSASET	31:0	Write sets selected bits in DCH0SSA, read yields undefined value							
BF88_309C	DCH0SSAINV	31:0	Write inverts selected bits in DCH0SSA, read yields undefined value							

Note 1: The starting address of the registers for DMA channel n is 0xbf883060 + 0xc0*n.
2: These bits are relevant in Extended Addressing mode only.

TABLE 5-4: DMA CHANNEL 0 SFR SUMMARY (CONTINUED)

Virtual Address[(1)]	Name		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
BF88_30A0	DCH0DSA	31:24	CHDSA<31:24>							
		23:16	CHDSA<23:16>							
		15:8	CHDSA<15:8>							
		7:0	CHDSA<7:0>							
BF88_30A4	DCH0DSACLR	31:0	Write clears selected bits in DCH0DSA, read yields undefined value							
BF88_30A8	DCH0DSASET	31:0	Write sets selected bits in DCH0DSA, read yields undefined value							
BF88_30AC	DCH0DSAINV	31:0	Write inverts selected bits in DCH0DSA, read yields undefined value							
BF88_30B0	DCH0SSIZ	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHSSIZ<7:0>							
BF88_30B4	DCH0SSIZCLR	31:0	Write clears selected bits in DCH0SSIZ, read yields undefined value							
BF88_30B8	DCH0SSIZSET	31:0	Write sets selected bits in DCH0SSIZ, read yields undefined value							
BF88_30BC	DCH0SSIZINV	31:0	Write inverts selected bits in DCH0SSIZ, read yields undefined value							
BF88_30C0	DCH0DSIZ	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHDSIZ<15:8>[(2)]							
		7:0	CHDSIZ<7:0>							
BF88_30C4	DCH0DSIZCLR	31:0	Write clears selected bits in DCH0DSIZ, read yields undefined value							
BF88_30C8	DCH0DSIZSET	31:0	Write sets selected bits in DCH0DSIZ, read yields undefined value							
BF88_30CC	DCH0DSIZINV	31:0	Write inverts selected bits in DCH0DSIZ, read yields undefined value							
BF88_30D0	DCH0SPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHSPTR<7:0>							
BF88_30E0	DCH0DPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHDPTR<15:8>[(2)]							
		7:0	CHDPTR<7:0>							
BF88_30F0	DCH0CSIZ	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHCSIZ<7:0>							
BF88_30F4	DCH0CSIZCLR	31:0	Write clears selected bits in DCH0CSIZ, read yields undefined value							
BF88_30F8	DCH0CSIZSET	31:0	Write sets selected bits in DCH0CSIZ, read yields undefined value							
BF88_30FC	DCH0CSIZINV	31:0	Write inverts selected bits in DCH0CSIZ, read yields undefined value							
BF88_3100	DCH0CPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHCPTR<7:0>							
BF88_3110	DCH0DAT	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHPDAT<7:0>							
BF88_3114	DCH0DATCLR	31:0	Write clears selected bits in DCH0DAT, read yields undefined value							
BF88_3118	DCH0DATSET	31:0	Write sets selected bits in DCH0DAT, read yields undefined value							
BF88_311C	DCH0DATINV	31:0	Write inverts selected bits in DCH0DAT, read yields undefined value							

Note 1: The starting address of the registers for DMA channel n is 0xbf883060 + 0xc0*n.

2: These bits are relevant in Extended Addressing mode only.

TABLE 5-5: DMA CHANNEL 0 INTERRUPT REGISTER SUMMARY(1)

Virtual Address	Name		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
BF88_1070	IEC1	23:16	—	—	—	—	DMA3IE	DMA2IE	DMA1IE	DMA0IE
BF88_1040	IFS1	23:16	—	—	—	—	DMA3IF	DMA2IF	DMA1IF	DMA0IF
BF88_1120	IPC9	7:0	—	—	—	DMA0IP<2:0>			DMA0IS<1:0>	

Note 1: This summary table contains partial register definitions that only pertain to the DMA peripheral. Refer to the PIC32MX Family Reference Manual (DS61132) for a detailed description of these registers.

TABLE 5-6: DMA CHANNEL 1 SFR SUMMARY

Virtual Address	Name		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
BF88_3120	DCH1CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	CHCHNS
		7:0	CHEN	CHAED	CHCHN	CHAEN	CHXM	CHEDET	CHPRI<1:0>	
BF88_3124	DCH1CONCLR	31:0	Write clears selected bits in DCH1CON, read yields undefined value							
BF88_3128	DCH1CONSET	31:0	Write sets selected bits in DCH1CON, read yields undefined value							
BF88_312C	DCH1CONINV	31:0	Write inverts selected bits in DCH1CON, read yields undefined value							
BF88_3130	DCH1ECON	31:24	—	—	—	—	—	—	—	—
		23:16	CHAIRQ<7:0>							
		15:8	CHSIRQ<7:0>							
		7:0	CFORCE	CABORT	PATEN	SIRQEN	AIRQEN	-	—	—
BF88_3134	DCH1ECONCLR	31:0	Write clears selected bits in DCH1ECON, read yields undefined value							
BF88_3138	DCH1ECONSET	31:0	Write sets selected bits in DCH1ECON, read yields undefined value							
BF88_313C	DCH1ECONINV	31:0	Write inverts selected bits in DCH1ECON, read yields undefined value							
BF88_3140	DCH1INT	31:24	—	—	—	—	—	—	—	—
		23:16	CHSDIE	CHSHIE	CHDDIE	CHDHIE	CHBCIE	CHCCIE	CHTAIE	CHERIE
		15:8	—	—	—	—	—	—	—	—
		7:0	CHSDIF	CHSHIF	CHDDIF	CHDHIF	CHBCIF	CHCCIF	CHTAIF	CHERIF
BF88_3144	DCH1INTCLR	31:0	Write clears selected bits in DCH1INT, read yields undefined value							
BF88_3148	DCH1INTSET	31:0	Write sets selected bits in DCH1INT, read yields undefined value							
BF88_314C	DCH1INTINV	31:0	Write inverts selected bits in DCH1INT, read yields undefined value							
BF88_3150	DCH1SSA	31:24	CHSSA<31:24>							
		23:16	CHSSA<23:16>							
		15:8	CHSSA<15:8>							
		7:0	CHSSA<7:0>							
BF88_3154	DCH1SSACLR	31:0	Write clears selected bits in DCH1SSA, read yields undefined value							
BF88_3158	DCH1SSASET	31:0	Write sets selected bits in DCH1SSA, read yields undefined value							
BF88_315C	DCH1SSAINV	31:0	Write inverts selected bits in DCH1SSA, read yields undefined value							
BF88_3160	DCH1DSA	31:24	CHDSA<31:24>							
		23:16	CHDSA<23:16>							
		15:8	CHDSA<15:8>							
		7:0	CHDSA<7:0>							
BF88_3164	DCH1DSACLR	31:0	Write clears selected bits in DCH1DSA, read yields undefined value							
BF88_3168	DCH1DSASET	31:0	Write sets selected bits in DCH1DSA, read yields undefined value							
BF88_316C	DCH1DSAINV	31:0	Write inverts selected bits in DCH1DSA, read yields undefined value							
BF88_3170	DCH1SSIZ	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHSSIZ<7:0>							
BF88_3174	DCH1SSIZCLR	31:0	Write clears selected bits in DCH1SSIZ, read yields undefined value							
BF88_3178	DCH1SSIZSET	31:0	Write sets selected bits in DCH1SSIZ, read yields undefined value							
BF88_317C	DCH1SSIZINV	31:0	Write inverts selected bits in DCH1SSIZ, read yields undefined value							

Note 1: The starting address of the registers for DMA channel n is 0xbf883060 + 0xc0*n.
2: These bits are relevant in Extended Addressing mode only.

TABLE 5-6: DMA CHANNEL 1 SFR SUMMARY (CONTINUED)

Virtual Address	Name		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
BF88_3180	DCH1DSIZ	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHDSIZ<15:8>(2)							
		7:0	CHDSIZ<7:0>							
BF88_3184	DCH1DSIZCLR	31:0	Write clears selected bits in DCH1DSIZ, read yields undefined value							
BF88_3188	DCH1DSIZSET	31:0	Write sets selected bits in DCH1DSIZ, read yields undefined value							
BF88_318C	DCH1DSIZINV	31:0	Write inverts selected bits in DCH1DSIZ, read yields undefined value							
BF88_3190	DCH1SPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHSPTR<7:0>							
BF88_31A0	DCH1DPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHDPTR<15:8>(2)							
		7:0	CHDPTR<7:0>							
BF88_31B0	DCH1CSIZ	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHCSIZ<7:0>							
BF88_31B4	DCH1CSIZCLR	31:0	Write clears selected bits in DCH1CSIZ, read yields undefined value							
BF88_31B8	DCH1CSIZSET	31:0	Write sets selected bits in DCH1CSIZ, read yields undefined value							
BF88_31BC	DCH1CSIZINV	31:0	Write inverts selected bits in DCH1CSIZ, read yields undefined value							
BF88_31C0	DCH1CPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHCPTR<7:0>							
BF88_31D0	DCH1DAT	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHPDAT<7:0>							
BF88_31D4	DCH1DATCLR	31:0	Write clears selected bits in DCH1DAT, read yields undefined value							
BF88_31D8	DCH1DATSET	31:0	Write sets selected bits in DCH1DAT, read yields undefined value							
BF88_31DC	DCH1DATINV	31:0	Write inverts selected bits in DCH1DAT, read yields undefined value							

Note 1: The starting address of the registers for DMA channel n is 0xbf883060 + 0xc0*n.
2: These bits are relevant in Extended Addressing mode only.

TABLE 5-7: DMA CHANNEL 1 INTERRUPT REGISTER SUMMARY(1)

Virtual Address	Name		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
BF88_1070	IEC1	23:16	—	—	—	—	DMA3IE	DMA2IE	DMA1IE	DMA0IE
BF88_1040	IFS1	23:16	—	—	—	—	DMA3IF	DMA2IF	DMA1IF	DMA0IF
BF88_1120	IPC9	15:8	—	—	—	DMA1IP<2:0>			DMA1IS<1:0>	

Note 1: This summary table contains partial register definitions that only pertain to the DMA peripheral. Refer to the "*PIC32MX Family Reference Manual*" (DS61132) for a detailed description of these registers.

TABLE 5-8: DMA CHANNEL 2 SFR SUMMARY

Virtual Address[1]	Name		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
BF88_31E0	DCH2CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	CHCHNS
		7:0	CHEN	CHAED	CHCHN	CHAEN	CHXM	CHEDET	CHPRI<1:0>	
BF88_31E4	DCH2CONCLR	31:0	Write clears selected bits in DCH2CON, read yields undefined value							
BF88_31E8	DCH2CONSET	31:0	Write sets selected bits in DCH2CON, read yields undefined value							
BF88_31EC	DCH2CONINV	31:0	Write inverts selected bits in DCH2CON, read yields undefined value							
BF88_31F0	DCH2ECON	31:24	—	—	—	—	—	—	—	—
		23:16	CHAIRQ<7:0>							
		15:8	CHSIRQ<7:0>							
		7:0	CFORCE	CABORT	PATEN	SIRQEN	AIRQEN	—	—	—
BF88_31F4	DCH2ECONCLR	31:0	Write clears selected bits in DCH2ECON, read yields undefined value							
BF88_31F8	DCH2ECONSET	31:0	Write sets selected bits in DCH2ECON, read yields undefined value							
BF88_31FC	DCH2ECONINV	31:0	Write inverts selected bits in DCH2ECON, read yields undefined value							
BF88_3200	DCH2INT	31:24	—	—	—	—	—	—	—	—
		23:16	CHSDIE	CHSHIE	CHDDIE	CHDHIE	CHBCIE	CHCCIE	CHTAIE	CHERIE
		15:8	—	—	—	—	—	—	—	—
		7:0	CHSDIF	CHSHIF	CHDDIF	CHDHIF	CHBCIF	CHCCIF	CHTAIF	CHERIF
BF88_3204	DCH2INTCLR	31:0	Write clears selected bits in DCH2INT, read yields undefined value							
BF88_3208	DCH2INTSET	31:0	Write sets selected bits in DCH2INT, read yields undefined value							
BF88_320C	DCH2INTINV	31:0	Write inverts selected bits in DCH2INT, read yields undefined value							
BF88_3210	DCH2SSA	31:24	CHSSA<31:24>							
		23:16	CHSSA<23:16>							
		15:8	CHSSA<15:8>							
		7:0	CHSSA<7:0>							
BF88_3214	DCH2SSACLR	31:0	Write clears selected bits in DCH2SSA, read yields undefined value							
BF88_3218	DCH2SSASET	31:0	Write sets selected bits in DCH2SSA, read yields undefined value							
BF88_321C	DCH2SSAINV	31:0	Write inverts selected bits in DCH2SSA, read yields undefined value							
BF88_3220	DCH2DSA	31:24	CHDSA<31:24>							
		23:16	CHDSA<23:16>							
		15:8	CHDSA<15:8>							
		7:0	CHDSA<7:0>							
BF88_3224	DCH2DSACLR	31:0	Write clears selected bits in DCH2DSA, read yields undefined value							
BF88_3228	DCH2DSASET	31:0	Write sets selected bits in DCH2DSA, read yields undefined value							
BF88_322C	DCH2DSAINV	31:0	Write inverts selected bits in DCH2DSA, read yields undefined value							
BF88_3230	DCH2SSIZ	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHSSIZ<7:0>							
BF88_3234	DCH2SSIZCLR	31:0	Write clears selected bits in DCH2SSIZ, read yields undefined value							
BF88_3238	DCH2SSIZSET	31:0	Write sets selected bits in DCH2SSIZ, read yields undefined value							
BF88_323C	DCH2SSIZINV	31:0	Write inverts selected bits in DCH2SSIZ, read yields undefined value							
BF88_3240	DCH2DSIZ	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHDSIZ<15:8>[2]							
		7:0	CHDSIZ<7:0>							
BF88_3244	DCH2DSIZCLR	31:0	Write clears selected bits in DCH2DSIZ, read yields undefined value							
BF88_3248	DCH2DSIZSET	31:0	Write sets selected bits in DCH2DSIZ, read yields undefined value							
BF88_324C	DCH2DSIZINV	31:0	Write inverts selected bits in DCH2DSIZ, read yields undefined value							

Note 1: The starting address of the registers for DMA channel n is 0xbf883060 + 0xc0*n.
2: These bits are relevant in Extended Addressing mode only.

TABLE 5-8: DMA CHANNEL 2 SFR SUMMARY (CONTINUED)

Virtual Address(1)	Name		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
BF88_3250	DCH2SPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHSPTR<7:0>							
BF88_3260	DCH2DPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHDPTR<15:8>(2)							
		7:0	CHDPTR<7:0>							
BF88_3270	DCH2CSIZ	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHCSIZ<7:0>							
BF88_3274	DCH2CSIZCLR	31:0	Write clears selected bits in DCH2CSIZ, read yields undefined value							
BF88_3278	DCH2CSIZSET	31:0	Write sets selected bits in DCH2CSIZ, read yields undefined value							
BF88_327C	DCH2CSIZINV	31:0	Write inverts selected bits in DCH2CSIZ, read yields undefined value							
BF88_3280	DCH2CPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHCPTR<7:0>							
BF88_3290	DCH2DAT	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHPDAT<7:0>							
BF88_3294	DCH2DATCLR	31:0	Write clears selected bits in DCH2DAT, read yields undefined value							
BF88_3298	DCH2DATSET	31:0	Write sets selected bits in DCH2DAT, read yields undefined value							
BF88_329C	DCH2DATINV	31:0	Write inverts selected bits in DCH2DAT, read yields undefined value							

Note 1: The starting address of the registers for DMA channel n is 0xbf883060 + 0xc0*n.
2: These bits are relevant in Extended Addressing mode only.

TABLE 5-9: DMA CHANNEL 2 INTERRUPT REGISTER SUMMARY(1)

Virtual Address	Name		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
BF88_1070	IEC1	23:16	—	—	—	—	DMA3IE	DMA2IE	DMA1IE	DMA0IE
BF88_1040	IFS1	23:16	—	—	—	—	DMA3IF	DMA2IF	DMA1IF	DMA0IF
BF88_1120	IPC9	23:16	—	—	—	DMA2IP<2:0>			DMA2IS<1:0>	

Note 1: This summary table contains partial register definitions that only pertain to the DMA peripheral. Refer to the PIC32MX Family Reference Manual (DS61132) for a detailed description of these registers.

TABLE 5-10: DMA CHANNEL 3 SFR SUMMARY

Virtual Address[1]	Name		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
BF88_32A0	DCH3CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	CHCHNS
		7:0	CHEN	CHAED	CHCHN	CHAEN	CHXM	CHEDET	CHPRI<1:0>	
BF88_32A4	DCH3CONCLR	31:0	Write clears selected bits in DCH3CON, read yields undefined value							
BF88_32A8	DCH3CONSET	31:0	Write sets selected bits in DCH3CON, read yields undefined value							
BF88_32AC	DCH3CONINV	31:0	Write inverts selected bits in DCH3CON, read yields undefined value							
BF88_32B0	DCH3ECON	31:24	—	—	—	—	—	—	—	—
		23:16	CHAIRQ<7:0>							
		15:8	CHSIRQ<7:0>							
		7:0	CFORCE	CABORT	PATEN	SIRQEN	AIRQEN	—	—	—
BF88_32B4	DCH3ECONCLR	31:0	Write clears selected bits in DCH3ECON, read yields undefined value							
BF88_32B8	DCH3ECONSET	31:0	Write sets selected bits in DCH3ECON, read yields undefined value							
BF88_32BC	DCH3ECONINV	31:0	Write inverts selected bits in DCH3ECON, read yields undefined value							
BF88_32C0	DCH3INT	31:24	—	—	—	—	—	—	—	—
		23:16	CHSDIE	CHSHIE	CHDDIE	CHDHIE	CHBCIE	CHCCIE	CHTAIE	CHERIE
		15:8	—	—	—	—	—	—	—	—
		7:0	CHSDIF	CHSHIF	CHDDIF	CHDHIF	CHBCIF	CHCCIF	CHTAIF	CHERIF
BF88_32C4	DCH3INTCLR	31:0	Write clears selected bits in DCH3INT, read yields undefined value							
BF88_32C8	DCH3INTSET	31:0	Write sets selected bits in DCH3INT, read yields undefined value							
BF88_32CC	DCH3INTINV	31:0	Write inverts selected bits in DCH3INT, read yields undefined value							
BF88_32D0	DCH3SSA	31:24	CHSSA<31:24>							
		23:16	CHSSA<23:16>							
		15:8	CHSSA<15:8>							
		7:0	CHSSA<7:0>							
BF88_32D4	DCH3SSACLR	31:0	Write clears selected bits in DCH3SSA, read yields undefined value							
BF88_32D8	DCH3SSASET	31:0	Write sets selected bits in DCH3SSA, read yields undefined value							
BF88_32DC	DCH3SSAINV	31:0	Write inverts selected bits in DCH3SSA, read yields undefined value							
BF88_32E0	DCH3DSA	31:24	CHDSA<31:24>							
		23:16	CHDSA<23:16>							
		15:8	CHDSA<15:8>							
		7:0	CHDSA<7:0>							
BF88_32E4	DCH3DSACLR	31:0	Write clears selected bits in DCH3DSA, read yields undefined value							
BF88_32E8	DCH3DSASET	31:0	Write sets selected bits in DCH3DSA, read yields undefined value							
BF88_32EC	DCH3DSAINV	31:0	Write inverts selected bits in DCH3DSA, read yields undefined value							
BF88_32F0	DCH3SSIZ	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHSSIZ<7:0>							
BF88_32F4	DCH3SSIZCLR	31:0	Write clears selected bits in DCH3SSIZ, read yields undefined value							
BF88_32F8	DCH3SSIZSET	31:0	Write sets selected bits in DCH3SSIZ, read yields undefined value							
BF88_32FC	DCH3SSIZINV	31:0	Write inverts selected bits in DCH3SSIZ, read yields undefined value							
BF88_3300	DCH3DSIZ	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHDSIZ<15:8>[2]							
		7:0	CHDSIZ<7:0>							
BF88_3304	DCH3DSIZCLR	31:0	Write clears selected bits in DCH3DSIZ, read yields undefined value							
BF88_3308	DCH3DSIZSET	31:0	Write sets selected bits in DCH3DSIZ, read yields undefined value							
BF88_330C	DCH3DSIZINV	31:0	Write inverts selected bits in DCH3DSIZ, read yields undefined value							
BF88_3310	DCH3SPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHSPTR<7:0>							

Note 1: The starting address of the registers for DMA channel n is 0xbf883060 + 0xc0*n.
2: These bits are relevant in Extended Addressing mode only.

TABLE 5-10: DMA CHANNEL 3 SFR SUMMARY (CONTINUED)

Virtual Address(1)	Name		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
BF88_3320	DCH3DPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHDPTR<15:8>(2)							
		7:0	CHDPTR<7:0>							
BF88_3330	DCH3CSIZ	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHCSIZ<7:0>							
BF88_3334	DCH3CSIZCLR	31:0	Write clears selected bits in DCH3CSIZ, read yields undefined value							
BF88_3338	DCH3CSIZSET	31:0	Write sets selected bits in DCH3CSIZ, read yields undefined value							
BF88_333C	DCH3CSIZINV	31:0	Write inverts selected bits in DCH3CSIZ, read yields undefined value							
BF88_3340	DCH3CPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHCPTR<7:0>							
BF88_3350	DCH3DAT	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHPDAT<7:0>							
BF88_3354	DCH3DATCLR	31:0	Write clears selected bits in DCH3DAT, read yields undefined value							
BF88_3358	DCH3DATSET	31:0	Write sets selected bits in DCH3DAT, read yields undefined value							
BF88_335C	DCH3DATINV	31:0	Write inverts selected bits in DCH3DAT, read yields undefined value							

Note 1: The starting address of the registers for DMA channel n is 0xbf883060 + 0xc0*n.
2: These bits are relevant in Extended Addressing mode only.

TABLE 5-11: DMA CHANNEL 3 INTERRUPT REGISTER SUMMARY

Virtual Address	Name		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
BF88_1070	IEC1	23:16	—	—	—	—	DMA3IE	DMA2IE	DMA1IE	DMA0IE
BF88_1040	IFS1	23:16	—	—	—	—	DMA3IF	DMA2IF	DMA1IF	DMA0IF
BF88_1120	IPC9	31:24	—	—	—	DMA3IP<2:0>			DMA3IS<1:0>	

Note 1: This summary table contains partial register definitions that only pertain to the DMA peripheral. Refer to the PIC32MX Family Reference Manual (DS61132) for a detailed description of these registers.

REGISTER 5-1: DMACON: DMA CONTROLLER CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
ON	FRZ	SIDL	SUSPEND	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** DMA On bit
1 = DMA module is enabled
0 = DMA module is disabled

bit 14 **FRZ:** DMA Freeze bit(1)
1 = DMA is frozen during Debug mode
0 = DMA continues to run during Debug mode
Note: FRZ is writable in Debug Exception mode only, it is forced to ‘0’ in Normal mode.

bit 13 **SIDL:** Stop in Idle Mode bit
1 = DMA transfers are frozen during Sleep
0 = DMA transfers continue during Sleep

bit 12 **SUSPEND:** DMA Suspend bit
1 = DMA transfers are suspended to allow CPU uninterrupted access to data bus
0 = DMA operates normally

bit 11-0 **Unimplemented:** Read as ‘0’

REGISTER 5-2: DMASTAT: DMA STATUS REGISTER(1)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	R-0	U-0	R-0	R-0
—	—	—	—	RDWR	—	DMACH<1:0>	
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-4 **Unimplemented:** Read as ‘0’

bit 3 **RDWR:** Read/Write Status bit
1 = Last DMA bus access was a read
0 = Last DMA bus access was a write

bit 2 **Unimplemented:** Read as ‘0’

bit 1-0 **DMACH<1:0>:** DMA Channel bits

Note 1: This register contains the value of the most recent active DMA channel.

REGISTER 5-3: DMAADDR: DMA ADDRESS REGISTER(1)

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DMAADDR<31:24>							
bit 31							bit 24

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DMAADDR<23:16>							
bit 23							bit 16

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DMAADDR<15:8>							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DMAADDR<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-0 **DMAADDR<31:0>:** DMA Module Address bits

Note 1: This register contains the address of the most recent DMA access.

REGISTER 5-4: DCRCCON: DMA CRC CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	PLEN<4:0>				
bit 15							bit 8

R/W-0	R/W-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
CRCEN	CRCAPP	—	—	—	—	CRCCH<1:0>	
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-13 **Unimplemented:** Read as ‘0’

bit 12-8 **PLEN<4:0>:** Polynomial Length bits

Denotes the length of the polynomial –1.

bit 7 **CRCEN:** CRC Enable bit

1 = CRC module is enabled and channel transfers are routed through the CRC module
0 = CRC module is disabled and channel transfers proceed normally

bit 6 **CRCAPP:** CRC Append Mode bit

1 = Data read will be passed to the CRC, to be included in the CRC calculation, but is not written to the destination register. When a block transfer completes, the calculated CRC will be written to the location given by DCHxDSA
0 = Channel behaves normally, with the CRC being calculated as data is transferred from the source to the destination

bit 5-2 **Unimplemented:** Read as ‘0’

bit 1-0 **CRCCH<1:0>:** CRC Channel Select bits

11 = CRC is assigned to Channel 3
10 = CRC is assigned to Channel 2
01 = CRC is assigned to Channel 1
00 = CRC is assigned to Channel 0

 Advance Information

REGISTER 5-5: DCRCDATA: DMA CRC DATA REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DCRCDATA<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DCRCDATA<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **DCRCDATA<15:0>:** CRC Data Register bits

Writing to this register will seed the CRC generator. Reading from this register will return the current value of the CRC. Bits > PLEN will return ‘0’ on any read.

REGISTER 5-6: DCRCXOR: DMA CRC XOR ENABLE REGISTER[1]

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DCRCXOR<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DCRCXOR<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **DCRCXOR<15:0>:** CRC XOR Register bits

1 = Enable the XOR input to the Shift register

0 = Disable the XOR input to the Shift register; data is shifted directly in from the previous stage in the register

Note 1: The LSb of the DCRCXOR register will be always set.

REGISTER 5-7: DCHXCON: DMA CHANNEL X CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	CHCHNS
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0
CHEN	CHAED	CHCHN	CHAEN	CHXM	CHEDET	CHPRI<1:0>	
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-9 **Unimplemented:** Read as ‘0’

bit 8 **CHCHNS:** Chain Channel Selection bit
1 = Chain to channel lower in natural priority (CH1 will be enabled by CH2 transfer complete)
0 = Chain to channel higher in natural priority (CH1 will be enabled by CH0 transfer complete)
Note: The chain selection bit takes effect when chaining is enabled, i.e., CHCHN = 1.

bit 7 **CHEN:** Channel Enable bit
1 = Channel is enabled
0 = Channel is disabled

bit 6 **CHAED:** Channel Allow Events If Disabled bit
1 = Channel start/abort events will be registered, even if the channel is disabled
0 = Channel start/abort events will be ignored if the channel is disabled

bit 5 **CHCHN:** Channel Chain Enable bit
1 = Allow channel to be chained to channel higher in natural priority
0 = Do not chain to channel higher in natural priority

bit 4 **CHAEN:** Channel Automatic Enable bit
1 = Channel is continuously enabled, and not automatically disabled after a block transfer is complete
0 = Channel is disabled on block transfer complete

bit 3 **CHXM:** Channel Extended Addressing Mode Enable bit
1 = Extended Addressing mode is enabled
0 = Extended Addressing mode is disabled

bit 2 **CHEDET:** Channel Event Detected bit
1 = An event has been detected
0 = No events have been detected

bit 1-0 **CHPRI<1:0>:** Channel Priority bits
11 = Channel has priority 3 (highest)
10 = Channel has priority 2
01 = Channel has priority 1
00 = Channel has priority 0

REGISTER 5-8: DCHXECON: DMA CHANNEL X EVENT CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
CHAIRQ<7:0>							
bit 23							bit 16

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
CHSIRQ<7:0>							
bit 15							bit 8

S-0	S-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0
CFORCE	CABORT	PATEN	SIRQEN	AIRQEN	—	—	—
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-24 **Unimplemented:** Read as ‘0’

bit 23-16 **CHAIRQ<7:0>:** IRQ that will abort Channel Transfer bits
11111111 = Interrupt 255 will abort any transfers in progress and set CHAIF flag

• • •

00000001 = Interrupt 1 will abort any transfers in progress and set CHAIF flag
00000000 = Interrupt 0 will abort any transfers in progress and set CHAIF flag

bit 15-8 **CHSIRQ<7:0>:** IRQ that will Start Channel Transfer bits
11111111 = Interrupt 255 will initiate a DMA transfer

• • •

00000001 = Interrupt 1 will initiate a DMA transfer
00000000 = Interrupt 0 will initiate a DMA transfer

bit 7 **CFORCE:** DMA Forced Transfer bit
1 = A DMA transfer is forced to begin when this bit is written to a ‘1’
0 = This bit always reads ‘0’

bit 6 **CABORT:** DMA Abort Transfer bit
1 = A DMA transfer is aborted when this bit is written to a ‘1’
0 = This bit always reads ‘0’

bit 5 **PATEN:** Channel Pattern Match Abort Enable bit
1 = Abort transfer and clear CHEN on pattern match
0 = Pattern match is disabled

bit 4 **SIRQEN:** Channel Start IRQ Enable bit
1 = Start channel cell transfer if an interrupt matching CHSIRQ occurs
0 = Interrupt number CHSIRQ is ignored and does not start a transfer

REGISTER 5-8: DCHXECON: DMA CHANNEL X EVENT CONTROL REGISTER (CONTINUED)

bit 3	**AIRQEN:** Channel Abort IRQ Enable bit 1 = Channel transfer is aborted if an interrupt matching CHAIRQ occurs 0 = Interrupt number CHAIRQ is ignored and does not terminate a transfer
bit 2-0	**Unimplemented:** Read as '0'

REGISTER 5-9: DCHXINT: DMA CHANNEL X INTERRUPT CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CHSDIE	CHSHIE	CHDDIE	CHDHIE	CHBCIE	CHCCIE	CHTAIE	CHERIE
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CHSDIF	CHSHIF	CHDDIF	CHDHIF	CHBCIF	CHCCIF	CHTAIF	CHERIF
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-24 **Unimplemented:** Read as ‘0’

bit 23 **CHSDIE:** Channel Source Done Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 22 **CHSHIE:** Channel Source Half Empty Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 21 **CHDDIE:** Channel Destination Done Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 20 **CHDHIE:** Channel Destination Half Full Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 19 **CHBCIE:** Channel Block Transfer Complete Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 18 **CHCCIE:** Channel Cell Transfer Complete Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 17 **CHTAIE:** Channel Transfer Abort Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 16 **CHERIE:** Channel Address Error Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 15-8 **Unimplemented:** Read as ‘0’

REGISTER 5-9: DCHXINT: DMA CHANNEL X INTERRUPT CONTROL REGISTER (CONTINUED)

bit 7 **CHSDIF:** Channel Source Done Interrupt Flag bit
1 = Channel Source Pointer has reached end of source (CHSPTR == CHSSIZ)
0 = No interrupt is pending

bit 6 **CHSHIF:** Channel Source Half Empty Interrupt Flag bit
1 = Channel Source Pointer has reached midpoint of source (CHSPTR == CHSSIZ/2)
0 = No interrupt is pending

bit 5 **CHDDIF:** Channel Destination Done Interrupt Flag bit
1 = Channel Destination Pointer has reached end of destination (CHDPTR == CHDSIZ)
0 = No interrupt is pending

bit 4 **CHDHIF:** Channel Destination Half Full Interrupt Flag bit
1 = Channel Destination Pointer has reached midpoint of destination (CHDPTR == CHDSIZ/2)
0 = No interrupt is pending

bit 3 **CHBCIF:** Channel Block Transfer Complete Interrupt Flag bit
1 = A block transfer has been completed (the larger of CHSSIZ/CHDSIZ bytes has been transferred) or a pattern match event occurs
0 = No interrupt is pending

bit 2 **CHCCIF:** Channel Cell Transfer Complete Interrupt Flag bit
1 = A cell transfer has been completed (CHCSIZ bytes have been transferred)
0 = No interrupt is pending

bit 1 **CHTAIF:** Channel Transfer Abort Interrupt Flag bit
1 = An interrupt matching CHAIRQ has been detected and the DMA transfer has been aborted
0 = No interrupt is pending

bit 0 **CHERIF:** Channel Address Error Interrupt Flag bit
1 = A channel address error has been detected
Either the source or the destination address is invalid.
0 = No interrupt is pending

REGISTER 5-10: DCHXSSA: DMA CHANNEL X SOURCE START ADDRESS REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHSSA<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHSSA<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHSSA<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHSSA<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-0 **CHSSA<31:0>** Channel Source Start Address bits
Channel source start address.
Note: This must be the physical address of the source.

REGISTER 5-11: DCHXDSA: DMA CHANNEL X DESTINATION START ADDRESS REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHDSA<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHDSA<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHDSA<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHDSA<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-0 **CHDSA<31:0>:** Channel Destination Start Address bits

Channel destination start address.

Note: This must be the physical address of the destination.

REGISTER 5-12: DCHXSSIZ: DMA CHANNEL X SOURCE SIZE REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHSSIZ<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-8 **Unimplemented:** Read as ‘0’

bit 7-0 **CHSSIZ<7:0>:** Channel Source Size bits

CHXM = 0 (DCHxCON<3>) (Normal Addressing mode):

255 = 255-byte source size

• • •

2 = 2-byte source size
1 = 1-byte source size
0 = 256-byte source size

CHXM = 1 (DCHxCON<3>) (Extended Addressing mode):

These bits make up the Most Significant bits of the transfer size.

REGISTER 5-13: DCHXDSIZ: DMA CHANNEL X DESTINATION SIZE REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
CHDSIZ<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHDSIZ<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **CHDSIZ<15:0>:** Channel Destination Size bits

CHXM = 0 (DCHxCON<3>) (Normal Addressing mode):
CHDSIZ<15:8> Unused, read as ‘0’, write has no effect.
CHDSIZ<7:0> Read/Write Normal mode transfer size:
255 = 255-byte destination size

• • •

2 = 2-byte destination size
1 = 1-byte destination size
0 = 256-byte destination size

CHXM = 1 (DCHxCON<3>) (Extended Addressing mode):
CHDSIZ<15:0> Read Extended mode transfer size:
65535 = 65535-byte destination size

• • •

2 = 2-byte destination size
1 = 1-byte destination size
0 = 65536-byte destination size.
CHDSIZ<15:8> write has no effect.
CHDSIZ<7:0> write sets the LSB of Extended mode transfer size.

REGISTER 5-14: DCHXSPTR: DMA CHANNEL X SOURCE POINTER REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHSPTR<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-8 **Unimplemented:** Read as ‘0’

bit 7-0 **CHSPTR<7:0>:** Channel Source Pointer bits

CHXM = 0 (DCHxCON<3>) (Normal Addressing mode):

255 = Points to 255th byte of the source

• • •

1 = Points to 1st byte of the source

0 = Points to 0th byte of the source

CHXM =1 (DCHxCON<3>) (Extended Addressing mode):

These bits comprise the Most Significant bits of the pointer.

Note: This is reset on pattern detect, when in Pattern Detect mode.

REGISTER 5-15: DCHXDPTR: CHANNEL X DESTINATION POINTER REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHDPTR<15:8>							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHDPTR<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **CHDPTR<15:0>:** Channel Destination Pointer bits

CHXM = 0 (DCHxCON<3>) (Normal Addressing mode):
CHDPTR<15:8> Unused, read as ‘0’.
CHDPTR<7:0> Normal Mode Destination Pointer:
255 = Points to 255th byte of the destination

• • •

1 = Points to 1st byte of the destination
0 = Points to 0th byte of the destination

CHXM = 1 (DCHxCON<3>) (Extended Addressing mode):
CHDPTR<15:0> Extended Mode Destination Pointer:
65535 = Points to byte 65535 (0xFFFF) of the source/destination

• • •

255 = Points to byte 255 of the source/destination

• • •

1 = Points to byte 1 of the source/destination
0 = Points to byte 0 of the source/destination

REGISTER 5-16: DCHXCSIZ: DMA CHANNEL X CELL-SIZE REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHCSIZ<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: ('0', '1', x = Unknown)

bit 31-8 **Unimplemented:** Read as '0'

bit 7-0 **CHCSIZ<7:0>:** Channel Cell Size bits

CHXM = 0 (DCHxCON<3>) (Normal Addressing mode):

255 = 255 bytes transferred on an event

• • •

2 = 2 bytes transferred on an event
1 = 1 byte transferred on an event
0 = 256 bytes transferred on an event

CHXM = 1 (DCHxCON<3>) (Extended Addressing mode):
These bits are not used in Extended Addressing mode.

REGISTER 5-17: DCHXCPTR: DMA CHANNEL X CELL POINTER REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHCPTR<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-8 **Unimplemented:** Read as ‘0’

bit 7-0 **CHCPTR<7:0>:** Channel Cell Progress Pointer bits

CHXM = 0 (DCHxCON<3>) (Normal Addressing mode):

255 = 255 Bytes have been transferred since the last event

• • •

1 = 1 Bytes have been transferred since the last event

0 = 0 Bytes have been transferred since the last event

CHXM = 1 (DCHxCON<3>) (Extended Addressing mode):

These bits are not used in Extended Addressing mode.

Note: This is reset on pattern detect, when in Pattern Detect mode

REGISTER 5-18: DCHXDAT: DMA CHANNEL X PATTERN DATA REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHPDAT<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-8 **Unimplemented:** Read as ‘0’

bit 7-0 **CHPDAT<7:0>:** Channel Data Register bits

Pattern Terminate mode:
Data to be matched must be stored in this register to allow terminate on match.

All other modes:
Unused.

 Advance Information

5.2 DMA Controller Operation

A DMA channel will transfer data from a source to a destination without CPU intervention.

DMA controller configuration resources:

- The DMA Controller and the corresponding DMA channel have to be enabled using the ON (DMACON<15>) and the CHEN (DCHxCON<7>) bits.
- The source and destination of the transfer are programmable using the DCHxSSA and DCHxDSA registers respectively.
- The source and destination are further independently configurable using the DCHxSSIZ and DCHxDSIZ registers.
- A DMA transfer can be initiated in one of two ways:
  - Software can initiate a transfer by setting the channel CFORCE (DCHxECON<7>) bit.
  - An interrupt event occurs that matches the CHSIRQ (DCHxECON<15:8>) interrupt and SIRQEN = 1 (DCHxECON<4>). The user can select any interrupt on the device to start a DMA transfer.
- At each event requiring a DMA transfer, a number of bytes specified by the cell size (DCHxCSIZ) will be transferred (one or more transactions will occur).
- The channel keeps track of the number of bytes transferred from the source to destination, using Source and Destination Pointers (DCHxSPTR and DCHxDPTR).
- The Source and Destination Pointers are read-only and are updated after every transaction.
- Interrupts are generated when the Source or Destination pointer is half of the source or destination size (DCHxSSIZ/2 or DCHxDSIZ/2), or when the source or destination counter equals the size of the source or destination. These interrupts are CHSHIF, CHDHIF and CHSDIF, CHDDIF, respectively.
- The Source and Destination Pointers are reset:
  - On any device Reset.
  - When the DMA is turned off (ON bit (DMACON<15>) is ‘0’).
  - A block transfer completes (regardless of the state of CHAEN (DCHxCON<4>)).
  - A pattern match terminates a transfer (regardless of the state of auto-enable CHAEN (DCHxCON<4>)).
  - The CABORT (DCHxECON<6>) flag is written.
  - If the channel source address (DCHxSSA) is updated, the Source Pointer (DCHxSPTR) will be reset.
  - Similarly, updates to the Destination Address (DCHxDSA) will cause the Destination Pointer (DCHxDPTR) to be reset.
- Normally, the DMA channel remains enabled until the DMA channel has completed a block transfer unless the auto-enable feature is turned on (i.e., CHAEN = 1).
- When the channel is disabled, further transfers will be prohibited until the channel is re-enabled (CHEN is set to ‘1’).
- A DMA transfer request will be stopped/aborted by:
  - Writing the CABORT bit (DCHxECON<6>).
  - Pattern match occurs if pattern match is enabled PATEN = 1 (DCHxECON<5>), provided that channel CHAEN is not set.
  - Interrupt event occurs on the device that matches the CHAIRQ (DCHxECON<23:16>) interrupt if enabled by AIRQEN (DCHxECON<3>).
  - An address error is detected.
  - A block transfer completes provided that Channel Auto-Enable mode (CHAEN) is not set.
- When a channel abort interrupt occurs, the Channel Abort Interrupt Flag, CHTAIF, (DCHxINT<1>) is set. This allows the user to detect and recover from an aborted DMA transfer. When a transfer is aborted, any transaction currently underway will be completed.

5.2.1 DMA CONTROLLER TERMINOLOGY

Event: Any system event that can initiate or abort a DMA transfer.

Transaction: A single-word transfer (up to 4 bytes), comprised of read and write operations.

Cell Transfer: The number of bytes transferred when a DMA channel has a transfer initiated before waiting for another event (given by the DCHCSIZ register). A cell transfer comprises one or more transactions.

Block Transfer: Defined as the number of bytes transferred when a channel is enabled. The number of bytes is the larger of either DCHxSSIZ or DCHxDSIZ. A block transfer comprises one or more cell transfers.

5.3 Normal Addressing Mode

The mode is enabled by clearing the CHXM bit (DCHxCON<3>).

Normal Addressing mode transfer features:

- In Normal Addressing mode, the transfer size is limited to a maximum of 256 bytes transferred per channel.
- The Source and Destination Pointers wrap around based on the selected source and destination size.
- A block transfer is complete when the block size bytes have been transferred. The block size is the larger of source and destination sizes:
  - blockSize = max (DCHxSSIZ, DCHxDSIZ).
- A DMA event will transfer cell size (DCHxCSIZ) bytes from source to destination. However, if DCHxCSIZ is greater than the block size, then just block size bytes will be transferred.

5.3.1 NORMAL ADDRESSING MODE TRANSFER CONFIGURATION

Microchip recommends taking the following steps to configure a DMA transfer in Normal Addressing mode:

- Disable the DMA channel interrupts in the INT controller.
- Clear any existing channel interrupt flags in the INT controller.
- Enable the DMA controller (if not already enabled) in DMACON register.
- Set Channel Control register: Priority, Auto-Enable mode, etc., in DCHxCON. Use CHXM = 0 (DCHxCON<3>) for Normal Addressing mode. (Don't enable the channel yet!)
- Set the channel event control: clear/set the events starting and aborting the transfer. If needed, also set the pattern match enable in DCHxECON.
- If using a pattern match, set the pattern in the DCHxDAT register.
- Set the transfer source and destination physical addresses (DCHxSSA and DCHxDSA registers).
- Set the source and destination sizes (DCHxSSIZ, DCHxDSIZ registers).
- Set the cell transfer size (DCHxCSIZ).
- Clear any existing event flag in the DCHxINT register.
- If using interrupts:
  - Set the conditions that will generate an interrupt in the DCHxINT register (at least error interrupt enable and abort interrupt enable, usually block complete interrupt).
  - Set the DMA channel interrupt priority and subpriority in the INT controller.
  - Enable the DMA channel interrupt in the INT controller.
- Enable the selected DMA channel with CHEN (DCHxCON<7>).
- If not using system events to start the DMA transfer use CFORCE (DCHxECON<7>) to start transfer.
- Until the DMA transfer is complete you can do some other processing.
- If transfer complete interrupts (cell complete, block complete, etc.) are enabled, a notification will be presented in the ISR that the DMA transfer completed.
- Otherwise, the DMA channel can be polled to see if the transfer is completed using, for example, CHBCIF (DCHxINT<3>).

Refer to Example 5-1.

EXAMPLE 5-1: CONFIGURING THE DMA FOR NORMAL ADDRESSING MODE OPERATION

```
/*
The following code example illustrates a DMA channel 0 configuration for a normal addressing
transfer.
*/
       IEC1CLR=0x00010000;           // disable DMA channel 0 interrupts
       IFS1CLR=0x00010000;           // clear existing DMA channel 0 interrupt flag

       DMACONSET=0x00008000;         // enable the DMA controller
       DCH0CON=0x3;                  // channel off, pri 3, normal mode, no chaining

       CH0ECON=0;                    // no start or stop irq's, no pattern match

                                     // program the transfer
       DCH0SSA=0x1d010000;           // transfer source physical address
       DCH0DSA=0x1d020000;           // transfer destination physical address
       DCH0SSIZ=0;                   // source size 256 bytes
       DCH0DSIZ=0;                   // destination size 256 bytes
       DCH0CSIZ=0;                   // 256 bytes transferred per event

       DCH0INTCLR=0x00ff00ff;        // clear existing events, disable all interrupts
       DCH0CONSET=0x80;              // turn channel on

                                     // initiate a transfer
       DCH0ECONSET=0x00000080;       // set CFORCE to 1

       // do something else

       // poll to see that the transfer was done

       while(TRUE)
       {
              register int pollCnt;  // use a poll counter.
                                     // polling continuously the DMA controller in a tight
                                     // loop would affect the performance of the DMA transfer

              int dmaFlags=DCH0INT;
              if( (dmaFlags&0xb)
              {                      // one of CHERIF (DCHxINT<0>), CHTAIF (DCHxINT<1>)
                                     // or CHBCIF (DCHxINT<3>) flags set
                     break;          // transfer completed
              }
              pollCnt=100;           // use an adjusted value here
              while(pollCnt--);      // wait before reading again the DMA controller
       }


       // check the transfer completion result
```

5.4 Extended Addressing Mode

The mode is enabled by setting the CHXM bit (DCHxCON<3>)

Extended Addressing mode transfer features:

- The maximum transfer size per channel is 64 Kbytes.
- The source and destination sizes are concatenated and the size is 16 bits wide. DCHxSSIZ will comprise the Most Significant bits of the channel transfer size, the Destination Size Register (DCHxDSIZ) will make up the Least Significant bits of the transfer size.
- The Source and Destination Pointers (DCHxSPTR/DCHxDPTR) are concatenated in the same way as the source and destination sizes. A read of the DCHxDPTR register will return the full 16-bit Channel Transfer Pointer (DCHxSPTR concatenated with DCHxDPTR). A read of DCHxSPTR in this mode will return the high-order bits of the Transfer pointer.
- Cell size is identical to block size. DCHxCSIZ and DCHxCPTR are not used.

5.4.1 EXTENDED ADDRESSING MODE CONFIGURATION

The following steps are recommended to be taken to configure a DMA transfer in Extended Addressing mode:

- Disable the DMA channel interrupts in the INT controller.
- Clear any existing channel interrupt flags in the INT controller
- Enable the DMA controller (if not already enabled) in DMACON register.
- Set Channel Control register: Priority, Auto-Enable mode, etc., in DCHxCON. Use CHXM = `1` (DCHxCON<3>) for Extended Addressing mode. Don't enable the channel yet.
- Set the channel event control: clear/set the events starting and aborting the transfer. If needed, also set the pattern match enable in DCHxECON.
- If using a pattern match, set the pattern in the DCHxDAT register.
- Set the transfer source and destination physical addresses (DCHxSSA and DCHxDSA registers).
- Set the block transfer size (DCHxSSIZ and DCHxDSIZ).
- Clear any existing event flag in DCHxINT register.
- If using interrupts:
  - Set the conditions that will generate an interrupt in the DCHxINT register (at least error interrupt enable and abort interrupt enable, usually block complete interrupt).
  - Set the DMA channel interrupt priority and subpriority in the INT controller.
  - Enable the DMA channel interrupt in the INT controller.
- Enable the selected DMA channel with CHEN (DCHxCON<7>).
- If not using system events to start the DMA transfer use CFORCE (DCHxECON<7>) to start transfer.
- Until the DMA transfer is complete, you can do some other processing.
- If you enabled block complete interrupt you'll be notified in the ISR that the DMA transfer completed.
- Otherwise, you can poll the DMA channel to see if the transfer is completed using, for example, CHBCIF (DCHxINT<3>).

Refer to Example 5-2.

EXAMPLE 5-2: CONFIGURING THE DMA FOR EXTENDED ADDRESSING MODE OPERATIONC

```
/*
The following code example illustrates a DMA channel 0 configuration for the extended
addressing mode transfer.
*/

       IEC1CLR=0x00010000;            // disable DMA channel 0 interrupts
       IFS1CLR=0x00010000;            // clear any existing DMA channel 0 interrupt flag

       DMACONSET=0x00008000;          // enable the DMA controller
       DCH0CON=0xb;                   // channel off, priority 3, extended mode, no chaining

       DCH0ECON=0;                    // no start or stop irq's, no pattern match
                                      // program the transfer
       DCH0SSA=0x1d010000;            // transfer source physical address
       DCH0DSA=0x1d020000;            // transfer destination physical address
       DCH0SSIZ=(BYTE)(1024>>8);      // set block size MSB in src size
       DCH0DSIZ=(BYTE)1024;           // set block size LSB in dst size

       DCH0INTCLR=0x00ff00ff;         // clear existing events, disable all interrupts
       DCH0CONSET=0x80;               // turn channel on

                                      // initiate a transfer
       DCH0ECONSET=0x00000080;        // set CFORCE to 1

       // do something else

       // poll to see that the transfer was done

       while(TRUE)
       {
              register int pollCnt;   // use a poll counter.
                                      // polling continuously the DMA controller in a tight
                                      // loop would affect the performance of the DMA transfer

              int dmaFlags=DCH0INT;
              if( (dmaFlags&0xb)
              {                       // one of CHERIF (DCHxINT<0>), CHTAIF (DCHxINT<1>)
                                      // or CHBCIF (DCHxINT<3>) flags set
                     break;           // transfer completed
              }
              pollCnt=100;            // use an adjusted value here
              while(pollCnt--);       // wait before reading again the DMA controller
       }

       // check the transfer completion result
```

5.5 Pattern Match Termination

The Pattern Match mode is enabled by setting the PATEN bit (DCHxECON<5>).

This feature is useful in applications where a variable data size is required and eases the setup of the DMA channel. A good usage is for transferring ASCII command strings from an UART, <CR> ended. This is also useful for implementing string copy routines with DMA support.

Pattern Match mode features:

- Allows the user to end a transfer if a byte of data written during a transaction matches a specific pattern.
- A pattern match is treated the same way as a block transfer complete, where the CHBCIF (DCHxINT<3>) bit is set and the CHEN (DCHxCON<7>) bit is cleared provided auto-enable CHAEN = `0` (DCHxCON<4>).
- The pattern is stored in the DCHxDAT register.
- If any byte in the source matches DCHxDAT, a pattern match is detected.

5.5.1 PATTERN MATCH MODE CONFIGURATION

The Pattern Match mode is an option for use when performing DMA transfers in Normal or Extended Addressing modes. Therefore, the steps needed in Pattern Match mode are identical to those used in Normal/Extended Addressing mode configuration. An extra step is needed to store the desired pattern in DCHxDAT register.

The following steps are recommended to be taken to configure a DMA transfer in Pattern Match mode:

- Disable the DMA channel interrupts in the INT controller.
- Clear any existing channel interrupt flags in the INT controller.
- Enable the DMA controller (if not already enabled) in DMACON register.
- Set Channel Control register: Priority, Auto-Enable mode, etc., in DCHxCON. Use CHXM = `0/1` for Normal/Extended Addressing mode. Don't enable the channel yet.
- Set the channel event control: clear/set the events starting and aborting the transfer. Set the pattern match enable PATEN in DCHxECON.
- Set the pattern in the DCHxDAT register.
- Set the transfer source and destination physical addresses (DCHxSSA and DCHxDSA registers).
- If using Normal Addressing mode:
  - Set the source and destination sizes (DCHxSSIZ, DCHxDSIZ registers).
  - Set the cell transfer size (DCHxCSIZ).
- If using Extended Addressing mode:
  - Set the block transfer size (DCHxSSIZ and DCHxDSIZ).
- Clear any existing event flag in DCHxINT register.
- If using interrupts:
  - Set the conditions that will generate an interrupt in the DCHxINT register (at least error interrupt enable and abort interrupt enable, usually block complete interrupt).
  - Set the DMA channel interrupt priority and subpriority in the INT controller.
  - Enable the DMA channel interrupt in the INT controller.
- Enable the selected DMA channel with CHEN (DCHxCON<7>).
- If not using system events to start the DMA transfer use CFORCE (DCHxECON<7>) to start transfer.
- Until the DMA transfer is complete, you can do some other processing.
- If you enabled transfer complete interrupts (cell complete, block complete, etc) you'll be notified in the ISR that the DMA transfer completed.
- Otherwise, you can poll the DMA channel to see if the transfer is completed using, for example, CHBCIF (DCHxINT<3>).

Refer to Example 5-3.

EXAMPLE 5-3: CONFIGURING THE DMA FOR PATTERN MATCH OPERATION

```
/*
The following code example illustrates a DMA channel 0 configuration for the normal addressing
mode transfer with pattern match enabled. Transfer from the UART1 a <CR> ended string, at most
256 characters long
*/

       IEC1CLR=0x00010000;            // disable DMA channel 0 interrupts
       IFS1CLR=0x00010000;            // clear any existing DMA channel 0 interrupt flag

       DMACONSET=0x00008000;          // enable the DMA controller
       DCH0CON=0x03;                  // channel off, priority 3, normal mode, no chaining

       DCH0ECON=(27 <<8)|  0x30;      // start irq is UART1 RX, pattern match enabled
       DCH0DAT='\r';                  // pattern value, carriage return

                                      // program the transfer
       DCH0SSA=VirtToPhys(&U1RXREG);  // transfer source physical address
       DCH0DSA=0x1d020000;            // transfer destination physical address
       DCH0SSIZ=1;                    // source size is 1 byte
       DCH0DSIZ=0;                    // dst size at most 256 bytes
       DCH0CSIZ=1;                    // one byte per UART transfer request

       DCH0INTCLR=0x00ff00ff;         // clear existing events, disable all interrupts
       DCH0INTSET=0x00090000;         // enable Block Complete and error interrupts

       IPC9CLR=0x0000001f;            // clear the DMA channel 0 priority and subpriority
       IPC9SET=0x00000016;            // set IPL 5, subpriority 2
       IEC1SET=0x00010000;            // enable DMA channel 0 interrupt

       DCH0CONSET=0x80;               // turn channel on


       // wait for an UART RX interrupt to initiate a transfer

       // do something else

       // will get an interrupt when the transfer is done
       // or when an address error occurred
```

5.6 Channel Chaining Mode

The Chaining mode is enabled by setting the Chaining Enable it CHCEN bit (DCHxCON<5>) and Chaining Direction bit CHCHNS (DCHxCON<8>).

Channel chaining is an enhancement to the DMA channel operation.

A good usage is for transferring data packets from one peripheral to memory and then from memory to another peripheral. This module is also useful for implementing data acquisition in multiple buffers.

Chaining mode features:

- A channel (slave channel) can be chained to an adjacent channel (master channel). When the master channel completes a block transfer the slave channel will be enabled.
- At this point, any event on the slave channel will initiate a cell transfer. If the channel has an event pending, a cell transfer will begin immediately.
- Channels are chained in natural priority order where channel 0 has the highest priority and channel 3 the lowest. A specific channel can be enabled by an adjacent channel, either higher, or lower, in natural order, by configuring the CHCHNS (DCHxCON<8>) bit. Chaining must be enabled, CHCHN (DCHxCON<5>) = `1`.
- An important feature of the DMA controller is the ability to allow events while the channel is disabled using the CHAED (DCHxCON<6>) bit. This bit is particularly useful in Chained mode where the slave channel needs to be ready to start a transfer as soon as the channel is enabled by the master channel.

5.6.1 CHAINING MODE CONFIGURATION

The Chaining mode is an option for use when performing DMA transfers in Normal or Extended Addressing modes. Therefore, the steps needed in Chaining mode are identical to those used in Normal/Extended Addressing mode configuration, with the following differences (refer to **Section 5.3.1 “Normal Addressing Mode Transfer Configuration”**):

- Two diffferent channels have to be configured and the slave channel has to have chaining enable (CHCHN) and chaining direction (CHCHNS) set.

Refer to Example 5-4.

5.7 Channel Auto-Enable Mode

The Auto-Enable mode is enabled by setting the CHAEN bit (DCHxCON<4>).

Channel auto-enable function is an enhancement to the DMA channel operation.

The channel auto-enable can be used to keep a channel active, even if a block transfer completes or a pattern match occurs. This prevents the user from having to re-enable the channel each time a block transfer completes. This mode is useful for applications that do repeated pattern matching.

5.7.1 AUTO-ENABLE MODE CONFIGURATION

The Auto-Enable mode is an extra option for use when performing DMA transfers in Normal or Extended Addressing modes. Therefore, the steps needed in Auto-Enable mode are identical to those used in Normal/Extended Addressing mode configuration, with the following differences (refer to **Section 5.3.1 “Normal Addressing Mode Transfer Configuration”**):

- The CHAEN bit has to be set before enabling the channel (setting the CHEN bit (DCHxCON<7>)).
- The channel will behave as normal except that normal termination of a transfer will not result in the channel being disabled.
- Normal block transfer completion is defined as:
  - block transfer complete
  - pattern match detect
- As before, the Channel Pointers will be reset.

EXAMPLE 5-4: CONFIGURING THE DMA FOR CHAINING MODE OPERATION

```
/*
The following code example illustrates a DMA channel 0 configuration for the normal addressing
mode transfer with pattern match enabled. DMA channel 0 transfer from the UART1 to a RAM buffer
while DMA channel 1 transfers data from the RAM buffer to UART2. Transferred strings are at most
256 characters long. Transfer on UART2 will start as soon as the UART1 transfer is completed.
*/

   unsigned char myBuff<256>;// transfer buffer

  IEC1CLR=0x00010000;         // disable DMA channel 0 interrupts
  IFS1CLR=0x00010000;         // clear any existing DMA channel 0 interrupt flag

  DMACONSET=0x00008000;       // enable the DMA controller

  DCH0CON=0x3;                // channel 0 off, priority 3, normal mode, no chaining
  DCH1CON=0x62;               // channel 1 off, priority 2, normal mode,
                              // chain to higher priority
                              // (ch      0), enable events detection while disabled

  DCH0ECON=(27 <<8)| 0x30;    // start irq is UART1 RX, pattern enabled
  DCH1ECON=(42 <<8)| 0x30;    // start irq is UART1 TX, pattern enabled

  DCH0DAT=DCH1DAT='\r';       // pattern value, carriage return

                              // program channel 0 transfer
  DCH0SSA=VirtToPhys(&U1RXREG); // transfer source physical address
  DCH0DSA=VirtToPhys(myBuff); // transfer destination physical address
  DCH0SSIZ=1;                 // source size is 1 byte
  DCH0DSIZ=0;                 // dst size at most 256 bytes
  DCH0CSIZ=1;                 // one byte per UART transfer request

                              // program channel 1 transfer
  DCH1SSA=VirtToPhys(myBuff); // transfer source physical address
  DCH1DSA=VirtToPhys(&U2TXREG); // transfer destination physical address
  DCH1SSIZ=0;                 // source size at most 256 bytes
  DCH1DSIZ=0;                 // dst size is 1 byte
  DCH1CSIZ=1;                 // one byte per UART transfer request

  DCH0INTCLR=0x00ff00ff;      // DMA0: clear events, disable interrupts
  DCH1INTCLR=0x00ff00ff;      // DMA1: clear events, disable interrupts
  DCH1INTSET=0x00090000;      // DMA1: enable Block Complete and error interrupts


  IPC9CLR=0x00001f1f;         // clear the DMA channels 0 and 1 priority and
                              // subpriority
  IPC9SET=0x00000b16;         // set IPL 5, subpriority 2 for DMA channel 0
                              // set IPL 2, subpriority 3 for DMA channel 1
  IEC1SET=0x00020000;         // enable DMA channel 1 interrupt

  DCH0CONSET=0x80;            // turn channel on

  // do something else

  // the UART1 RX interrupts will initiate the DMA channel 0 transfer
  // once this transfer is complete, the DMA channel 1 will start
  // upon DMA channel 1 transfer completion will get an interrupt

  while(!intCh1Ocurred);      // poll DMA channel 1 interrupt
```

5.8 CRC Module Operation

The DMA module has one integrated CRC generation module shared by all channels. The CRC module is a highly configurable, 16-bit CRC generator. The CRC module can be assigned to any available DMA channel by setting the CRCCH bits (DCRCCON<1:0>) appropriately. The CRC is enabled by setting the CRCEN bit (DCRCCON<7>).

The CRC generator will take 1 system clock to process each byte of data read from the source. This implies that if 32 bits of data are read from the source, the CRC generation will take 4 system clocks to process the data.

The CRC module modifies the behavior of the DMA channel associated with the CRC module. The two operating modes for a DMA channel associated with the CRC module are:

- Background Mode: CRC is calculated in the background, with normal DMA behavior maintained.
- Append Mode: Data read from the source is not written to the destination, but the CRC data is accumulated in the CRC data register. The accumulated CRC is written to the destination address when a block transfer completes.

CRC Configurable resources:

- The terms of the polynomial can be programmed using the DCRCXOR<15:0> bits. Considering the CRC polynomial: $x^{16} + x^{12} + x^5 + 1$, 17 bits are needed to define this polynomial. However, the value to be written to the DCRCXOR register will be `0b0001 0000 0010 0000`, i.e., 0x1020.

Note:	The LSb and MSb do not have to be specified, they are always set. The actual value used for the polynomial generator will be 0x11021.

- The length of the polynomial generator can be programmed using the PLEN (DCRCCON<12:8>) bits. For the above polynomial, the size will be 16. The PLEN will be programmed with length -1, i.e., 0x0F.
- The CRC module can be assigned to any available DMA channel by setting the CRCCH bits (DCRCCON<2:0>) appropriately.
- The CRC is enabled by setting the CRCEN bit (DCRCCON<7>).
- The CRC generator can be seeded by writing to the DCRCDATA register before enabling the channel that will use the CRC module.
- The CRC can be read as it progresses by reading the DCRCDATA register at any time during the CRC generation.
- Data Order: As data is read from the source register, the data is fed into the CRC generator MSB first.

5.9 CRC Background Mode

The CRC Background mode is enabled by clearing CRCAPP (DCRCCON<6>).

In this mode, the behavior of the DMA channel is maintained with data read from the channel source being passed to the CRC module and then written back to the destination.

In the Background mode, the calculated CRC is left in the DCRCDATA register at the end of the block transfer.

This mode can be used to calculate a CRC as data is moved from source to destination. A good example of where this can be used is to calculate a CRC as data is transmitted to or received from the UART module. When the data transfer is complete the user can read the calculated CRC and either append it to the transmitted data or verify the received CRC data.

5.9.1 CRC BACKGROUND MODE CONFIGURATION

Microchip recommends taking the following steps to configure a CRC calculation in Background mode:

- Seed the CRC generator by writing the initial seed to the DCRCDATA register.
- Set the polynomial generator by writing to the DCRCXOR register.
- Set the polynomial generator length by writing the PLEN (DCRCCON<12:8>).
- Attach the CRC calculation to the desired DMA channel performing the Normal/Extended Addressing mode transfer by writing the CRCCH (DCRCCON<2:0>).
- Use the Background mode by clearing the CRCAPP (DCRCCON<6>) bit.
- Enable the CRC calculation by setting the CRCEN (DCRCCON<7>).
- Once the DMA transfer begins, the CRC calculation will begin as well.
- Once the DMA transfer ends, the CRC result will be available by reading the DCRCDATA register.

Refer to Example 5-5..

Note:	The configuration steps specific for the CRC configuration are shown. The DMA transfer configuration is the same as previously explained (see **Section 5.2 “DMA Controller Operation”**).

EXAMPLE 5-5: CRC BACKGROUND MODE OPERATION

```
/*
The following code example illustrates a DMA calculation using the CRC background mode. Data is
transferred from a 12K bytes Flash buffer to a RAM buffer and the CRC is calculated while the
transfer takes place. */

 unsigned int blockCrc;           // CRC of the flash block

 IEC1CLR=0x00010000;              // disable DMA channel 0 interrupts
 IFS1CLR=0x00010000;              // clear any existing DMA channel 0 interrupt flag


 DMACONSET=0x00008000;            // enable the DMA controller

 DCRCDATA=0xffff;                 // seed the CRC generator
 DCRCXOR=0x1021;                  // Use the standard CCITT CRC 16 polynomial: X^16+X^12+X^5+1
 DCRCCON=0x0f80;                  // CRC enabled, polynomial length 16, background mode
                                  // CRC attached to the DMA channel 0.


 DCH0CON=0x0b;                    // channel off, priority 3, extended mode, no chaining
 DCH0ECON=0;                      // no start irqs, no match enabled

                                  // program channel transfer
 DCH0SSA=VirtToPhys(flashBuff);   // transfer source physical address
 DCH0DSA=VirtToPhys(ramBuff);     // transfer destination physical address
 DCH0SSIZ=(BYTE)((12*1024)>>8);   // source size takes MSB
 DCH0DSIZ=(BYTE)((12*1024));      // dst size takes LSB

 DCH0INTCLR=0x00ff00ff;           // DMA0: clear events, disable interrupts

 DCH0CONSET=0x80;                 // channel 0 on

                                  // initiate a transfer
 DCH0ECONSET=0x00000080;          // set CFORCE to 1

 // do something else while the transfer takes place

 // poll to see that the transfer was done

 BOOL error=FALSE;
 while(TRUE)
 {
       register int pollCnt;      // don't poll in a tight loop
       int dmaFlags=DCH0INT;
       if( (dmaFlags& 0x3)
       {                          // CHERIF (DCHxINT<0>) or CHTAIF (DCHxINT<1> set
              error=TRUE;         // error or aborted...
              break;
              }
                 else if (dmaFlags&0x8)
              {                   // CHBCIF (DCHxINT<3>) set
              break;              // transfer completed normally
       }
       pollCnt=100;               // use an adjusted value here
       while(pollCnt--);          // wait before polling again
 }

 if(!error)
 {
       blockCrc=DCRDATA;          // read the CRC of the transferred flash block
 }
 else
 {
                                  // process error
 }
```

5.10 CRC Append Mode

The CRC Append mode is enabled by setting CRCAPP (DCRCCON<6>).

In this mode, the behavior of the DMA channel is changed.

Data read from the source will be fed into the CRC generation module. No data is written to the destination address in CRC Append mode until a block transfer completes or a pattern match occurs. On completion, the CRC value will be written to the address given by the Destination register (DCHxDSA).

This mode can be used for the CRC calculation of a memory buffer, without actually performing a DMA transfer to a destination.

CRC Append mode Features:

- Only the source is considered when deciding if a block transfer is complete.
- The destination address (DCHxDSA) is only used as the location to write the generated CRC to.
- The destination size (DCHxDSIZ) can have a maximum size of 4.
  - If DCHxDSIZ is greater than 4, only 4 bytes are written at the end of the transfer.
  - If DCHxDSIZ is less than 4, only DCHxDSIZ bytes of the CRC are written to the destination address.
  - The high bytes (bits 31:16) are written as 0's if more than 16 bits of the CRC are written.
  - PLEN (CRCCON<12:8>) has no effect on the number of CRC bits that will be written to the Destination register.
  - When Extended Addressing mode is enabled, DCHxDSIZ forms part of the size of the data block to be transferred. DCHxDSIZ is not available to be used to limit number of bytes of the CRC to be stored and the entire 32-bit CRC will be written to DCHxDSA when a block transfer completes.
- No CRC written back on an abort IRQ, user abort, bus error, etc.

5.10.1 CRC APPEND MODE CONFIGURATION

Microchip recommends taking the following steps to configure a CRC calculation in Background mode:

- Seed the CRC generator by writing the initial seed to the DCRCDATA register.
- Set the polynomial generator by writing to the DCRCXOR register.
- Set the polynomial generator length by writing the PLEN (DCRCCON<12:8>).
- Attach the CRC calculation to the desired DMA channel performing the Normal/Extended Addressing mode transfer by writing the CRCCH (DCRCCON<2:0>).
- Use the Append mode by setting the CRCAPP (DCRCCON<6>) bit.
- Enable the CRC calculation by setting the CRCEN (DCRCCON<7>).
- Program the DMA transfer destination with the physical address of a variable where the CRC is to be stored.
- Once the DMA transfer begins, the CRC calculation will begin as well.
- Once the DMA transfer ends, the CRC result will be deposited at the programmed DMA destination address.

Refer to Example 5-6.

Note:	The configuration steps specific for the CRC configuration are shown. The DMA transfer configuration is the same as previously explained (see **Section 5.2 "DMA Controller Operation"**).

EXAMPLE 5-6: CRC APPEND MODE OPERATION

```
/*
The following code example illustrates a DMA calculation using the CRC append mode. The CRC of a
12K bytes flash buffer is calculated without performing any data transfer. As soon as the CRC
calculation is completed the CRC value of the flash buffer is available in a local variable for
further use. */

 unsigned int blockCrc;            // CRC of the flash block

 IEC1CLR=0x00010000;               // disable DMA channel 0 interrupts
 IFS1CLR=0x00010000;               // clear any existing DMA channel 0 interrupt flag


 DMACONSET=0x00008000;             // enable the DMA controller

 DCRCDATA=0xffff;                  // seed the CRC generator
 DCRCXOR=0x1021;                   // Use the standard CCITT CRC 16 polynomial: X^16+X^12+X^5+1
 DCRCCON=0x0fc0;                   // CRC enabled, polynomial length 16, append mode
                                   // CRC attached to the DMA channel 0.


 DCH0CON=0x0b;                     // channel off, priority 3, extended mode, no chaining
 DCH0ECON=0;                       // no start irqs, no match enabled

                                   // program channel transfer
 DCH0SSA=VirtToPhys(flashBuff);    // transfer source physical address
 DCH0DSA=VirtToPhys(&blockCrc);    // transfer destination physical address
 DCH0SSIZ=(BYTE)((12*1024)>>8);    // source size takes MSB
 DCH0DSIZ=(BYTE)((12*1024));       // dst size takes LSB

 DCH0INTCLR=0x00ff00ff;            // DMA0: clear events, disable interrupts
 DCH1INTCLR=0x00ff00ff;            // DMA1: clear events, disable interrupts

 DCH0CONSET=0x80;                  // channel 0 on

                                   // initiate a transfer
 DCH0ECONSET=0x00000080;           // set CFORCE to 1

 // do something else while the CRC calculation takes place

 // poll to see that the transfer was done
 BOOL error=FALSE;
 while(TRUE)
 {
       register int pollCnt;       // don't poll in a tight loop
       int dmaFlags=DCH0INT;
       if( (dmaFlags& 0x3)
       {                           // CHERIF (DCHxINT<0>) or CHTAIF (DCHxINT<1> set
              error=TRUE;          // error or aborted...
              break;
              }
                     else if (dmaFlags&0x8)
              {                    // CHBCIF (DCHxINT<3>) set
                     break;        // transfer completed normally
       }
       pollCnt=100;                // use an adjusted value here
       while(pollCnt--);           // wait before polling again
 }

 if(error)
 {
                          // process error
 }

 // the block CRC is available in the blockCrc variable
```

5.11 DMA Interrupts

The DMA device has the ability to generate interrupts reflecting the events that occurr during the channel's data transfer. The different kinds of DMA interrupt flags are:

- CHERIF (DCHxINT<0>): Channel Error interrupts, enabled using CHERIE (DCHxINT<16>).
- CHTAIF (DCHxINT<1>): Channel Abort interrupts, enabled using CHTAIE (DCHxINT<17>).
- CHBCIF (DCHxINT<3>): Channel Block complete interrupts, enabled using CHBCIE (DCHxINT<19>).
- CHCCIF (DCHxINT<2>): Channel Cell complete interrupts, enabled using CHCCIE (DCHxINT<18>).
- CHSDIF (DCHxINT<7>): Channel Source pointer reached the end of the source, enabled by CHSDIE (DCHxINT<23>).
- CHSHIF (DCHxINT<6>): Channel Source pointer reached midpoint of the source, enabled by CHSHIE (DCHxINT<22>).
- CHDDIF (DCHxINT<5>): Channel Destination Pointer reached the end of the destination, enabled by CHDDIE (DCHxINT<21>)
- CHDHIF (DCHxINT<4>): Channel Destination Pointer reached midpoint of the destination, enabled by CHDHIE (DCHxINT<20>).

All the interrupts belonging to a DMA channel map to the corresponding channel interrupt vector.

The corresponding interrupt flags are:

- DMA0IF (IFS1<16>)
- DMA1IF (IFS1<17>)
- DMA2IF (IFS1<18>)
- DMA3IF (IFS1<19>)

All these interrupt flags must be cleared in software.

A DMA channel is enabled as a source of interrupts via the respective DMA interrupt enable bits:

- DMA0IE (IEC1<16>)
- DMA1IE (IEC1<17>)
- DMA2IE (IEC1<18>)
- DMA3IE (IEC1<19>)

The interrupt priority level bits and interrupt subpriority level bits must be also be configured:

- DMA0IP<2:0> (IPC9<4:2>), DMA0IS<1:0> (IPC9<1:0>).
- DMA1IP<2:0> (IPC9<12:10>), DMA1IS<1:0> (IPC9<9:8>).
- DMA2IP<2:0> (IPC9<20:18>), DMA2IS<1:0> (IPC9<17:16>).
- DMA3IP<2:0> (IPC9<28:26>), DMA3IS<1:0> (IPC9<25:24>).

In addition to enabling the DMA interrupts, Interrupt Service Routines (ISRs) are required for each different interrupt vector used. See Example 5-7 and Example 5-8.

Note:	It is the user's responsibility to clear the corresponding interrupt flag bit before returning from an ISR.

EXAMPLE 5-7: DMA INITIALIZATION WITH INTERRUPTS

```
/*
The following code example illustrates a DMA channel 0 interrupt configuration.
When the DMA channel 0 interrupt is generated, the CPU will jump to the vector assigned to DMA0
interrupt.
*/

       IEC1CLR=0x00010000;          // disable DMA channel 0 interrupts
       IFS1CLR=0x00010000;          // clear any existing DMA channel 0 interrupt flag

       DMACONSET=0x00008000;        // enable the DMA controller
       DCH0CON=0x03;                // channel off, priority 3, normal mode, no chaining

       DCH0ECON=0;                  // no start or stop irq's, no pattern match

                                    // program the transfer
       DCH0SSA=0x1d010000;          // transfer source physical address
       DCH0DSA=0x1d020000;          // transfer destination physical address
       DCH0SSIZ=0;                  // source size 256 bytes
       DCH0DSIZ=0;                  // destination size 256 bytes
       DCH0CSIZ=0;                  // 256 bytes transferred pe event

       DCH0INTCLR=0x00ff00ff;       // clear existing events, disable all interrupts
       DCH0INTSET=0x00090000;       // enable Block Complete and error interrupts

       IPC9CLR=0x0000001f;          // clear the DMA channel 0 priority and subpriority
       IPC9SET=0x00000016;          // set IPL 5, subpriority 2
       IEC1SET=0x00010000;          // enable DMA channel 0 interrupt

       DCH0CONSET=0x80;             // turn channel on
                                    // initiate a transfer
       DCH0ECONSET=0x00000080;      // set CFORCE to 1

       // do something else

       // will get an interrupt when the block transfer is done
       // or when error occurred
```

EXAMPLE 5-8: DMA CHANNEL 0 ISR

```
/*
The following code example demonstrates a simple Interrupt Service Routine for DMA channel 0
interrupts. The user's code at this vector should perform any application specific operations
and must clear the DMA0 interrupt flags before exiting.
*/

void __ISR(_DMA0_VECTOR, IPL5) __DMA0Interrupt(void)
{
       int dmaFlags=DCH0INT&0xff;     // read the interrupt flags

       // perform application specific operations in response to any interrupt flag set

       DCH0INTCLR=0x000000ff;         // clear the DMA channel interrupt flags
       IFS1CLR = 0x00010000;          // Be sure to clear the DMA0 interrupt flags
                                      // before exiting the service routine.
}
```

Note: The DMA ISR code example shows MPLAB® C32 C compiler specific syntax. Refer to your compiler manual regarding support for ISRs.

5.12 I/O Pin Control

The DMA controller module does not use any I/O pins.

6.0 MEMORY ORGANIZATION

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *“PIC32MX Family Reference Manual”* (DS61132) for a detailed description of this peripheral.

The PIC32MX microcontrollers provides 4 GB of unified virtual memory address space. All memory regions including program, data memory, SFRs, and Configuration registers reside in this address space at their respective unique addresses. The program and data memories can be optionally partitioned into user and kernel memories. In addition, the data memory can be made executable, allowing PIC32MX to execute from data memory.

Key Features:

- 32-bit native data width
- Separate User and Kernel mode address space
- Flexible program Flash memory partitioning
- Flexible data RAM partitioning for data and program space
- Separate boot Flash memory for protected code
- Robust bus exception handling to intercept run-away code.
- Simple memory mapping with Fixed Mapping Translation (FMT) unit
- Cacheable and non-cacheable address regions

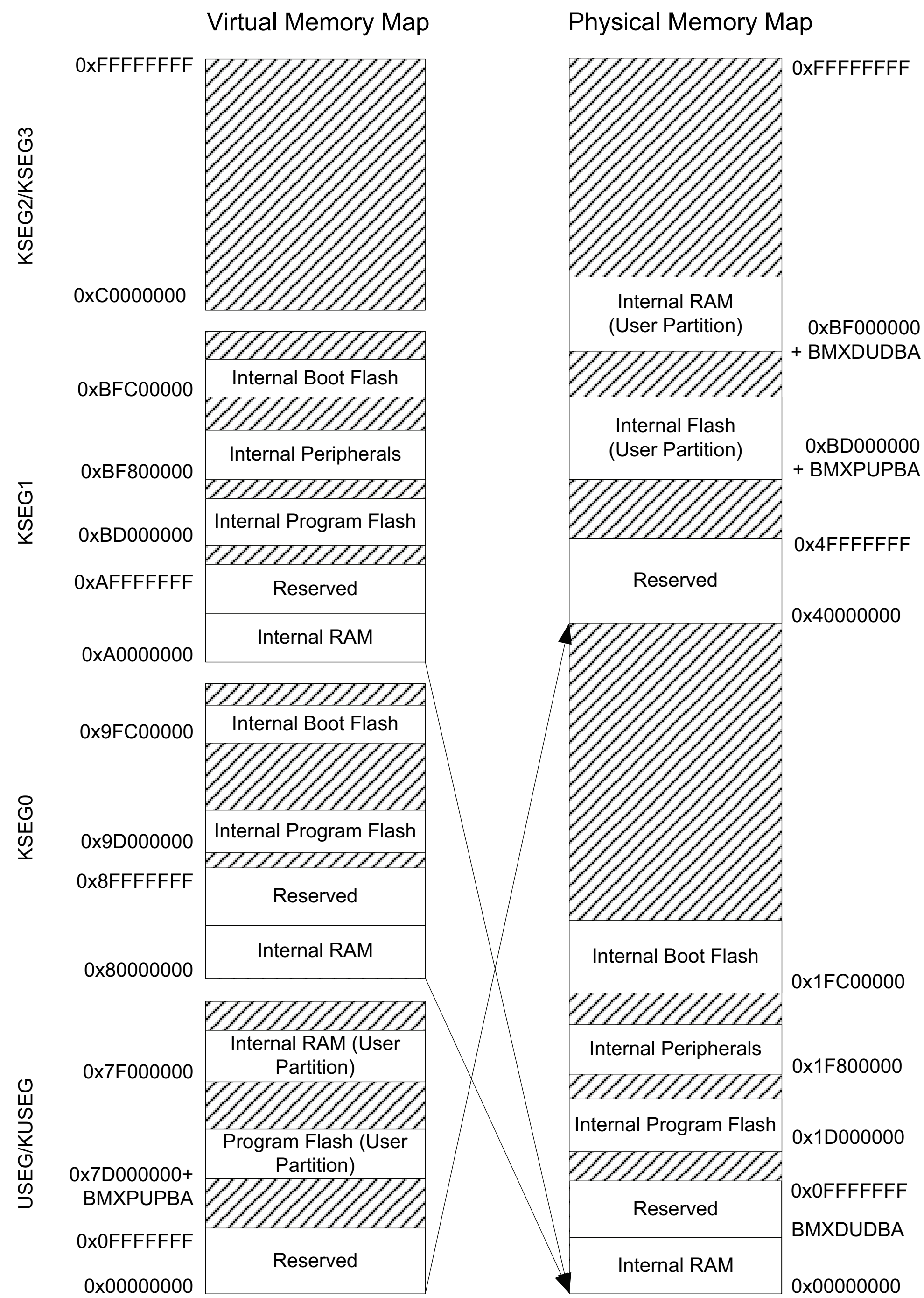
6.1 PIC32MX Memory Layout

The PIC32MX microcontrollers implement two address spaces: Virtual and Physical. All hardware resources such as program memory, data memory and peripherals are located at their respective physical addresses. Virtual addresses are exclusively used by the CPU to fetch and execute instructions as well as access peripherals. Physical addresses are used by peripherals such as DMA and Flash controller that access memory independently of CPU.

The entire 4 GB virtual address space is divided into two primary regions – user and kernel space. The lower 2 GB of space forms the User mode segment, called useg/kuseg. The upper 2 GB of virtual address space forms the kernel-only space. The kernel space is divided into four segments of 512 MB each: kseg 0, kseg 1, kseg 2 and kseg 3. Only Kernel mode applications can access kernel space memory. The peripheral registers are only visible through kernel space.

The Fixed Mapping Translation (FMT) unit translates the memory segments into corresponding physical address regions. A virtual memory segment may also be cached, provided the cache module is available on the device. Please note that the kseg 1 memory segment is not cacheable, while kseg 0 and useg/kuseg are cacheable.

FIGURE 6-1: VIRTUAL TO PHYSICAL FIXED MEMORY MAPPING



 Advance Information

6.2 Bus Matrix Registers

TABLE 6-1: BUS MATRIX REGISTER SUMMARY

Virtual Address	Name		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
BF88_2000	BMXCON	31:24	—	—	—	—	—	BMX-CHEDMA	—	—
		23:16	—	—	—	BMXERRIXI	BMXERRICD	BMXER-RDMA	BMXERRDS	BMXERRIS
		15:8	—	—	—	—	—	—	—	—
		7:0	—	BMXWS-DRM	—	—	—	BMXARB<2:0>		
BF88_2004	BMXCONCLR	31:0	Write clears selected bits in BMXCON, read yields undefined value							
BF88_2008	BMXCONSET	31:0	Write sets selected bits in BMXCON, read yields undefined value							
BF88_200C	BMXCONINV	31:0	Write inverts selected bits in BMXCON, read yields undefined value							
BF88_2010	BMXDKPBA	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	BMXDKPBA<15:8>							
		7:0	BMXDKPBA<7:0>							
BF88_2014	BMX-DKPBACLR	31:0	Write clears selected bits in BMXDKPBA, read yields undefined value							
BF88_2018	BMX-DKPBASET	31:0	Write sets selected bits in BMXDKPBA, read yields undefined value							
BF88_201C	BMX DKPBAINV	31:0	Write inverts selected bits in BMXDKPBA, read yields undefined value							
BF88_2020	BMXDUDBA	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	BMXDUDBA<15:8>							
		7:0	BMXDUDBA<7:0>							
BF88_2024	BMX-DUDBACLR	31:0	Write clears selected bits in BMXDUDBA, read yields undefined value							
BF88_2028	BMX-DUDBASET	31:0	Write sets selected bits in BMXDUDBA, read yields undefined value							
BF88_202C	BMX-DUDBAINV	31:0	Write inverts selected bits in BMXDUDBA, read yields undefined value							
BF88_2030	BMX DUPBA	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	BMXDUPBA<15:8>							
		7:0	BMXDUPBA<7:0>							
BF88_2034	BMX DUPBACLR	31:0	Write clears selected bits in BMXDUPBA, read yields undefined value							
BF88_2038	BMX DUPBASET	31:0	Write sets selected bits in BMXDUPBA, read yields undefined value							
BF88_203C	BMX DUPBAINV	31:0	Write inverts selected bits in BMXDUPBA, read yields undefined value							
BF88_2040	BMXDRMSZ	31:24	BMXDRMSZ<31:24>							
		23:16	BMXDRMSZ<23:16>							
		15:8	BMXDRMSZ<15:8>							
		7:0	BMXDRMSZ<7:0>							
BF88_2044	BMXPUPBA	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	BMXPUPBA<19:16>			
		15:8	BMXPUPBA<15:8>							
		7:0	BMXPUPBA<7:0>							
BF88_2048	BMXPFMSZ	31:24	BMXPFMSZ<31:24>							
		23:16	BMXPFMSZ<23:16>							
		15:8	BMXPFMSZ<15:8>							
		7:0	BMXPFMSZ<7:0>							
BF88_204C	BMXBOOTSZ	31:24	BMXBOOTSZ<31:24>							
		23:16	BMXBOOTSZ<23:16>							
		15:8	BMXBOOTSZ<15:8>							
		7:0	BMXBOOTSZ<7:0>							

REGISTER 6-1: BMXCON: BUS MATRIX CONFIGURATION REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0	U-0	U-0
—	—	—	—	—	BMX-CHEDMA	—	—
bit 31							bit 24

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	BMXERRIXI	BMXER-RICD	BMXER-RDMA	BMXER-RDS	BMXERRIS
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-1	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	BMXWS-DRM	—	—	—	BMXARB<2:0>		
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-27 **Unimplemented:** Read as ‘0’

bit 26 **BMXCHEDMA:** BMX PFM Cacheability for DMA Accesses bit
1 = Enable program Flash memory (data) cacheability for DMA accesses (requires cache to have data caching enabled)
0 = Disable program Flash memory (data) cacheability for DMA accesses (hits are still read from the cache, but misses do not update the cache)

bit 25-21 **Unimplemented:** Read as ‘0’

bit 20 **BMXERRIXI:** Enable Bus Error from IXI bit
1 = Enable bus error exceptions for unmapped address accesses initiated from IXI shared bus
0 = Disable bus error exceptions for unmapped address accesses initiated from IXI shared bus

bit 19 **BMXERRICD:** Enable Bus Error from ICD Debug Unit bit
1 = Enable bus error exceptions for unmapped address accesses initiated from ICD
0 = Disable bus error exceptions for unmapped address accesses initiated from ICD

bit 18 **BMXERRDMA:** Bus Error from DMA bit
1 = Enable bus error exceptions for unmapped address accesses initiated from DMA
0 = Disable bus error exceptions for unmapped address accesses initiated from DMA

bit 17 **BMXERRDS:** Bus Error from CPU Data Access bit (disabled in Debug mode)
1 = Enable bus error exceptions for unmapped address accesses initiated from CPU data access
0 = Disable bus error exceptions for unmapped address accesses initiated from CPU data access

bit 16 **BMXERRIS:** Bus error from CPU Instruction Access bit (disabled in Debug mode)
1 = Enable bus error exceptions for unmapped address accesses initiated from CPU instruction access
0 = Disable bus error exceptions for unmapped address accesses initiated from CPU instruction access

bit 15-7 **Unimplemented:** Read as ‘0’

REGISTER 6-1: BMXCON: BUS MATRIX CONFIGURATION REGISTER (CONTINUED)

bit 6	**BMXWSDRM:** CPU Instruction or Data Access from Data RAM Wait State bit 1 = Data RAM accesses from CPU have one wait state for address setup 0 = Data RAM accesses from CPU have zero wait states for address setup
bit 5-3	**Unimplemented:** Read as '0'
bit 2-0	**BMXARB<2:0>:** Bus Matrix Arbitration Mode bits 111...011 = Reserved (using these Configuration modes will produce undefined behavior) 010 = Arbitration Mode 2 001 = Arbitration Mode 1 000 = Arbitration Mode 0

REGISTER 6-2: BMXDKPBA: DATA RAM KERNEL PROGRAM BASE ADDRESS REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-0
BMXDKPBA<15:8>							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
BMXDKPBA<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-16	**Unimplemented:** Read as ‘0’
bit 15-11	**BMXDKPBA<15:11>:** DRM Kernel Program Base Address bits When non-zero, this value selects the relative base address for kernel program space in RAM
bit 10-0	**BMXDKPBA<10:0>:** Read-Only bits Value is always ‘0’, which forces 2 KB increments

REGISTER 6-3: BMXDUDBA: DATA RAM USER DATA BASE ADDRESS REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-0
BMXDUDBA<15:8>							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
BMXDUDBA<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-11 **BMXDUDBA<15:11>:** DRM User Data Base Address bits
When non-zero, the value selects the relative base address for User mode data space in RAM

Note: If non-zero, the value must be greater than BMXDKPBA.

bit 10-0 **BMXDUDBA<10:0>:** Read-Only bits
Value is always ‘0’, which forces 2 KB increments

REGISTER 6-4: BMXDUPBA: DATA RAM USER PROGRAM BASE ADDRESS REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-0
BMXDUPBA<15:8>							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
BMXDUPBA<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-11 **BMXDUPBA<15:11>:** DRM User Program Base Address bits
When non-zero, the value selects the relative base address for User mode program space in RAM

Note: If non-zero, BMXDUPBA must be greater than BMXDUDBA.

bit 10-0 **BMXDUPBA<10:0>:** Read-Only bits
Value is always ‘0’, which forces 2 KB increments

REGISTER 6-5: BMXDRMSZ: DATA RAM SIZE REGISTER

R	R	R	R	R	R	R	R
BMXDRMSZ<31:24>							
bit 31							bit 24

R	R	R	R	R	R	R	R
BMXDRMSZ<23:16>							
bit 23							bit 16

R	R	R	R	R	R	R	R
BMXDRMSZ<15:8>							
bit 15							bit 8

R	R	R	R	R	R	R	R
BMXDRMSZ<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-0 **BMXDRMSZ:** Data RAM Memory (DRM) Size bits

Static value that indicates the size of the Data RAM in bytes:
.......0x00002000 = device has 8 KB RAM
0x00004000 = device has 16 KB RAM
0x00008000 = device has 32 KB RAM

REGISTER 6-6: BMXPUPBA: PROGRAM FLASH (PFM) USER PROGRAM BASE ADDRESS REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	BMXPUPBA<19:16>			
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-0
BMXPUPBA<15:8>							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
BMXPUPBA<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-20 Unimplemented: Read as ‘0’

bit 19-11 **BMXPUPBA<19:11>:** Program Flash (PFM) User Program Base Address bits
When non-zero, this value selects the PFM relative base address for User mode program space.

bit 10-0 **BMXPUPBA<10:0>:** Read-Only bits
Value is always ‘0’, which forces 2 KB increments

REGISTER 6-7: BMXPFMSZ: PROGRAM FLASH (PFM) SIZE REGISTER

R	R	R	R	R	R	R	R
BMXPFMSZ<31:24>							
bit 31							bit 24

R	R	R	R	R	R	R	R
BMXPFMSZ<23:16>							
bit 23							bit 16

R	R	R	R	R	R	R	R
BMXPFMSZ<15:8>							
bit 15							bit 8

R	R	R	R	R	R	R	R
BMXPFMSZ<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-0 **BMXPFMSZ:** Program Flash Memory (PFM) Size bits

Static value that indicates the size of the PFM in bytes:

0x00008000 = device has 32 KB Flash
0x00010000 = device has 64 KB Flash
0x00020000 = device has 128 KB Flash
0x00040000 = device has 256 KB Flash
0x00080000 = device has 512 KB Flash

REGISTER 6-8: BMXBOOTSZ: BOOT FLASH (IFM) SIZE REGISTER

R	R	R	R	R	R	R	R
BMXBOOTSZ<31:24>							
bit 31							bit 24

R	R	R	R	R	R	R	R
BMXBOOTSZ<23:16>							
bit 23							bit 16

R	R	R	R	R	R	R	R
BMXBOOTSZ<15:8>							
bit 15							bit 8

R	R	R	R	R	R	R	R
BMXBOOTSZ<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-0 **BMXBOOTSZ:** Boot Flash Memory (BFM) Size bits

Static value that indicates the size of the Boot PFM in bytes:

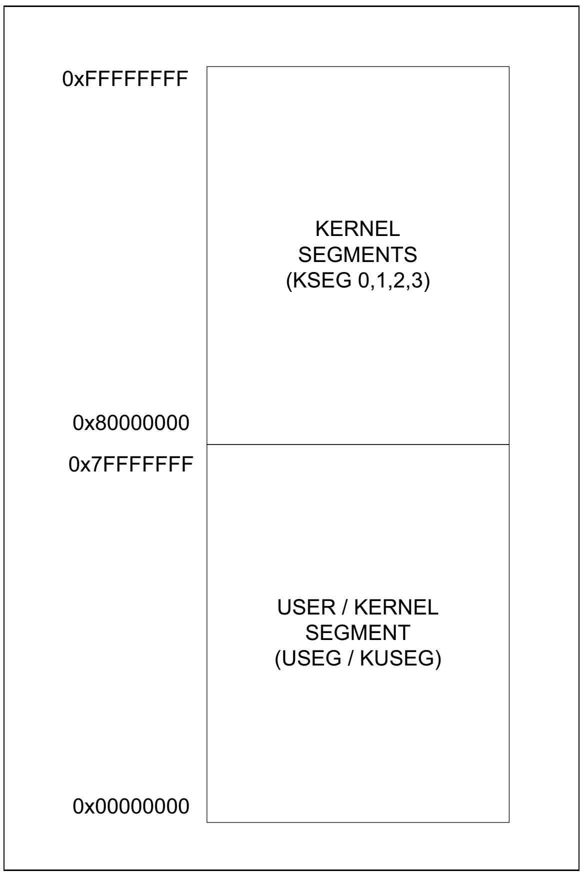
0x00003000 = device has 12 KB boot Flash

6.3 User and Kernel Memory Areas

There are two modes of operation of the PIC32MX core: User mode and Kernel mode. To support these modes, the virtual address space is also divided into two segments, kernel segments and user segments. The lower 2 GBytes of virtual addresses form the User mode partition, and the upper 2 GBytes forms the Kernel mode partition.

Most application will run only in Kernel mode. For these applications, the entire program can reside in the kernel address space providing full access to all resources.

FIGURE 6-2: USER/KERNEL ADDRESS SEGMENTS



6.4 PIC32MX Address Map

Table 6-2 shows the address map of the PIC32MX microcontroller.

On reset, the PIC32MX starts executing code from 0xBFC0_0000 virtual address which reside in the kseg1 segment (non cacheable segment).

6.4.1 PHYSICAL MEMORY ADDRESS

The Kernel Program Flash address space starts at physical address 0x1D000000, whereas the user program flash space starts at physical address 0xBD000000 + BMXPUPBA register value.

Similarly, the internal RAM is also divided into Kernel and User partitions. The kernal RAM space starts at physical address 0x00000000, whereas the User RAM space starts at physical address 0xBF000000 + BMXDUDBA register value.

By default the entire Flash memory and RAM are mapped to the Kernel mode application only.

TABLE 6-2: PIC32MX ADDRESS MAP

		Virtual Addresses		Physical Addresses		Size in Bytes
	Memory Type	**Begin Address**	**End Address**	**Begin Address**	**End Address**	**Calculation**
Kernel Address Space	**Boot Flash**	0xBFC00000	0xBFC02FFF	0x1FC00000	0x1FC02FFF	12 KB
	Program Flash(1)	0xBD000000	0xBD000000 + BMXPUPBA - 1	0x1D000000	0x1D00000 + BMXPUPBA - 1	BMXPUPBA
	Program Flash(2)	0x9D000000	0x9D000000 + BMXPUPBA - 1	0x1D000000	0x1D000000 + BMXPUPBA - 1	BMXPUPBA
	RAM (Data)	0x80000000	0x80000000 + BMXDKPBA - 1	0x00000000	BMXDKPBA - 1	BMXDKPBA
	RAM (Prog)	0x80000000 + BMXDKPBA	0x80000000 + BMXDUDBA -1	BMXDKPBA	BMXDUDBA -1	BMXDUDBA - BMXDKPBA
	Peripheral	0xBF800000	0xBF8FFFFF	0x1F800000	0x1F8FFFFF	1 MB
User Address Space	**Program Flash**	0x7D000000 + BMXPUPBA	0x7D000000 + PFM Size - 1	0xBD000000 + BMXPUPBA	0xBD000000 + PFM Size - 1	PFM Size - BMXPUPBA
	RAM (Data)	0x7F000000 + BMXDUDBA	0x7F000000 + BMXDUPBA - 1	0xBF000000 + BMXDUDBA	0xBF000000 + BMXDUPBA - 1	BMXDUPBA - BMXDUDBA
	RAM (Prog)	0x7F000000 + BMXDUPBA	0x7F000000 + RAM Size(3) - 1	0xBF000000 + BMXDUPBA	0xBF000000 + RAM Size(3) - 1	DRM Size - BMXDUPBA

Note 1: Program Flash virtual addresses in the non-cacheable range (KSEG1).
2: Program Flash virtual addresses in the cacheable and prefetchable range (KSEG0).
3: The RAM size varies between PIC32MX device family members.

6.5 Program Flash Memory Wait States

For optimal performance, PFMWS(CHECON<2:0>) must be programmed to the minimum value possible. There are two parameters that determine this value:

Flash Access Time - 50 nSec for the PIC32MX processor family.

CPU Core frequency - The Core frequency is programmable. Care must be taken when changing frequencies to make sure that there are enough wait states to prevent any Flash memory access timing violations.

To find out the number of flash wait states required, divide the core clock frequency (in MHz) by 20 and take the integer part of the result. The value that is written to PFMWS (CHECON<2:0>) is one less. For example, core clock frequency is 72MHZ. The number of wait states will be 72 / 20 = 3.6, i.e., 3 wait states. Therefore the actual value written to PFMWS bits will be 2.

6.6 Program Flash Memory Partitioning

The Program Flash Memory can be partitioned for User and Kernel mode programs as shown in Figure 6-3.

At Reset, the User mode partition does not exist (BMXPUPBA is initialized to '0'). The entire Program Flash Memory is mapped to Kernel mode program space starting at virtual address KSEG1: 0xBD000000 (or KSEG0: 0x9D000000). To set up a partition for the User mode program, initialize BMXPUPBA as follows:

BMXPUPBA = BMXPFMSZ – USER_FLASH_PGM_SZ

The USER_FLASH_PGM_SZ is the partition size of the User mode program. BMXPFMSZ is the bus matrix register that holds the total size of Program Flash Memory.

Example:

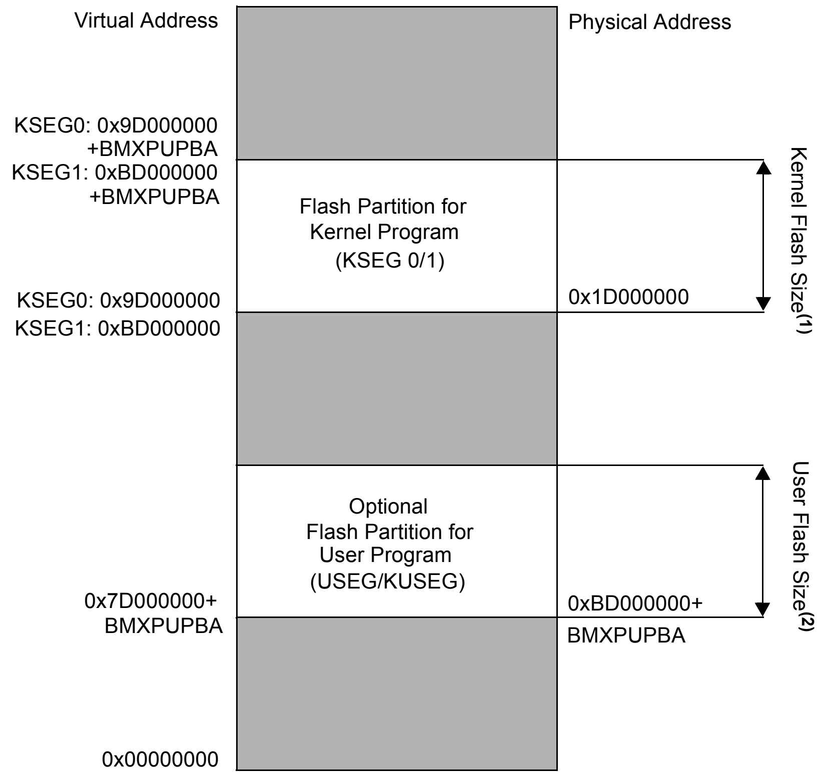
> Assuming the PIC32MX device has 512 Kbytes of Flash memory, the BMXPFMSZ will contain 0x00080000.
>
> To create a user Flash program partition of 20 Kbytes (0x5000):
>
> BMXPUPBA = 0x80000 – 0x5000 = 0x7B000

The size of the user Flash will be 20K and the size left for the kernel Flash will be 512k – 20k = 492K.

The user Flash partition will extend from 0x7D07B000 to 0x7d07FFFF (virtual addresses).

The Kernel mode partition always starts from KSEG1: 0xBD000000 or KSEG0: 0x9D000000. In the above example, the kernel partition will extend from 0xBD000000 to 0xBD07AFFF (492 Kbytes in size).

FIGURE 6-3: FLASH PARTITIONING



Note 1: Kernel Flash Size = BMXPUPBA

2: User Flash Size = BMXPFMSZ-BMXPUPBA

3: If BMXPUPBA is '0', then:
K Flash Size = BMXPFMSZ (i.e., all the Flash)
Usr Flash Size = 0

6.6.1 RAM PARTITIONING

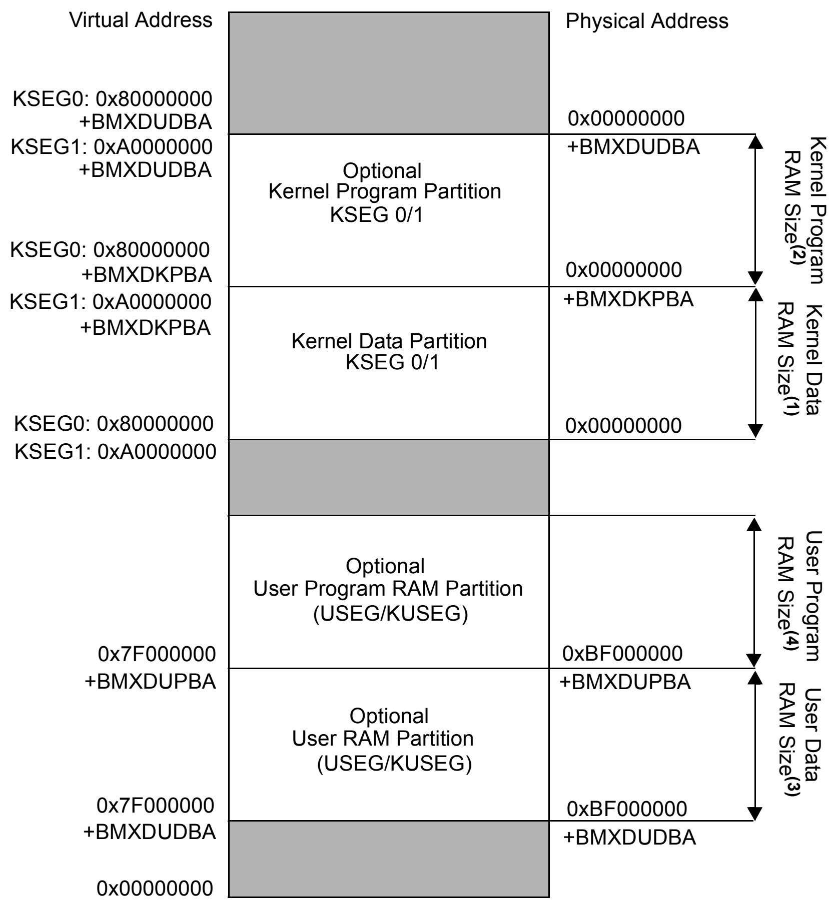
The RAM memory can be divided into 4 partitions. These are:

1. Kernel Data
2. Kernel Program
3. User Data
4. User Program

In order to execute from data RAM, a kernel or user program partition must be defined. At Power-on Reset, the entire data RAM is assigned to the kernel data partition. This partition always starts from the base of the data RAM. See Figure 6-4 for details.

The registers controlling the RAM partitions are BMXDKPBA, BMXDUDBA, and BMXDUPBA. For a detailed discussion on how to use these registers for partitioning the RAM, please refer to the Memory Organization chapter of PIC32MX Family Reference Manual (DS61132).

FIGURE 6-4: RAM PARTITIONING



Note 1: Kernel Data RAM Size = BMXDKPBA

2: Kernel Program RAM Size = BMXDUDBA – BMXDKPBA

3: User Data RAM Size = BMXDUPBA – BMXDUDBA

4: User Program RAM Size = DRM Size – BMXDUPBA

5: If BMXDKPBA, BMXDUDBA or BMXDUPBA is ‘0’, then:
Kernel Data RAM Size = BMXDRMSZ (i.e., all RAM)
Kernel Program RAM Size = 0
User Data RAM Size = 0
User Program RAM Size = 0

6.6.2 ADDRESS DECODE

Table 6-3 shows the address map for system resources available to the CPU when it is operating in either User mode or Kernel mode.

Table 6-4 shows the address map for system resources mapped in KSEG0 that are available to the CPU when it is operating in Kernel mode.

Table 6-5 shows the address map for system resources mapped in KSEG1 that are available to the CPU when it is operating in Kernel mode.

TABLE 6-3: USEG/KUSEG ADDRESS MAP

Virtual Address	Physical Address	PIC32MX3XXF 032x	PIC32MX3XXF 064x	PIC32MX3XXF 128x	PIC32MX3XXF 256x	PIC32MX3XXF 512x
0x0000_0000 0x7D00_0000 + BMXPUPBA - 1	0x4000_0000 0xBD00_0000 + BMXPUPBA - 1	RSVD	RSVD	RSVD	RSVD	RSVD
0x7D00_0000 + BMXPUPBA 0x7D00_7FFF	0xBD00_0000 + BMXPUPBA 0xBD00_7FFF	PFM User Program	PFM User Program	PFM User Program	PFM User Program	PFM User Program
0x7D00_FFFF	0xBD00_FFFF	RSVD				
0x7D01_FFFF	0xBD01_FFFF		RSVD			
0x7D03_FFFF	0xBD03_FFFF			RSVD		
0x7D07_FFFF	0xBD07_FFFF				RSVD	
0x7D08_0000 0x7D08_0000 + BMXDUPBA - 1	0xBD08_0000 0xBD08_0000 + BMXDUPBA - 1					RSVD
0x7F00_0000 + BMXDUDBA 0x7F00_0000 + BMXDUPBA - 1	0xBF00_0000 + BMXDUDBA 0xBF00_0000 + BMXDUPBA - 1	DRM User Data	DRM User Data	DRM User Data	DRM User Data	DRM User Data
0x7F00_0000 + BMXDUPBA 0x7F00_1FFF	0xBF00_0000 + BMXDUPBA 0xBF00_1FFF	DRM User Program DRM=8KB	DRM User Program DRM=8KB	DRM User Program	DRM User Program	DRM User Program
0x7F00_3FFF	0xBF00_3FFF	RSVD	DRM=16KB	DRM=16KB	DRM=16KB	
0x7F00_7FFF	0xBF00_7FFF		RSVD	DRM=32KB	DRM=32KB	DRM=32KB
0x7F0_8000 0x7FFF_FFFF	0xBF0_8000 0xBFFF_FFFF			RSVD	RSVD	RSVD

 Advance Information

TABLE 6-4: KSEG0 ADDRESS MAP

Virtual Address	Physical Address	PIC32MX3XXF 032x	PIC32MX3XXF 064x	PIC32MX3XXF 128x	PIC32MX3XXF 256x	PIC32MX3XXF 512x
0x8000_0000 0x8000_0000 + BMXDKPBA - 1	0x0000_0000 0x0000_0000 + BMXDKPBA - 1	DRM Kernel Data	DRM Kernel Data	DRM Kernel Data	DRM Kernel Data	DRM Kernel Data
0x8000_0000 + BMXDKBPA 0x8000_0000 + BMXDUDBA - 1	0x0000_0000 + BMXDKBPA 0x0000_0000 + BMXDUDBA - 1	DRM Kernel Program	DRM Kernel Program	DRM Kernel Program	DRM Kernel Program	DRM Kernel Program
0x8000_1FFF	0x0000_1FFF	Note 1 DRM=8KB	Note 1 DRM=8KB	Note 1	Note 1	Note 1
0x8000_3FFF	0x0000_3FFF	RSVD	DRM=16KB	DRM=16KB	DRM=16KB	
0x8000_7FFF	0x0000_7FFF		RSVD	DRM=32KB	DRM=32KB	DRM=32KB
0x9CFF_FFFF	0x1CFF_FFFF			RSVD	RSVD	RSVD
0x9D00_0000 0x9D00_0000 + BMXPUPBA - 1	0x1D00_0000 0x1D00_0000 + BMXPUPBA - 1	PFM Kernel Program	PFM Kernel Program	PFM Kernel Program	PFM Kernel Program	PFM Kernel Program
0x9D00_7FFF	0x1D00_7FFF	Note 2	Note 2	Note 2	Note 2	Note 2
0x9D00_FFFF	0x1D00_FFFF	RSVD				
0x9D01_FFFF	0x1D01_FFFF		RSVD			
0x9D03_FFFF	0x1D03_FFFF			RSVD		
0x9D07_FFFF	0x1D07_FFFF				RSVD	
0x9D08_0000 0x9FBF_FFFF	0x1D08_0000 0x1FBF_FFFF					RSVD
0x9FC0_0000 0x9FC0_2FFF	0x1FC0_0000 0x1FC0_2FFF	Boot Flash	Boot Flash	Boot Flash	Boot Flash	Boot Flash
0x9FC0_3000 0x9FFF_EFFF	0x1FC0_3000 0x1FFF_EFFF	RSVD	RSVD	RSVD	RSVD	RSVD
0x9FFF_F000 0x9FFF_FFFF	0x1FFF_F000 0x1FFF_FFFF	Test Flash	Test Flash	Test Flash	Test Flash	Test Flash

Note 1: Not available in KSEG0 if mapped to USEG/KUSEG (i.e. BMXDUDBA or BMXDUPBA non-zero).
2: Not available in KSEG0 if mapped in USEG/KUSEG (i.e. BMXPUPBA non-zero).

TABLE 6-5: KSEG1 ADDRESS MAP

Virtual Address	Physical Address	PIC32MX3XXF 032x	PIC32MX3XXF 064x	PIC32MX3XXF 128x	PIC32MX3XXF 256x	PIC32MX3XXF 512x
0xA000_0000 0xA000_0000 + BMXDKPBA - 1	0x0000_0000 0x0000_0000 + BMXDKPBA - 1	DRM Kernel Data	DRM Kernel Data	DRM Kernel Data	DRM Kernel Data	DRM Kernel Data
0xA000_0000 + BMXDKPBA 0xA000_0000 + BMXDUDBA - 1	0x0000_0000 + BMXDKPBA 0x0000_0000 + BMXDUDBA - 1	DRM Kernel Program	DRM Kernel Program	DRM Kernel Program	DRM Kernel Program	DRM Kernel Program
0xA000_1FFF	0x0000_1FFF	Note 1 DRM=8KB	Note 1 DRM=8KB	Note 1	Note 1	Note 1
0xA000_3FFF	0x0000_3FFF	RSVD	DRM=16KB	DRM=16KB	DRM=16KB	
0xA000_7FFF	0x0000_7FFF		RSVD	DRM=32KB	DRM=32KB	DRM=32KB
0xA000_8000 0xBCFF_FFFF	0x0000_8000 0x1CFF_FFFF			RSVD	RSVD	RSVD
0xBD00_0000 0xBD00_0000 + BMXPUPBA - 1	0x1D00_0000 0x1D00_0000 + BMXPUPBA - 1	PFM Kernel Program	PFM Kernel Program	PFM Kernel Program	PFM Kernel Program	PFM Kernel Program
0xBD00_0000 + BMXPUPBA 0xBD00_7FFF	0x1D00_0000 + BMXPUPBA 0x1D00_7FFF	Note 2	Note 2	Note 2	Note 2	Note 2
0xBD00_FFFF	0x1D00_FFFF	RSVD				
0xBD01_FFFF	0x1D01_FFFF		RSVD			
0xBD03_FFFF	0x1D03_FFFF			RSVD		
0xBD07_FFFF	0x1D07_FFFF				RSVD	
0xBD08_0000 0xBF7F_FFFF	0x1D08_0000 0x1F7F_FFFF					RSVD
0xBF80_0000 0xBF8F_FFFF	0x1F80_0000 0x1F8F_FFFF	Peripherals	Peripherals	Peripherals	Peripherals	Peripherals
0xBF90_0000 0xBFB_FFFF	0x1F90_0000 0x1FB_FFFF	RSVD	RSVD	RSVD	RSVD	RSVD
0xBFC0_0000 0xBFC0_2FFF	0x1FC0_0000 0x1FC0_2FFF	Boot Flash	Boot Flash	Boot Flash	Boot Flash	Boot Flash
0xBFC0_3000 0xBFFF_EFFF	0x1FC0_3000 0x1FFF_EFFF	RSVD	RSVD	RSVD	RSVD	RSVD
0xBFFF_F000 0xBFFF_FFFF	0x1FFF_F000 0x1FFF_FFFF	Test Flash	Test Flash	Test Flash	Test Flash	Test Flash

Note 1: Not available in KSEG1 if mapped to USEG/KUSEG (i.e. BMXDUDBA or BMXDUPBA non-zero).
2: Not available in KSEG1 if mapped in USEG/KUSEG (i.e. BMXPUPBA non-zero).

6.6.3 PERIPHERAL REGISTERS LOCATIONS

Table 6-6 contains the peripheral address map for the PIC32MX device. Peripherals located on the PB Bus are mapped to 512 byte boundaries. Peripherals on the FPB Bus are mapped to 4 Kbyte boundaries.

TABLE 6-6: PERIPHERAL ADDRESS TABLE

Peripheral	Virtual Address		Physical Address	
	Start	End	Start	End
WDT	BF80_0000	BF80_01FF	1F80_0000	1F80_01FF
RTCC	BF80_0200	BF80_03FF	1F80_0200	1F80_03FF
TMR1	BF80_0600	BF80_07FF	1F80_0600	1F80_07FF
TMR2	BF80_0800	BF80_09FF	1F80_0800	1F80_09FF
TMR3	BF80_0A00	BF80_0BFF	1F80_0A00	1F80_0BFF
TMR4	BF80_0C00	BF80_0DFF	1F80_0C00	1F80_0DFF
TMR5	BF80_0E00	BF80_0FFF	1F80_0E00	1F80_0FFF
Input Capture1	BF80_2000	BF80_21FF	1F80_2000	1F80_21FF
Input Capture2	BF80_2200	BF80_23FF	1F80_2200	1F80_23FF
Input Capture3	BF80_2400	BF80_25FF	1F80_2400	1F80_25FF
Input Capture4	BF80_2600	BF80_27FF	1F80_2600	1F80_27FF
Input Capture5	BF80_2800	BF80_29FF	1F80_2800	1F80_29FF
Output Compare1	BF80_3000	BF80_31FF	1F80_3000	1F80_31FF
Output Compare2	BF80_3200	BF80_33FF	1F80_3200	1F80_33FF
Output Compare3	BF80_3400	BF80_35FF	1F80_3400	1F80_35FF
Output Compare4	BF80_3600	BF80_37FF	1F80_3600	1F80_37FF
Output Compare5	BF80_3800	BF80_39FF	1F80_3800	1F80_39FF
I2C1	BF80_5000	BF80_51FF	1F80_5000	1F80_51FF
I2C2	BF80_5200	BF80_53FF	1F80_5200	1F80_53FF
SPI1	BF80_5800	BF80_59FF	1F80_5800	1F80_59FF
SPI2	BF80_5A00	BF80_5BFF	1F80_5A00	1F80_5BFF
UART1	BF80_6000	BF80_61FF	1F80_6000	1F80_61FF
UART2	BF80_6200	BF80_63FF	1F80_6200	1F80_63FF
Parallel Master Port	BF80_7000	BF80_71FF	1F80_7000	1F80_71FF
GPIO	BF80_8000	BF80_81FF	1F80_8000	1F80_81FF
ADC	BF80_9000	BF80_91FF	1F80_9000	1F80_91FF
Comparator Voltage REF	BF80_9800	BF80_99FF	1F80_9800	1F80_99FF
Comparator	BF80_A000	BF80_A1FF	1F80_A000	1F80_A1FF
Oscillator	BF80_F000	BF80_F1FF	1F80_F000	1F80_F1FF
Configuration	BF80_F200	BF80_F3FF	1F80_F200	1F80_F3FF
Flash (NVM)	BF80_F400	BF80_F5FF	1F80_F400	1F80_F5FF
Reset	BF80_F600	BF80_F7FF	1F80_F600	1F80_F7FF
Interrupts	BF88_1000	BF88_1FFF	1F88_1000	1F88_1FFF
Bus Matrix	BF88_2000	BF88_2FFF	1F88_2000	1F88_2FFF
DMA	BF88_3000	BF88_3FFF	1F88_3000	1F88_3FFF
Prefetch Cache	BF88_4000	BF88_4FFF	1F88_4000	1F88_4FFF
GPIO	BF88_6000	BF88_61FF	1F88_6000	1F88_61FF

NOTES:

7.0 FLASH PROGRAM MEMORY

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

The PIC32MX Family of devices contain internal program Flash memory for executing user code. There are three methods by which the user can program this memory:

1. Run-Time Self Programming (RTSP)
2. In-Circuit Serial Programming™ (ICSP™)
3. EJTAG Programming

RTSP is performed by software executing from either Flash or RAM memory. EJTAG is performed using the EJTAG port of the device and a EJTAG capable programmer. ICSP is performed using a serial data connection to the device and allows much faster programming times than RTSP. RTSP techniques are described in this chapter. The ICSP and EJTAG methods are described in the "*PIC32MX Programming Specification*" (DS61145) document, which may be downloaded from the Microchip web site.

7.1 FLASH Controller Registers

TABLE 7-1: FLASH CONTROLLER SFR SUMMARY

Virtual Address	Name		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
BF80_F400	NVMCON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	NVMWR	NVMWREN	NVMERR	LVDERR	LVDSTAT	—	—	—
		7:0	—	—	—	—	NVMOP<3:0>			
BF80_F404	NVMCON-CLR	31:0	Write clears selected bits in NVMCON, read yields undefined value							
BF80_F408	NVMCON-SET	31:0	Write sets selected bits in NVMCON, read yields undefined value							
BF80_F40C	NVMCON-INV	31:0	Write inverts selected bits in NVMCON, read yields undefined value							
BF80_F410	NVMKEY	31:24	NVMKEY<31:24>							
		23:16	NVMKEY<23:16>							
		15:8	NVMKEY<15:8>							
		7:0	NVMKEY<7:0>							
BF80_F420	NVMADDR	31:24	NVMADDR<31:24>							
		23:16	NVMADDR<23:16>							
		15:8	NVMADDR<15:8>							
		7:0	NVMADDR<7:0>							
BF80_F424	NVMADDR-CLR	31:0	Write clears selected bits in NVMADDR, read yields undefined value							
BF80_F428	NVMADDR-SET	31:0	Write sets selected bits in NVMADDR, read yields undefined value							
BF80_F42C	NVMADDR INV	31:0	Write inverts selected bits in NVMADDR, read yields undefined value							
BF80_F430	NVMDATA	31:24	NVMDATA<31:24>							
		23:16	NVMDATA<23:16>							
		15:8	NVMDATA<15:8>							
		7:0	NVMDATA<7:0>							
BF80_F440	NVMSR-CADDR	31:24	NVMSRCADDR<31:24>							
		23:16	NVMSRCADDR<23:16>							
		15:8	NVMSRCADDR<15:8>							
		7:0	NVMSRCADDR<7:0>							

TABLE 7-2: FLASH CONTROLLER INTERRUPT REGISTER SUMMARY

Virtual Address	Name		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
BF88_1050	IEC1	31:24	—	—	—	—	—	—	—	FCEIE
BF88_1020	IFS1	31:24	—	—	—	—	—	—	—	FCEIF
BF88_1120	IPC11	7:0	—	—	—	FCEIP<2:0>			FCEIS<1:0>	

Note: This summary table contains partial register definitions that only pertain to the FLASH memory controller peripheral. Refer to the "*PIC32MX Family Reference Manual*" (DS61132) for a detailed description of these registers.

REGISTER 7-1: NVMCON: PROGRAMMING CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

S/HC-0	R/W-0	R/HS-0	R/HS-0	R/HSHC-0	U-0	U-0	U-0
NVMWR	NVMWREN	NVMERR	LVDERR	LVDSTAT	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	NVMOP3	NVMOP2	NVMOP1	NVMOP0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **NVMWR:** Write Control bit

This bit is writeable when NVMWREN = 1 and the unlock sequence is followed.
1 = Initiate a Flash operation (Hardware clears this bit when the operation completes.)
0 = Flash operation complete or inactive

bit 14 **NVMWREN:** Write Enable bit

1 = Enables writes to NVMWR
0 = Disables writes to NVMWR
Note: This is the only bit in this register that is reset by a device Reset.

bit 13 **NVMERR:** Write Error bit

1 = Program or erase sequence did not complete successfully
0 = Program or erase sequence completed normally
Note: Cleared by setting NVMOP==0000b, and initiating a Flash operation (i.e., NVMWR).

bit 12 **LVDERR:** Low-Voltage Detect Error bit (LVD circuit must be enabled)

This error is only captured for programming/erase operations
1 = Low-voltage detected
0 = Voltage level ok for programming
Note: Cleared by setting NVMOP==0000b, and initiating a Flash operation (i.e., NVMWR).

bit 11 **LVDSTAT:** Low-Voltage Detect Status bit (LVD circuit must be enabled)

This bit is read-only and is automatically set by hardware
1 = Low-voltage event active
0 = Low-voltage event NOT active
Note: Cleared by setting NVMOP==0000b, and initiating a Flash operation (i.e., NVMWR).

bit 10-4 **Unimplemented:** Read as ‘0’

REGISTER 7-1: NVMCON: PROGRAMMING CONTROL REGISTER (CONTINUED)

bit 15-0 **NVMOP<3:0>**: NVM Operation bits

`0111` = Reserved
`0110` = No operation
`0101` = Program Flash (PFM) erase operation: erases PFM, if all pages are not write-protected
`0100` = Page erase operation: erases page selected by NVMADDR, if it is not write-protected
`0011` = Row program operation: programs row selected by NVMADDR, if it is not write-protected
`0010` = No operation
`0001` = Word program operation: programs word selected by NVMADD,R if it is not write-protected
`0000` = No operation

7.2 RTSP Operation

RTSP allows the user code to modify Flash program memory contents. The device Flash memory is divided into two logical Flash partitions: the Program Flash Memory (PFM), and the Boot Flash Memory (BFM). The last page in Boot Flash Memory contains the DEBUG Page, which is reserved for use by the debugger tool while debugging.

The program Flash array for the PIC32MX device is built up of a series of rows. A row contains 128 32-bit instruction words or 512 bytes. A group of 8 rows compose a page; which, therefore, contains 8 × 512 = 4096 bytes or 1024 instruction words. A page of Flash is the smallest unit of memory that can be erased at a single time. The program Flash array can be programmed in one of two ways:

- Row programming, with 128 instruction words at a time.
- Word programming, with 1 instruction word at a time.

The CPU stalls (waits) until the programming operation is finished. The CPU will not execute any instruction, or respond to interrupts, during this time. If any interrupts occur during the programming cycle, they remain pending until the cycle completes.

7.3 Control Registers

There are two SFRs used to erase and write the PFM: NVMCON and NVMKEY.

The NVMCON register (Register 7-1) controls which blocks are to be erased, which memory block is to be programmed and the start of the programming cycle.

NVMKEY is a write-only register that is used for write-protection. To start a programming or erase sequence, the user must consecutively write 0xAA996655 and 0x556699AA to the NVMKEY register. Interrupts should be disabled. Refer to **Section 7.4 "Programming Operations"** for further details.

7.4 Programming Operations

A complete programming sequence is necessary for programming or erasing the internal Flash in RTSP mode. A programming operation is nominally 5 ms in duration and the processor stalls (`WAITS`) until the operation is finished. Setting the NVMWR bit (NVMCON<15>) starts the operation, and the NVMWR bit is automatically cleared when the operation is finished.

7.4.1 PROGRAMMING ALGORITHM

The user can program one row of program Flash memory at a time. To do this, it is necessary to erase the 8-row erase block containing the desired row. The general process is:

1. Read eight rows of program memory (1024 instructions) and store in data RAM.
2. Update the program data in RAM with the desired new data.
3. Erase the page (see Example 7-1):
4. Write the first 128 words from data RAM into the program memory buffers (see Example 7-1).
5. Repeat steps 4 and 5, using the next available 128 words from the block in data RAM by incrementing the value in NVMADDR and NVMASRCADDR, until all 1024 instructions are written back to Flash memory.

For protection against accidental operations, the write initiate sequence for NVMKEY must be used to allow any erase or program operation to proceed. After the programming command has been executed, the user must wait for the programming time until programming is complete.

EXAMPLE 7-1: ERASING FLASH PAGE

```
unsigned int NVMUnlock (unsigned int nvmop)
{
    unsigned int status;

    // Suspend or Disable all Interrupts
    asm volatile (“di %0” : “=r” (status));

    // Enable Flash Write/Erase Operations and Select
    // Flash operation to perform
    NVMCON = 0x8000 \ nvmop;

    // Write Keys
    NVMKEY = 0xAA996655;
    NVMKEY = 0x556699AA;

    // Start the operation using the Set Register
    NVMCONSET = 0x8000;

    // Wait for operation to complete
    while (NVMCON & 0x8000);

    // Restore Interrupts
    if (status & 0x00000001
        asm volatile (“ei”);
    else
        asm volatile (“di”);

    // Return NVMERR and LVDERR Error Status Bits
    return (NVMCON & 0x3000)
}

unsigned int NVMErasePage(void* address)
{
    unsigned int res;

    // Set NVMADDR to the Start Address of page to erase
    NVMADDR = (unsigned int) address;

    // Unlock and Erase Page
    res = NVMUnlock(0x4004);

    // Return Result
    return res;
}
```


EXAMPLE 7-2: ROW PROGRAMMING SEQUENCE

```
unsigned int NVMUnlock (unsigned int nvmop)
{
    unsigned int status;

    // Suspend or Disable all Interrupts
    asm volatile ("di %0" : "=r" (status));

    // Enable Flash Write/Erase Operations and Select
    // Flash operation to perform
    NVMCON = 0x8000 \ nvmop;

    // Write Keys
    NVMKEY = 0xAA996655;
    NVMKEY = 0x556699AA;

    // Start the operation using the Set Register
    NVMCONSET = 0x8000;

    // Wait for operation to complete
    while (NVMCON & 0x8000);

    // Restore Interrupts
    if (status & 0x00000001
        asm volatile ("ei");
    else
        asm volatile ("di");

    // Return NVMERR and LVDERR Error Status Bits
    return (NVMCON & 0x3000)
}

unsigned int NVMWriteRow (void* address, void* data)
{
    unsigned int res;

    // Set NVMADDR to Start Address of row to program
    NVMADDR = (unsigned int) address;

    // Set NVMSRCADDR to the SRAM data buffer Address
    NVMSRCADDR = (unsigned int) data;

    // Unlock and Write Row
    res = NVMUnlock(0x4003);

    // Return Result
    return res;
}
```

EXAMPLE 7-3: WORD PROGRAMMING SEQUENCE

```
unsigned int NVMUnlock (unsigned int nvmop)
{
    unsigned int status;

    // Suspend or Disable all Interrupts
    asm volatile (“di %0” : “=r” (status));

    // Enable Flash Write/Erase Operations and Select
    // Flash operation to perform
    NVMCON = 0x8000 \ nvmop;

    // Write Keys
    NVMKEY = 0xAA996655;
    NVMKEY = 0x556699AA;

    // Start the operation using the Set Register
    NVMCONSET = 0x8000;

    // Wait for operation to complete
    while (NVMCON & 0x8000);

    // Restore Interrupts
    if (status & 0x00000001
        asm volatile (“ei”);
    else
        asm volatile (“di”);

    // Return NVMERR and LVDERR Error Status Bits
    return (NVMCON & 0x3000)
}

unsigned int NVMWriteWord (void* address, unsigned int data)
{
    unsigned int res;

    // Load data into NVMDATA register
    NVMDATA = data;

    // Load address to program into NVMADDR register
    NVMADDR = (unsigned int) address;

    // Unlock and Write Word
    res = NVMUnlock (0x4001);

    // Return Result
    return res;
}
```


EXAMPLE 7-4: PROGRAM FLASH ERASE SEQUENCE

```
unsigned int NVMUnlock (unsigned int nvmop)
{
    unsigned int status;

    // Suspend or Disable all Interrupts
    asm volatile (“di %0” : “=r” (status));

    // Enable Flash Write/Erase Operations and Select
    // Flash operation to perform
    NVMCON = 0x8000 \ nvmop;

    // Write Keys
    NVMKEY = 0xAA996655;
    NVMKEY = 0x556699AA;

    // Start the operation using the Set Register
    NVMCONSET = 0x8000;

    // Wait for operation to complete
    while (NVMCON & 0x8000);

    // Restore Interrupts
    if (status & 0x00000001
        asm volatile (“ei”);
    else
        asm volatile (“di”);

    // Return NVMERR and LVDERR Error Status Bits
    return (NVMCON & 0x3000)
}

unsigned int NVMErasePFM(void)
{
    unsigned int res;

    // Unlock and Erase Program Flash
    res = NVMUnlock(0x4005);

    // Return Result
    return res;
}
```

NOTES:

8.0 RESETS

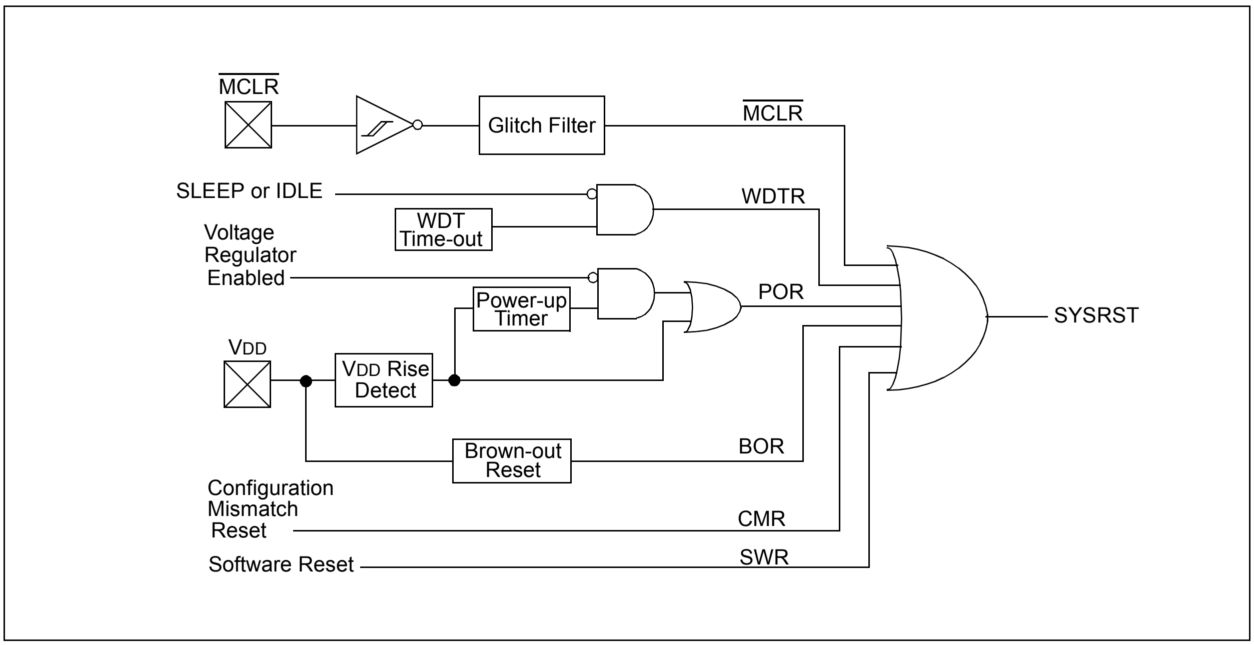
Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

The Reset module combines all Reset sources and controls the device Master Reset signal, SYSRST. The following is a list of device Reset sources:

- POR: Power-on Reset
- $\overline{\text{MCLR}}$: Master Clear Reset Pin
- SWR: Software Reset
- WDTR: Watchdog Timer Reset
- BOR: Brown-out Reset
- CMR: Configuration Mismatch Reset

A simplified block diagram of the Reset module is shown in Figure 8-1.

FIGURE 8-1: SYSTEM RESET BLOCK DIAGRAM



8.1 Reset Registers

TABLE 8-1: RESET SFR SUMMARY

Virtual Address	Name		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
BF80_F600	RCON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	CMR	VREGS
		7:0	EXTR	SWR	—	WDTO	SLEEP	IDLE	BOR	POR
BF80_F604	RCONCLR	31:0	Write clears selected bits in RCON, read yields undefined value							
BF80_F608	RCONSET	31:0	Write sets selected bits in RCON, read yields undefined value							
BF80_F60C	RCONINV	31:0	Write inverts selected bits in RCON, read yields undefined value							
BF80_F610	RSWRST	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	—	—	—	—	—	—	—	SWRST
BF80_F614	RSWRSTCLR	31:0	Write clears selected bits in RSWRST, read yields undefined value							
BF80_F618	RSWRSTSET	31:0	Write sets selected bits in RSWRST, read yields undefined value							
BF80_F61C	RSWRSTINV	31:0	Write inverts selected bits in RSWRST, read yields undefined value							

REGISTER 8-1: RCON: RESET CONTROL REGISTER(3)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	CMR	VREGS
bit 15							bit 8

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1
EXTR	SWR	—	WDTO	SLEEP	IDLE	BOR	POR
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-10 **Unimplemented:** Read as ‘0’

bit 9 **CMR:** Configuration Mismatch Flag bit
1 = A Configuration Mismatch Reset has occurred
0 = A Configuration Mismatch Reset has not occurred
Note: This bit is set in hardware, it can only be cleared (= 0) in software.

bit 8 **VREGS:** Voltage Regulator Standby Enable bit
1 = Regulator will be active during Sleep
0 = Regulator will go to Standby mode during Sleep

bit 7 **EXTR:** External Reset ($\overline{MCLR}$) Pin Flag bit
1 = A Master Clear (pin) Reset has occurred
0 = A Master Clear (pin) Reset has not occurred
Note: This bit is set in hardware, it can only be cleared (= 0) in software.

bit 6 **SWR:** Software Reset Flag bit
1 = A Software Reset was executed
0 = A Software Reset was not executed
Note: This bit is set in hardware, it can only be cleared (= 0) in software.

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **WDTO:** Watchdog Timer Time-out Flag bit
1 = WDT time-out has occurred
0 = WDT time-out has not occurred
Note: This bit is set in hardware, it can only be cleared (= 0) in software.

bit 3 **SLEEP:** Wake From Sleep Flag bit
1 = Device was in Sleep mode
0 = Device was not in Sleep mode
Note: This bit is set in hardware, it can only be cleared (= 0) in software.

bit 2 **IDLE:** Wake-up From Idle Flag bit
1 = Device was in Idle mode
0 = Device was not in Idle mode
Note: This bit is set in hardware, it can only be cleared (= 0) in software.

REGISTER 8-1: RCON: RESET CONTROL REGISTER(3) (CONTINUED)

bit 1	**BOR:** Brown-out Reset Flag bit(1)(2) 1 = A Brown-out Reset has occurred. 0 = A Brown-out Reset has not occurred Note: This bit is set in hardware, it can only be cleared (= 0) in software.
bit 0	**POR:** Power-on Reset Flag bit(1) 1 = A Power-on Reset has occurred 0 = A Power-on Reset has not occurred Note: This bit is set in hardware, it can only be cleared (= 0) in software.

Note 1: User must clear this bit to view next detection.

2: BOR is also set after a Power-on Reset.

3: The RCON flag bits only serve as status bits. Setting a particular Reset status bit in software will not cause a device Reset to occur.

 Advance Information

REGISTER 8-2: RSWRST: SOFTWARE RESET REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	W-0
—	—	—	—	—	—	—	SWRST
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-1 **Unimplemented:** Read as ‘0’

bit 0 **SWRST:** Software Reset Trigger bit
1 = Enable software reset event

Note: The system unlock sequence must be performed before the SWRST bit can be written. A read must follow the write of this bit to generate a Reset.

8.2 Reset Modes

The PIC32MX internal device Reset signal is SYSRST and can be generated from multiple Reset sources, such as POR (Power-on Reset), BOR (Brown-out Reset), MCLR (Master Clear Reset), WDTO (Watchdog Time-out Reset), SWR (Software Reset) and CMR (Configuration Mismatch Reset). A Reset source sets a corresponding status bit in the RCON register to indicate the type of Reset (see Register 8-1). A system Reset is active at first the POR and asserted until device configuration settings are loaded and the clock oscillator sources become stable. The system Reset is then deasserted allowing the CPU to start fetching code after 8 system clock cycles (SYSCLK) .

8.2.1 POWER-ON RESET (POR)

A power-on event generates an internal Power-on Reset pulse when a VDD rise is detected above VPOR. The device supply voltage characteristics must meet the specified starting voltage and rise rate requirements to generate the POR pulse. In particular, VDD must fall below VPOR before a new POR is initiated. For more information on the VPOR and VDD rise rate specifications, refer to **Section 29.0 "Electrical Characteristics"** of this device family data sheet.

8.2.2 MCLR RESET (EXTR)

Whenever the MCLR pin is driven low, the device asynchronously asserts SYSRST provided the input pulse on MCLR is longer than a certain minimum width, as specified in **Section 29.0 "Electrical Characteristics"** of this device family data sheet.

MCLR provides a filter to minimize the effects of noise and to avoid unwanted Reset conditions. The EXTR bit (RCON<7>) is set to indicate the MCLR Reset.

8.2.3 SOFTWARE RESET (SWR)

The PIC32MX CPU core doesn't provide a specific `RESET` "instruction"; however, a hardware Reset can be performed in software (Software Reset) by executing a Software Reset command sequence:

- Write the system unlock sequence
- Set bit, SWRST (RSWRST<0>) = `1`
- Read RSWRST register – Reset occurs
- Follow with "while(1);" or 4 "`NOP`" instructions

Writing a '`1`' to the RSWRST register sets bit SWRST, arming the Software Reset. The subsequent read of the RSWRST register triggers the Software Reset, which should occur on the next clock cycle following the read operation. To ensure no other user code is executed before the Reset event occurs, it is recommended that 4 '`NOP`' instructions or a "while(1);" statement be placed after the `READ` instruction.

The SWR Status bit (RCON<6>) is set to indicate the Software Reset.

EXAMPLE 8-1: SOFTWARE RESET COMMAND SEQUENCE

```
/* The following code illustrates a software Reset */
// assume interrupts are disabled
// assume the DMA controller is suspended
// assume the device is locked
/* perform a system unlock sequence */
// starting critical sequence
SYSKEY = 0xaa996655; //write first unlock key to SYSKEY
SYSKEY = 0x556699aa  //write second unlock key to SYSKEY

/* set SWRST bit to arm reset */
RSWRSTSET = 1;

/* read RSWRST register to trigger reset */
unsigned int dummy;
dummy = RSWRST;
/* prevent any unwanted code execution until reset occurs*/
while(1);
```


8.2.4 WATCHDOG TIMER TIME-OUT RESET (WDTR)

A Watchdog Timer time-out causes the device Reset, $\overline{\text{SYSRST}}$, to be asserted asynchronously. Note that a WDT time-out during SLEEP or IDLE mode will wake-up the processor and branch to the PIC32MX Reset vector, but not reset the processor. The only bits affected are WDTO and the SLEEP or IDLE bits in the RCON register. For more information, refer to **Section 26.0 "Watchdog Timer"**.

Note:	In this document, a distinction is made between a power mode as it is used in a specific module, and a power mode as it is used by the device, e.g., Sleep mode of the comparator and SLEEP mode of the CPU. To indicate which type of power mode is intended, uppercase and lowercase letters (Sleep, Idle, Debug) signify a module power mode, and all uppercase letters (SLEEP, IDLE, DEBUG) signify a device power mode.

8.2.5 BROWN-OUT RESET (BOR)

PIC32MX Family devices have a simple brown-out capability. If the voltage supplied to the regulator is inadequate to maintain a regulated level, the regulator Reset circuitry will generate a Brown-out Reset. This event is captured by the BOR flag bit (RCON<1>). Refer to **Section 29.2 "AC Characteristics and Timing Parameters"** for further details.

8.2.6 CONFIGURATION MISMATCH RESET

To maintain the integrity of the stored configuration values, all device Configuration bits are implemented as a complementary set of register bits. For each bit, as the actual value of the register is written as '1', a complementary value, '0', is stored into its corresponding background register and vice versa. The bit pairs are compared every time, 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 bit (RCON<9>) is set.

8.3 Reset States

8.3.1 SPECIAL FUNCTION REGISTER RESET STATES

Most of the Special Function Registers (SFRs) associated with the PIC32MX CPU and peripherals are reset to a particular value at a device Reset. Refer to the corresponding data sheet section for a peripheral's SFR details. The Reset value for each SFR will depend on the type of Reset.

8.3.2 CONFIGURATION WORD REGISTER RESET STATES

All Reset conditions force the Flash Configuration Word registers to be re-loaded. However, a POR forces Flash Configuration Word registers to be reset prior to being reloaded. For all other Reset conditions, the Flash Configuration Word registers are not reset prior to being re-loaded. This difference accommodates $\overline{\text{MCLR}}$ assertions during Debug mode without affecting the state of the debug operations.

8.3.3 RCON REGISTER STATES

Status bits from the RCON register are set or cleared differently in different Reset situations, as indicated in Table 8-2. The RCON bits only serve as status bits. The user may set or clear any of the bits at any time during code execution. Setting a particular Reset bit in software will not cause a device Reset to occur. The RCON register also has other bits associated with the Watchdog Timer and device power-saving states.

TABLE 8-2: STATUS BITS, INITIALIZATION CONDITION FOR RCON REGISTER

Condition	Program Counter	EXTR	SWR	WDTO	SLEEP	IDLE	CM	BOR	POR
Power-on Reset	0xBFC0_0000	0	0	0	0	0	0	1	1
Brown-out Reset	0xBFC0_0000	0	0	0	0	0	0	1	0
$\overline{\text{MCLR}}$ During Run Mode	0xBFC0_0000	1	u	u	u	u	u	u	u
$\overline{\text{MCLR}}$ During Idle Mode	0xBFC0_0000	1	u	u	u	1(2)	u	u	u
$\overline{\text{MCLR}}$ During Sleep Mode	0xBFC0_0000	1	u	u	1(2)	u	u	u	u
Software Reset Command	0xBFC0_0000	u	1	u	u	u	u	u	u
Configuration Word Mismatch Reset	0xBFC0_0000	u	u	u	u	u	1	u	u
WDT Time-out Reset During Run Mode	0xBFC0_0000	u	u	1	u	u	u	u	u
WDT Time-out Reset During Idle Mode	0xBFC0_0000	u	u	1	u	1(2)	u	u	u
WDT Time-out Reset During Sleep Mode	0xBFC0_0000	u	u	1	1(2)	u	u	u	u
Interrupt Exit from Idle Mode	Vector(1)	u	u	0	u	1(2)	u	u	u
Interrupt Exit from Sleep Mode	Vector(1)	u	u	0	1(2)	u	u	u	u

Legend: u = unchanged

Note 1: Depends on Interrupt source.

2: SLEEP and IDLE bits states defined by previously executed WAIT instruction.

8.4 Using the RCON Status Bits

The user can read the RCON register after any device Reset to determine the cause of the Reset. 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.

 Advance Information

TABLE 8-3: RESET FLAG BIT OPERATION

Flag Bit	Set by:	Cleared by:
POR (RCON<0>)	POR	User Software
BOR (RCON<1>)	POR, BOR	User Software
EXTR (RCON<7>)	$\overline{\text{MCLR}}$ Reset	User Software, POR, BOR
SWR (RCON<6>)	Software Reset Command	User Software, POR, BOR
CMR (RCON<9>	Configuration Mismatch	User Software, POR, BOR
WDTO (RCON<4>)	WDT Time-Out	User Software, POR, BOR
SLEEP (RCON<3>)	`WAIT` Instruction	User Software, POR, BOR
IDLE (RCON<2>)	`WAIT` Instruction	User Software, POR, BOR

Note: All Reset flag bits may be set or cleared by the user software.

8.4.1 DEVICE RESET TO CODE EXECUTION START TIME

The delay between the end of a Reset event and when the device actually begins to execute code is determined by two main factors: the type of Reset and the system clock source coming out of the Reset. The code execution start time for various types of device Resets are summarized in Table 8-4. Individual delays are characterized in **Section 29.2 "AC Characteristics and Timing Parameters"**.

TABLE 8-4: CODE EXECUTION START TIME FOR VARIOUS DEVICE RESETS

Reset Type	Clock Source	Code Execution Delay	System Clock Delay	FSCM Delay	Notes
POR	EC, FRC, FRCDIV, LPRC	TPOR + TRST + TSTARTUP	—	—	**1, 2, 3, 7**
	ECPLL, FRCPLL	TPOR + TRST + TSTARTUP	TLOCK	TFSCM	**1, 2, 3, 5, 6, 7**
	XT, HS, SOSC	TPOR + TRST + TSTARTUP	TOST	TFSCM	**1, 2, 3, 4, 6, 7**
	XTPLL	TPOR + TRST + TSTARTUP	TOST + TLOCK	TFSCM	**1, 2, 3, 4, 5, 6, 7**
BOR	EC, FRC, FRCDIV, LPRC	TRST + TSTARTUP	—	—	**2, 3, 7**
	ECPLL, FRCPLL	TRST + TSTARTUP	TLOCK	TFSCM	**2, 3, 5, 6, 7**
	XT, HS, SOSC	TRST + TSTARTUP	TOST	TFSCM	**2, 3, 4, 6, 7**
	XTPLL	TRST + TSTARTUP	TOST + TLOCK	TFSCM	**2, 3, 4, 5, 6, 7**
$\overline{\text{MCLR}}$	Any Clock	TRST + TSTARTUP	—	—	**3, 7**
WDTO	Any Clock	TRST + TSTARTUP	—	—	**3, 7**
SWR	Any Clock	TRST + TSTARTUP	—	—	**3, 7**
CMR	Any Clock	TRST + TSTARTUP	—	—	**3, 7**

Note 1: TPOR = Power-on Reset delay.
2: TRST = TVREG if on-chip regulator is enabled or TPWRT if on-chip regulator is disabled.
3: TSTARTUP = Load configuration settings, and depending on the oscillator settings, may include TOST, TLOCK and TFSCM.
4: TOST = Oscillator Start-up Timer.
5: TLOCK = PLL lock time.
6: TFSCM = Fail-Safe Clock Monitor delay.
7: Included is a required delay of 8 system clock cycles before the Reset to the CPU core is deasserted.

8.5 Interrupts

There are no interrupts for this module.

8.6 I/O Pin Control

There are not I/O pin controls associated with this module.

NOTES:

9.0 INTERRUPTS

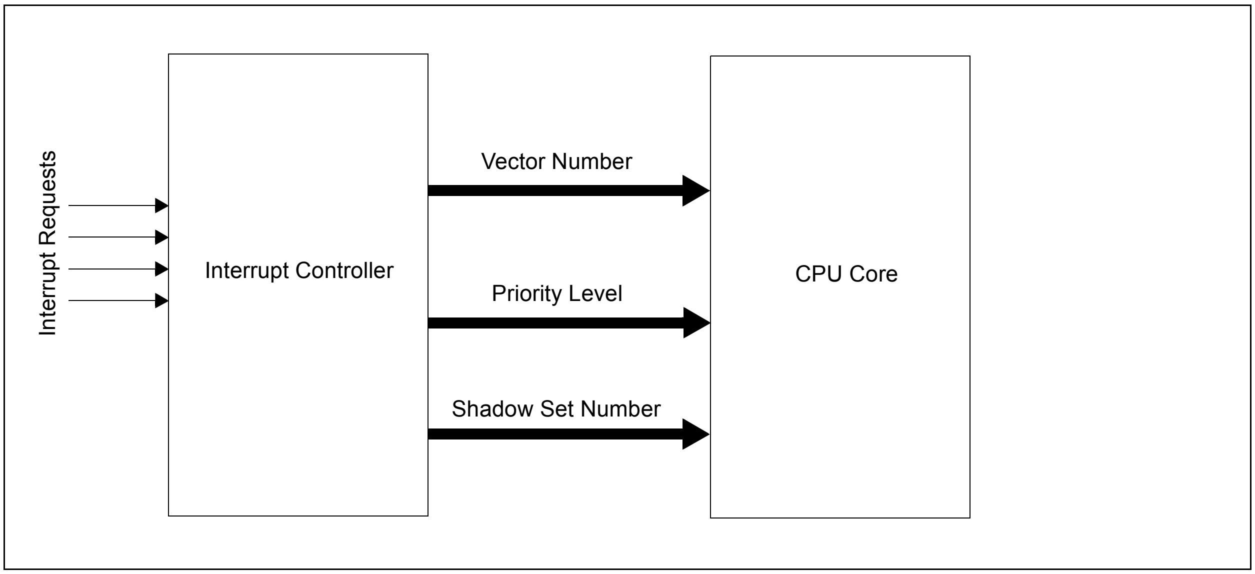
Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

PIC32MX Family generates interrupt requests in response to interrupt events from peripheral modules. The interrupts module exists external to the CPU logic and prioritizes the interrupt events before presenting them to the CPU.

The PIC32MX Family interrupts module includes the following features:

- Up to 96 interrupt sources
- Up to 64 interrupt vectors
- Single and Multi-Vector mode operations
- 5 external interrupts with edge polarity control
- Interrupt proximity timer
- Module Freeze in Debug mode
- 7 user-selectable priority levels for each vector
- 4 user-selectable subpriority levels within each priority
- Dedicated shadow set for highest priority level
- Software can generate any interrupt
- User-configurable interrupt vector table location
- User-configurable interrupt vector spacing

FIGURE 9-1: INTERRUPT CONTROLLER MODULE



Note:	Several of the registers cited in this section are not in the interrupt controller module. These registers (and bits) are associated with the CPU. Details about them are available in **Section 2.0. "CPU".** To avoid confusion, a typographic distinction is made for registers in the CPU. The register names in this section, and all other sections of this manual, are signified by uppercase letters only.CPU register names are signified by upper and lowercase letters. For example, INTSTAT is an Interrupts register; whereas, IntCtl is a CPU register.

9.1 Control Registers

Note:	Each PIC32MX device variant may have one or more Interrupt channels. An 'x' used in the names of control/Status bits and registers denotes the particular channel. Refer to the specific device data sheets for more details.

The interrupts module consists of the following Special Function Registers (SFRs):

- INTCON: Interrupt Control Register INTCONCLR, INTCONSET, INTCONINV: Atomic Bit Manipulation, Write-Only Registers for INTCON
- INTSTAT: Interrupt Status Register INTSTATCLR, INTSTATSET, INTSTATINV: Atomic Bit Manipulation, Write-Only Registers for INTSTAT
- TPTMR: Temporal Proximity Timer Register TPTMRCLR, TPTMRSET, TPTMRNINV: Atomic Bit Manipulation, Write-Only Registers for TPTMR
- IFS0, IFS1: Interrupt Flag Status Registers IFSxCLR, IFSxSET, IFSxINV: Atomic Bit Manipulation, Write-Only Registers for IFSx
- IEC0, IEC1: Interrupt Enable Control Registers IECxCLR, IECxSET, IECxINV: Atomic Bit Manipulation, Write-Only Registers for IECx
- IPC0 - IPC11: Interrupt Priority Control Registers IPCxCLR, IPCxSET, IPCxINV: Atomic Bit Manipulation, Write-Only Registers for IPCx

The following table provides a brief summary of interrupts module related registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

TABLE 9-1: INTERRUPT SFR SUMMARY

Name		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
INTCON	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	SS0
	15:8	—	FRZ	—	MVEC	—	TPC<2:0>		
	7:0	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
INTCONCLR	31:0	Write clears the selected bits in INTCON, read yields undefined value							
INTCONSET	31:0	Write sets the selected bits in INTCON, read yields undefined value							
INTCONINV	31:0	Write inverts the selected bits in INTCON, read yields undefined value							
INTSTAT	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	RIPL<2:0>		
	7:0	—	—	VEC<5:0>					
INTSTATCLR	31:0	Write clears the selected bits in INTSTAT, read yields undefined value							
INTSTATSET	31:0	Write sets the selected bits in INTSTAT, read yields undefined value							
INTSTATINV	31:0	Write inverts the selected bits in INTSTAT, read yields undefined value							
TPTMR	31:24	TPTMR<31:0>							
	23:16								
	15:8								
	7:0								
TPTMRCLR	31:0	Write clears the selected bits in TPTMR, read yields undefined value							
TPTMRSET	31:0	Write clears the selected bits in TPTMR, read yields undefined value							
TPTMRINV	31:0	Write clears the selected bits in TPTMR, read yields undefined value							
IFS0	31:24	I2C1MIF	I2C1SIF	I2C1BIF	U1TXIF	U1RXIF	U1EIF	SPI1RXIF	SPI1TXIF
	23:16	SPI1EIF	OC5IF	IC5IF	T5IF	INT4IF	OC4IF	IC4IF	T4IF
	15:8	INT3IF	OC3IF	IC3IF	T3IF	INT2IF	OC2IF	IC2IF	T2IF
	7:0	INT1IF	OC1IF	IC1IF	T1IF	INT0IF	CS1IF	CS0IF	CTIF
IFS0CLR	31:0	Write clears the selected bits in IFS0, read yields undefined value							
IFS0SET	31:0	Write sets the selected bits in IFS0, read yields undefined value							
IFS0INV	31:0	Write inverts the selected bits in IFS0, read yields undefined value							

TABLE 9-1: INTERRUPT SFR SUMMARY (CONTINUED)

Name		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
IFS1	31:24	—	—	—	—	—	—	—	FCEIF
	23:16	—	—	—	—	DMA3IF	DMA2IF	DMA1IF	DMA0IF
	15:8	RTCCIF	FSCMIF	I2C2MIF	I2C2SIF	I2C2BIF	U2TXIF	U2RXIF	U2EIF
	7:0	SPI2RXIF	SPI2TXIF	SPI2EIF	CMP2IF	CMP1IF	PMPIF	AD1IF	CNIF
IFS1CLR	31:0	Write clears the selected bits in IFS1, read yields undefined value							
IFS1SET	31:0	Write sets the selected bits in IFS1, read yields undefined value							
IFS1INV	31:0	Write inverts the selected bits in IFS1, read yields undefined value							
IEC0	31:24	I2C1MIE	I2C1SIE	I2C1BIE	U1TXIE	U1RXIE	U1EIE	SPI1RXIE	SPI1TXIE
	23:16	SPI1EIE	OC5IE	IC5IE	T5IE	INT4IE	OC4IE	IC4IE	T4IE
	15:8	INT3IE	OC3IE	IC3IE	T3IE	INT2IE	OC2IE	IC2IE	T2IE
	7:0	INT1IE	OC1IE	IC1IE	T1IE	INT0IE	CS1IE	CS0IE	CTIE
IEC0CLR	31:0	Write clears the selected bits in IEC0, read yields undefined value							
IEC0SET	31:0	Write sets the selected bits in IEC0, read yields undefined value							
IEC0INV	31:0	Write inverts the selected bits in IEC0, read yields undefined value							
IEC1	31:24	—	—	—	—	—	—	—	FCEIE
	23:16	—	—	—	—	DMA3IE	DMA2IE	DMA1IE	DMA0IE
	15:8	RTCCIE	FSCMIE	I2C2MIE	I2C2SIE	I2C2BIE	U2TXIE	U2RXIE	U2EIE
	7:0	SPI2RXIE	SPI2TXIE	SPI2EIE	CMP2IE	CMP1IE	PMPIE	AD1IE	CNIE
IEC1CLR	31:0	Write clears the selected bits in IEC1, read yields undefined value							
IEC1SET	31:0	Write sets the selected bits in IEC1, read yields undefined value							
IEC1INV	31:0	Write inverts the selected bits in IEC1, read yields undefined value							
IPC0	31:24	—	—	—	INT0IP<2:0>			INT0IS<1:0>	
	23:16	—	—	—	CS1IP<2:0>			CS1IS<1:0>	
	15:8	—	—	—	CS0IP<2:0>			CS0IS<1:0>	
	7:0	—	—	—	CTIP<2:0>			CTIS<1:0>	
IPC0CLR	31:0	Write clears the selected bits in IPC0, read yields undefined value							
IPC0SET	31:0	Write sets the selected bits in IPC0, read yields undefined value							
IPC0INV	31:0	Write inverts the selected bits in IPC0, read yields undefined value							
IPC1	31:24	—	—	—	INT1IP<2:0>			INT1IS<1:0>	
	23:16	—	—	—	OC1IP<2:0>			OC1IS<1:0>	
	15:8	—	—	—	IC1IP<2:0>			IC1IS<1:0>	
	7:0	—	—	—	T1IP<2:0>			T1IS<1:0>	
IPC1CLR	31:0	Write clears the selected bits in IPC1, read yields undefined value							
IPC1SET	31:0	Write sets the selected bits in IPC1, read yields undefined value							
IPC1INV	31:0	Write inverts the selected bits in IPC1, read yields undefined value							
IPC2	31:24	—	—	—	INT2IP<2:0>			INT2IS<1:0>	
	23:16	—	—	—	OC2IP<2:0>			OC2IS<1:0>	
	15:8	—	—	—	IC2IP<2:0>			IC2IS<1:0>	
	7:0	—	—	—	T2IP<2:0>			T2IS<1:0>	
IPC2CLR	31:0	Write clears the selected bits in IPC2, read yields undefined value							
IPC2SET	31:0	Write sets the selected bits in IPC2, read yields undefined value							
IPC2INV	31:0	Write inverts the selected bits in IPC2, read yields undefined value							

TABLE 9-1: INTERRUPT SFR SUMMARY (CONTINUED)

Name		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
IPC3	31:24	—	—	—	INT3IP<2:0>			INT3IS<1:0>	
	23:16	—	—	—	OC3IP<2:0>			OC3IS<1:0>	
	15:8	—	—	—	IC3IP<2:0>			IC3IS<1:0>	
	7:0	—	—	—	T3IP<2:0>			T3IS<1:0>	
IPC3CLR	31:0	Write clears the selected bits in IPC3, read yields undefined value							
IPC3SET	31:0	Write sets the selected bits in IPC3, read yields undefined value							
IPC3INV	31:0	Write inverts the selected bits in IPC3, read yields undefined value							
IPC4	31:24	—	—	—	INT4IP<2:0>			INT4IS<1:0>	
	23:16	—	—	—	OC4IP<2:0>			OC4IS<1:0>	
	15:8	—	—	—	IC4IP<2:0>			IC4IS<1:0>	
	7:0	—	—	—	T4IP<2:0>			T4IS<1:0>	
IPC4CLR	31:0	Write clears the selected bits in IPC4, read yields undefined value							
IPC4SET	31:0	Write sets the selected bits in IPC4, read yields undefined value							
IPC4INV	31:0	Write inverts the selected bits in IPC4, read yields undefined value							
IPC5	31:24	—	—	—	SPI1IP<2:0>			SPI1IS<1:0>	
	23:16	—	—	—	OC5IP<2:0>			OC5IS<1:0>	
	15:8	—	—	—	IC5IP<2:0>			IC5IS<1:0>	
	7:0	—	—	—	T5IP<2:0>			T5IS<1:0>	
IPC5CLR	31:0	Write clears the selected bits in IPC5, read yields undefined value							
IPC5SET	31:0	Write sets the selected bits in IPC5, read yields undefined value							
IPC5INV	31:0	Write inverts the selected bits in IPC5, read yields undefined value							
IPC6	31:24	—	—	—	AD1IP<2:0>			AD1IS<1:0>	
	23:16	—	—	—	CNIP<2:0>			CNIS<1:0>	
	15:8	—	—	—	I2C1IP<2:0>			I2C1IS<1:0>	
	7:0	—	—	—	U1IP<2:0>			U1IS<1:0>	
IPC6CLR	31:0	Write clears the selected bits in IPC6, read yields undefined value							
IPC6SET	31:0	Write sets the selected bits in IPC6, read yields undefined value							
IPC6INV	31:0	Write inverts the selected bits in IPC6, read yields undefined value							
IPC7	31:24	—	—	—	SPI2IP<2:0>			SPI2IS<1:0>	
	23:16	—	—	—	CMP2IP<2:0>			CMP2IS<1:0>	
	15:8	—	—	—	CMP1IP<2:0>			CMP1IS<1:0>	
	7:0	—	—	—	PMPIP<2:0>			PMPIS<1:0>	
IPC7CLR	31:0	Write clears the selected bits in IPC7, read yields undefined value							
IPC7SET	31:0	Write sets the selected bits in IPC7, read yields undefined value							
IPC7INV	31:0	Write inverts the selected bits in IPC7, read yields undefined value							
IPC8	31:24	—	—	—	RTCCIP<2:0>			RTCCIS<1:0>	
	23:16	—	—	—	FSCMIP<2:0>			FSCMIS<1:0>	
	15:8	—	—	—	I2C2IP<2:0>			I2C2IS<1:0>	
	7:0	—	—	—	U2IP<2:0>			U2IS<1:0>	
IPC8CLR	31:0	Write clears the selected bits in IPC8, read yields undefined value							
IPC8SET	31:0	Write sets the selected bits in IPC8, read yields undefined value							
IPC8INV	31:0	Write inverts the selected bits in IPC8, read yields undefined value							

TABLE 9-1: INTERRUPT SFR SUMMARY (CONTINUED)

Name		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
IPC9	31:24	—	—	—	DMA3IP<2:0>			DMA3IS<1:0>	
	23:16	—	—	—	DMA2IP<2:0>			DMA2IS<1:0>	
	15:8	—	—	—	DMA1IP<2:0>			DMA1IS<1:0>	
	7:0	—	—	—	DMA0IP<2:0>			DMA0IS<1:0>	
IPC9CLR	31:0	Write clears the selected bits in IPC9, read yields undefined value							
IPC9SET	31:0	Write sets the selected bits in IPC9, read yields undefined value							
IPC9INV	31:0	Write inverts the selected bits in IPC9, read yields undefined value							
IPC10	31:24	—	—	—	—			_	
	23:16	—	—	—	—			—	
	15:8	—	—	—	—			_	
	7:0	—	—	—	—			—	
IPC10CLR	31:0	Write clears the selected bits in IPC10, read yields undefined value							
IPC10SET	31:0	Write sets the selected bits in IPC10, read yields undefined value							
IPC10INV	31:0	Write inverts the selected bits in IPC10, read yields undefined value							
IPC11	31:24	—	—	—	—			_	
	23:16	—	—	—	—			—	
	15:8	—	—	—	—			—	
	7:0	—	—	—	FCEIP<2:0>			FCEIS<1:0>	
IPC11CLR	31:0	Write clears the selected bits in IPC11, read yields undefined value							
IPC11SET	31:0	Write sets the selected bits in IPC11, read yields undefined value							
IPC11INV	31:0	Write inverts the selected bits in IPC11, read yields undefined value							

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	SS0
bit 23							bit 16

U-0	R/W-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	FRZ	—	MVEC	—	TPC<2:0>		
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit value at POR (‘0’, ‘1’, x = Unknown)		

bit 31-17 **Unimplemented:** Read as ‘0’

bit 16 **SS0:** Single Vector Shadow Register Set bit
1 = Single vector is presented with a shadow register set
0 = Single vector is not presented with a shadow register set

bit 15 **Unimplemented:** Read as ‘0’

bit 14 **FRZ:** Freeze in Debug Exception Mode bit
1 = Freeze operation when CPU is in Debug Exception mode
0 = Continue operation even when CPU is in Debug Exception mode
Only writable in Debug Exception mode, otherwise, read “0”.

bit 13 **Unimplemented:** Read as ‘0’

bit 12 **MVEC:** Multi-Vector Configuration bit
1 = Interrupt controller configured for Multi-Vectored mode
0 = Interrupt controller configured for Single Vectored mode

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **TPC<2:0>:** Temporal Proximity Control bits
111 = Interrupt of group priority 7 or lower starts the IP timer
110 = Interrupt of group priority 6 or lower starts the IP timer
101 = Interrupt of group priority 5 or lower starts the IP timer
100 = Interrupt of group priority 4 or lower starts the IP timer
011 = Interrupt of group priority 3 or lower starts the IP timer
010 = Interrupt of group priority 2 or lower starts the IP timer
001 = Interrupt of group priority 1 starts the IP timer
000 = Disables proximity timer

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4 **INT4EP:** External Interrupt 4 Edge Polarity Control bit
1 = Rising edge
0 = Falling edge

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER (CONTINUED)

bit 3	**INT3EP:** External Interrupt 3 Edge Polarity Control bit 1 = Rising edge 0 = Falling edge
bit 2	**INT2EP:** External Interrupt 2 Edge Polarity Control bit 1 = Rising edge 0 = Falling edge
bit 1	**INT1EP:** External Interrupt 1 Edge Polarity Control bit 1 = Rising edge 0 = Falling edge
bit 0	**INT0EP:** External Interrupt 0 Edge Polarity Control bit 1 = Rising edge 0 = Falling edge

REGISTER 9-2: INTSTAT: INTERRUPT STATUS REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	R-0	R-0	R-0
—	—	—	—	—	RIPL<2:0>		
bit 15							bit 8

U-0	U-0	R-0	R-0	R-0	R-0	R-0	R-0
—	—	VEC<5:0>					
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit value at POR (‘0’, ‘1’, x = Unknown)		

bit 31-11 **Unimplemented:** Read as ‘0’

bit 10-8 **RIPL<2:0>:** Requested Priority Level bits

000 – 111 = The priority level of the latest interrupt presented to the CPU

Note: This value should only be used when the interrupt controller is configured for Single Vector mode.

bit 5-0 **VEC:** Interrupt Vector bits

00000 – 11111 = The interrupt vector that is presented to the CPU

Note: This value should only be used when the interrupt controller is configured for Single Vector mode.

REGISTER 9-3: TPTMR: TEMPORAL PROXIMITY TIMER REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TPTMR<31:24>							
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TPTMR<23:16>							
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TPTMR<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TPTMR<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit value at POR (‘0’, ‘1’, x = Unknown)		

bit 31-0 **TPTMR:** Temporal Proximity Timer Reload bits
Used by the temporal proximity timer as a reload value when the temporal proximity timer is triggered by an interrupt event.

REGISTER 9-4: IFS0: INTERRUPT FLAG STATUS REGISTER 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
I2C1MIF	I2CSIF	I2CBIF	U1TXIF	U1RXIF	U1EIF	SPI1RXIF	SPI1TXIF
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPI1EIF	OC5IF	IC5IF	T5IF	INT4IF	OC4IF	IC4IF	T4IF
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT3IF	OC3IF	IC3IF	T3IF	INT2IF	OC2IF	IC2IF	T2IF
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT1IF	OC1IF	IC2IF	T1IF	INT0IF	CS1IF	CS0IF	CTIF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit value at POR (‘0’, ‘1’, x = Unknown)		

bit 31 **I2C1MIF:** I2C1 Master Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 30 **I2CSIF:** I2C1 Slave Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 29 **I2CBIF:** I2C1 Bus Collision Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 28 **U1TXIF:** UART1 Transmitter Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 27 **U1RXIF:** UART1 Receiver Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 26 **U1EIF:** UART1 Error Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 25 **SPI1RXIF:** SPI1 Receive Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 24 **SPI1TXIF:** SPI1 Transmitter Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 23 **SPI1EIF:** SPI1 Error Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

REGISTER 9-4: IFS0: INTERRUPT FLAG STATUS REGISTER 0 (CONTINUED)

bit 22	**OC5IF:** Output Compare 5 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 21	**IC5IF:** Input Compare 5 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 20	**T5IF:** Timer5 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 19	**INT4IF:** External Interrupt 4 Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 18	**OC4IF:** Output Compare 4 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 17	**IC4IF:** Input Compare 4 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 16	**T4IF:** Timer4 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 15	**INT3IF:** External Interrupt 3 Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 14	**OC3IF:** Output Compare 3 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 13	**IC3IF:** Input Compare 3 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 12	**T3IF:** Timer3 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 11	**INT2IF:** External Interrupt 2 Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 10	**OC2IF:** Output Compare 2 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 9	**IC2IF:** Input Compare 2 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 8	**T2IF:** Timer2 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 7	**INT1IF:** External Interrupt 1 Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 6	**OC1IF:** Output Compare 1 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred

REGISTER 9-4: IFS0: INTERRUPT FLAG STATUS REGISTER 0 (CONTINUED)

bit 5	**IC1IF:** Input Compare 1 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 4	**T1IF:** Timer1 Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 3	**INT0IF:** External Interrupt 0 Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 2	**CS1IF:** Core Software Interrupt 1 Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 1	**CS0IF:** Core Software Interrupt 0 Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred
bit 0	**CTIF:** Core Timer Interrupt Request Flag bit 1 = Interrupt request has occurred 0 = No interrupt request has a occurred

 Advance Information

REGISTER 9-5: IFS1: INTERRUPT FLAG STATUS REGISTER 1

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	FCEIF
bit 31							bit 24

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	DMA3IF	DMA2IF	DMA1IF	DMA0IF
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RTCCIF	FSCMIF	I2C2MIF	I2C2SIF	I2C2BIF	U2TXIF	U2RXIF	U2EIF
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPI2RXIF	SPI2TXIF	SPI2EIF	CMP2IF	CMP1IF	PMPIF	AD1IF	CNIF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit value at POR (‘0’, ‘1’, x = Unknown)		

bit 31-25 **Unimplemented:** Read as ‘0’

bit 24 **FCEIF:** Flash Control Event Interrupt Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 23-20 **Unimplemented:** Read as ‘0’

bit 19 **DMA3IF:** DMA3 Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 18 **DMA2IF:** DMA2 Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 17 **DMA1IF:** DMA1 Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 16 **DMA0IF:** DMA0 Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 15 **RTCCIF:** Real Time Clock Interrupt Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 14 **FSCMIF:** Fail-Safe Clock Monitor Interrupt Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 13 **I2C2MIF:** I2C2 Master Interrupt Request bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

REGISTER 9-5: IFS1: INTERRUPT FLAG STATUS REGISTER 1 (CONTINUED)

bit 12 **I2C2SIF:** I2C2 Slave Interrupt Request bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 11 **I2C2BIF:** I2C2 Bus Collision Interrupt Request bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 10 **U2TXIF:** UART2 Transmitter Interrupt Request bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 9 **U2RXIF:** UART2 Receiver Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 8 **U2EIF:** UART2 Error Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 7 **SPI2RXIF:** SPI2 Receiver Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 6 **SPI2TXIF:** SPI2 Transmitter Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 5 **SPI2EIF:** SPI2 Error Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 4 **CMP2IF:** Comparator 2 Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 3 **CMP1IF:** Comparator 1 Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 2 **PMPIF:** Parallel Master Port Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 1 **AD1IF:** Analog-to-Digital 1 Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

bit 0 **CNIF:** Input Change Interrupt Request Flag bit
1 = Interrupt request has occurred
0 = No interrupt request has a occurred

REGISTER 9-6: IEC0: INTERRUPT ENABLE CONTROL REGISTER 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
I2C1MIE	I2C1SIE	I2C1BIE	U1TXIE	U1RXIE	U1EIE	SPI1RXIE	SPI1TXIE
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPI1EIE	OC5IE	IC5IE	T5IE	INT4IE	OC4IE	IC4IE	T4IE
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT3IE	OC3IE	IC3IE	T3IE	INT2IE	OC2IE	IC2IE	T2IE
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT1IE	OC1IE	IC1IE	T1IE	INT0IE	CS1IE	CS0IE	CTIE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31 **I2C2MIE:** I2C2 Master Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 30 **I2C1SIE:** I2C1 Slave Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 29 **I2C1BIE:** I2C1 Bus Collision Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 28 **U1TXIE:** UART1 Transmitter Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 27 **U1RXIE:** UART1 Receiver Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 26 **U1EIE:** UART1 Error Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 25 **SPI1RXIE:** SPI1 Receive Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 24 **SPI1TXIE:** SPI1 Transmitter Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 23 **SPI1EIE:** SPI1 Error Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

REGISTER 9-6: IEC0: INTERRUPT ENABLE CONTROL REGISTER 0 (CONTINUED)

bit 22 **OC5IE:** Output Compare 5 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 21 **IC5IE:** Input Compare 5 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 20 **T5IE:** Timer5 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 19 **INT4IE:** External Interrupt 4 Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 18 **OC4IE:** Output Compare 4 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 17 **IC4IE:** Input Compare 4 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 16 **T4IE:** Timer4 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 15 **INT3IE:** External Interrupt 3 Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 14 **OC3IE:** Output Compare 3 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 13 **IC3IE:** Input Compare 3 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 12 **T3IE:** Timer3 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 11 **INT2IE:** External Interrupt 2 Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 10 **OC2IE:** Output Compare 2 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 9 **IC2IE:** Input Compare 2 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 8 **T2IE:** Timer2 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 7 **INT1IE:** External Interrupt 1 Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 6 **OC1IE:** Output Compare 1 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

REGISTER 9-6: IEC0: INTERRUPT ENABLE CONTROL REGISTER 0 (CONTINUED)

bit 5	**IC1IE:** Input Compare 1 Interrupt Enable bit 1 = Interrupt is enabled 0 = Interrupt is disabled
bit 4	**T1IE:** Timer1 Interrupt Enable bit 1 = Interrupt is enabled 0 = Interrupt is disabled
bit 3	**INT0IE:** External Interrupt 0 Enable bit 1 = Interrupt is enabled 0 = Interrupt is disabled
bit 2	**CS1IE:** Core Software Interrupt 1 Enable bit 1 = Interrupt is enabled 0 = Interrupt is disabled
bit 1	**CS0IE:** Core Software Interrupt 0 Enable bit 1 = Interrupt is enabled 0 = Interrupt is disabled
bit 0	**CTIE:** Core Timer Interrupt Enable bit 1 = Interrupt is enabled 0 = Interrupt is disabled

REGISTER 9-7: IEC1: INTERRUPT ENABLE CONTROL REGISTER 1

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	FCEIE
bit 31							bit 24

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	DMA3IE	DMA2IE	DMA1IE	DMA0IE
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RTCCIE	FSCMIE	I2C2MIE	I2C2SIE	I2C2BIE	U2TXIE	U2RXIE	U2EIE
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPI2RXIE	SPI2TXIE	SPI2EIE	CMP2IE	CMP1IE	PMPIE	AD1IE	CNIE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit value at POR (‘0’, ‘1’, x = Unknown)		

bit 31-25 **Unimplemented:** Read as ‘0’

bit 24 **FCEIE:** Flash Control Event Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 23-20 **Unimplemented:** Read as ‘0’

bit 19 **DMA3IE:** DMA3 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 18 **DMA2IE:** DMA2 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 17 **DMA1IE:** DMA1 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 16 **DMA0IE:** DMA0 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 15 **RTCCIE:** Real-Time Clock Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 14 **FSCMIE:** Fail-Safe Clock Monitor Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 13 **I2C2MIE:** I2C2 Master Interrupt Request bit
1 = Interrupt is enabled
0 = Interrupt is disabled

REGISTER 9-7: IEC1: INTERRUPT ENABLE CONTROL REGISTER 1 (CONTINUED)

bit 12 **I2C2SIE:** I2C2 Slave Interrupt Request bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 11 **I2C2BIE:** I2C2 Bus Collision Interrupt Request bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 10 **U2TXIE:** UART2 Transmitter Interrupt Request bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 9 **U2RXIE:** UART2 Receiver Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 8 **U2EIE:** UART2 Error Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 7 **SPI2RXIE:** SPI2 Receiver Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 6 **SPI2TXIE:** SPI2 Transmitter Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 5 **SPI2EIE:** SPI2 Error Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 4 **CMP2IE:** Comparator 2 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 3 **CMP1IE:** Comparator 1 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 2 **PMPIE:** Parallel Master Port Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 1 **AD1IE:** Analog-to-Digital 1 Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 0 **CNIE:** Input Change Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

REGISTER 9-8: IPC0: INTERRUPT PRIORITY CONTROL REGISTER 0

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT0IP<2:0>			INT0IS<1:0>	
bit 31							bit 24

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CS1IP<2:0>			CS1IS<1:0>	
bit 23							bit 16

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CS0IP<2:0>			CS0IS<1:0>	
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CTIP<2:0>			CTIS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-29 **Unimplemented:** Read as ‘0’

bit 28-26 **INT0IP<2:0>:** External Interrupt 0 Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 25-24 **INT0IS<1:0>:** External Interrupt 0 Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 23-21 **Unimplemented:** Read as ‘0’

bit 20-18 **CS1IP<2:0>:** Core Software 1 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 17-16 **CS1IS<1:0>:** Core Software 1 Interrupt subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-8: IPC0: INTERRUPT PRIORITY CONTROL REGISTER 0 (CONTINUED)

bit 15-13	**Unimplemented:** Read as ‘0’
bit 12-10	**CS0IP<2:0>:** Core Software 0 Interrupt Priority bits 111 = Interrupt priority is 7 110 = Interrupt priority is 6 101 = Interrupt priority is 5 100 = Interrupt priority is 4 011 = Interrupt priority is 3 010 = Interrupt priority is 2 001 = Interrupt priority is 1 000 = Interrupt is disabled
bit 9-8	**CS0IS<1:0>:** Core Software 0 Interrupt Subpriority bits 11 = Interrupt subpriority is 3 10 = Interrupt subpriority is 2 01 = Interrupt subpriority is 1 00 = Interrupt subpriority is 0
bit 7-5	**Unimplemented:** Read as ‘0’
bit 4-2	**CTIP<2:0>:** Core Timer Interrupt Priority bits 111 = Interrupt priority is 7 110 = Interrupt priority is 6 101 = Interrupt priority is 5 100 = Interrupt priority is 4 011 = Interrupt priority is 3 010 = Interrupt priority is 2 001 = Interrupt priority is 1 000 = Interrupt is disabled
bit 1-0	**CTIS<1:0>:** Core Timer Interrupt Subpriority bits 11 = Interrupt subpriority is 3 10 = Interrupt subpriority is 2 01 = Interrupt subpriority is 1 00 = Interrupt subpriority is 0

REGISTER 9-9: IPC1: INTERRUPT PRIORITY CONTROL REGISTER 1

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT1IP<2:0>			INT1IS<1:0>	
bit 31							bit 24

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OC1IP<2:0>			OC1IS<1:0>	
bit 23							bit 16

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IC1IP<2:0>			IC1IS<1:0>	
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	T1IP<2:0>			T1IS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-29 **Unimplemented:** Read as ‘0’

bit 28-26 **INT1IP<2:0>:** External Interrupt 1 Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 25-24 **INT1IS<1:0>:** External Interrupt 1 Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 23-21 **Unimplemented:** Read as ‘0’

bit 20-18 **OC1IP<2:0>:** Output Compare 1 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 17-16 **OC1IS<1:0>:** Output Compare 1 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-9: IPC1: INTERRUPT PRIORITY CONTROL REGISTER 1 (CONTINUED)

bit 15-13 **Unimplemented:** Read as '0'

bit 12-10 **IC1IP<2:0>:** Input Compare 1 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 9-8 **IC1IS<1:0>:** Input Compare 1 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4-2 **T1IP<2:0>:** Timer1 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 1-0 **T1IS<1:0>:** Timer1 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-10: IPC2: INTERRUPT PRIORITY CONTROL REGISTER 2

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT2IP<2:0>			INT2IS<1:0>	
bit 31							bit 24

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OC2IP<2:0>			OC2IS<1:0>	
bit 23							bit 16

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IC2IP<2:0>			IC2IS<1:0>	
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	T2IP<2:0>			T2IS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-29 **Unimplemented:** Read as ‘0’

bit 28-26 **INT2IP<2:0>:** External Interrupt 2 Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 25-24 **INT2IS<1:0>:** External Interrupt 2 Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 23-21 **Unimplemented:** Read as ‘0’

bit 20-18 **OC2IP<2:0>:** Output Compare 2 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 17-16 **OC2IS<1:0>:** Output Compare 2 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-10: IPC2: INTERRUPT PRIORITY CONTROL REGISTER 2 (CONTINUED)

bit 15-13 **Unimplemented:** Read as ‘0’

bit 12-10 **IC2IP<2:0>:** Input Compare 2 Interrupt Priority bits

111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 9-8 **IC2IS<1:0>:** Input Compare 2 Interrupt Subpriority bits

11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4-2 **T2IP<2:0>:** Timer2 Interrupt Priority bits

111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 1-0 **T2IS<1:0>:** Timer2 Interrupt Subpriority bits

11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-11: IPC3: INTERRUPT PRIORITY CONTROL REGISTER 3

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT3IP<2:0>			INT3IS<1:0>	
bit 31							bit 24

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OC3IP<2:0>			OC3IS<1:0>	
bit 23							bit 16

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IC3IP<2:0>			IC3IS<1:0>	
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	T3IP<2:0>			T3IS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-29 **Unimplemented:** Read as ‘0’

bit 28-26 **INT3IP<2:0>:** External Interrupt 3 Priority bits
111 = Interrupt Priority is 7
110 = Interrupt Priority is 6
101 = Interrupt Priority is 5
100 = Interrupt Priority is 4
011 = Interrupt Priority is 3
010 = Interrupt Priority is 2
001 = Interrupt Priority is 1
000 = Interrupt is disabled

bit 25-24 **INT3IS<1:0>:** External Interrupt 3 Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 23-21 **Unimplemented:** Read as ‘0’

bit 20-18 **OC3IP<2:0>:** Output Compare 3 Interrupt Priority bits
111 = Interrupt Priority is 7
110 = Interrupt Priority is 6
101 = Interrupt Priority is 5
100 = Interrupt Priority is 4
011 = Interrupt Priority is 3
010 = Interrupt Priority is 2
001 = Interrupt Priority is 1
000 = Interrupt is disabled

bit 17-16 **OC3IS<1:0>:** Output Compare 3 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-11: IPC3: INTERRUPT PRIORITY CONTROL REGISTER 3 (CONTINUED)

bit 15-13 **Unimplemented:** Read as '0'

bit 12-10 **IC3IP<2:0>:** Input Compare 3 Interrupt Priority bits
111 = Interrupt Priority is 7
110 = Interrupt Priority is 6
101 = Interrupt Priority is 5
100 = Interrupt Priority is 4
011 = Interrupt Priority is 3
010 = Interrupt Priority is 2
001 = Interrupt Priority is 1
000 = Interrupt is disabled

bit 9-8 **IC3IS<1:0>:** Input Compare 3 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4-2 **T3IP<2:0>:** Timer3 Interrupt Priority bits
111 = Interrupt Priority is 7
110 = Interrupt Priority is 6
101 = Interrupt Priority is 5
100 = Interrupt Priority is 4
011 = Interrupt Priority is 3
010 = Interrupt Priority is 2
001 = Interrupt Priority is 1
000 = Interrupt is disabled

bit 1-0 **T3IS<1:0>:** Timer3 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-12: IPC4: INTERRUPT PRIORITY CONTROL REGISTER 4

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT4IP<2:0>			INT4IS<1:0>	
bit 31							bit 24

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OC4IP<2:0>			OC4IS<1:0>	
bit 23							bit 16

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IC4IP<2:0>			IC4IS<1:0>	
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	T4IP<2:0>			T4IS<1:0>	
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-29 **Unimplemented:** Read as ‘0’

bit 28-26 **INT4IP<2:0>:** External Interrupt 4 Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 25-24 **INT4IS<1:0>:** External Interrupt 4 Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 23-21 **Unimplemented:** Read as ‘0’

bit 20-18 **OC4IP<2:0>:** Output Compare 4 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 17-16 **OC4IS<1:0>:** Output Compare 4 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-12: IPC4: INTERRUPT PRIORITY CONTROL REGISTER 4 (CONTINUED)

bit 15-13 **Unimplemented:** Read as '0'

bit 12-10 **IC4IP<2:0>:** Input Compare 4 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 9-8 **IC4IS<1:0>:** Input Compare 4 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4-2 **T4IP<2:0>:** Timer4 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 1-0 **T4IS<1:0>:** Timer4 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-13: IPC5: INTERRUPT PRIORITY CONTROL REGISTER 5

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	SPI1IP<2:0>			SPI1IS<1:0>	
bit 31							bit 24

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OC5IP<2:0>			OC5IS<1:0>	
bit 23							bit 16

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IC5IP<2:0>			IC5IS<1:0>	
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	T5IP<2:0>			T5IS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-29 **Unimplemented:** Read as ‘0’

bit 28-26 **SPI1IP<2:0>:** SPI1 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 25-24 **SPI1IS<1:0>:** SPI1 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 23-21 **Unimplemented:** Read as ‘0’

bit 20-18 **OC5IP<2:0>:** Output Compare 5 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 17-16 **OC5IS<1:0>:** Output Compare 5 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-13: IPC5: INTERRUPT PRIORITY CONTROL REGISTER 5 (CONTINUED)

bit 15-13 **Unimplemented:** Read as ‘0’

bit 12-10 **IC5IP<2:0>:** Input Compare 5 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 9-8 **IC5IS<1:0>:** Input Compare 5 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4-2 **T5IP<2:0>:** Timer5 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 1-0 **T5IS<1:0>:** Timer5 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-14: IPC6: INTERRUPT PRIORITY CONTROL REGISTER 6

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	AD1IP<2:0>			AD1IS<1:0>	
bit 31							bit 24

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CNIP<2:0>			CNIS<1:0>	
bit 23							bit 16

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	I2C1IP<2:0>			I2C1IS<1:0>	
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	U1IP<2:0>			U1IS<1:0>	
bit 7							bit 0

bit 31-29 **Unimplemented:** Read as '0'

bit 28-26 **AD1IP<2:0>:** Analog-to-Digital 1 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 25-24 **AD1IS<1:0>:** Analog-to-Digital 1 Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 23-21 **Unimplemented:** Read as '0'

bit 20-18 **CNIP<2:0>:** Input Change Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 17-16 **CNIS<1:0>:** Input Change Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 15-13 **Unimplemented:** Read as '0'

REGISTER 9-14: IPC6: INTERRUPT PRIORITY CONTROL REGISTER 6 (CONTINUED)

bit 12-10 **I2C1IP<2:0>:** I2C1 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 9-8 **I2C1IS<1:0>:** I2C1 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4-2 **U1IP<2:0>:** UART1 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 1-0 **U1IS<1:0>:** UART1 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-15: IPC7: INTERRUPT PRIORITY CONTROL REGISTER 7

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	SPI2IP<2:0>			SPI2IS<1:0>	
bit 31							bit 24

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CMP2IP<2:0>			CMP2IS<1:0>	
bit 23							bit 16

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CMP1IP<2:0>			CMP1IS<1:0>	
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	PMPIP<2:0>			PMPIS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-29 **Unimplemented:** Read as ‘0’

bit 28-26 **SPI2IP<2:0>:** SPI2 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 25-24 **SPI2IS<1:0>:** SPI2 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 23-21 **Unimplemented:** Read as ‘0’

bit 20-18 **CMP2IP<2:0>:** Compare 2 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 17-16 **CMP2IS<1:0>:** Compare 2 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-15: IPC7: INTERRUPT PRIORITY CONTROL REGISTER 7 (CONTINUED)

bit 15-13 **Unimplemented:** Read as ‘0’

bit 12-10 **CMP1IP<2:0>:** Compare 1 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 9-8 **CMP1IS<1:0>:** Compare 1 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4-2 **PMPIP<2:0>:** Parallel Master Port Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 1-0 **PMPIS<1:0>:** Parallel Master Port Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-16: IPC8: INTERRUPT PRIORITY CONTROL REGISTER 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RTCCIP<2:0>			RTCCIS<1:0>	
bit 31							bit 24

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FSCMIP<2:0>			FSCMIS<1:0>	
bit 23							bit 16

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	I2C2IP<2:0>			I2C2IS<1:0>	
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	U2IP<2:0>			U2IS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-29 **Unimplemented:** Read as ‘0’

bit 28-26 **RTCCIP<2:0>:** Real-Time Clock Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 25-24 **RTCCIS<1:0>:** Real-Time Clock Interrupt subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 23-21 **Unimplemented:** Read as ‘0’

bit 12-10 **FSCMIP<2:0>:** Fail-Safe Clock Monitor Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 9-8 **FSCMIS<1:0>:** Fail-Safe Clock Monitor Interrupt subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-16: IPC8: INTERRUPT PRIORITY CONTROL REGISTER 8 (CONTINUED)

bit 15-13 **Unimplemented:** Read as '0'

bit 12-10 **I2C2IP<2:0>:** I2C2 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 9-8 **I2C2IS<1:0>:** I2C2 Interrupt subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4-2 **U2IP<2:0>:** UART2 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 1-0 **U2IS<1:0>:** UART2 subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-17: IPC9: INTERRUPT PRIORITY CONTROL REGISTER 9

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	DMA3IP<2:0>			DMA3IS<1:0>	
bit 31							bit 24

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	DMA2IP<2:0>			DMA2IS<1:0>	
bit 23							bit 16

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	DMA1IP<2:0>			DMA1IS<1:0>	
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	DMA0IP<2:0>			DMA0IS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-29 **Unimplemented:** Read as ‘0’

bit 28-26 **DMA3IP<2:0>:** DMA3 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 25-24 **DMA3IS<1:0>:** DMA3 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 23-21 **Unimplemented:** Read as ‘0’

bit 20-18 **DMA2IP<2:0>:** DMA2 Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 17-16 **DMA2IS<1:0>:** DMA2 Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

REGISTER 9-17: IPC9: INTERRUPT PRIORITY CONTROL REGISTER 9 (CONTINUED)

bit 15-13	**Unimplemented:** Read as ‘0’
bit 12-10	**DMA1IP<2:0>:** DMA1 Interrupt Priority bits 111 = Interrupt priority is 7 110 = Interrupt priority is 6 101 = Interrupt priority is 5 100 = Interrupt priority is 4 011 = Interrupt priority is 3 010 = Interrupt priority is 2 001 = Interrupt priority is 1 000 = Interrupt is disabled
bit 9-8	**DMA1IS<1:0>:** DMA1 Interrupt Subpriority bits 11 = Interrupt subpriority is 3 10 = Interrupt subpriority is 2 01 = Interrupt subpriority is 1 00 = Interrupt subpriority is 0
bit 7-5	**Unimplemented:** Read as ‘0’
bit 4-2	**DMA0IP<2:0>:** DMA0 Interrupt Priority bits 111 = Interrupt priority is 7 110 = Interrupt priority is 6 101 = Interrupt priority is 5 100 = Interrupt priority is 4 011 = Interrupt priority is 3 010 = Interrupt priority is 2 001 = Interrupt priority is 1 000 = Interrupt is disabled
bit 1-0	**DMA0IS<1:0>:** DMA0 Subpriority bits 11 = Interrupt subpriority is 3 10 = Interrupt subpriority is 2 01 = Interrupt subpriority is 1 00 = Interrupt subpriority is 0

REGISTER 9-18: IPC10: INTERRUPT PRIORITY CONTROL REGISTER 10

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **Unimplemented:** Read as ‘0’

REGISTER 9-19: IPC11: INTERRUPT PRIORITY CONTROL REGISTER 11

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FCEIP<2:0>			FCEIS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-5 **Unimplemented:** Read as ‘0’

bit 4-2 **FCEIP<2:0>:** Flash Control Event Interrupt Priority bits
111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 1-0 **FCEIS<1:0>:** Flash Control Event Interrupt Subpriority bits
11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

TABLE 9-2: INTERRUPT IRQ AND VECTOR LOCATION

Interrupt Source	IRQ(1)	Vector Number	Input Type
Highest Natural Order Priority			
CT – Core Timer Interrupt	0	0	Synchronous Edge
CS0 – Core Software Interrupt 0	1	1	Synchronous Edge
CS1 – Core Software Interrupt 1	2	2	Synchronous Edge
INT0 – External Interrupt 0	3	3	Edge
T1 – Timer1	4	4	Edge
IC1 – Input Capture 1	5	5	Synchronous Edge w/Idle
OC1 – Output Compare 1	6	6	Synchronous Edge
INT1 – External Interrupt 1	7	7	Edge
T2 – Timer2	8	8	Synchronous Edge
IC2 – Input Capture 2	9	9	Synchronous Edge w/Idle
OC2 – Output Compare 2	10	10	Synchronous Edge
INT2 – External Interrupt 2	11	11	Edge
T3 – Timer3	12	12	Synchronous Edge
IC3 – Input Capture 3	13	13	Synchronous Edge w/Idle
OC3 – Output Compare 3	14	14	Synchronous Edge
INT3 – External Interrupt 3	15	15	Edge
T4 – Timer4	16	16	Synchronous Edge
IC4 – Input Capture 4	17	17	Synchronous Edge w/Idle
OC4 – Output Compare 4	18	18	Synchronous Edge
INT4 – External Interrupt 4	19	19	Edge
T5 – Timer5	20	20	Synchronous Edge
IC5 – Input Capture 5	21	21	Synchronous Edge w/Idle
OC5 – Output Compare 5	22	22	Synchronous Edge
SPI1E – SPI1 Fault	23	23	Synchronous Edge
SPI1TX – SPI1 Transfer Done	24	23	Synchronous Edge
SPI1RX – SPI1 Receive Done	25	23	Synchronous Edge w/Idle
U1E – UART1 Error	26	24	Synchronous Edge
U1RX – UART1 Receiver	27	24	Synchronous Edge
U1TX – UART1 Transmitter	28	24	Synchronous Edge
I2C1B – I2C1 Bus Collision Event	29	25	Synchronous Edge
I2C1S – I2C1 Slave Event	30	25	Synchronous Edge w/Idle
I2C1M – I2C1 Master Event	31	25	Synchronous Edge
CN – Input Change Interrupt	32	26	Synchronous Level
AD1 – ADC1 convert done	33	27	Edge
PMP – Parallel Master Port	34	28	Synchronous Edge w/Idle
CMP1 – Comparator Interrupt	35	29	Level
CMP2 – Comparator Interrupt	36	30	Level

Note 1: The “IRQ Number” in Table 9-2 is also the “Interrupt Number” listed in the IFSx, IECx and IPSx register definitions.

TABLE 9-2: INTERRUPT IRQ AND VECTOR LOCATION (CONTINUED)

Interrupt Source	IRQ(1)	Vector Number	Input Type
SPI2E – SPI2 Fault	37	31	Synchronous Edge
SPI2TX – SPI2 Transfer Done	38	31	Synchronous Edge
SPI2RX – SPI2 Receive Done	39	31	Synchronous Edge w/ Idle
U2E – UART2 Error	40	32	Synchronous Edge
U2RX – UART2 Receiver	41	32	Synchronous Edge
U2TX – UART2 Transmitter	42	32	Synchronous Edge
I2C2B – I2C2 Bus Collision Event	43	33	Synchronous Edge
I2C2S – I2C2 Slave Event	44	33	Synchronous Edge
I2C2M – I2C2 Master Event	45	33	Synchronous Edge
FSCM – Fail-Safe Clock Monitor	46	34	Edge
RTCC – Real-Time Clock	47	35	Edge
DMA0 – DMA Channel 0	48	36	Synchronous Edge
DMA1 – DMA Channel 1	49	37	Synchronous Edge
DMA2 – DMA Channel 2	50	38	Synchronous Edge
DMA3 – DMA Channel 3	51	39	Synchronous Edge
FCE – Flash Control Event	56	44	Edge
(Reserved)			Edge
Lowest Natural Order Priority			

Note 1: The “IRQ Number” in Table 9-2 is also the “Interrupt Number” listed in the IFSx, IECx and IPSx register definitions.

9.2 Operation

The interrupt controller is responsible for pre-processing Interrupt Requests (IRQ) from a number of on-chip peripherals and presenting them in the appropriate order to the processor.

Figure 9-2 depicts the process within the interrupt controller module. The interrupt controller is designed to receive up to 96 IRQs from the processor core and from on-chip peripherals capable of generating interrupts. All IRQs are sampled on the rising edge of the SYSCLK and latched in associated IFSx registers. A pending IRQ is indicated by the flag bit being equal to '1' in an IFSx register. The pending IRQ will not cause further processing if the corresponding bit in the Interrupt Enable (IECx) register is clear. The IECx bits act to gate the interrupt flag. If the interrupt is enabled, all IRQs are encoded into a 5-bit wide vector number. The 5-bit vector results in 0 to 63 unique interrupt vector numbers. Since there are more IRQs than available vector numbers, some IRQs share common vector numbers. Each vector number is assigned an interrupt priority level and shadow set number. The priority level is determined by the IPCx register setting of the associated vector. In Multi-Vector mode, all priority level 7 interrupts use a dedicated register set, while in Single Vector mode, all interrupts may receive a dedicated shadow set. The interrupt controller selects the highest priority IRQ among all pending IRQs and presents the associated vector number, priority level and shadow set number to the processor core.

The processor core samples the presented vector information between the 'E' and 'M' stage of the pipeline. If the vector's priority level presented to the core is greater than the current priority indicated by the CPU Interrupt Priority bits IPLx (Status<15:10>), the interrupt is serviced; otherwise, it will remain pending until the current priority is less than the interrupt's priority. When servicing an interrupt, the processor core pushes the program counter into the Exception Program Counter (EPC) register in the CPU and sets Exception Level bit EXL (Status<1>) in the CPU. The EXL bit disables further interrupts until the application explicitly re-enables them by clearing the EXL bit. Next, it branches to the vector address calculated from the presented vector number.

The INTSTAT register contains the Interrupt Vector Number bits, VEC (INTSTAT<5:0>), and Requested Interrupt Priority bits, RIPLx (INTSTAT<10:8>), of the current pending interrupt. This may not be the same as the interrupt which caused the core to diverge from normal execution.

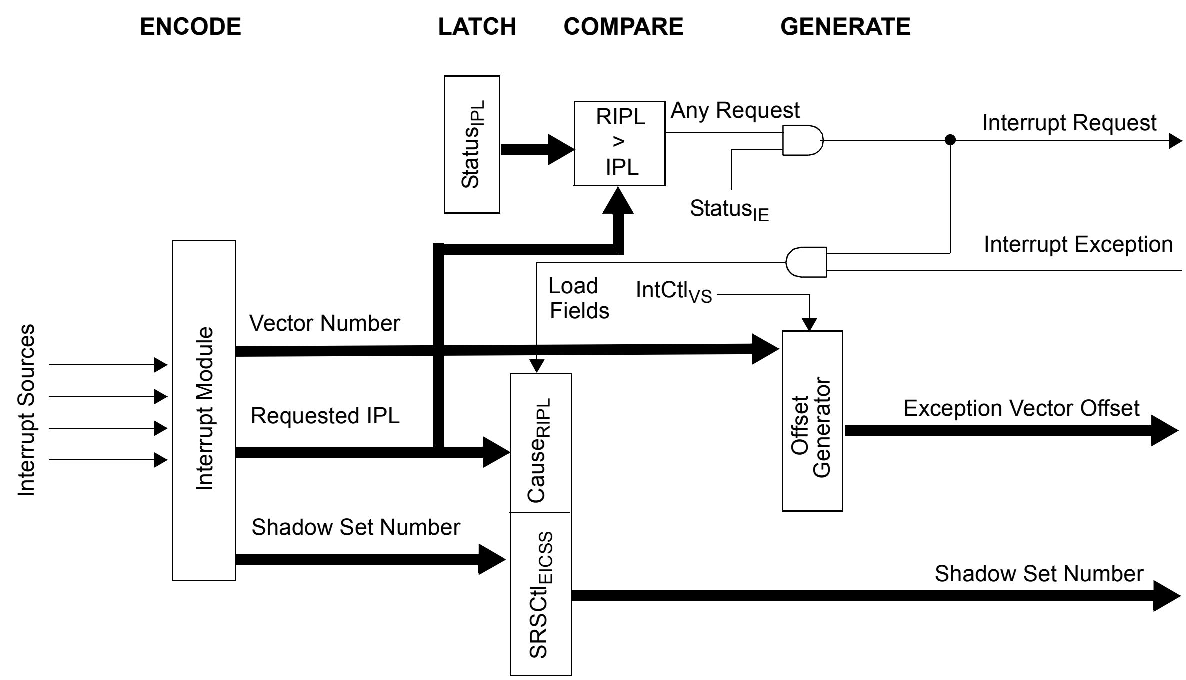
The processor returns to the previous state when the `ERET` (Exception Return) instruction is executed. `ERET` clears the EXL bit, restores the program counter and reverts the current shadow set to the previous one.

The PIC32MX Family interrupt controller can be configured to operate in one of two modes:

- Single Vector mode – all interrupt requests will be serviced at one vector address (mode out of Reset).
- Multi-Vector mode – interrupt requests will be serviced at the calculated vector address.

Notes:	While the user can, during run time, reconfigure the interrupt controller from Single Vector to Multi-Vector mode (or vice versa), such action is strongly discouraged. Changing interrupt controller modes after initialization may result in undefined behavior. The M4K core supports several different interrupt processing modes. The interrupt controller is designed to work in External Interrupt Controller mode.

FIGURE 9-2: INTERRUPT PROCESS



Note: SRSCtl, Cause, Status, and IntCtl registers are CPU registers and are described in **Section 2. "CPU"**.

9.3 Single Vector Mode

On any form of Reset, the interrupt controller initializes to Single Vector mode. When the MVEC (INTCON<12>) bit is '0', the interrupt controller operates in Single Vector mode. In this mode, the CPU always vectors to the same address.

Note:	Users familiar with MIPS32 Architecture must note that the M4K core in PIC32MX Family is still operating in External Interrupt Controller (EIC) mode. The PIC32MX Family achieves Single Vector mode by forcing all IRQs to use a vector number of 0x00. Because the M4K core in PIC32MX Family always operates in EIC mode, the single vector behavior through "Interrupt Compatibility Mode," as defined by MIPS32 Architecture, is not recommended.

To configure the CPU in Single Vector mode, the following CPU registers (IntCtl, Cause, and Status) and INTCON register must be configured as follows:

- EBase ≠ 00000
- VS (IntCtl<9:5>) ≠ 00000
- IV (Cause<23>) = 1
- EXL (Status<1>) = 0
- BEV (Status<22>) = 0
- MVEC (INTCON<12>) = 0
- IE (Status<0>) = 1

EXAMPLE 9-1: SINGLE VECTOR MODE INITIALIZATION

```
/*
  Set the CP0 registers for multi-vector interrupt
  Place EBASE at 0xBD000000

  This code example uses MPLAB C32 intrinsic functions to access CP0 registers.
  Check your compiler documentation to find equivalent functions or use inline assembly
*/
  unsigned int temp;

  asm("di");                         // Disable all interrupts

  temp = _CP0_GET_STATUS();          // Get Status
  temp |= 0x00400000;                // Set BEV bit
  _CP0_SET_STATUS(temp);             // Update Status

  _CP0_SET_EBASE(0xBD000000);        // Set an EBase value of 0xBD000000
  _CP0_SET_INTCTL(0x00000020);       // Set the Vector Spacing to non-zero value

  temp = _CP0_GET_CAUSE();           // Get Cause
  temp |= 0x00800000;                // Set IV
  _CP0_SET_CAUSE(temp);              // Update Cause

  temp = _CP0_GET_STATUS();          // Get Status
  temp &= 0xFFBFFFFD;                // Clear BEV and EXL
  _CP0_SET_STATUS(temp);             // Update Status

  INTCONCLR = 0x1000;                // Clear MVEC bit

  asm("ei");                         // Enable all interrupts
```

 Advance Information

9.4 Multi-Vector Mode

When the MVEC (INTCON<12>) bit is '1', the interrupt controller operates in Multi-Vector mode. In this mode, the CPU vectors to the unique address for each vector number. Each vector is located at a specific offset, with respect to a base address specified by the EBase register in the CPU. The individual vector address offset is determined by the vector space that is specified by the VS bits in the IntCtl register. (The IntCtl register is located in the CPU; refer to Section 2.0 of this manual for more information.)

To configure the CPU in Multi-Vector mode, the following CPU registers (IntCtl, Cause, and Status) and the INTCON register must be configured as follows:

- EBase ≠ 00000
- VS (IntCtl<9:5>) ≠ 00000
- IV (Cause<23>) = 1
- EXL (Status<1>) = 0
- BEV (Status<22>) = 0
- MVEC (INTCON<12>) = 1
- IE (Status<0>) = 1

EXAMPLE 9-2: MULTI-VECTOR MODE INITIALIZATION

```
/*
   Set the CP0 registers for multi-vector interrupt
   Place EBASE at 0xBD000000 and Vector Spacing to 32 bytes

   This code example uses MPLAB C32 intrinsic functions to access CP0 registers.
   Check your compiler documentation to find equivalent functions or use inline assembly
*/
   unsigned int temp;

   asm("di");                          // Disable all interrupts

   temp = _CP0_GET_STATUS();           // Get Status
   temp |= 0x00400000;                 // Set BEV bit
   _CP0_SET_STATUS(temp);              // Update Status

   _CP0_SET_EBASE(0xBD000000);         // Set an EBase value of 0xBD000000
   _CP0_SET_INTCTL(0x00000020);        // Set the Vector Spacing to non-zero value

   temp = _CP0_GET_CAUSE();            // Get Cause
   temp |= 0x00800000;                 // Set IV
   _CP0_SET_CAUSE(temp);               // Update Cause

   temp = _CP0_GET_STATUS();           // Get Status
   temp &= 0xFFBFFFFD;                 // Clear BEV and EXL
   _CP0_SET_STATUS(temp);              // Update Status

   INTCONSET = 0x1000;                 // Set MVEC bit

   asm("ie");                          // Enable all interrupts
```

9.5 Interrupt Priorities

9.5.1 INTERRUPT GROUP PRIORITY

The user is able to assign a group priority to each of the interrupt vectors. The groups' priority level bits are located in the IPCx register. Each IPCx register contains group priority bits for four interrupt vectors. The user-selectable priority levels range from 1 (the lowest priority) to 7 (the highest). If an interrupt priority is set to zero, the interrupt vector is disabled for both interrupt and wake-up purposes. Interrupt vectors with a higher priority level preempt lower priority interrupts. The user must move the Requested Interrupt Priority bit of the Cause register, RIPLx (Cause<15:10>), into the Status register's Interrupt Priority bits, IPLx (Status<15:10>), before re-enabling interrupts. (The Cause and Status registers are located in the CPU; refer to Section 2.0 of this manual for more information.) This action will disable all lower priority interrupts until the completion of the Interrupt Service Routine.

Note:	The Interrupt Service Routine (ISR) must clear the associated interrupt flag in the IFSx register before lowering the interrupt priority level to avoid recursive interrupts.

EXAMPLE 9-3: SETTING GROUP PRIORITY LEVEL

```
/*
The following code example will set the priority to level 2.   Multi-Vector initialization
must be performed (See Example 9-2)
*/
IPC0CLR = 0x0000001C;              // clear the priority level
IPC0SET = 0x00000008;              // set priority level to 2
```

9.5.2 INTERRUPT SUBPRIORITY

The user can assign a subpriority level within each group priority. The subpriority will not cause preemption of an interrupt in the same priority; rather, if two interrupts with the same priority are pending, the interrupt with the highest subpriority will be handled first. The subpriority bits are located in the IPCx register. Each IPCx register contains subpriority bits for four of the interrupt vectors. These bits define the subpriority within the priority level of the vector. The user-selectable subpriority levels range from 0 (the lowest subpriority) to 3 (the highest).

EXAMPLE 9-4: SETTING SUBPRIORITY LEVEL

```
/*
The following code example will set the subpriority to level 2.  Multi-Vector initialization
must be performed (See Example 9-2)
*/

IPC0CLR = 0x00000003;              // clear the subpriority level
IPC0SET = 0x00000002;              // set the subpriority to 2
```

 Advance Information

9.6 Interrupt Processing

When the priority of a requested interrupt is greater than the current CPU priority, the interrupt request is taken and the CPU branches to the vector address associated with the requested interrupt. Depending on the priority of the interrupt, the prologue and epilogue of the interrupt handler must perform certain tasks before executing any useful code. The following examples provide recommended prologues and epilogues.

9.6.1 INTERRUPT PROCESSING IN SINGLE VECTOR MODE

When the interrupt controller is configured in Single Vector mode, all of the interrupt requests are serviced at the same vector address. The interrupt handler routine must generate a prologue and an epilogue to properly configure, save and restore all of the core registers, along with General Purpose Registers. At a worst case, all of the modifiable General Purpose Registers must be saved and restored by the prologue and epilogue.

9.6.1.1 Single Vector Mode Prologue

When entering the interrupt handler routine, the interrupt controller must first save the current priority and exception PC counter from Interrupt Priority bits, IPLx (Status<15:10>), and the ErrorEPC register, respectively, on the stack. (Status and ErrorEPC are CPU registers.) If the routine is presented a new register set, the previous register set's stack register must be copied to the current set's stack register. Then, the requested priority may be stored in the IPLx from the Requested Interrupt Priority bits, RIPLx (Cause<15:10>), Exception Level bit, EXL, and Error Level bit, ERL, in the Status register (Status<1> and Status<2>) are cleared and the Master Interrupt Enable bit (Status<0>) is set. Finally, the General Purpose Registers will be saved on the stack. (The Cause and Status registers are located in the CPU.)

EXAMPLE 9-5: SINGLE VECTOR INTERRUPT HANDLER PROLOGUE IN ASSEMBLY CODE

```
rdpgpr sp, sp
mfc0   k0, Cause
mfc0   k1, EPC
srl    k0, k0, 0xa
addiu  sp, sp, -76
sw     k1, 0(sp)
mfc0   k1, Status
sw     k1, 4(sp)
ins    k1, k0, 10, 6
ins    k1,zero, 1, 4
mtc0   k1, Status
sw     s8, 8(sp)
sw     a0, 12(sp)
sw     a1, 16(sp)
sw     a2, 20(sp)
sw     a3, 24(sp)
sw     v0, 28(sp)
sw     v1, 32(sp)
sw     t0, 36(sp)
sw     t1, 40(sp)
sw     t2, 44(sp)
sw     t3, 48(sp)
sw     t4, 52(sp)
sw     t5, 56(sp)
sw     t6, 60(sp)
sw     t7, 64(sp)
sw     t8, 68(sp)
sw     t9, 72(sp)
addu    s8, sp, zero

// start interrupt handler code here
```

9.6.1.2 Single Vector Mode Epilogue

After completing all useful code of the interrupt handler routine, the original state of the Status and EPC registers, along with the General Purpose Registers saved on the stack, must be restored.

EXAMPLE 9-6: SINGLE VECTOR INTERRUPT HANDLER EPILOGUE IN ASSEMBLY CODE

```
// end of interrupt handler code

addu   sp, s8, zero
lw     t9, 72(sp)
lw     t8, 68(sp)
lw     t7, 64(sp)
lw     t6, 60(sp)
lw     t5, 56(sp)
lw     t4, 52(sp)
lw     t3, 48(sp)
lw     t2, 44(sp)
lw     t1, 40(sp)
lw     t0, 36(sp)
lw     v1, 32(sp)
lw     v0, 28(sp)
lw     a3, 24(sp)
lw     a2, 20(sp)
lw     a1, 16(sp)
lw     a0, 12(sp)
lw     s8, 8(sp)
di
lw     k0, 0(sp)
mtc0   k0, EPC
lw     k0, 4(sp)
mtc0   k0, Status
eret
```

9.6.2 INTERRUPT PROCESSING IN MULTI-VECTOR MODE

When the interrupt controller is configured in Multi-Vector mode, the interrupt requests are serviced at the calculated vector addresses. The interrupt handler routine must generate a prologue and an epilogue to properly configure, save and restore all of the core registers, along with General Purpose Registers. At a worst case, all of the modifiable General Purpose Registers must be saved and restored by the prologue and epilogue. If the interrupt priority is set to receive its own General Purpose Register set, the prologue and epilogue will not need to save or restore any of the modifiable General Purpose Registers, thus providing the lowest latency.

9.6.2.1 Multi-Vector Mode Prologue

When entering the interrupt handler routine, the Interrupt Service Routine (ISR) must first save the current priority and exception PC counter from Interrupt Priority bits, IPL (Status<15:10>), and the ErrorEPC register, respectively, on the stack. If the routine is presented a new register set, the previous register set's stack register must be copied to the current set's stack register. Then, the requested priority may be stored in the IPLx from Requested Interrupt Priority bits, RIPLx (Cause<15:10>), Exception Level bit, EXL, and Error Level bit, ERL, in the Status register (Status<1> and Status<2>) are cleared, and the Master Interrupt Enable bit (Status<0>) is set. If the interrupt handler is not presented a new General Purpose Register set, these resisters will be saved on the stack. (Cause and Status are CPU registers; refer to Section 2.0 of this manual for more information.)

EXAMPLE 9-7: PROLOGUE WITHOUT A DEDICATED GENERAL PURPOSE REGISTER SET IN ASSEMBLY CODE

```
rdpgpr sp, sp
mfc0   k0, Cause
mfc0   k1, EPC
srl    k0, k0, 0xa
addiu  sp, sp, -76
sw     k1, 0(sp)
mfc0   k1, Status
sw     k1, 4(sp)
ins    k1, k0, 10, 6
ins    k1,zero, 1, 4
mtc0   k1, Status
sw     s8, 8(sp)
sw     a0, 12(sp)
sw     a1, 16(sp)
sw     a2, 20(sp)
sw     a3, 24(sp)
sw     v0, 28(sp)
sw     v1, 32(sp)
sw     t0, 36(sp)
sw     t1, 40(sp)
sw     t2, 44(sp)
sw     t3, 48(sp)
sw     t4, 52(sp)
sw     t5, 56(sp)
sw     t6, 60(sp)
sw     t7, 64(sp)
sw     t8, 68(sp)
sw     t9, 72(sp)
addu    s8, sp, zero

// start interrupt handler code here
```

EXAMPLE 9-8: PROLOGUE WITH A DEDICATED GENERAL PURPOSE REGISTER SET IN ASSEMBLY CODE

```
rdpgpr sp, sp
mfc0   k0, Cause
mfc0   k1, EPC
srl    k0, k0, 0xa
addiu  sp, sp, -76
sw     k1, 0(sp)
mfc0   k1, Status
sw     k1, 4(sp)
ins    k1, k0, 10, 6
ins    k1,zero, 1, 4
mtc0   k1, Status
addu    s8, sp, zero

// start interrupt handler code here
```

9.6.2.2 Multi-Vector Mode Epilogue

After completing all useful code of the interrupt handler routine, the original state of the Status and ErrorEPC registers, along with the General Purpose Registers saved on the stack, must be restored. (The Status and ErrorEPC registers are located in the CPU; refer to Section 2.0 of this manual for more information.)

EXAMPLE 9-9: EPILOGUE WITHOUT A DEDICATED GENERAL PURPOSE REGISTER SET IN ASSEMBLY CODE

```
// end of interrupt handler code

addu  sp, s8, zero
lw     t9, 72(sp)
lw     t8, 68(sp)
lw     t7, 64(sp)
lw     t6, 60(sp)
lw     t5, 56(sp)
lw     t4, 52(sp)
lw     t3, 48(sp)
lw     t2, 44(sp)
lw     t1, 40(sp)
lw     t0, 36(sp)
lw     v1, 32(sp)
lw     v0, 28(sp)
lw     a3, 24(sp)
lw     a2, 20(sp)
lw     a1, 16(sp)
lw     a0, 12(sp)
lw     s8, 8(sp)
di
lw     k0, 0(sp)
mtc0   k0, EPC
lw     k0, 4(sp)
mtc0   k0, Status
eret
```

EXAMPLE 9-10: EPILOGUE WITH A DEDICATED GENERAL PURPOSE REGISTER SET IN ASSEMBLY CODE

```
// end of interrupt handler code

addu  sp, s8, zero
di
lw     k0, 0(sp)
mtc0   k0, EPC
lw     k0, 4(sp)
mtc0   k0, Status
eret
```

9.7 External Interrupts

The interrupt controller supports five external interrupt-request signals (INT4-INT0). These inputs are edge sensitive; they require a low-to-high or a high-to-low transition to create an interrupt request. The INTCON register has five bits that select the polarity of the edge detection circuitry: INT4EP (INTCON<4>), INT3EP (INTCON<3>), INT2EP (INTCON<2>), INT1EP (INTCON<1>) and INT0EP (INTCON<0>).

Note:	Changing the external interrupt polarity may trigger an interrupt request. It is recommended that before changing the polarity, the user disables that interrupt, changes the polarity, clears the interrupt flag and re-enables the interrupt.

EXAMPLE 9-11: SETTING EXTERNAL INTERRUPT POLARITY

```
/*
The following code example will set INT3 to trigger on a high to low transition edge. The CPU
must be set up for either multi or single vector interrupts to handle external interrupts
*/
IEC0CLR = 0x00008000;              // disable INT3
INTCONCLR = 0x00000008;            // clear the bit for falling edge trigger
IFS0CLR = 0x00008000;              // clear the interrupt flag
IEC0SET = 0x00008000;              // enable INT3
```


9.8 Temporal Proximity Interrupt Coalescing

The PIC32MX Family CPU responds to interrupt events as if they are all immediately critical because the interrupt controller asserts the interrupt request to the CPU when the interrupt request occurs. The CPU immediately recognizes the interrupt if the current CPU priority is lower than the pending priority. Entering and exiting an ISR consumes clock cycles for saving and restoring context. Events are asynchronous with respect to the main program and have a limited possibility of occurring simultaneously or close together in time. This prevents the ability of a shared ISR to process multiple interrupts at one time.

Temporal proximity interrupt uses the interrupt proximity timer, TPTMR, to create a temporal window in which a group of interrupts of the same, or lower, priority will be held off. The user can activate temporal proximity interrupt coalescing by performing the following steps:

- Set the TPCx bit to the preferred priority level. (Setting TPC to zero will disable the proximity timer.)
- Load the preferred 32-bit value to TPTMR.

The interrupt proximity timer will trigger when an interrupt request of a priority equal, or lower, matches the TPC value.

EXAMPLE 9-12: TEMPORAL PROXIMITY INTERRUPT COALESCING EXAMPLE

```
/*
The following code example will set the Temporal Proximity Coalescing to trigger on interrupt
priority level of 3 or below and the temporal timer to be set to 0x12345678.
*/

INTCONCLR = 0x00000700;            // clear TPC
TPTMPCLR  = 0xFFFFFFFF;            // clear the timer
NTCONSET = 0x00000300;             // set TPC->3
TPTMR = 0x12345678;                // set the timer to 0x12345678
```

NOTES:

10.0 OSCILLATORS

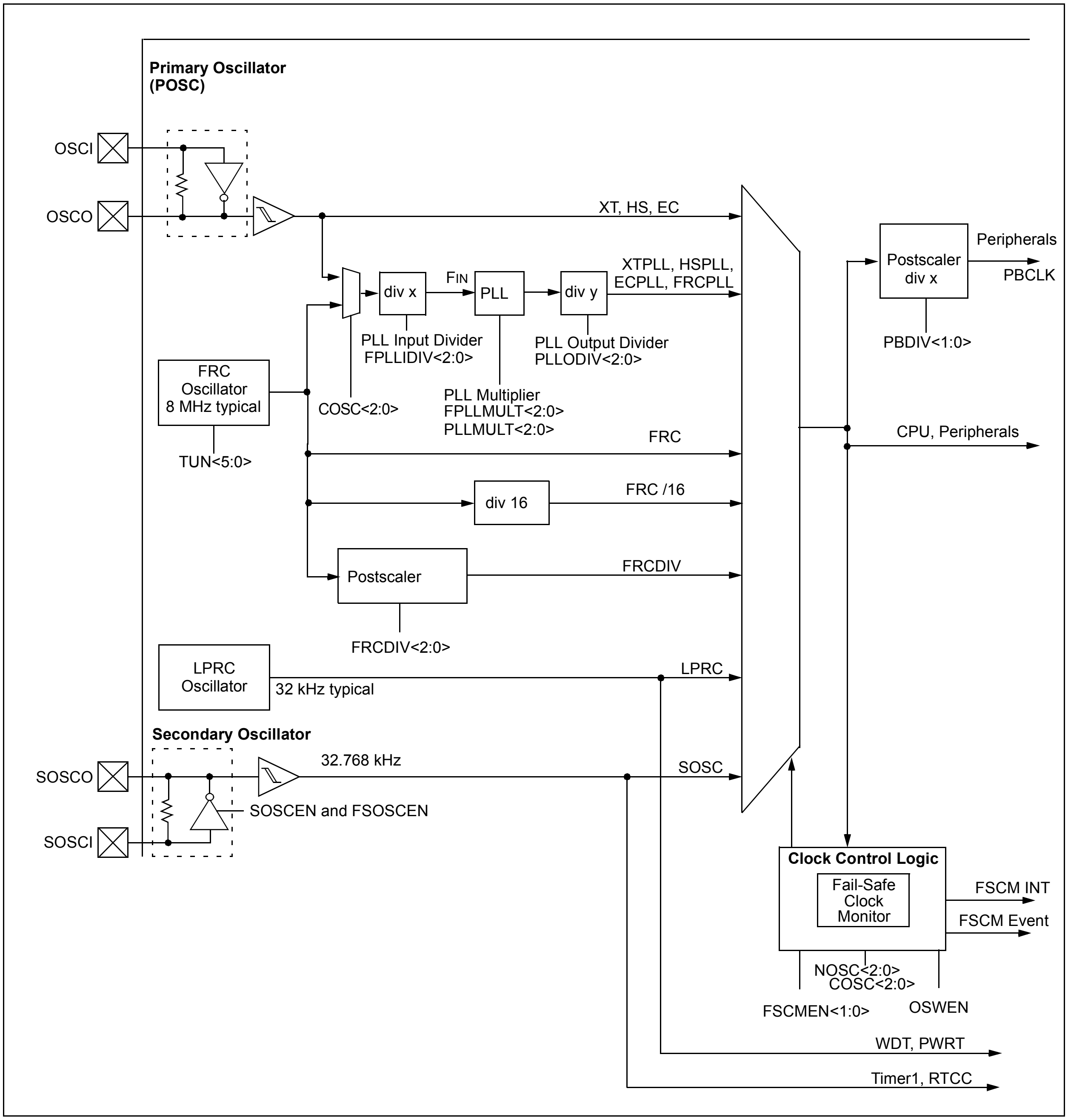
Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

This section describes the PIC32MX Family oscillator system and its operation. The PIC32MX Family oscillator system has the following modules and features:

- A total of four external and internal oscillator options as clock sources
- On-chip PLL with user-selectable input divider, multiplier, and output divider to boost operating frequency on select internal and external oscillator sources
- On-chip user-selectable divisor postscaler on select oscillator sources
- Software-controllable switching between various clock sources
- A Fail-Safe Clock Monitor (FSCM) that detects clock failure and permits safe application recovery or shutdown

A simplified diagram of the oscillator system is shown in Figure 10-1.

FIGURE 10-1: PIC32MX FAMILY CLOCK DIAGRAM



 Advance Information

10.1 Control Registers

The Oscillator module consists of the following Special Function Registers (SFRs):

- OSCCON: Control Register for the Oscillator module

  OSCCONCLR, OSCCONSET, OSCCONINV: Atomic Bit Manipulation Write-only Registers for OSCCON register
- OSCTUN: FRC Tuning Register for the Oscillator module

  OSCTUNCLR, OSCTUNSET, OSCTUNINV: Atomic Bit Manipulation Write-only Registers for OSCTUN register

The Oscillator module also has the following associated bits for interrupt control:

- Interrupt Flag Status bits (IFS1<14>) for Clock Fail FSCMIF in IFS1 Interrupt register
- Interrupt Enable Control bits (IEC1<14>) for Clock Fail FSCMIE in IEC1 Interrupt register
- Interrupt Priority Control bits (FSCMIP<12:10>) for Clock Fail in IPC8 Interrupt register
- Interrupt Subpriority Control bits (FSCMIP<9:8>) for Clock Fail in IPC8 Interrupt register

The following tables provide brief summaries of Oscillator-module-related registers. Corresponding registers appear after the summaries, followed by a detailed description of each register.

TABLE 10-1: OSCILLATORS SFR SUMMARY

Virtual Address	Name		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
BF80_F000	OSCCON	31:24	—	—	PLLODIV<2:0>			FRCDIV<2:0>		
		23:16	—	SOSCRDY	—	PBDIV<1:0>		PLLMULT<2:0>		
		15:8	—	COSC<2:0>			—	NOSC<2:0>		
		7:0	CLKLOCK	—	SLOCK	SLPEN	CF	—	SOSCEN	OSWEN
BF80_F004	OSCCONCLR	31:0	Write clears selected bits in OSCCON, read yields undefined value							
BF80_F008	OSCCONSET	31:0	Write sets selected bits in OSCCON, read yields undefined value							
BF80_F00C	OSCCONINV	31:0	Write inverts selected bits in OSCCON, read yields undefined value							
BF80_F010	OSCTUN	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	—	—	TUN<5:0>					
BF80_F014	OSCTUNCLR	31:0	Write clears selected bits in OSCTUN, read yields undefined value							
BF80_F018	OSCTUNSET	31:0	Write sets selected bits in OSCTUN, read yields undefined value							
BF80_F01C	OSCTUNINV	31:0	Write inverts selected bits in OSCTUN, read yields undefined value							
BF80_0000	WDTCON		—	—	—	—	—	—	—	—
			—	—	—	—	—	—	—	—
		15:8	ON	—	—	—	—	—	—	—
			—	WDTPS<4:0>					—	WDTCLR
BF80_0004	WDTCONCLR	31:0	Write clears selected bits in WDTCON, read yields an undefined value							
BF80_0008	WDTCONSET	31:0	Write sets selected bits in WDTCON, read yields an undefined value							
BF80_000C	WDTCONINV	31:0	Write inverts selected bits in WDTCON, read yields an undefined value							
BF88_1040	IFS1	15:8	RTCCIF	FSCMIF	I2C2MIF	I2C2SIF	I2C2BIF	U2TXIF	U2RXIF	U2EIF
BF88_1070	IEC1	15:8	RTCCIE	FSCMIE	I2C2MIE	I2C2SIE	I2C2BIE	U2TXIE	U2RXIE	U2EIE
BF88_1110	IPC8	23:16	—	—	—	—	FSCMIP<2:0>			FSC-MIS<1:0>
BFC0_2FF8	DEVCFG1	31:24	—	—	—	—	—	—	—	—
		23:16	FWDTEN	—	—	FWDTPS<4:0>				
		15:8	FCKSM<1:0>		FPBDIV<1:0>		—	OSCIOFNC	POSCMD<1:0>	
		7:0	IESO	—	FSOSCEN	—	—	FNOSC<2:0>		
BFC0_2FF4	DEVCFG2	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	FPLLODIV<2:0>		
		15:8	—	—	—	—	—	—	—	—
		7:0	—	FPLLMULT<2:0>			—	FPLLIDIV<2:0>		

REGISTER 10-1: OSCCON: OSCILLATOR CONTROL REGISTER

U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-0	R/W-0	R/W-1
—	—	PLLODIV<2:0>			FRCDIV<2:0>		
bit 31							bit 24

U-0	R-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	SOSCRDY	—	PBDIV<1:0>		PLLMULT<2:0>		
bit 23							bit 16

U-0	R-0	R-0	R-0	U-0	R/W-x	R/W-x	R/W-x
—	COSC<2:0>			—	NOSC<2:0>		
bit 15							bit 8

R/W-0	r-0	R-0	R/W-0	R/W-0	r-0	R/W-0	R/W-0
CLKLOCK	—	SLOCK	SLPEN	CF	—	SOSCEN	OSWEN
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-30 **Unimplemented:** Read as ‘0’

bit 29-27 **PLLODIV<2:0>:** Output Divider for PLL bits

111 = PLL output divided by 256
110 = PLL output divided by 64
101 = PLL output divided by 32
100 = PLL output divided by 16
011 = PLL output divided by 8
010 = PLL output divided by 4
001 = PLL output divided by 2
000 = PLL output divided by 1

Note: On Reset these bits are set to the value of the FPLLODIV Configuration bits (DEVCFG2<18:16>).

bit 26-24 **FRCDIV<2:0>:** Fast Internal RC Clock Divider bits

111 = FRC divided by 256
110 = FRC divided by 64
101 = FRC divided by 32
100 = FRC divided by 16
011 = FRC divided by 8
010 = FRC divided by 4
001 = FRC divided by 2 (default setting)
000 = FRC divided by 1

bit 23 **Unimplemented:** Maintain as ‘0’

bit 22 **SOSCRDY:** Secondary Oscillator Ready Indicator bit

1 = Indicates that the Secondary Oscillator is running and is stable
0 = Secondary oscillator is either turned off or is still warming up

bit 21 **Unimplemented:** Read as ‘0’

 Advance Information

REGISTER 10-1: OSCCON: OSCILLATOR CONTROL REGISTER

bit 20-19 **PBDIV<1:0>:** Peripheral Bus Clock Divisor bits

11 = PBCLK is SYSCLK divided by 8
10 = PBCLK is SYSCLK divided by 4
01 = PBCLK is SYSCLK divided by 2
00 = PBCLK is SYSCLK divided by 1
Note: Initial value is loaded from DEVCFG1<13:12>.

bit 18-16 **PLLMULT<2:0>:** PLL Multiplier bits

111 = Clock is multiplied by 24
110 = Clock is multiplied by 21
101 = Clock is multiplied by 20
100 = Clock is multiplied by 19
011 = Clock is multiplied by 18
010 = Clock is multiplied by 17
001 = Clock is multiplied by 16
000 = Clock is multiplied by 15
Note: On Reset these bits are set to the value of the FPLLMULT Configuration bits (DEVCFG2<6:4>).

bit 15 **Unimplemented:** Read as '0'

bit 14-12 **COSC<2:0>:** Current Oscillator Selection bits

111 = Fast Internal RC Oscillator divided by OSCCON.FRCDIV
110 = Fast Internal RC Oscillator divided by 16
101 = Low-Power Internal RC Oscillator (LPRC)
100 = Secondary Oscillator (SOSC)
011 = Primary Oscillator with PLL module (XTPLL, HSPLL or ECPLL)
010 = Primary Oscillator (XT, HS or EC)
001 = Fast RC Oscillator with PLL module via Postscaler (FRCPLL)
000 = Fast RC Oscillator (FRC)
Note: On Reset these bits are set to the value of the FNOSC Configuration bits (DEVCFG1<2:0>).

bit 11 **Unimplemented:** Read as '0'

bit 10-8 **NOSC<2:0>:** New Oscillator Selection bits

111 = Fast Internal RC Oscillator divided by OSCCON.FRCDIV
110 = Fast Internal RC Oscillator divided by 16
101 = Low-Power Internal RC Oscillator (LPRC)
100 = Secondary Oscillator (SOSC)
011 = Primary Oscillator with PLL module (XTPLL, HSPLL or ECPLL)
010 = Primary Oscillator (XT, HS or EC)
001 = Fast Internal RC Oscillator with PLL module via Postscaler (FRCPLL)
000 = Fast Internal RC Oscillator (FRC)
Note: On Reset these bits are set to the value of the FNOSC Configuration bits (DEVCFG1<2:0>).

bit 7 **CLKLOCK:** Clock Selection Lock Enable bit

If FSCM is enabled (FCKSM1 =1):
1 = Clock and PLL selections are locked.
0 = Clock and PLL selections are not locked and may be modified
If FSCM is disabled (FCKSM1 =0):
Note: Clock and PLL selections are never locked and may be modified.

bit 6 **Reserved:** Maintain as '0'

bit 5 **SLOCK:** PLL Lock Status bit

1 = PLL module is in lock or PLL module start-up timer is satisfied
0 = PLL module is out of lock, PLL start-up timer is running or PLL is disabled

bit 4 **SLPEN:** Sleep Mode Enable bit

1 = Device will enter Sleep mode when a `WAIT` instruction is executed
0 = Device will enter Idle mode when a `WAIT` instruction is executed

REGISTER 10-1: OSCCON: OSCILLATOR CONTROL REGISTER

bit 3	**CF:** Clock Fail Detect bit 1 = FSCM (Fail Safe Clock Monitor) has detected a clock failure 0 = No clock failure has been detected
bit 2	**Reserved:** Maintain as '0'
bit 1	**SOSCEN:** 32.768 kHz Secondary Oscillator (SOSC) Enable bit 1 = Enable Secondary Oscillator 0 = Disable Secondary Oscillator
bit 0	**OSWEN:** Oscillator Switch Enable bit 1 = Initiate an oscillator switch to selection specified by NOSC2:NOSC0 bits 0 = Oscillator switch is complete

REGISTER 10-2: OSCTUN: FRC TUNING REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	TUN<5:0>					
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31:6 **Unimplemented:** Read as ‘0’

bit 5-0 **TUN<5:0>:** FRC Oscillator Tuning bits

011111 = Maximum frequency.
011110 =
•
000001 =
000000 = Center frequency. Oscillator runs at calibrated frequency.
111111 =
•
100001 =
100000 = Minimum frequency.

REGISTER 10-3: WDTCON: WATCHDOG TIMER CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	U-0	U-0	U-0	U-0	R-1	R-1	R-0
ON	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R-x	R-x	R-x	R-x	R-x	r-0	R/W-0
—	WDTPS<4:0>					—	WDTCLR
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 15 **ON:** Watchdog Timer Enable bit

1 = Enables the WDT if it is not enabled by the device configuration
0 = Disable the WDT if it was enabled in software

Note 1: A read of this bit will result in a ‘1’ if the WDT is enabled by the device configuration or by software.

2: The LPRC oscillator will automatically be enabled when this bit is set.

Note: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the oscillator.

REGISTER 10-4: IFS1: INTERRUPT FLAG STATUS REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	r-0	R/W-0
—	—	—	—	—	—	—	FCEIF
bit 31							bit 24

r-0	r-0	r-0	r-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	DMA3IF	DMA2IF	DMA1IF	DMA0IF
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RTCCIF	FSCMIF	I2C2MIF	I2C2SIF	I2C2BIF	U2TXIF	U2RXIF	U2EIF
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPI2RXIF	SPI2TXIF	SPI2EIF	CMP2IF	CMP1IF	PMPIF	AD1IF	CNIF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 14 **FSCMIF:** Fail-Safe Clock Monitor Interrupt Flag bit

1 = Interrupt request has occured

0 = No interrupt request has a occurred

Note: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the oscillator.

REGISTER 10-5: IEC1: INTERRUPT ENABLE CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	r-0	R/W-0
—	—	—	—	—	—	—	FCEIE
bit 31							bit 24

r-0	r-0	r-0	r-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	DMA3IE	DMA2IE	DMA1IE	DMA0IE
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RTCCIE	FSCMIE	I2C2MIE	I2C2SIE	I2C2BIE	U2TXIE	U2RXIE	U2EIE
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPI2RXIE	SPI2TXIE	SPI2EIE	CMP2IE	CMP1IE	PMPIE	AD1IE	CNIE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 14 **FSCMIE:** Fail-Safe Clock Monitor Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

Note: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the oscillator.

REGISTER 10-6: IPC8: INTERRUPT PRIORITY CONTROL REGISTER 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	DMA0IP<2:0>			DMA0IS<1:0>	
bit 31							bit 24

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RTCCIP<2:0>			RTCCIS<1:0>	
bit 23							bit 16

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FSCMIP<2:0>			FSCMIS<1:0>	
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	I2C2IP<2:0>			I2C2IS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 12-10 **FSCMIP<2:0>:** Fail-Safe Clock Monitor Interrupt Priority bits

`111` = Interrupt priority is 7
`110` = Interrupt priority is 6
`101` = Interrupt priority is 5
`100` = Interrupt priority is 4
`011` = Interrupt priority is 3
`010` = Interrupt priority is 2
`001` = Interrupt priority is 1
`000` = Interrupt is disabled

bit 9-8 **FSCMIS<1:0>:** Fail-Safe Clock Monitor Interrupt Subpriority bits

`11` = Interrupt subpriority is 3
`10` = Interrupt subpriority is 2
`01` = Interrupt subpriority is 1
`00` = Interrupt subpriority is 0

Note: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the oscillator.

REGISTER 10-7: DEVCFG1 BOOT CONFIGURATION REGISTER

r-1	r-1	r-1	r-1	r-1	r-1	r-1	r-1
—	—	—	—	—	—	—	—
bit 31							bit 24

R/P-1	R/P-1	r-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
FWDTEN	—	—	FWDTPS4	FWDTPS3	FWDTPS2	FWDTPS1	FWDTPS0
bit 23							bit 16

R/P-1	R/P-1	R/P-1	R/P-1	r-1	R/P-1	R/P-1	R/P-1
FCKSM<1:0>		FPBDIV<1:0>		—	OSCIOFNC	POSCMD<1:0>	
bit 15							bit 8

R/P-1	r-1	R/P-1	r-1	r-1	R/P-1	R/P-1	R/P-1
IESO	—	FSOSCEN	—	—	FNOSC2	FNOSC1	FNOSC0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Maintain ‘1’

bit 15-14 **FCKSM<1:0>:** Fail-safe Clock Monitor (FSCM) and Clock Switch Configuration bits

1x = FSCM and Clock Switching are disabled
01 = Clock Switching is enabled, FSCM is disabled
00 = Clock Switching and FSCM are enabled

bit 13-12 **FPBDIV<1:0>:** Peripheral Bus Clock Divisor Default Value bits

11 = PBCLK is SYSCLK divided by 8
10 = PBCLK is SYSCLK divided by 4
01 = PBCLK is SYSCLK divided by 2
00 = PBCLK is SYSCLK divided by 1

bit 11 **Unimplemented:** Maintain as ‘1’

bit 10 **OSCIOFNC:** CLKO Enable Configuration bit

1 = CLKO output signal active on the OSCO pin; primary oscillator must be disabled or configured for the External Clock mode (EC) for the CLKO to be active (POSCMD<1:0> = 11 or = 00)
0 = CLKO output disabled

bit 9-8 **POSCMD<1:0>:** Primary Oscillator Configuration bits

11 = Primary Oscillator Disabled
10 = HS mode
01 = XT Mode
00 = EC Mode

bit 7 **IESO:** Internal External Clock Switchover Select bit

1 = Internal External Clock Switchover mode enabled; Two-Speed Start-up mode
0 = Internal External Clock Switchover mode disabled; Single-Speed Start-up mode

bit 5 **FSOSCEN:** Secondary Oscillator Enable bit

1 = Enable secondary oscillator
0 = Disable secondary oscillator

REGISTER 10-7: DEVCFG1 BOOT CONFIGURATION REGISTER

bit 2-0 **FNOSC<2:0>:** CPU Clock Oscillator Select bits

111 = Fast RC Oscillator with divide-by-N (FRCDIV)
110 = FRC Divided by 16 (FRCDIV16)
101 = Low-Power RC Oscillator (LPRC)
100 = Secondary Oscillator (SOSC)
011 = Primary Oscillator with PLL (XTPLL, HSPLL, or ECPLL)
010 = Primary Oscillator without PLL (XT, HS, or EC)
001 = Fast RC Oscillator with PLL
000 = Fast RC Oscillator (FRC)

REGISTER 10-8: DEVCFG2 BOOT CONFIGURATION REGISTER

r-1	r-1	r-1	r-1	r-1	r-1	r-1	r-1
—	—	—	—	—	—	—	—
bit 31							bit 24

r-1	r-1	r-1	r-1	r-1	R/P-1	R/P-1	R/P-1
—	—	—	—	—	FPLLODIV<2:0>		
bit 23							bit 16

r-1	r-1	r-1	r-1	r-1	r-1	r-1	r-1
—	—	—	—	—	—	—	—
bit 15							bit 8

U-1	R/P-1	R/P-1	R/P-1	U-1	R/P-1	R/P-1	R/P-1
—	FPLLMULT<2:0>			—	FPLLIDIV<2:0>		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 18-16 **FPLLODIV<2:0>:** Default Postscaler for PLL bits
- `111` = PLL output divided by 256
- `110` = PLL output divided by 64
- `101` = PLL output divided by 32
- `100` = PLL output divided by 16
- `011` = PLL output divided by 8
- `010` = PLL output divided by 4
- `001` = PLL output divided by 2
- `000` = PLL output divided by 1 (default setting)

bit 6-4 **FPLLMULT<2:0>:** Default PLL Multiplier Value bits
- `111` = 24x multiplier
- `110` = 21x multiplier
- `101` = 20x multiplier
- `100` = 19x multiplier
- `011` = 18x multiplier
- `010` = 17x multiplier
- `001` = 16x multiplier
- `000` = 15x multiplier

bit 2-0 **FPLLIDIV<2:0>:** Default PLL Input Divider Value bits
- `111` = Divide by 12
- `110` = Divide by 10
- `101` = Divide by 6
- `100` = Divide by 5
- `011` = Divide by 4
- `010` = Divide by 3
- `001` = Divide by 2
- `000` = Divide by 1

10.2 Operation: Clock Generation and Clock Sources

The PIC32MX device has two internal clocks: CPU clock and PB clock. They are derived from the currently selected clock source. The clock source can be chosen from the 4 available internal or external clock sources. Some of these clock sources have Phase Locked Loops (PLLs), programmable output dividers, or input divider to scale the input frequency to suit the application. The clock source can be changed on the fly by software. The oscillator control register is locked by hardware, it must be unlocked by a series of writes before software can perform a clock switch.

There are two main clocks in the PIC32MX Family device

- The System clock (SYSCLK) used by CPU and some peripherals
- The Peripheral Bus Clock (PBCLK) used by most peripherals

The PIC32MX Family clocks are derived from one of the following sources:

- Primary Oscillator (POSC) on the OSCI and OSCO pins
- Secondary Oscillator (SOSC) on the SOSCI and SOSCO pins
- Internal Fast RC Oscillator (FRC)
- Internal Low-Power RC Oscillator (LPRC)

Each of the clock sources has unique configurable options, such as a PLL, input divider, and/or output divider, that are detailed in their respective sections.

There are up to four internal clocks depending on the specific device. The clocks are derived from the currently selected oscillator source.

Note:	Clock sources for peripherals that use external clocks, such as the RTC and Timer1, are covered in their respective sections.

10.2.1 SYSTEM CLOCK (SYSCLK) GENERATION

The SYSCLK is the primary clock used by the CPU and select peripherals such as DMA, Interrupt Controller, and Prefetch Cache. The SYSCLK is derived from one of the four clock sources: POSC, SOSC, FRC, and LPRC. Some of the clock sources have specific clock multipliers and/or divider options. No clock scaling is applied other than the user specified values. The SYSCLK source is selected by the device configuration and can be changed by software during operation. The ability to switch clock sources during operation allows the application to reduce power consumption by reducing the clock speed. Refer to Table 10-2 for a list of SYSCLK sources.

TABLE 10-2: CLOCK SELECTION CONFIGURATION BIT VALUES

Oscillator Mode	Oscillator Source	POSCMD<1:0>	FNOSC2: FNOSC0	Notes
Fast RC Oscillator with Postscaler (FRCDIV)	Internal	xx	111	**1, 2**
Fast RC Oscillator divided by 16 (FRCDIV16)	Internal	xx	110	**1**
Low-Power RC Oscillator (LPRC)	Internal	xx	101	**1**
Secondary (Timer1/RTCC) Oscillator (SOSC)	Secondary	xx	100	**1**
Primary Oscillator (HS) with PLL Module (HSPLL)	Primary	10	011	**3**
Primary Oscillator (XT) with PLL Module (XTPLL)	Primary	01	011	**3**
Primary Oscillator (EC) with PLL Module (ECPLL)	Primary	00	011	**3**
Primary Oscillator (HS)	Primary	10	010	
Primary Oscillator (XT)	Primary	01	010	
Primary Oscillator (EC)	Primary	00	010	
Fast RC Oscillator with PLL Module (FRCPLL)	Internal	10	001	**1**
Fast RC Oscillator (FRC)	Internal	xx	000	**1**

Note 1: OSCO pin function as PBCLK out or Digital I/O is determined by the OSCIOFNC Configuration bit. When the pin is not required by the Oscillator mode it may be configured for one of these options.

2: Default Oscillator mode for an unprogrammed (erased) device.

3: When using the PLL modes the input divider must be chosen such that resulting frequency applied to the PLL is in the range of 4 MHz to 5 MHz.

10.2.1.1 Primary Oscillator (POSC)

The POSC has six operating modes, as summarized in Table 10-3. The first three modes can each be combined with a PLL module to form the last three modes. Figures 10-2 through 10-4 show various POSC configurations. The primary oscillator is connected to the OSCI and OSCO pins of the device family. The primary oscillator can be configured for an external clock input or an external crystal or resonator.

The XT, XTPLL, HS, and HSPLL modes are External Crystal or Resonator Controller Oscillator modes. The XT and HS modes are functionally very similar. The primary difference is the gain of the internal inverter of the oscillator circuit (see Figure 10-2). The XT mode is a medium power, medium frequency mode and has medium inverter gain. HS mode is higher power and provides the highest oscillator frequencies and has the highest inverter gain. OSCO provides crystal/resonator feedback in both XT and HS Oscillator modes and hence is not available for use as a input or output in these modes. The XTPLL and HSPLL modes have a Phase Locked Loop (PLL) with user selectable input divider, multiplier, and output divider to provide a wide range of output frequencies. The oscillator circuit will consume more current when the PLL is enabled.

The External Clock modes, EC and ECPLL, allow the system clock to be derived from an external clock source. These modes configure the OSCI pin as a high-impedance input that can be driven by a CMOS driver. The external clock can be used to drive the system clock directly (EC) or the ECPLL module with prescale and postscaler can be used to change the input clock frequency (ECPLL). The External Clock modes also disables the internal feedback buffer allowing the OSCO pin to be used for other functions. In the External Clock mode the OSCO pin can be used as an additional device I/O pin (see Figure 10-4) or a PBCLK output pin (see Figure 10-3).

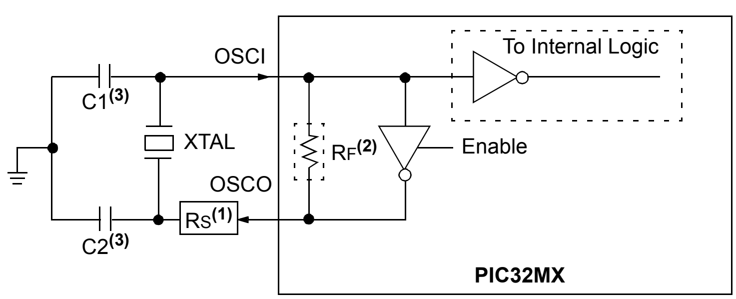
Note: When using the PLL modes the input divider must be chosen such that resulting frequency applied to the PLL is in the range of 4 MHz to 5 MHz.

TABLE 10-3: PRIMARY OSCILLATOR OPERATING MODES

Oscillator Mode	Description
HS	10 MHz-40 MHz crystal
XT	3.5 MHz-10 MHz resonator
EC	External clock input (0-72 MHz)
HSPLL	10 MHz-40 MHz crystal, PLL enabled
XTPLL	4 MHz-10 MHz resonator, PLL enabled
ECPLL	External clock input (5-72 MHz), PLL enabled

Note: The clock applied to the CPU after applicable prescalers, postscalers, and PLL multipliers must not exceed the maximum allowable processor frequency.

FIGURE 10-2: CRYSTAL OR CERAMIC RESONATOR OPERATION (XT, XTPLL, HS, OR HSPLL OSCILLATOR MODE)



Note 1: A series resistor, Rs, may be required for AT strip cut crystals.
2: The internal feedback resistor, RF, is typically in the range of 2 to 10 MΩ.
3: Refer to the "*PIC32MX Family Reference Manual*" (DS61132) for help determining the best oscillator components.

FIGURE 10-3: EXTERNAL CLOCK INPUT OPERATION WITH CLOCK-OUT (EC, ECPLL MODE)

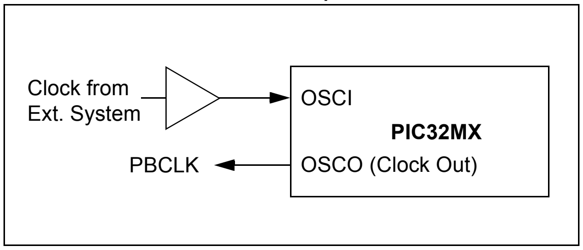
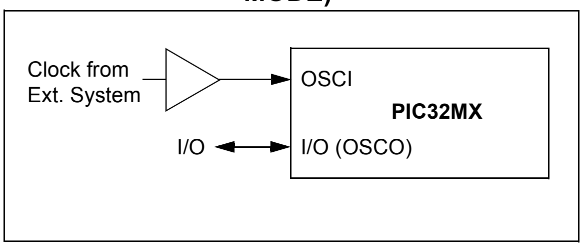


FIGURE 10-4: EXTERNAL CLOCK INPUT OPERATION WITH NO CLOCK-OUT (EC, ECPLL MODE)



10.2.1.2 Primary Oscillator (POSC) Configuration

To configure the POSC the following steps should be performed:

1. Select POSC as the default oscillator in the device Configuration register, DEVCFG1, by setting FNOSC<2:0> = '010' without PLL or '011' with PLL.
2. Select the desired mode HS, XT, or EC, using POSCMD<1:0> in DEVCFG1.
3. If the PLL is to be used:
   a) Select the appropriate Configuration bits for the PLL input divider to scale the input frequency to be between 4 MHz and 5 MHz using FPLLIDIV<2:0> in DEVCFG2.
   b) Select the desired PLL multiplier ratio using FPLLMULT<2:0>) in DEVCFG2.
   c) At runtime, select the desired PLL output divider using PLLODIV (OSCCON<29:27>) to provide the desired clock frequency. The default value is set by DEVCFG1.

10.2.1.3 Oscillator Start-up Timer

In order to ensure that a crystal oscillator (or ceramic resonator) has started and stabilized, an Oscillator Start-up Timer (OST) is provided. The OST is a simple 10-bit counter that counts 1024 T_{OSC} cycles before releasing the oscillator clock to the rest of the system. This time-out period is designated 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 has to restart (i.e., on POR, BOR and wake-up from Sleep mode). The Oscillator Start-up Timer is applied to the MS and HS modes for the primary oscillator, as well as the secondary oscillator, see **Section 10.2.1.5 "Secondary Oscillator (SOSC)"**.

10.2.1.4 System Clock Phase Locked Loop (PLL)

The system clock PLL provides a user configurable input divider, multiplier, and output divider which can be used with the XT, HS and EC Primary Oscillator modes and with the Internal Fast RC Oscillator (FRC) mode to create a variety of clock frequencies from a single clock source.

The Input divider, multiplier, and output divider control initial value bits are contained in the in the DEVCFG2 device Configuration register. The multiplier and output divider bits are also contained in the OSCCON register. As part of a device Reset, values from the device Configuration register, DEVCFG2, are copied to the OSCCON register. This allows the user to preset the input divider to provide the appropriate input frequency to the PLL and set an initial PLL multiplier when programming the device. At runtime the multiplier, divider and output divider can be changed by software to scale the clock frequency to suit the application. The PLL input divider cannot be changed at run time. This is to prevent applying an input frequency outside the specified limits to the PLL.

To configure the PLL the following steps are required:

1. Calculate the PLL input divider, PLL multiplier, and PLL output divider values.
2. Set the PLL input divider and the initial PLL multiplier value in the DEVCFG2 register when programming the part.
3. At runtime the PLL multiplier and PLL output divider can be changed to suit the application.

Combinations of PLL input divider, multiplier and output divider provide a combined multiplier of approximately 0.006 to 24 times the input frequency. For reliable operation the output of the PLL module must not exceed the maximum clock frequency of the device. The PLL input divider value should be chosen to limit the input frequency to the PLL to the range of 4 MHz to 5 MHz.

Due to the time required for the PLL to provide a stable output, a Status bit SLOCK (OSCCON<5>) is provided. When the clock input to the PLL is changed, this bit is driven low ('0'). After the PLL has achieved a lock or the PLL start-up timer has expired, the bit is set. The bit will be set upon the expiration of the timer even if the PLL has not achieved a lock.

TABLE 10-4: NET MULTIPLIER OUTPUT FOR SELECTED PLL AND OUTPUT DIVIDER VALUES

Multiplier	Postscaler	Net Multiplication factor	PLLODIV <2:0>	PLLMULT <2:0>	Multiplier	Postscaler	Net Multiplication factor	PLLODIV <2:0>	PLLMULT <2:0>
15	1	15	‘000’	‘000’	15	16	.938	‘100’	‘000’
16	1	16	‘000’	‘001’	16	16	1	‘100’	‘001’
17	1	17	‘000’	‘010’	17	16	1.063	‘100’	‘010’
18	1	18	‘000’	‘011’	18	16	1.125	‘100’	‘011’
19	1	19	‘000’	‘100’	19	16	1.188	‘100’	‘100’
20	1	20	‘000’	‘101’	20	16	1.250	‘100’	‘101’
21	1	21	‘000’	‘110’	21	16	1.313	‘100’	‘110’
24	1	24	‘000’	‘111’	24	16	1.5	‘100’	‘111’
15	2	7.5	‘001’	‘000’	15	32	.4688	‘101’	‘000’
16	2	8	‘001’	‘001’	16	32	.5	‘101’	‘001’
17	2	8.5	‘001’	‘010’	17	32	.5313	‘101’	‘010’
18	2	9	‘001’	‘011’	18	32	.5625	‘101’	‘011’
19	2	9.5	‘001’	‘100’	19	32	.5938	‘101’	‘100’
20	2	10	‘001’	‘101’	20	32	.6250	‘101’	‘101’
21	2	10.5	‘001’	‘110’	21	32	.6563	‘101’	‘110’
24	2	12	‘001’	‘111’	24	32	.7500	‘101’	‘111’
15	4	3.75	‘010’	‘000’	15	64	.234	‘110’	‘000’
16	4	4	‘010’	‘001’	16	64	.250	‘110’	‘001’
17	4	4.25	‘010’	‘010’	17	64	.266	‘110’	‘010’
18	4	4.5	‘010’	‘011’	18	64	.281	‘110’	‘011’
19	4	4.75	‘010’	‘100’	19	64	.297	‘110’	‘100’
20	4	5	‘010’	‘101’	20	64	.313	‘110’	‘101’
21	4	5.25	‘010’	‘110’	21	64	.328	‘110’	‘110’
24	4	6	‘010’	‘111’	24	64	.375	‘110’	‘111’
15	8	1.875	‘011’	‘000’	15	256	.05859	‘111’	‘000’
16	8	2	‘011’	‘001’	16	256	.06250	‘111’	‘001’
17	8	2.125	‘011’	‘010’	17	256	.06641	‘111’	‘010’
18	8	2.250	‘011’	‘011’	18	256	.07031	‘111’	‘011’
19	8	2.375	‘011’	‘100’	19	256	.07422	‘111’	‘100’
20	8	2.5	‘011’	‘101’	20	256	.07813	‘111’	‘101’
21	8	2.625	‘011’	‘110’	21	256	.08203	‘111’	‘110’
24	8	3	‘011’	‘111’	24	256	.09375	‘111’	‘111’

10.2.1.4.1 PLL Lock Status

The SLOCK bit (OSCCON<5>) is a read-only Status bit that indicates the lock status of the PLL. It is automatically set after the typical time delay for the PLL to achieve lock, also designated as TLOCK. If the PLL does not stabilize properly during start-up, SLOCK may not reflect the actual status of PLL lock, nor does it detect when the PLL loses lock during normal operation.

The SLOCK bit is cleared at a Power-on Reset and on clock switches when the PLL is selected as a destination clock source. It remains clear when any clock source not using the PLL is selected.

Refer to the Electrical Characteristics section in the specific device data sheet for further information on the PLL lock interval.

10.2.1.4.2 Primary Oscillator Start-up from Sleep Mode

To ensure reliable wake-up from Sleep, care must be taken to properly design the primary oscillator circuit. This is because the load capacitors have both partially charged to some quiescent value and phase differential at wake-up is minimal. Thus, more time is required to achieve stable oscillation. Remember also that low-voltage, high temperatures and the lower frequency clock modes also impose limitations on loop gain, which in turn, affects start-up.

Each of the following factors increases the start-up time:

- Low-frequency design (with a Low Gain Clock mode)
- Quiet environment (such as a battery operated device)
- Operating in a shielded box (away from the noisy RF area)
- Low voltage
- High temperature
- Wake-up from Sleep mode

10.2.1.5 Secondary Oscillator (SOSC)

The Secondary Oscillator (SOSC) is designed specifically for low-power operation with a external 32.768 kHz crystal. The oscillator is located on the SOSCO and SOSCI device pins and serves as a secondary crystal clock source for low-power operation. It can also drive Timer1 and/or the Real-Time Clock/Calendar module for Real-Time Clock applications.

10.2.1.5.1 Enabling the SOSC Oscillator

The SOSC is hardware enabled by the FSOSCEN Configuration bit (DEVCFG1<5>). Once SOSC is enabled, software can control it by modifying SOSCEN bit (OSCCON<1>). Setting SOSCEN enables the oscillator; the SOSCO and SOSCI pins are controlled by the oscillator and cannot be used for port I/O or other functions.

Note:	An unlock sequence is required before a write to OSCCON can occur. Refer to **Section 10.2.5.2 “Oscillator Switching Sequence”** for more information.

The Secondary Oscillator requires a warm-up period before it can be used as a clock source. When the oscillator is enabled, a warm-up counter increments to 1024. When the counter expires the SOSCRDY (OSCCON<22>) is set to ‘1’.

10.2.1.5.2 SOSC Continuous Operation

The SOSC is always enabled when SOSCEN (OSCCON<1>) is set. Leaving the oscillator running at all times allows a fast switch to the 32 kHz system clock for lower power operation. Returning to the faster main oscillator will still require an oscillator start-up time if it is a crystal type source and/or uses the PLL.

In addition, the oscillator will need to remain running at all times for Real-Time Clock applications and may be required for Timer1.

EXAMPLE 10-1: ENABLING THE SOSC

```
SYSKEY = 0x12345678;            // ensure OSCCON is locked
SYSKEY = 0xAA996655;            // Write Key1 to SYSKEY
SYSKEY = 0x556699AA;            // Write Key2 to SYSKEY
                                // OSCCON is now unlocked

                                // make the desired change
OSCCONSET = 2;                  // request clock switch

                                // Relock the SYSKEY
SYSKEY = 0x12345678;            // Write any value other than Key1 or Key2
```


10.2.1.6 Internal Fast RC Oscillator (FRC)

The FRC oscillator is a fast (8 MHz nominal), user trimmable, internal RC oscillator with user selectable input divider, PLL multiplier, and output divider.

10.2.1.6.1 FRC Postscaler Mode (FRCDIV)

Users are not limited to the nominal 8 MHz FRC output if they wish to use the fast internal oscillator as a clock source. An additional FRC mode, FRCDIV, implements a selectable output divider that allows the choice of a lower clock frequency from 7 different options, plus the direct 8 MHz output. The output divider is configured using the FRCDIV<2:0> bits (OSCCON<26:24>). Assuming a nominal 8 MHz output, available lower frequency options range from 4 MHz (divide-by-2) to 31 kHz (divide-by-256). The range of frequencies allows users the ability to save power at any time in an application by simply changing the FRCDIV bits. The FRCDIV mode is selected whenever the COSC bits (OSCCON<14:12>) are '`111`'.

10.2.1.6.2 FRC Oscillator with PLL Mode (FRCPLL)

The output of the FRC may also be combined with a user selectable PLL multiplier and output divider to produce a SYSCLK across a wide range of frequencies. The FRC PLL mode is selected whenever the COSC bits (OSCCON<14:12>) are '`001`'. In this mode, the PLL input divider is forced to '2' to provide a 4 MHz input to the PLL. The desired PLL multiplier and output divider values can be chosen to provide the desired device frequency

10.2.1.6.3 Oscillator Tune Register (OSCTUN)

The FRC Oscillator Tuning register OSCTUN allows the user to fine tune the FRC oscillator over a range of approximately ±12% (typical). Each bit increment or decrement changes the factory calibrated frequency of the FRC oscillator by a fixed amount.

10.2.1.7 Internal Low-Power RC Oscillator (LPRC)

The LPRC oscillator is separate from the FRC. It oscillates at a nominal frequency of 31.25 kHz. The LPRC oscillator is the clock source for the Power-up Timer (PWRT), Watchdog Timer (WDT), Fail Safe Clock Monitor (FSCM) and PLL reference circuits. It may also be used to provide a low-frequency clock source option for the device in those applications where power consumption is critical, and timing accuracy is not required.

10.2.1.7.1 Enabling the LPRC Oscillator

Since it serves the PWRT clock source, the LPRC oscillator is disabled at Power-on Reset whenever the on-board voltage regulator is enabled. After the PWRT expires, the LPRC oscillator will remain on if any one of the following is true:

- The Fail-Safe Clock Monitor is enabled.
- The WDT is enabled.
- The LPRC oscillator is selected as the system clock (COSC2:COSC0 = `100`).

If none of the above is true, the LPRC will shut off after the PWRT expires.

10.2.2 PERIPHERAL BUS CLOCK (PBCLK) GENERATION

The PBCLK is derived from the System Clock (SYSCLK) divided by PBDIV<1:0> (OSCCON<20:19>). The PBCLK Divisor bits PBDIV<1:0> allow postscalers of 1:1, 1:2, 1:4, and 1:8. Refer to the individual peripheral module section(s) for information regarding which bus a specific peripheral uses.

Notes: When the PBDIV divisor is set to a ratio of '1:1' the SYSCLK and PBCLK are equivalent in frequency. The PBCLK frequency is never greater than the processor clock frequency.

The effect of changing the PBCLK frequency on individual peripherals should be taken into account when selecting or changing the PBDIV value.

Performing back-to-back operations on PBCLK peripheral registers when the PB divisor is not set at 1:1 will cause the CPU to stall for a number of cycles. This stall occurs to prevent an operation from occurring before the pervious one has completed. The length of the stall is determined by the ratio of the CPU and PBCLK and synchronizing time between the two busses.

Changing the PBCLK frequency has no effect on the SYSCLK peripherals operation.

10.2.3 TWO-SPEED START-UP

Two-Speed Start-up mode can be used to reduce the device start-up latency when using all External Crystal POSC modes, including PLL. Two-Speed Start-up uses the FRC clock as the SYSCLK source until the Primary Oscillator (POSC) has stabilized. After the user selected oscillator has stabilized, the clock source will switch to POSC. This allows the CPU to begin running code, at a lower speed, while the oscillator is stabilizing. When the POSC has met the start-up criteria an automatic clock switch occurs to switch to POSC. This mode is enabled by the device Configuration bits FCKSM<1:0> (DEVCFG1<15:14>). Two-Speed Start-up operates after a Power-on Reset (POR) or exit from SLEEP. Software can determine the oscillator source currently in use by reading the COSC<2:0> bits in the OSCCON register.

Note:	The Watchdog Timer (WDT), if enabled, will continue to count at the same rate regardless of the SYSCLK frequency. Care must be taken to service the WDT during Two-Speed Start-up, taking into account the change in SYSCLK.

10.2.4 FAIL-SAFE CLOCK MONITOR OPERATION

The Fail-Safe Clock Monitor (FSCM) is designed to allow continued device operation if the current oscillator fails. It is intended for use with the Primary Oscillator (POSC) and automatically switches to the FRC oscillator if a POSC failure is detected. The switch to the Fast Internal RC Oscillator (FRC) oscillator allows continued device operation and the ability to retry the POSC or to execute code appropriate for a clock failure.

The FSCM mode is controlled by the FCKSM<1:0> bits in the device Configuration register, DEVCFG1. Any of the POSC modes can be used with FSCM.

When a clock failure is detected with FSCM enabled and the FSCM Interrupt Enable bit FSCMIE (IEC1<14>) set, the clock source will be switched from POSC to FRC. An Oscillator Fail interrupt will be generated, with the CF bit (OSCCON<3>) set. This interrupt has a user settable priority FSCMIP<2:0> (IPC8<12:10>) and subpriority FSCMIS<1:0> (IPC8<9:8>). The clock source will remain FRC until a device Reset or a clock switch is performed. Failure to enable the FSCM interrupt will not inhibit the actual clock switch.

The FSCM module takes the following actions when switching to the FRC oscillator:

1. The COSC bits (OSCCON<14:12>) are loaded with '`000`'.
2. The CF OSCCON<3> bit is set to indicate the clock failure
3. The OSWEN control bit (OSCCON<0>) is cleared to cancel any pending clock switches.

To enable FSCM the following steps should be performed:

1. Enable the FSCM in the device Configuration register, DEVCFG1, by configuring the FCKSM<1:0> bits to '`00`'.
   `01` = Clock Switching is enabled, FSCM is disabled
   `00` = Clock Switching and FSCM are enabled
2. Select the desired mode HS, XT, or EC using FNOSC<2:0> in DEVCFG1.
3. Select POSC as the default oscillator in the device Configuration register, DEVCFG1 by configuring FNOSC<2:0> = `010` without PLL or '`011`' with PLL.

If the PLL is to be used:

1. Select the appropriate Configuration bits for the PLL input divider to scale the input frequency to be between 4 MHz and 5 MHz using FPLLIDIV<2:0> (DEVCFG2<2:0>).
2. Select the desired PLL multiplier using FPLLMULT<2:0> (DEVCFG2<6:4>).
3. Select the desired PLL output divider using FPLLODIV<2:0> (DEVCFG2<18:16>).

If a FSCM interrupt is desired when a FSCM event occurs, the following steps should be performed during start-up code:

1. Clear the FSCM interrupt bit FSCMIF (IFS1<14>).
2. Set the Interrupt priority FSCMIP<2:0> (IPC8<12:10>) and subpriority FSCMIS<1:0> (IPC8<9:8>).
3. Set the FSCM Interrupt Enable bit FSCMIE (IEC1<14>)

Note:	The Watchdog Timer, if enabled, will continue to count at the same rate regardless of the SYSCLK frequency. Care must be taken to service the WDT after a Fail-Safe Clock Monitor event, taking into account the change in SYSCLK.

10.2.4.1 FSCM Delay

On a POR, BOR or wake from Sleep mode event, a nominal delay (TFSCM) may be inserted before the FSCM begins to monitor the system clock source. Refer to **Section 8.0 “Resets”** for FSCM delay timing information.

The TFSCM interval is applied whenever the FSCM is enabled and the HS, HSPLL, XT, XTPLL, or SOSC Oscillator modes are selected as the system clock.

> **Note:** Please refer to the Electrical Characteristics section for TFSCM specification values.

10.2.4.2 FSCM and Slow Oscillator Start-up

A slow oscillator start-up will not generate a FSCM event. The FSCM does not begin monitoring until the source to be monitored is running. If the oscillator does not start-up the device will not run due to the lack of a clock source. To detect the failure and prevent this the user should use Two-Speed Start-Up to allow the device to run using the FRC oscillator while the POSC oscillator starts up. The COSC<2:0> bits can then be polled to test for the clock switch to POSC. Refer to **Section 10.2.3 “Two-Speed Start-up”** for further information.

10.2.4.3 FSCM and Slow Clock Sources

Use of the FSCM with slow clock sources (below 100 kHz) is not recommended. Slow clock sources may cause the FSCM to incorrectly detect a clock failure event.

10.2.4.4 FSCM and WDT

The FSCM and the WDT both use the LPRC oscillator as their time base. In the event of a clock failure, the WDT is unaffected and continues to run.

10.2.5 CLOCK SWITCHING OPERATION

With few limitations, applications are free to switch between any of the four clock sources (POSC, SOSC, FRC and LPRC) under software control and at any time. To limit the possible side effects that could result from this flexibility, PIC32MX Family devices have a safeguard lock built into the switch process.

> **Note:** Primary Oscillator mode has three different submodes (XT, HS and EC) which are determined by the POSCMD Configuration bits in DEVCFG1. While an application can switch to and from Primary Oscillator mode in software, it cannot switch between the different primary submodes without reprogramming the device.

> **Note:** The device does not prevent changing the PLL postscaler or multiplier values on the clock source that is in use. The device will not permit direct switching between PLL clock sources. The user should not change the PLL multiplier values or postscaler values when running from the affected PLL source. To perform either of the above clock switching functions, the clock switch should be performed in two steps. The clock source should first be switched to a non-PLL source, such as FRC, and then switched to the desired source. This requirement only applies to PLL-based clock sources.

10.2.5.1 Enabling Clock Switching

To enable clock switching, the FCKSM1 Configuration bit (DEVCFG1<15>) must be programmed to ‘0’. If the FCKSM1 Configuration bit is unprogrammed (= 1), the clock switching function and Fail-Safe Clock Monitor function are disabled. This is the default setting.

The NOSC control bits (OSCCON<10:8>) do not control the clock selection when clock switching is disabled. However, the COSC bits (OSCCON<14:12>) will reflect the clock source selected by the FNOSC Configuration bits.

The OSWEN control bit (OSCCON<0>) has no effect when clock switching is disabled. It is held at ‘0’ at all times.

10.2.5.2 Oscillator Switching Sequence

At a minimum, performing a clock switch requires the following sequence:

1. If desired, read the COSC<2:0> bits (OSCCON<14:12>) to determine the current oscillator source.
2. Perform the unlock sequence to allow a write to the OSCCON register. The unlock sequence has critical timing requirements and should be performed with interrupts and DMA disabled.
3. Write the appropriate value to the NOSC<2:0> control bits (OSCCON<10:8>) for the new oscillator source.
4. Set the OSWEN bit (OSCCON<0>) to initiate the oscillator switch.
5. Optionally perform the lock sequence to lock the OSCCON. The lock sequence must be performed separately from any other operation.

Once the basic sequence is completed, the system clock hardware responds automatically as follows:

1. The clock switching hardware compares the COSC<2:0> Status bits with the new value of the NOSC control bits. If they are the same, then the clock switch is a redundant operation. In this case, the OSWEN bit is cleared automatically

and the clock switch is aborted.

2. The new oscillator is turned on by the hardware if it is not currently running. If a crystal oscillator must be turned on, the hardware will wait until the Oscillator Start-up timer (OST) expires. If the new source is using the PLL, then the hardware waits until a PLL lock is detected (SLOCK = `1`).
3. The hardware clears the OSWEN bit to indicate a successful clock transition. In addition, the NOSC bit values are transferred to the COSC Status bits.
4. The old clock source is turned off at this time if the clock is not being used by any modules.

Note:	The processor will continue to execute code throughout the clock switching sequence. Timing-sensitive code should not be executed during this time.

The following is a recommended code sequence for a clock switch:

1. Disable interrupts and DMA prior to the system unlock sequence.
2. Execute the system unlock sequence by writing the Key values of 0xAA996655 and 0x556699AA to the SYSKEY register in two back-to-back assembly or 'C' instructions.
3. Write the new oscillator source value to the NOSC control bits.
4. Set the OSWEN bit in the OSCCON register to initiate the clock switch.
5. Write a non-key value (such as 0x12345678) to the SYSKEY register to perform a lock. Continue to execute code that is not clock-sensitive (optional).
6. Check to see if OSWEN is '`0`'. If it is, the switch was successful. Loop until the bit is '`0`'.
7. Re-enable interrupts and DMA.

Notes:	There are no timing requirements for the steps other than the initial back-to-back writing of the Key values to perform the unlock sequence. The unlock sequence unlocks all registers that are secured by the lock function. It is recommended that amount to time is the system is unlock is kept to a minimum. The core sequence for unlocking the OSCCON register and initiating a clock switch is shown in Example 10-2.

10.2.5.3 Clock Switching Considerations

When incorporating clock switching into an application, users should keep certain things in mind when designing their code.

- The SYSLOCK unlock sequence is timing critical. The two Key values must be written back-to-back with no in-between peripheral register access. To prevent unintended peripheral register accesses, it is recommended that all interrupts and DMA transfers are disabled.
- The system will not relock automatically. The user should perform the relock sequence as soon after the clock switch as is possible.
- The unlock sequence unlocks other registers such as the those related to Real-Time Clock control.
- If the destination clock source is a crystal oscillator, the clock switch time will be dictated by the oscillator start-up time.
- If the new clock source does not start, or is not present, the OSWEN bit will remain set.
- A clock switch to a different frequency will affect the clocks to peripherals. Peripherals may require reconfiguration to continue operation at the same rate as they did before the clock switch occurred.
- If the new clock source uses the PLL, a clock switch will not occur until lock has been achieved.
- If the WDT is used, care must be taken to ensure it can be serviced in a timely manner at the new clock rate.

Note:	The application should not attempt to switch to a clock with a frequency lower than 100 kHz when the Fail-Safe Clock Monitor is enabled. Clock switching in these instances may generate a false oscillator fail event and result in a switch to the Internal Fast RC oscillator.

Note:	The device does not prevent changing the PLL postscaler or multiplier values on the clock source that is in use. The device will not permit direct switching between PLL clock sources. The user should not change the PLL multiplier values or postscaler values when running from the affected PLL source. To perform either of the above clock switching functions, the clock switch should be performed in two steps. The clock source should first be switched to a non-PLL source, such as FRC, and then switched to the desired source. This requirement only applies to PLL-based clock sources.

EXAMPLE 10-2: PERFORMING A CLOCK SWITCH

```
SYSKEY = 0x12345678;            // write invlid key to force lock
SYSKEY = 0xAA996655;            // Write Key1 to SYSKEY
SYSKEY = 0x556699AA;            // Write Key2 to SYSKEY
                                // OSCCON is now unlocked

                                // make the desired change
                                // This can be in 'C' or assembly
OSCCONCLR = 111 << 16;          // clear the PLL multiplier bits
OSCCONSET = 101 << 16;          // set he new PLL multiplier value
OSCCONSET = 1;                  // request clock switch

                                // Relock the SYSKEY
SYSKEY = 0x12345678;            // Write any value other than Key1 or Key2
                                // OSCCON is relocked
```

10.2.5.4 Entering Sleep Mode During a Clock Switch

If the device enters Sleep mode during a clock switch operation, the clock switch operation is aborted. The processor keeps the old clock selection and the OSWEN bit (OSCCON<0>) is cleared. The `WAIT` instruction is then executed normally.

10.2.5.5 SOSC Control

The SOSC can be used by modules as well as the CPU, therefore, the SOSC is controlled by a combination of software and hardware. Setting the SOSCEN bit (OSCCON<1>) to a '1' enables the SOSC. The SOSC is disabled when it is not being used by the CPU module and the SOSCEN bit is '0'. If the SOSC is being used as SYSCLK, such as after a clock switch, it cannot be disabled by writing to the SOSCEN bit. If the SOSC is enabled by the SOSCEN bit, it will continue to operate when the device is in SLEEP. To prevent inadvertent clock changes the OSCCON register is locked. It must be unlocked prior to software enabling or disabling the SOSC.

10.3 Input/Output Pins

The pins used by the POSC and SOSC are shared by other peripherals modules. Table shows the function of these shared pins in the available oscillator modes. When the pins are not used by a oscillator they are available for use as general I/O pins or by use by a peripheral sharing the pin. .

TABLE 10-5: CONFIGURATION OF PINS ASSOCIATED WITH THE OSCILLATOR MODULE

Pin Name	Clock Mode	Configuration Bit FIeld(1)	TRIS	Pin Type
OSCI	HS, HSPLL, XT, XTPLL	COSC<2:0>, POSCMD<1:0>	X	OSC
OSCO	HS, HSPLL, XT, XTPLL	COSC<2:0>, POSCMD	X	OSC
OSCI	EC, ECPLL	COSC<2:0>, POSCMD	X	CLOCK IN
OSCO	EC, ECPLL	COSC<2:0>, POSCMD, OSCOFNC	X	PBCLK OUT
OSCO	EC, ECPLL	COSC<2:0>, POSCMD, OSCOFNC	INPUT	INPUT
OSCO	EC, ECPLL	COSC<2:0>, POSCMD, OSCOFNC	OUTPUT	OUTPUT
N/A	FRC, FRCPLL, FRCDIV16, FRCDIV, LPRC	COSC<2:0>	X	GPIO
N/A	FRC, FRCPLL, FRCDIV16, FRCDIV, LPRC	COSC<2:0>	X	GPIO
N/A	FRC, FRCPLL, FRCDIV16, FRCDIV, LPRC	COSC<2:0>	X	GPIO
N/A	FRC, FRCPLL, FRCDIV16, FRCDIV, LPRC	COSC<2:0>	X	GPIO
SOSCI	SOSC	COSC<2:0>	X	OSC
SOSCO	SOSC	COSC<2:0>	X	OSC

Note 1: During device start-up, the device oscillator configuration data is copied from device configuration to COSC.

10.3.1 OSCI AND OSCO PIN FUNCTIONS IN NON-EXTERNAL OSCILLATOR MODES

When the primary oscillator (POSC) on OSCI and OSCO is not configured as a clock source the OSCI pin is automatically reconfigured as a digital I/O. In this configuration, as well as when the primary oscillator is configured for EC mode (POSCMD1:POSCMD0 = `00`), the OSCO pin can also be configured as a digital I/O by programming the OSCIOFCN Configuration bit.

When OSCIOFCN is unprogrammed (‘`1`’), a PBCLK is available on OSCO for testing or synchronization purposes. With OSCIOFCN programmed (‘`0`’), the OSCO pin becomes a general purpose I/O pin. In both of these configurations, the feedback device between OSCI and OSCO is turned off to save current.

10.3.2 SOSCI AND SOCI PIN FUNCTIONS IN NON-EXTERNAL OSCILLATOR MODES

When the secondary oscillator (SOSC) on SOSCI and SOSCO pin is not configured as a clock source the pins are automatically reconfigured as a digital I/O.

11.0 POWER SAVING

> **Note:** This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

This section describes power saving for the PIC32MX family. The PIC32MX family devices offer a total of nine methods and modes that are organized into two categories that allow the user to balance power consumption with device performance. In all of the methods and modes described in this section, power saving is controlled by software.

11.1 Power Saving with CPU Running

When the CPU is running, power consumption can be controlled by reducing the CPU clock frequency, lowering the PBCLK, and by individually disabling modules. These methods are grouped into the following categories:

- FRC RUN mode: the CPU is clocked from the FRC clock source with or without postscalers.
- LPRC RUN mode: the CPU is clocked from the LPRC clock source.
- SOSC RUN mode: the CPU is clocked from the SOSC clock source.
- Peripheral Bus Scaling mode:
  Peripherals are clocked at programmable fraction of the CPU clock (SYSCLK).

11.2 CPU Halted Methods

The device supports two power-saving modes, SLEEP and IDLE, both of which halt the clock to the CPU. These modes operate with all clock sources, as listed below:

- POSC IDLE Mode: the system clock is derived from the POSC. The system clock source continues to operate.

  Peripherals continue to operate, but can optionally be individually disabled.
- FRC IDLE Mode: the system clock is derived from the FRC with or without postscalers.

  Peripherals continue to operate, but can optionally be individually disabled.
- SOSC IDLE Mode: the system clock is derived from the SOSC.

  Peripherals continue to operate, but can optionally be individually disabled.
- LPRC IDLE Mode: the system clock is derived from the LPRC.

  Peripherals continue to operate, but can optionally be individually disabled. This is the lowest power mode for the device with a clock running.
- SLEEP Mode: the CPU, the system clock source, and any peripherals that operate from the system clock source, are halted.

  Some peripherals can operate in SLEEP using specific clock sources. This is the lowest power mode for the device.

11.3 Power-Saving Modes Control Registers

Power-Saving modes control consists of the following Special Function Registers (SFRs):

- OSCCON: Control Register for the Oscillators Module

  OSCCONCLR, OSCCONSET, OSCCONINV: Atomic Bit Manipulation Write-only Registers for OSCCON
- WDTCON: Control Register for the Watchdog Timer Module

  WDTCONCLR, WDTCONSET, WDTCONINV: Atomic Bit Manipulation Write-only Registers for WDTCON
- RCON: Control Register for the Resets Module

  RCONCLR, RCONSET, RCONINV: Atomic Bit Manipulation Write-only Registers for RCON

The following table summarizes Power-Saving modes registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

TABLE 11-1: POWER-SAVING MODES SFR SUMMARY

Virtual Address	Name		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
BF80_F000	OSCCON	31:24	—	—	PLLODIV<2:0>			RCDIV<2:0>		
		23:16	—	SOSCRDY	—	PBDIV<1:0>		PLLMULT<2:0>		
		15:8	—	COSC<2:0>			—	NOSC<2:0>		
		7:0	CLKLOCK	—	SLOCK	SLPEN	CF	—	SOSCEN	OSWEN
BF80_F004	OSCCONCLR	31:0	Write clears selected bits in OSCCON, read yields undefined value							
BF80_F008	OSCCONSET	31:0	Write sets selected bits in OSCCON, read yields undefined value							
BF80_F00C	OSCCONINV	31:0	Write inverts selected bits in OSCCON, read yields undefined value							
BF80_0000	WDTCON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	—	—	—	—	—	—	—
		7:0	—	SWDTPS<4:0>					—	WDTCLR
BF80_0004	WDTCONCLR	31:0	Write clears selected bits in WDTCON; read yields undefined value							
BF80_0008	WDTCONSET	31:0	Write sets selected bits in WDTCON; read yields undefined value							
BF80_000C	WDTCONINV	31:0	Write inverts selected bits in WDTCON; read yields undefined value							
BF80_F600	RCON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	CM	VREGS
		7:0	EXTR	SWR	—	WDTO	SLEEP	IDLE	BOR	POR
BF80_F604	RCONCLR	31:0	Write clears selected bits in RCON; read yields undefined value							
BF80_F608	RCONSET	31:0	Write sets selected bits in RCON; read yields undefined value							
BF80_F60C	RCONINV	31:0	Write inverts selected bits in RCON; read yields undefined value							

Advance Information

REGISTER 11-1: OSCCON: OSCILLATOR CONTROL REGISTER

U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	PLLODIV<2:0>			FRCDIV<2:0>		
bit 31							bit 24

R/W-0	R-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
DRMEN	SOSCRDY	—	PBDIV<1:0>		PLLMULT<2:0>		
bit 23							bit 16

U-0	R-0	R-0	R-0	U-0	R/W-x	R/W-x	R/W-x
—	COSC<2:0>			—	NOSC<2:0>		
bit 15							bit 8

R/W-0	U-0	R-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
CLKLOCK	—	SLOCK	SLPEN	CF	—	SOSCEN	OSWEN
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 29-27 **PLLODIV<2:0>:** Output Divider for PLL bits

- `111` = PLL output divided by 256
- `110` = PLL output divided by 64
- `101` = PLL output divided by 32
- `100` = PLL output divided by 16
- `011` = PLL output divided by 8
- `010` = PLL output divided by 4
- `001` = PLL output divided by 2
- `000` = PLL output divided by 1

Note: On Reset these bits are set to the value of the FPLLODIV Configuration bits (DEVCFG2<18:16>)

bit 26-24 **FRCDIV<2:0>:** Fast Internal RC Clock Divider bits

- `111` = FRC divided by 256
- `110` = FRC divided by 64
- `101` = FRC divided by 32
- `100` = FRC divided by 16
- `011` = FRC divided by 8
- `010` = FRC divided by 4
- `001` = FRC divided by 2 (default setting)
- `000` = FRC divided by 1

bit 23 **DRMEN:** Dream Mode Enable bit

- `1` = DMA transfer completion puts device back into SLEEP mode, if the DMA was triggered while device was in SLEEP mode
- `0` = DMA transfer completion has no effect on SLEEP mode

bit 22 **SOSCRDY:** Secondary Oscillator Ready Indicator bit

- `1` = Indicates that the secondary oscillator is running and is stable
- `0` = Secondary oscillator is either turned off or is still warming up

bit 21 **Unimplemented:** Read as ‘`0`’

REGISTER 11-1: OSCCON: OSCILLATOR CONTROL REGISTER (CONTINUED)

bit 20-19	**PBDIV<1:0>:** Peripheral Bus Clock Divisor bits 11 = PBCLK is SYSCLK divided by 8 10 = PBCLK is SYSCLK divided by 4 01 = PBCLK is SYSCLK divided by 2 00 = PBCLK is SYSCLK divided by 1 **Note:** Initial value is loaded from DEVCFG1<13:12>
bit 18-16	**PLLMULT<2:0>:** PLL Multiplier bits 111 = Clock is multiplied by 24 110 = Clock is multiplied by 21 101 = Clock is multiplied by 20 100 = Clock is multiplied by 19 011 = Clock is multiplied by 18 010 = Clock is multiplied by 17 001 = Clock is multiplied by 16 000 = Clock is multiplied by 15 **Note:** On Reset these bits are set to the value of the PLLMULT Configuration bits (DEVCFG2<6:4>)
bit 15	**Unimplemented:** Read as ‘0’
bit 14-12	**COSC<2:0>:** Current Oscillator Selection bits 111 = Fast internal RC oscillator divided by OSCCON.RCDIV 110 = Fast internal RC oscillator divided by 16 101 = Low-Power Internal RC oscillator (LPRC) 100 = Secondary oscillator (SOSC) 011 = Primary oscillator with PLL module (XTPLL, HSPLL or ECPLL) 010 = Primary oscillator (XT, HS or EC) 001 = Fast RC oscillator with PLL module via Postscaler (FRCPLL) 000 = Fast RC oscillator (FRC) **Note:** On Reset these bits are set to the value of the FNOSC Configuration bits (DEVCFG1<2:0>).
bit 11	**Unimplemented:** Read as ‘0’
bit 10-8	**NOSC<2:0>:** New Oscillator Selection bits 111 = Fast internal RC oscillator divided by OSCCON.RCDIV 110 = Fast internal RC oscillator divided by 16 101 = Low-Power Internal RC Oscillator (LPRC) 100 = Secondary oscillator (SOSC) 011 = Primary oscillator with PLL module (XTPLL, HSPLL or ECPLL) 010 = Primary oscillator (XT, HS or EC) 001 = Fast Internal RC oscillator with PLL module via postscaler (FRCPLL) 000 = Fast internal RC oscillator (FRC) On Reset these bits are set to the value of the FNOSC Configuration bits (DEVCFG1<2:0>).
bit 7	**CLKLOCK:** Clock Selection Lock Enable bit If FSCM is enabled (FCKSM1 = 1): 1 = Clock and PLL selections are locked 0 = Clock and PLL selections are not locked and may be modified If FSCM is disabled (FCKSM1 = 0): Clock and PLL selections are never locked and may be modified
bit 6	**Unimplemented:** Read as ‘0’
bit 5	**SLOCK:** PLL Lock Status bit 1 = PLL module is in lock or PLL module start-up timer is satisfied 0 = PLL module is out of lock, PLL start-up timer is running or PLL is disabled
bit 4	**SLPEN:** SLEEP Mode Enable bit 1 = Device will enter SLEEP mode when a WAIT instruction is executed 0 = Device will enter IDLE mode when a WAIT instruction is executed
bit 3	**CF:** Clock Fail Detect bit 1 = FSCM (Fail Safe Clock Monitor) has detected a clock failure 0 = No clock failure has been detected

REGISTER 11-1: OSCCON: OSCILLATOR CONTROL REGISTER (CONTINUED)

bit 2	**Unimplemented:** Read as '0'
bit 1	**SOSCEN:** 32.768 kHz Secondary Oscillator (SOSC) Enable bit 1 = Enable secondary oscillator 0 = Disable secondary oscillator
bit 0	**OSWEN:** Oscillator Switch Enable bit 1 = Initiate an oscillator switch to selection specified by NOSC2:NOSC0 bits 0 = Oscillator switch is complete

REGISTER 11-2: RCON: RESETS CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	CM	VREGS
bit 15							bit 8

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EXTR	SWR	—	WDTO	SLEEP	IDLE	BOR	POR
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 3 **SLEEP:** Wake from Sleep bit

1 = The device woke up from SLEEP mode
0 = The device did not wake from SLEEP mode

Note: Must clear this bit to detect future wake-ups from SLEEP.

bit 2 **IDLE:** Wake from IDLE bit

1 = The device woke up from IDLE mode
0 = The device did not wake from IDLE mode

Note: Must clear this bit to detect future wake-ups from IDLE.

REGISTER 11-3: WDTCON: WATCHDOG TIMER CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
ON	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R-0	R-0	R-0	R-0	R-0	r-0	R/W-0
—	SWDTPS<4:0>					—	WDTCLR
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 15 **ON:** Watchdog Peripheral On bit

1 = Watchdog peripheral is enabled. The status of other bits in the register are not affected by setting this bit. The LPRC oscillator will not be disabled when entering Sleep.

0 = Watchdog peripheral is disabled and not drawing current. SFR modifications are allowed. The status of other bits in this register are not affected by clearing this bit.

11.4 Power-Saving Operation

Note:	In this data sheet, a distinction is made between a power mode as it is used in a specific module, and a power mode as it is used by the device, e.g., Sleep mode of the Comparator and SLEEP mode of the CPU. To indicate which type of power mode is intended, uppercase and lower-case letters (Sleep, Idle, Debug) signify a module power mode, and all uppercase letters (SLEEP, IDLE, DEBUG) signify a device power mode.

The purpose of all power saving is to reduce power consumption by reducing the device clock frequency. To achieve this, low-frequency clock sources can be selected. In addition, the peripherals and CPU can be halted or disabled to further reduce power consumption.

11.5 SLEEP Mode

SLEEP mode has the lowest power consumption of the device Power-Saving operating modes. 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.

SLEEP mode includes the following characteristics:

- The CPU is halted.
- The system clock source is typically shut down. See **Section 11.5.1 "Oscillator Shutdown In Sleep Mode"** for specific information.
- There can be a wake-up delay based on the oscillator selection (refer to Table 11-2).
- The Fail-Safe Clock Monitor (FSCM) does not operate during Sleep mode.
- The BOR circuit, if enabled, remains operative during SLEEP mode.
- The WDT, if enabled, is not automatically cleared prior to entering SLEEP mode.
- Some peripherals can continue to operate in SLEEP mode. These peripherals include I/O pins that detect a change in the input signal, WDT, ADC, UART, and peripherals that use an external clock input or the internal LPRC oscillator, e.g., RTCC and Timer 1.
- I/O pins continue to sink or source current in the same manner as they do when the device is not in SLEEP.
- Some modules can be individually disabled by software prior to entering SLEEP in order to further reduce consumption.

The processor will exit, or 'wake-up', from SLEEP on one of the following events:

- On any interrupt from an enabled source that is operating in Sleep. The interrupt priority must be greater than the current CPU priority.
- On any form of device Reset.
- On a WDT time-out. See **Section 11.10 "Wake-up from SLEEP or IDLE on Watchdog Time-out (NMI)"**.

If the interrupt priority is lower than or equal to current priority, the CPU will remain halted, but the PBCLK will start running and the device will enter into IDLE mode.

Refer Example 11-1 for example code.

Note:	There is no FRZ mode for this module.

11.5.1 OSCILLATOR SHUTDOWN IN SLEEP MODE

The criteria for the device disabling the clock source in SLEEP are: the oscillator type, peripherals using the clock source, and (for select sources) the clock enable bit.

- If the CPU clock source is POSC, it is turned off in SLEEP. See Table 11-2 for applicable delays when waking from SLEEP.
- If the CPU clock source is FRC, it is turned off in SLEEP. See Table 11-2 for applicable delays when waking from SLEEP.
- If the CPU clock source is SOSC, it will be turned off if the SOSCEN bit is not set. See Table 11-2 for applicable delays when waking from SLEEP.
- If the CPU clock source is LPRC, it will be turned off if the clock source is not being used by a peripheral that will be operating in SLEEP, such as the WDT. See Table 11-2 for applicable delays when waking from SLEEP.

11.5.2 CLOCK SELECTION ON WAKE-UP FROM SLEEP

The processor will resume code execution and use the same clock source that was active when SLEEP mode was entered. The device is subject to a start-up delay if a crystal oscillator and/or PLL is used as a clock source when the device exits SLEEP.

11.5.3 DELAY ON WAKE-UP FROM SLEEP

The oscillator start-up and Fail-Safe Clock Monitor delays (if enabled) associated with waking up from SLEEP mode are shown in Table 11-2.

TABLE 11-2: DELAY TIMES FOR EXIT FROM SLEEP MODE

Clock Source	Oscillator Delay	FSCM Delay
EC, EXTRC	—	—
EC + PLL	TLOCK	TFSCM
XT + PLL	TOST + TLOCK	TFSCM
XT, HS, XTL	TOST	TFSCM
LP (OFF during Sleep)	TOST	TFSCM
LP (ON during Sleep)	—	—
FRC, LPRC	—	—

Note: Please refer to the “Electrical Specifications” section of the PIC32MX family device data sheet for TPOR, TFSCM and TLOCK specification values.

11.5.4 WAKE-UP FROM SLEEP MODE WITH CRYSTAL OSCILLATOR OR PLL

If the system clock source is derived from a crystal oscillator and/or the PLL, then the Oscillator Start-up Timer (OST) and/or PLL lock times will be applied before the system clock source is made available to the device. As an exception to this rule, no oscillator delays are applied if the system clock source is the POSC oscillator and it was running while in SLEEP mode.

Note:	In spite of the various delays applied the crystal oscillator (and PLL) may not be up and running at the end of the TOST, or TLOCK delays. For proper operation the user must design the external oscillator circuit such that reliable oscillation will occur within the delay period.

11.5.5 FAIL-SAFE CLOCK MONITOR DELAY AND SLEEP MODE

The Fail-Safe Clock Monitor (FSCM) does not operate while the device is in SLEEP. If the FSCM is enabled it will resume operation when the device wakes from Sleep.

11.5.6 SLOW OSCILLATOR START-UP

When an oscillator starts slowly, the OST and PLL lock times may not have expired before FSCM times out.

If the FSCM is enabled, then the device will detect this condition as a clock failure and a clock event trap will occur. The device will switch to the FRC oscillator and the user can re-enable the crystal oscillator source in the clock failure Interrupt Service Routine.

If the FSCM is not enabled, then the device will simply not start executing code until the clock is stable. From the user's perspective, the device will appear to be in SLEEP until the oscillator clock has started.

EXAMPLE 11-1: PUT DEVICE IN SLEEP, THEN WAKE WITH WDT

```
// Code example to put the Device in sleep and then Wake the device
// with the WDT

OSCCONSET = 0x10;      // set Power-Saving mode to Sleep

WDTCONCLR = 0x0002;    // Disable WDT window mode
WDTCONSET = 0x8000;    // Enable WDT
                       // WDT timeout period is set in the device configuration

while (1)
{
    ... user code ...

    WDTCONSET = 0x01;  // service the WDT
    asm ( "wait" );    // put device in selected Power-Saving mode

                       // code execution will resume here after wake

    ... user code ...
}

// The following code fragment is at the beginning of the 'C' start-up code

if ( RCON & 0x18 )
{
                       // The WDT caused a wake from Sleep
    asm ( "eret" );    // return from interrupt
}
```


11.6 Peripheral Bus Scaling Method

Most of the peripherals on the device are clocked using the PBCLK. The peripheral bus can be scaled relative to the SYSCLK to minimize the dynamic power consumed by the peripherals. The PBCLK divisor is controlled by PBDIV<1:0> (OSCCON<20:19>), allowing SYSCLK-to-PBCLK ratios of 1:1, 1:2, 1:4, and 1:8. All peripherals using PBCLK are affected when the divisor is changed. Peripherals such as the Interrupt Controller, DMA, Bus Matrix, and Prefetch Cache are clocked directly from SYSCLK, as a result, they are not affected by PBCLK divisor changes.

Changing the PBCLK divisor affects:

- The CPU to peripheral access latency. The CPU has to wait for next PBCLK edge for a read to complete. In 1:8 mode this results in a latency of one to seven SYSCLKs.
- The power consumption of the peripherals. Power consumption is directly proportional to the frequency at which the peripherals are clocked. The greater the divisor, the lower the power consumed by the peripherals.

To minimize dynamic power the PB divisor should be chosen to run the peripherals at the lowest frequency that provides acceptable system performance. When selecting a PBCLK divider, peripheral clock requirements such as baud rate accuracy should be taken into account. For example, the UART peripheral may not be able to achieve all baud rate values at some PBCLK divider depending on the SYSCLK value.

11.6.1 DYNAMIC PERIPHERAL BUS SCALING METHOD

The PBCLK can be scaled dynamically, by software, to save additional power when the device is in a low activity mode. The following issues need to be taken into account when scaling the PBCLK:

- All the peripherals clocked from PBCLK will scale at the same ratio, at the same time. This needs to be accounted in peripherals which need to maintain a constant baud rate, or pulse period even in low-power modes.
- Any communication through a peripheral on the peripheral bus that is in progress when the PBCLK changes may cause a data or protocol error due to a frequency change during transmission or reception.

The following steps are recommended if the user intends to scale the PBCLK divisor dynamically:

- Disable all communication peripherals whose baud rate will be affected. Care should be taken to ensure that no communication is currently in progress before disabling the peripherals as it may result in protocol errors.
- Update the Baud Rate Generator (BRG) settings for peripherals as required for operation at the new PBCLK frequency.
- Change the peripheral bus ratio to the desired value.
- Enable all communication peripherals whose baud rate were affected.

> **Note:** Modifying the peripheral baud rate is done by writing to the associated peripheral SFRs. To minimize latency, the peripherals should be modified in the mode where the PBCLK is running at its highest frequency.

EXAMPLE 11-2: CHANGING THE PB CLOCK DIVISOR

```
// Code example to change the PBCLK divisor
// This example is for a device running at 40 MHz

// Make sure that there is no UART send/receive in progress

... user code ...
U1BRG = 0x81;                // set baud rate for UART1 for 9600
... user code ...
OSCCONCLR = 0x3 << 19;       // set PB divisor to minimum (1:1)


... user code ...


// Change Peripheral Clock value

U1BRG = 0x0F;                // set baud rate for UART1 for 9600 based on
                             // new PB clock frequency
OSCCONSET = 0x3 << 19;       // set PB divisor to maximum (1:8)




// Reset Peripheral Clock
OSCCONCLR = 0x3 << 19;       // set PB divisor to minimum (1:1)
U1BRG = 0x81;                // restore baud rate for UART1 to 9600 based
                             // on new PB clock frequency
```

 Advance Information

11.7 IDLE Modes

In the IDLE modes, the CPU is halted but the System clock (SYSCLK) source is still enabled. This allows peripherals to continue operation when the CPU is halted. Peripherals can be individually configured to halt when entering IDLE by setting their respective SIDL bit. Latency when exiting Idle mode is very low due to the CPU oscillator source remaining active.

> **Notes:** Changing the PBCLK divider ratio requires recalculation of peripheral timing. For example, assume the UART is configured for 9600 baud with a PB clock ratio of 1:1 and a POSC of 8 MHz. When the PB clock divisor of 1:2 is used, the input frequency to the baud clock is cut in half; therefore, the baud rate is reduced to 1/2 its former value. Due to numeric truncation in calculations (such as the baud rate divisor), the actual baud rate may be a tiny percentage different than expected. For this reason, any timing calculation required for a peripheral should be performed with the new PB clock frequency instead of scaling the previous value based on a change in PB divisor ratio.
>
> Oscillator start-up and PLL lock delays are applied when switching to a clock source that was disabled and that uses a crystal and/or the PLL. For example, assume the clock source is switched from POSC to LPRC just prior to entering Sleep in order to save power. No oscillator start-up delay would be applied when exiting Idle. However, when switching back to POSC, the appropriate PLL and or oscillator startup/lock delays would be applied.

The device enters IDLE mode when the SLPEN (OSCCON<4>) bit is clear and a `WAIT` instruction is executed.

The processor will wake or exit from IDLE mode on the following events:

- On any interrupt event for which the interrupt source is enabled. The priority of the interrupt event must be greater than the current priority of CPU. If the priority of the interrupt event is lower than or equal to current priority of CPU, the CPU will remain halted and the device will remain in IDLE mode.
- On any source of device Reset.
- On a WDT time-out interrupt. See **Section 11.10 "Wake-up from SLEEP or IDLE on Watchdog Time-out (NMI)"** and **Section 26.0 "Watchdog Timer"**.

TABLE 11-3: PLACING DEVICE IN IDLE AND WAKING BY ADC EVENT

```
// Code example to put the Device in Idle and then Wake the device
// when the ADC completes a conversion

OSCCONCLR = 0x10;                 // set Power-Saving mode to Idle

asm ( "wait" );                   // put device in selected Power-Saving mode

// code execution will resume here after wake and the ISR is complete

    ... user code ...




// interrupt handler


__ADC1Interrupt:
... ISR code ...

asm ( "eret" );                   // return from interrupt
```

11.8 Interrupts

There are two sources of interrupts that will wake the device from a Power-Saving mode: peripheral interrupts, and a Non-Maskable Interrupt (NMI) generated by the WDT in Power-Saving mode.

11.9 Wake-Up from SLEEP or IDLE on Peripheral Interrupt

Any source of interrupt that is individually enabled using the corresponding IE control bit in the IECx register and is operational in the current Power-Saving mode will be able to wake-up the processor from SLEEP or IDLE mode. When the device wakes, one of two events will occur, based on the interrupt priority:

- If the assigned priority for the interrupt is less than, or equal to, the current CPU priority, the CPU will remain halted and the device enters, or remains in, IDLE mode.
- If the assigned priority level for the interrupt source is greater than the current CPU priority, the device will wake-up and the CPU will jump to the corresponding interrupt vector. Upon completion of the ISR, the CPU will start executing the next instruction after `WAIT`.

The IDLE Status bit (RCON<2>) is set upon wake-up from IDLE mode. The SLEEP Status bit (RCON<3>) is set upon wake-up from SLEEP mode.

Notes: A peripheral with an interrupt priority setting of zero cannot wake the device.

Any applicable oscillator start-up delays are applied before the CPU resumes code execution.

11.10 Wake-up from SLEEP or IDLE on Watchdog Time-out (NMI)

When the WDT times out in SLEEP or IDLE mode, an NMI is generated. The NMI causes the CPU code execution to jump to the device Reset vector. Although the CPU executes the Reset vector, it is not a device Reset, peripherals and most CPU registers do not change their states.

> **Note:** Any applicable oscillator start-up delays are applied before the CPU resumes code execution.

To detect a wake from a Power-Saving mode caused by WDT expiration, the WDTO (RCON<4>), SLEEP (RCON<3>), and IDLE (RCON<2>) bits must be tested. If the WDTO bit is '1' the event was due to a WDT time-out. The SLEEP and IDLE bits can then be tested to determine if the WDT event occurred in Sleep or Idle.

To use a WDT time-out during SLEEP mode as a wake-up interrupt, a return from interrupt (`ERET`) instruction must be used in the start-up code after the event was determined to be a WDT wake-up. This will cause code execution to continue from the instruction following the `WAIT` instruction that put the device in Power-Saving mode.

> **Note:** If a peripheral interrupt and WDT event occur simultaneously, or in close proximity, the NMI may not occur, due to the device being awakened by the peripheral interrupt. To avoid unexpected WDT Reset in this scenario, the WDT is automatically cleared when the device awakens.

See **Section 26.0 "Watchdog Timer"** for detailed information on the WDT operation.

11.11 Interrupts Coincident with Power-Saving Instruction

Any peripheral interrupt that coincides with the execution of a `WAIT` instruction will be held off until entry into SLEEP or IDLE mode has completed. The device will then wake-up from SLEEP or IDLE mode.

11.12 I/O Pins Associated with Power-Saving Modes

No device pins are associated with Power-Saving modes.

NOTES:

12.0 I/O PORTS

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

The general purpose I/O pins can be considered the simplest of peripherals. They allow the PIC® MCU to monitor and control other devices. To add flexibility and functionality, some pins are multiplexed with alternate function(s). These functions depend on which peripheral features are on the device. In general, when a peripheral is functioning, that pin may not be used as a general purpose I/O pin.

Following are some of the key features of this module:

- Individual output pin open-drain enable/disable
- Individual input pin weak pull-up enable/disable
- Monitor selective inputs and generate interrupt when change in pin state is detected
- Operation during CPU Sleep and Idle modes
- Fast bit manipulation using CLR, SET and INV registers

Figure 12-1 shows a block diagram of a typical I/O port, whereas Figure 12-2 shows a block diagram of a typical multiplexed I/O port.

FIGURE 12-1: BLOCK DIAGRAM OF A TYPICAL PORT STRUCTURE

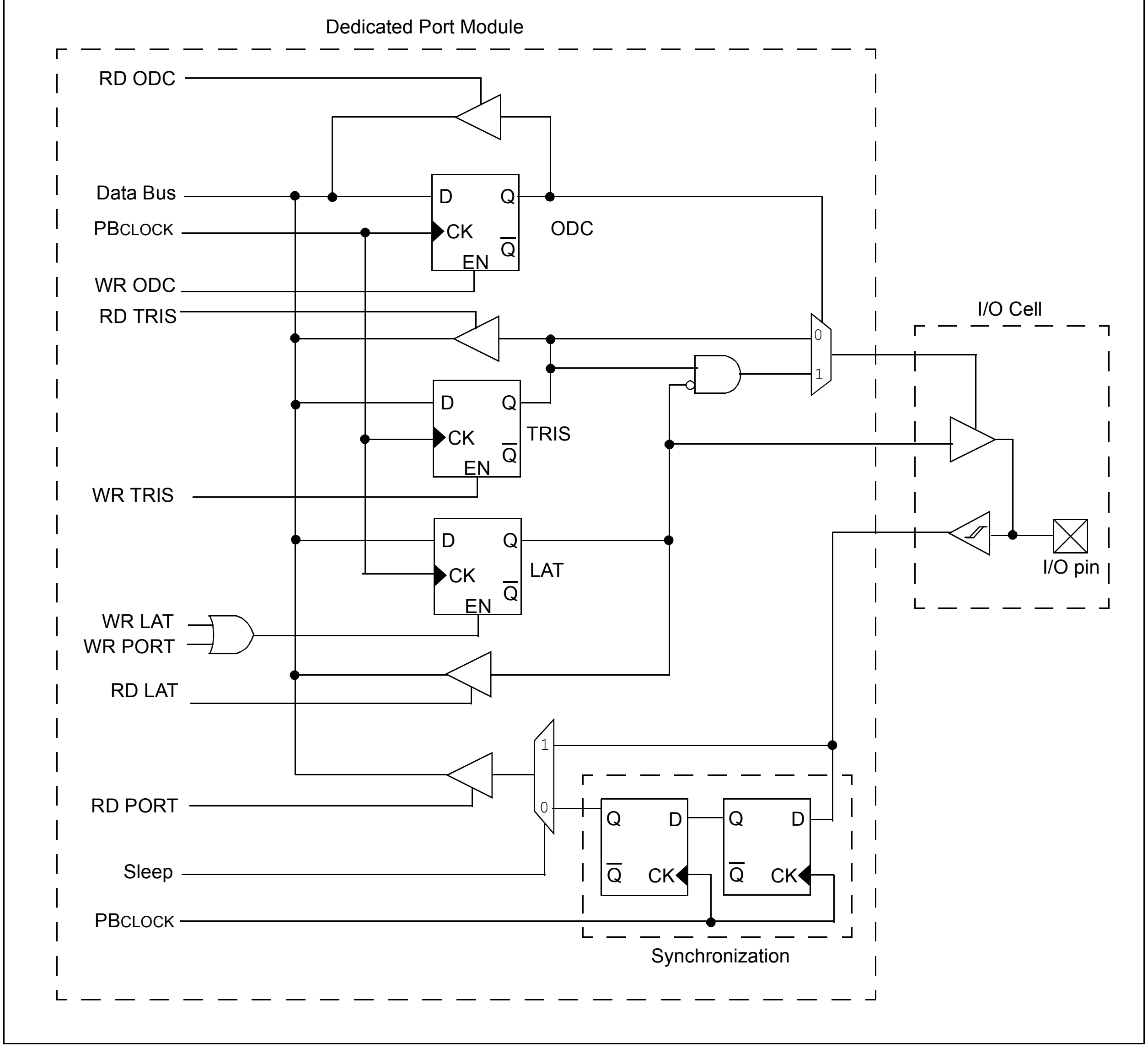
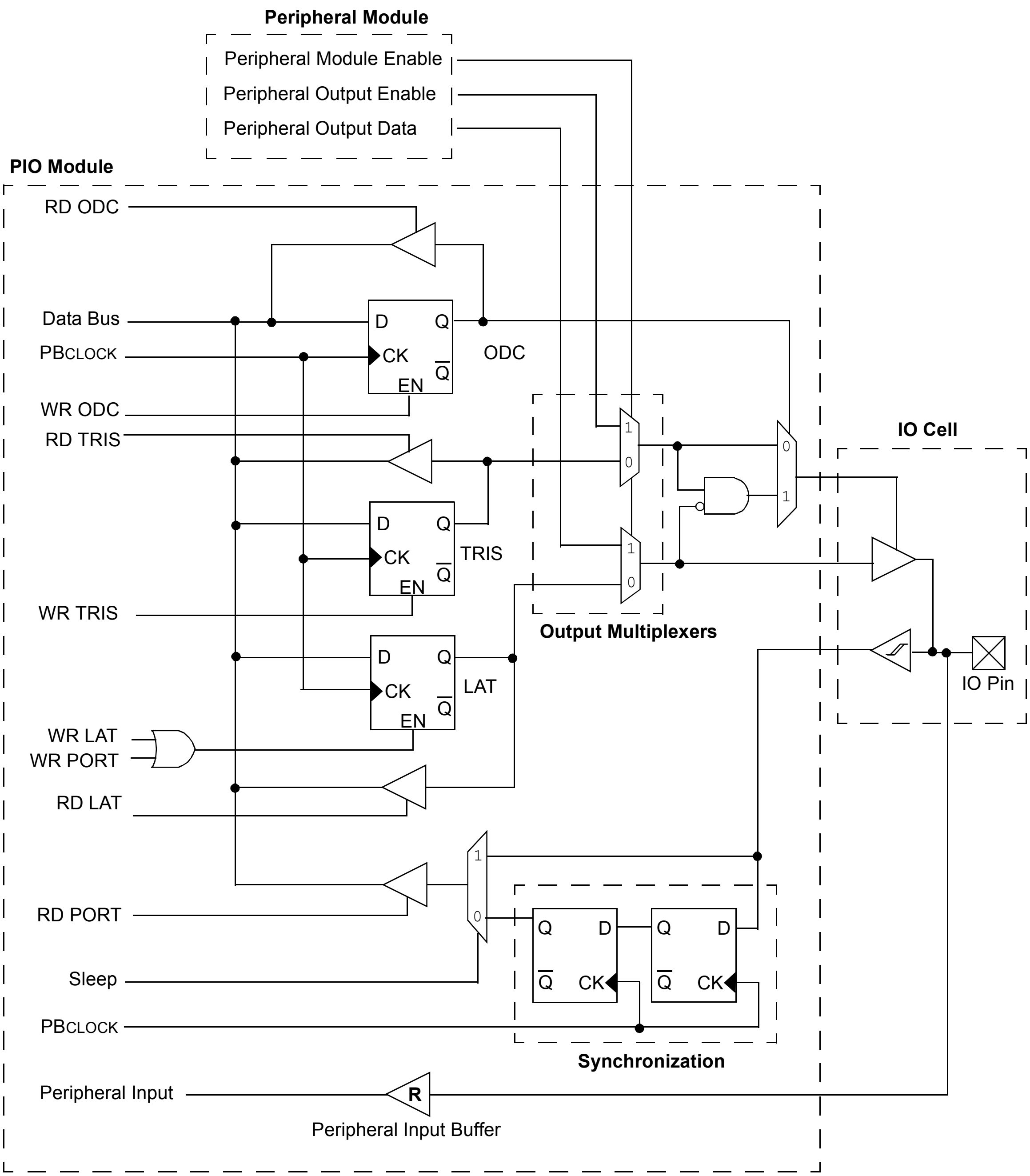


FIGURE 12-2: BLOCK DIAGRAM OF A TYPICAL MULTIPLEXED PORT STRUCTURE



Notes: This block diagram is a general representation of a shared port/peripheral structure for illustration purposes only. The actual structure for any specific port/peripheral combination may be different than what is shown here.

Legend: R = Peripheral input buffer types may vary. Refer to the specific PIC32MX data sheet for peripheral details.

 Advance Information

12.1 Port Registers

TABLE 12-1: PORTA SFR SUMMARY

Virtual Address	Name		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
BF88_6000	TRISA	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	TRISA15	TRISA14	—	—	—	TRISA10	TRISA9	—
		7:0	TRISA<7:0>							
BF88_6004	TRISACLR	31:0	Write clears selected bits in TRISA, read yields undefined value							
BF88_6008	TRISASET	31:0	Write sets selected bits in TRISA, read yields undefined value							
BF88_600C	TRISAINV	31:0	Write inverts selected bits in TRISA, read yields undefined value							
BF88_6010	PORTA	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	RA15	RA14	—	—	—	RA10	RA9	—
		7:0	RA<7:0>							
BF88_6014	PORTACLR	31:0	Write clears selected bits in PORTA, read yields undefined value							
BF88_6018	PORTASET	31:0	Write sets selected bits in PORTA, read yields undefined value							
BF88_601C	PORTAINV	31:0	Write inverts selected bits in PORTA, read yields undefined value							
BF88_6020	LATA	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	LATA15	LATA14	—	—	—	LATA10	LATA9	—
		7:0	LATA<7:0>							
BF88_6024	LATACLR	31:0	Write clears selected bits in LATA, read yields undefined value							
BF88_6028	LATASET	31:0	Write sets selected bits in LATA, read yields undefined value							
BF88_602C	LATAINV	31:0	Write inverts selected bits in LATA, read yields undefined value							
BF88_6030	ODCA	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ODCA15	ODCA14	—	—	—	ODCA10	ODCA9	—
		7:0	ODCA<7:0>							
BF88_6034	ODCACLR	31:0	Write clears selected bits in ODCA, read yields undefined value							
BF88_6038	ODCFASET	31:0	Write sets selected bits in ODCA, read yields undefined value							
BF88_603C	ODCAINV	31:0	Write inverts selected bits in ODCA, read yields undefined value							

Note: TRISA, PORTA, LATA and ODCA registers are not implemented on 64-pin devices, and read as ‘0’.

TABLE 12-2: PORTB SFR SUMMARY

Virtual Address	Name		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
BF88_6040	TRISB	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	TRISB<15:8>							
		7:0	TRISB<7:0>							
BF88_6044	TRISBCLR	31:0	Write clears selected bits in TRISB, read yields undefined value							
BF88_6048	TRISBSET	31:0	Write sets selected bits in TRISB, read yields undefined value							
BF88_604C	TRISBINV	31:0	Write inverts selected bits in TRISB, read yields undefined value							
BF88_6050	PORTB	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	RB<15:8>							
		7:0	RB<7:0>							
BF88_6054	PORTBCLR	31:0	Write clears selected bits in PORTB, read yields undefined value							
BF88_6058	PORTBSET	31:0	Write sets selected bits in PORTB, read yields undefined value							
BF88_605C	PORTBINV	31:0	Write inverts selected bits in PORTB, read yields undefined value							
BF88_6060	LATB	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	LATB<15:8>							
		7:0	LATB<7:0>							
BF88_6064	LATBCLR	31:0	Write clears selected bits in LATB, read yields undefined value							
BF88_6068	LATBSET	31:0	Write sets selected bits in LATB, read yields undefined value							
BF88_606C	LATBINV	31:0	Write inverts selected bits in LATB, read yields undefined value							
BF88_6070	ODCB	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ODCB<15:8>							
		7:0	ODCB<7:0>							
BF88_6074	ODCBCLR	31:0	Write clears selected bits in ODCB, read yields undefined value							
BF88_6078	ODCBSET	31:0	Write sets selected bits in ODCB, read yields undefined value							
BF88_607C	ODCBINV	31:0	Write inverts selected bits in ODCB, read yields undefined value							

TABLE 12-3: PORTC SFR SUMMARY

Virtual Address	Name		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
BF88_6080	TRISC	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	TRISC15	TRISC14	TRISC13	TRISC12	—	—	—	—
		7:0	—	—	—	TRISC4(1)	TRISC3(1)	TRISC2(1)	TRISC1(1)	—
BF88_6084	TRISCCLR	31:0	Write clears selected bits in TRISC, read yields undefined value							
BF88_60088	TRISCSET	31:0	Write sets selected bits in TRISC, read yields undefined value							
BF88_6088C	TRISCINV	31:0	Write inverts selected bits in TRISC, read yields undefined value							
BF88_6090	PORTC	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	RC15	RC14	RC13	RC12	—	—	—	—
		7:0	—	—	—	RC4(1)	RC3(1)	RC2(1)	RC1(1)	—
BF88_6094	PORTCCLR	31:0	Write clears selected bits in PORTC, read yields undefined value							
BF88_6098	PORTCSET	31:0	Write sets selected bits in PORTC, read yields undefined value							
BF88_609C	PORTCINV	31:0	Write inverts selected bits in PORTC, read yields undefined value							
BF88_60A0	LATC	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	LATC15	LATC14	LATC13	LATC12	—	—	—	—
		7:0	—	—	—	LATC4(1)	LATC3(1)	LATC2(1)	LATC1(1)	—
BF88_60A4	LATCCLR	31:0	Write clears selected bits in LATC, read yields undefined value							
BF88_60A8	LATCSET	31:0	Write sets selected bits in LATC, read yields undefined value							
BF88_60AC	LATCINV	31:0	Write inverts selected bits in LATC, read yields undefined value							
BF88_60B0	ODCC	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ODCC15	ODCC14	ODCC13	ODCC12	—	—	—	—
		7:0	—	—	—	ODCC4(1)	ODCC3(1)	ODCC2(1)	ODCC1(1)	—
BF88_60B4	ODCCCLR	31:0	Write clears selected bits in ODCC, read yields undefined value							
BF88_60B8	ODCCSET	31:0	Write sets selected bits in ODCC, read yields undefined value							
BF88_60BC	ODCCINV	31:0	Write inverts selected bits in ODCC, read yields undefined value							

Note 1: TRIS, PORT, LAT and ODC bit(s) are not implemented on 64-pin devices, and read as ‘0’.

TABLE 12-4: PORTD SFR SUMMARY

Virtual Address	Name		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
BF88_60C0	TRISD	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	TRISD15(1)	TRISD14(1)	TRISD13(1)	TRISD12(1)	TRISD<11:8>			
		7:0	TRISD<7:0>							
BF88_60C4	TRISDCLR	31:0	Write clears selected bits in TRISD, read yields undefined value							
BF88_60C8	TRISDSET	31:0	Write sets selected bits in TRISD, read yields undefined value							
BF88_60CC	TRISDINV	31:0	Write inverts selected bits in TRISD, read yields undefined value							
BF88_60D0	PORTD	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	
		15:8	RD15(1)	RD14(1)	RD13(1)	RD12(1)	RD<11:8>			
		7:0	RD<7:0>							
BF88_60D4	PORTDCLR	31:0	Write clears selected bits in PORTD, read yields undefined value							
BF88_60D8	PORTDSET	31:0	Write sets selected bits in PORTD, read yields undefined value							
BF88_60DC	PORTDINV	31:0	Write inverts selected bits in PORTD, read yields undefined value							
BF88_60E0	LATD	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	LAT15(1)	LAT14(1)	LAT13(1)	LAT12(1)	LATD<11:8>			
		7:0	LATD<7:0>							
BF88_60E4	LATDCLR	31:0	Write clears selected bits in LATD, read yields undefined value							
BF88_60E8	LATDSET	31:0	Write sets selected bits in LATD, read yields undefined value							
BF88_60EC	LATDINV	31:0	Write inverts selected bits in LATD, read yields undefined value							
BF88_60F0	ODCD	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ODCD15(1)	ODCD14(1)	ODCD13(1)	ODCD12(1)	ODCD<11:8>			
		7:0	ODCD<7:0>							
BF88_60F4	ODCDCLR	31:0	Write clears selected bits in ODCD, read yields undefined value							
BF88_60F8	ODCDSET	31:0	Write sets selected bits in ODCD, read yields undefined value							
BF88_60FC	ODCDINV	31:0	Write inverts selected bits in ODCD, read yields undefined value							

Note 1: TRIS, PORT, LAT and ODC bit(s) are not implemented on 64-pin devices, and read as '0'.

TABLE 12-5: PORTE SFR SUMMARY

Virtual Address	Name		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
BF88_6100	TRISE	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	TRISE9(1)	TRISE8(1)
		7:0	TRISE<7:0>							
BF88_6104	TRISECLR	31:0	Write clears selected bits in TRISE, read yields undefined value							
BF88_6108	TRISESET	31:0	Write sets selected bits in TRISE, read yields undefined value							
BF88_610C	TRISEINV	31:0	Write inverts selected bits in TRISE, read yields undefined value							
BF88_6110	PORTE	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	RE9(1)	RE8(1)
		7:0	RE<7:0>							
BF88_6114	PORTECLR	31:0	Write clears selected bits in PORTE, read yields undefined value							
BF88_6118	PORTESET	31:0	Write sets selected bits in PORTE, read yields undefined value							
BF88_611C	PORTEINV	31:0	Write inverts selected bits in PORTE, read yields undefined value							
BF88_6120	LATE	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	LATE9(1)	LATE8(1)
		7:0	LATE<7:0>							
BF88_6124	LATECLR	31:0	Write clears selected bits in LATE, read yields undefined value							
BF88_6128	LATESET	31:0	Write sets selected bits in LATE, read yields undefined value							
BF88_612C	LATEINV	31:0	Write inverts selected bits in LATE, read yields undefined value							
BF88_6130	ODCE	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	ODCE9(1)	ODCE8(1)
		7:0	ODCE<7:0>							
BF88_6134	ODCECLR	31:0	Write clears selected bits in ODCE, read yields undefined value							
BF88_6138	ODCESET	31:0	Write sets selected bits in ODCE, read yields undefined value							
BF88_613C	ODCEINV	31:0	Write inverts selected bits in ODCE, read yields undefined value							

Note 1: TRIS, PORT, LAT and ODC bit(s) are not implemented on 64-pin devices, and read as '0'.

TABLE 12-6: PORTF SFR SUMMARY

Virtual Address	Name		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
BF88_6140	TRISF	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	TRISF13(1)	TRISF12(1)	—	—	—	TRISF8(1)
		7:0	TRISF7(1)	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0
BF88_6144	TRISFCLR	31:0	Write clears selected bits in TRISF, read yields undefined value							
BF88_6148	TRISFSET	31:0	Write sets selected bits in TRISF, read yields undefined value							
BF88_614C	TRISFINV	31:0	Write inverts selected bits in TRISF, read yields undefined value							
BF88_6150	PORTF	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	RF13(1)	RF12(1)	—	—	—	RF8(1)
		7:0	RF7(1)	RF6	RF5	RF4	RF3	RF2	RF1	RF0
BF88_6154	PORTFCLR	31:0	Write clears selected bits in PORTF, read yields undefined value							
BF88_6158	PORTFSET	31:0	Write sets selected bits in PORTF, read yields undefined value							
BF88_615C	PORTFINV	31:0	Write inverts selected bits in PORTF, read yields undefined value							
BF88_6160	LATF	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	LATF13(1)	LATF12(1)	—	—	—	LATF8(1)
		7:0	LATF7(1)	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0
BF88_6164	LATFCLR	31:0	Write clears selected bits in LATF, read yields undefined value							
BF88_6168	LATFSET	31:0	Write sets selected bits in LATF, read yields undefined value							
BF88_616C	LATFINV	31:0	Write inverts selected bits in LATF, read yields undefined value							
BF88_6170	ODCF	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	ODCF13(1)	ODCF12(1)	—	—	—	ODCF8(1)
		7:0	ODCF7(1)	ODCF6	ODCF5	ODCF4	ODCF3	ODCF2	ODCF1	ODCF0
BF88_6174	ODCFCLR	31:0	Write clears selected bits in ODCF, read yields undefined value							
BF88_6178	ODCFSET	31:0	Write sets selected bits in ODCF, read yields undefined value							
BF88_617C	ODCFINV	31:0	Write inverts selected bits in ODCF, read yields undefined value							

Note 1: TRIS, PORT, LAT and ODC bit(s) are not implemented on 64-pin devices, and read as '0'.

TABLE 12-7: PORTG SFR SUMMARY

Virtual Address	Name		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
BF88_6180	TRISG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	TRISG15(1)	TRISG14(1)	TRISG13(1)	TRISG12(1)	—	—	TRISG9	TRISG8
		7:0	TRISG7	TRISG6	—	—	TRISG3	TRISG2	TRISG1(1)	TRISG0(1)
BF88_6184	TRISGCLR	31:0	Write clears selected bits in TRISG, read yields undefined value							
BF88_6188	TRISGSET	31:0	Write sets selected bits in TRISG, read yields undefined value							
BF88_618C	TRISGINV	31:0	Write inverts selected bits in TRISG, read yields undefined value							
BF88_6190	PORTG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	RG15(1)	RG14(1)	RG13(1)	RG12(1)	—	—	RG9	RG8
		7:0	RG7	RG6	—	—	RG3	RG2	RG1(1)	RG0(1)
BF88_6194	PORTGCLR	31:0	Write clears selected bits in PORTG, read yields undefined value							
BF88_6198	PORTGSET	31:0	Write sets selected bits in PORTG, read yields undefined value							
BF88_619C	PORTGINV	31:0	Write inverts selected bits in PORTG, read yields undefined value							
BF88_61A0	LATG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	LATG15(1)	LATG14(1)	LATG13(1)	LATG12(1)	—	—	LATG9	LATG8
		7:0	LATG7	LATG6	—	—	LATG3	LATG2	LATG1(1)	LATG0(1)
BF88_61A4	LATGCLR	31:0	Write clears selected bits in LATG, read yields undefined value							
BF88_61A8	LATGSET	31:0	Write sets selected bits in LATG, read yields undefined value							
BF88_61AC	LATGINV	31:0	Write inverts selected bits in LATG, read yields undefined value							
BF88_61B0	ODCG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ODCG15(1)	ODCG14(1)	ODCG13(1)	ODCG12(1)	—	—	ODCG9	ODCG8
		7:0	ODCG7	ODCG6	—	—	ODCG3	ODCG2	ODCG1(1)	ODCG0(1)
BF88_61B4	ODCGCLR	31:0	Write clears selected bits in ODCG, read yields undefined value							
BF88_61B8	ODCGSET	31:0	Write sets selected bits in ODCG, read yields undefined value							
BF88_61BC	ODCGINV	31:0	Write inverts selected bits in ODCG, read yields undefined value							

Note 1: TRIS, PORT, LAT and ODC bit(s) are not implemented on 64-pin devices, and read as ‘0’.

TABLE 12-8: CHANGE NOTICE AND PULL UP SFR SUMMARY

Virtual Address	Name		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
BF88_61C0	CNCON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	—	—
		7:0	—	—	—	—	—	—	—	—
BF88_61C4	CNCONCLR	31:0	Write clears selected bits in CNCON, read yields undefined value							
BF88_61C8	CNCONSET	31:0	Write sets selected bits in CNCON, read yields undefined value							
BF88_61CC	CNCONINV	31:0	Write inverts selected bits in CNCON, read yields undefined value							
BF88_61D0	CNEN	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	CNEN21(1)	CNEN20(1)	CNEN19(1)	CNEN18	CNEN17	CNEN16
		15:8	CNEN<15:8>							
		7:0	CNEN<7:0>							
BF88_61D4	CNENCLR	31:0	Write clears selected bits in CNEN, read yields undefined value							
BF88_61D8	CNENSET	31:0	Write sets selected bits in CNEN, read yields undefined value							
BF88_61DC	CNENINV	31:0	Write inverts selected bits in CNEN, read yields undefined value							
BF88_61E0	CNPUE	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	CNPUE21(1)	CNPUE20(1)	CNPUE9(1)	CNPUE18	CNPUE17	CNPUE16
		15:8	CNPUE<15:8>							
		7:0	CNPUE<7:0>							
BF88_61E4	CNPUECLR	31:0	Write clears selected bits in CNPUE, read yields undefined value							
BF88_61E8	CNPUESET	31:0	Write sets selected bits in CNPUE, read yields undefined value							
BF88_61EC	CNPUEINV	31:0	Write inverts selected bits in CNPUE, read yields undefined value							

Note 1: CNEN and CNPUE bit(s) are not implemented on 64-pin devices, and read as '0'.

TABLE 12-9: CHANGE NOTICE INTERRUPT REGISTER SUMMARY

Virtual Address	Name		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
BF88_1050	IEC1	7:0	SPI2RXIE	SPI2TXIE	SPI2EIE	CMP2IE	CMP1IE	PMPIE	AD1IE	CNIE
BF88_1020	IFS1	7:0	SPI2RXIF	SPI2TXIF	SPI2EIF	CMP2IF	CMP1IF	PMPIF	AD1IF	CNIF
BF88_10D0	IPC6	23:16	—	—	—	CNIP<2:0>			CNIS<1:0>	

Note: This summary table contains partial register definitions that only pertain to the GPIO peripheral. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of these registers.

REGISTER 12-1: TRISx: TRIS REGISTERS(1)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISx<15:8>							
bit 15							bit 8

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISx<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **TRISx<15:0>:** TRISx Register bits
1 = Corresponding port pin ‘Input’
0 = Corresponding port pin ‘Output’

Note 1: Depending on the device, certain register bits or the entire register may not be implemented. Refer to Table 12.1 for specific register and bit assignments.

REGISTER 12-2: PORTx: PORT REGISTERS(1)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
Rx<15:8>							
bit 15							bit 8

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
Rx<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **PORTx<15:0>:** PORTx Register bits
Read = Value on port pins
Write = Value written to the LATx register, port latch and I/O pins

Note 1: Depending on the device family variant, certain register bits or the entire register may not be implemented. Refer to Table 12.1 for specific register and bit assignments.

REGISTER 12-3: LATx: LAT REGISTERS[1]

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
LATx<15:8>							
bit 15							bit 8

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
LATx<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **LATx<15:0>:** LATx Register bits
Read = Value on port latch, not I/O pins
Write = Value written to port latch and I/O pins

Note 1: Depending on the device, certain register bits or the entire register may not be implemented. Refer to Table 12.1 for specific register and bit assignments.

REGISTER 12-4: ODCx: OPEN DRAIN CONFIGURATION REGISTERS(1)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ODCx<15:8>							
bit 15							bit 8

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ODCx<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable	r = Reserved bit
U = Unimplemented bit, read as ‘0’		-n = Bit value at POR: (‘0’, ‘1’, x = unknown)	

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **ODCx<15:0>:** ODCx Register bits

If a port pin is configured as an output (corresponding TRISx bit = 0).
1 = Port pin open-drain output enabled
0 = Port pin open-drain output disabled
If a port pin is configured as an input, ODCx bits have no effect.

Note 1: Depending on the device, certain register bits or the entire register may not be implemented. Refer to Table 12.1 for specific register and bit assignments.

 Advance Information

REGISTER 12-5: CNCON: CHANGE NOTICE CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
ON	FRZ	SIDL	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** Change Notice Module On bit
1 = CN module is enabled
0 = CN module is disabled

bit 14 **FRZ:** Freeze in Debug Exception Mode bit
1 = Freeze operation when CPU is in Debug Exception mode
0 = Continue operation when CPU is in Debug Exception mode

bit 13 **SIDL:** Stop in Idle Mode bit
1 = Discontinue operation when device enters Idle mode
0 = Continue operation in Idle mode

bit 12-0 **Unimplemented:** Read as ‘0’

REGISTER 12-6: CNEN: INPUT CHANGE NOTIFICATION INTERRUPT ENABLE REGISTER(1)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CNEN21	CNEN20	CNEN19	CNEN18	CNEN17	CNEN16
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNEN15	CNEN14	CNEN13	CNEN12	CNEN11	CNEN10	CNEN9	CNEN8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNEN7	CNEN6	CNEN5	CNEN4	CNEN3	CNEN2	CNEN1	CNEN0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-22 **Unimplemented:** Read as ‘0’

bit 21-0 **CNEN<21:0>:** CNEN Register
If a port pin is configured as an input (corresponding TRISx bit = 1)
1 = Port pin input change notice enabled
0 = Port pin input change notice disabled
If a port pin is configured as an output, CNENx bits have no effect

Note 1: Depending on the device, certain register bits or the entire register may not be implemented. Refer to Table 12.1 for specific register and bit assignments.

REGISTER 12-7: CNPUE: INPUT CHANGE NOTIFICATION PULL-UP ENABLE(1)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CNPUE21	CNPUE20	CNPUE19	CNPUE18	CNPUE17	CNPUE16
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNPUE15	CNPUE14	CNPUE13	CNPUE12	CNPUE11	CNPUE10	CNPUE9	CNPUE8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNPUE7	CNPUE6	CNPUE5	CNPUE4	CNPUE3	CNPUE2	CNPUE1	CNPUE0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-22 **Unimplemented:** Read as ‘0’

bit 21-0 **CNPUE<21:0>:** CNPUE Register bits
If a port pin is configured as an input (corresponding TRISx bit = 1).
1 = Port pin pull-up enabled
0 = Port pin pull-up disabled
If a port pin is configured as an output, it is recommended to disable the corresponding CNPUEx bit.

Note 1: Depending on the device, certain register bits or the entire register may not be implemented. Refer to Table 12.1 for specific register and bit assignments.

12.2 Parallel I/O (PIO) Ports

All port pins have three registers (TRIS, LAT, and PORT) that are directly associated with their operation.

TRIS is a data direction or tri-state control register that determines whether a pin is an input or an output. Setting a TRISx register bit = `1` configures the corresponding I/O pin as an input; setting a TRISx register bit = `0` configures the corresponding I/O pin as an output. All port I/O pins are defined as inputs after a Power-On Reset (POR).

PORT is a register used to read the current state of the signal applied to the port I/O pins. Writing to a PORTx register performs a write to the port's latch, LATx register, latching the data to the port's I/O pins.

LAT is a register used to write data to the port I/O pins. The LATx latch register holds the data written to either the LATx or PORTx registers. Reading the LATx latch register reads the last value written to the corresponding port or latch register.

Not all port I/O pins are implemented on some devices, therefore, the corresponding PORTx, LATx and TRISx register bits will read as zeros. See **Section 12.1 "Port Registers"**.

12.2.1 CLR, SET AND INV REGISTERS

Every I/O module register has a corresponding CLR (clear), SET (set) and INV (invert) register designed to provide fast atomic bit manipulations. As the name of the register implies, a value written to a SET, CLR or INV register effectively performs the implied operation, but only on the corresponding base register and only bits specified as '`1`' are modified. Bits specified as '`0`' are not modified.

Reading SET, CLR and INV registers returns undefined values. To see the affects of a write operation to a SET, CLR or INV register, the base register must be read instead.

To set PORTC bit 0, use the LATSET register as follows:

```
LATCSET = 0x0001;
```

To clear PORTC bit 0, use the LATCLR register as follows:

```
LATCCLR = 0x0001;
```

To toggle PORTC bit 0, use the LATINV register as follows:

```
LATCINV = 0x0001;
```

Note:	Using a PORTxINV register to toggle a bit is recommended because the operation is performed in hardware atomically, using fewer instructions as compared to the traditional read-modify-write method shown below: PORTC ^= 0x0001;

12.2.2 DIGITAL INPUTS

Pins are configured as digital inputs by setting the corresponding TRIS register bits = `1`. When configured as inputs, they are either TTL buffers or Schmitt Triggers. Several digital pins share functionality with analog inputs and default to the analog inputs at POR. Setting the corresponding bit in the ADP1CFG register = `1` enables the pin as a digital pin.

Digital only pins are capable of input voltages up to 5.5v. Any pin that shares digital and analog functionality is limited to voltages up to V_{DD} + 0.3V.

TABLE 12-10: MAXIMUM INPUT PIN VOLTAGES

Input Pin Mode(s)	V_{IH} (max)
Digital Only	V_{IH} = 5.5v
Digital + Analog	V_{IH} = V_{DD} + 0.03v
Analog	V_{IH} = V_{DD} + 0.03v

Note: Refer to **Section 29.0 "Electrical Characteristics"** regarding the V_{IH} specification.

Note:	Pins configured as digital inputs will not convert an analog input. Analog levels on any pin that is defined as a digital input (including the ANx pins) may cause the input buffer to consume current that exceeds the device specifications.

12.2.3 ANALOG INPUTS

Certain pins can be configured as analog inputs used by the ADC and Comparator modules. Setting the corresponding bits in the ADP1CFG register = `0` enables the pin as an analog input pin and must have the corresponding TRIS bit set = `1` (input). If the TRIS bit is cleared = `0` (output), the digital output level (V_{OH} or V_{OL}) will be converted. Any time a port I/O pin is configured as analog, its digital input is disabled and the corresponding PORTx register bit will read '`0`'.

12.2.4 DIGITAL OUTPUTS

Pins are configured as digital outputs by setting the corresponding TRIS register bits = `0`. When configured as digital outputs, these pins are CMOS drivers or can be configured as open drain outputs by setting the corresponding bits in the ODCx Open-Drain Configuration register.

Digital output pin voltage is limited to V_{DD}.

12.2.5 ANALOG OUTPUTS

Certain pins can be configured as analog outputs, such as the CV_{REF} output voltage used in the comparator module. Configuring the Comparator module to provide this output will present the analog output voltage on the pin independent of the TRIS register setting for the corresponding pin.

12.2.6 OPEN-DRAIN CONFIGURATION

In addition to the PORT, LAT and TRIS registers for data control, each port pin configured as a digital output can also select between an active drive output and open-drain output. This is controlled by the Open-Drain Control register, ODCx, associated with each port. From POR, when an IO pin is configured as a digital output, its output is active drive by default. Setting a bit in the ODCx register = `1` configures the corresponding pin as an open-drain output.

The open-drain feature allows the generation of outputs higher than V_{DD}, e.g., 5V, on any desired digital-only pins by using external pull-up resistors. The maximum open-drain voltage allowed is the same as the maximum V_{IH} specification, typically 5.5v.

12.2.7 PERIPHERAL MULTIPLEXING

A parallel I/O port that shares a pin with a peripheral is, in general, subservient to the peripheral. The peripheral's output buffer data and control signals are provided to a pair of multiplexers. The multiplexers select whether the peripheral or the associated port has ownership of the output data and control signals of the I/O pin. The logic also prevents "loop through", in which a port's digital output can drive the input of a peripheral that shares the same pin. Figure 12-2 shows how ports are shared with other peripherals and the associated I/O pin to which they are connected.

When a peripheral is enabled and actively driving an associated pin, the use of the pin as a general purpose output pin is disabled. The I/O pin may be read, but the output driver for the parallel port bit will be disabled. If a peripheral is enabled, but the peripheral is not actively driving a pin, that pin may be driven by a port.

12.2.8 SOFTWARE INPUT PIN CONTROL

Some peripheral inputs assigned to an I/O pin may not take control of the I/O pin output driver. If the I/O pin associated with the peripheral is configured as an output, using the appropriate TRIS control bit, the user can manually affect the state of the peripheral's input pin through its corresponding LAT register. This behavior can be useful in some situations, especially for testing purposes, when no external signal is connected to the input pin.

In general, the following peripherals allow their input pins to be controlled manually through the LAT registers:

- External Interrupt pins
- Timer Clock Input pins
- Input Capture pins
- PWM Fault pins

Most serial communication peripherals, when enabled, take full control of the I/O pin so that the input pins associated with the peripheral cannot be affected through the corresponding PORT registers. These peripherals include the following modules:

- SPI
- I^2C™
- UART

12.2.9 INPUT CHANGE NOTIFICATION

Certain PIC32MX I/O port pins provide Input Change notification that can generate interrupt requests to the processor in response to a Change-Of-State (COS) on those selected input pins. The initial state of any enabled Change Notice (CN) pin must be established by reading the corresponding PORT register. This feature is capable of detecting input COS even in Sleep mode, when the clocks are disabled. Depending on the device pin count, there are up to 22 external signals (CN0 through CN21) that may be selected (enabled) for generating an interrupt request on a COS.

The following control registers are associated with the change notice module:

- CNCON
- CNEN
- CNPUE

The CNCON control register ON bit enables or disables the CN module and its ability to generate interrupts or respond to mismatch conditions.

The CNEN (change notice enable) register control bits enable each CN input. Setting any of these bits enables a CN for the corresponding pins.

The CNPUE (change notice pull-up enable) register control bits enable a weak pull-up to a corresponding CN input pin. The pull-ups act as a current source that is connected to the pin, and eliminate the need for external resistors when push button or keypad devices are connected.

Note:	Pull-up resistors on change notification pins should always be disabled whenever the port pin is configured as a digital output.

TABLE 12-11: CHANGE NOTICE PIN AND PULL-UP TABLE

Change Notice	Weak Pull-Up	Port Pin	64-Pin Device	100-Pin Device
			Pin#	
CN0	CNPUE0	RC14	48	74
CN1	CNPUE1	RC13	47	73
CN2	CNPUE2	RB0	16	25
CN3	CNPUE3	RB1	15	24
CN4	CNPUE4	RB2	14	23
CN5	CNPUE5	RB3	13	22
CN6	CNPUE6	RB4	12	21
CN7	CNPUE7	RB5	11	20
CN8	CNPUE8	RG6	4	10
CN9	CNPUE9	RG7	5	11
CN10	CNPUE10	RG8	6	12
CN11	CNPUE11	RG9	8	14
CN12	CNPUE12	RB15	30	44
CN13	CNPUE13	RD4	52	81
CN14	CNPUE14	RD5	53	82
CN15	CNPUE15	RD6	54	83
CN16	CNPUE16	RD7	55	84
CN17	CNPUE17	RF4	31	49
CN18	CNPUE18	RF5	32	50
CN19	CNPUE19	RD13	—	80
CN20	CNPUE20	RD14	—	47
CN21	CNPUE21	RD15	—	48

12.2.10 CHANGE NOTICE INTERRUPTS

The Change Notice module is enabled as a source of interrupts via the respective CN interrupt enable bits:

- CNIE (IEC1<0>)
- CNIF (IFS1<0>)

The interrupt priority level bits and interrupt subpriority level bits must also be configured:

- CNIP<2:0> (IPC6<20:18>)
- CNIS<1:0> (IPC6<17:16>)

To enable CN interrupts, the ON bit (CNCON<15>) must = `1`, one or more CN input pins must be enabled and the Change Notice Interrupt Enable bit, CNIE, must = `1`.

To prevent possible spurious interrupts when configuring change notice interrupts, the following steps are recommended:

1. Disable CPU interrupts.
2. Set desired CN I/O pin as input by setting corresponding TRISx register bits = `1`.
   Note: If the I/O pin is shared with an analog peripheral, it may be necessary to set the corresponding AD1PCFG bit = `1` to ensure that the I/O pin is a digital input.
3. Enable change notice module ON (CNCON<15>) = `1`.
4. Enable individual CN input pin(s); enable optional pull-up(s).
5. Read corresponding PORT registers to clear mismatch condition on CN input pins.
6. Configure the CN interrupt priority, CNIP<2:0>, and subpriority CNIS<1:0>.
7. Clear CN interrupt flag, CNIF = `0`.
8. Enable CN interrupt enable, CNIE = `1`.
9. Enable CPU interrupts.

The port must be read to clear the mismatch condition and, then CN interrupt flag, CNIF (IFS1<0>), can be cleared in software. Failing to read the port before attempting to clear the CNIF bit may not allow the CNIF bit to be cleared.

In addition to enabling the CN interrupt, an Interrupt Service Routine (ISR), is required. Example 12-1 and Example 12-2 show a partial code example of an ISR.

Note:	It is the user's responsibility to clear the corresponding interrupt flag bit before returning from an ISR.

EXAMPLE 12-1: CN CONFIGURATION AND INTERRUPT INITIALIZATION EXAMPLE CODE

```
/*
    The following code example illustrates a Change Notice
    interrupt configuration for pins CN1(PORTC), CN4(PORTB) and CN18(PORTF).
*/
    unsigned int value;

    /* NOTE: disable vector interrupts prior to configuration */

    CNCON = 0x8000;         // Enable Change Notice module
    CNEN=   0x00040012;     // Enable CN1, CN4 and CN18 pins
    CNPUE=  0x00040012;     // Enable weak pull ups for CN1, CN4 and CN18 pins

    /* read port(s) to clear mismatch on change notice pins */
    value = PORTB;
    value = PORTC;
    value = PORTF;

    IPS6SET = 0x00140000;  // Set priority level=5
    IPS6SET = 0x00030000;  // Set subpriority level=3
                           // Could have also done this in single
                           // operation by assigning IPS6SET = 0x00170000

    IFS1CLR  = 0x0001;     // Clear the interrupt flag status bit
    IEC1SET  = 0x0001;     // Enable Change Notice interrupts

    /* re-enable vector interrupts after configuration */
```

EXAMPLE 12-2: CN ISR EXAMPLE CODE

```
/*
    The following code example demonstrates a simple interrupt service
    routine for CN interrupts. The user’s code at this vector should perform
    any application specific operations and must read the CN corresponding
    PORT registers to clear the mismatch conditions.
    Finally, the CN interrupt status flag must be cleared before exiting.
*/
void __ISR(_CHANGE_NOTICE_VECTOR, ipl3) CN_Interrupt_ISR(void)
{
    unsigned int value;

    value = PORTB          // Read PORTB to clear CN4 mismatch condition
    value = PORTC          // Read PORTC to clear CN1,CN0 mismatch condition

    ... perform application specific operations in response to the interrupt

    IFS1CLR  = 0x0001;     // Be sure to clear the CN interrupt status
                           // flag before exiting the service routine.
}
```

Note:	The CN ISR code example shows MPLAB® C32 C compiler-specific syntax. Refer to your compiler manual regarding support for ISRs.

NOTES:

13.0 TIMER1

> **Note:** This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

This family of PIC32MX devices features one synchronous/asynchronous 16-bit timer that can operate as a free-running interval timer for various timing applications and counting external events. This timer can also be used with the Low-Power Secondary Oscillator, SOSC, for real-time clock applications. The following modes are supported:

- Synchronous Internal Timer
- Synchronous Internal Gated Timer
- Synchronous External Timer
- Asynchronous External Timer

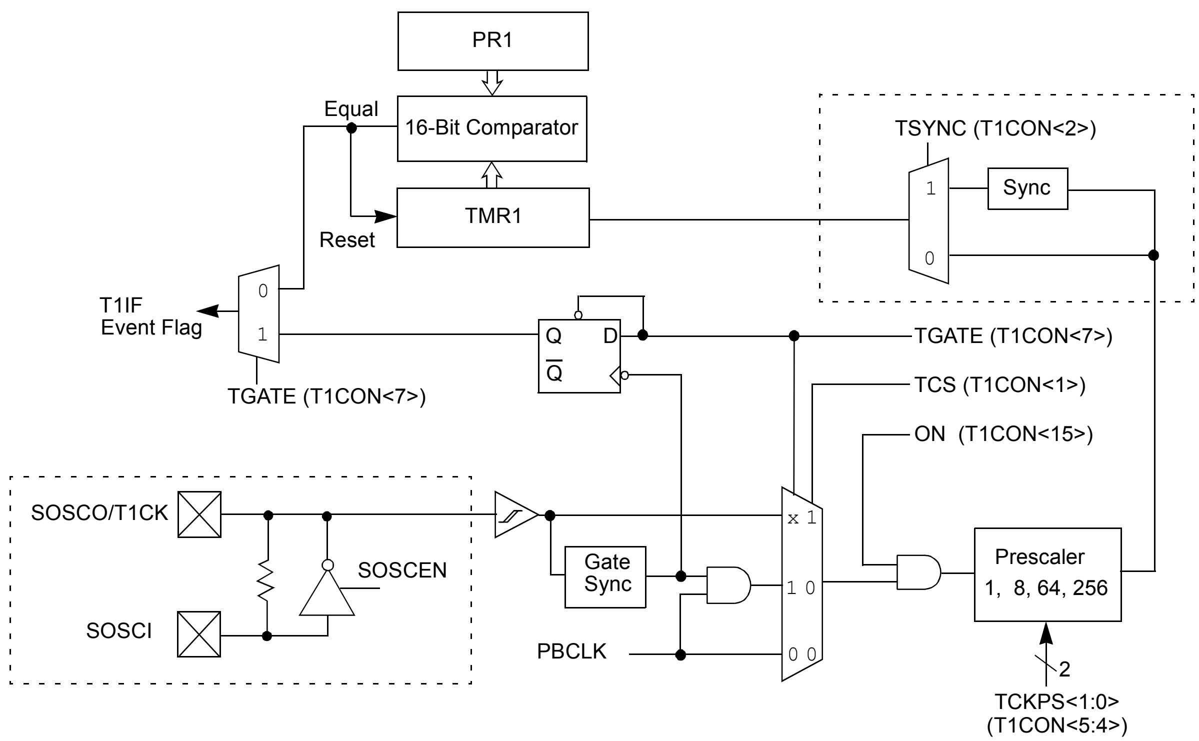
13.1 Additional Supported Features

- Selectable clock prescaler
- Timer operation during CPU Idle and Sleep mode
- Fast bit manipulation using CLR, SET and INV registers
- Asynchronous mode can be used with the Low-Power Secondary Oscillator to function as a Real-Time Clock (RTC).

TABLE 13-1: TIMER1 FEATURES

Timer	Low-Power Oscillator	Asynchronous External Clock	16-Bit Synchronous Timer/Counter	32-Bit Synchronous Timer/Counter	Gated Timer	Special Event Trigger
Timer 1	Yes	Yes	Yes	No	Yes	No

FIGURE 13-1: TIMER1 BLOCK DIAGRAM(1)



Note 1: The default state of the SOSCEN (OSCCON<1>) during a device Reset is controlled by the FSOSCEN bit in Configuration Word DEVCFG1.

 Advance Information

13.2 Timer Registers

TABLE 13-2: TIMER1 SFR SUMMARY

Virtual Address	Name		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
BF80_0600	T1CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	TMWDIS	TMWIP	—	—	—
		7:0	TGATE	—	TCKPS<1:0>		—	TSYNC	TCS	—
BF80_0604	T1CONCLR	31:0	Write clears selected bits in T1CON, read yields undefined value							
BF80_0608	T1CONSET	31:0	Write sets selected bits in T1CON, read yields undefined value							
BF80_060C	T1CONINV	31:0	Write inverts selected bits in T1CON, read yields undefined value							
BF80_0610	TMR1	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	TMR1<15:8>							
		7:0	TMR1<7:0>							
BF80_0614	TMR1CLR	31:0	Write clears selected bits in TMR1, read yields undefined value							
BF80_0618	TMR1SET	31:0	Write sets selected bits in TMR1, read yields undefined value							
BF80_061C	TMR1INV	31:0	Write inverts selected bits in TMR1, read yields undefined value							
BF80_0620	PR1	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	PR1<15:8>							
		7:0	PR1<7:0>							
BF80_0624	PR1CLR	31:0	Write clears selected bits in PR1, read yields undefined value							
BF80_0628	PR1SET	31:0	Write sets selected bits in PR1, read yields undefined value							
BF80_062C	PR1INV	31:0	Write inverts selected bits in PR1, read yields undefined value							

TABLE 13-3: TIMER1 INTERRUPT REGISTER SUMMARY(1)

Virtual Address	Name		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
BF88_1000	IEC0	7:0	INT1IE	OC1IE	IC2IE	T1IE	INT0IE	CS1IE	CS0IE	CTIE
BF88_1010	IFS0	7:0	INT1IF	OC1IF	IC2IF	T1IF	INT0IF	CS1IF	CS0IF	CTIF
BF88_1080	IPC1	7:0	—	—	—	T1IP<2:0>			T1IS<1:0>	

Note 1: This summary table contains partial register definitions that only pertain to the Timer1 peripheral. Refer to the *“PIC32MX Family Reference Manual”* (DS61132) for a detailed description of these registers.

REGISTER 13-1: T1CON: TIMER1 CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R-0	U-0	U-0	U-0
ON	FRZ	SIDL	TMWDIS	TMWIP	—	—	—
bit 15							bit 8

R/W-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
TGATE	—	TCKPS<1:0>		—	TSYNC	TCS	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** Timer On bit
1 = Timer is enabled
0 = Timer is disabled

bit 14 **FRZ:** Freeze in Debug Exception Mode bit
1 = Freeze operation when CPU is in Debug Exception mode
0 = Continue operation when CPU is in Debug Exception mode

bit 13 **SIDL:** Stop in Idle Mode bit
1 = Discontinue operation when device enters Idle mode
0 = Continue operation in Idle mode

bit 12 **TMWDIS:** Asynchronous Timer Write Disable bit
In Asynchronous Timer mode:
1 = Writes to asynchronous TMR1 are ignored until pending write operation completes
0 = Back-to-back writes are enabled (legacy asynchronous timer functionality)
In Synchronous Timer mode:
This bit has no effect.

bit 11 **TMWIP:** Asynchronous Timer Write in Progress bit
In Asynchronous Timer mode:
1 = Asynchronous write to TMR1 register in progress
0 = Asynchronous write to TMR1 register complete
In Synchronous Timer mode:
This bit is read as ‘0’.

bit 10-8 **Unimplemented:** Read as ‘0’

REGISTER 13-1: T1CON: TIMER1 CONTROL REGISTER (CONTINUED)

bit 7	**TGATE:** Gated Time Accumulation Enable bit When TCS = 1: This bit is ignored and read '0'. When TCS = 0: 1 = Gated time accumulation is enabled 0 = Gated time accumulation is disabled
bit 6	**Unimplemented:** Read as '0'
bit 5-4	**TCKPS<1:0>:** Timer Input Clock prescaler Select bits 11 = 1:256 prescale value 10 = 1:64 prescale value 01 = 1:8 prescale value 00 = 1:1 prescale value
bit 3	**Unimplemented:** Read as '0'
bit 2	**TSYNC:** Timer External Clock Input Synchronization Selection bit When TCS = 1: 1 = External clock input is synchronized 0 = External clock input is not synchronized When TCS = 0: This bit is ignored and read '0'.
bit 1	**TCS:** Timer Clock Source Select bit 1 = External clock from T1CKI pin 0 = Internal peripheral clock
bit 0	**Unimplemented:** Read as '0'

13.3 Modes of Operation

The 16-bit Timer1 peripheral can operate as a synchronous timer using internal or external clock sources, or as a gated timer using internal clock source and external clock pin, or as an asynchronous timer using an external asynchronous clock source, such as the low-power secondary oscillator. Each mode is easily configured and described in the following sections.

13.3.1 CONSIDERATIONS FOR ALL TIMER 1 MODES

- Timer1 module is disabled and powered off when the ON bit (T1CON<15>) = 0, thus providing maximum power savings. All other TxCON bits remain unchanged.
- Updates to the T1CON register should only be performed when the timer module is disabled, ON bit (T1CON<15>) = 0.
- Timer1 continues operating when the CPU goes into Idle mode if the “Stop In Idle mode” control bit is disabled, SIDL (TxCON<13>) bit = 0. If enabled, SIDL = 1, the timer module stops operation while the CPU is in Idle mode.
- Setting or clearing the ON bit (T1CON<15>) and any other bits in T1CON in the same instruction may cause undefined behavior. The user is advised to program the T1CON register with the desired settings with one instruction, and then set the ON bit in a subsequent instruction.

13.3.2 SYNCHRONOUS INTERNAL TIMER

In this mode, the timer clock source is the internal PBCLK (Peripheral Bus Clock), TCS (TxCON<1>) = 0. Clock synchronization is not required, therefore the Timer1 Synchronization bit, TSYNC (T1CON<2>), is ignored. The TMR1 Count register increments on every PBCLK clock cycle when the timer clock prescale <TCKPS> is 1:1.

Timer1 generates a timer match event after the TMR1 Count register matches the PR1 Period register value (mid-clock cycle on the falling edge), then resets to 0x0000 on the next PBCLK clock cycle. See **Section 13.5 “Timer Interrupts”** regarding timer events and interrupts.

For clock prescale = N (other than 1:1), the timer operates at a clock rate = PBCLK/N and the TMRx Count register increments on every Nth PBCLK clock. For further details regarding the timer prescaler, refer to **Section 13.4.2 “Timer Clock Prescaler”**.

The following steps should be performed to properly configure the Timer1 peripheral for Timer mode operation.

1. Clear ON control bit (T1CON<15>) = 0 to disable timer.
2. Configure TCKPS control bits (T1CON<5:4) to select desired timer clock prescale.
3. Set TCS control bit (T1CON<1>) = 0 to select the internal PBCLK clock source.
4. Clear TMR1 register.
5. Load PR1 register with desired 16-bit match value.
6. If timer interrupts are to be used, refer to **Section 13.5 “Timer Interrupts”** for interrupt configuration steps.
7. Set ON control bit = 1 to enable Timer.

EXAMPLE 13-1: SYNCHRONOUS INTERNAL TIMER INITIALIZATION

```
T1CON = 0x0        // Stop and Init Timer

TMR1 = 0x0;        // Clear timer register

PR1 = 0xFFFF;      // Load period register

T1CONSET = 0x8000;// Start Timer
```

 Advance Information

13.3.3 SYNCHRONOUS EXTERNAL TIMER

In this mode, the timer clock source is an external clock source or pulse applied to the T1CK pin, TCS (T1CON<1>) = `1`. To provide synchronization, Timer1 synchronization bit TSYNC (T1CON<2>) must be set (= `1`). The 16-bit TMR1 Count register increments on every synchronized rising edge of an external clock when the timer clock prescale <TCKPS> is 1:1.

Timer1 generates a timer match event after the TMR1 Count register matches the PR1 Period register value (mid-clock cycle on the falling edge), then resets to 0x0000 on the next synchronized external clock cycle. The timer continues to increment and repeat the period match until the timer is disabled. For further details regarding timer events and interrupts, see **Section 13.5 "Timer Interrupts"**.

For clock prescale = N (other than 1:1), the timer operates at a clock rate = (external clock/N), therefore, the TMRx Count register increments on every Nth external synchronized clock cycle. For further details regarding timer prescaler, refer to **Section 13.4.2 "Timer Clock Prescaler"**.

13.3.3.1 Considerations

- When using an external clock source, regardless of the Timer1 prescale value, 2-3 external clock cycles are required, after the ON bit = `1`, before the TMR1 register begins incrementing.
- Timer1 will not operate from a synchronized external clock source while the CPU is in SLEEP mode, since the synchronizing PB clock is disabled during Sleep mode.

The following steps should be performed to properly configure the Timer1 peripheral for Synchronous Counter mode operation.

1. Clear control bit, ON (T1CON<15>) = `0`, to disable Timer1.
2. Select the desired timer prescaler using bits, TCKPS<1:0> (T1CON<5:4).
3. Set control bit, TCS (T1CON<1>) = `1`, to select an external clock source.
4. Set control bit, TSYNC (T1CON<2>) = `1`, to enable synchronization.
5. Clear Timer register TMR1.
6. Load Period register PR1 with desired 16-bit match value.
7. If timer interrupts are used, refer to **Section 13.5 "Timer Interrupts"** for interrupt configuration steps.
8. Set control bit, ON (T1CON<15>) = `1`, to enable Timer1.

EXAMPLE 13-2: SYNCHRONOUS EXTERNAL TIMER INITIALIZATION

```
T1CON = 0x0;        // Stop Timer and reset
T1CON = 0x0036      // Set prescaler=1:256,
                    // external clock,
                    // synchronous mode

TMR1 = 0x0;         // Clear timer register
PR1 = 0x3FFF;       // Load period register

T1CONSET = 0x8000; // Start Timer
```

13.3.4 ASYNCHRONOUS EXTERNAL TIMER

In this mode, the timer clock source is an external clock source or pulse applied to the T1CK pin, TCS (T1CON<1>) = `1`. Clock synchronization is not required, therefore, the Timer1 clock synchronization bit should be cleared, TSYNC (T1CON<2>) = `0`. The 16-bit TMR1 Count register increments on every rising edge of an external clock when the timer clock prescale <TCKPS> is 1:1.

Timer1 generates a timer match event after the TMR1 Count register matches the PR1 register value (mid-clock cycle on the falling edge), then resets to 0x0000 on the next external clock cycle. The timer continues to increment and repeat the period match until the timer is disabled. For further details regarding timer events and interrupts, see **Section 13.5 "Timer Interrupts"**.

For clock prescale = N (other than 1:1), the timer operates at a clock rate = (external clock/N), therefore, the TMR1 Count register increments on every Nth external clock cycle. For further details regarding the timer prescaler, refer to **Section 13.4.2 "Timer Clock Prescaler"**.

13.3.4.1 Considerations

- Regardless of the Timer1 prescale setting, 2-3 external clocks are required after the ON bit = `1`, before the TMR1 register begins incrementing.
- Timer1 can operate while the CPU is in Sleep mode.
- The Timer1 interrupt can be used to wake the CPU from Sleep mode.
- Typical use is with the Secondary Low-Power Oscillator, SOSC and RTCC Real-Time Clock Calendar peripheral.

Note:	The SOSC oscillator may be used by the CPU as a low-power clock source. Timer 1 does not have exclusive usage to this oscillator. Refer to the "*PIC32MX Family Reference Manual*" (DS61132) regarding the operation of the Secondary Low-Power Oscillator.

13.4 Reading and Writing TMR1 Register

Due to the asynchronous nature of Timer1 operating in Asynchronous Clock mode, reading and writing to the TMR1 Count register requires synchronization between the asynchronous clock source and the internal PBCLK (Peripheral Bus Clock). Timer1 features a Timer Write Disable (TMWDIS) control bit (T1CON<12>) and a TMWIP (TImer Write in Progress) Status bit (T1CON<11>). These bits provide the user with 2 options for safely writing to the TMR1 Count register while Timer1 is enabled. These bits have no affect in Synchronous Clock modes.

- Option 1 – Legacy Timer1 Write mode, TMWDIS bit = 0. To determine when it is safe to write to the TMR1 Count register, it is recommended to poll the TMWIP bit. When TMWIP = 0, it is safe to perform the next write operation to the TMR1 Count register. When TMWIP = 1, the previous write operation to the TMR1 Count register is still being synchronized and any additional write operations should wait until TMWIP = 0.
- Option 2 – New synchronized Timer1 Write mode, TMWDIS 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.

Writing to the TMR1 Count register requires 2 to 3 asynchronous external clock cycles for the value to be synchronized into the TMR1 Count register.

Reading from the TMR1 Count register requires 2 PBCLK cycle delays between the current unsynchronized value in the TMR1 Count register and the synchronized value returned by the read operation. In other words, the value read is always 2 PBCLK cycles behind the actual value in the TMR1 Count register.

The following steps should be performed to properly configure the Timer1 peripheral for Asynchronous Counter mode operation.

1. Clear control bit, ON (T1CON<15>) = 0, to disable Timer1.
2. Select the desired timer prescaler using bits, TCKPS<1:0> (T1CON<5:4>).
3. Set control bit, TCS (T1CON<1>) = 1, to select an external clock source.
4. Set control bit, TSYNC (T1CON<2>) = 0, to disable synchronization.
5. Clear Timer Register, TMR1.
6. Load Period Register, PR1, with desired 16-bit match value.
7. If timer interrupts are used, refer to **13.5 "Timer Interrupts"** for interrupt configuration steps.
8. Set control bit, ON (T1CON<15>) = 1, to enable Timer1.

EXAMPLE 13-3: ASYNCHRONOUS EXTERNAL TIMER INITIALIZATION

```
T1CON = 0x0;        // Stop Time and reset
T1CON = 0x0012;     // Set prescaler at 1:8,
                    // external clock source,
                    // asynchronous mode
TMR1 = 0x0;         // Clear timer register
PR1 = 0x7FFF;       // Load period register

T1CONSET = 0x8000; // Start Timer
```

 Advance Information

13.4.1 Synchronous Internal Gated Timer

In this mode, the timer clock source can only be the internal PBCLK (Peripheral Bus Cock), TCS (T1CON<1>) = 0. The T1CK pin provides the gating mechanism to enable and disable the timer counting, TGATE (T1CON<7>) = 1. Clock synchronization is not required, therefore Timer1 synchronization bit, TSYNC (T1CON<2>), is ignored. The 16-bit TMR1 Count register is enabled on the rising edge of the T1CK pin and increments on every internal PBCLK cycle when the timer clock prescale <TCKPS> is 1:1.

The timer increments until the TMR1 Count register matches the PR1 register value. The TMR1 Count register resets to 0x0000 on the next PBCLK clock cycle. A timer match event is not generated. The timer continues to increment and repeat the period match until the falling edge of the T1CK pin or the timer is disabled. On the falling edge of the gate signal, a timer gate event is generated and the TMR1 Count register stops counting, but is not reset to 0x0000. The TMR1 Count register must be reset in software. For further details regarding timer events and interrupts, see **Section 13.5 "Timer Interrupts"**.

For clock prescale = N (other than 1:1), the timer operates at a clock rate = (PBCLK/N); therefore, the TMR1 Count register increments on every Nth PBCLK clock cycle. For further details regarding timer prescaler, refer to **Section 13.4.2 "Timer Clock Prescaler"**.

The following steps should be performed to properly configure the Timer1 peripheral for Gated Timer mode operation:

1. Clear control bit, ON (T1CON<15>) = 0, to disable Timer1.
2. Select the desired timer prescaler using bits, TCKPS<1:0> (T1CON<5:4>).
3. Set control bit, TCS (T1CON<1>) = 0, to select the internal clock source.
4. Set control bit TGATE (T1CON<6>) = 1.
5. Clear Timer register, TMR1.
6. Load Period register, PR1, with desired 16-bit match value.
7. If timer interrupts are used, refer to **Section 13.5 "Timer Interrupts"** for interrupt configuration steps.
8. Set control bit ON, (T1CON<15>) = 1, to enable Timer1.

EXAMPLE 13-4: SYNCHRONOUS INTERNAL GATED TIMER INITIALIZATION

```
T1CON = 0x0;        // Stop Timer and reset
T1CON = 0x0060;     // Enable gated mode,
                    // prescaler at 1:64,
                    // internal clock source
TMR1 = 0x0;         // Clear timer register
PR1 = 0xFFFF;       // Load period register

T1CONSET = 0x8000;// Start Timer
```

13.4.2 TIMER CLOCK PRESCALER

Timer clock prescale bits, TCKPS<1:0> (T1CON<5:4>), are used to divide the timer clock source, permitting the TMR register to increment on every 1, 8, 64, or 256 (PBCLK or external) clock cycles. For example, if the clock prescale is 1:8, then the timer increments on every 8th timer clock cycle.

Associated with the clock prescale selection bits is a prescale counter. This prescale counter is cleared when any of the following conditions occur:

- Any device Reset, except a Power-on Reset
- The timer is disabled
- A write to the TMR register

Note:	When the timer clock source is external and the timer clock prescale = N (other than 1:1), 2 to 3 external clock cycles are required to reset and synchronize the prescaler.

- When the timer clock source is external and the timer clock prescale = N (other than 1:1), 2 to 3 external clock cycles are required, after the timer ON bit is set = 1, before the TMR1 Count register increments.
- After a timer match event (TMR1 = PR1) and depending on the timer clock prescale setting N (other than 1:1), the timer will require N/2 additional (PBCLK or external) clock cycles before the TMR1 Counter register reset to 0x0000. Reading the TMR1 Count register just after the timer match event, but before the TMR1 Count register is rest, will return the timer match value.

13.5 Timer Interrupts

Timer1 can generate an interrupt on a period match event or a gate event, caused by the falling edge of the external gate signal.

Timer1 sets the interrupt flag bit, T1IF (IFS0<4>), whenever a Timer1 event is generated. Refer to a specific Timer mode for details regarding event conditions. When a Timer1 event is generated, the interrupt flag bit is set within 1 PBCLK + 2 SYSCLK cycles. If Timer1 Interrupt Enable bit is set, T1IE (IEC0<4>) = `1`, an interrupt is generated.

The Timer1 module is enabled as a source of interrupts through its respective interrupt enable bit, T1IE (IEC0<4>). The Timer1 Interrupt Flag, T1IF (IFS0<4>), must be cleared in software.

The interrupt priority level bits and interrupt subpriority level bits must be also be configured:

- T1IP<2:0> (IPC1<4:2>)
- T1IS<1:0> (IPC1<1:0)

Setting Timer1 interrupt priority level = `0` effectively disables the timer's ability to generate an interrupt.

In addition to enabling the Timer1 interrupt, an Interrupt Service Routine, ISR, is generally required. Below is a partial code example of an ISR.

Note:	It is the user's responsibility to clear the corresponding interrupt flag bit before returning from an ISR.

EXAMPLE 13-5: TIMER INTERRUPT AND PRIORITIES

```
T1CON = 0x0             // Stop the Timer and Reset Control register
                        // Set prescaler at 1:1, internal clock source

TMR1 = 0x0;             // Clear timer register
PR1 = 0xFFFF;           // Load period register

IPC1SET = 0x000C;       // Set priority level=3
IPC1SET = 0x0001;       // Set subpriority level=1
                        // Could have also done this in single
                        // operation by assigning IPC1SET = 0x000D

IFS0CLR = 0x0010;       // Clear Timer interrupt status flag
IEC0SET = 0x0010;       // Enable Timer interrupts
T1CONSET = 0x8000;      // Start Timer
```

EXAMPLE 13-6: TIMER ISR

```
void __ISR(TIMER_1_VECTOR, IPL3) T1_Interrupt_ISR(void)
{
    ... perform application specific operations in response to the interrupt

    IFS0CLR  = 0x0010;     // Be sure to clear the Timer 1 interrupt status
}
```

Note:	The timer ISR code example shows MPLAB® C32 C Compiler specific syntax. Refer to your compiler manual regarding support for ISRs.

13.6 I/O Pin Configuration

Table 13-4 provides a summary of I/O pin resources associated with Timer1. The table shows the settings required to make each I/O pin work with a specific timer module.

TABLE 13-4: I/O PIN CONFIGURATION FOR USE WITH THE TIMER MODULE

		Required Settings for Module Pin Control					
I/O Pin Name	**Required**	**Module Enable[(2)]**	**Bit Field[(2)]**	**TRIS**	**Pin Type**	**Buffer Type**	**Description**
T1CK	Yes[(1)]	ON	TCS, TGATE	Input	I	ST	Timer1 External Clock/Gate Input

Legend:
CMOS = CMOS compatible input or output ST = Schmitt Trigger input with CMOS levels
I = Input O = Output

Note 1: This pin is only required for Gated Timer or External Synchronous Clock modes. Otherwise, this pin can be used for general purpose I/O and requires the user to set the corresponding TRIS control register bits.

2: This bit is located in the T1CON register.

NOTES:

14.0 TIMERS 2,3,4,5

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

This family of PIC32MX devices feature four synchronous 16-bit timers (default) that can operate as a free-running interval timer for various timing applications and counting external events. The following modes are supported:

- Synchronous Internal 16-Bit Timer
- Synchronous Internal 16-Bit Gated Timer
- Synchronous External 16-Bit Timer

Two 32-bit synchronous timers are available by combining Timer2 with Timer3 and Timer4 with Timer5. The 32-bit timers can operate in three modes:

- Synchronous Internal 16-Bit Timer
- Synchronous Internal 16-Bit Gated Timer
- Synchronous External 16-Bit Timer

Note:	Throughout this chapter, references to registers TxCON, TMRx, and PRx use 'x' to represent Timer2 through 5 in 16-bit modes. In 32-bit modes, 'x' represents Timer2 or 4; 'y' represents Timer3 or 5.

14.1 Additional Supported Features

- Selectable clock prescaler
- Timers operational during CPU IDLE
- Time base for input capture and output compare modules (Timer2 and Timer3 only)
- ADC event trigger (Timer3 only)
- Fast bit manipulation using CLR, SET and INV registers

Table 14-1 highlights the available features of these timers.

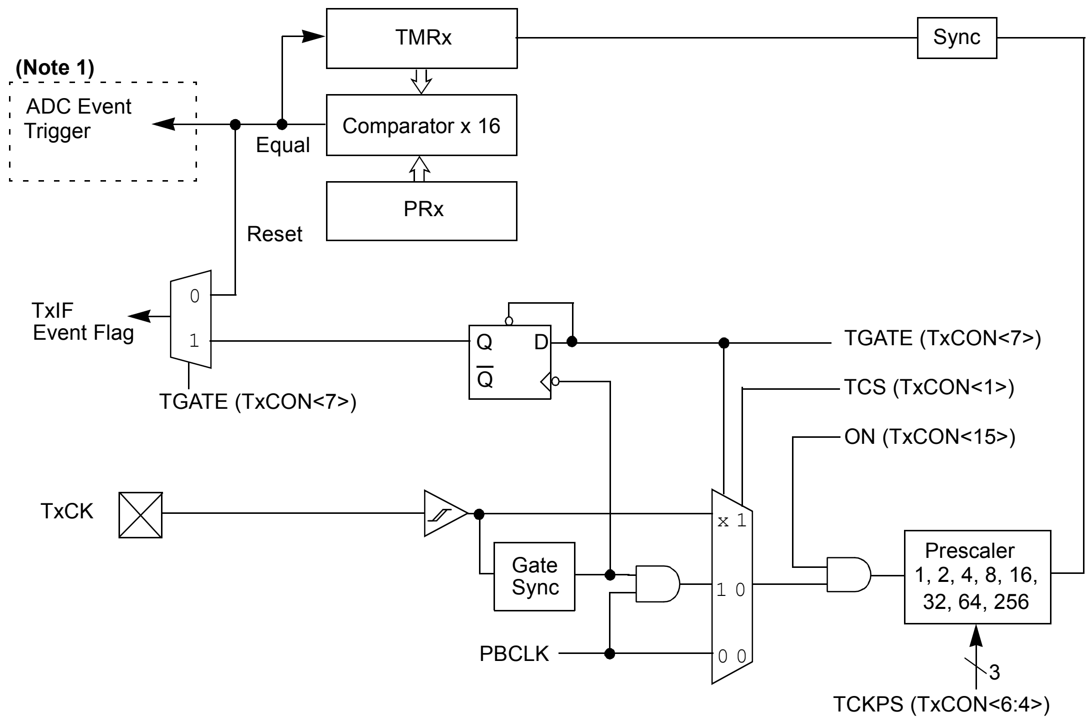
TABLE 14-1: TIMER FEATURES

Timers	Low-Power Oscillator	Asynchronous External Clock	16-Bit Synchronous Timer	32-Bit Synchronous Timer(1)	Gated Timer	Special Event Trigger
2, 4	No	No	Yes	Yes	Yes	No
3, 5	No	No	Yes	Yes	Yes	Yes(2)

Note 1: 32-bit mode requires combining timers 2 and 3 or timers 4 and 5.

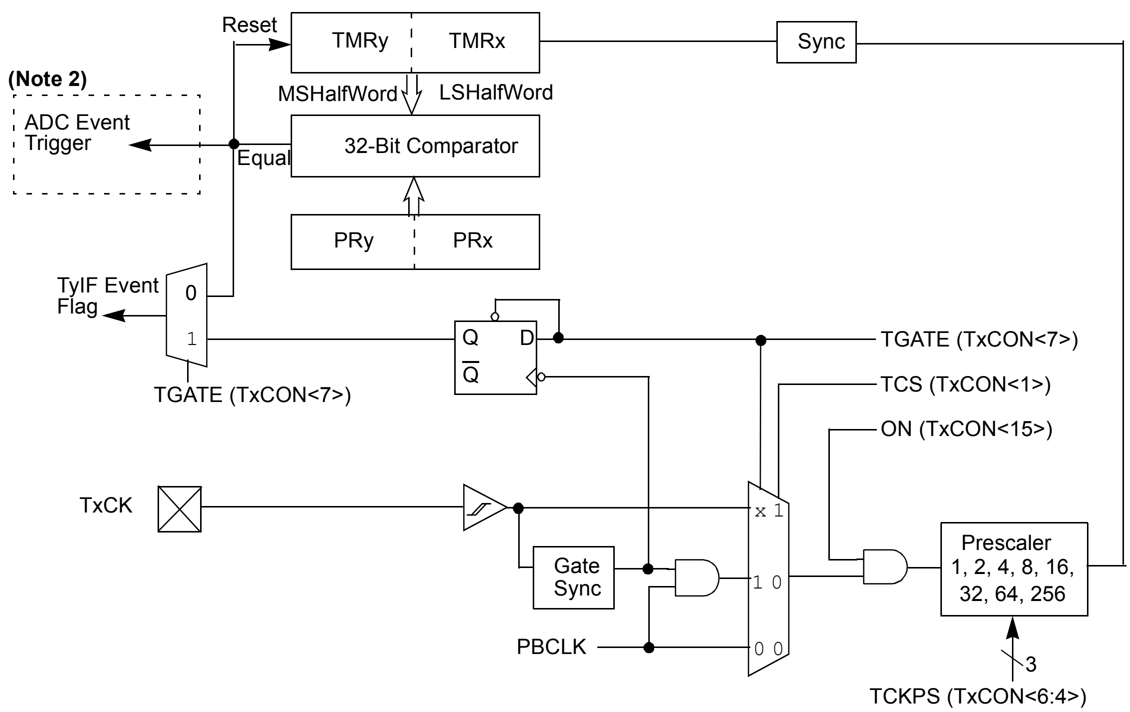
2: ADC event trigger supported by Timer3 only.

FIGURE 14-1: TIMER2, 3, 4, 5 BLOCK DIAGRAM (16-BIT)



Note 1: ADC event trigger is available on Timer3 only.

FIGURE 14-2: TIMER2/3, 4/5 BLOCK DIAGRAM (32-BIT)



Note 1: In this diagram, the use of of "x' in registers TxCON, TMRx, PRx, TxCK refers to either Timer2 or Timer4; the use of 'y' in registers TyCON, TMRy, PRy, TyIF refers to either Timer3 or Timer5.

2: ADC event trigger is available only on Timer2/3 pair.

Advance Information

TABLE 14-1: TIMER2 SFR SUMMARY

Virtual Address	Name		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
BF80_0800	T2CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	—	—
		7:0	TGATE	TCKPS<2:0>			T32	—	TCS	—
BF80_0804	T2CONCLR	31:0	Write clears selected bits in T2CON, read yields undefined value							
BF80_0808	T2CONSET	31:0	Write sets selected bits in T2CON, read yields undefined value							
BF80_080C	T2CONINV	31:0	Write inverts selected bits in T2CON, read yields undefined value							
BF80_0810	TMR2	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	TMR2<15:8>							
		7:0	TMR2<7:0>							
BF80_0814	TMR2CLR	31:0	Write clears selected bits in TMR2, read yields undefined value							
BF80_0818	TMR2SET	31:0	Write sets selected bits in TMR2, read yields undefined value							
BF80_081C	TMR2INV	31:0	Write inverts selected bits in TMR2, read yields undefined value							
BF80_0820	PR2	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	PR2<15:8>							
		7:0	PR2<7:0>							
BF80_0824	PR2CLR	31:0	Write clears selected bits in PR2, read yields undefined value							
BF80_0828	PR2SET	31:0	Write sets selected bits in PR2, read yields undefined value							
BF80_082C	PR2INV	31:0	Write inverts selected bits in PR2, read yields undefined value							

TABLE 14-2: TIMER2 INTERRUPT REGISTER SUMMARY(1)

Virtual Address	Name		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
BF88_1040	IEC0	15:8	INT3IE	OC3IE	IC3IE	T3IE	INT2IE	OC2IE	IC2IE	T2IE
BF88_1010	IFS0	15:8	INT3IF	OC3IF	IC3IF	T3IF	INT2IF	OC2IF	IC2IF	T2IF
BF88_1090	IPC2	7:0	—	—	—	T2IP<2:0>			T2IS<1:0>	

Note 1: This summary table contains partial register definitions that only pertain to the Timer2 peripheral. Refer to the "*PIC32MX Family Reference Manual*" (DS61132) for a detailed description of these registers.

TABLE 14-3: TIMER3 SFR SUMMARY

Virtual Address	Name		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
BF80_0A00	T3CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	—	—
		7:0	TGATE	TCKPS<2:0>			—	—	TCS	—
BF80_0A04	T3CONCLR	31:0	Write clears selected bits in T3CON, read yields undefined value							
BF80_0A08	T3CONSET	31:0	Write sets selected bits in T3CON, read yields undefined value							
BF80_0A0C	T3CONINV	31:0	Write inverts selected bits in T3CON, read yields undefined value							
BF80_0A10	TMR3	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	TMR3<15:8>							
		7:0	TMR3<7:0>							
BF80_0A14	TMR3CLR	31:0	Write clears selected bits in TMR3, read yields undefined value							
BF80_0A18	TMR3SET	31:0	Write sets selected bits in TMR3, read yields undefined value							
BF80_0A1C	TMR3INV	31:0	Write inverts selected bits in TMR3, read yields undefined value							

TABLE 14-3: TIMER3 SFR SUMMARY (CONTINUED)

Virtual Address	Name		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
BF80_0A20	PR3	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	PR3<15:8>							
		7:0	PR3<7:0>							
BF80_0A24	PR3CLR	31:0	Write clears selected bits in PR3, read yields undefined value							
BF80_0A28	PR3SET	31:0	Write sets selected bits in PR3, read yields undefined value							
BF80_0A2C	PR3INV	31:0	Write inverts selected bits in PR3, read yields undefined value							

TABLE 14-4: TIMER3 INTERRUPT REGISTER SUMMARY(1)

Virtual Address	Name		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
BF88_1040	IEC0	15:8	INT3IE	OC3IE	IC3IE	T3IE	INT2IE	OC2IE	IC2IE	T2IE
BF88_1010	IFS0	15:8	INT3IF	OC3IF	IC3IF	T3IF	INT2IF	OC2IF	IC2IF	T2IF
BF88_10A0	IPC3	7:0	—	—	—	T3IP<2:0>			T3IS<1:0>	

Note 1: This summary table contains partial register definitions that only pertain to the Timer 3 peripheral. Refer to the PIC32MX Family Reference Manual (DS61132) for a detailed description of these registers.

REGISTER 14-5: TIMER4 SFR SUMMARY

Virtual Address	Name		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
BF80_0C00	T4CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	—	—
		7:0	TGATE	TCKPS<2:0>			T32	—	TCS	—
BF80_0C04	T4CONCLR	31:0	Write clears selected bits in T4CON, read yields undefined value							
BF80_0C08	T4CONSET	31:0	Write sets selected bits in T4CON, read yields undefined value							
BF80_0C0C	T4CONINV	31:0	Write inverts selected bits in T4CON, read yields undefined value							
BF80_0C10	TMR4	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	TMR4<15:8>							
		7:0	TMR4<7:0>							
BF80_0C14	TMR4CLR	31:0	Write clears selected bits in TMR4, read yields undefined value							
BF80_0C18	TMR4SET	31:0	Write sets selected bits in TMR4, read yields undefined value							
BF80_0C1C	TMR4INV	31:0	Write inverts selected bits in TMR4, read yields undefined value							
BF80_0C20	PR4	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	PR4<15:8>							
		7:0	PR4<7:0>							
BF80_0C24	PR4CLR	31:0	Write clears selected bits in PR4, read yields undefined value							
BF80_0C28	PR4SET	31:0	Write sets selected bits in PR4, read yields undefined value							
BF80_0C2C	PR4INV	31:0	Write inverts selected bits in PR4, read yields undefined value							

REGISTER 14-6: TIMER 4 INTERRUPT REGISTER SUMMARY(1)

Virtual Address	Name		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
BF88_1040	IEC0	23:16	SPI1EIE	OC5IE	IC5IE	T5IE	INT4IE	OC4IE	IC4IE	T4IE
BF88_1010	IFS0	23:16	SPI1EIF	OC5IF	IC5IF	T5IF	INT4IF	OC4IF	IC4IF	T4IF
BF88_10B0	IPC4	7:0	—	—	—	T4IP<2:0>			T4IS<1:0>	

Note 1: This summary table contains partial register definitions that only pertain to the Timer4 peripheral. Refer to the "*PIC32MX Family Reference Manual*" (DS61132) for a detailed description of these registers.

TABLE 14-7: TIMER5 SFR SUMMARY

Virtual Address	Name		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
BF80_0E00	T5CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	—	—
		7:0	TGATE	TCKPS<2:0>			—	—	TCS	—
BF80_0E04	T5CONCLR	31:0	Write clears selected bits in T5CON, read yields undefined value							
BF80_0E08	T5CONSET	31:0	Write sets selected bits in T5CON, read yields undefined value							
BF80_0E0C	T5CONINV	31:0	Write inverts selected bits in T5CON, read yields undefined value							
BF80_0E10	TMR5	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	TMR5<15:8>							
		7:0	TMR5<7:0>							
BF80_0E14	TMR5CLR	31:0	Write clears selected bits in TMR5, read yields undefined value							
BF80_0E18	TMR5SET	31:0	Write sets selected bits in TMR5, read yields undefined value							
BF80_0E1C	TMR5INV	31:0	Write inverts selected bits in TMR5, read yields undefined value							
BF80_0E20	PR5	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	PR5<15:8>							
		7:0	PR5<7:0>							
BF80_0E24	PR5CLR	31:0	Write clears selected bits in PR5, read yields undefined value							
BF80_0E28	PR5SET	31:0	Write sets selected bits in PR5, read yields undefined value							
BF80_0E2C	PR5INV	31:0	Write inverts selected bits in PR5, read yields undefined value							

TABLE 14-8: TIMER5 INTERRUPT REGISTER SUMMARY(1)

Virtual Address	Name		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
BF88_1040	IEC0	23:16	SPI1EIE	OC5IE	IC5IE	T5IE	INT4IE	OC4IE	IC4IE	T4IE
BF88_1010	IFS0	23:16	SPI1EIF	OC5IF	IC5IF	T5IF	INT4IF	OC4IF	IC4IF	T4IF
BF88_10C0	IPC5	7:0	—	—	—	T5IP<2:0>			T5IS<1:0>	

Note 1: This summary table contains partial register definitions that only pertain to the Timer5 peripheral. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of these registers.

14.2 Control Registers

REGISTER 14-9: T2CON, T4CON: TIMER2 AND TIMER4 CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
ON	FRZ	SIDL	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0
TGATE	TCKPS<2:0>			T32	—	TCS	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** Timer On bit
1 = Timer is enabled
0 = Timer is disabled

bit 14 **FRZ:** Freeze in Debug Exception Mode bit
1 = Freeze operation when CPU is in Debug Exception mode
0 = Continue operation when CPU is in Debug Exception mode

bit 13 **SIDL:** Stop in Idle Mode bit
1 = Discontinue operation when device enters Idle mode
0 = Continue operation in Idle mode

bit 12-8 **Unimplemented:** Read as ‘0’

bit 7 **TGATE:** Gated Time Accumulation Enable bit
<u>When TCS = 1:</u>
This bit is ignored and read ‘0’.
<u>When TCS = 0:</u>
1 = Gated time accumulation is enabled
0 = Gated time accumulation is disabled

bit 6-4 **TCKPS<2:0>:** Timer Input Clock prescaler Select bits
111 = 1:256 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	**T32:** 32-Bit Timer Mode Select bits 1 = TMRx and TMRy form a 32-bit timer 0 = TMRx and TMRy form separate 16-bit timers
bit 2	**Unimplemented:** Read as ‘0’
bit 1	**TCS:** Timer Clock Source Select bit 1 = External clock from TxCK pin 0 = Internal peripheral clock
bit 0	**Unimplemented:** Read as ‘0’

REGISTER 14-10: T3CON, T5CON: TIMER3 AND TIMER5 CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
ON	FRZ	SIDL	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	U-0
TGATE	TCKPS<2:0>			—	—	TCS	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** Timer On bit
1 = Module is enabled
0 = Module is disabled

bit 14 **FRZ:** Freeze in Debug Exception Mode bit
1 = Freeze operation when CPU is in Debug Exception mode
0 = Continue operation when CPU is in Debug Exception mode

bit 13 **SIDL:** Stop in Idle Mode bit
1 = Discontinue operation when device enters Idle mode
0 = Continue operation in Idle mode

bit 12-8 **Unimplemented:** Read as ‘0’

bit 7 **TGATE:** Gated Time Accumulation Enable bit
When TCS = 1:
This bit is ignored and read ‘0’.
When TCS = 0:
1 = Gated time accumulation is enabled
0 = Gated time accumulation is disabled

bit 6-4 **TCKPS<1:0>:** Timer Input Clock Prescaler Select bits
111 = 1:256 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 3-2 **Unimplemented:** Read as ‘0’

bit 1	**TCS:** Timer Clock Source Select bit 1 = External clock from TxCK pin 0 = Internal peripheral clock	
bit 0	**Unimplemented:** Read as '0'	

14.3 Modes of Operation

The 16-bit (default) and 32-bit mode timer peripherals can operate as synchronous timer/counters using internal or external clock sources, or as synchronous gated timers using an internal clock source and external clock/gate pins. Each mode is easily configured and described in the following sections.

14.3.1 CONSIDERATIONS FOR ALL TIMER MODES

- A timer module is disabled and powered off when the ON bit (TxCON<15>) = `0`, thus providing maximum power savings. All other TxCON bits remain unchanged.
- Updates to the TxCON register should only be performed when the timer module is disabled, ON bit (TxCON<15>) = `0`.
- A timer continues operating when the CPU goes into Idle mode if the “Stop In Idle mode” control bit is disabled, SIDL (TxCON<13>) bit = `0`. If enabled, SIDL = `1`, the timer module stops operation while the CPU is in Idle mode.
- Setting or clearing the ON bit (TxCON<15>) and any other bits in the TxCON register during a single instruction may cause undefined behavior. The user is advised to program the TxCON register with the desired settings with one instruction, and then set the ON bit in a subsequent instruction.

14.3.2 SYNCHRONOUS INTERNAL 16-BIT TIMER

In this mode, the timer clock source is the internal PBCLK (Peripheral Bus Clock), TCS (TxCON<1>) = `0`. The 16-bit TMRx Count register increments on every internal PBCLK cycle when the timer clock prescale <TCKPS> is 1:1.

The timer generates a timer match event after the TMRx Count register matches the PRx Period register value, then resets to 0x0000 on the next PBCLK clock cycle. The timer continues to increment and repeat the period match until the timer is disabled. For further details regarding timer events and interrupts, see **Section 14.4 Timer Interrupts**.

For clock prescale = N (other than 1:1), the timer operates at a clock rate = (PBCLK/N); therefore, the TMRx Count register increments on every Nth PBCLK clock cycle. For further details regarding timer prescaler, refer to **Section 14.3.9 Timer Clock Prescaler**.

The following steps should be performed to properly configure the 16-bit Timer peripherals for Timer mode operation:

1. Clear ON control bit, (TxCON<15>) = `0`, to disable timer.
2. Configure TCKPS control bits, (TxCON<6:4>), to select desired timer clock prescale.
3. Set TCS control bit, (TxCON<1>) = `0`, to select the internal PBCLK clock source.
4. Clear TMRx register.
5. Load PRx register with desired 16-bit match value.
6. If timer interrupts are to be used, refer to **Section 14.4 Timer Interrupts** for interrupt configuration steps.
7. Set ON control bit = `1` to enable Timer.

EXAMPLE 14-1: SYNCHRONOUS INTERNAL 16-BIT TIMER INITIALIZATION

```
T2CON = 0x0;        //Stop and Init Timer

TMR2 = 0x0;        //Clear timer register

PR2 = 0xFFFF;      //Load period register

T2CONSET = 0x8000; // Start Timer
```

14.3.3 SYNCHRONOUS EXTERNAL 16-BIT TIMER

In this mode, the timer clock source is an external clock source or pulse applied to the TxCK pin, TCS (TxCON<1>) = `1`. The 16-bit TMRx Count register increments on every rising edge of an external clock when the timer clock prescale <TCKPS> is 1:1.

The timer generates a timer match event after the TMRx Count register matches the PRx register value, then resets to 0x0000 on the next external clock cycle. The timer continues to increment and repeat the period match until the timer is disabled. For further details regarding timer events and interrupts, see **Section 14.4 Timer Interrupts**.

For clock prescale = N (other than 1:1), the timer operates at a clock rate = (external clock/N); therefore, the TMRx Count register increments on every Nth external clock cycle. For further details regarding the timer prescaler, refer to **Section 14.3.9 Timer Clock Prescaler**.

The following steps should be performed to properly configure the timer peripheral for Synchronous Counter mode operation:

1. Clear control bit, ON (TxCON<15>) = 0, to disable timer.
2. Select the desired timer prescaler using bits, TCKPS<2:0> (TxCON<6:4).
3. Set control bit, TCS (TxCON<1>) = 1, to select an external clock source.
4. Clear Timer register, TMRx.
5. Load Period register, PRx, with desired 16-bit match value.
6. If timer interrupts are used, refer to **Section 14.4 Timer Interrupts** for interrupt configuration steps.
7. Set control bit, ON (TxCON<15>) = 1, to enable timer.

EXAMPLE 14-2: SYNCHRONOUS EXTERNAL 16-BIT TIMER INITIALIZATION

```
T3CON = 0x0;        //Stop and Init Timer

T3CONSET = 0x0072; //Prescaler=1:256,
                    //external clock
TMR3 = 0x0;         //Clear timer register

PR3 = 0x3FFF;       //Load period register

T3CONSET = 0x8000;//Start Timer
```

14.3.4 SYNCHRONOUS INTERNAL 16-BIT GATED TIMER

In this mode, the timer clock source can only be the internal PBCLK (Peripheral Bus Clock), TCS (TxCON<1>) = 0. The TxCK pin provides the gating mechanism to enable and disable the timer counting, TGATE (TxCON<7>) = 1. The 16-bit TMRx Count register is enabled on the rising edge of the TxCK pin and increments on every internal PBCLK cycle when the timer clock prescale <TCKPS> is 1:1.

The timer increments until the TMRx Count register matches the PRx register value. The TMRx Count register resets to 0x0000 on the next PBCLK clock cycle. A timer match event is not generated. The timer continues to increment and repeat the period match until the falling edge of the TxCK pin or the timer is disabled. On the falling edge of the gate signal, a timer gate event is generated and the TMRx Count register stops counting, but is not reset to 0x0000. The TMRx Count register must be reset in software. For further details regarding timer events and interrupts, see **Section 14.4 Timer Interrupts**.

For clock prescale = N (other than 1:1), the timer operates at a clock rate = (PBCLK/N); therefore, the TMRx Count register increments on every Nth PBCLK clock cycle. For further details regarding timer prescaler, refer to **Section 14.3.9 Timer Clock Prescaler**.

The following steps should be performed to properly configure the timer peripheral for Gated Timer mode operation:

1. Clear control bit, ON (TxCON<15>) = 0, to disable Timer.
2. Select the desired timer prescaler using bits, TCKPS<2:0> (TxCON<6:4>).
3. Set control bit, TCS (TxCON<1>) = 0, to select the internal clock source.
4. Set control bit, TGATE (TxCON<7>) = 1.
5. Clear Timer register, TMRx.
6. Load Period register, PRx, with desired 16-bit match value.
7. If timer interrupts are to be used, refer to **Section 14.4 Timer Interrupts** for interrupt configuration steps.
8. Set control bit, ON (TxCON<15>) = 1, to enable timer.

EXAMPLE 14-3: SYNCHRONOUS INTERNAL 16-BIT GATED TIMER INITIALIZATION

```
T4CON = 0x0;        //Stop and Init Timer

T4CON = 0x00E0;     //Enable gated mode,
                    //prescaler=1:64,
                    //internal clock
TMR4 = 0;           //Clear timer register

PR4 = 0xFFFF;       //Load period register

T4CONSET = 0x8000;//Start Timer
```

14.3.5 SYNCHRONOUS INTERNAL 32-BIT TIMER

In this mode, T32 (TxCON<3>) = 1 and the timer clock source is the internal PBCLK (Peripheral Bus Clock), TCS (TxCON<1>) = 0. The 32-bit TMRxy Count register increments on every internal PBCLK cycle when the timer clock prescale <TCKPS> is 1:1.

The timer generates a timer match event after the TMRxy Count register matches the PRxy Period register value, then resets to 0x00000000 on the next PBCLK clock cycle. The timer continues to increment and repeat the period match until the timer is disabled. For further details regarding timer events and interrupts, see **Section 14.4 Timer Interrupts**.

For clock prescale = N (other than 1:1), the timer operates at a clock rate = (PBCLK/N); therefore, the TMRxy Count register increments on every Nth PBCLK clock cycle. For further details regarding the timer prescaler, refer to **Section 14.3.9 Timer Clock Prescaler**.

14.3.6 CONSIDERATIONS

- 32-bit timer pairs can be created using Timer2 with Timer3, or Timer4 with Timer5.
- With Timer2 or Timer4 enabled, setting the T32 bit (T2CON<3> or T4CON<3>) = `1` automatically enables the corresponding Timer3 or Timer5 module. For this reason, it is not necessary to manually enable Timer3 or Timer5.
- T2CON and T4CON control registers are used for configuring the 32-bit timer operations; Writes to T3CON and T5CON are ignored.
- T2CK and T4CK input pins are utilized for the 32-bit gated timer or external synchronous counter operations; T3CK and T5CK are ignored.
- 32-bit timer interrupts use Timer3 or Timer5 interrupt enable bits and interrupt flag bits; Timer2 and Timer4 interrupt enable and interrupt flag bits are ignored.
- Load TMRxy pair by writing the 32-bit value to TMRx.
- Load PRxy pair by writing the 32-bit value to PRx.

The following steps should be performed to properly configure the 32-bit timer peripherals for Timer mode operation.

1. Clear control bit, ON (TxCON<15>) = `0`, to disable timer.
2. Set control bit, T32 (TxCON<3>).
3. Select the desired timer prescaler using bits TCKPS<2:0> (TxCON<6:4>).
4. Set control bit, TCS (TxCON<1>) = `0`, to select the internal clock source.
5. Clear Timer register, TMRxy.
6. Load Period register, PRxy, with desired 32-bit match value.
7. If timer interrupts are used, refer to **Section 14.4 Timer Interrupts** for interrupt configuration steps.
8. Set control bit, ON (TxCON<15>) = `1`, to enable timer.

EXAMPLE 14-4: SYNCHRONOUS INTERNAL 32-BIT TIMER INITIALIZATION

```
T4CON = 0x0;        //Stop Timer4 and clear
T5CON = 0x0;        //Stop Timer5 and clear
T4CONSET = 0x0038; // Enable 32-bit mode,
                   // prescaler at 1:8,
                   // internal clock
TMR4 = 0x0;        // Clear TMR4 and TMR5
                   // Same as TMR4 = 0x0
PR4 = 0xFFFFFFFF;  // Load PR4 and PR5
                   // with 32-bit value
                   // Same as PR4=0xFFFFFFFF
T4CONSET = 0x8000; // Start Timer
```

14.3.7 SYNCHRONOUS EXTERNAL 32-BIT TIMER

In this mode, T32 (TxCON<3>) = `1` and the timer clock source is an external clock source or pulse applied to the TxCK pin, TCS (TxCON<1>) = `1`. The 32-bit TMRxy Count register increments on every synchronized rising edge of an external clock when the timer clock prescale <TCKPS> is 1:1.

The timer generates a timer match event after the TMRxy Count register matches the PRxy register value, then resets to 0x00000000 on the next external clock cycle. The timer continues to increment and repeat the period match until the timer is disabled. For further details regarding timer events and interrupts, see **Section 14.4 Timer Interrupts**.

For clock prescale = N (other than 1:1), the timer operates at a clock rate = (external clock/N); therefore, the TMRxy Count register increments on every Nth external clock cycle. For further details regarding timer prescaler, refer to **Section 14.3.9 Timer Clock Prescaler**.

The following steps should be performed to properly configure the 32-bit timer peripheral for Synchronous Counter mode operation:

1. Clear control bit, ON (TxCON<15>) = `0`, to disable Timer.
2. Set control bit, T32 (TxCON<3>).
3. Select the desired timer prescaler using bits TCKPS<2:0> (TxCON<6:4>).
4. Set control bit, TCS (TxCON<1>) = `1`, to select an external clock source.
5. Clear Timer register, TMRx.
6. Load Period register, PRx, with desired 32-bit match value.
7. If timer interrupts are used, refer to **Section 14.4 Timer Interrupts** for interrupt configuration steps.
8. Set control bit, ON (TxCON<15>) = `1`, to enable timer.

EXAMPLE 14-5: SYNCHRONOUS EXTERNAL 32-BIT TIMER INITIALIZATION

```
T2CON = 0x0;      //Stop Timer2 and clear
T3CON = 0x0;      //Stop Timer3 and clear
T2CONSET = 0x006A  //32-bit mode,
                   //external clock,
                   //prescale=1:64
TMR2 = 0x0;        // Clear TMR2 and TMR3
                   // Same as TMR2 = 0x0
PR2 = 0xFFFFFFFF; // Load PR2 and PR3
                   // Same as PR2=0xFFFFFFFF
T2CONSET = 0x8000;  // Start timer
```

14.3.8 SYNCHRONOUS INTERNAL 32-BIT GATED TIMER

In this mode, the timer clock source is the internal PBCLK (Peripheral Bus Clock), TCS (TxCON<1>) = `0`. The TxCK pin provides the gating mechanism to enable and disable the timer counting, TGATE (TxCON<7>) = `1`. The 32-bit TMRxy Count register is enabled on the rising edge of the TxCK pin and increments on every internal PBCLK cycle when the timer clock prescale <TCKPS> is 1:1.

The timer increments until the TMRxy Count register matches the PRxy register value. The TMRxy Count register resets to 0x00000000 on the next PBCLK clock cycle. A timer match event is not generated. The timer continues to increment and repeat the period match until the falling edge of the TxCK pin or the timer is disabled. On the falling edge of the gate signal, a timer gate event is generated and the TMRxy Count register stops counting, but is not reset to 0x00000000. The TMRxy Count register must be reset in software. For further details regarding timer events and interrupts, see **Section 14.4 Timer Interrupts**.

For clock prescale = N (other than 1:1), the timer operates at a clock rate = (PBCLK/N); therefore, the TMRxy Count register increments on every Nth timer clock cycle. For further details regarding timer prescaler, refer to **Section 14.3.9 Timer Clock Prescaler**.

The following steps should be performed to properly configure the timer peripheral for Gated Timer mode operation:

1. Clear control bit, ON (TxCON<15>) = `0`, to disable timer.
2. Set control bit, T32 (TxCON<3>).
3. Select the desired timer prescaler using bits TCKPS<2:0> (TxCON<6:4>).
4. Set control bit, TCS (TxCON<1>) = `0`, to select the internal clock source.
5. Set control bit, TGATE (TxCON<7>) = `1`.
6. Clear Timer register, TMRx.
7. Load Period register, PRx, with desired 32-bit match value.
8. Set control bit, ON (TxCON<15>) = `1`, to enable timer.

EXAMPLE 14-6: SYNCHRONOUS INTERNAL 32-BIT GATED TIMER INITIALIZATION

```
T4CON = 0x0;        //Stop Timer4 and clear
T5CON = 0x0;        //Stop Timer5 and clear
T4CONSET = 0x00C8; //32-bit mode,
                    //gate enable,
                    //internal clock,
                    //1:16 prescale
TMR4 = 0x0;         //Clear TMR4 and TMR5
                    //Same as TMR4 = 0x0
PR4 = 0xFFFFFFFF;  //Load PR4 and PR5 regs
                   //Same as PR4 =0xFFFFFFFF
T4CONSET = 0x8000; //Start 32-bit timer
```

14.3.9 TIMER CLOCK PRESCALER

Timer clock prescale bits, TCKPS<1:0> (TxCON<6:4>), are used to divide the timer clock source permitting the TMR register to increment on every 1, 2, 4, 8, 16, 32, 64, or 256 (PBCLK or external) clock cycles. For example, if the clock prescale is 1:8, then the timer increments on every 8th timer clock cycle.

14.3.10 CONSIDERATIONS

Associated with the clock prescale selection bits is a prescale counter. The timer prescale counter is cleared when any of the following conditions occur:

1. Any device Reset, except a Power-on Reset.
2. The timer is disabled.
3. Any write to the TMR register.

Note:	When the timer clock source is external and the timer clock prescale = N (other than 1:1), 2 to 3 external clock cycles are required to reset and synchronize the prescaler.

- When the timer clock source is external and the timer clock prescale = N (other than 1:1), 2 to 3 external clock cycles are required, after the timer ON bit is set = `1`, before the TMRx Count register increments.
- After a timer match event (TMRx = PRx) and depending on the timer clock prescale setting N (other than 1:1), the timer will require N additional (PBCLK or external) clock cycles before the TMRx Counter register resets to 0x0000. Reading the TMRx Count register just after the timer match event, but before the TMRx Count register is reset, will return the timer match value.

14.4 Timer Interrupts

A timer can generate an interrupt on a period match event or a gate event, caused by the falling edge of the external gate signal.

A timer sets its corresponding interrupt flag bit, TxIF, whenever the timer event is generated. Refer to a specific timer mode for details regarding these event conditions. When a timer event is generated, the interrupt flag bit is set within 1 PBCLK + 2 SYSCLK cycles. If the timer interrupt enable bit is set, TxIE = `1`, an interrupt is generated.

The timer module is enabled as a source of interrupts via the respective Timer Interrupt Enable bit, TxIE (IECx<n>). The Timer Interrupt Flag, TxIF (IFSx<n>), must be cleared in software.

The interrupt priority level bits and interrupt subpriority level bits must be also be configured:

- TxIP<2:0> (IPCx<4:2>)
- TxIS<1:0> (IPCx<1:0)

Setting the timer's interrupt priority level = `0` effectively disables the timer's ability to generate an interrupt.

In addition to enabling the timer interrupt, an Interrupt Service Routine, ISR, is required. Example 14-7 through Example 14-9 show a partial code example of an ISR.

EXAMPLE 14-7: 16-BIT TIMER INTERRUPT AND PRIORITIES

```
T2CON = 0x0;           // Stop Timer and clear control register,
                       // prescaler at 1:1,internal clock source

TMR2 = 0x0;            // Clear timer register
PR2 = 0xFFFF;          // Load period register

IPC2SET = 0x0000000C;  // Set priority level=3
IPC2SET = 0x00000001;  // Set subpriority level=1
                       // Could have also done this in single
                       // operation by assigning IPC2SET = 0x0000000D

IFS0CLR = 0x00000100;  // Clear Timer interrupt status flag
IEC0SET = 0x00000100;  // Enable Timer interrupts

T2CONSET = 0x8000;     // Start Timer
```

EXAMPLE 14-8: 32-BIT TIMER INTERRUPT AND PRIORITIES

```
T4CON = 0x0;           // Stop 16-bit Timer4 and clear control register
T5CON = 0x0;           // Stop 16-bit Timer5 and clear control register
T4CONSET = 0x0038;     // Enable 32-bit mode, prescaler at 1:8,
                       // internal clock source

TMR4= 0x0;             // Clear contents of the TMR4 and TMR5
                       // registers in one 32-bit load operation
PR4 = 0xFFFFFFFF;      // Load PR4 and PR5 registers with 32-bit value
                       // 0xFFFFFFFF in one 32-bit load operation

IPC5SET = 0x00000004;  // Set priority level=1 and
IPC5SET = 0x00000001;  // Set subpriority level=1
                       // Could have also done this in single
                       // operation by assigning IPC5SET = 0x00000005

IFS0CLR = 0x10000000;  // Clear the Timer5 interrupt status flag
IEC0SET = 0x10000000;  // Enable Timer5 interrupts

T4CONSET = 0x8000;     // Start Timer
```

 Advance Information

EXAMPLE 14-9: TIMER ISR

```
void __ISR(TIMER_2_VECTOR, ipl3) T2_Interrupt_ISR(void)
{
    ... perform application specific operations in response to the interrupt

    IFS0CLR  = 0x00000100; // Be sure to clear the Timer2 interrupt status
}
```

Note:	The timer ISR code example shows MPLAB® C32 Compiler specific syntax. Refer to your compiler manual regarding support for ISRs.

14.4.1 I/O Pin Configuration

The table below provides a summary of I/O pin resources associated with the timer modules. The table shows the settings required to make an I/O pin available for a specific Timer module.

TABLE 14-2: I/O PIN CONFIGURATION FOR USE WITH TIMER MODULES

		Required Settings for Module Pin Control					
I/O Pin Name	**Required**	**Module Enable(2)**	**Bit Field(2)**	**TRIS**	**Pin Type**	**Buffer Type**	**Description**
T2CK	Yes(1)	ON	TCS, TGATE	Input	I	ST	Timer2 External Clock/Gate Input
T3CK	Yes1)	ON	TCS, TGATE	Input	I	ST	Timer3 External Clock/Gate Input
T4CK	Yes(1)	ON	TCS, TGATE	Input	I	ST	Timer4 External Clock/Gate Input
T5CK	Yes(1)	ON	TCS, TGATE	Input	I	ST	Timer5 External Clock/Gate Input

Legend:
CMOS = CMOS compatible input or output   ST = Schmitt Trigger input with CMOS levels
I = Input   O = Output

Note 1: These pins are only required for modes using gated timer or external clock inputs. Otherwise, these pins can be used for general purpose I/O and require the user to set the corresponding TRIS register bits.

2: These bits are located in the TxCON register.

NOTES:

15.0 INPUT CAPTURE

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

The input capture module is useful in applications requiring frequency (period) and pulse measurement. The PIC32MX Family devices support up to five input capture channels.

The input capture module captures the 16-bit or 32-bit value of the selected Time Base registers when an event occurs at the ICx pin. The events that cause a capture event are listed below in three categories:

1. Simple Capture Event modes
   - Capture timer value on every falling edge of input at ICx pin
   - Capture timer value on every rising edge of input at ICx pin
2. Capture timer value on every edge (rising and falling)
3. Capture timer value on every edge (rising and falling), specified edge first.
4. Prescaler Capture Event modes
   - Capture timer value on every 4th rising edge of input at ICx pin
   - Capture timer value on every 16th rising edge of input at ICx pin

Each input capture channel can select between one of two 16-bit timers (Timer2 or Timer3) for the time base, or two 16-bit timers (Timer2 and Timer3) together to form a 32-bit timer. The selected timer can use either an internal or external clock.

Other operational features include:

- Device wake-up from capture pin during CPU Sleep and Idle modes
- Interrupt on input capture event
- 4-word FIFO buffer for capture values
  - Interrupt optionally generated after 1, 2, 3 or 4 buffer locations are filled
- Input capture can also be used to provide additional sources of external interrupts

FIGURE 15-1: INPUT CAPTURE BLOCK DIAGRAM

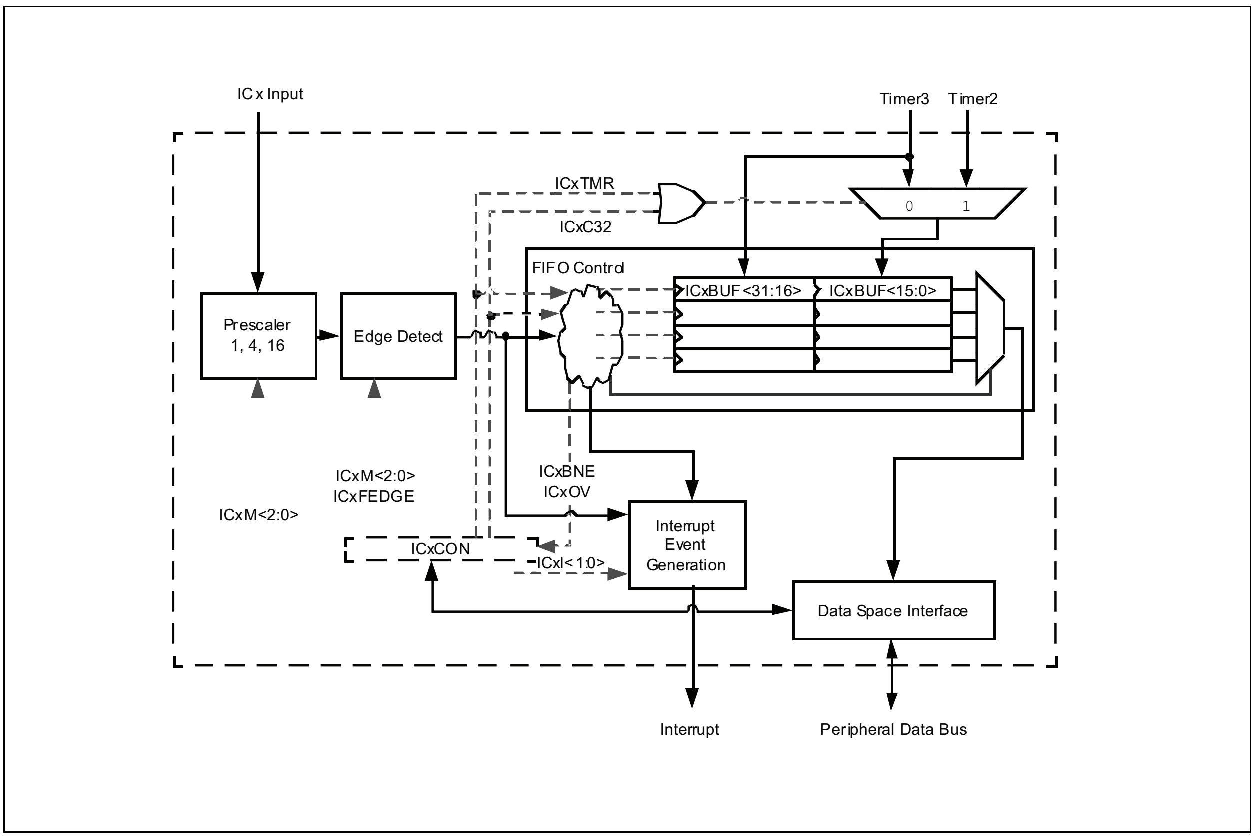


TABLE 15-1: INPUT CAPTURE REGISTER SUMMARY

Virtual Address	Name		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
BF80_2000	IC1CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	ICFEDGE	ICC32
		7:0	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>		
BF80_2004	IC1CONCLR	31:0	Write clears selected bits in IC1CON, read yields an undefined value							
BF80_2008	IC1CONSET	31:0	Write sets selected bits in IC1CON, read yields an undefined value							
BF80_200C	IC1CONINV	31:0	Write inverts selected bits in IC1CON, read yields an undefined value							
BF80_2010	IC1BUF	31:24	IC1BUF<31:24>							
		23:16	IC1BUF<23:16>							
		15:8	IC1BUF<15:8>							
		7:0	IC1BUF<7:0>							
BF80_2200	IC2CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	ICFEDGE	ICC32
		7:0	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>		
BF80_2204	IC2CONCLR	31:0	Write clears selected bits in IC2CON, read yields an undefined value							
BF80_2208	IC2CONSET	31:0	Write sets selected bits in IC2CON, read yields an undefined value							
BF80_220C	IC2CONINV	31:0	Write inverts selected bits in IC2CON, read yields an undefined value							
BF80_2210	IC2BUF	31:24	IC2BUF<31:24>							
		23:16	IC2BUF<23:16>							
		15:8	IC2BUF<15:8>							
		7:0	IC2BUF<7:0>							
BF80_2400	IC3CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	ICFEDGE	ICC32
		7:0	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>		
BF80_2404	IC3CONCLR	31:0	Write clears selected bits in IC3CON, read yields an undefined value							
BF80_2408	IC3CONSET	31:0	Write sets selected bits in IC3CON, read yields an undefined value							
BF80_240C	IC3CONINV	31:0	Write inverts selected bits in IC3CON, read yields an undefined value							
BF80_2410	IC3BUF	31:24	IC3BUF<31:24>							
		23:16	IC3BUF<23:16>							
		15:8	IC3BUF<15:8>							
		7:0	IC3BUF<7:0>							
BF80_2600	IC4CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	ICFEDGE	ICC32
		7:0	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>		
BF80_2604	IC4CONCLR	31:0	Write clears selected bits in IC4CON, read yields an undefined value							
BF80_2608	IC4CONSET	31:0	Write sets selected bits in IC4CON, read yields an undefined value							
BF80_260C	IC4CONINV	31:0	Write inverts selected bits in IC4CON, read yields an undefined value							
BF80_2610	IC4BUF	31:24	IC4BUF<31:24>							
		23:16	IC4BUF<23:16>							
		15:8	IC4BUF<15:8>							
		7:0	IC4BUF<7:0>							
BF80_2800	IC5CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	ICFEDGE	ICC32
		7:0	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>		
BF80_2804	IC5CONCLR	31:0	Write clears selected bits in IC5CON, read yields an undefined value							
BF80_2808	IC5CONSET	31:0	Write sets selected bits in IC5CON, read yields an undefined value							
BF80_280C	IC5CONINV	31:0	Write inverts selected bits in IC5CON, read yields an undefined value							
BF80_2810	IC5BUF	31:24	IC5BUF<31:24>							
		23:16	IC5BUF<23:16>							
		15:8	IC5BUF<15:8>							
		7:0	IC5BUF<7:0>							

REGISTER 15-1: ICxCON: INPUT CAPTURE x CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
ON	FRZ	SIDL	—	—	—	ICxFEDGE	ICxC32
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0
ICxTMR	ICxI<1:0>		ICxOV	ICxBNE	ICxM<2:0>		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** ON bit
1 = Module enabled
0 = Disable and Reset module, disable clocks, disable interrupt generation, and allow SFR modifications

bit 14 **FRZ:** Freeze in Debug Mode Control bit (read/write only in Debug mode; otherwise read as ‘0’)
1 = Freeze module operation when in Debug mode
0 = Do not freeze module operation when in Debug mode

bit 13 **SIDL:** Stop in Idle Control bit
1 = Halt in CPU Idle mode
0 = Continue to operate in CPU Idle mode

bit 12-10 Unimplemented: Read as ‘0’

bit 9 **ICxFEDGE:** First Capture Edge Select bit **(**only used in mode 6, ICxM = 110)
1 = Capture rising edge first
0 = Capture falling edge first

bit 8 **ICxC32:** 32-Bit Capture Select bit
1 = 32-Bit timer resource capture
0 = 16-Bit timer resource capture

bit 7 **ICxTMR:** Timer Select bit **(**Does not affect timer selection when ICxC32 (ICxCON<8>) is ‘1’)
0 = Timer3 is the counter source for capture
1 = Timer2 is the counter source for capture

bit 6-5 **ICxI<1:0>:** Interrupt Control bits
11 = Interrupt on every fourth capture event
10 = Interrupt on every third capture event
01 = Interrupt on every second capture event
00 = Interrupt on every capture event

REGISTER 15-1: ICxCON: INPUT CAPTURE x CONTROL REGISTER (CONTINUED)

bit 4	**ICxOV:** Input Capture Overflow Status Flag bit (read-only) 1 = Input capture overflow occurred 0 = No input capture overflow occurred
bit 3	**ICxBNE:** Input Capture Buffer Not Empty Status bit (read-only) 1 = Input capture buffer is not empty; at least one more capture value can be read 0 = Input capture buffer is empty
bit 2-0	**ICxM<2:0>:** Input Capture Mode Select bits 111 = Interrupt Only mode 110 = Simple Capture Event mode – every edge, specified edge first and every edge thereafter 101 = Prescaled Capture Event mode – every 16th rising edge 100 = Prescaled Capture Event mode – every 4th rising edge 011 = Simple Capture Event mode – every rising edge 010 = Simple Capture Event mode – every falling edge 001 = Edge Detect mode – every edge (rising and falling) 000 = Capture Disable mode

REGISTER 15-2: ICxBUF: INPUT CAPTURE x BUFFER REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
ICxBUF<31:24>							
bit 31							bit 24

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
ICxBUF<23:16>							
bit 23							bit 16

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
ICxBUF<15:8>							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
ICxBUF<7:0>							
bit 7							bit 0

Legend:	
R = Readable bit	W = Writable bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)

bit 31-0 **ICxBUF<31:0>:** Buffer Register bits
Value of the current captured input timer count

15.1 Timer Selection

The input capture module can select between one of two 16-bit timers for the time base, or two 16-bit timers together to form a 32-bit timer. Setting ICTMR (ICxCON<7>) to '0' selects the Timer3 for capture. Setting ICTMR (ICxCON<7>) to 1 selects the Timer2 for capture.

An input capture channel configured to support 32-bit capture, may use a 32-bit timer resource for capture. By setting ICC32 (ICxCON<8>) to '1', a 32-bit timer resource is captured. The 32-bit timer resource is routed into the module using the existing 16-bit timer inputs.

The timers clock can be setup using the internal peripheral clock source, or using a synchronized external clock source applied at the TxCK pin.

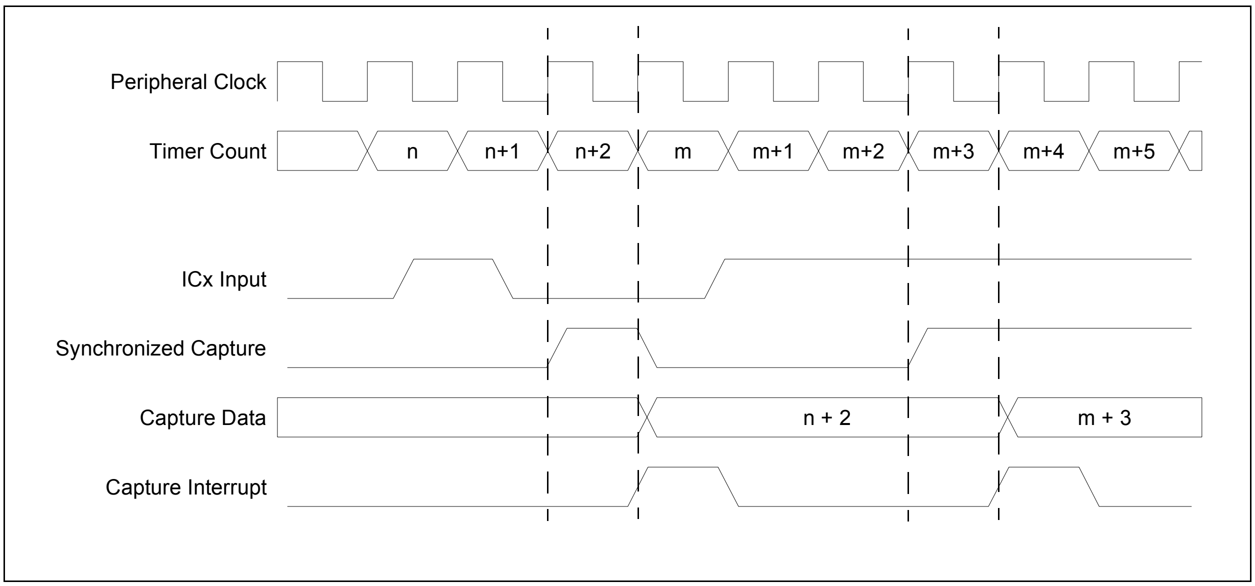
15.2 Simple Capture Event Modes

These modes are specified by setting the ICM (ICxCON<2:0>) bits to '010', '011', or '110'. Setting ICM = '011' configures the module to capture the timer value on any rising edge of the capture input. ICM = '010' configures the module to capture the timer on any falling edge of the capture input. Setting ICM = '110' configures the channel to capture the timer on every transition of the capture input, beginning with the edge specified by ICFEDGE (ICxCON<9>). In Simple Capture Event mode, the prescaler is not used. See Figure 15-2 for simplified timing diagrams of a simple capture event.

Note: Since the capture input must be synchronized to the peripheral clock, the module captures the timer count value, which is valid 2-3 peripheral clock cycles (TPB) after the capture event.

An input capture interrupt event is generated after one, two, three or four timer count captures, as configured by ICI (ICxCON<6:5>).

FIGURE 15-2: SIMPLE CAPTURE EVENT TIMING DIAGRAM CAPTURE EVERY RISING EDGE



15.3 Prescaled Capture Event Modes

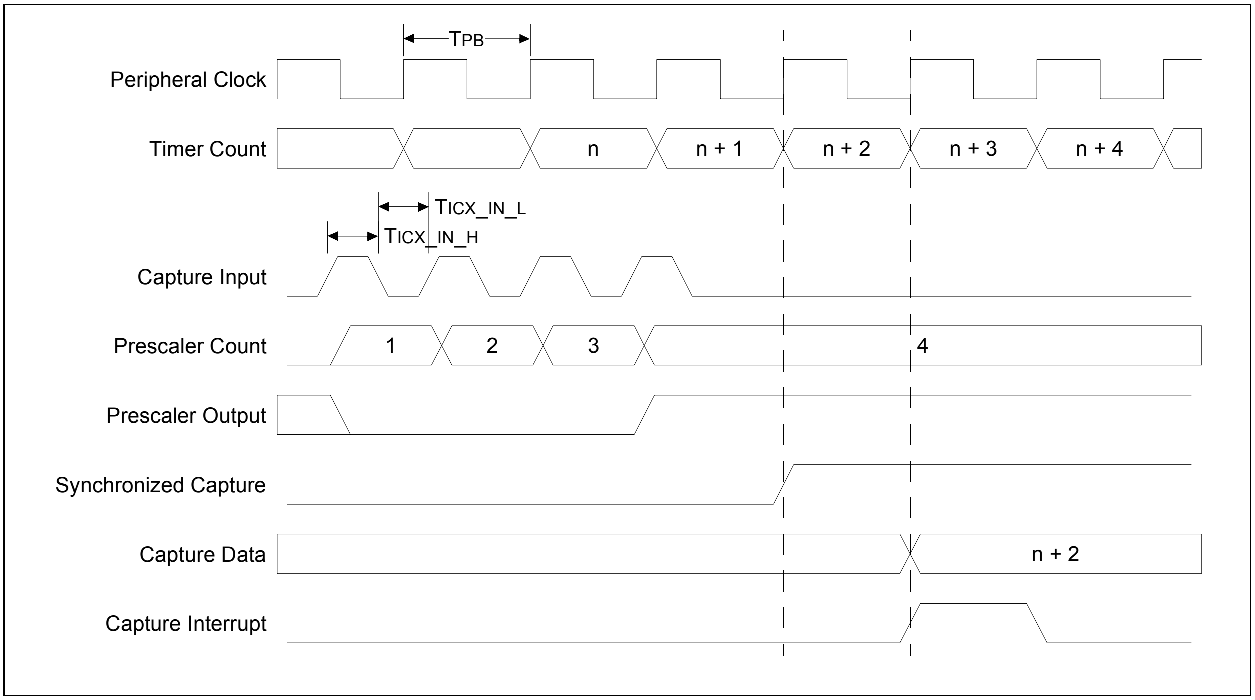
In Prescaled Capture Event mode, the input capture module triggers a capture event on either every fourth or every sixteenth rising edge. These modes are selected by setting the ICM (ICxCON<2:0>) bits to '100' or '101', respectively.

Note: Since the capture input must be synchronized to the peripheral clock, the timer count value is captured, which is valid 2-3 peripheral clock periods after the capture event.

Note:	It is recommended that the user disable (i.e., clear ON bit, ICxCON<15>) the capture module before switching to Prescaler Capture Event mode. Simply switching to Prescaler Capture Event mode from another active mode does not reset the prescaler and may cause an inadvertent capture event.

Figure 15-3 depicts a capture event when the input capture module is in Prescaler Capture Event mode.

FIGURE 15-3: PRESCALER CAPTURE EVENT TIMING DIAGRAM



15.4 Edge Detect (Hall Sensor) Mode

In Edge Detect mode, the input capture module captures a timer count value on every edge of the capture input. Edge Detect mode is selected by setting the ICM bit to '`001`'. In this mode, the capture prescaler is not used and the capture overflow bit, ICOV (ICxCON<4>) is not updated. In this mode, the Interrupt Control bits (ICI, ICxCON<6:5>) are ignored and an interrupt event is generated for every timer count capture

15.5 Interrupt Only Mode

When the Input Capture module is set for Interrupt Only mode (ICM = '`111`') and the device is in Sleep or Idle mode, the capture input functions as an interrupt pin. Any rising edge on the capture input triggers an interrupt. No timer values are captured and the FIFO buffer is not updated. When the device leaves Sleep or Idle mode, the interrupt signal is deasserted.

In this mode, since no timer values are captured, the Timer Select bit ICTMR (ICxCON<7>) is ignored and there is no need to configure the timer source. A wake-up interrupt is generated on the first rising edge, therefore the Interrupt Control bits ICI (ICxCON<6:5>) are also ignored. The prescaler is not used in this mode.

EXAMPLE 15-1: INPUT CAPTURE EXAMPLE CODE

```
/*
The following code segment initialized the timer and setup the input capture
module.
*/
    ...
    //Initialize timer 2
    T2CON = 0x0       // Stop and Init Timer
    TMR2 = 0x0;       // Clear timer register
    PR2 = 0x7000;     // Load period register
    T2CONSET = 0x8000;// Start Timer

    // Init IC1 module
    IC1CON = 0x8081;//Enable Module, use timer 2,
                    //Capture mode 1 (capture every edge)
    ...
    //Read the capture data if available
    int cap_data;
    while( IC1CONbits.ICBNE ) // while data available in capture FIFO
    {
        cap_data = IC1BUF;
        ... //process data
    }
    ...
```

 Advance Information

16.0 OUTPUT COMPARE

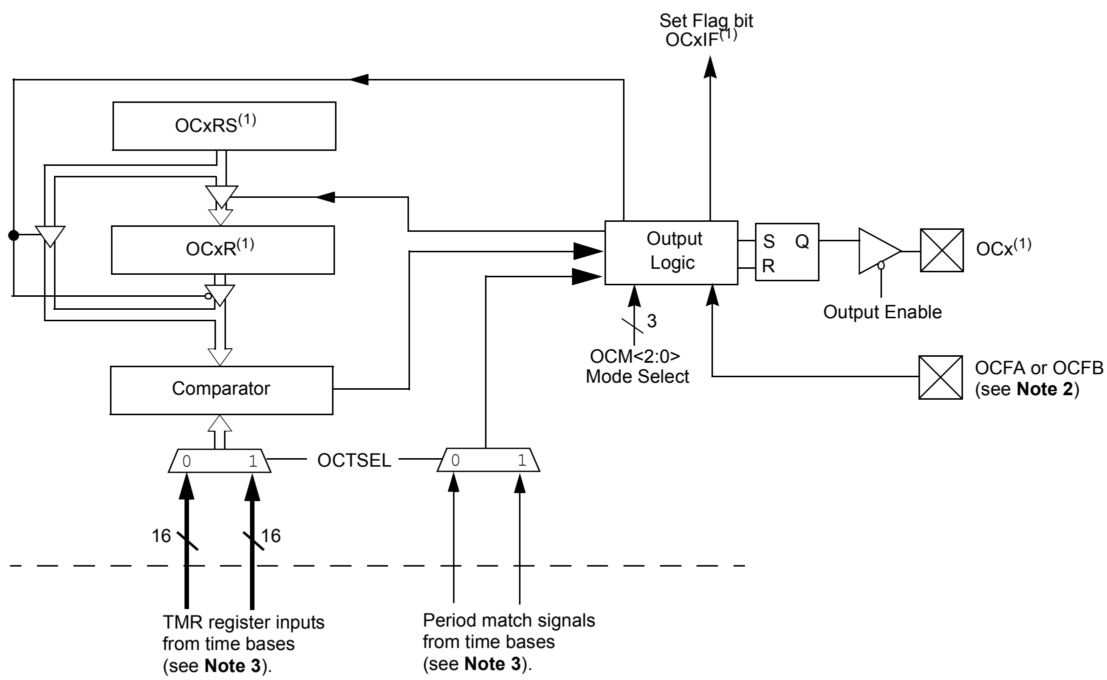
Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

The Output Compare module (OCMP) is used to generate a single pulse or a train of pulses in response to selected time base events. For all modes of operation, the OCMP module compares the values stored in the OCxR and/or the OCxRS registers to the value in the selected timer. When a match occurs, the OCMP module generates an event based on the selected mode of operation.

The following are some of the key features:

- Multiple output compare modules in a device
- Programmable interrupt generation on compare event
- Single and Dual Compare modes
- Single and continuous output pulse generation
- Pulse-Width Modulation (PWM) mode
- Hardware-based PWM Fault detection and automatic output disable
- Programmable selection of 16-bit or 32-bit time bases.
- Can operate from either of two available 16-bit time bases or a single 32-bit time base.

FIGURE 16-1: OUTPUT COMPARE MODULE BLOCK DIAGRAM



Note 1: Where 'x' is shown, reference is made to the registers associated with the respective output compare channels 1 through 5.

2: The OCFA pin controls the OC1-OC3 channels. The OCFB pin controls the OC4-OC5 channels.

3: Each output compare channel can use one of two selectable 16-bit time bases or a single 32-bit timer base.

TABLE 16-1: OUTPUT COMPARE SFR SUMMARY

Virtual Address	Name		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
BF80_3000	OC1CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	—	—
		7:0	—	—	OC32	OCFLT	OCTSEL	OCM<2:0>		
BF80_3004	OC1CONCLR	31:0	Write clears selected bits in OC1CON, Read yields an undefined value.							
BF80_3008	OC1CONSET	31:0	Write sets selected bits in OC1CON, Read yields an undefined value.							
BF80_300C	OC1CONINV	31:0	Write inverts selected bits in OC1CON, Read yields an undefined value.							
BF80_3010	OC1R	31:24	OC1R<31:24>							
		23:16	OC1R<23:16>							
		15:8	OC1R<15:8>							
		7:0	OC1R<7:0>							
BF80_3014	OC1RCLR	31:0	Write clears selected bits in OC1R, Read yields an undefined value.							
BF80_3018	OC1RSET	31:0	Write sets selected bits in OC1R, Read yields an undefined value.							
BF80_301C	OC1RINV	31:0	Write inverts selected bits in OC1R, Read yields an undefined value.							
BF80_3020	OC1RS	31:24	OC1RS<31:24>							
		23:16	OC1RS<23:16>							
		15:8	OC1RS<15:8>							
		7:0	OC1RS<7:0>							
BF80_3024	OC1RSCLR	31:0	Write clears selected bits in OC1RS, Read yields an undefined value.							
BF80_3028	OC1RSSET	31:0	Write sets selected bits in OC1RS, Read yields an undefined value.							
BF80_302C	OC1RSINV	31:0	Write inverts selected bits in OC1RS, Read yields an undefined value.							
BF80_3200	OC2CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	—	—
		7:0	—	—	OC32	OCFLT	OCTSEL	OCM<2:0>		
BF80_3204	OC2CONCLR	31:0	Write clears selected bits in OC2CON, Read yields an undefined value.							
BF80_3208	OC2CONSET	31:0	Write sets selected bits in OC2CON, Read yields an undefined value.							
BF80_320C	OC2CONINV	31:0	Write inverts selected bits in OC2CON, Read yields an undefined value.							
BF80_3210	OC2R	31:24	OC2R<31:24>							
		23:16	OC2R<23:16>							
		15:8	OC2R<15:8>							
		7:0	OC2R<7:0>							
BF80_3214	OC2RCLR	31:0	Write clears selected bits in OC2R, Read yields an undefined value.							
BF80_3218	OC2RSET	31:0	Write sets selected bits in OC2R, Read yields an undefined value.							
BF80_321C	OC2RINV	31:0	Write inverts selected bits in OC2R, Read yields an undefined value.							
BF80_3220	OC2RS	31:24	OC2RS<31:24>							
		23:16	OC2RS<23:16>							
		15:8	OC2RS<15:8>							
		7:0	OC2RS<7:0>							
BF80_3224	OC2RSCLR	31:0	Write clears selected bits in OC2RS, Read yields an undefined value.							
BF80_3228	OC2RSSET	31:0	Write sets selected bits in OC2RS, Read yields an undefined value.							
BF80_322C	OC2RSINV	31:0	Write inverts selected bits in OC2RS, Read yields an undefined value.							
BF80_3400	OC3CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	—	—
		7:0	—	—	OC32	OCFLT	OCTSEL	OCM<2:0>		
BF80_3404	OC3CONCLR	31:0	Write clears selected bits in OC3CON, Read yields an undefined value.							
BF80_3408	OC3CONSET	31:0	Write sets selected bits in OC3CON, Read yields an undefined value.							
BF80_340C	OC3CONINV	31:0	Write inverts selected bits in OC3CON, Read yields an undefined value.							
BF80_3410	OC3R	31:24	OC3R<31:24>							
		23:16	OC3R<23:16>							
		15:8	OC3R<15:8>							
		7:0	OC3R<7:0>							
BF80_3414	OC3RCLR	31:0	Write clears selected bits in OC3R, Read yields an undefined value.							
BF80_3418	OC3RSET	31:0	Write sets selected bits in OC3R, Read yields an undefined value.							
BF80_341C	OC3RINV	31:0	Write inverts selected bits in OC3R, Read yields an undefined value.							

TABLE 16-1: OUTPUT COMPARE SFR SUMMARY (CONTINUED)

Virtual Address	Name		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
BF80_3420	OC3RS	31:24	OC3RS<31:24>							
		23:16	OC3RS<23:16>							
		15:8	OC3RS<15:8>							
		7:0	OC3RS<7:0>							
BF80_3424	OC3RSCLR	31:0	Write clears selected bits in OC3RS, Read yields an undefined value							
BF80_3428	OC3RSSET	31:0	Write sets selected bits in OC3RS, Read yields an undefined value							
BF80_342C	OC3RSINV	31:0	Write inverts selected bits in OC3RS, Read yields an undefined value							
BF80_3600	OC4CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	—	—
		7:0	—	—	OC32	OCFLT	OCTSEL	OCM<2:0>		
BF80_3604	OC4CONCLR	31:0	Write clears selected bits in OC4CON, read yields an undefined value							
BF80_3608	OC4CONSET	31:0	Write sets selected bits in OC4CON, read yields an undefined value							
BF80_360C	OC4CONINV	31:0	Write inverts selected bits in OC4CON, read yields an undefined value							
BF80_3610	OC4R	31:24	OC4R<31:24>							
		23:16	OC4R<23:16>							
		15:8	OC4R<15:8>							
		7:0	OC4R<7:0>							
BF80_3614	OC4RCLR	31:0	Write clears selected bits in OC4R, read yields an undefined value							
BF80_3618	OC4RSET	31:0	Write sets selected bits in OC4R, read yields an undefined value							
BF80_361C	OC4RINV	31:0	Write inverts selected bits in OC4R, read yields an undefined value							
BF80_3620	OC4RS	31:24	OC4RS<31:24>							
		23:16	OC4RS<23:16>							
		15:8	OC4RS<15:8>							
		7:0	OC4RS<7:0>							
BF80_3624	OC4RSCLR	31:0	Write clears selected bits in OC4RS, read yields an undefined value							
BF80_3628	OC4RSSET	31:0	Write sets selected bits in OC4RS, read yields an undefined value							
BF80_362C	OC4RSINV	31:0	Write inverts selected bits in OC4RS, read yields an undefined value							
BF80_3800	OC5CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	—	—
		7:0	—	—	OC32	OCFLT	OCTSEL	OCM<2:0>		
BF80_3804	OC5CONCLR	31:0	Write clears selected bits in OC5CON, read yields an undefined value							
BF80_3808	OC5CONSET	31:0	Write sets selected bits in OC5CON, read yields an undefined value							
BF80_380C	OC5CONINV	31:0	Write inverts selected bits in OC5CON, read yields an undefined value							
BF80_3810	OC5R	31:24	OC5R<31:24>							
		23:16	OC5R<23:16>							
		15:8	OC5R<15:8>							
		7:0	OC5R<7:0>							
BF80_3814	OC5RCLR	31:0	Write clears selected bits in OC5R, read yields an undefined value							
BF80_3818	OC5RSET	31:0	Write sets selected bits in OC5R, read yields an undefined value							
BF80_381C	OC5RINV	31:0	Write inverts selected bits in OC5R, read yields an undefined value							
BF80_3820	OC5RS	31:24	OC5RS<31:24>							
		23:16	OC5RS<23:16>							
		15:8	OC5RS<15:8>							
		7:0	OC5RS<7:0>							
BF80_3824	OC5RSCLR	31:0	Write clears selected bits in OC5RS, read yields an undefined value							
BF80_3828	OC5RSSET	31:0	Write sets selected bits in OC5RS, read yields an undefined value							
BF80_382C	OC5RSINV	31:0	Write inverts selected bits in OC5RS, read yields an undefined value							
0xBF881000	INTCON	31:24	IPTMR<31:24>							
		23:16	IPTMR<23:16>							
		15:8	—	FRZ	—	—	IPRST	TPC[2:0>		
		7:0	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
0xBF881010	IFS0	31:24	I2CMIF	I2CSIF	U1EIF	U1TXIF	U1RXIF	SPI1RXIF	SPI1TXIF	SPI1EIF
		23:16	CNIF	OC5IF	IC5IF	T5IF	INT4IF	OC4IF	IC4IF	T4IF
		15:8	INT3IF	OC3IF	IC3IF	T3IF	INT2IF	OC2IF	IC3IF	T2IF
		7:0	INT1IF	OC1IF	IC1IF	T1IF	INT0IF	CS1IF	CS0IF	CTIF

TABLE 16-1: OUTPUT COMPARE SFR SUMMARY (CONTINUED)

Virtual Address	Name		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
0xBF881040	IEC0	31:24	I2CMIE	I2CSIE	U1EIE	U1TXIE	U1RXIE	SPI1RXIE	SPI1TXIE	SPI1EIE
		23:16	CNIE	OC5IE	IC5IE	T5IE	INT4IE	OC4IE	IC4IE	T4IE
		15:8	INT3IE	OC3IE	IC3IE	T3IE	INT2IE	OC2IE	IC3IE	T2IE
		7:0	INT1IE	OC1IE	IC1IE	T1IE	INT0IE	CS1IE	CSOIE	CTIE
0xBF881080	IPC1	31:24	—	—	—	INT1IP<2:0>			INT1IS<1:0>	
		23:16	—	—	—	OC1IP<2:0>			OC1IS<1:0>	
		15:8	—	—	—	IC1IP<2:0>			IC1IS<1:0>	
		7:0	—	—	—	T1IP<2:0>			T1IS<1:0>	
0xBF881090	IPC2	31:24	—	—	—	INT2IP<2:0>			INT2IS<1:0>	
		23:16	—	—	—	OC2IP<2:0>			OC2IS<1:0>	
		15:8	—	—	—	IC2IP<2:0>			IC2IS<1:0>	
		7:0	—	—	—	T2IP<2:0>			T2IS<1:0>	
0xBF8810A0	IPC3	31:24	—	—	—	INT3IP<2:0>			INT3IS<1:0>	
		23:16	—	—	—	OC3IP<2:0>			OC3IS<1:0>	
		15:8	—	—	—	IC3IP<2:0>			IC3IS<1:0>	
		7:0	—	—	—	T3IP<2:0>			T3IS<1:0>	
0xBF8810B0	IPC4	31:24	—	—	—	INT4IP<2:0>			INT4IS<1:0>	
		23:16	—	—	—	OC4IP<2:0>			OC4IS<1:0>	
		15:8	—	—	—	IC4IP<2:0>			IC4IS<1:0>	
		7:0	—	—	—	T4IP<2:0>			T4IS<1:0>	
0xBF8810C0	IPC5	31:24	—	—	—	CNIP<2:0>			CNIS<1:0>	
		23:16	—	—	—	OC5IP<2:0>			OC5IS<1:0>	
		15:8	—	—	—	IC5IP<2:0>			IC5IS<1:0>	
		7:0	—	—	—	T5IP<2:0>			T5IS<1:0>	
BF80_0800	T2CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	—	—
		7:0	TGATE	TCKPS<2:0>			T32	—	TCS	—
BF80_0A00	T3CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	—	—	—
		7:0	TGATE	TCKPS<2:0>			—	—	TCS	—
BF80_0820	PR2	31:24	PR2<31:24>							
		23:16	PR2<23:16>							
		15:8	PR2<15:8>							
		7:0	PR2<7:0>							
BF80_0A20	PR3	31:24	PR3<31:24>							
		23:16	PR3<23:16>							
		15:8	PR3<15:8>							
		7:0	PR3<7:0>							

REGISTER 16-1: OCxCON: OUTPUT COMPARE x CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
ON	FRZ	SIDL	—	—	—	—	—
bit 15							bit 8

U-0	U-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	OC32	OCFLT	OCTSEL	OCM<2:0>		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** Output Compare Peripheral On bit
1 = Output compare peripheral is enabled. The status of other bits in the register are not affected by setting this bit
0 = Output compare peripheral is disabled and not drawing current. SFR modifications are allowed. The status of other bits in this register are not affected by clearing this bit

bit 14 **FRZ:** Freeze in Debug Exception Mode bit(1)
1 = Freeze operation when CPU enters in Debug Exception mode
0 = Continue operation when CPU enters in Debug Exception mode

bit 13 **SIDL:** Stop in Idle Mode bit
1 = Discontinue operation when CPU enters in Idle mode
0 = Continue operation in Idle mode

bit 12-6 **Unimplemented:** Read as ‘0’

bit 5 **OC32:** 32-Bit Compare Mode bit
1 = OCxR<31:0> and/or OCxRS<31:0> are used for comparisions to the 32-bit timer source
0 = OCxR<15:0> and OCxRS<15:0> are used for comparisons to the 16-bit timer source

bit 4 **OCFLT:** PWM Fault Condition Status bit(2)
1 = PWM Fault condition has occurred (cleared in HW only)
0 = No PWM Fault condition has occurred

Note: (This bit is only used when OCM<2:0> = 111.

bit 3 **OCTSEL:** Output Compare Timer Select bit
1 = Timer3 is the clock source for compare x
0 = Timer2 is the clock source for compare x

Note: OCTSEL must be set to ‘1’ when using 32-bit mode (OC32 = 1)

Note 1: FRZ is writable in Debug Exception mode only, it is forced to read ‘0’ in Normal mode.
2: Reads as ‘0’ in modes other than PWM mode.

bit 2-0 **OCM<2:0>:** Output Compare Mode Select bits
111 = PWM mode on OCx, Fault pin enabled
110 = PWM mode on OCx, Fault pin disabled
101 = Initialize OCx pin low, generate continuous output pulses on OCx pin
100 = Initialize OCx pin low, generate single output pulse on OCx pin
011 = Compare event toggles OCx pin
010 = Initialize OCx pin high, compare event forces OCx pin low
001 = Initialize OCx pin low, compare event forces OCx pin high
000 = Output compare peripheral is disabled

Note 1: FRZ is writable in Debug Exception mode only, it is forced to read ‘0’ in Normal mode.
2: Reads as ‘0’ in modes other than PWM mode.

Register 16-1: OCxR: OUTPUT COMPARE x COMPARE PRIMARY REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
OCR<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
OCR<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
OCR<15>8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
OCR<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **OCxR<31:16>:** Upper 16 bits of 32-bit compare value when OC32 (OCxCON<5>) = `1`

bit 15-0 **OCxR<15:0>:** Lower 16 bits of 32-bit compare value or entire 16 bits of 16-bit compare value

Register 16-2: OCxRS: OUTPUT COMPARE x COMPARE SECONDARY REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
OCRS<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
OCRS<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
OCRS<5>8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
OCRS<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **OCxRS<31:16>:** Upper 16 bits of 32-bit compare value, when OC32 (OCxCON<5>) = 1

bit 15-0 **OCxRS<15:0>:** Lower 16 bits of 32-bit compare value or entire 16 bits of 16-bit compare value

16.1 Setup for Single Output Change

There are three modes of operation that change the state of the output pin; these modes can be referred to as drive high, drive low and toggle. The configuration for these modes is identical, the mode is selected by the OCM bits. For this example, Tx will represent Timer2.

Drive High: When the OCM control bits (OCxCON<2:0>) are set to '`001`', the selected output compare channel initializes the OCx pin to the low state and drives the output pin high when a compare event occurs.

Drive Low: When the OCM control bits (OCxCON<2:0>) are set to '`010`', the selected output compare channel initializes the OCx pin to the high state and drives the output pin low when a compare event occurs.

Toggle: When the OCM control bits (OCxCON<2:0>) are set to '`011`', the selected output compare channel OCx pin is not initialized. The OCx pin is driven to the opposite state when a compare event occurs.

To generate a output change signal, the following steps are required (these steps assume the timer source is initially turned off, but this is not a requirement for the module operation):

1. Determine the timer clock cycle time. Take into account the frequency of the external clock to the timer source (if one is used) and the timer prescaler settings.
2. Calculate time to the rising edge of the output pulse relative to the timer start value (0000h).
3. Determine if the output compare module will be used in 16 or 32-bit mode based on the previous calculations.
4. Configure the timer to be used as the time base for 16 or 32-bit mode by writing to the T32 bit (TxCON<T32>).
5. Configure the output compare channel for 16 or 32-bit operation by writing to the OC32 bit (OCxCON<5>).
6. Write the value computed in step 2 above into the Compare register, OCxR.
7. Set Timer Period register, PRx, to the value equal to or greater than the value in OCxRS, the Secondary Compare register.
8. Set the OCM bits to the desired mode of operation and the OCTSEL (OCxCON<3>) bit to the desired timer source. The OCx pin state will now be driven low.
9. Set the ON (TxCON<15>) bit to '`1`' which enables the compare time base to count.
10. Upon the first match between TMRx and OCxR, the OCx pin will be driven high.
11. When the incrementing timer, TMRx, matches the Secondary Compare register, OCxRS, the second and trailing edge (high-to-low) of the pulse is driven onto the OCx pin. No additional pulses are driven onto the OCx pin and it remains at low. As a result of the second compare match event, the OCxIF interrupt flag bit is set, which will result in an interrupt if it is enabled, by setting the OCxIE bit. For further information on peripheral interrupts, refer to **Section 9.0 "Interrupts"**.
12. To initiate another single pulse output, change the Timer and Compare register settings, if needed, and then issue a write to set the OCM bits to the desired mode of operation. Disabling and re-enabling of the timer and clearing the Timer register are not required, but may be advantageous for defining a pulse from a known event time boundary.

16.2 Setup for Single Output Pulse Generation

When the OCM control bits (OCxCON<2:0>) are set to '`100`', the selected output compare channel initializes the OCx pin to the low state and generates a single output pulse.

To generate a single output pulse, the following steps are required (these steps assume the timer source is initially turned off, but this is not a requirement for the module operation): For this example Tx will represent Timer2.

1. Determine the timer clock cycle time. Take into account the frequency of the external clock to the timer source (if one is used) and the timer prescaler settings.
2. Calculate time to the rising edge of the output pulse relative to the timer start value (0000h).
3. Calculate the time to the falling edge of the pulse based on the desired pulse width and the time to the rising edge of the pulse.
4. Determine if the output compare module will be used in 16 or 32-bit mode based on the previous calculations.
5. Configure the timer to be used as the time base for 16 or 32-bit mode by writing to the T32 bit (TxCON<T32>).
6. Configure the output compare channel for 16 or 32-bit operation by writing to the OC32 bit (OCxCON<5>).
7. Write the values computed in steps 2 and 3 above into the Compare register, OCxR, and the Secondary Compare register, OCxRS, respectively.
8. Set Timer Period register, PRx, to the value equal to or greater than the value in the OCxRS, the Secondary Compare register.

9. Set the OCM bits to '100' and the OCTSEL (OCxCON<3>) bit to the desired timer source. The OCx pin state will now be driven low.
10. Set the ON (TxCON<15>) bit to '1' which enables the compare time base to count.
11. Upon the first match between TMRx and OCxR, the OCx pin will be driven high.
12. When the incrementing timer matches the Secondary Compare register, OCxRS, the second and trailing edge (high-to-low) of the pulse is driven onto the OCx pin. No additional pulses are driven onto the OCx pin and it remains at low. As a result of the second compare match event, the OCxIF interrupt flag bit is set, which will result in an interrupt if it is enabled, by setting the OCxIE bit. For further information on peripheral interrupts, refer to **Section 9.0 "Interrupts"**.
13. To initiate another single pulse output, change the Timer and Compare register settings, if needed, and then issue a write to set the OCM bits to '100'. Disabling and re-enabling of the timer and clearing the TMRx register are not required, but may be advantageous for defining a pulse from a known event time boundary.

EXAMPLE 16-1: EXAMPLE CODE

```
//  The following code example will set the Output Compare 1 module
//  for interrupts on the single pulse event and select Timer 2
//  as the clock source for the compare time base.

    T2CON = 0x0010;              // Configure Timer 2 for a prescaler of 2

    OC1CON = 0x0000;             // Turn off OC1 while doing setup.
    OC1CON = 0x0004;             // Configure for single pulse mode
    OC1R = 0x3000;               // Initialize primary Compare Register
    OC1RS = 0x3003;              // Initialize secondary Compare Register
    PR2 = 0x3003;                // Set period (PR2 is now 32-bits wide)

                                 // configure int
    IF0CLR = 0x00000080;         // Clear the OC1 interrupt flag
    IE0SET = 0x00000080;         // Enable OC1 interrupt
    IPC1SET = 0x0030000;         // Set OC1 interrupt subpriority to 3,
                                 // the highest level
    IPC1SET = 0x00000003;        // Set subpriority to 3, maximum

    T2CONSET = 0x8000;           // Enable timer2
    OC1CONSET = 0x8000;          // Enable the OC1 module


    // Example code for Output Compare 1 ISR:

#pragma interupt OC1IntHandler ipl4 vector 6
void CmpIntHandler(void)
{
    // insert user code here
    IFS0CLR = 0x00000080;  // Clear the OC1 interrupt flag
}
```


16.3 Setup for Continuous Output Pulse Generation

When the OCM control bits (OCxCON<2:0>) are set to '101', the selected output compare channel initializes the OCx pin to the low state and generates output pulses on each and every compare match event.

For the user to configure the module for the generation of a continuous stream of output pulses, the following steps are required (these steps assume the timer source is initially turned off, but this is not a requirement for the module operation). For this example, Tx will represent Timer2.

1. Determine the timer clock cycle time. Take into account the frequency of the external clock to the timer source (if one is used) and the timer prescaler settings.
2. Calculate time to the rising edge of the output pulse relative to the TMRx start value (0000h).
3. Calculate the time to the falling edge of the pulse based on the desired pulse width and the time to the rising edge of the pulse.
4. Determine if the output compare module will be used in 16 or 32-bit mode based on the previous calculations.
5. Configure the timer to be used as the time base for 16 or 32-bit mode by writing to the T32 bit (TxCON<T32>).
6. Configure the output compare channel for 16 or 32-bit operation by writing to the OC32 bit (OCxCON<5>).
7. Write the values computed in step 2 and 3 above into the Compare register, OCxR, and the Secondary Compare register, OCxRS, respectively.
8. Set Timer Period register, PRx, to the value equal to or greater than the value in OCxRS, the Secondary Compare register.
9. Set the OCM bits to '101' and the OCTSEL bit to the desired timer source. The OCx pin state will now be driven low.
10. Enable the compare time base by setting the ON (TxCON<15>) bit to '1'.
11. Upon the first match between TMRx and OCxR, the OCx pin will be driven high.
12. When the compare time base, TMRy, matches the Secondary Compare register, OCxRS, the second and trailing edge (high-to-low) of the pulse is driven onto the OCx pin.
13. As a result of the second compare match event, the OCxIF interrupt flag bit set.
14. When the compare time base and the value in its respective Period register match, the TMRx register resets to 0x0000 and resumes counting.
15. Steps 8 through 11 are repeated and a continuous stream of pulses is generated, indefinitely. The OCxIF flag is set on each OCxRS-TMRx compare match event.

16.4 Pulse-Width Modulation Mode

There are two modes of PWM operation for this device: PWM and PWM with Fault input. The configuration of both modes is identical with the exception of the value written to the OCM bits to select the desired mode.

The following steps should be taken when configuring the output compare module for PWM operation:

1. Calculate the PWM period.
2. Calculate the PWM duty cycle.
3. Determine if the Output Compare module will be used in 16 or 32-bit mode based on the previous calculations.
4. Configure the timer to be used as the time base for 16 or 32-bit mode by writing to the T32 bit (TxCON<T32>).
5. Configure the output compare channel for 16 or 32-bit operation by writing to the OC32 bit (OCxCON<5>).
6. Set the PWM period by writing to the selected Timer Period register (PR).
7. Set the PWM duty cycle by writing to the OCxRS register.
8. Write the OCxR register with the initial duty cycle.
9. Enable interrupts, if required, for the timer and output compare modules. The output compare interrupt is required for PWM Fault pin utilization.
10. Configure the output compare module for one of two PWM operation modes by writing to the Output Compare mode bits OCM<2:0> (OCxCON<2:0>).
11. Set the TMRx prescale value and enable the time base by setting ON (TxCON<15>) = 1.

Note:	The OCxR register should be initialized before the output compare module is first enabled. The OCxR register becomes a read-only Duty Cycle register when the module is operated in the PWM modes. The value held in OCxR will become the PWM duty cycle for the first PWM period. The contents of the Duty Cycle Buffer register, OCxRS, will not be transferred into OCxR until a time base period match occurs.

16.4.1 PWM PERIOD

The PWM period is specified by writing to PR, the Timer Period register. The PWM period can be calculated using Equation 16-1.

EQUATION 16-1: CALCULATING THE PWM PERIOD

$$\text{PWM Period} = [(\text{PRy}) + 1> \cdot T_{PB} \cdot (\text{TMRy Prescale Value})]$$

$$\text{PWM Frequency} = 1/[\text{PWM Period}]$$

> **Note:** A PRy value of N will produce a PWM period of N + 1 time base count cycles. For example, a value of 7 written into the PRy register will yield a period consisting of 8 time base cycles.

16.4.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the OCxRS register. The OCxRS register can be written to at any time, but the duty cycle value is not latched into OCxR until a match between the PR and timer occurs (i.e., the period is complete). This provides a double buffer for the PWM duty cycle and is essential for glitch-less PWM operation. In the PWM mode, OCxR is a read-only register.

Some important boundary parameters of the PWM duty cycle include:

- If the Duty Cycle register, OCxR, is loaded with 0000h, the OCx pin will remain low (0% duty cycle).
- If OCxR is greater than PR (Timer Period register), the pin will remain high (100% duty cycle).
- If OCxR is equal to PR, the OCx pin will be low for one time base count value and high for all other count values.

See Example 16-2 for PWM mode timing details. Table 16-2 shows example PWM frequencies and resolutions for a device peripheral bus operating at 10 MHz.

EQUATION 16-2: CALCULATION FOR MAXIMUM PWM RESOLUTION

$$\text{Maximum PWM Resolution (bits)} = \frac{\log_{10}\left(\frac{F_{PB}}{F_{PWM} \cdot \text{TMRy} \cdot \text{Prescaler}}\right)}{\log_{10}(2)} \text{bits}$$

EXAMPLE 16-2: PWM PERIOD AND DUTY CYCLE CALCULATION

Desired PWM frequency is 52.08 kHz,
F_{PB} = 10 MHz
Timer 2 prescale setting: 1:1

$$1/52.08\text{ kHz} = (\text{PR2} + 1) \cdot F_{BP} \cdot (\text{Timer2 prescale value})$$
$$19.20\ \mu s = (\text{PR2} + 1) \cdot 0.1\ \mu s \cdot (1)$$
$$\text{PR2} = 191$$

Find the maximum resolution of the duty cycle that can be used with a 52.08 kHz PWM frequency and a 10 MHz peripheral bus clock rate.

$$1/52.08\text{ kHz} = 2^{\text{PWM RESOLUTION}} \cdot 1/10\text{ MHz} \cdot 1$$
$$19.20\ \mu s = 2^{\text{PWM RESOLUTION}} \cdot 100\text{ ns} \cdot 1$$
$$192 = 2^{\text{PWM RESOLUTION}}$$
$$\log_{10}(192) = (\text{PWM Resolution}) \cdot \log_{10}(2)$$
$$\text{PWM Resolution} = 7.6\text{ bits}$$

> **Note:** If the PR value exceeds 16-bits the module must be used in 32-bit mode to maintain the calculated PWM resolution. If reduced resolution is acceptable the Timer prescaler may be increased and the calculation repeated until the result is a 16-bit value. Increasing the Timer prescaler to allow operation in 16-bit mode may result in reduced PWM resolution.

EXAMPLE 16-3: PWM MODE PULSE SETUP AND INTERRUPT SERVICING (32-BIT MODE)

```
//  The following code example will set the Output Compare 1 module
//  for PWM mode with FAULT pin disabled, a 50% duty cycle and a
//  PWM frequency of 52.08 kHz at FPB = 40 MHz. Timer2 is selected as
//  the clock for the PWM time base and Output Compare 1 interrupts
//  are enabled.

    OC1CON = 0x0000;            // Turn off OC1 while doing setup.
    OC1R = 0x00600000;          // Initialize primary Compare Register
    OC1RS = 0x00600000;         // Initialize secondary Compare Register
    OC1CON = 0x0006;            // Configure for PWM mode, Fault pin Disabled
    PR2 = 0x00600000;           // Set period

                                // configure int
    IFS0 &= ~0x00000080;        // Clear the OC1 interrupt flag
    IEC0 |= 0x00000080;         // Enable OC1 interrupt
    IPC1 |= 0x001C0000;         // Set OC1 interrupt priority to 7,
                                // the highest level
    IPC1 |= 0x00000003;         // Set subpriority to 3, maximum

    T2CON |= 0x8000;            // Enable timer2
    OC1CON |= 0x8000;           // turn on OC1 module


// Example code for Output Compare 1 ISR:
#pragma interupt OC1IntHandler ipl4 vector 36
void OC1IntHandler(void)
{
               // insert user code here
                   IFS0CLR = 0x00000080;  // Clear the interrupt flag
}
```

TABLE 16-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS WITH PERIPHERAL BUS CLOCK OF 10 MHZ (16-BIT MODE)

PWM Frequency	19.5 Hz	153 Hz	305 Hz	2.44 kHz	9.77 kHz	78.1 kHz	313 kHz
Timer Prescaler Ratio	8	1	1	1	1	1	1
Period Register Value (hex)	0xFA65	0xFF4E	0x8011	0x1001	0x03FE	0x007F	0x001E
Resolution (bits) (decimal)	16	16	15	12	10	7	5

TABLE 16-3: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS WITH PERIPHERAL BUS CLOCK OF 30 MHZ (16-BIT MODE)

PWM Frequency	58 Hz	458 Hz	916 Hz	7.32 kHz	29.3 kHz	234 kHz	938 kHz
Timer Prescaler Ratio	8	1	1	1	1	1	1
Period Register Value (hex)	0xFC8E	0xFFDD	0x7FEE	0x1001	0x03FE	0x007F	0x001E
Resolution (bits) (decimal)	16	16	15	12	10	7	5

TABLE 16-4: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS WITH PERIPHERAL BUS CLOCK OF 50 MHZ (16-BIT MODE)

PWM Frequency	58 Hz	458 Hz	916 Hz	7.32 kHz	29.3 kHz	234 kHz	938 kHz
Timer Prescaler Ratio	64	8	1	1	1	1	1
Period Register Value (hex)	0x349C	0x354D	0xD538	0x1AAD	0x06A9	0x00D4	0x0034
Resolution (bits) (decimal)	13.7	13.7	15.7	12.7	10.7	7.7	5.7

TABLE 16-5: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS WITH PERIPHERAL BUS CLOCK OF 50 MHZ (16-BIT MODE)

PWM Frequency	100 Hz	200 Hz	500 Hz	1 kHz	2 kHz	5 kHz	10 kHz
Timer Prescaler Ratio	8	8	8	1	8	1	1
Period Register Value (hex)	0xF423	0x7A11	0x30D3	0xC34F	0x0C34	0x270F	0x1387
Resolution (bits) (decimal)	15.9	14.9	13.6	15.6	11.6	13.3	12.3

TABLE 16-6: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS WITH PERIPHERAL BUS CLOCK OF 50 MHZ (16-BIT MODE)

PWM Frequency	100 Hz	200 Hz	500 Hz	1 kHz	2 kHz	5 kHz	10 kHz
Timer Prescaler Ratio	8	4	2	1	1	1	1
Period Register Value (hex)	0xF423	0xF423	0xC34F	0x0C34F	0x61A7	0x270F	0x1387
Resolution (bits) (decimal)	15.9	15.9	15.6	15.6	14.6	13.3	12.3

TABLE 16-7: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS WITH PERIPHERAL BUS CLOCK OF 50 MHZ (32-BIT MODE)

PWM Frequency	100 Hz	200 Hz	500 Hz	1 kHz	2 kHz	5 kHz	10 kHz
Period Register Value (hex)	1	1	1	1	1	8	1
Resolution (bits) (decimal)	0x0007A11F	0x0003D08F	0x0001869F	0x0000C34F	0x000061A7	0x000004E1	0x00001387
Resolution (bits)	18.9	17.9	16.6	15.6	14.6	10.3	12.3

16.5 Output Compare Register I/O Pin Control

When the output compare module is enabled, the I/O pin direction is controlled by the compare module. The compare module returns the I/O pin control back to the appropriate pin LAT and TRIS control bits when it is disabled.

When the PWM with Fault Protection Input mode is enabled, the OCFx Fault pin must be configured as an input by setting the respective TRIS SFR bit. The OCFx Fault input pin is not automatically configured as an input when PWM with Fault Input mode is selected.

TABLE 16-8: PINS ASSOCIATED WITH OUTPUT COMPARE MODULES 1- 5

Pin Name	Module Control	Controlling Bit Field	Required TRIS bit Setting	Pin Type	Buffer Type	Description
OC1	ON[2]	OCM<2:0>[1,3]	—	D, O	—	Output Compare/PWM Channel 1
OC2	ON[2]	OCM<2:0>[1,3]	—	D, O	—	Output Compare/PWM Channel 2
OC3	ON[2]	OCM<2:0>[1,3]	—	D, O	—	Output Compare/PWM Channel 3
OC4	ON[2]	OCM<2:0>[1,3]	—	D, O	—	Output Compare/PWM Channel 4
OC5	ON[2]	OCM<2:0>[1,3]	—	D, O	—	Output Compare/PWM Channel 5
OCFA	ON[2]	OCM<2:0>[1,3]	Input	D, I	ST	PWM Fault Protection A Input (For Channels 1-3)[4]
OCFB	ON[2]	OCM<2:0>[1,3]	Input	D, I	ST	PWM Fault Protection B Input (For Channels 4 -5)[4]

Legend: ST = Schmitt Trigger input with CMOS levels, I = Input, O = Output, A = Analog, D = Digital

Note 1: All pins are subject to device pin priority control.

2: ON (OCxCON<15>) = `1`. When the module is turned off, pins controlled by the module are released.

3: Mode select bits OCM<2:0> (CMxCON<2:0>).

4: Use of PWM Fault input is optional and is controlled by the OCM bits.

NOTES:

17.0 SERIAL PERIPHERAL INTERFACE (SPI)

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

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 Serial EEPROMs, shift registers, display drivers, A/D converters, etc. The PIC32MX SPI module is compatible with Motorola® SPI and SIOP interfaces.

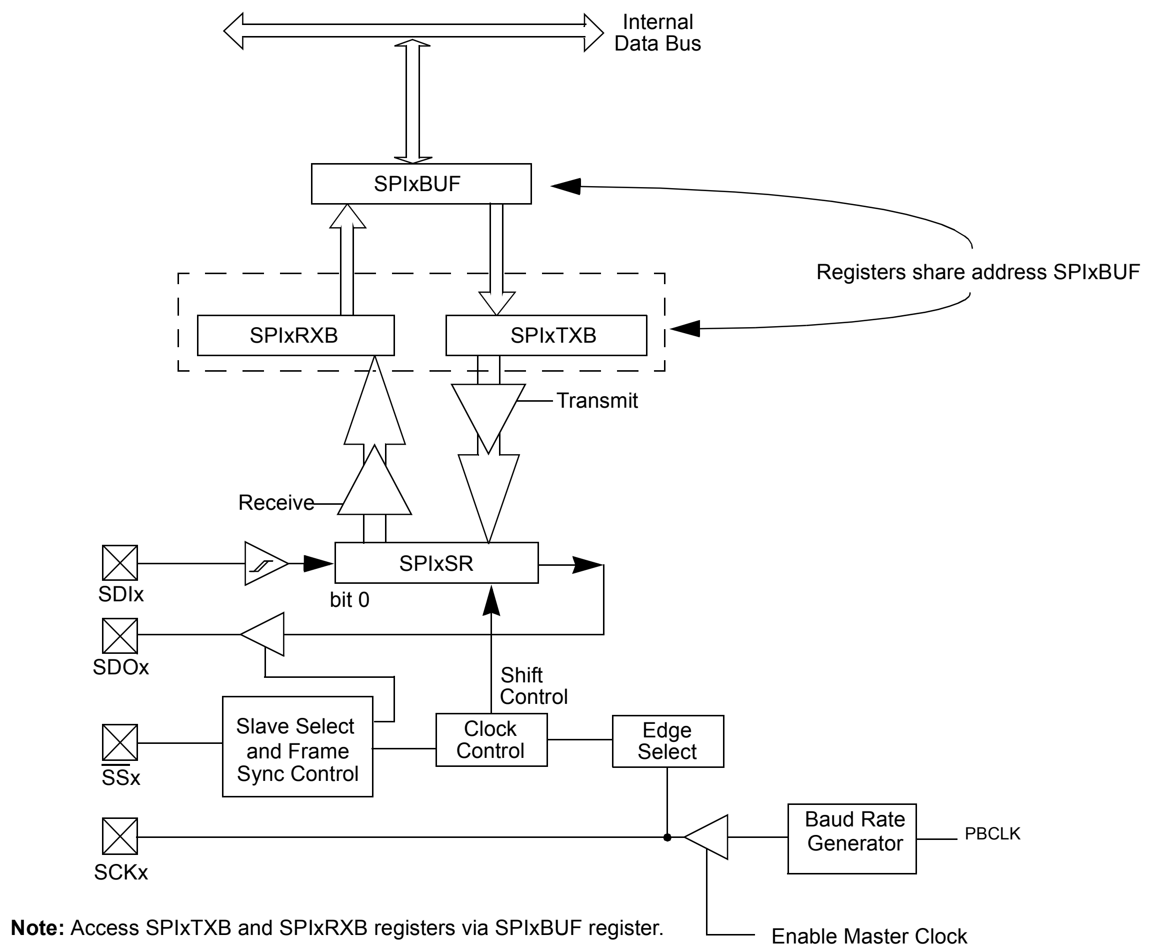
Following are some of the key features of this module:

- Master and Slave Modes Support
- Four Different Clock Formats
- Framed SPI Protocol Support
- User Configurable 8-Bit, 16-Bit and 32-Bit Data Width
- Separate SPI Data Registers for Receive and Transmit
- Programmable Interrupt Event on every 8-Bit, 16-Bit and 32-Bit Data Transfer
- Operation during CPU Sleep and Idle Mode
- Fast Bit Manipulation using CLR, SET and INV Registers

TABLE 17-1: SPI FEATURES

Available SPI Modes	SPI Master	SPI Slave	Frame Master	Frame Slave	8-Bit, 16-Bit and 32-Bit Modes	Selectable Clock Pulses and Edges	Selectable Frame Sync Pulses and Edges	Slave Select Pulse
Normal Mode	Yes	Yes	—	—	Yes	Yes	—	Yes
Framed Mode	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No

FIGURE 17-1: SPI MODULE BLOCK DIAGRAM



Note: Access SPIxTXB and SPIxRXB registers via SPIxBUF register.

17.1 SPI Registers

TABLE 17-2: SPI1 SFR SUMMARY

Virtual Address	Name		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
BF80_5800	SPI1CON	31:24	FRMEN	FRMSYNC	FRMPOL	—	—	—	—	—
		23:16	—	—	—	—	—	—	SPIFE	—
		15:8	ON	FRZ	SIDL	DISSDO	MODE32	MODE16	SMP	CKE
		7:0	SSEN	CKP	MSTEN	—	—	—	—	—
BF80_5804	SPI1CONCLR	31:0	Write clears selected bits in SPI1CON, read yields an undefined value							
BF80_5808	SPI1CONSET	31:0	Write sets selected bits in SPI1CON, read yields an undefined value							
BF80_580C	SPI1CONINV	31:0	Write inverts selected bits in SPI1CON, read yields an undefined value							
BF80_5810	SPI1STAT	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	SPIBUSY	—	—	—
		7:0	—	SPIROV	—	—	SPITBE	—	—	SPIRBF
BF80_5814	SPI1STATCLR	31:0	Write clears selected bits in SPI1STAT, read yields an undefined value							
BF80_5820	SPI1BUF	31:24	DATA<31:24>							
		23:16	DATA<23:16>							
		15:8	DATA<15:8>							
		7:0	DATA<7:0>							
BF80_5830	SPI1BRG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	BRG<8>
		7:0	BRG<7:0>							
BF80_5834	SPI1BRGCLR	31:0	Write clears selected bits in SPI1BRG, read yields an undefined value							
BF80_5838	SPI1BRGSET	31:0	Write sets selected bits in SPI1BRG, read yields an undefined value							
BF80_583C	SPI1BRGINV	31:0	Write inverts selected bits in SPI1BRG, read yields an undefined value							

TABLE 17-3: SPI1 INTERRUPT REGISTER SUMMARY

Virtual Address	Name		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
BF88_1040	IEC0	31:24	I2C1MIE	I2C1SIE	I2C1BIE	U1TXIE	U1RXIE	U1EIE	SPI1RXIE	SPI1TXIE
		23:16	SPI1EIE	OC5IE	IC5IE	T5IE	INT4IE	OC4IE	IC4IE	T4IE
BF88_1010	IFS0	31:24	I2C1MIF	I2C1SIF	I2C1BIF	U1TXIF	U1RXIF	U1EIF	SPI1RXIF	SPI1TXIF
		23:16	SPI1EIF	OC5IF	IC5IF	T5IF	INT4IF	OC4IF	IC4IF	T4IF
BF88_10C0	IPC5	31:24	—	—	—	SPI1IP<2:0>			SPI1IS<1:0>	

Note: This summary table contains partial register definitions that only pertain to the SPI1 peripheral. Refer to the "*PIC32MX Family Reference Manual*" (DS61132) for a detailed description of these registers.

TABLE 17-4: SPI2 SFR SUMMARY

Virtual Address	Name		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
BF80_5A00	SPI2CON	31:24	FRMEN	FRMSYNC	FRMPOL	—	—	—	—	—
		23:16	—	—	—	—	—	—	SPIFE	—
		15:8	ON	FRZ	SIDL	DISSDO	MODE32	MODE16	SMP	CKE
		7:0	SSEN	CKP	MSTEN	—	—	—	—	—
BF80_5A04	SPI2CONCLR	31:0	Write clears selected bits in SPI2CON, read yields an undefined value							
BF80_5A08	SPI2CONSET	31:0	Write sets selected bits in SPI2CON, read yields an undefined value							
BF80_5A0C	SPI2CONINV	31:0	Write inverts selected bits in SPI2CON, read yields an undefined value							
BF80_5A10	SPI2STAT	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	SPIBUSY	—	—	—
		7:0	—	SPIROV	—	—	SPITBE	—	—	SPIRBF
BF80_5A14	SPI2STATCLR	31:0	Write clears selected bits in SPI2STAT, read yields an undefined value							
BF80_5A20	SPI2BUF	31:24	DATA<31:24>							
		23:16	DATA<23:16>							
		15:8	DATA<15:8>							
		7:0	DATA<7:0>							
BF80_5A30	SPI2BRG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	BRG<8>
		7:0	BRG<7:0>							
BF80_5A34	SPI2BRGCLR	31:0	Write clears selected bits in SPI2BRG, read yields an undefined value							
BF80_5A38	SPI2BRGSET	31:0	Write sets selected bits in SPI2BRG, read yields an undefined value							
BF80_5A3C	SPI2BRGINV	31:0	Write inverts selected bits in SPI2BRG, read yields an undefined value							

TABLE 17-5: SPI2 INTERRUPT REGISTER SUMMARY

Virtual Address	Name		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
BF88_1060	IEC1	7:0	SPI2RXIE	SPI2TXIE	SPI2EIE	CMP2IE	CMP1IE	PMPIE	AD1IE	CNIE
BF88_1030	IFS1	7:0	SPI2RXIF	SPI2TXIF	SPI2EIF	CMP2IF	CMP1IF	PMPIF	AD1IF	CNIF
BF88_10E0	IPC7	23:16	—	—	—	SPI2IP<2:0>			SP2IS<1:0>	

Note: This summary table contains partial register definitions that only pertain to the SPI2 peripheral. Refer to the "*PIC32MX Family Reference Manual*" (DS61132) for a detailed description of these registers.

REGISTER 17-1: SPIXCON: SPI CONTROL REGISTER

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
FRMEN	FRMSYNC	FRMPOL	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	U-0
—	—	—	—	—	—	SPIFE	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ON	FRZ	SIDL	DISSDO	MODE32	MODE16	SMP	CKE
bit 15							bit 8

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
SSEN	CKP	MSTEN	—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31 **FRMEN:** Framed SPI Support bit
1 = Framed SPI support is enabled (SSx pin used as FSYNC input/output)
0 = Framed SPI support is disabled

bit 30 **FRMSYNC:** Frame Sync Pulse Direction Control on SSx pin bit (Framed SPI mode only)
1 = Frame sync pulse input (Slave mode)
0 = Frame sync pulse output (Master mode)

bit 29 **FRMPOL:** Frame Sync Polarity bit (Framed SPI mode only)
1 = Frame pulse is active-high
0 = Frame pulse is active-low

bit 28-18 **Unimplemented:** Read as ‘0’

bit 17 **SPIFE:** Frame Sync Pulse Edge Select bit (framed SPI mode only)
1 = Frame synchronization pulse coincides with the first bit clock
0 = Frame synchronization pulse precedes the first bit clock

bit 16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** SPI Peripheral On bit
1 = SPI Peripheral is enabled
0 = SPI Peripheral is disabled

bit 14 **FRZ:** Freeze in DEBUG Exception Mode bit
1 = Freeze operation when CPU enters Debug Exception mode
0 = Continue operation when CPU enters Debug Exception mode
Note: FRZ is writable in Debug Exception mode only, it is forced to ‘0’ in Normal mode.

bit 13 **SIDL:** Stop in IDLE Mode bit
1 = Discontinue operation when CPU enters in Idle mode
0 = Continue operation in Idle mode

bit 12 **DISSDO:** Disable SDOx pin bit
1 = SDOx pin is not used by the module. Pin is controlled by associated PORT register
0 = SDOx pin is controlled by the module

bit 11-10	**MODE<32,16>:** 32/16-Bit Communication Select bits 1x = 32-bit data width 01 = 16-bit data width 00 = 8-bit data width
bit 9	**SMP:** SPI Data Input Sample Phase bit Master mode (MSTEN = 1): 1 = Input data sampled at end of data output time 0 = Input data sampled at middle of data output time Slave mode (MSTEN = 0): SMP value is ignored when SPI is used in Slave mode. The module always uses SMP = 0.
bit 8	**CKE:** SPI Clock Edge Select bit 1 = Serial output data changes on transition from active clock state to Idle clock state (see CKP bit) 0 = Serial output data changes on transition from Idle clock state to active clock state (see CKP bit) **Note:** The CKE bit is not used in the Framed SPI mode. The user should program this bit to '0' for the Framed SPI mode (FRMEN = 1).
bit 7	**SSEN:** Slave Select Enable (Slave mode) bit 1 = $\overline{SSx}$ pin used for Slave mode 0 = $\overline{SSx}$ pin not used for Slave mode, pin controlled by port function.
bit 6	**CKP:** Clock Polarity Select bit 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:** Master Mode Enable bit 1 = Master mode 0 = Slave mode
bit 4-0	**Unimplemented:** Read as '0'

REGISTER 17-2: SPIXSTAT: SPI STATUS REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	R-0	U-0	U-0	U-0
—	—	—	—	SPIBUSY	—	—	—
bit 15							bit 8

U-0	R/W-0	U-0	U-0	R-0	U-0	U-0	R-0
—	SPIROV	—	—	SPITBE	—	—	SPIRBF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-12 **Unimplemented:** Read as ‘0’

bit 11 **SPIBUSY:** SPI Activity Status bit
1 = SPI peripheral is currently busy with some transactions
0 = SPI peripheral is currently idle

bit 10-7 **Unimplemented:** Read as ‘0’

bit 6 **SPIROV:** Receive Overflow Flag bit
1 = A new data is completely received and discarded. The user software has not read the previous data in the SPIxBUF register.
0 = No overflow has occurred
This bit is set in hardware; can only be cleared (= 0) in software.

bit 5-4 **Unimplemented:** Read as ‘0’

bit 3 **SPITBE:** SPI Transmit Buffer Empty Status bit
1 = Transmit buffer, SPIxTXB is empty
0 = Transmit buffer, SPIxTXB is not empty
Automatically set in hardware when SPI transfers data from SPIxTXB to SPIxSR.
Automatically cleared in hardware when SPIxBUF is written to, loading SPIxTXB.

bit 2 **Unimplemented:** Read as ‘0’

bit 1 **Unimplemented:** Read as ‘0’

bit 0 **SPIRBF:** SPI Receive Buffer Full Status bit
1 = Receive buffer, SPIxRXB is full
0 = Receive buffer, SPIXRXB is not full
Automatically set in hardware when SPI transfers data from SPIxSR to SPIxRXB.
Automatically cleared in hardware when SPIxBUF is read from, reading SPIxRXB.

17.2 Master and Slave Modes

The PIC32MX SPI module operates in normal Master or Slave modes and offers the following additional modes:

- Framed Master
- Framed Slave
- 8, 16, 32-Bit Data Width Transfers
- Slave Select (Slave mode only)

Below is a typical system Master – Slave connection diagram.

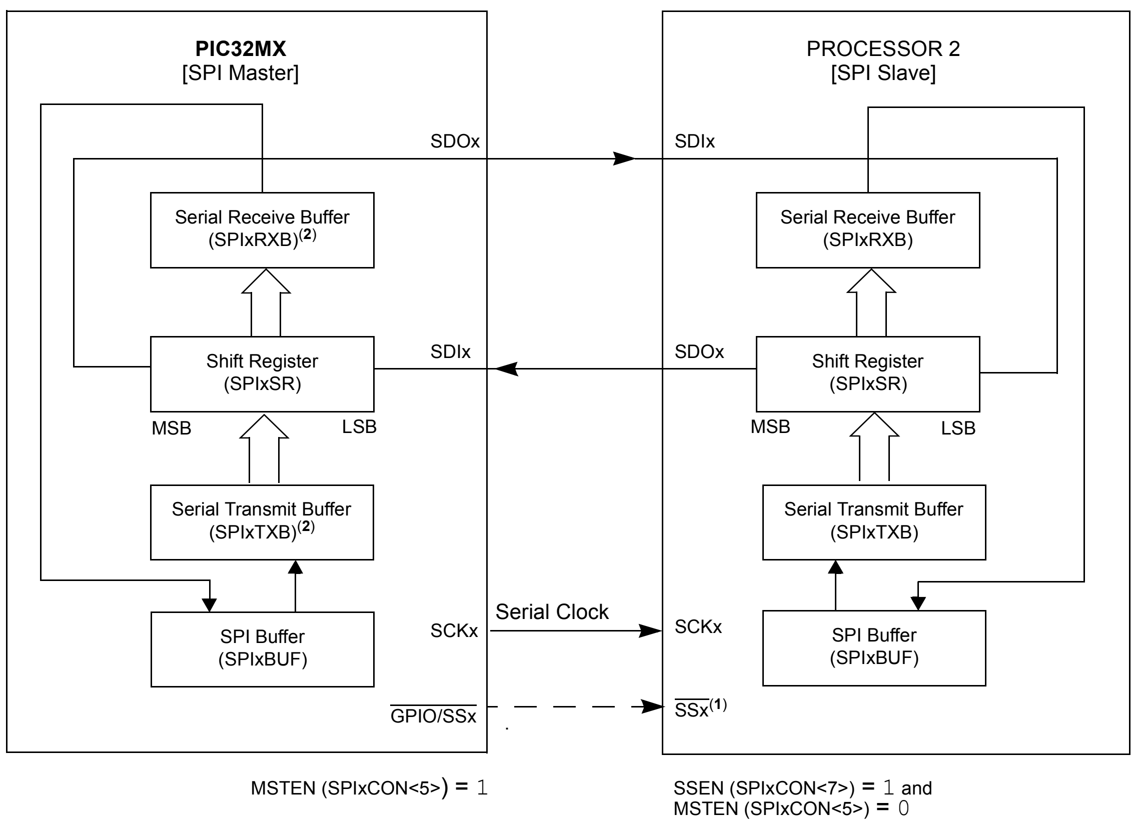
17.2.1 8, 16, 32-BIT OPERATION

The PIC32MX 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.

Two control bits, MODE32 and MODE16 (SPIxCON<11:10>), 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 (SPIxCON<15> = 0), the new mode will be available when the module is again switched on.

The number of clock pulses at the SCKx pin are dependent on the selected mode of operation. For 8-Bit mode, 8 clocks; for 16-Bit mode, 16 clocks; and for 32-Bit mode, 32 clocks are required.

FIGURE 17-2: SPI MASTER/SLAVE CONNECTION



Note 1: Using the $\overline{SSx}$ pin in Slave mode of operation is optional.

2: User must write transmit data to SPIxBUF and read received data from SPIxBUF. The SPIxTXB and SPIxRXB registers are memory mapped to SPIxBUF.

17.2.2 MASTER MODE

In Master mode, data from the SPIxBUF register is transmitted synchronously on the SDO (output) pin while synchronous data is received from the slave device on the SDI (input) pin. In this mode, the Master controls the synchronous data transfer with the SCK clock pin by generating 8, 16 or 32 clock pulses, depending on the selected data size.

17.2.2.1 Master Mode Operations

In Master mode the SCK and SDO pins are outputs and the SDI pin is an input. Setting the control bit, DISSDO (SPIxCON<12>), disables transmission at the SDO pin if Receive Only mode of operation is desired.

Refer to Table 17-7.

The SDI (input) must be configured to properly sample the data received from the slave device by configuring the sample bit, SMP (SPIxCON<9>).

In Master mode, the SCK clock edge and polarity must be configured properly for the master and slave device to correctly transfer data synchronously.

Refer to the timing diagram shown in Figure 17-3 to determine the appropriate settings.

In Master mode, the data transfers can be 8, 16, or 32-bits and are configured using control bits, MODE<32,16> (SPIxCON<11:10>).

Refer to **Section 17.2.1 "8, 16, 32-Bit Operation"** for details.

In Master mode, the system clock is divided and then used as the serial clock. The division is based on the settings in the SPIxBRG register.

Refer to **Section 17.2.5 "SPI Master Mode Clock Frequency"**.

17.2.2.2 Master SPIxCON Configuration

The following bits must be configured as shown for the Master mode of operation when configuring the SPIxCON register:

- Enable Master Mode
  MSTEN (SPIxCON<5>) = `1`.
- Disable Framed SPI support
  FRMEN (SPIxCON<31>) = `0`

The remaining bits are shown with example configurations and may be configured as desired:

- Enable module control of SDO pin – DISSDO (SPIxCON<12>) = `0`
- Configure SCK clock polarity to idle high –
  CKP (SPIxCON<6>) = `1`
- Configure SCK clock edge transition from Idle to active – CKE (SPIxCON<8>) = `0`
- Select 16-bit data width –
  MODE<32,16> (SPIxCON<11:10>) = `01`
- Sample data input at middle –
  SMP (SPIxCON<9>) = `0`
- Enable SPI module when CPU idle –
  SIDL (SPIxCON<13>) = 0

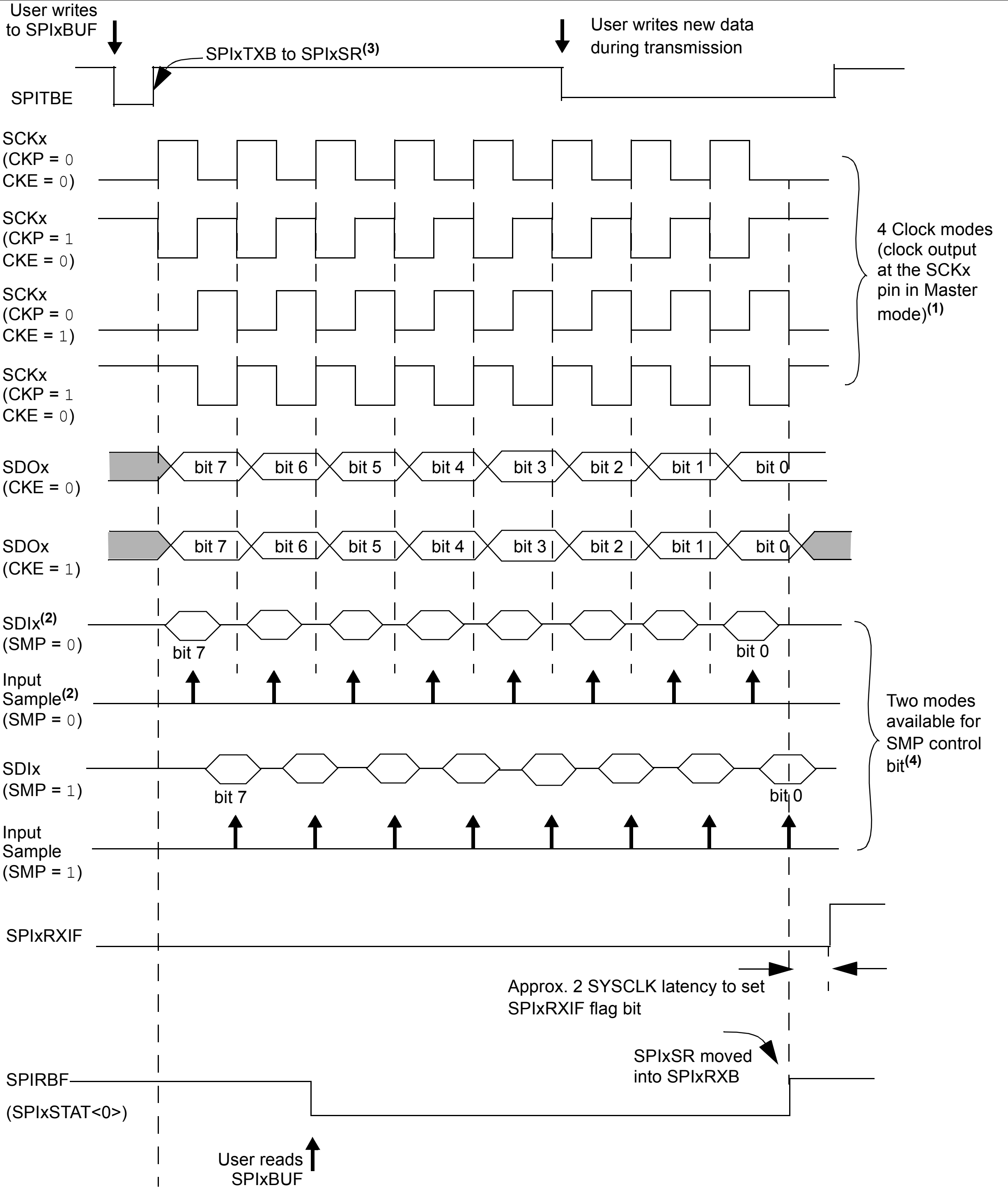
17.2.2.3 Master Mode Initialization

The following steps should be performed to setup the SPI module for the Master mode of operation:

1. If interrupts are used, disable the SPI interrupts in the respective IEC0/1 register.
2. Stop and reset the SPI module by clearing the ON bit.
3. Clear the receive buffer.
4. If interrupts are used, the following additional steps are performed:
   - Clear the SPIx interrupt flags/events in the respective IFS0/1 register.
   - Set the SPIx interrupt enable bits in the respective IEC0/1 register.
   - Write the SPIx interrupt priority and subpriority bits in the respective IPC5/7 register.
5. Write the Baud Rate register, SPIxBRG.
6. Clear the SPIROV bit (SPIxSTAT<6>).
7. Write the selected configuration settings to the SPIxCON register.
8. Enable SPI operation by setting the ON bit (SPIxCON<15>).
9. 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 1:	When using the Slave Select mode, the $\overline{\text{SSx}}$ or another GPIO pin is used to control the slave's $\overline{\text{SSx}}$ input. The pin must be controlled in software.
2:	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.
3:	The SPI device must be turned off prior to changing the mode from Slave to Master.
4:	The SPIxSR register cannot be written to directly by the user. All writes to the SPIxSR register are performed through the SPIxBUF register.

FIGURE 17-3: SPI MASTER MODE OPERATION IN 8-BIT MODE (MODE32 = 0, MODE16 = 0)



Note 1: Four SPI Clock modes are shown here to demonstrate the functionality of bits CKP (SPIxCON<6>) and CKE (SPIxCON<8>). Only one of the four modes can be chosen for operation.

2: The SDI and input samples shown here for two different values of the SMP bit (SPIxCON<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 SPIxSR as soon as the user writes to SPIxBUF.

4: The operation for 8-Bit mode is shown. The 16-Bit and 32-Bit modes are similar.

EXAMPLE 17-1: INITIALIZATION FOR 16-BIT SPI MASTER MODE

```
/*
    The following code example will initialize the SPI1 in master mode.
    It assumes that none of the SPI1 input pins are shared with an analog input.
    If so, the AD1PCFG and corresponding TRIS registers have to be properly configured.
*/
    int rData;

    IEC0CLR=0x03800000;          // disable all interrupts
    SPI1CON = 0;                 // Stops and resets the SPI1.
    rData=SPI1BUF;               // clears the receive buffer
    IFS0CLR=0x03800000;          // clear any existing event
    IPC5CLR=0x1f000000;          // clear the priority
    IPC5SET=0x0d000000;          // Set IPL=3, subpriority 1
    IEC0SET=0x03800000;          // Enable Rx, Tx and Error interrupts

    SPI1BRG=0x1;                 // use FPB/4 clock frequency
    SPI1STATCLR=0x40;            // clear the Overflow
    SPI1CON=0x8220;              // SPI ON, 8 bits transfer, SMP=1, Master Mode

                                 // from now on, the device is ready to transmit and receive
                                    data
    SPI1BUF='A';                 // transmit an A character
```

17.2.3 SLAVE MODE

In Slave mode, data from the SPIxBUF register is transmitted synchronously on the SDO (output) pin while synchronous data is received from the Master device on the SDI (input) pin. In this mode, the Master device controls the synchronous data transfer with the SCK clock pin by generating 8, 16 or 32 clock pulses, depending on the selected data size.

17.2.3.1 Slave Mode Operations

The SDO pin is an output and the SPI pin is an input. Setting the control bit, DISSDO (SPIxCON<12>), disables transmission at the SDO pin if Receive Only mode of operation is desired.

Refer to Table 17-7.

The SDI (input) must be configured to properly sample the data received from the slave device by configuring the sample bit, SMP (SPIxCON<9>).

Refer to timing diagram shown in Figure 17-4 to determine the appropriate settings.

Data transfers can be 8, 16, or 32-bits and are configured using control bits. MODE<32,16> (SPIxCON<11:10>).

Refer to **Section 17.2.1 "8, 16, 32-Bit Operation"** for details.

Slave Select Synchronization: The $\overline{\text{SSx}}$ pin allows a Synchronous Slave mode. If the SSEN (SPIxCON<7>) bit is set, transmission and reception is enabled in Slave mode only if the $\overline{\text{SSx}}$ pin is driven to a low state. If the SSEN bit is not set, the $\overline{\text{SSx}}$ pin does not affect the module operation in Slave mode.

17.2.3.2 Slave SPIxCON Configuration

The following bits must be configured as shown for the Slave mode of operation when configuring the SPIxCON register:

- Enable Slave Mode –
  MSTEN (SPIxCON<5>) = `0`.
- Disable Framed SPI support – FRMEN (SPIxCON<31>) = `0`

The remaining bits are shown with example configurations and may be configured as desired:

- Enable module control of SDO pin –
  DISSDO (SPIxCON<12>) = `0`
- Configure SCK clock polarity to Idle high –
  CKP (SPIxCON<6>) = `1`
- Configure SCK clock edge transition from Idle to active – CKE (SPIxCON<8>) = `0`
- Disable Slave Select Pin –
  SSEN (SPIxCON<7>) = `0`
- Select 16-bit data width –
  MODE<32,16> (SPIxCON<11:10>) = `01`
- Sample data input at middle –
  SMP (SPIxCON<9>) = `0`
- Enable SPI module when CPU Idle –
  SIDL (SPIxCON<13>) = `0`

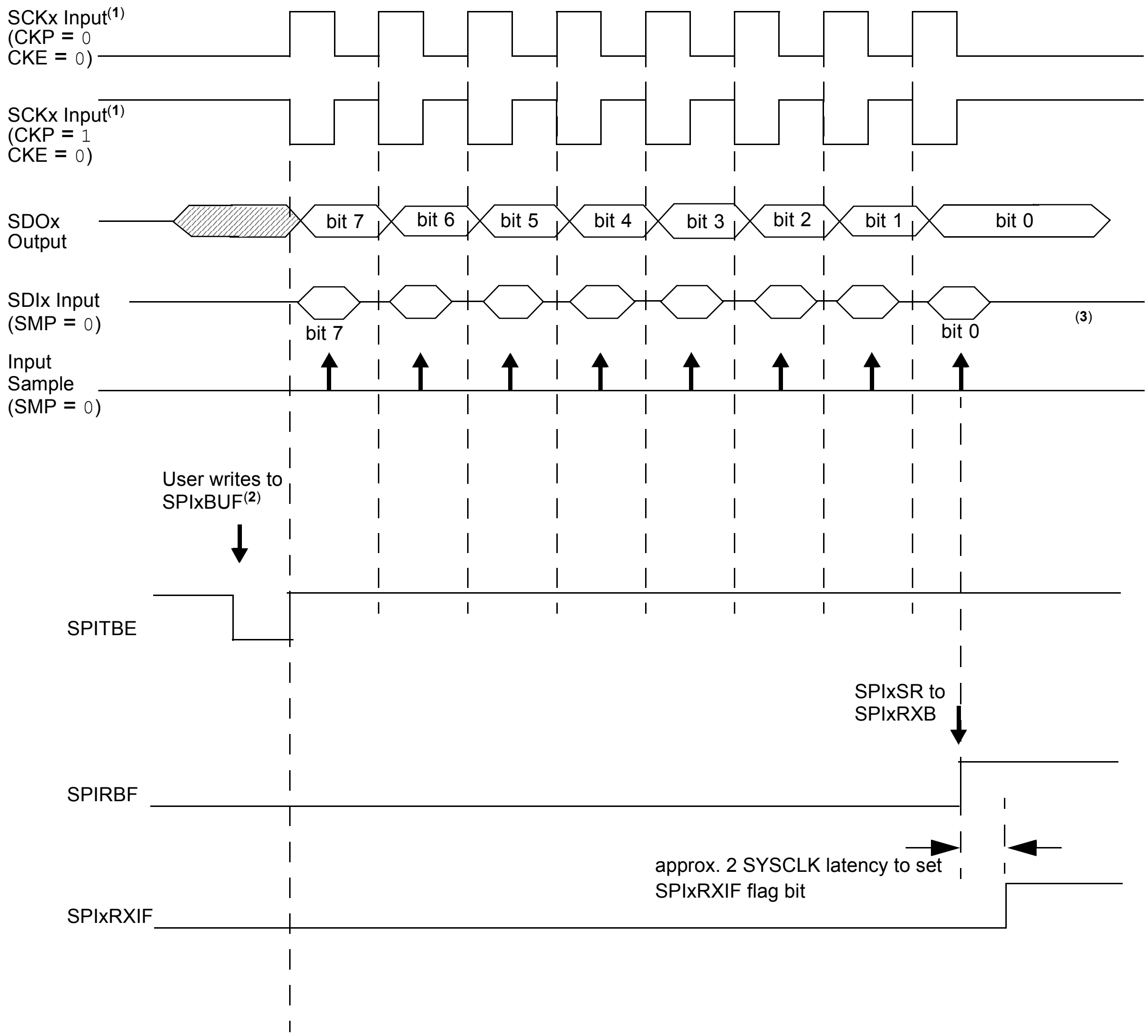
17.2.3.3 Slave Mode Initialization

The following steps are used to set up the SPI module for the Slave mode of operation:

1. If interrupts are used, disable the SPI interrupts in the respective IEC0/1 register.
2. Stop and reset the SPI module by clearing the ON bit.
3. Clear the receive buffer.
1. If using interrupts, the following additional steps are performed:
   - Clear the SPIx interrupt flags/events in the respective IFS0/1 register.
   - Set the SPIx interrupt enable bits in the respective IEC0/1 register.
   - Write the SPIx interrupt priority and subpriority bits in the respective IPC5/7 register.
2. Clear the SPIROV bit (SPIxSTAT<6>).
3. Write the selected configuration settings to the SPIxCON register with MSTEN (<SPIxCON<5>) = `0`.
4. Enable SPI operation by setting the ON bit (SPIxCON<15>).
5. Transmission (and reception) will start as soon as the master provides the serial clock.

> **Note 1:** 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.
>
> **2:** The SPI device must be turned off prior to changing the mode from Master to Slave.
>
> **3:** The SPIxSR register cannot be written into directly by the user. All writes to the SPIxSR register are performed through the SPIxBUF register.

FIGURE 17-4: SPI SLAVE MODE OPERATION IN 8-BIT MODE WITH SLAVE SELECT PIN DISABLED (MODE32 = 0, MODE16 = 0, SSEN = 0)



Note 1: Two SPI Clock modes shown only to demonstrate CKP (SPIxCON<6>) and CKE (SPIxCON<8>) bit functionality. Any combination of CKP and CKE bits can be chosen for module operation.

2: If there are no pending transmissions or a transmission in progress, SPIxBUF is transferred to SPIxSR as soon as the user writes to SPIxBUF.

3: The operation for 8-Bit mode is shown. The 16-Bit and 32-Bit modes are similar.

EXAMPLE 17-2: FOR 16-BIT SPI SLAVE MODE INITIALIZATION

```
/*
    The following code example will initialize the SPI1 in slave mode with SSEN.
    It assumes that the SPI1 SS input pin on RB2 is shared with the AN2 analog input.
    It thus properly configures the corresponding AD1PCFG and TRIS registers bits.
*/
    int     rData;

    IEC0CLR=0x03800000;             // disable all interrupts
    SPI1CON = 0;                    // Stops and resets the SPI1.
    TRISBSET = 0x4;                 // Set RB2 as a digital input
    AD1PCFGSET = 0x4;               // Analog input pin in digital mode
    rData=SPI1BUF;                  // clears the receive buffer
    IFS0CLR=0x03800000;             // clear any existing event
    IPC5CLR=0x1f000000;             // clear the priority
    IPC5SET=0x0d000000;             // Set IPL=3, subpriority 1
    IEC0SET=0x03800000;             // Enable Rx, Tx and Error interrupts

    SPI1STATCLR=0x40;               // clear the Overflow
    SPI1CON=0x8000;                 // SPI ON, 8 bits transfer, Slave Mode

                                    // from now on, the device is ready to receive and
                                       transmit data
```


17.2.4 FRAMED SPI MODES

The module supports a very basic framed SPI protocol while operating in either Master or Slave modes. The following features are provided in the SPI module to support Framed SPI modes:

- The control bit, FRMEN (SPIxCON<31>), enables Framed SPI mode and causes the $\overline{SSx}$ pin to be used as a frame synchronization pulse input or output pin. The state of the SSEN (SPIxCON<7>) is ignored.
- The control bit, FRMSYNC (SPIxCON<30>), 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 (SPIxCON<29>) determines the frame synchronization pulse polarity for a single SPI clock cycle.

The following framed SPI modes are supported by the SPI module:

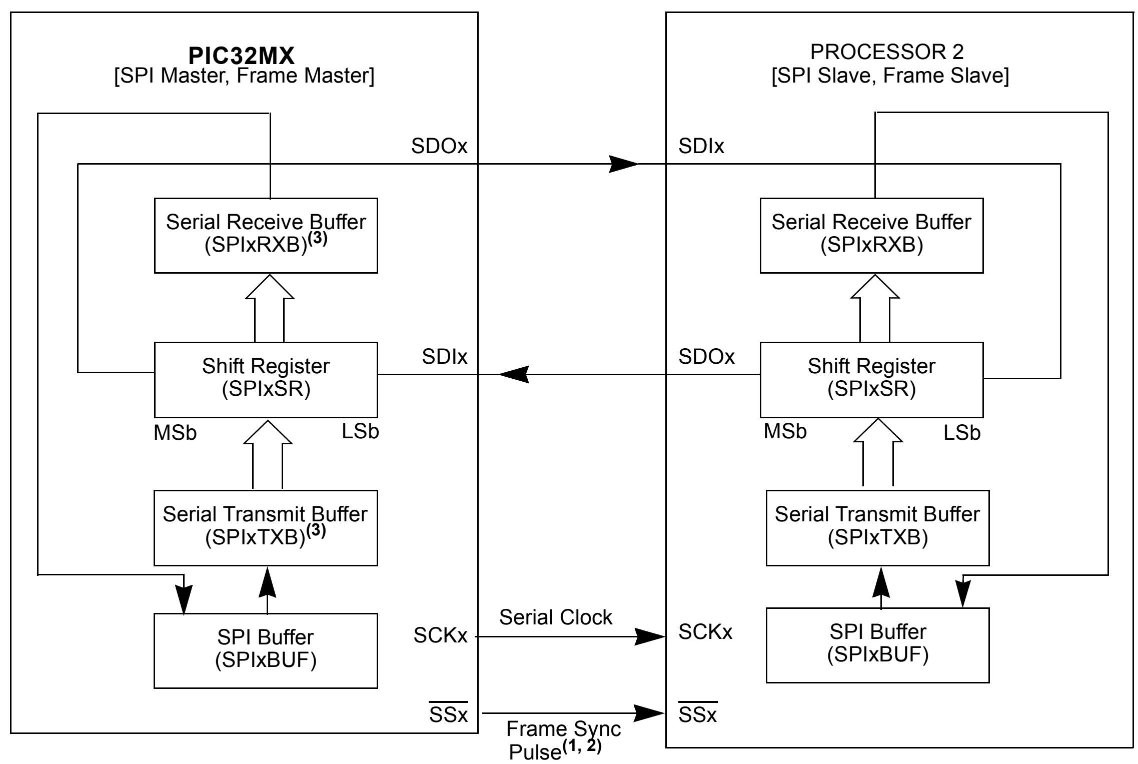
- Frame Master mode: The SPI module generates the frame synchronization pulse and provides this pulse to other devices at the $\overline{SSx}$ pin.
- Frame Slave mode: The SPI module uses a frame synchronization pulse received at the $\overline{SSx}$ pin.

The Framed SPI modes are supported in conjunction with the Master and Slave modes. Thus, the following framed SPI configurations are available:

- SPI Master mode and Frame Master mode
- SPI Master mode and Frame Slave mode
- SPI Slave mode and Frame Master mode
- SPI Slave mode and Frame Slave mode

These four modes determine whether or not the SPI module generates the serial clock and the frame synchronization pulse.

FIGURE 17-5: SPI MASTER, FRAME MASTER CONNECTION DIAGRAM



Note 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.

17.2.4.1 SPI Master Mode and Frame Master Mode Operations

This Framed SPI mode is enabled by setting bits MSTEN (SPIxCON<5>) and FRMEN (SPIxCON<31>) to '1', and bit FRMSYNC (SPIxCON<30>) 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 $\overline{\text{SSx}}$ pin will be driven active, high or low depending on bit FRMPOL (SPIxCON<29>), on the next transmit edge of the SCKx clock. The $\overline{\text{SSx}}$ pin will be active for one SCKx clock cycle. The module will start transmitting data on the same or on the next transmit edge of the SCKx, depending on the SPIFE (SPIxCON<17>) setting, as shown in Figure 17-6. A connection diagram indicating signal directions for this operating mode is shown in Figure 17-5.

The SCK, SDO and $\overline{\text{SSx}}$ pins are outputs, the SDI pin is an input. Setting the control bit, DISSDO (SPIxCON<12>), disables transmission at the SDO pin if Receive Only mode of operation is desired.

Refer to Table 17-7.

The SDI (input) must be configured to properly sample the data received from the slave device by configuring the sample bit, SMP (SPIxCON<9>).

In Master mode, the SCK clock edge and polarity must be configured properly for the master and slave device to correctly transfer data synchronously.

Refer to timing diagram shown in Figure 17-3 to determine the appropriate settings.

17.2.4.2 Master SPIxCON Configuration

The following bits must be configured as shown for the Master mode of operation when configuring the SPIxCON register:

- Enable Master Mode – MSTEN (SPIxCON<5>) = `1`
- Enable Framed SPI support – FRMEN (SPIxCON<31>) = `1`
- Select $\overline{\text{SSx}}$ pin as Frame Master (output) – FRMSYNC(SPIxCON<30>) = `0`

The remaining bits are shown with example configurations and may be configured as desired:

- Enable module control of SDO pin – SDO (SPIxCON<12>) = `0`
- Configure SCK clock polarity to Idle high – CKP (SPIxCON<6>) = `1`
- Configure SCK clock edge transition from Idle to active – CKE (SPIxCON<8>) = `0`
- Select $\overline{\text{SSx}}$ active-low pin polarity – FRMPOL (SPIxCON<29>) = `0`
- Select 16-bit data width – MODE<32,16> (SPIxCON<11:10>) = `01`
- Sample data input at middle – SMP (SPIxCON<9>) = `0`
- Enable SPI module when CPU Idle – SIDL (SPIxCON<13>) = `0`

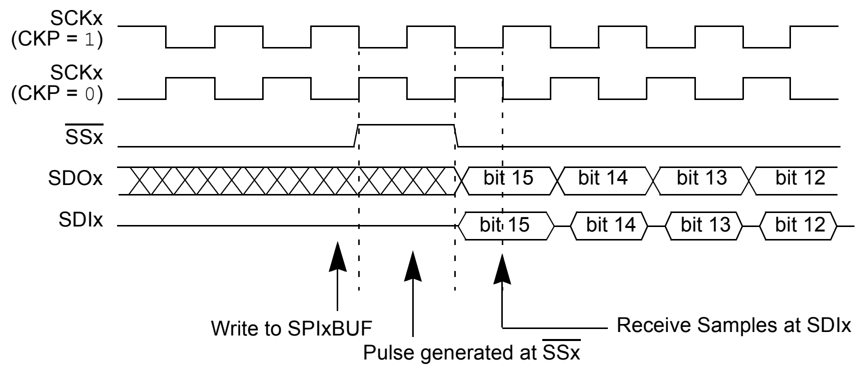
17.2.4.3 Framed Master Mode Initialization

The following steps are used to set up the SPI module for the Master mode of operation:

1. If interrupts are used, disable the SPI interrupts in the respective IEC0/1 register.
2. Stop and reset the SPI module by clearing the ON bit.
3. Clear the receive buffer.
4. If using interrupts, the following additional steps are performed:
   - Clear the SPIx interrupt flags/events in the respective IFS0/1 register.
   - Set the SPIx interrupt enable bits in the respective IEC0/1 register.
   - Write the SPIx interrupt priority and subpriority bits in the respective IPC5/7 register.
5. Clear the SPIROV bit (SPIxSTAT<6>).
6. Write the selected configuration settings to the SPIxCON register.
7. Enable SPI operation by setting the ON bit (SPIxCON<15>).

> **Note 1:** 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.
>
> **2:** The SPIxSR register cannot be written into directly by the user. All writes to the SPIxSR register are performed through the SPIxBUF register.

FIGURE 17-6: SPI MASTER, FRAME MASTER MODE32 = 0, MODE16 = 1, SPIFE = 0, FRMPOL = 1)(



17.2.4.4 SPI Master Mode and Frame Slave Mode Operations

This Framed SPI mode is enabled by setting bits MSTEN (SPIxCON<5>), FRMEN (SPIxCON<31>), and FRMSYNC (SPIxCON<30>) to '1'. The $\overline{\text{SSx}}$ pin is an input, and it is sampled on the sample edge of the SPI clock. When it is sampled active, high, or low depending on bit FRMPOL (SPIxCON<29>), data will be transmitted on the subsequent transmit edge of the SPI clock, as shown in Figure 17-7. The interrupt flag SPIxIF is set when the transmission is complete. The user must make sure that the correct data is loaded into the SPIxBUF for transmission before the signal is received at the $\overline{\text{SSx}}$ pin. A connection diagram indicating signal directions for this operating mode is shown in Figure 17-8.

The SCK and SDO pins are outputs, the SDI and $\overline{\text{SSx}}$ pins are inputs. Setting the control bit, DISSDO (SPIxCON<12>), disables transmission at the SDO pin if Receive Only mode of operation is desired.

Refer to Table 17-7.

The SDI pin must be configured to properly sample the data received from the slave device by configuring the sample bit, SMP (SPIxCON<9>).

In Master mode, the SCK clock edge and polarity must be configured properly for the master and slave device to correctly transfer data synchronously.

Refer to timing diagram shown in Figure 17-3 to determine the appropriate settings.

17.2.4.5 Master SPIxCON Configuration

The following bits must be configured as shown for the Master mode of operation when configuring the SPIxCON register:

- Enable Master Mode – MSTEN (SPIxCON<5>) = `1`
- Enable Framed SPI support – FRMEN (SPIxCON<31>) = `1`
- Select $\overline{\text{SSx}}$ pin as Frame Slave (input) – FRMSYNC (SPIxCON<30>) = `1`

The remaining bits are shown with example configurations and may be configured as desired:

- Enable module control of SDO pin – DISSDO (SPIxCON<12>) = `0`
- Configure SCK clock polarity to Idle high – CKP (SPIxCON<6>) = `1`
- Configure SCK clock edge transition from Idle to active – CKE (SPIxCON<8>) = `0`
- Select $\overline{\text{SSx}}$ active low pin polarity – FRMPOL (SPIxCON<29>) = `0`
- Select 16-bit data width – MODE<32,16> (SPIxCON<11:10>) = `01`
- Sample data input at middle – SMP (SPIxCON<9>) = `0`
- Enable SPI module when CPU Idle – SIDL (SPIxCON<13>) = `0`

17.2.4.6 Framed Slave Mode Initialization

The following steps are used to set up the SPI module for the Slave mode of operation:

1. If interrupts are used, disable the SPI interrupts in the respective IEC0/1 register.
2. Stop and reset the SPI module by clearing the ON bit.
3. Clear the receive buffer.
4. If using interrupts, the following additional steps are performed:
   - Clear the SPIx interrupt flags/events in the respective IFS0/1 register.
   - Set the SPIx interrupt enable bits in the respective IEC0/1 register.
   - Write the SPIx interrupt priority and subpriority bits in the respective IPC5/7 register.
5. Clear the SPIROV bit (SPIxSTAT<6>).
6. Write the selected configuration settings to the SPIxCON register.
7. Enable SPI operation by setting the ON bit (SPIxCON<15>).

Note 1:	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.
2:	The SPIxSR register cannot be written into directly by the user. All writes to the SPIxSR register are performed through the SPIxBUF register.
3:	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.

 Advance Information

FIGURE 17-7: SPI MASTER, FRAME SLAVE MODE32 = 0, MODE16 = 1, SPIFE = 0, FRMPOL = 1)(

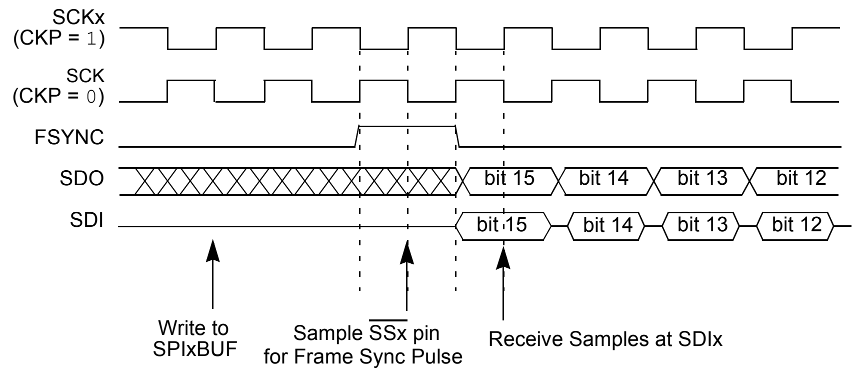
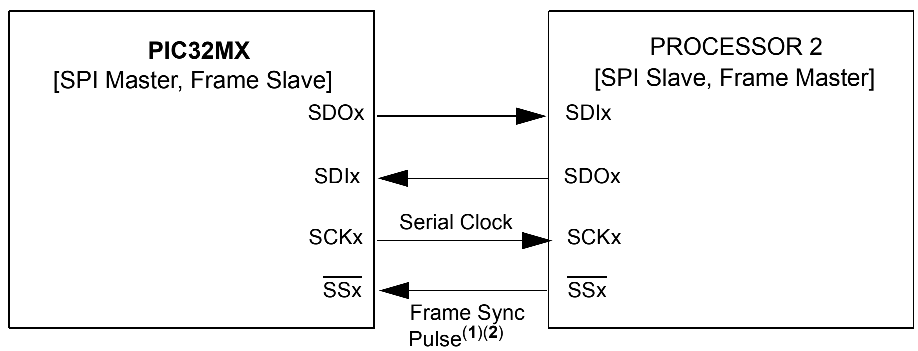


FIGURE 17-8: SPI MASTER, FRAME SLAVE CONNECTION DIAGRAM



Note 1: In Framed SPI modes, the 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 SSx pin is not optional).

17.2.4.7 SPI Slave Mode and Frame Master Mode

This Framed SPI mode is enabled by setting bit MSTEN (SPIxCON<5>) to '0', bit FRMEN (SPIxCON<31>) to '1' and bit FRMSYNC (SPIxCON<30>) to '0'. The input SPI clock will be continuous in Slave mode. The SSx pin will be an output when bit FRMSYNC is low. Therefore, when SPIBUF is written, the module will drive the SSx pin active, high or low depending on bit FRMPOL (SPIxCON<29>), on the next transmit edge of the SPI clock. The SSx pin will be driven active 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 17-9.

The SDO and SSx pins are outputs and the SCK and SDI pins are inputs. Setting the control bit, DISSDO (SPIxCON<12>), disables transmission at the SDO pin if Receive Only mode of operation is desired.

Refer to Table 17-7.

The SDI pin must be configured to properly sample the data received from the slave device by configuring the sample bit, SMP (SPIxCON<9>).

Refer to timing diagram shown in Figure 17-6 to determine the appropriate settings.

17.2.4.8 Slave SPIxCON Configuration

The following bits must be configured as shown for the Slave mode of operation when configuring the SPIxCON register:

- Enable Slave Mode – MSTEN (SPIxCON<5>) = 1
- Enable Framed SPI support – FRMEN (SPIxCON<31>) = 1
- Select SSx pin as Frame Master (output) – FRMSYNC(SPIxCON<30>) = 0

The remaining bits are shown with example configurations and may be configured as desired:

- Enable module control of SDO pin – DISSDO (SPIxCON<12>) = 0
- Configure SCK clock polarity to Idle high – CKP (SPIxCON<6>) = 1
- Configure SCK clock edge transition from Idle to active – CKE (SPIxCON<8>) = 0
- Select SSx active low pin polarity – FRMPOL (SPIxCON<29>) = 0
- Select 16-bit data width – MODE<32,16> (SPIxCON<11:10>) = 01
- Sample data input at middle – SMP (SPIxCON<9>) = 0
- Enable SPI module when CPU Idle – SIDL (SPIxCON<13>) = 0

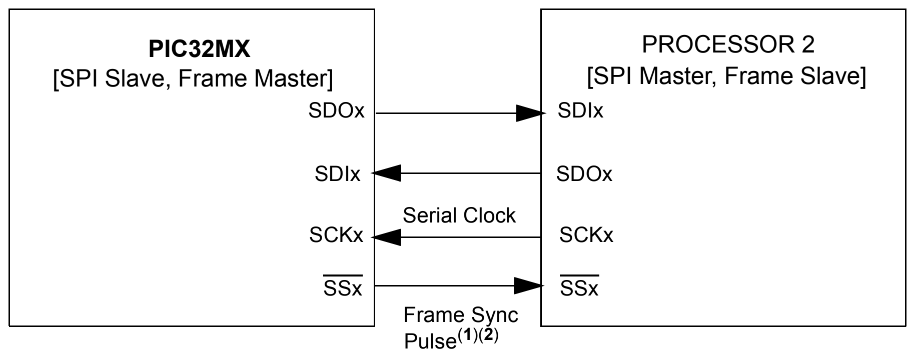
17.2.4.9 Framed Master Mode Initialization

The following steps are used to set up the SPI module for the Slave mode of operation:

1. If interrupts are used, disable the SPI interrupts in the respective IEC0/1 register.
2. Stop and reset the SPI module by clearing the ON bit.
3. Clear the receive buffer.
4. If using interrupts, the following additional steps are performed:
   - Clear the SPIx interrupt flags/events in the respective IFS0/1 register.
   - Set the SPIx interrupt enable bits in the respective IEC0/1 register.
   - Write the SPIx interrupt prioity and subpriority bits in the respective IPC5/7 register.
5. Clear the SPIROV bit (SPIxSTAT<6>).
6. Write the selected configuration settings to the SPIxCON register.
7. Enable SPI operation by setting the ON bit (SPIxCON<15>).
8. Transmission (and reception) will start as soon as the master provides the serial clock.

> **Note 1:** 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.
>
> **2:** The SPIxSR register cannot be written into directly by the user. All writes to the SPIxSR register are performed through the SPIxBUF register.

FIGURE 17-9: SPI SLAVE, FRAME MASTER CONNECTION DIAGRAM



Note 1: In Framed SPI modes, the 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 SSx pin is not optional).

17.2.4.10 SPI Slave Mode and Frame Slave Mode

This Framed SPI mode is enabled by setting bits MSTEN (SPIxCON<5>) to '0', FRMEN (SPIxCON<31>) to '1', and FRMSYNC (SPIxCON<30>) to '1'. Therefore, both the SCKx and SSx pins will be inputs. The SSx pin will be sampled on the sample edge of the SPI clock. When SSx is sampled active, high or low depending on bit, FRMPOL (SPIxCON<29>), 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 17-10.

The SDO pins is an output and the SCK, SDI and SSx pins are inputs. Setting the control bit, DISSDO (SPIxCON<12>), disables transmission at the SDO pin if Receive Only mode of operation is desired.

Refer to Table 17-7.

The SDI pin must be configured to properly sample the data received from the slave device by configuring the sample bit, SMP (SPIxCON<9>).

Refer to timing diagram shown in Figure 17-7 to determine the appropriate settings.

17.2.4.11 Slave SPIxCON Configuration

The following bits must be configured as shown for the Slave mode of operation when configuring the SPIxCON register:

- Enable Slave Mode – MSTEN (SPIxCON<5>) = 0
- Enable Framed SPI support – FRMEN (SPIxCON<31>) = 1
- Select SSx pin as Frame Slave (input) – FRMSYNC(SPIxCON<30>) = 1

The remaining bits are shown with example configurations and may be configured as desired:

- Enable module control of SDO pin – DISSDO (SPIxCON<12>) = 0
- Configure SCK clock polarity to Idle high – CKP (SPIxCON<6>) = 1
- Configure SCK clock edge transition from Idle to active – CKE (SPIxCON<8>) = 0
- Select SSx active-low pin polarity – FRMPOL (SPIxCON<29>) = 0
- Select 16-bit data width – MODE<32,16> (SPIxCON<11:10>) = '01'
- Sample data input at middle – SMP (SPIxCON<9>) = 0
- Enable SPI module when CPU Idle – SIDL (SPIxCON<13>) = 0

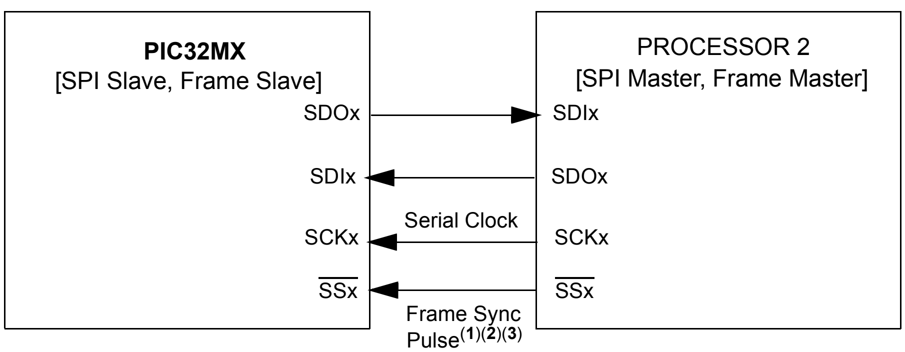
17.2.4.12 Framed Slave Mode Initialization

The following steps are used to set up the SPI module for the Slave mode of operation:

1. If interrupts are used, disable the SPI interrupts in the respective IEC0/1 register.
2. Stop and reset the SPI module by clearing the ON bit.
3. Clear the receive buffer.
4. If using interrupts, the following additional steps are performed:
   - Clear the SPIx interrupt flags/events in the respective IFS0/1 register.
   - Set the SPIx interrupt enable bits in the respective IEC0/1 register.
   - Write the SPIx interrupt priority and subpriority bits in the respective IPC5/7 register.
5. Clear the SPIROV bit (SPIxSTAT<6>).
6. Write the selected configuration settings to the SPIxCON register.
7. Enable SPI operation by setting the ON bit (SPIxCON<15>).
8. Transmission (and reception) will start as soon as the master provides the serial clock.

> **Note 1:** 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.
>
> **2:** The SPIxSR register cannot be written into directly by the user. All writes to the SPIxSR register are performed through the SPIxBUF register.
>
> **3:** 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.

FIGURE 17-10: SPI SLAVE, FRAME SLAVE CONNECTION DIAGRAM



Note 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: Slave Select is not available when using Frame mode as a slave device.

17.2.5 SPI MASTER MODE CLOCK FREQUENCY

In Master mode, the SPI module clock source is the peripheral bus clock (PBCLK) and the SCK clock baud rate is derived from the PBCLK clock and the SPIxBRG register.

Equation 17-1 defines the SCKx clock frequency as a function of the SPIxBRG register settings.

Note that the maximum possible baud rate is $F_{PB}/2$ (SPIXBRG = 0) and the minimum possible baud rate is F_{PB} /1024.

Sample SPI clock frequencies are shown in the table Table 17-6.

EQUATION 17-1: SPI CLOCK FREQUENCY

$$F_{SCK} = \frac{F_{PB}}{2 * (SPIxBRG+1)}$$

.

> **Note:** The SCKx signal clock is not free running for nonframed SPI modes. It will only run for 8, 16 or 32 pulses when the SPIxBUF is loaded with data. It will however, be continuous for Framed modes.

TABLE 17-6: SAMPLE SCKX FREQUENCIES

SPIxBRG setting	0	15	31	63	85	127
F_{PB} = 50 MHz	25.00 MHz	1.56 MHz	781.25 KHz	390.63 KHz	290.7 KHz	195.31 KHz
F_{PB} = 40 MHz	20.00 MHz	1.25 MHz	625.00 KHz	312.50 KHz	232.56 KHz	156.25 KHz
F_{PB} = 25 MHz	12.50 MHz	781.25 KHz	390.63 KHz	195.31 KHz	145.35 KHz	97.66 KHz
F_{PB} = 20 MHz	10.00 MHz	625.00 KHz	312.50 KHz	156.25 KHz	116.28 KHz	78.13 KHz
F_{PB} = 10 MHZ	5.00 MHz	312.50 KHz	156.25 KHz	78.13 KHz	58.14 KHz	39.06 KHz

17.3 SPI 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, signalled by SPI1RXIF (IFS0<25>), SPI2RXIF (IFS1<7>). This event occurs when there is new data assembled in the SPIxBUF receive buffer.
- Transmit buffer empty interrupts, signalled by SPI1TXIF (IFS0<24>), SPI2TXIF (IFS1<6>). This event occurs when there is space available in the SPIxBUF transmit buffer and new data can be written.
- Receive buffer overflow interrupts, signalled by SPI1EIF (IFS0<23>), SPI2EIF(IFS1<5>). This event occurs when there is an overflow condition for the SPIxBUF receive buffer, i.e., new receive data assembled but the previous one is not read.

An SPI device is enabled as a source of interrupts via the respective SPI interrupt enable bits:

- SPI1RXIE (IEC0<25>) and SPI2RXIE (IEC1<7>)
- SPI1TXIE (IEC0<24>) and SPI2TXIE (IEC1<6>)
- SPI1EIE (IEC0<23>) and SPI2EIE (IEC1<5>)

The interrupt priority level bits and interrupt subpriority level bits must be also be configured:

- SPI1IP (IPC5<28:26>), SPI1IS (IPC5<25:24>)
- SPI2IP (IPC7<28:26>), SPI2IS (IPC7<25:24>)

In addition to enabling the SPI interrupts, an Interrupt Service Routine, ISR, is required. Example 17-3 is a partial code example of an ISR.

Note:	It is the user's responsibility to clear the corresponding interrupt flag bit before returning from an ISR.

EXAMPLE 17-3: SPI INITIALIZATION WITH INTERRUPTS ENABLED

```
/*
    The following code example illustrates an SPI1 interrupt configuration.
    When the SPI1 interrupt is generated, the cpu will jump to the vector assigned to SPI1
    interrupt.
    It assumes that none of the SPI1 input pins are shared with an analog input.
    If so, the AD1PCFG and corresponding TRIS registers have to be properly configured.
*/

    int rData;

    IEC0CLR=0x03800000;              // disable all SPI interrupts
    SPI1CON = 0;                     // Stops and resets the SPI1.
    rData=SPI1BUF;                   // clears the receive buffer
    IFS0CLR=0x03800000;              // clear any existing event
    IPC5CLR=0x1f000000;              // clear the priority
    IPC5SET=0x0d000000;              // Set IPL=3, subpriority 1
    IEC0SET=0x03800000;              // Enable Rx, Tx and Error interrupts

    SPI1BRG=0x1;                     // use FPB/4 clock frequency
    SPI1STATCLR=0x40;                // clear the Overflow
    SPI1CON=0x8220;                  // SPI ON, 8 bits transfer, SMP=1, Master Mode
```

 Advance Information

EXAMPLE 17-4: SPI1 ISR

```
/*
    The following code example demonstrates a simple interrupt service routine for SPI1
    interrupts. The user’s code at this vector should perform any application specific operations
    and must clear the SPI1 interrupt flags before exiting.
*/

void __ISR(_SPI1_VECTOR, IPL7) __SPI1Interrupt(void)
{
    // ... perform application specific operations in response to the interrupt

    IFS0CLR = 0x03800000;              // Be sure to clear the SPI1 interrupt flags
                                       // before exiting the service routine.
}
```

Note: The SPI1 ISR code example shows MPLAB® C32 C Compiler specific syntax. Refer to your compiler manual regarding support for ISRs.

17.4 I/O Pin Control

Enabling the SPI modules will configure the I/O pin direction as defined by the SPI control bits (see Table 17-7). The port TRIS and LATCH registers will be overridden.

TABLE 17-7: I/O PIN CONFIGURATION FOR USE WITH SPI MODULES

Required Settings for Module Pin Control							
IO Pin Name	**Required**	**Module Control(3)**	**Bit Field(3)**	**TRIS(4)**	**Pin Type**	**Buffer Type**	**Description**
SCK1, SCK2	Yes	ON and MSTEN	—	X	O	CMOS	SPI1, SPI2 module Clock Output in Master Mode.
SCK1, SCK2	Yes	ON and $\overline{\text{MSTEN}}$	—	X(5)	I	CMOS	SPI1, SPI2 module Clock Input in Slave Mode.
SDI1, SDI2	Yes	ON	—	X(5)	I	CMOS	SPI1, SPI2 module Data Receive pin.
SDO1, SDO2	Yes(1)	ON	DISSDO	X	O	CMOS	SPI1, SPI2 module Data Transmit pin.
$\overline{\text{SS1}}$, $\overline{\text{SS2}}$	Yes(2)	ON and $\overline{\text{FRMEN}}$ and $\overline{\text{MSTEN}}$	SSEN	X(5)	I	CMOS	SPI1, SPI2 module Slave Select Control pin.
$\overline{\text{SS1}}$, $\overline{\text{SS2}}$	Yes	ON and FRMEN and FRMSYNC	—	X(5)	I	CMOS	SPI1, SPI2 Frame Sync Pulse input in Frame Mode.
$\overline{\text{SS1}}$, $\overline{\text{SS2}}$	Yes	ON and FRMEN and $\overline{\text{FRMSYNC}}$	—	X	O	CMOS	SPI1,SPI2 Frame Sync Pulse output in Frame Mode.

Legend: CMOS = CMOS compatible input or output; ST = Schmitt Trigger input withCMOS levels; I = Input; O = Output; X = Don't Care

Note 1: The SDO pins are only required when SPI data output is needed. Otherwise, these pins can be used for general purpose I/O and require the user to set the corresponding TRIS control register bits.

2: The Slave Select pins are only required when a select signal to the slave device is needed. Otherwise, these pins can be used for general purpose I/O and require the user to set the corresponding TRIS control register bits.

3: These bits are contained in the SPIxCON register.

4: The setting of the TRIS bit is irrelevant.

5: If the input pin is shared with an analog input, then the AD1PCFG and the corresponding TRIS register have to be properly set to configure this input as digital.

18.0 INTER-INTEGRATED CIRCUIT (I^2C™)

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

The Inter-Integrated Circuit (I^2C) module provides complete hardware support for both Slave and Multi-Master modes of the I^2C serial communication standard. Figure 18-1 shows the I^2C module block diagram.

The PIC32MX Family devices have up to two I^2C interface modules, denoted as I2C1 and I2C2. Each I^2C module has a 2-pin interface: the SCLx pin is clock and the SDAx pin is data.

Each I^2C module 'I2Cx' (x = 1 or 2) offers the following key features:

- I^2C Interface Supporting both Master and Slave Operation.
- I^2C Slave Mode Supports 7 and 10-Bit Address.
- I^2C Master Mode Supports 7 and 10-Bit Address.
- I^2C Port allows Bidirectional Transfers between Master and Slaves.
- Serial Clock Synchronization for I^2C Port can be used as a Handshake Mechanism to Suspend and Resume Serial Transfer (SCLREL control).
- I^2C Supports Multi-master Operation; Detects Bus Collision and Arbitrates Accordingly.
- Provides Support for Address Bit Masking.

18.1 Operating Modes

The hardware fully implements all the master and slave functions of the I^2C Standard and Fast mode specifications, as well as 7 and 10-bit addressing.

The I^2C module can operate either as a slave or a master on an I^2C bus.

The following types of I^2C operation are supported:

- I^2C Slave Operation with 7 or 10-Bit Address
- I^2C Master Operation with 7 or 10-Bit Address

For details about the communication sequence in each of these modes, please refer to the "*PIC32MX Family Reference Manual*" (DS61132).

18.2 I^2C Registers

The I2CxCON register allows control of the module's operation. The I2CxCON register is readable and writable. I2CxSTAT register contains status flags indicating the module's state during operation.

I2CxRCV is the receive register. When the incoming data is shifted completely, it is moved to the I2CxRCV register. I2CxTRN is the transmit register to which bytes are written during a transmit operation.

The I2CxADD register holds the slave address. A Status bit, ADD10, indicates 10-Bit Addressing mode. The I2CxBRG acts as the Baud Rate Generator (BRG) reload value.

In receive operations, I2CxRSR and I2CxRCV together form a double-buffered receiver. When I2CxRSR receives a complete byte, it is transferred to I2CxRCV and an interrupt pulse is generated. The I2CxRSR shift register is not directly accessable to the programmer.

18.3 I^2C Interrupts

The I^2C module generates three interrupt signals: Slave Interrupt (I2CxSIF), Master Interrupt (I2CxMIF) and Bus Collision Interrupt (I2CxBIF).

18.4 Baud Rate Generator

In I^2C Master mode, the reload value for the Baud Rate Generator (BRG) resides in the I2CxBRG register. When the BRG is loaded with this value, the BRG counts down to '0' and stops until another reload has taken place. If clock arbitration is taking place, for instance, the BRG is reloaded when the SCLx pin is sampled high.

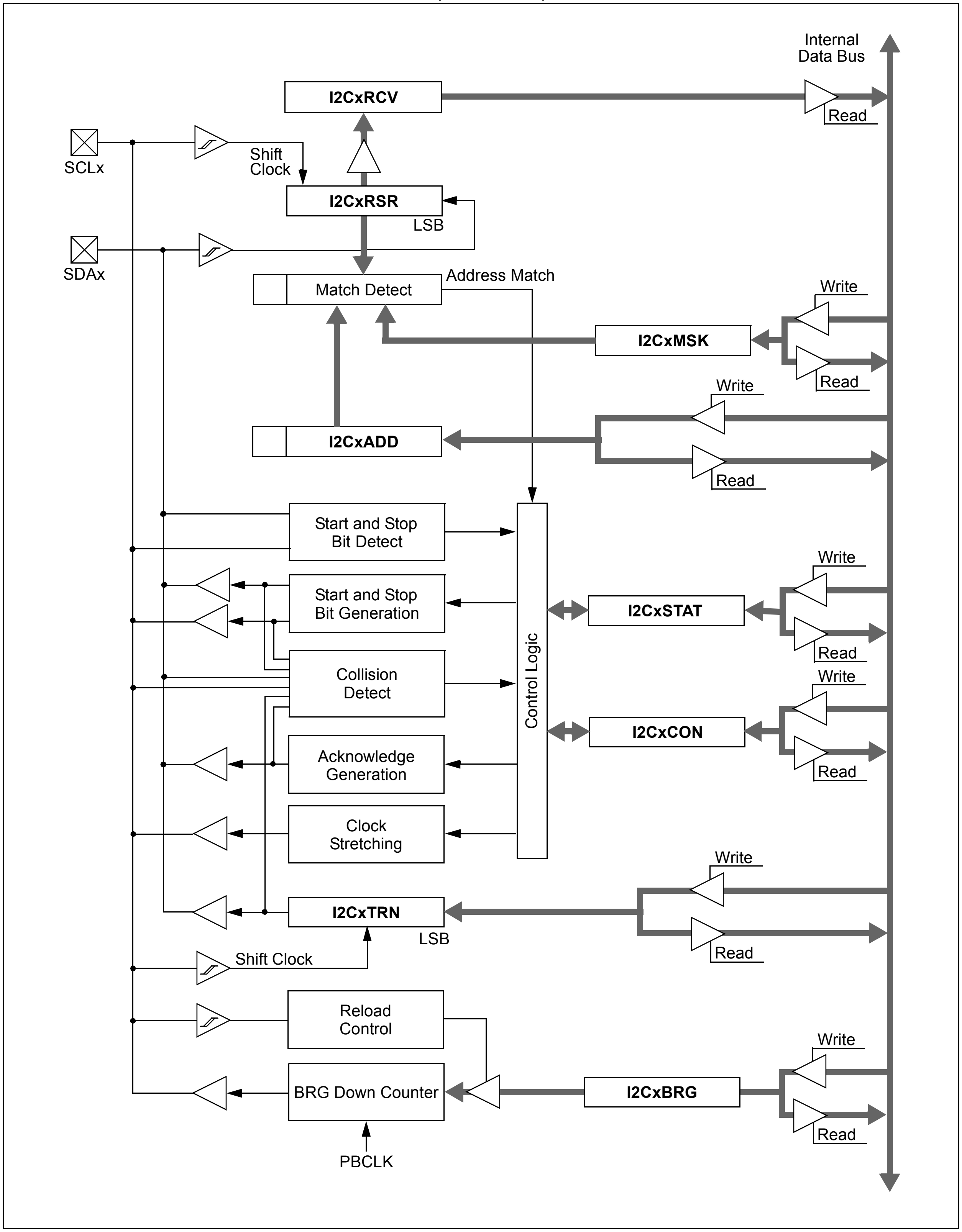
As per the I^2C standard, FSCL may be 100 kHz or 400 kHz. However, the user can specify any baud rate up to 1 MHz. I2CxBRG values of '0' or '1' are illegal.

EQUATION 18-1: SERIAL CLOCK RATE

$$I2C_xBRG = \left[\frac{PBCLK}{FSCL \times 2}\right] - 2$$

PBCLK is the peripheral clock speed. FSCL is the desired I^2C bus speed.

FIGURE 18-1: I²C™ BLOCK DIAGRAM (x = 1 OR 2)



 Advance Information

18.5 I²C Module Addresses

The I2CxADD register contains the Slave mode addresses. The register is a 10-bit register.

If the A10M bit (I2CxCON<10>) is '0', the address is interpreted by the module as a 7-bit address. When an address is received, it is compared to the 7 Least Significant bits of the I2CxADD register.

If the A10M bit is '1', the address is assumed to be a 10-bit address. When the first address byte is received, it will be compared with the binary value, '11110 A9 A8 R/$\overline{W}$ = 0 (where A9 and A8 are Most Significant bits of the 10-bit address stored in I2CxADD). If that value matches, the next address byte will be compared with the Least Significant 8 bits of I2CxADD, as specified in the 10-bit addressing protocol.

TABLE 18-1: 7-BIT I²C™ SLAVE ADDRESSES SUPPORTED BY PIC32MX FAMILY

0x00	General call address or Start byte
0x01-0x03	Reserved
0x04-0x07	Hs mode Master codes
0x08-0x77	Valid 7-bit addresses
0x78-0x7b	10-bit address upper byte
0x7c-0x7f	Reserved

18.6 Slave Address Masking

The I2CxMSK register (Register 18-4) designates address bit positions as "don't care" (= 1) for both 7-bit and 10-Bit Addressing modes. Setting a particular bit location (= 1) in the I2CxMSK register, causes the slave module to respond, whether the corresponding address bit value is a '0' or '1'. For example, when I2CxMSK is set to '00110000', the slave module will detect both addresses, '0000000' and '00100000'.

18.7 Strict Addressing Support

The control bit, STRICT, enables the module to support the strict addressing. It enables the module to enforce all reserved addresses if they fall within the reserved address table. If the user wants to enforce the reserved address space, the STRICT (I2CxCON<11>) bit must be set to '1'. Once the bit is set, the device will not acknowledge reserved addresses, regardless of the address mask settings.

18.8 General Call Address Support

The general call address is used to address all devices. When this address is used, all devices should, in theory, respond with an Acknowledgement.

The general call address is one of eight addresses reserved for specific purposes by the I²C protocol. It consists of all '0's with R/W = 0.

The general call address is recognized when the General Call Enable (GCEN) bit is set (I2CxCON<7> = 1). When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the I2CxRCV to determine if the address was device-specific or a general call address. Upon detection of general call address, GCSTAT (I2CxSTAT<9>) bit is set. This method is available in both 7-Bit and 10-Bit Addressing modes.

18.9 Automatic Clock Stretch

In Slave modes, the module can synchronize buffer reads and writes to the master device by clock stretching.

18.9.1 TRANSMIT CLOCK STRETCHING

Both 10-Bit and 7-Bit Transmit modes implement clock stretching by asserting the SCLREL bit after the falling edge of the ninth clock, if the TBF bit is cleared, indicating the buffer is empty.

In Slave Transmit modes, clock stretching is always performed, irrespective of the STREN bit. The user's ISR must set the SCLREL bit before transmission is allowed to continue. By holding the SCLx line low, the user has time to service the ISR and load the contents of the I2CxTRN before the master device can initiate another transmit sequence.

18.9.2 RECEIVE CLOCK STRETCHING

The STREN bit in the I2CxCON register can be used to enable clock stretching in Slave Receive mode. When the STREN bit is set, the SCLx pin will be held low at the end of each data receive sequence.

The user's ISR must set the SCLREL bit before reception is allowed to continue. By holding the SCLx line low, the user has time to service the ISR and read the contents of the I2CxRCV before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring.

18.10 Software Controlled Clock Stretching (STREN = 1)

When the STREN bit is '1', the SCLREL bit may be cleared by software to allow software to control the clock stretching.

If the STREN bit is '0', a software write to the SCLREL bit will be disregarded and have no effect on the SCLREL bit.

18.11 Slope Control

The I²C standard requires slope control on the SDAx and SCLx signals for Fast mode (400 kHz). The control bit, DISSLW, enables the user to disable slew rate control if desired. It is necessary to disable the slew rate control for 1 MHz mode.

18.12 Clock Arbitration

Clock arbitration occurs when the master deasserts the SCLx pin (SCLx allowed to go high by external pull-up resistors) during any receive, transmit or Restart/Stop condition. When the SCLx pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCLx pin is actually sampled high. When the SCLx pin is sampled high, the Baud Rate Generator is reloaded with the contents of I2CxBRG and 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.

18.13 Multi-Master Communication, Bus Collision and Bus Arbitration

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDAx pin, arbitration takes place when the master outputs a '1' on SDAx by letting SDAx float high while another master asserts a '0'. When the SCLx pin floats high, data should be stable. If the expected data on SDAx is a '1' and the data sampled on the SDAx pin = 0, then a bus collision has taken place. The master will set the I²C master events interrupt flag and reset the master portion of the I²C port to its Idle state.

FIGURE 18-2: TYPICAL I²C™ INTERCONNECTION BLOCK DIAGRAM

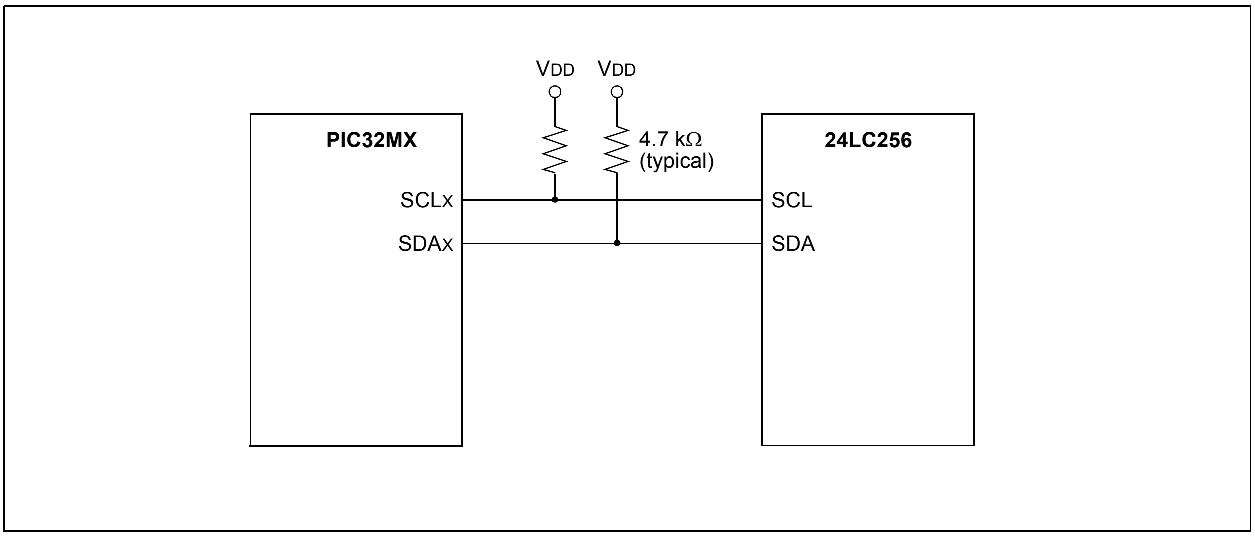


TABLE 18-2: 32-BIT REGISTER SUMMARY

Virtual Address	Name		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
0000h	I2CxCON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	SCLREL	STRICT	A10M	DISSLW	SMEN
		7:0	GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
0004	I2CxCONCLR	31:0	Clears selected bits of I2CxCON, read yields undefined value							
0008h	I2CxCONSET	31:0	Sets selected bits of I2CxCON, read yields undefined value							
000Ch	I2CxCONINV	31:0	Inverts selected bits of I2CxCON, read yields undefined value							
0010h	I2CxSTAT	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ACKSTAT	TRSTAT	—	—	—	BCL	GCSTAT	ADD10
		7:0	IWCOL	I2COV	D/A	P	S	R/W	RBF	TBF
0014h	I2CxSTATCLR	31:0	Clears selected bits of I2CxSTAT, read yields undefined value							
0018h	I2CxSTATSET	31:0	Sets selected bits of I2CxSTAT, read yields undefined value							
001Ch	I2CxSTATINV	31:0	Inverts selected bits of I2CxSTAT, read yields undefined value							
0020h	I2CxADD	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	ADD<9:8>	
		7:0	ADD<7:0>							
0024h	I2CxADDCLR	31:0	Clears selected bits of I2CxADD, read yields undefined value							
0028h	I2CxADDSET	31:0	Sets selected bits of I2CxADD, read yields undefined value							
002Ch	I2CxADDINV	31:0	Inverts selected bits of I2CxADD, read yields undefined value							
0030h	I2CxMSK	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	MSK<9:8>	
		7:0	MSK<7:0>							
0034h	I2CxMSKCLR	31:0	Clears selected bits of I2CxMSK, read yields undefined value							
0038h	I2CxMSKSET	31:0	Sets selected bits of I2CxMSK, read yields undefined value							
003Ch	I2CxMSKINV	31:0	Inverts selected bits of I2CxMSK, read yields undefined value							
0040h	I2CxBRG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	I2CxBRG<11:8>	
		7:0	I2CxBRG<7:0>							
0044h	I2CxBRGCLR	31:0	Clears selected bits of I2CxBRG, read yields undefined value							
0048h	I2CxBRGSET	31:0	Sets selected bits of I2CxBRG, read yields undefined value							
004Ch	I2CxBRGINV	31:0	Inverts selected bits of I2CxBRG, read yields undefined value							
0050h	I2CxTRN	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	I2CTXDATA							
0054h	I2CxTRNCLR	31:0	Clears selected bits of I2CxTRN, read yields undefined value							
0058h	I2CxTRNSET	31:0	Sets selected bits of I2CxTRN, read yields undefined value							
005Ch	I2CxTRNINV	31:0	Inverts selected bits of I2CxTRN, read yields undefined value							
0060h	I2CxRCV	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	I2CRXDATA							

REGISTER 18-1: I2CxCON: I^2C™ CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
ON	FRZ	I2CSIDL	SCLREL	STRICT	A10M	DISSLW	SMEN
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** I^2C Enable bit
1 = Enables the I^2C module and configures the SDA and SCL pins as serial port pins
0 = Disables I^2C module; all I^2C pins are controlled by PORT functions

bit 14 **FRZ:** Freeze in Debug Mode Control bit (read/write only in Debug mode; otherwise read as ‘0’)
1 = Freeze module operation when in Debug mode
0 = Do not freeze module operation when in Debug mode

bit 13 **I2CSIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 12 **SCLREL:** SCL Release Control bit
In I^2C Slave mode only
Module Reset and (ON = 0) sets SCLREL = 1
<u>If STREN = 0:</u>
1 = Release clock
0 = Force clock low (clock stretch)

Note: Automatically cleared to ‘0’ at beginning of slave transmission.

<u>If STREN = 1:</u>
1 = Release clock
0 = Holds clock low (clock stretch). User may program this bit to ‘0’ to force a clock stretch at the next SCL low.

Note: Automatically cleared to ‘0’ at beginning of slave transmission; automatically cleared to ‘0’ at end of slave reception.

bit 11 **STRICT:** Strict I^2C Reserved Address Rule Enable bit
1 = Strict reserved addressing is enforced. Device doesn’t respond to reserved address space or generate addresses in reserved address space.
0 = Strict I^2C Reserved Address Rule not enabled

REGISTER 18-1: I2CxCON: I²C™ CONTROL REGISTER (CONTINUED)

bit 10	**A10M:** 10-bit Slave Address Flag bit 1 = I2CxADD is a 10-bit slave address 0 = I2CADD is a 7-bit slave address
bit 9	**DISSLW:** Slew Rate Control Disable bit 1 = Slew rate control disabled for Standard Speed mode (100 kHz); also disabled for 1 MHz mode 0 = Slew rate control enabled for High-Speed mode (400 kHz)
bit 8	**SMEN:** SMBus Input Levels Disable bit 1 = Enable input logic so that thresholds are compliant with SMBus specification 0 = Disable SMBus specific inputs
bit 7	**GCEN:** General Call Enable bit In I²C Slave mode only 1 = Enable interrupt when a general call address is received in I2CSR. Module is enabled for reception 0 = General call address disabled
bit 6	**STREN:** SCL Clock Stretch Enable bit In I²C Slave mode only; used in conjunction with SCLREL bit. 1 = Enable clock stretching 0 = Disable clock stretching
bit 5	**ACKDT:** Acknowledge Data bit In I²C Master mode only; applicable during master receive. Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive. 1 = A NACK is sent 0 = $\overline{\text{ACK}}$ is sent
bit 4	**ACKEN:** Acknowledge Sequence Enable bit In I²C Master mode only; applicable during master receive 1 = Initiate Acknowledge sequence on SDA and SCL pins, and transmit ACKDT data bit; cleared by module 0 = Acknowledge sequence idle
bit 3	**RCEN:** Receive Enable bit In I²C Master mode only. 1 = Enables Receive mode for I²C, automatically cleared by module at end of 8-bit receive data byte 0 = Receive sequence not in progress
bit 2	**PEN:** Stop Condition Enable bit In I²C Master mode only. 1 = Initiate Stop condition on SDA and SCL pins; cleared by module 0 = Stop condition idle
bit 1	**RSEN:** Restart Condition Enable bit In I²C Master mode only. 1 = Initiate Restart condition on SDA and SCL pins; cleared by module 0 = Restart condition idle
bit 0	**SEN:** Start Condition Enable bit In I²C Master mode only. 1 = Initiate Start condition on SDA and SCL pins; cleared by module 0 = Start condition idle

REGISTER 18-2: I2CxSTAT: I²C STATUS REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R-0	R-0	U-0	U-0	U-0	R/W-0	R-0	R-0
ACKSTAT	TRSTAT	—	—	—	BCL	GCSTAT	ADD10
bit 15							bit 8

R/W-0	R/W-0	R-0	R/W-0	R/W-0	R-0	R-0	R-0
IWCOL	I2COV	$D/\overline{A}$	P	S	$R/\overline{W}$	RBF	TBF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ACKSTAT:** Acknowledge Status bit
In both I²C Master and Slave modes; applicable to both transmit and receive.
1 = Acknowledge was not received
0 = Acknowledge was received

bit 14 **TRSTAT:** Transmit Status bit
In I²C Master mode only; applicable to Master Transmit mode.
1 = Master transmit is in progress (8 bits + $\overline{\text{ACK}}$)
0 = Master transmit is not in progress

bit 13-11 **Unimplemented:** Read as ‘0’

bit 10 **BCL:** Master Bus Collision Detect bit
Cleared when the I²C module is disabled (ON = 0).
1 = A bus collision has been detected during a master operation
0 = No collision has been detected

bit 9 **GCSTAT:** General Call Status bit
Cleared after Stop detection.
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.
1 = 10-bit address was matched
0 = 10-bit address was not matched

bit 7 **IWCOL:** Write Collision Detect bit
1 = An attempt to write the I2CxTRN register collided because the I²C module is busy. Must be cleared in software.
0 = No collision

REGISTER 18-2: I2CxSTAT: I²C STATUS REGISTER (CONTINUED)

bit 6	**I2COV:** I²C Receive Overflow Status bit 1 = A byte is received while the I2CxRCV register is still holding the previous byte. I2COV is a “don’t care” in Transmit mode. Must be cleared in software. 0 = No overflow
bit 5	**D/$\overline{\text{A}}$:** Data/Address bit Valid only for Slave mode operation. 1 = Indicates that the last byte received or transmitted was data 0 = Indicates that the last byte received or transmitted was address
bit 4	**P:** Stop bit Updated when Start, Reset or Stop detected; cleared when the I²C module is disabled (ON = 0). 1 = Indicates that a Stop bit has been detected last 0 = Stop bit was not detected last
bit 3	**S:** Start bit Updated when Start, Reset or Stop detected; cleared when the I²C module is disabled (ON = 0). 1 = Indicates that a start (or restart) bit has been detected last 0 = Start bit was not detected last
bit 2	**R/$\overline{\text{W}}$:** Read/Write Information bit Valid only for Slave mode operation. 1 = Read – indicates data transfer is output from slave 0 = Write – indicates data transfer is input to slave
bit 1	**RBF:** Receive Buffer Full Status bit 1 = Receive complete; I2CxRCV is full 0 = Receive not complete; I2CxRCV is empty
bit 0	**TBF:** Transmit Buffer Full Status bit 1 = Transmit in progress; I2CxTRN is full (8-bits of data) 0 = Transmit complete; I2CxTRN is empty

REGISTER 18-3: I2CxADD: I^{2}C SLAVE ADDRESS REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	ADD<9:8>	
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADD<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-10 **Unimplemented:** Read as ‘0’

bit 9-0 **ADD<9:0>:** I^{2}C Slave Device Address bits
Either Master or Slave mode.

REGISTER 18-4: I2CxMSK: I^2C ADDRESS MASK REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	MSK<9:8>	
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
MSK<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-10 **Unimplemented:** Read as ‘0’

bit 9-0 **MSK<9:0>:** I^2C Address Mask bits
1 = Forces a “don’t care” in the particular bit position on the incoming address match sequence
0 = Address bit position must match the incoming I^2C address match sequence

Note: MSK<9:8> and MSK<0> are only used in I^2C 10-Bit mode.

REGISTER 18-5: I2CxBRG: I²C BAUD RATE GENERATOR REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	I2CxBRG<11:8>			
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
I2CxBRG<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-12 **Unimplemented:** Read as ‘0’

bit 11-0 **I2CxBRG<11:0>:** I²C Baud Rate Generator Value bits
A divider function of the Peripheral Clock.

REGISTER 18-6: I2CxTRN: I²C TRANSMIT DATA REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
I2CTXDATA<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-8 **Unimplemented:** Read as ‘0’

bit 7-0 **I2CTXDATA<7:0>:** I²C Transmit Data Buffer bits

REGISTER 18-7: I2CXRCV: I^{2}C RECEIVE DATA REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
I2CRXDATA<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-8 **Unimplemented:** Read as ‘0’

bit 7-0 **I2CRXDATA<7:0>:** I^{2}C Receive Data Buffer bits

19.0 UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER (UART)

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *“PIC32MX Family Reference Manual”* (DS61132) for a detailed description of this peripheral.

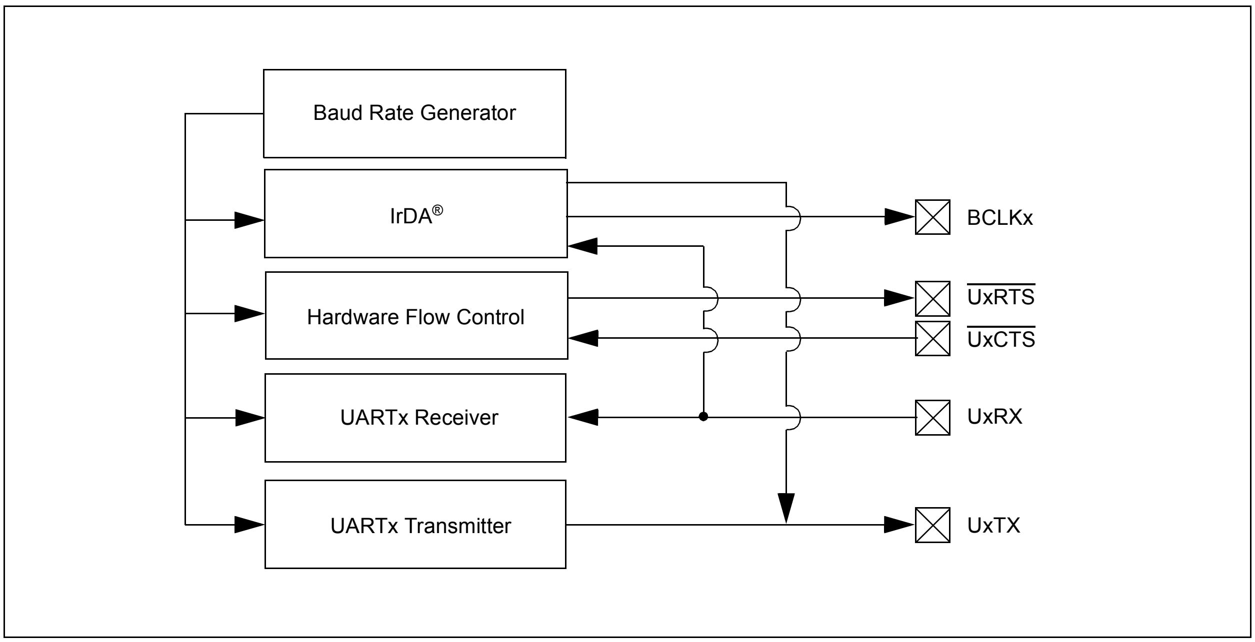
The Universal Asynchronous Receiver Transmitter (UART) module is one of the serial I/O modules available in PIC32MX family devices. The UART is a full-duplex, asynchronous communication channel that communicates with peripheral devices and personal computers through protocols such as RS-232, RS-485, LIN 1.2 and IrDA®. The module also supports the hardware flow control option, with $\overline{\text{UxCTS}}$ and $\overline{\text{UxRTS}}$ pins, and also includes the IrDA encoder and decoder.

The primary features of the UART module are:

- Full-duplex, 8-bit or 9-bit data transmission
- Even, odd or no parity options (for 8-bit data)
- One or two Stop bits
- Hardware auto-baud feature
- Hardware flow control option
- Fully integrated Baud Rate Generator (BRG) with 16-bit prescaler
- Baud rates ranging from 47.7 bps to 3.125 Mbps at 50 MHz
- 4-level-deep First-In-First-Out (FIFO) Transmit Data Buffer
- 4-level-deep FIFO Receive Data Buffer
- Parity, framing and buffer overrun error detection
- Support for interrupt only on address detect (9th bit = `1`)
- Separate transmit and receive interrupts
- Loopback mode for diagnostic support
- LIN 1.2 protocol support
- IrDA encoder and decoder with 16x baud clock output for external IrDA encoder/decoder support

Figure 19-1 shows a simplified block diagram of the UART.

FIGURE 19-1: UART SIMPLIFIED BLOCK DIAGRAM



19.1 UART Registers

TABLE 19-1: UART1 SFR SUMMARY

Virtual Address	Name		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
BF80_6000	U1MODE	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	IREN	RTSMD	—	UEN<1:0>	
		7:0	WAKE	LPBACK	ABAUD	RXINV	BRGH	PDSEL<2:0>		STSEL
BF80_6004	U1MODECLR	31:0	Write clears selected bits in U1MODE, read yields undefined value							
BF80_6008	U1MODESET	31:0	Write sets selected bits in U1MODE, read yields undefined value							
BF80_600C	U1MODEINV	31:0	Write inverts selected bits in U1MODE, read yields undefined value							
BF80_6010	U1STA	31:24	—	—	—	—	—	—	—	ADM_EN
		23:16	ADDR<7:0>							
		15:8	UTXISEL<1:0>		TXINV	RXEN	TXBRK	TXEN	UTXBF	TRMT
		7:0	URXISEL<1:0>		ADDEN	RIDLE	PERR	FERR	OERR	RXDA
BF80_6014	U1STACLR	31:0	Write clears selected bits in U1STA, read yields undefined value							
BF80_6018	U1STASET	31:0	Write sets selected bits in U1STA, read yields undefined value							
BF80_601C	U1STAINV	31:0	Write inverts selected bits in U1STA, read yields undefined value							
BF80_6020	U1TXREG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	TX8
		7:0	Transmit Register							
BF80_6030	U1RXREG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	RX8
		7:0	Receive Register							
BF80_6040	U1BRG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	BRG<15:8>							
		7:0	BRG<7:0>							
BF80_6044	U1BRGCLR	31:0	Write clears selected bits in U1BRG, read yields undefined value							
BF80_6048	U1BRGSET	31:0	Write sets selected bits in U1BRG, read yields undefined value							
BF80_604C	U1BRGINV	31:0	Write inverts selected bits in U1BRG, read yields undefined value							

TABLE 19-2: UART1 INTERRUPT REGISTER SUMMARY

Virtual Address	Name		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
BF88_1060	IEC0	31:24	—	—	—	U1TXIE	U1RXIE	U1EIE	—	—
BF88_1030	IFS0	31:24	—	—	—	U1TXIF	U1RXIF	U1EIF	—	—
BF88_10F0	IPC6	7:0	—	—	—	U1IP[2:0]			U1IS[1:0]	

Note 1: This summary table contains partial register definitions that only pertain to the UART peripheral. Refer to the PIC32MX Family Reference Manual for a detailed description of these registers.

TABLE 19-3: UART2 SFR SUMMARY

Virtual Address	Name		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
BF80_6200	U2MODE	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	IREN	RTSMD	—	UEN<1:0>	
		7:0	WAKE	LPBACK	ABAUD	RXINV	BRGH	PDSEL<2:0>		STSEL
BF80_6204	U2MODECLR	31:0	Write clears selected bits in U2MODE, read yields undefined value							
BF80_6208	U2MODESET	31:0	Write sets selected bits in U2MODE, read yields undefined value							
BF80_620C	U2MODEINV	31:0	Write inverts selected bits in U2MODE, read yields undefined value							
BF80_6210	U2STA	31:24	—	—	—	—	—	—	—	ADM_EN
		23:16	ADDR<7:0>							
		15:8	UTXISEL<1:0>		TXINV	RXEN	TXBRK	TXEN	UTXBF	TRMT
		7:0	URXISEL<1:0>		ADDEN	RIDLE	PERR	FERR	OERR	RXDA
BF80_6214	U2STACLR	31:0	Write clears selected bits in U2STA, read yields undefined value							
BF80_6218	U2STASET	31:0	Write sets selected bits in U2STA, read yields undefined value							
BF80_621C	U2STAINV	31:0	Write inverts selected bits in U2STA, read yields undefined value							
BF80_6220	U2TXREG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	TX8
		7:0	Transmit Register							
BF80_6230	U2RXREG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	RX8
		7:0	Receive Register							
BF80_6240	U2BRG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	BRG<15:8>							
		7:0	BRG<7:0>							
BF80_6244	U2BRGCLR	31:0	Write clears selected bits in U2BRG, read yields undefined value							
BF80_6248	U2BRGSET	31:0	Write sets selected bits in U2BRG, read yields undefined value							
BF80_624C	U2BRGINV	31:0	Write inverts selected bits in U2BRG, read yields undefined value							

TABLE 19-4: UART2 INTERRUPT REGISTER SUMMARY

Virtual Address	Name		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
0xBF881070	IEC1	15:8	—	—	—	—	—	U2TXIE	U2RXIE	U2EIE
0xBF881040	IFS1	15:8	—	—	—	—	—	U2TXIF	U2RXIF	U2EIF
0xBF881110	IPC8	7:0	—	—	—	U2IP<2:0>			U2IS<1:0>	

REGISTER 19-1: UxMODE: UARTx MODE REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ON	FRZ	SIDL	IREN	RTSMD	—	UEN<1:0>	
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WAKE	LPBACK	ABAUD	RXINV	BRGH	PDSEL<1:0>		STSEL
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: ('0', '1', x = Unknown)		

bit 31-16 **Unimplemented:** Read as '0'

bit 15 **ON:** UARTx Enable bit
- 1 = UARTx is enabled; UARTx pins are controlled by UARTx as defined by UEN<1:0> and UTXEN control bits
- 0 = UARTx is disabled, all UARTx pins are controlled by corresponding PORT TRIS and LAT bits; UARTx power consumption is minimal

bit 14
- 1 = Freeze operation when CPU is in Debug Exception mode
- 0 = Continue operation when CPU is in Debug Exception mode

Note: FRZ: Freeze in Debug Exception Mode bit· FRZ is writable in Debug Exception mode only, it is forced to '0' in normal mode.

bit 13 **SIDL:** Stop in Idle Mode bit
- 1 = Discontinue operation when device enters in Idle mode
- 0 = Continue operation in Idle mode

bit 12 **IREN:** IrDA Encoder and Decoder Enable bit
- 1 = IrDA is enabled
- 0 = IrDA is disabled

bit 11 **RTSMD:** Mode Selection for UxRTS Pin bit
- 1 = UxRTS pin is in simplex mode
- 0 = UxRTS pin is in flow control mode

bit 10 **Unimplemented:** Read as '0

bit 9-8 **UEN<1:0>:** UARTx Enable bits
- 11 = UxTX, UxRX, and UxBCLK pins are enabled and used; UxCTS pin is controlled by port latches
- 10 = UxTX, UxRX, UxCTS, and UxRTS pins are enabled and used
- 01 = UxTX, UxRX and UxRTS pins are enabled and used; UxCTS pin is controlled by port latches
- 00 = UxTX and UxRX pins are enabled and used; UxCTS and UxRTS/UxBCLK pins are controlled by port latches

bit 7 **WAKE:** Enable Wake-up on Start bit Detect During Sleep mode bit
- 1 = Wake-up enabled
- 0 = Wake-up disabled

 Advance Information

REGISTER 19-1: UxMODE: UARTx MODE REGISTER (CONTINUED)

bit 6 **LPBACK:** UARTx Loopback Mode Select bit
1 = Enable Loopback mode
0 = Loopback mode is disabled

bit 5 **ABAUD:** Auto-Baud Enable bit
1 = Enable baud rate measurement on the next character – requires reception of SYNCH character (0x55); cleared by hardware upon completion
0 = Baud rate measurement disabled or completed

bit 4 **RXINV:** Receive Polarity Inversion bit
1 = UxRX idle state is '0'
0 = UxRX idle state is '1'

bit 3 **BRGH:** High Baud Rate Enable bit
1 = High speed mode – 4x baud clock enabled
0 = Standard speed mode – 16x baud clock enabled

bit 2-1 **PDSEL<1:0>:** Parity and Data Selection bits
11 = 9-bit data, no parity
10 = 8-bit data, odd parity
01 = 8-bit data, even parity
00 = 8-bit data, no parity

bit 0 **STSEL:** Stop Selection bit
1 = 2 Stop bits
0 = 1 Stop bit

REGISTER 19-2: UxSTA: UARTx STATUS REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	ADM_EN
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADDR<7:0>							
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-1
UTXISEL<1:0>		TXINV	RXEN	TXBRK	TXEN	TXBF	TRMT
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R-1	R-0	R-0	R/C-0	R-0
URXISEL<1:0>		ADDEN	RIDLE	PERR	FERR	OERR	RXDA
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-25 **Unimplemented:** Read as ‘0’

bit 24 **ADM_EN:** Automatic Address Detect Mode Enable bit
1 = Automatic Address Detect Mode is enabled
0 = Automatic Address Detect Mode is disabled

bit 23-16 **ADDR<7:0>:** Automatic Address Mask bits
When ADM_EN bit is ‘1’, this value defines the bits that are don’t care when comparing incoming address reception

bit 15-14 **UTXISEL<1:0>:** Tx Interrupt Mode Selection bits
11 = Reserved, do not use
10 = Interrupt is generated when the Transmit buffer becomes empty
01 = Interrupt is generated when all characters are transmitted
00 = Interrupt is generated when the Transmit buffer contains at least one empty space

bit 13 **TXINV:** Transmit Polarity Inversion bit
If IrDA mode is disabled (i.e., IREN (UxMOD<12>) is ‘0’)
1 = UxTX idle state is ‘0’
0 = UxTX idle state is ‘1’
If IrDA mode is enabled (i.e., IREN (UxMOD<12>) is ‘1’)
1 = IrDA encoded UxTX idle state is ‘1’
0 = IrDA encoded UxTX idle state is ‘0’

bit 12 **RXEN:** Receiver Enable bit
1 = UARTx receiver is enabled, UxRX pin controlled by UARTx (if ON = 1)
0 = UARTx receiver is disabled, the UxRX pin is ignored by the UARTx module. UxRX pin controlled by PORT.

bit 11 **TXBRK:** Transmit Break bit
1 = Send BREAK on next transmission - Start bit followed by twelve ‘0’ bits, followed by Stop bit; cleared by hardware upon completion
0 = BREAK transmission is disabled or completed

REGISTER 19-2: UxSTA: UARTx STATUS REGISTER (CONTINUED)

bit 10	**TXEN:** Transmit Enable bit 1 = UARTx transmitter enabled, UxTX pin controlled by UARTx (if ON = 1) 0 = UARTx transmitter disabled, any pending transmission is aborted and buffer is reset. UxTX pin controlled by PORT.
bit 9	**UTXBF:** Transmit Buffer Full Status bit (read-only) 1 = Transmit buffer is full 0 = Transmit buffer is not full, at least one more character can be written
bit 8	**TRMT:** Transmit Shift Register is Empty bit (read-only) 1 = Transmit shift register is empty and transmit buffer is empty (the last transmission has completed) 0 = Transmit shift register is not empty, a transmission is in progress or queued in the transmit buffer
bit 7-6	**URXISEL<1:0>:** Receive Interrupt Mode Selection bit 11 = Interrupt flag bit is set when Receive Buffer is full (i.e., has 4 data characters) 10 = Interrupt flag bit is set when Receive Buffer is 3/4 full (i.e., has 3 data characters) 0x = Interrupt flag bit is set when a character is received
bit 5	**ADDEN:** Address Character Detect (bit 8 of received data = 1) 1 = Address Detect mode enabled. If 9-bit mode is not selected, this control bit has no effect. 0 = Address Detect mode disabled
bit 4	**RIDLE:** Receiver Idle bit (read-only) 1 = Receiver is Idle 0 = Data is being received
bit 3	**PERR:** Parity Error Status bit (read-only) 1 = Parity error has been detected for the current character 0 = Parity error has not been detected
bit 2	**FERR:** Framing Error Status bit (read-only) 1 = Framing Error has been detected for the current character 0 = Framing Error has not been detected
bit 1	**OERR:** Receive Buffer Overrun Error Status bit (clear/read-only) 1 = Receive buffer has overflowed 0 = Receive buffer has not overflowed (clearing a previously set OERR bit will reset the receiver buffer and Receive Shift Register (RSR) to an empty state)
bit 0	**RXDA:** Receive Buffer Data Available bit (read-only) 1 = Receive buffer has data, at least one more character can be read 0 = Receive buffer is empty

REGISTER 19-3: UxRXREG: UART RECEIVE REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R-0
—	—	—	—	—	—	—	RX8
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
RX<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-9	**Unimplemented:** Read as ‘0’
bit 8	**RX8:** Data bit 8 of the Received Character (in 9-bit mode)
bit 7-0	**RX<7:0>:** Data bits 7-0 of the Received Character

REGISTER 19-4: UxTXREG: UARTx TRANSMIT REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	W-0
—	—	—	—	—	—	—	TX8
bit 15							bit 8

W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
TX<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-9 **Unimplemented:** Read as ‘0’

bit 8 **TX8:** Data bit 8 of the Character to be Transmitted (in 9-bit mode)

bit 7-0 **TX<7:0>:** Data bits 7-0 of the Character to be Transmitted

19.2 UART Baud Rate Generator (BRG)

The UART module includes a dedicated 16-bit Baud Rate Generator. The BRGx register controls the period of a free-running 16-bit timer. Equation 19-1 shows the formula for computation of the baud rate with BRGH = 0.

EQUATION 19-1: UART BAUD RATE WITH BRGH = 0(1)

$$Baud\ Rate = \frac{F_{PB}}{16 \cdot (UxBRG + 1)}$$

$$UxBRG = \frac{F_{PB}}{16 \cdot Baud\ Rate} - 1$$

Note 1: FPB denotes the peripheral bus clock frequency.

Example 19-1 shows the calculation of the baud rate error for the following conditions:

- FPB = 4 MHz
- Desired Baud Rate = 9600

The maximum possible baud rate with BRGH = 0 is FPB/16.

The minimum possible baud rate is FPB/(16 * 65536).

Equation 19-2 shows the formula for computation of the baud rate with BRGH = 1.

EQUATION 19-2: UART BAUD RATE WITH BRGH = 1(1)

$$Baud\ Rate = \frac{F_{PB}}{4 \cdot (UxBRG + 1)}$$

$$UxBRG = \frac{F_{PB}}{4 \cdot Baud\ Rate} - 1$$

Note 1: FPB denotes the instruction cycle clock frequency.

The maximum possible baud rate with BRGH = 1 is FPB/4 for UxBRG = 0, and the minimum possible baud rate is FPB/(4 * 65536).

Writing a new value to the UxBRG register causes the baud rate counter to be cleared. This ensures that the BRG does not wait for a timer overflow before it generates the new baud rate.

EXAMPLE 19-1: BAUD RATE ERROR CALCULATION (BRGH = 0)

```
Desired Baud Rate      =   Fpb/(16 (UxBRG + 1))
Solving for UxBRG value:
    UxBRG              =   ( (Fpb/Desired Baud Rate)/16) - 1
    UxBRG              =   ((4000000/9600)/16) - 1
    UxBRG              =   [25.042] = 25
Calculated Baud Rate   =   4000000/(16 (25 + 1))
                       =   9615
Error                  =   (Calculated Baud Rate - Desired Baud Rate)
            Desired Baud Rate
                       =   (9615 - 9600)/9600
                       =   0.16%
```


19.3 Transmitting in 8-Bit Data Mode

1. Set up the UART:
   a) Write appropriate values for data, parity and Stop bits.
   b) Write appropriate baud rate value to the UxBRG register.
   c) Set up transmit and receive interrupt enable and priority bits.
2. Enable the UART.
3. Set the UTXEN bit (causes a transmit interrupt).
4. Write data byte to UxTXREG word. The value will be immediately transferred to the Transmit Shift Register (TSR), and the serial bit stream will start shifting out with next rising edge of the baud clock.
5. Alternately, the data byte may be transferred while TXEN = 0, and then the user may set TXEN. This will cause the serial bit stream to begin immediately because the baud clock will start from a cleared state.
6. A transmit interrupt will be generated as per interrupt control bit, UTXISEL<1:0>.

EXAMPLE 19-2: EXAMPLE 8-BIT DATA MODE

```
/* The following code example demonstrates configuring
   UART1 for 8-bit Data Transmit mode.
*/

U1BRG   = #BaudRate;// Set Uart baud rate.
U1MODESET= 0x8000;// Enable Uart for 8-bit Data, no Parity, and 1 Stop bit
U1STASET= 0x1400;// Enable Transmitter and Receiver
```

19.4 Transmitting in 9-Bit Data Mode

1. Set up the UART (as described in Section 19.3).
2. Enable the UART.
3. Set the TXEN bit (causes a transmit interrupt).
4. Write UxTXREG as a 16-bit value only.
5. A write to UxTXREG triggers the transfer of the 9-bit data to the TSR. Serial bit stream will start shifting out with the first rising edge of the baud clock.
6. A transmit interrupt will be generated as per the setting of control bit, UTXISEL<1:0>.

EXAMPLE 19-3: EXAMPLE 9-BIT DATA MODE

```
/* The following code example demonstrates configuring
   UART1 for 9-bit Data Transmit mode.
*/

U1BRG   = #BaudRate;// Set Uart baud rate.
U1MODESET= 0x8006;// Enable Uart for 8-bit Data, no Parity, and 1 Stop bit
U1STASET= 0x1211420;// Enable Address Detect, Set Address = 0x21, Enable Transmitter and
Receiver
```

19.5 Auto-Baud Support

The UART will begin an automatic baud rate measurement sequence whenever a Start bit is received when the Auto-Baud Rate Detect is enabled (ABAUD = 1). This feature is active only while the auto-wake-up is disabled (WAKE = 0). In addition, LPBACK must equal '0' for the auto-baud operation. Following the Start bit, the auto-baud expects to receive an ASCII 'U' (0x55) in order to calculate the proper bit rate. On the 5th UxRX pin rising edge, an accumulated BRG counter value totaling the proper BRG period is transferred to the UxBRG register. The ABAUD bit is automatically cleared.

19.6 Break and Sync Transmit Sequence

The following sequence is performed to send a message frame header that is composed of a Break character, followed by an auto-baud Sync byte. This sequence is typical of a LIN bus master:

1. Configure the UART for the desired mode.
2. Set TXEN and TXBRK to set up the Break character.
3. Load the UxTXREG with a dummy character to initiate transmission (value is ignored).
4. Write '0x55' to UxTXREG to load the Sync character into the transmit FIFO.

After the Break has been sent, the UTXBRK bit is reset by hardware. The Sync character now transmits.

19.7 Receiving in 8-Bit or 9-Bit Data Mode

1. Set up the UART (as described in Section 19.3).
2. Enable the UART.
3. A receive interrupt will be generated when one or more data characters have been received as per interrupt control bit, URXISEL<1:0>.
4. Read the OERR bit to determine if an overrun error has occurred. The OERR bit must be reset in software.
5. Read UxRXREG.

The act of reading the UxRXREG character will move the next character to the top of the receive FIFO, including a new set of PERR and FERR values.

19.8 Operation of $\overline{\text{UxCTS}}$ and $\overline{\text{UxRTS}}$ Control Pins

UARTx Clear to Send ($\overline{\text{UxCTS}}$) and Request to Send ($\overline{\text{UxRTS}}$) are the two hardware controlled pins that are associated with the UART module. These two pins allow the UART to operate in Simplex and Flow Control mode. They are implemented to control the transmission and reception between the Data Terminal Equipment (DTE). The UEN<1:0> bits in the UxMODE register configure these pins.

19.9 Infrared Support

The UART module provides two types of infrared UART support:

- IrDA clock output to support external IrDA encoder and decoder device (legacy module support)
- Full implementation of the IrDA encoder and decoder

19.10 External IrDA Support – IrDA Clock Output

To support external IrDA encoder and decoder devices, the BCLKx pin (same as the $\overline{\text{UxRTS}}$ pin) can be configured to generate the 16x baud clock. With UEN<1:0> = 11, the BCLKx pin will output the 16x baud clock (if the UART module is enabled). It can be used to support the IrDA codec chip.

19.11 Built-in IrDA Encoder and Decoder

The UART has full implementation of the IrDA encoder and decoder as part of the UART module. The built-in IrDA encoder and decoder functionality is enabled using the IREN bit (UxMODE<12>). When enabled (IREN = 1), the receive pin (UxRX) acts as the input from the infrared receiver. The transmit pin (UxTX) acts as the output to the infrared transmitter.

19.12 UART Interrupts

The UART device has the ability to generate interrupts, reflecting the events that occur during data communication. The following types of interrupts can be generated:

- Receiver-data-available interrupts, signalled by U1RXIF (IFS0<27>), U2RXIF (IFS1<9>). This event occurs when there is new data assembled in the UxRXBUF receive buffer.
- Transmitter-buffer-empty interrupts, signalled by U1TXIF (IFS0<28>), U2TXIF (IFS1<10>). This event occurs when there is space available in the UxTXBUF transmit buffer and new data can be written.
- Receiver-buffer-overflow interrupt, signalled by U1EIF (IFS0<26>), U2EIF (IFS1<8>). This event occurs when there is an overflow condition for the UxRXBUF receive buffer, i.e., new receive data assembled but the previous one not read.

A UART device is enabled as a source of interrupts via the respective UART interrupt enable bits:

- U1RXIE (IEC0<27>) and U2RXIE (IEC1<9>)
- U1TXIE (IEC0<28>) and U2TXIE (IEC1<10>)
- U1EIE (IEC0<26>) and U2EIE (IEC1<8>)

The interrupt priority level bits and interrupt subpriority level bits must be also be configured:

- U1IP (IPC6<4:2>), U1IS (IPC6<1:0>)
- U2IP (IPC8<4:2>), U2IS (IPC8<1:0>).

In addition to enabling the UART interrupts, an Interrupt Service Routine (ISR) is required. Below is a partial code example of an ISR.

Note:	It is the user's responsibility to clear the corresponding interrupt flag bit before returning from an ISR.

EXAMPLE 19-4: UART INITIALIZATION WITH INTERRUPTS ENABLE

```
/*
   The following code example illustrates a UART1 interrupt configuration.
   When the UART1 interrupt is generated, the cpu will jump to the vector assigned to UART1
   interrupt.
*/

   IEC0CLR=0x1c000000;              // disable all UART1 interrupts
   IFS0CLR=0x1c000000;              // clear any existing event
   IPC6CLR=0x0000001f;              // clear the priority
   IPC6SET=0x000d;                  // Set IPL=3, subpriority 1
   IEC0SET=0x1c000000;              // Enable Rx, Tx and Error interrupts

   U1BRG   = #BaudRate;             // Set Uart baud rate.
   U1MODESET= 0x8000;               // Enable Uart for 8-bit Data, no Parity, and 1 Stop bit
   U1STASET= 0x1400;                // Enable Transmitter and Receiver
```

EXAMPLE 19-5: UART1 ISR

```
/*
   The following code example demonstrates a simple interrupt service routine for UART1
   interrupts. The user's code at this vector should perform any application specific operation
   and must clear the UART1 interrupt flags before exiting.
*/
#pragma interupt Uart1IntHandler ipl4 vector 25
void Uart1IntHandler(void)
{
   ... perform application specific operations in response to the interrupt

IFS0CLR = 0x1c000000;               // Be sure to clear the UART1 interrupt flags
                                    // before exiting the service routine.
}
```

19.13 I/O Pin Control

The UART module shares pins with port input/output control and, in some cases, with other modules. To configure a pin for use by the UART, any modules sharing the pin must be disabled. After configuring the UART, the corresponding I/O pins must be configured using the TRIS bit to be an input or output as is required by the UART.

TABLE 19-5: PINS ASSOCIATED WITH A UART

Pin Name	Module Control(2)	Controlling Bit Field	Required TRIS bit Setting	Pin Type(1)	Description
U1TX	ON	UTXEN(3), UEN(2)	Output	D, O	UART1 Transmit pin
U2RX	ON	URXEN(3), UEN(2)	Input	D, I	UART1 Receive pin
$\overline{\text{U1CTS}}$	ON	UEN(2)	Input	D, I	UART1 Clear to Send (CTS) Duplex mode
$\overline{\text{U1RTS}}$	ON	RTSMD(2), UEN(2)	Output	D, O	UART1 Ready to Send (RTS) Duplex mode
BCLK1	ON	IREN(2)	Output	D, O	UART1 IRDA baud clock output
U2TX	ON	UTXEN(3), UEN(2)	Output	D, O	UART2 Transmit pin
U2RX	ON	URXEN(3), UEN(2)	Input	D, I	UART2 Receive pin
$\overline{\text{U2CTS}}$	ON	UEN(2)	Input	D, I	UART2 Clear to Send (CTS) Duplex mode
$\overline{\text{U2RTS}}$	ON	RTSMD(2), UEN(2)	Output	D, O	UART2 Ready to Send (RTS) Duplex mode
BCLK2	ON	IREN(2)	Output	D, O	UART2 IRDA baud clock output

Legend:
ST = Schmitt Trigger input with CMOS levels I = Input
O = Output A = Analog D = Digital

Note 1: All pins are subject to the Device Pin Priority Control.
2: Bits are contained in the UxMODE register.
3: Bits are contained in the UxSTA register

20.0 PARALLEL MASTER PORT

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

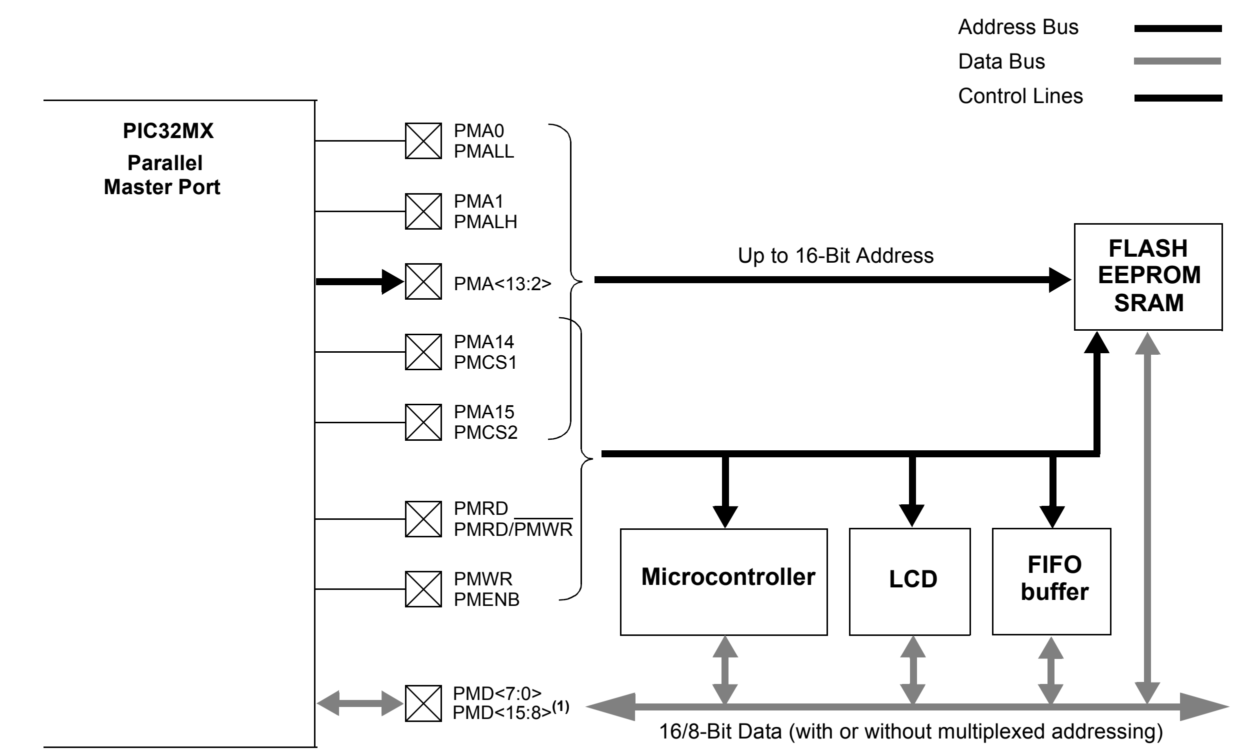
The Parallel Master Port (PMP) is a parallel 8-bit/16-bit input/output module specifically designed to communicate with a wide variety of parallel devices, such as communications peripherals, LCDs, external memory devices, and microcontrollers. Because the interface to parallel peripherals varies significantly, the PMP module is highly configurable.

Key features of the PMP module include:

- 8-bit,16-bit interface
- Up to 16 programmable address lines
- Up to two chip select lines
- Programmable strobe options
  - Individual read and write strobes, or
  - Read/write strobe with enable strobe
- Address auto-increment/auto-decrement
- Programmable address/data multiplexing
- Programmable polarity on control signals
- Parallel Slave Port support
  - Legacy addressable
  - Address support
  - 4-byte deep auto-incrementing buffer
- Programmable Wait states
- Operate during CPU Sleep and Idle modes
- Fast bit manipulation using CLR, SET and INV registers
- Freeze option for in-circuit debugging

Note:	On 64-pin devices, data pins PMD<15:8> are not available.

FIGURE 20-1: PMP MODULE PINOUT AND CONNECTIONS TO EXTERNAL DEVICES



Note 1: On 64-pin devices, data pins PMD<15:8> are not available in 16-Bit Master modes

20.1 PMP Registers

TABLE 20-1: PMP SFR SUMMARY

Virtual Address	Name		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
BF80_7000	PMCON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	ADRMUX<1:0>		PMPTTL	PTWREN	PTRDEN
		7:0	CSF<1:0>		ALP	CS2P	CS1P	—	WRSP	RDSP
BF80_7004	PMCONCLR	31:0	Write clears selected bits in PMCON, read yields undefined value							
BF80_7008	PMCONSET	31:0	Write sets selected bits in PMCON, read yields undefined value							
BF80_700C	PMCONINV	31:0	Write inverts selected bits in PMCON, read yields undefined value							
BF80_7010	PMMODE	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	BUSY	IRQM<1:0>		INCM<1:0>		MODE16	MODE<1:0>	
		7:0	WAITB<1:0>		WAITM<3:0>				WAITE<1:0>	
BF80_7014	PMMODECLR	31:0	Write clears selected bits in PMMODE, read yields undefined value							
BF80_7018	PMMODESET	31:0	Write sets selected bits in PMMODE, read yields undefined value							
BF80_701C	PMMODEINV	31:0	Write inverts selected bits in PMMODE, read yields undefined value							
BF80_7020	PMADDR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CS2EN/A15	CS1EN/A14	ADDR<13:8>					
		7:0	ADDR<7:0>							
BF80_7024	PMADDRCLR	31:0	Write clears selected bits in PRx, read yields undefined value							
BF80_7028	PMADDRSET	31:0	Write sets selected bits in PRx, read yields undefined value							
BF80_702C	PMADDRINV	31:0	Write inverts selected bits in PRx, read yields undefined value							
BF80_7030	PMDOUT	31:24	DATAOUT<31:24>							
		23:16	DATAOUT<23:16>							
		15:8	DATAOUT<15:8>							
		7:0	DATAOUT<7:0>							
BF80_7034	PMDOUTCLR	31:0	Write clears selected bits in PMDOUT, read yields undefined value							
BF80_7038	PMDOUTSET	31:0	Write sets selected bits in PMDOUT, read yields undefined value							
BF80_703C	PMDOUTINV	31:0	Write inverts selected bits in PMDOUT, read yields undefined value							
BF80_7040	PMDIN	31:24	DATAIN<31:24>							
		23:16	DATAIN<23:16>							
		15:8	DATAIN<15:8>							
		7:0	DATAIN<7:0>							
BF80_7044	PMDINCLR	31:0	Write clears selected bits in PMDIN, read yields undefined value							
BF80_7048	PMDINSET	31:0	Write sets selected bits in PMDIN, read yields undefined value							
BF80_704C	PMDININV	31:0	Write inverts selected bits in PMDIN, read yields undefined value							
BF80_7050	PMAEN	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	PTEN<15:8>							
		7:0	PTEN<7:0>							
BF80_7054	PMAENCLR	31:0	Write clears selected bits in PMAEN, read yields undefined value							
BF80_7058	PMAENSET	31:0	Write sets selected bits in PMAEN, read yields undefined value							
BF80_705C	PMAENINV	31:0	Write inverts selected bits in PMAEN, read yields undefined value							
BF80_7060	PMSTAT	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	IBF	IBOV	—	—	IB3F	IB2F	IB1F	IB0F
		7:0	OBE	OBUF	—	—	OB3E	OB2E	OB1E	OB0E
BF80_7064	PMSTATCLR	31:0	Write clears selected bits in PMSTAT, read yields undefined value							
BF80_7068	PMSTATSET	31:0	Write sets selected bits in PMSTAT, read yields undefined value							
BF80_706C	PMSTATINV	31:0	Write inverts selected bits in PMSTAT, read yields undefined value							

TABLE 20-2: PMP INTERRUPT REGISTER SUMMARY

Virtual Address	Name		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
BF88_1040	IEC1	7:0	SPI2RXIE	SPI2TXIE	SPI2EIE	CMP2IE	CMP1IE	PMPIE	AD1IE	I2C1MIE
BF88_1010	IFS1	7:0	SPI2RXIF	SPI2TXIF	SPI2EIF	CMP2IF	CMP1IF	PMPIF	AD1IF	I2C1MIF
BF88_10D0	IPC7	7:0	—	—	—	PMPIP<2:0>			PMPIS<1:0>	

Note: This summary table contains partial register definitions that only pertain to the PMP peripheral. Refer to the "*PIC32MX Family Reference Manual*" (DS61132) for a detailed description of these registers.

REGISTER 20-1: PMCON: PARALLEL PORT CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ON	FRZ	SIDL	ADRMUX1	ADRMUX0	PMPTTL	PTWREN	PTRDEN
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
CSF1(1)	CSF0(1)	ALP(1)	CS2P(1)	CS1P(1)	—	WRSP	RDSP
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** Parallel Master Port Enable bit
1 = PMP enabled
0 = PMP disabled, no off-chip access performed

bit 14 **FRZ:** Freeze in Debug Exception Mode bit
1 = Freeze operation when CPU is in Debug Exception mode
0 = Continue operation when CPU is in Debug Exception mode

bit 13 **SIDL:** Stop in Idle Mode
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 12-11 **ADRMUX1:ADRMUX0:** Address/Data Multiplexing Selection bits
11 = All 16 bits of address are multiplexed on PMD<15:0> pins
10 = All 16 bits of address are multiplexed on PMD<7:0> pins
01 = Lower 8 bits of address are multiplexed on PMD<7:0> pins, upper 8 bits are on PMA<15:8>
00 = Address and data appear on separate pins

bit 10 **PMPTTL:** PMP Module TTL Input Buffer Select bit
1 = PMP module uses TTL input buffers
0 = PMP module uses Schmitt input buffers

bit 9 **PTWREN:** Write Enable Strobe Port Enable bit
1 = PMWR/PMENB port enabled
0 = PMWR/PMENB port disabled

bit 8 **PTRDEN:** Read/Write Strobe Port Enable bit
1 = PMRD/$\overline{PMWR}$ port enabled
0 = PMRD/$\overline{PMWR}$ port disabled

Note 1: These bits have no effect when their corresponding pins are used as address lines

REGISTER 20-1: PMCON: PARALLEL PORT CONTROL REGISTER (CONTINUED)

bit 7-6	**CSF<1:0>:** Chip Select Function bits(1) 11 = Reserved 10 = PMCS2 and PMCS1 function as chip select 01 = PMCS2 functions as chip select, PMCS1 functions as address bit 14 00 = PMCS2 and PMCS1 function as address bits 15 and 14
bit 5	**ALP:** Address Latch Polarity bit(1) 1 = Active-high (PMALL and PMALH) 0 = Active-low ($\overline{\text{PMALL}}$ and $\overline{\text{PMALH}}$)
bit 4	**CS2P:** Chip Select 1 Polarity bit(1) 1 = Active-high (PMCS2) 0 = Active-low ($\overline{\text{PMCS2}}$)
bit 3	**CS1P:** Chip Select 0 Polarity bit(1) 1 = Active-high (PMCS1/PMCS) 0 = Active-low ($\overline{\text{PMCS1}}$/$\overline{\text{PMCS}}$)
bit 2	**Unimplemented:** Read as '0'
bit 1	**WRSP:** Write Strobe Polarity bit For Slave modes and Master Mode 2 (PMMODE<9:8> = 00,01,10): 1 = Read Strobe active-high (PMWR) 0 = Read Strobe active-low ($\overline{\text{PMWR}}$) For Master Mode 1 (PMMODE<9:8> = 11): 1 = Enable strobe active-high (PMENB) 0 = Enable strobe active-low ($\overline{\text{PMENB}}$)
bit 0	**RDSP:** Read Strobe Polarity bit For Slave modes and Master Mode 2 (PMMODE<9:8> = 00,01,10): 1 = Read strobe active-high (PMRD) 0 = Read strobe active-low ($\overline{\text{PMRD}}$) For Master Mode 1 (PMMODE<9:8> = 11): 1 = Read/Write strobe active-high (PMRD/$\overline{\text{PMWR}}$) 0 = Read/Write strobe active-low ($\overline{\text{PMRD}}$/PMWR)

Note 1: These bits have no effect when their corresponding pins are used as address lines

REGISTER 20-2: PMMODE: PARALLEL PORT MODE REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUSY	IRQM<1:0>		INCM<1:0>		MODE16	MODE<1:0>	
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WAITB1<1:0>[1]		WAITM<3:0>				WAITE1<1:0>[1]	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **BUSY:** Busy bit (Master modes only)
- 1 = Port is busy
- 0 = Port is not busy

bit 14-13 **IRQM<1:0>:** Interrupt Request Mode bits
- 11 = Reserved – do not use
- 10 = Interrupt generated when Read Buffer 3 is read or Write Buffer 3 is written (Buffered PSP mode) or on a read or write operation when PMA<1:0> = 11 (Addressable PSP mode only)
- 01 = Interrupt generated at the end of the read/write cycle
- 00 = No Interrupt generated

bit 12-11 **INCM<1:0>:** Increment Mode bits
- 11 = Slave mode read and write buffers auto-increment (MODE<1:0> = 00 only)
- 10 = Decrement ADDR<15:0> by 1 every read/write cycle[2]
- 01 = Increment ADDR<15:0> by 1 every read/write cycle[2]
- 00 = No increment or decrement of address

bit 10 **MODE16:** 8/16-Bit Mode bit
- 1= 16-bit mode: a read or write to the data register invokes a single 16-bit transfer[4]
- 0= 8-bit mode: a read or write to the data register invokes a single 8-bit transfer

Note 1: Whenever WAITM3:WAITM0 = 0000, WAITB and WAITE bits are ignored and forced to 1 T_{PBCLK} cycle for a write operation; WAITB = 1 T_{PBCLK} cycle, WAITE = 0 T_{PBCLK} cycles for a read operation.

2: When ADDR15 and ADDR14 are used as CS2 and CS1, or ADDR15 is used as CS2, these bits are not subject to auto-increment/decrement.

3: In Master Mode 1 or Master Mode 2, data pins PMD<15:0> are active when MODE16 = 1; data pins PMD<7:0> are active when MODE16 = 0.

4: On 64-pin devices, data pins PMD<15:8> are not available.

REGISTER 20-2: PMMODE: PARALLEL PORT MODE REGISTER (CONTINUED)

bit 9-8 **MODE1:MODE0:** Parallel Port Mode Select bits

11 =Master Mode 1 (PMCSx, PMRD/$\overline{\text{PMWR}}$, PMENB, PMA<x:0>, PMD<15:0>)[3,4]
10 =Master Mode 2 (PMCSx, PMRD, PMWR, PMA<x:0>, PMD<15:0>)[3,4]
01 =Addressable Slave Mode, control signals ($\overline{\text{PMRD}}$, $\overline{\text{PMWR}}$, $\overline{\text{PMCS}}$, PMD<7:0>, PMA<1:0>)
00 =Legacy Parallel Slave Port, control signals ($\overline{\text{PMRD}}$, $\overline{\text{PMWR}}$, $\overline{\text{PMCS}}$, PMD<7:0>)

bit 7-6 **WAITB1:WAITB0:** Wait State Configuration bits (Data Setup to Read/Write) [1]

11 =Data wait of 4 TPB; multiplexed address phase of 4 TPB
10 =Data wait of 3 TPB; multiplexed address phase of 3 TPB
01 =Data wait of 2 TPB; multiplexed address phase of 2 TPB
00 =Data wait of 1 TPB; multiplexed address phase of 1 TPB

bit 5-2 **WAITM3:WAITM0:** Wait State Configuration bits (Read to Byte Enable Strobe)

1111 =Wait of additional 16 TPB
...
0001 =Wait of additional 2 TPB
0000 =No additional wait cycles (operation forced into one TPB)

bit 3-0 **WAITE1:WAITE0:** Wait State Configuration bits (Data Hold After Strobe)[1]

11 =Wait of 4 TPB
10 =Wait of 3 TPB
01 =Wait of 2 TPB
00 =Wait of 1 TPB

for Read operations:
11 =Wait of 3TPB
10 =Wait of 2TPB
01 =Wait of 1TPB
00 =Wait of 0TPB

Note 1: Whenever WAITM3:WAITM0 = 0000, WAITB and WAITE bits are ignored and forced to 1 TPBCLK cycle for a write operation; WAITB = 1 TPBCLK cycle, WAITE = 0 TPBCLK cycles for a read operation.

2: When ADDR15 and ADDR14 are used as CS2 and CS1, or ADDR15 is used as CS2, these bits are not subject to auto-increment/decrement.

3: In Master Mode 1 or Master Mode 2, data pins PMD<15:0> are active when MODE16 = 1; data pins PMD<7:0> are active when MODE16 = 0.

4: On 64-pin devices, data pins PMD<15:8> are not available.

REGISTER 20-3: PMADDR: PARALLEL PORT ADDRESS REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CS2EN/A15	CS1EN/A14	ADDR<13:8>					
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADDR<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **CS2EN:** Chip Select 2 bit
1 = Chip Select 2 is active
0 = Chip Select 2 is inactive (pin functions as PMA<15>)

bit 14 **CS1EN:** Chip Select 1 bit
1 = Chip Select 1 is active
0 = Chip Select 1 is inactive (pin functions as PMA<14>)

bit 13-0 **ADDR13:ADDR0:** Destination Address bits

REGISTER 20-4: PMDOUT: PARALLEL PORT DATAOUT REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DATAOUT<31:24>							
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DATAOUT<23:16>							
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DATAOUT<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DATAOUT<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 15-0 **DATAOUT<31:0>:** Output Data Port bits for 8-bit write operations in Slave modes.

REGISTER 20-5: PMDIN REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DATAIN<31:24>							
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DATAIN<23:16>							
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DATAIN<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DATAIN<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **DATAIN<31:0>:** Input and Output Data Port bits for 8-bit or 16-bit read/write operations in Master modes; Input Data Port bits for read operations in Slave modes.

REGISTER 20-6: PMAEN: PARALLEL PORT PIN ENABLE REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTEN<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTEN<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-14 **PTEN15:PTEN14:** PMCSx Strobe Enable bits
1 = PMA15 and PMA14 function as either PMA<15:14> or PMCS2 and PMCS1[(1)]
0 = PMA15 and PMA14 function as port I/O

bit 13-2 **PTEN13:PTEN2:** PMP Address Port Enable bits
1 = PMA<13:2> function as PMP address lines
0 = PMA<13:2> function as port I/O

bit 1-0 **PTEN1:PTEN0:** PMALH/PMALL Strobe Enable bits
1 = PMA1 and PMA0 function as either PMA<1:0> or PMALH and PMALL[(2)]
0 = PMA1 and PMA0 pads functions as port I/O

Note 1: The use of these pins as PMA15/PMA14 or CS2/CS1 are selected by bits CSF<1:0> in the PMCON register.

2: The use of these pins as PMA1/PMA0 or PMALH/PMALL depends on the Address/Data Multiplex mode selected by bits, ADRMUX<1:0>, in the PMCON register.

REGISTER 20-7: PMSTAT: PARALLEL PORT STATUS REGISTER (SLAVE MODE ONLY)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R-0	R/W-0	U-0	U-0	R-0	R-0	R-0	R-0
IBF	IBOV	—	—	IB3F	IB2F	IB1F	IB0F
bit 15							bit 8

R-1	R/W-0	U-0	U-0	R-1	R-1	R-1	R-1
OBE	OBUF	—	—	OB3E	OB2E	OB1E	OB0E
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **IBF:** Input Buffer Full Status bit
1 = All writable input buffer registers are full
0 = Some or all of the writable input buffer registers are empty

bit 14 **IBOV:** Input Buffer Overflow Status bit
1 = A write attempt to a full input byte register occurred (must be cleared in software)
0 = No overflow occurred

bit 13-12 **Unimplemented:** Read as ‘0’

bit 11-8 **IB3F:IB0F:** Input Buffer n Status Full bit
1 = Input Buffer contains data that has not been read (reading buffer will clear this bit)
0 = Input Buffer does not contain any unread data

bit 7 **OBE:** Output Buffer Empty Status bit
1 = All readable output buffer registers are empty
0 = Some or all of the readable output buffer registers are full

bit 6 **OBUF:** Output Buffer Underflow Status bit
1 = A read occurred from an empty output byte register (must be cleared in software)
0 = No underflow occurred

bit 5-4 **Unimplemented:** Read as ‘0’

bit 3-0 **OB3E:OB0E:** Output Buffer n Status Empty bit
1 = Output buffer is empty (writing data to the buffer will clear this bit)
0 = Output buffer contains data that has not been transmitted

20.2 Modes Of Operation

20.2.1 CONSIDERATIONS

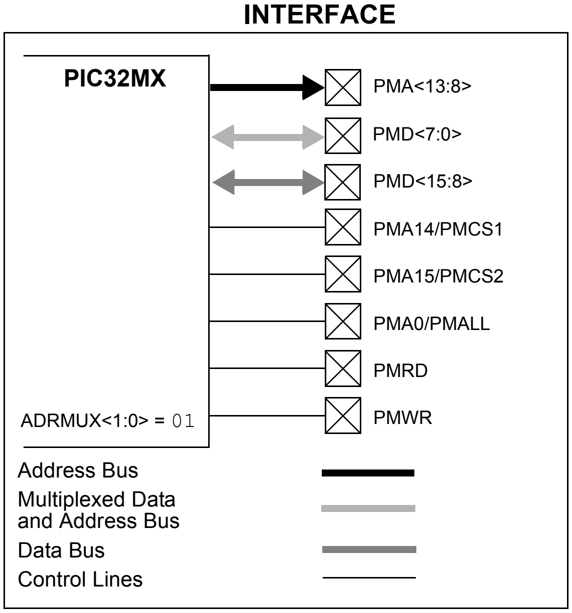
- The PMP module is enabled and ready when the ON bit (PMCON<15>) is set = 1, therefore it is recommended to configure the desired operating mode prior to enabling the module.
- The PMP module is disabled and powered off when the ON bit (PMPCON<15>) = 0, thus providing maximum power savings.
- It is recommended to wait for any pending read or write operation to be completed before enabling/disabling or re-configuring the module

20.2.2 CONSIDERATIONS FOR MASTER MODES

- Setting address bits A15 and A14 = 1 when PMCS2 and PMCS1 are enabled as chip selects will cause both PMCS2 and PMCS1 to be active during a read or write operation. This may enable two devices simultaneously and should be avoided.
- It is always recommended to poll the PMP's BUSY bit prior to any read or write operation to ensure the prior PMP operation has completed.

The PMP module offers two Master modes of operation featuring 16-bit or 8-bit data (default), up to 16 bits of address, and all control signals to operate a variety of external parallel devices such as memory devices, peripherals, and slave microcontrollers. An example using Master Mode 2 is shown in Figure 20-2.

FIGURE 20-2: EXAMPLE PMP MASTER MODE 2, PARTIAL MULTIPLEXED INTERFACE



20.2.3 MASTER MODE SELECTION

The two Master modes are selected using MODE<1:0> bits (PMCON<9:8>). Master Mode 1 is selected by configuring MODE<1:0> bits = 11; Master Mode 2 is selected by configuring MODE<1:0> bits = 10.

20.2.4 8, 16-BIT DATA MODES

The PMP in Master mode supports data widths 8 and 16 bits wide. By default, the data width is 8-bit, MODE16 (PMMODE<10>) bit = 0. To select 16-bit data width, set MODE16 = 1. When configured in 8-Bit Data mode, the upper 8 bits of the data bus, PMD<15:8>, are not controlled by the PMP module and are available as general purpose I/O pins.

Note:	On 64-pin devices, data pins PMD<15:8> are not available.

20.2.5 CHIP SELECTS

Two chip select lines, PMCS1 and PMCS2, are available for the Master modes. The two chip select lines are multiplexed with the Most Significant bits of the address bus A14 and A15. If a pin is configured as a chip select, it is not included in any PMA<15:0> address auto-increment/decrement. It is possible to enable both PMCS2 and PMCS1 as chip selects, or enable only PMCS2 as a chip select, allowing PMCS1 to function strictly as address line A14. It is not possible to enable only PMCS1. The chip select signals are configured using the Chip Select Function bits CSF<1:0> (PMCON <7:6>).

TABLE 20-3: CHIP SELECT CONTROL

CSF<1:0>	FUNCTION
00	PMCS2 = A15, PMCS1 = A14
01	PMCS2 = Enabled, PMCS1 = A14
10	PMCS2, PMCS1 = Enabled

Refer to **Section 20.2.16 "Addressing Considerations"** for information regarding chip select address mapping.

20.2.6 PORT PIN CONTROL

The PMAEN register controls the functionality of the address pins PMA<15:0>. Setting any PMAEN bit = 1 configures the corresponding PMA pin as an address line. Those bits set = 0 remain as general purpose I/O pins.

Refer to **Section 20.5 "I/O Pin Control"** regarding I/O pin configuration.

20.2.7 READ/WRITE CONTROL

The PMP module supports two distinct read/write signaling methods. In Master Mode 1, Read and Write strobe are combined into a single control line, PMRD/PMWR; a second control line, PMENB, determines when a read or write action is to be taken.

In Master Mode 2, Read and Write strobes (PMRD and PMWR) are supplied on separate pins.

To enable the PMRD/PMWR and PMWR/PMENB pins, set PTRDEN bit (PMCON<8>) and PTWREN bit (PMCON<9>) = 1.

20.2.8 CONTROL LINE POLARITY

All control signals (PMRD, PMWR, PMALL, PMALH, PMCS2 and PMCS1) can be individually configured for either positive (active-high) or negative (active-low) polarity. The polarity for each control line is controlled by separate bits in the PMCON register.

TABLE 20-4: MASTER MODE PIN POLARITY

CONTROL PIN	PMCON Control Bit	Active-High Select	Active-Low Select
PMRD	RDSP	1	0
PMWR	WRSP	1	0
PMCS2	CS2P	1	0
PMCS1	CS1P	1	0
PMALL/H	ALP	1	0

Note that the polarity of control signals that share the same output pin (for example, PMWR and PMENB) are controlled by the same bit; the configuration depends on which Master Port mode is being used.

20.2.9 AUTO-INCREMENT/DECREMENT

While the module is operating in a Master mode, the auto-address increment/decrement bits INCM<1:0> (PMMODE<12:11>) control the behavior of the address value that appears on the PMA<15:0> address pins. The address can be made to automatically increment or decrement after each read and write operation, once each operation is completed, and the BUSY bit goes to ‘0’.

TABLE 20-5: ADDRESS AUTO-INCREMENT/DECREMENT CONFIGURATION

INCM<1:0>	FUNCTION
00	No Increment, No Decrement
01	Increment every R/W Cycle
10	Decrement every R/W Cycle

If the Chip Select signals are disabled and configured as address bits, the bits will participate in the increment and decrement operations; otherwise, the PMCS2 and PMCS1 bit values will be unaffected.

20.2.10 WAIT STATES

In Master modes, the user has control over the duration of the read, write, and address cycles by configuring the module Wait states. Three portions of the cycle, the beginning, middle, and end are configured using the corresponding WAITB, WAITM, and WAITE bits in the PMMODE register.

20.2.11 ADDRESS MULTIPLEXING

In either of the Master modes the address bus can be multiplexed together with the data bus. There are three Address Multiplexing modes available; Demultiplexed, Partial Multiplexed and Full Multiplexed. The Addressing Multiplex mode is configured using bits ADRMUX<1:0> (PMCON<12:11>).

For detailed examples illustrating address multiplexing configurations, refer to the PMP chapter in the “*PIC32MX Family Reference Manual*” (DS61132).

TABLE 20-6: ADDRESS MULTIPLEX CONFIGURATIONS

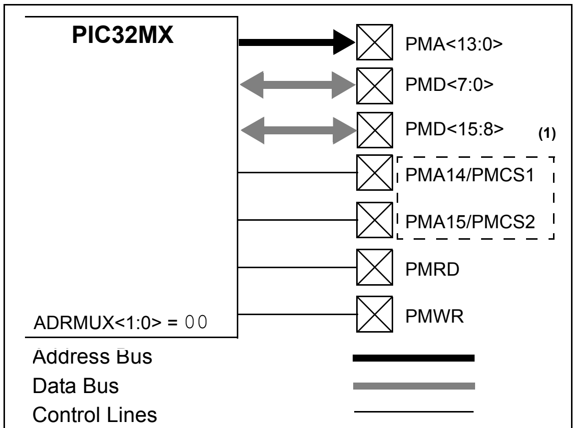
ADRMUX<1:0>	Multiplex Modes
00	Demultiplexed
01	Partial (uses PMD<7:0>)
10	Full (uses PMD<7:0>)
11	Full (uses PMD<15:0>)

> **Note:** A design implementing partial or full multiplexed address and data bus allows the unused PMA address pins to be used as general purpose I/O pins. However, depending on the Multiplexing mode, read and write operations will be extended by several peripheral bus clock cycles, TPBCLK.

20.2.12 DEMULTIPLEXED MODE

In Demultiplexed mode, address bits are presented on pins PMA<15:0>. Note, PMA15 is not available if PMCS2 is enabled and PMA14 is not available if PMCS1 is enabled. Data bits are presented on pins PMD<15:0> in 16-Bit Data mode; pins PMD<7:0> in 8-Bit Data mode. Demultiplexed mode is selected by configuring bits ADRMUX<1:0> = 00.

FIGURE 20-3: DEMULTIPLEXED ADDRESSING

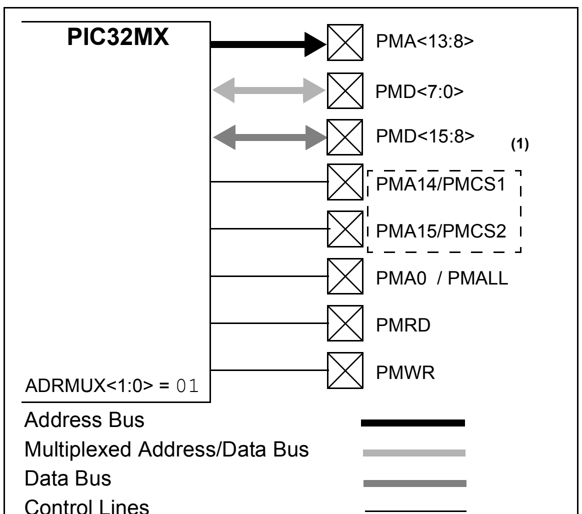


Note 1: PMA15 is not available if PMCS2 is enabled. PMA14 is not available if PMCS1 is enabled.

20.2.13 PARTIAL MULTIPLEXED MODE

In Partial Multiplexed mode, the lower eight address bits are multiplexed with data pins PMD<7:0>. The upper eight address bits are unaffected and are presented on PMA<15:8>. Note, PMA15 is not available if PMCS2 is enabled and PMA14 is not available if PMCS1 is enabled. The PMA<0> pin is used as an Address Latch, and presents the Address Latch Low enable strobe (PMALL). PMA<7:1> are available as general purpose I/O pins. Partial Multiplexed mode is selected by configuring bits ADRMUX<1:0> = `00`.

FIGURE 20-4: PARTIAL MULTIPLEXED ADDRESSING

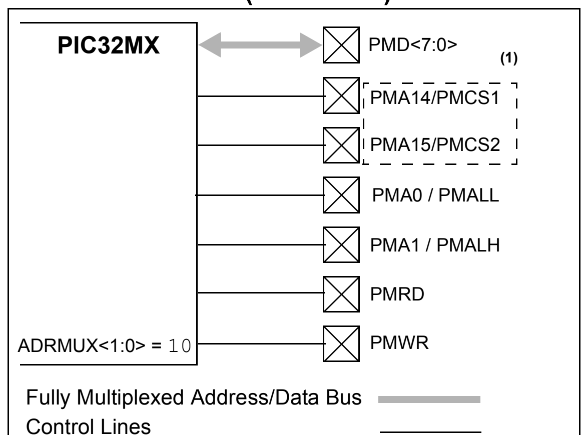


Note 1: PMA15 is not available if PMCS2 is enabled. PMA14 is not available if PMCS1 is enabled.

20.2.14 FULL MULTIPLEXED MODE (8-BIT DATA PINS)

In 8-Bit Full Multiplexed mode, the entire 16 bits of the address are multiplexed with the data pins on PMD<7:0>. The PMA<0> and PMA<1> pins are used to present Address Latch Low enable (PMALL) and Address Latch High enable PMALH strobes, respectively. Pins PMA<13:2> are not used as address pins and can be used as general purpose I/O pins. In the event address bits PMA15 or PMA14 are configured as chip selects, the corresponding address bits PMADDR<15> and PMADDR<14> are automatically forced = `0`. Full 8-Bit Multiplexed mode is selected by configuring bits ADRMUX<1:0> (PMCON<12:11>) = `10`.

FIGURE 20-5: FULL MULTIPLEXED ADDRESSING (8-BIT BUS)

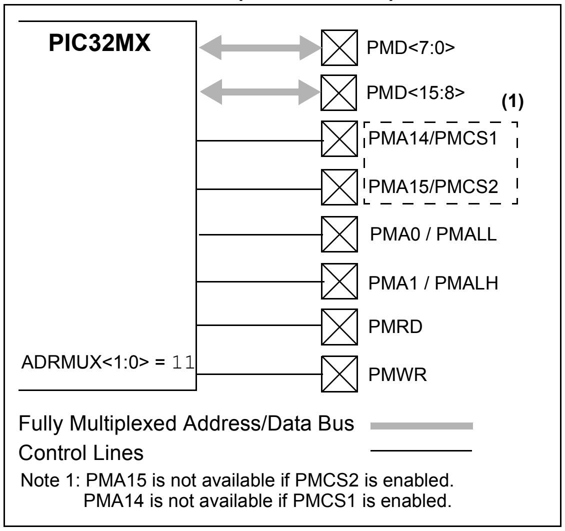


Note 1: PMA15 is not available if PMCS2 is enabled. PMA14 is not available if PMCS1 is enabled.

20.2.15 FULL MULTIPLEXED MODE (16-BIT DATA PINS)

In Full 16-Bit Multiplexed mode, the entire 16 bits of the address are multiplexed with the data pins on PMD<15:0>. Pins PMA<0> and PMA<1> provide Address Latch Low enable PMALL and Address Latch High enable PMALH strobes, respectively, and at the same time. Pins PMA<13:2> are not used as address pins and can be used as general purpose I/O pins. In the event address bits PMA15 or PMA14 are configured as chip selects, the corresponding address bits PMADDR<15> and PMADDR<14> are automatically forced = `0`. Full 16-Bit Multiplexed mode is selected by configuring bits:
ADRMUX<1:0>(PMCON<12:11>) = `11`

FIGURE 20-6: FULL MULTIPLEXED ADDRESSING (16-BIT BUS)



20.2.16 ADDRESSING CONSIDERATIONS

PMCS2 and PMCS1 chip select pins share functionality with address lines A15 and A14. It is possible to enable both PMCS2 and PMCS1 as chip selects, or enable only PMCS2 as a chip select, allowing PMCS1 to function strictly as address line A14. It is not possible to enable only PMCS1.

When configured as chip selects, a `1` must be written into bit position 15 or 14 of the PMADDR register in order for PMCS2 or PMCS1 to become active during a read or write operation. Failing to write a 1 to PMCS2 or PMCS1 does not prevent address pins PMA<13:0> from being active as the specified address appears, however, no chip select signal will be active.

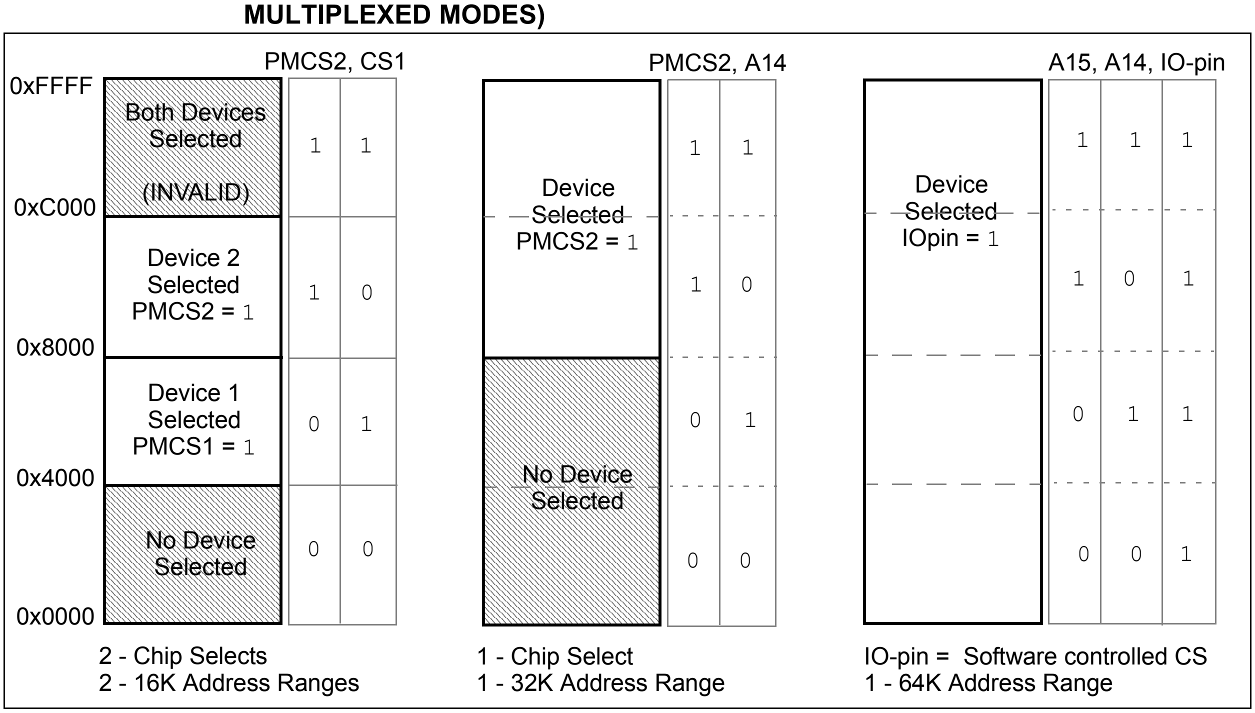
> **Note:** When using Auto-Increment Address mode, PMCS2 and PMCS1 do not participate and must be controlled by the user's software by writing '1' to PMADDR<15:14> explicitly.

Disabling one or both chip selects PMCS2 and PMCS1 makes these pins available as address lines A15 and A14.

In Full Multiplexed mode, address bits PMADDR<15:0> are multiplexed with the data bus and in the event address bits PMA15 or PMA14 are configured as chip selects, the corresponding PMADDR<15:14> address bits are automatically forced = `0`. Disabling one or both PMCS2 and PMCS1 makes these bits available as address bits PMADDR<15:14>.

In any of the Master mode multplexing schemes, disabling both chip select pins PMCS2 and PMCS1 requires the user to provide chip select line control through some other I/O pin under software control. See Figure 20-7.

FIGURE 20-7: PMP CHIP SELECT ADDRESS MAPPING (DEMULTIPLEXED AND PARTIAL MULTIPLEXED MODES)



 Advance Information

20.3 Master Mode Timing

The PMP Master mode timing for control, address and data signals is dependent on the PBCLK peripheral bus clock speed, address/data multiplexing and number of Wait states, if any. Table 20-7 provides a summary of PMP read and write maximum sustainable speeds for each of its Address Multiplex modes.

Note: During any Master mode read or write operation, the busy flag will always de-assert 1 peripheral bus clock cycle (T_{PBCLK}), before the end of the operation, including Wait states.

TABLE 20-7: READ/WRITE SPEEDS, NO WAIT STATES

Address/Data Multiplex Configuration	ADRMUX	PMP Operation (PBCLK cycles)		Speed(1) (MHz)	
		Read	Write	Read	Write
Demultiplexed	00	2	3	36.0	24.0
Partial Multiplex	01	5	6	14.4	12.0
Full Multiplexed (8-bit data)	10	8	9	9.0	8.0
Full Multiplexed (16-bit data)	11	5	6	14.4	12.0

Note 1: Peripheral bus clock operating at 1:1 with SYSCLK (72MHz)

20.3.1 MASTER PORT CONFIGURATION

The Master mode configuration is determined primarily by the interface requirements to the external device. Address multiplexing, control signal polarity, data width and Wait states typically dictate the specific configuration of the PMP master port.

The following illustrates example settings for Master Mode 2 operation:

- Select Master Mode 2 - MODE<1:0> (PMMODE<9:8>) = `10`.
- Select 16-Bit Data mode - MODE16 (PMMODE<10>) = `1`.
- Select partial multiplexed addressing - ADRMUX<1:0> (PMCON<12:11>) = `01`.
- Select auto-address increment - INCM<1:0> (PMMODE<12:11>) = `01`.
- Enable Interrupt Request mode - IRQM<1:0> (PMMODE<14:13>) = `01`.
- Enable PMRD strobe - PTRDEN (PMCON<8>) = `1`.
- Enable PMWR strobe - PTWREN (PMCON<9>) = `1`.
- Enable PMCS2 and PMCS1 chip selects - CSF (PMCON<7:6>) = `10`.
- Select PMRD "active-low" pin polarity - RDSP (PMCON<0>) = `0`.
- Select PMWR "active-low" pin polarity - WRSP (PMCON<1>) = `0`.
- Select PMCS2, PMCS1 "active-low" pin polarity - CS2P (PMCON<4>) = `0` and CS1P (PMCON<3>) = `0`.
- Select 1 wait cycle for data setup - WAITB<1:0>(PMMODE<7:6>) = `00`.
- Select 2 wait cycles to extend PMRD/PMWR - WAITM<3:0>(PMMODE<5:2>) = `01`.
- Select 1 wait cycle for data hold - WAITB<1:0>(PMMODE<1:0>) = `00`.
- Enable upper 8 PMA<15:8> address pins - PMAEN<15:8> = `1` (lower 8 can be used as general purpose I/O).

20.3.2 MASTER PORT INITIALIZATION

The Master mode initialization properly prepares the PMP port for communicating with an external device.

The following steps should be performed to properly configure the PMP port:

1. If interrupts are used, disable the PMP interrupt by clearing the interrupt enable bit PMPIE (IEC1<2>) = 0.
2. Stop and reset the PMP module by clearing the control bit ON (PMCON<15>) = 0.
3. Configure the desired settings in the PMCON, PMMODE and PMAEN control registers.
4. If interrupts are used:
   a) Clear interrupt flag bit PMPIF (IFS1<2>) = 0.
   b) Configure the PMP interrupt priority bits PMPIP<2:0> (IPC7<4:2>) and interrupt sub priority bits PMPIS (IPC7<1:0>.
   c) Enable PMP interrupt by setting interrupt enable bit PMPIE = 1.
5. Enable the PMP master port by setting control bit ON = 1.

EXAMPLE 20-1: PARALLEL MASTER PORT INITIALIZATION

```
IEC1CLR = 0x0004;          //Disable PMP int

PMCON = 0x0BC0;            //Stop and Configure

PMMODE = 0x2A04;           //Config PMMODE reg

PMAEN = 0xFF00;            //Config PMAEN reg

IPC7SET = 0x001C;          //Priority level=7

IPC7SET = 0x0003;          //subpriority=3
                           //Same as..
                           //IPC7SET=0x001F
IFS1CLR  = 0x0004;         //Clear PMP flag

IEC1SET  = 0x0004;         //Enable PMP int

PMCONSET = 0x8000;         //Enable PMP

PMADDR = 0x4000;           //Set external address

PMDIN = 0x1234;            //Write to device
...
```

20.3.3 READ OPERATION

To perform a read on the parallel bus, the user reads the PMDIN register. The effect of reading the PMDIN register retrieves the current value and causes the PMP to activate the chip select lines and the address bus. The read line PMRD is strobed and the new data is latched into the PMDIN register, making it available for the next time the PMDIN register is read.

Note:	The read data obtained from the PMDIN register is actually the read value from the previous read operation. Hence, the first user read will be a dummy read to initiate the first bus read and fill the read register. Also, the requested read value will not be ready until after the BUSY bit is observed low. Therefore, in a back-to-back read operation, the data read from the register will be the same for both reads. The next read of the register will yield the new value.

Refer to the PIC32MX Family Reference Manual for a detailed description of the read operation and illustrated example.

20.3.4 WRITE OPERATION

To perform a write onto the parallel port, the user writes to the PMDIN register (same register used for a read operation). This causes the module to first activate the chip select lines and the address bus. The write data from the PMDIN register is placed onto the PMD data bus and the write line PMPWR is strobed.

20.3.5 PARALLEL MASTER PORT STATUS

In addition to the PMP interrupt, a BUSY bit is provided to indicate the status of the module. This bit is only used in Master modes.

While any read or write operation is in progress, the BUSY bit is set for all but the very last peripheral bus-cycle of the operation. While the bit is set, any request by the user to initiate a new operation will be ignored (i.e., writing or reading the PMDIN register will not initiate either a read nor a write).

If a large number of wait-states are used, or if the PBCLK clock is operating slower than the SYSCLK clock, it is possible for the PMP module to be in the process of completing a read or write operation when the next CPU instruction is attempting to read or write the PMP module. For this reason, it is highly recommended that the PMP's BUSY bit be checked prior to any read or write operation and any user operation that modifies the PMADDR address register. See the following code example.

EXAMPLE 20-2: POLLING THE BUSY FLAG

```
/*An generic C example PMP write function
  utilizing the BUSY bit.
*/
pmpWrite(unsigned int value)
{
   while(PMMODE & 0x8000);  // PMP busy?
   PMDIN = value;  // perform write
}

/*An MPLAB C32 example PMP write function
  utilizing BUSY bit.
*/

pmpWrite(unsigned int value)
{
   while(PMMODEbits.BUSY); // PMP busy?
   PMDIN = value;  // perform write
}
```

In most applications, the PMP's chip select pin(s) provide the chip select interface and are under the timing control of the PMP module. However, some applications may require the PMP chip select pin(s) not be configured as a chip select, but as a high-order address line, such as PMA<14> or PMA<15>. In this situation, the application's chip select function must be provided by an available I/O port pin under software control. In these cases, it is especially important that the user's software poll the BUSY bit to ensure any read or write operation is complete before de-asserting the software controlled chip select.

The following example illustrates a common technique.

EXAMPLE 20-3: POLLING THE BUSY FLAG AND SOFTWARE CONTROLLED CHIP SELECT

```
/* An generic C example PMP write function
   utilizing PORTD.RD1 as an active low
   chip select and the BUSY bit.
*/
pmpWrite(unsigned int value)
{
   PORTDCLR = 0x0002; //CS enabled
   while(PMMODE & 0x8000); // PMP busy?
   PMDIN = value;  //perform write

   while(PMMODE & 0x8000); //wait for PMP
   PORTDSET = 0x0002; //CS disabled
}
/* An MPLAB C32 example PMP write function
   utilizing PORTD.RD1 as an active low
   chip select and the BUSY bit.
*/
pmpWrite(unsigned int value)
{
   PORTDCLR = 0x0002; //CS enabled
   while(PMMODEbits.BUSY);  // PMP busy?
   PMDIN = value; // perform write

   while(PMMODEbits.BUSY);  // wait for PMP
   PORTDSET = 0x0002; //CS disabled
}
```


20.3.6 SLAVE MODE

20.3.6.1 Considerations for Slave Mode

- Do not enable or disable the module during any read or write operation
- Because of the asynchronous nature of the read and write operations, it is highly recommended that the user rely on the PSP status bits prior to any read or write operation.

The PMP module provides 8-Bit (byte) legacy Parallel Slave Port functionality as well as new Buffered and Addressable Slave modes.

20.3.7 MODE SELECTION

The three Master modes are selected using MODE<1:0> bits (PMCON<9:8>). Legacy Slave mode is selected by configuring MODE<1:0> bits = `00`; Buffered and Addressable Slave modes are selected by configuring MODE<1:0> = `01`. Additionally, Buffered Slave mode requires bits INCM<1:0> (PMMODE<12:11>) = `11`.

TABLE 20-8: Slave Mode Selection

Slave Mode	PMCON MODE<1:0>	PMMODE INCM<1:0>
Legacy	00	x = don't care
Buffered	00	11
Addressable	01	x = don't care

All Slave modes support 8-bit data only and the associated module control pins are automatically dedicated to the module when any of these modes are selected. The user only need to configure the polarity of the PMCS1, PMRD and PMWR signals.

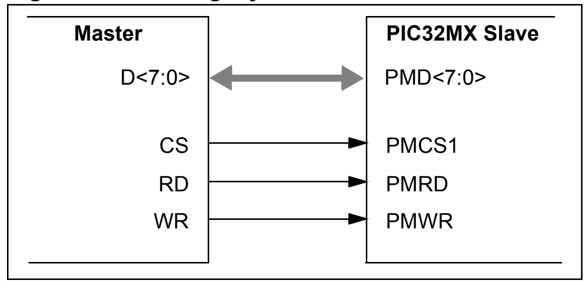
TABLE 20-9: Slave Mode Pin Polarity Configuration

CONTROL PIN	PMCON Control Bit	Active-High Select	Active-Low Select
PMRD	RDSP	1	0
PMWR	WRSP	1	0
PMCS1	CS1P	1	0

20.3.8 LEGACY PARALLEL SLAVE MODE

In Legacy Slave mode, an external device can asynchronously read and write data using the 8-bit data bus PMD<7:0>, the read PMRD, write PMWR, and chip-select PMCS1 inputs.

Figure 20-8: Legacy Slave Mode Interface



20.3.9 LEGACY SLAVE CONFIGURATION

The Legacy Slave mode configuration is determined automatically and dedicated to the PSP module when the Legacy Slave mode is selected. The user only need to configure the polarity of the PMCS1, PMRD and PMWR signals.

The following example illustrates which control bits are to be set for Legacy Slave mode configuration:

- Configure Legacy Slave mode bits - MODE<1:0> (PMMODE<9:8>) = `00`
- Select PMRD "active-low" pin polarity - RDSP (PMCON<0>) = `0`.
- Select PMWR "active-low" pin polarity - WRSP (PMCON<1>) = `0`.
- Select PMCS2, PMCS1 "active-low" pin polarity - CS2P (PMCON<4>) = `0` and CS1P (PMCON<3>) = `0`.

20.3.10 SLAVE PORT INITIALIZATION

The Legacy Slave mode initialization properly prepares the PMP port for communicating with an external master device.

1. If interrupts are used, disable the PMP interrupt by clearing the interrupt enable bit PMPIE (IEC1<2>) = `0`.
2. Stop and reset the PMP module by clearing the control bit ON (PMCON<15>) = `0`.
3. Configure the desired settings in the PMCON and PMMODE control registers.
4. If interrupts are used:
   a) Clear interrupt flag bit PMPIF (IFS1<2>) = `0`.
   b) Configure the PMP interrupt priority bits PMPIP<2:0> (IPC7<4:2>) and interrupt sub priority bits PMPIS (IPC7<1:0>.
   c) Enable PMP interrupt by setting interrupt enable bit PMPIE = `1`.
5. Enable the PMP slave port by setting control bit ON = `1`.

EXAMPLE 20-4: EXAMPLE CODE: LEGACY PARALLEL SLAVE PORT INITIALIZATION

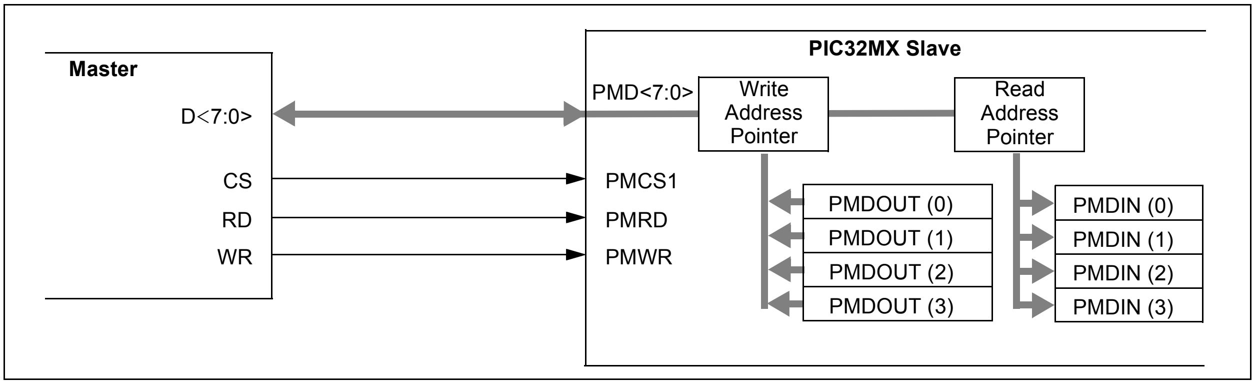
```
IEC1CLR = 0x0004        //Disable PMP int

PMCON = 0x0000          //Stop and Configure

PMMODE = 0x0000         //Config PMMODE

IPC7SET = 0x001C;       //Priority level=7

IPC7SET = 0x0003;       //subpriority =3
                      //Same as...
                      //IPC7SET=0x001F

IFS1CLR  = 0x0004;   //Clear PMP flag

IEC1SET  = 0x0004;   //Enable PMP int

PMCONSET = 0x8000;     //Enable PMP
```


20.3.11 Buffered Slave Mode

Buffered Parallel Slave Port mode is functionally identical to the Legacy Parallel Slave Port mode with one exception: the implementation of 4-level read and write buffers. Buffered Slave mode is enabled by setting the PMMODE<INCM1:INCM0> bits to '11'.

When the Buffered mode is active, the module uses the PMDIN register as write buffers and the PMDOUT register as read buffers, with respect to the master device. Each register is divided into four 8-bit buffer registers, four read buffers in PMDOUT and four write buffers in PMDIN. Buffers are numbered 0 through 3, starting with the lower byte <7:0> and progressing upward through the high byte <31:24>.

FIGURE 20-9: PARALLEL MASTER/SLAVE CONNECTION BUFFERED



20.3.12 BUFFERED SLAVE CONFIGURATION

The Buffered Slave mode configuration is determined automatically and dedicated to the PMP module when the Buffered Slave mode is selected. The user only need to configure the polarity of the PMCS1, PMRD and PMWR signals.

The following example illustrates which control bits are to be set for Buffered Slave mode configuration:

- Configure Buffered Slave mode bits - MODE<1:0> (PMMODE<9:8>) = 00 and INCM<1:0> (PMMODE<12:11>) = 11.
- Select PMRD "active-low" pin polarity - RDSP (PMCON<0>) = 0.
- Select PMWR "active-low" pin polarity - WRSP (PMCON<1>) = 0.
- Select PMCS2, PMCS1 "active-low" pin polarity - CS2P (PMCON<4>) = 0 and CS1P (PMCON<3>) = 0.

20.3.13 BUFFERED SLAVE MODE INITIALIZATION

The Buffered Slave mode initialization properly prepares the PSP port for communicating with an external master device.

The following steps should be performed to properly configure the PSP port:

1. If interrupts are used, disable the PMP interrupt by clearing the interrupt enable bit PMPIE (IEC1<2>) = 0.
2. Stop and reset the PMP module by clearing the control bit ON (PMCON<15>) = 0.
3. Configure the desired settings in the PMCON and PMMODE control registers.
4. If interrupts are used:
   a) Clear interrupt flag bit PMPIF (IFS1<2>) = 0.
   b) Configure the PMP interrupt priority bits PMPIP<2:0> (IPC7<4:2>) and interrupt sub priority bits PMPIS (IPC7<1:0>.
   c) Enable PSP interrupt by setting interrupt enable bit PMPIE = 1.
5. Enable the PMP slave port by setting control bit ON = 1.

EXAMPLE 20-5: BUFFERED PARALLEL SLAVE MODE INITIALIZATION

```
IEC1CLR = 0x0004        //Disable PMP

PMCON = 0x0000          //Stop and Configure

PMMODE = 0x1800         //Configure PMMODE

IPC7SET = 0x001C;       //Priority level=7

IPC7SET = 0x0003;       //subpriority=3
                        //Same as...
                        //IPC7SET=0x001F
IFS1CLR  = 0x0004;      //Clear PMP flag

IEC1SET  = 0x0004;      //Enable PMP int

PMCONSET = 0x8000;      //Enable PMP
```

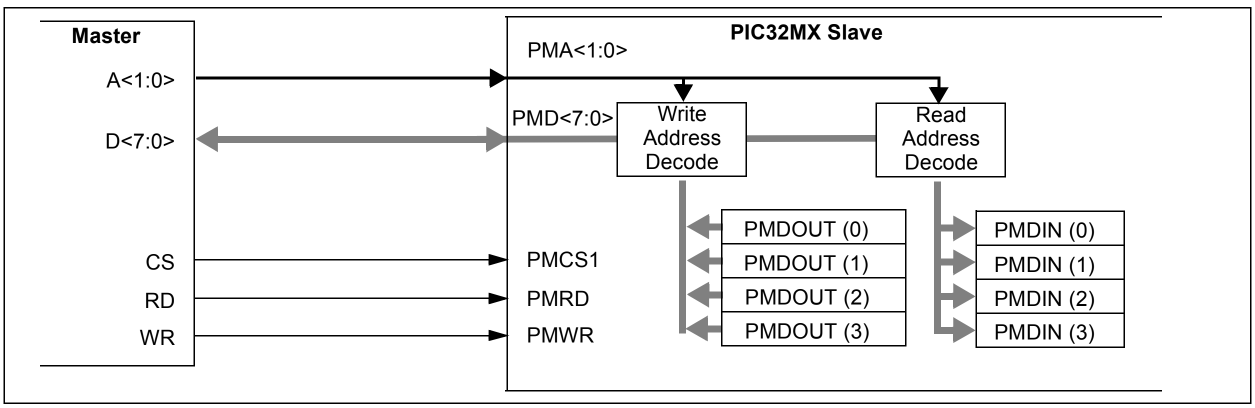
20.3.14 ADDRESSABLE SLAVE MODE

In the Addressable Parallel Slave Port mode, the module is configured with two extra inputs, PMA<1:0>. This makes the 4-byte buffer space directly addressable as fixed pairs of read and write buffers. As with Buffered Legacy mode, data is output from register PMDOUT and is input to register PMDIN. Table 20-1 shows the address resolution for the incoming address to the input and output registers.

TABLE 20-10: SLAVE MODE BUFFER ADDRESSES

PMA<1:0>	Output Register PMDOUT(Buffer)	Input Register PMDIN (Buffer)
00	<7:0> (0)	<7:0> (0)
01	<15:8> (1)	<15:8> (1)
10	<23:16> (2)	<23:16> (2)
11	<31:24> (3)	<31:24> (3)

FIGURE 20-10: PARALLEL MASTER/SLAVE CONNECTION ADDRESSABLE BUFFER



20.3.15 ADDRESSABLE SLAVE CONFIGURATION

The Addressable Slave mode configuration is determined automatically and dedicated to the PSP module when the Addressable Slave mode is selected. The user only need to configure the polarity of the PMCS1, PMRD and PMWR signals.

The following example illustrates which control bits are to be set for Addressable Slave mode configuration:

- Configure Addressable Slave mode bits - MODE<1:0> (PMMODE<9:8>) = `01`.
- Select PMRD “active-low” pin polarity - RDSP (PMCON<0>) = `0`.
- Select PMWR “active-low” pin polarity - WRSP (PMCON<1>) = `0`.
- Select PMCS2, PMCS1 “active-low” pin polarity - CS2P (PMCON<4>) = `0` and CS1P (PMCON<3>) = `0`.

20.3.16 ADDRESSABLE SLAVE PORT INITIALIZATION

The Addressable Slave mode initialization properly prepares the PSP port for communicating with an external master device.

The following steps should be performed to properly configure the PSP port:

 Advance Information

1. If interrupts are used, disable the PMP interrupt by clearing the interrupt enable bit PMPIE (IEC1<2>) = 0.
2. Stop and reset the PMP module by clearing the control bit ON (PMCON<15>) = 0.
3. Configure the desired settings in the PMCON and PMMODE control registers.
4. If interrupts are used:
   a) Clear interrupt flag bit PMPIF (IFS1<2>) = 0.
   b) Configure the PMP interrupt priority bits PMPIP<2:0> (IPC7<4:2>) and interrupt sub priority bits PMPIS (IPC7<1:0>.
   c) Enable PSP interrupt by setting interrupt enable bit PMPIE = 1.
5. Enable the PMP slave port by setting control bit ON = 1.

EXAMPLE 20-6: ADDRESSABLE PARALLEL SLAVE PORT INITIALIZATION

```
IEC1CLR = 0x0004      //Disable PMP int
PMCON = 0x0000        //Stop and Configure
PMMODE = 0x0100       //Config PMMODE
IPC7SET = 0x001C;     //Priority level=7
IPC7SET = 0x0003;     //subpriority=3
                      //Same as...
                      //IPC7SET=0x001F
IFS1CLR  = 0x0004;      //Clear PMP int flag
IEC1SET  = 0x0004;      //Enable PMP int
PMCONSET = 0x8000;    //Enable PMP module
```

20.4 PMP Interrupts

The PMP module has the ability to generate the following types of interrupts reflecting the events that occur during data transfers.

Master mode:

- Interrupt on every read and write operation.

Legacy Slave mode:

- Interrupt on every read and write byte

Buffered Slave mode:

- Interrupt on every read and write byte
- Interrupt on read or write byte of Buffer 3 (PMDOUT<31:24>)

Addressable Slave mode:

- Interrupt on every read and write byte
- Interrupt on read or write byte of Buffer 3 (PMDOUT<31:24>), PMA<1:0> = 11

The PMP module is enabled as a source of interrupt using the PMP interrupt enable bit:

- PMPIE (IEC1<2>).

The interrupt priority level and subpriority level bits must also be configured:

- PMPIP<2:0> (IPC7<4:2>)
- PMPIS<1:0> (IPC7<1:0>)

- The PMP interrupt status flag, PMPIF (IFS1<2>) is typically cleared by the user's software in the ISR.

Below is a partial code example of an ISR.

Note:	It is the user's responsibility to clear the corresponding interrupt flag bit before returning from an ISR.

EXAMPLE 20-7: PMP MODULE INTERRUPT INITIALIZATION

```
/*
    The following code example illustrates a PMP interrupt configuration.
    When the PMP interrupt is generated, the CPU will branch to the vector
    assigned to PMP interrupt.
*/

    // Configure PMP for desired mode of operation
    ...
    // Configure the PMP interrupts
    IPC7SET = 0x0014;      // Set priority level=5
    IPC7SET = 0x0003;      // Set subpriority level=3
                           // Could have also done this in single
                           // operation by assigning IPC7SET = 0x0017

    IFS1CLR  = 0x0002;     // Clear the PMP interrupt status flag
    IEC1SET  = 0x0002;     // Enable PMP interrupts
    PMCONSET = 0x8000;     // Enable PMP module
```

EXAMPLE 20-8: PMP ISR

```
/*
    The following code example demonstrates a simple interrupt
    service routine for PMP interrupts. The user's code at this
    vector should perform any application specific operations and must
    clear the PMP interrupt status flag before exiting.
*/
void _IRQ(_PMP_VECTOR, ipl3) PMP_Interrupt_ISR(void)
{
    ... perform application specific operations in response to the interrupt

    IFS1CLR  = 0x00002;    // Be sure to clear the PMP interrupt status
                           // flag before exiting the service routine.
}
```


20.5 I/O Pin Control

20.5.1 I/O PIN RESOURCES

TABLE 20-11: REQUIRED I/O PIN RESOURCES FOR MASTER MODES

I/O Pin Name	De-multiplex	Partial Multiplex	Full Multiplex	Functional Description
PMPCS2 / PMA15	Yes(2)	Yes(2)	Yes(2)	PMP Chip Select 2 / Address A15
PMPCS1 / PMA14	Yes(2)	Yes(2)	Yes(2)	PMP Chip Select 1 / Address A14
PMA<13:2>	Yes(2)	Yes(3)	No(1)	PMP Address A13..A2
PMA1 / PMALH	No(1)	No(1)	Yes(4)	PMP Address A1 / Address Latch High
PMA0 / PMALL	No(1)	Yes(2)	Yes(4)	PMP Address A0 / Address Latch Low
PMRD / PM$\overline{\text{WR}}$	Yes	Yes	Yes	PMP Read / Write Control
PMWR / PMENB	Yes	Yes	Yes	PMP Write / Enable Control
PMD<15:0>	Yes(5)	Yes(5)	Yes(5)	PMP Bidirectional Data Bus D15..D0

Note 1: "No" indicates the pin is not required and is available as a general purpose I/O pin when the corresponding PMAEN bit is cleared, = 0.

2: Depending on the application, not all PMA<15:0> or CS2, CS1 may be required.

3: When Partial Multiplex mode is selected (ADDRMUX<1:0> = `01`), the lower 8 Address lines are multiplexed with PMD<7:0>, PMA<0> becomes (PMALL) and PMA<7:1> are available as general purpose I/O pins.

4: When Full Multiplex mode is selected (ADDRMUX<1:0> = `10` or `11`), all 16 Address lines are multiplexed with PMD<15:0>, PMA<0> becomes (PMALL), PMA<1> becomes (PMALH) and PMA<13:2> are available as general purpose I/O pins.

5: If MODE16 = `0`, then only PMD<7:0> are required. PMD<15:8> are available as general purpose I/O pins.

6: Data pins PMD<15:0> are available on 100-pin PIC32MX devices and larger. For all other device variants, only pins PMD<7:0> are available.

When enabling any of the PMP module for Slave mode operations, the PMPCS1, PMRD, PMWR control pins, PMD<7:0> data pins and PMA<1:0> address pins are automatically enabled and configured. The user is however responsible for selecting the appropriate polarity for these control lines.

TABLE 20-12: REQUIRED I/O PIN RESOURCES FOR SLAVE MODES

I/O Pin Name	Legacy	Buffered	Addressable	Functional Description
PMPCS1 / PMA14	Yes	Yes	Yes	Chip Select
PMA1 / PMALH	No(1)	No(1)	Yes	Address A1
PMA0 / PMALL	No(1)	No(1)	Yes	Address A0
PMRD / PM$\overline{\text{WR}}$	Yes	Yes	Yes	Read Control
PMWR / PMENB	Yes	Yes	Yes	Write Control
PMD<7:0>	Yes(2)	Yes(2)	Yes(2)	Bidirectional Data Bus D7..D0

Note 1: "No" indicates the pin is not required and is available as a general purpose I/O pin when the corresponding PMAEN bit is cleared, = `0`.

2: Slave modes use PMD<7:0> only. Pins PMD<15:8> are available as general purpose I/O pins. Control bit MODE16 (PMMODE<10>) is ignored.

20.5.2 I/O PIN CONFIGURATION

The following table provides a summary of settings required to enable the I/O pin resources used with this module. The PMAEN register controls the functionality of pins PMA<15:0>. Setting any PMAEN bit = `1` configures the corresponding PMA pin as an address line. Those bits set = `0` remain as general purpose I/O pins.

TABLE 20-13: I/O PIN CONFIGURATION

		Required Settings for Module Pin Control					
I/O Pin Name	**Required(1)**	**Module Control**	**Bit Field**	**TRIS**	**Pin Type**	**Buffer Type(2)**	**Description**
PMPCS2 / PMA15	Yes	ON	CSF<1:0>, CS2, PTEN15	—	O	ST/TTL	PMP Chip Select 2 / Address A15
PMPCS1 / PMA14	Yes	ON	CSF<1:0>, CS1 PTEN14	—	O	ST/TTL	PMP Chip Select 1 / Address A14
PMA<13:2>	Yes	ON	PTEN<13:2>	—	O	ST/TTL	PMP Address A13 .. A2
PMA1 / PMALH	Yes	ON	PTEN<1>	—	I,O	ST/TTL	PMP Address A1 / Address Latch Hi
PMA0 / PMALL	Yes	ON	PTEN<0>	—	I,O	ST/TTL	PMP Address A0 / Address Latch Lo
PM$\overline{RD}$ / PM$\overline{WR}$	Yes	ON	PTRDEN	—	O	ST/TTL	PMP Read / Write Control
PMWR / PMENB	Yes	ON	PTWREN	—	O	ST/TTL	PMP Write / Enable Control
PMD<15:0>	Yes	ON	MODE16, ADRMUX<1:0>	—	I,O	ST/TTL	PMP Bidirectional Data Bus D15 .. D0

Legend: TTL = TTL compatible input or output, ST = Schmitt Trigger input with CMOS levels, I = Input, O = Output

Note 1: Depending on the PMP mode and the user's application, these pins may not be required. If not enabled, these pins can be used as general purpose I/O.

2: Default buffer type is ST.

21.0 REAL-TIME CLOCK AND CALENDAR (RTCC)

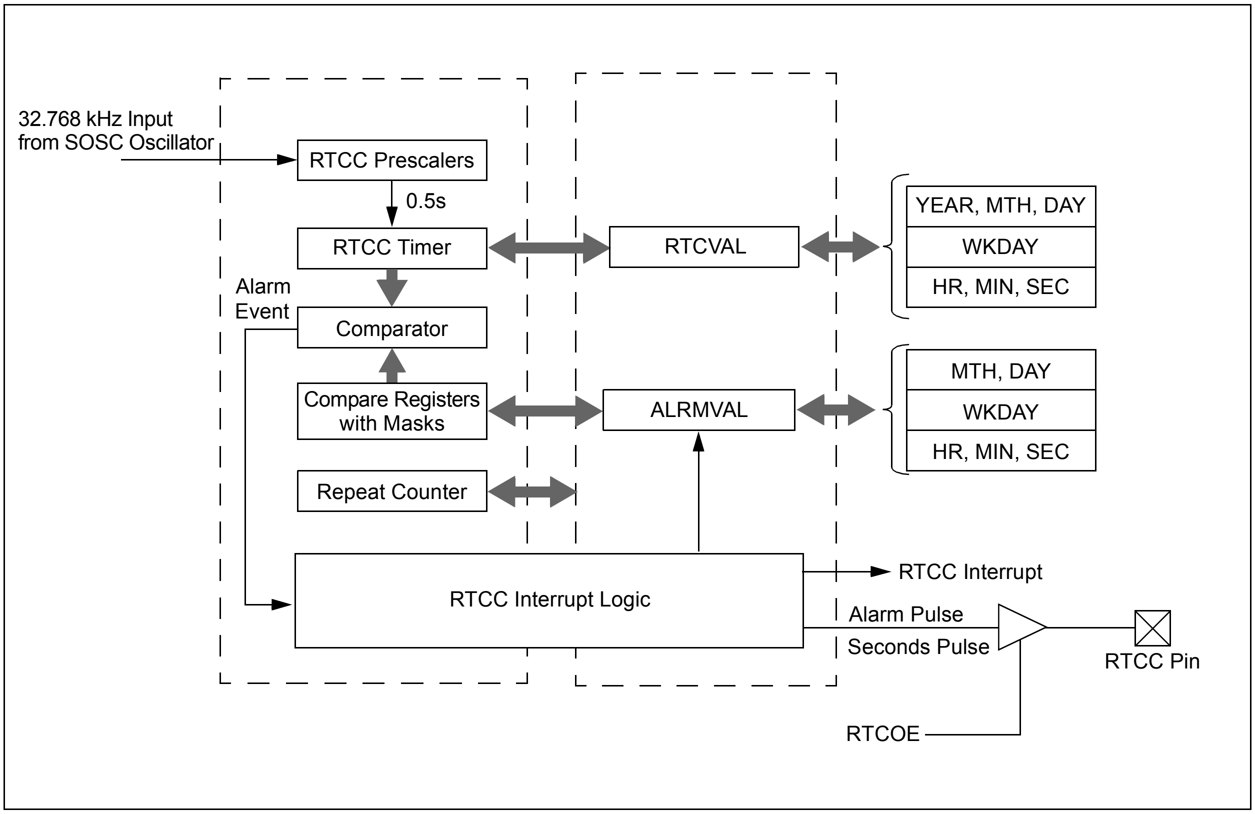
Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

The PIC32MX Real-Time Clock and Calendar (RTCC) module is intended for applications where accurate time must be maintained for extended periods of time with minimal or no CPU intervention. Low-power optimization provides extended battery lifetime while keeping track of time.

Following are some of the key features of this module:

- Time: Hours, Minutes and Seconds
- 24-Hour Format (Military Time)
- Visibility of One-Half-Second Period
- Provides Calendar: Weekday, Date, Month and Year
- Alarm Intervals are configurable for Half a Second, One Second, 10 Seconds, One Minute, 10 Minutes, One Hour, One Day, One Week, One Month and One Year
- Alarm Repeat with Decrementing Counter
- Alarm with Indefinite Repeat: Chime
- Year Range: 2000 to 2099
- Leap Year Correction
- BCD Format for Smaller Firmware Overhead
- Optimized for Long-Term Battery Operation
- Fractional Second Synchronization
- User Calibration of the Clock Crystal Frequency with Auto-Adjust
- Calibration Range: ±0.66 Seconds Error per Month
- Calibrates up to 260 ppm of Crystal Error
- Requirements: External 32.768 kHz Clock Crystal
- Alarm Pulse or Seconds Clock Output on RTCC pin

FIGURE 21-1: RTCC BLOCK DIAGRAM



21.1 RTCC Registers

TABLE 21-1: RTCC SFR SUMMARY

Virtual Address	Name		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
BF80_0200	RTCCON	31:24	—	—	—	—	—	—	CAL<9:8>	
		23:16	CAL<7:0>							
		15:8	ON	FRZ	SIDL	—	—	—	—	—
		7:0	RTSECSEL	RTCCLKON	—	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE
BF80_0204	RTCCONCLR	31:0	Write clears selected bits in RTCCON, read yields undefined value							
BF80_0208	RTCCONSET	31:0	Write sets selected bits in RTCCON, read yields undefined value							
BF80_020C	RTCCONINV	31:0	Write inverts selected bits in RTCCON, read yields undefined value							
BF80_0210	RTCALRM	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ALRMEN	CHIME	PIV	ALRMSYNC	AMASK<3:0>			
		7:0	ARPT<7:0>							
BF80_0214	RTCALRMCLR	31:0	Write clears selected bits in RTCALRM, read yields undefined value							
BF80_0218	RTCALRMSET	31:0	Write sets selected bits in RTCALRM, read yields undefined value							
BF80_021C	RTCALRMINV	31:0	Write inverts selected bits in RTCALRM, read yields undefined value							
BF80_0220	RTCTIME	31:24	HR10<3:0>				HR01<3:0>			
		23:16	MIN10<3:0>				MIN01<3:0>			
		15:8	SEC10<3:0>				SEC01<3:0>			
		7:0	—	—	—	—	—	—	—	—
BF80_0224	RTCTIMECLR	31:0	Write clears selected bits in RTCTIME, read yields undefined value							
BF80_0228	RTCTIMESET	31:0	Write sets selected bits in RTCTIME, read yields undefined value							
BF80_022C	RTCTIMEINV	31:0	Write inverts selected bits in RTCTIME, read yields undefined value							
BF80_0230	RTCDATE	31:24	YEAR10<3:0>				YEAR01<3:0>			
		23:16	MONTH10<3:0>				MONTH01<3:0>			
		15:8	DAY10<3:0>				DAY01<3:0>			
		7:0	—	—	—	—	WDAY01<3:0>			
BF80_0234	RTCDATECLR	31:0	Write clears selected bits in RTCDATE, read yields undefined value							
BF80_0238	RTCDATESET	31:0	Write sets selected bits in RTCDATE, read yields undefined value							
BF80_023C	RTCDATEINV	31:0	Write inverts selected bits in RTCDATE, read yields undefined value							
BF80_0240	ALRMTIME	31:24	HR10<3:0>				HR01<3:0>			
		23:16	MIN10<3:0>				MIN01<3:0>			
		15:8	SEC10<3:0>				SEC01<3:0>			
		7:0	—	—	—	—	—	—	—	—
BF80_0244	ALRMTIMCLR	31:0	Write clears selected bits in ALRMTIME, read yields undefined value							
BF80_0248	ALRMTIMESET	31:0	Write sets selected bits in ALRMTIME, read yields undefined value							
BF80_024C	ALRMTIMEINV	31:0	Write inverts selected bits in ALRMTIME, read yields undefined value							
BF80_0250	ALRMDATE	31:24	—	—	—	—	—	—	—	—
		23:16	MONTH10<3:0>				MONTH01<3:0>			
		15:8	DAY10<3:0>				DAY01<3:0>			
		7:0	—	—	—	—	WDAY01<3:0>			
BF80_0254	ALRMDATECLR	31:0	Write clears selected bits in ALRMDATE, read yields undefined value							
BF80_0258	ALRMDATESET	31:0	Write sets selected bits in ALRMDATE, read yields undefined value							
BF80_025C	ALRMDATEINV	31:0	Write inverts selected bits in ALRMDATE, read yields undefined value							

TABLE 21-2: RTCC INTERRUPT REGISTER SUMMARY

Virtual Address	Name		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
BF88_1070	IEC1	15:8	RTCCIE	FSCMIE	I2C2MIE	I2C2SIE	I2C2BIE	U2TXIE	U2RXIE	U2EIE
BF88_1040	IFS1	15:8	RTCCIF	FSCMIF	I2C2MIF	I2C2SIF	I2C2BIF	U2TXIF	U2RXIF	U2EIF
BF88_1110	IPC8	31:24	—	—	—	RTCCIP<2:0>			RTCCIS<1:0>	

Note: This summary table contains partial register definitions that only pertain to the RTCC peripheral. Refer to the "*PIC32MX Family Reference Manual*" (DS61132) for a detailed description of these registers.

REGISTER 21-1: RTCCON: RTC CONTROL REGISTER[1]

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	CAL<9:8>	
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CAL<7:0>							
bit 23							bit 16

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
ON	FRZ	SIDL	—	—	—	—	—
bit 15							bit 8

R/W-0	R-0	U-0	U-0	R/W-0	R-0	R-0	R/W-0
RTSECSEL	RTCCLKON	—	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-26 **Unimplemented:** Read as ‘0’

bit 25-16 **CAL<9:0>:** RTC Drift Calibration bits
Contains a signed 10-bit integer value.
0111111111= Maximum positive adjustment, adds 511 RTC clock pulses every one minute
...
0000000001= Minimum positive adjustment, adds 1 RTC clock pulse every one minute
0000000000= No adjustment
1111111111= Minimum negative adjustment, subtracts 1 RTC clock pulse every one minute
...
1000000000= Minimum negative adjustment, subtracts 512 clock pulses every one minute

bit 15 **ON:** RTCC On bit
1 = RTCC module is enabled
0 = RTCC module is disabled
Note: The ON bit is only writable when RTCWREN = 1.

bit 14 **FRZ:** Freeze in Debug Mode bit
1 = When emulator is in Debug mode, module freezes operation
0 = When emulator is in Debug mode, module continues operation
Note: The FRZ bit always reads ‘0’ unless in Debug mode.

bit 13 **SIDL:** Stop in Idle Mode bit
1 = Disables the PBCLK to the RTCC when CPU enters in Idle mode
0 = Continue normal operation in Idle mode

bit 12-8 **Unimplemented:** Read as ‘0’

bit 7 **RTSECSEL:** RTCC Seconds Clock Output Select bit
1 = RTCC seconds clock is selected for the RTCC pin
0 = RTCC alarm pulse is selected for the RTCC pin
Note: Requires RTCOE == 1 (RTCCON<0>) for the output to be active.

bit 6 **RTCCLKON:** Status of the RTCC Clock Enable bit
1 = RTCC clock is actively running
0 = RTCC clock is not running

REGISTER 21-1: RTCCON: RTC CONTROL REGISTER[1] (CONTINUED)

bit 5-4	**Unimplemented:** Read as '0'
bit 3	**RTCWREN:** RTC Value Registers Write Enable bit 1 = RTC Value registers can be written to by the user 0 = RTC Value registers are locked out from being written to by the user **Note:** The RTCWREN bit can be set only when the write sequence is enabled. The register can be written to a '0' at any time.
bit 2	**RTCSYNC:** RTCC Value Registers Read Synchronization bit 1 = RTC Value registers can change while reading due to a roll over ripple that results in an invalid data read. If the register is read twice and results in the same data, the data can be assumed to be valid. 0 = RTC Value registers can be read without concern about a roll over ripple
bit 1	**HALFSEC:** Half-Second Status bit 1 = Second half period of a second 0 = First half period of a second **Note:** This bit is read-only. It is cleared to '0' on a write to the SECONDS register.
bit 0	**RTCOE:** RTCC Output Enable bit 1 = RTCC clock output enabled – clock presented onto an I/O 0 = RTCC clock output disabled **Note:** This bit is ANDed with ON (RTCCON<15>) to produce the effective RTCC output enable.

Note 1: This register is only reset by Power-on Reset (POR).

REGISTER 21-2: RTCALRM: RTC ALARM CONTROL REGISTER(1)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
ALRMEN	CHIME	PIV	ALRMSYNC	AMASK<3:0>			
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ARPT<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ALRMEN:** Alarm Enable bit
1 = Alarm is enabled
0 = Alarm is disabled
Note: Hardware clears ALRMEN anytime the alarm event occurs, when ARPT<7:0> = 00 and CHIME = 0. This field should not be written when RTCCON = 1 (RTCCON<15>) and ALRMSYNC = 1.

bit 14 **CHIME:** Chime Enable bit
1 = Chime is enabled – ARPT<7:0> is allowed to roll over from 00 to FF
0 = Chime is disabled – ARPT<7:0> stops once it reaches 00
Note: This field should not be written when RTCCON = 1 (RTCCON<15>) and ALRMSYNC = 1.

bit 13 **PIV:** Alarm Pulse Initial Value bit
When ALRMEN = 0, PIV is writable and determines the initial value of the alarm pulse.
When ALRMEN = 1, PIV is read-only and returns the state of the alarm pulse.
Note: This field should not be written when RTCCON = 1 (RTCCON<15>) and ALRMSYNC = 1.

bit 12 **ALRMSYNC:** Alarm Sync bit
1 = ARPT<7:0> and ALRMEN may change as a result of a half-second rollover during a read. The ARPT must be read repeatedly until the same value is read twice. This must be done since multiple bits may be changing, which are then synchronized to the PB clock domain.
0 = ARPT<7:0> and ALRMEN can be read without concerns of rollover because prescaler is > 32 RTC clock away from a half-second rollover
Note: This assumes a CPU read will execute in less than 32 PBCLKs.

REGISTER 21-2: RTCALRM: RTC ALARM CONTROL REGISTER[1] (CONTINUED)

bit 11-8	**AMASK<3:0>:** Alarm Mask Configuration bits 0000 = Every half-second 0001 = Every second 0010 = Every 10 seconds 0011 = Every minute 0100 = Every 10 minutes 0101 = Every hour 0110 = Once a day 0111 = Once a week 1000 = Once a month 1001 = Once a year (except when configured for February 29th, once every 4 years) 1010 = Reserved – do not use 1011 = Reserved – do not use 11XX = Reserved – do not use **Note:** This field should not be written when RTCCON = 1 (RTCCON<15>) and ALRMSYNC = 1.
bit 7-0	**ARPT<7:0>:** Alarm Repeat Counter Value bits 11111111 = Alarm will trigger 256 times ... 00000000 = Alarm will trigger 1 time The counter decrements on any alarm event. The counter only rolls over from 00 to FF if CHIME = 1. **Note:** This field should not be written when RTCCON = 1 (RTCCON<15>) and ALRMSYNC = 1.

Note 1: This register is only reset by POR.

REGISTER 21-3: RTCTIME: RTC TIME VALUE REGISTER(1)

R-0	R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
HR10<3:0>				HR01<3:0>			
bit 31							bit 24

R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
MIN10<3:0>				MIN01<3:0>			
bit 23							bit 16

R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SEC10<3:0>				SEC01<3:0>			
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-28 **HR10<3:0>:** Binary Coded Decimal Value of Hours bits
10 digits; contains a value from 0 to 2.
Note: HR10<3:2> bits are always read ‘0’.

bit 27-24 **HR01<3:0>:** Binary Coded Decimal Value of Hours bits
1 digit; contains a value from 0 to 9.

bit 23-20 **MIN10<3:0>:** Binary Coded Decimal Value of Minutes bits
10 digits; contains a value from 0 to 5.
Note: MIN10<3> bit is always read ‘0’.

bit 19-16 **MIN01<3:0>:** Binary Coded Decimal Value of Minutes bits
1 digit; contains a value from 0 to 9.

bit 15-12 **SEC10<3:0>:** Binary Coded Decimal Value of Seconds bits
10 digits; contains a value from 0 to 5.
Note: SEC10<3> bit is always read ‘0’.

bit 11-8 **SEC01<3:0>:** Binary Coded Decimal Value of Seconds bits
1 digit; contains a value from 0 to 9.

bit 7-0 **Unimplemented:** Read as ‘0’

Note 1: This register is only writable when RTCWREN = 1 (RTCCON<3>).

REGISTER 21-4: RTCDATE: RTC DATE VALUE REGISTER(1)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
YEAR10<3:0>				YEAR01<3:0>			
bit 31							bit 24

R-0	R-0	R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
MONTH10<3:0>				MONTH01<3:0>			
bit 23							bit 16

R-0	R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
DAY10<3:0>				DAY01<3:0>			
bit 15							bit 8

U-0	U-0	U-0	U-0	R-0	R/W-x	R/W-x	R/W-x
—	—	—	—	WDAY01<3:0>			
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-28 **YEAR10<3:0>:** Binary Coded Decimal Value of Years bits (10 digits)

bit 27-24 **YEAR01<3:0>:** Binary Coded Decimal Value of Years bits (1 digit)

bit 23-20 **MONTH10<3:0>:** Binary Coded Decimal Value of Months bits (10 digits; contains a value from 0 to 1)
Note: MONTH10<3:1> bits are always read ‘0’.

bit 19-16 **MONTH01<3:0>:** Binary Coded Decimal Value of Months bits (1 digit; contains a value from 0 to 9)

bit 15-12 **DAY10<3:0>:** Binary Coded Decimal Value of Days bits (10 digits; contains a value from 0 to 3)
Note: DAY10<3:2> bits are always read ‘0’.

bit 11-8 **DAY01<3:0>:** Binary Coded Decimal Value of Days bits (1 digit; contains a value from 0 to 9)

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **WDAY01<3:0>:** Binary Coded Decimal Value of Weekdays bits (1 digit; contains a value from 0 to 6)
Note: WDAY01<3> bit is always read ‘0’.

Note 1: This register is only writable when RTCWREN = 1 (RTCCON<3>).

REGISTER 21-5: ALRMTIME: ALARM TIME VALUE REGISTER

R-0	R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
HR10<3:0>				HR01<3:0>			
bit 31							bit 24

R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
MIN10<3:0>				MIN01<3:0>			
bit 23							bit 16

R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SEC10<3:0>				SEC01<3:0>			
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-28 **HR10<3:0>:** Binary Coded Decimal Value of Hours bits (10 digit; contains a value from 0 to 2)
Note: HR10<3:2> bits are always read ‘0’.

bit 27-24 **HR01<3:0>:** Binary Coded Decimal Value of Hours bits (1 digit; contains a value from 0 to 9)

bit 23-20 **MIN10<3:0>:** Binary Coded Decimal Value of Minutes bits, (10 digit; contains a value from 0 to 5)
Note: MIN10<3> bit is always read ‘0’.

bit 19-16 **MIN01<3:0>:** Binary Coded Decimal Value of Minutes bits (1 digit; contains a value from 0 to 9)

bit 15-12 **SEC10<3:0>:** Binary Coded Decimal Value of Seconds bits (10 digit; contains a value from 0 to 5)
Note: SEC10<3> bit is always read ‘0’.

bit 11-8 **SEC01<3:0>:** Binary Coded Decimal Value of Seconds bits (1 digit; contains a value from 0 to 9)

bit 7-0 **Unimplemented:** Read as ‘0’

REGISTER 21-6: ALRMDATE: ALARM DATE VALUE REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

R-0	R-0	R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
MONTH10<3:0>				MONTH01<3:0>			
bit 23							bit 16

R-0	R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
DAY10<3:0>				DAY01<3:0>			
bit 15							bit 8

U-0	U-0	U-0	U-0	R-0	R/W-x	R/W-x	R/W-x
—	—	—	—	WDAY01<3:0>			
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-24 **Unimplemented:** Read as ‘0’.

bit 23-20 **MONTH10<3:0>:** Binary Coded Decimal Value of Months bits (10 digit; contains a value from 0 to 1)
Note: MONTH10<3:1> bits are always read ‘0’.

bit 19-16 **MONTH01<3:0>:** Binary Coded Decimal Value of Months bits (1 digit; contains a value from 0 to 9)

bit 15-12 **DAY10<3:0>:** Binary Coded Decimal Value of Days bits (10 digit; contains a value from 0 to 3)
Note: DAY10<3:2> bits are always read ‘0’.

bit 11-8 **DAY01<3:0>:** Binary Coded Decimal Value of Days bits (1 digit; contains a value from 0 to 9)

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **WDAY01<3:0>:** Binary Coded Decimal Value of Weekdays bits (1 digit; contains a value from 0 to 6)
Note: WDAY01<3> bit is always read ‘0’.

21.2 Clock Calendar Mode

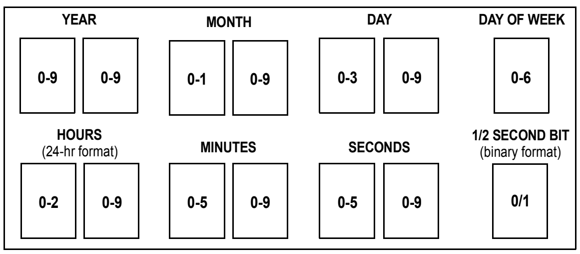
The PIC32MX RTCC module provides clock and calendar functions with the following features:

- 100-year clock and calendar with automatic leap year detection.
- Clock range from 00:00:00 (midnight) on January 1, 2000 to 23:59:59 on December 31, 2099.
- Clock granularity of one second with half-second visibility to the user.

21.2.1 RTCC CONFIGURATION

The RTCTIME and RTCDATE registers can be programmed with the desired time and date numeric values expressed in Binary Coded Decimal (BCD) format. This simplifies users' firmware as each of the digit values is contained within its own 4-bit value (see Figure 21-2).

FIGURE 21-2: TIMER DIGIT FORMAT



The user can configure the current time by simply writing the desired year, month, day, hour, minutes and seconds to the RTCTIME and RTCDATE registers. However, these registers are write-protected and require a special "unlock" sequence to be performed prior to writing to these registers. Additionally, the user should verify that the RTCSYNC bit (RTCCON<2>) = 0 (safe to access registers) for any read or write operations.

Refer to **Section 21.2.3 "Write Lock"** and Example 21-3

21.2.2 SAFETY WINDOW FOR REGISTER READS AND WRITES

The RTCTIME and RTCDATE registers can be safely accessed when the RTCC module is disabled (ON bit (RTCCON<15>) = `0`). However, when the RTCC module is enabled (ON bit = `1`), the module provides a single RTCSYNC bit (RTCCON<2>) that the user must use to determine when it is safe to read and update the time and date registers.

The RTCSYNC bit indicates a time window during which the RTCC time registers (RTCTIME, RTCDATE) are not about to be updated and can be safely read and written.

For read or write operations, the registers can be safely accessed by the CPU when RTCSYNC = `0`.

For a read operation when RTSYNC = `1`, the user must employ a firmware solution to assure that the data read did not fall on an update boundary, resulting in an invalid or partial read. For example, reading and comparing a Timer register value twice can ensure in code that the register read did not span an RTCC clock update.

Write operations to the Time and Date registers should not be performed when RTCSYNC = `1`.

Refer to Example 21-1 and Example 21-2.

EXAMPLE 21-1: UPDATING THE RTCC TIME AND DATE

```
/*
    The following code example will update the RTCC time and date.
*/
    // assume the secondary oscillator is enabled and ready, i.e.
    // OSCCON<1>=1, OSCCON<22>=1, and RTCC write is enabled i.e.
    // RTCWREN (RTCCON<3>) =1;

    unsigned long time=0x04153300;// set time to 04 hr, 15 min, 33 sec
    unsigned long date=0x06102705;// set date to Friday 27 Oct 2006

    RTCCONCLR=0x8000;              // turn off the RTCC
    while(RTCCON&0x40);            // wait for clock to be turned off
    RTCTIME=time;                  // safe to update the time
    RTCDATE=date;                  // update the date
    RTCCONSET=0x8000;              // turn on the RTCC
    while(!(RTCCON&0x40));         // wait for clock to be turned on

                                   // can disable the RTCC write
```

EXAMPLE 21-2: UPDATING THE RTCC TIME USING THE RTCSYNC WINDOW

```
/*
    The following code example will update the RTCC time and date.
*/

    // assume RTCC write is enabled i.e. RTCWREN (RTCCON<3>) =1;

    unsigned long time=0x04153300;// set time to 04 hr, 15 min, 33 sec
    unsigned long date=0x06102705;// set date to Friday 27 Oct 2006

    // disable interrupts, critical section follows
    __asm__ __volatile__ ("di");
    while((RTCCON&0x4)!=0);       // wait for not RTCSYNC
    RTCTIME=time;                 // safe to update the time
    RTCDATE=date;                 // update the date

    // restore interrupts, critical section ended
    __asm__ __volatile__ ("ei");

    // can disable the RTCC write
```

21.2.3 WRITE LOCK

In order to perform a write to any of the RTCC Time registers, the RTCWREN bit (RTCCON<3>) must be set. Setting of the RTCWREN bit is only allowed once the device level unlocking sequence has been executed. The unlocking sequence is as follows:

1. Suspend or disable all initiators that can access the peripheral bus and interrupt the unlock sequence. (i.e., DMA and Interrupts).
2. Store 0xAA996655 to the SYSKEY register.
3. Store 0x556699AA to the SYSKEY register.
4. Set RTCWREN bit into the RTCCON register.
5. Perform the device relock by writing a dummy value to the SYSKEY register.
6. Re-enable DMA and interrupts.

Note that steps 2 through 4 must be followed exactly to unlock RTCC write operations. If the sequence is not followed exactly, the RTCWREN bit will not be set.

Refer to Example 21-3 for a “C” language implementation of the write unlock operation.

EXAMPLE 21-3: WRITE UNLOCK SEQUENCE

```
// assume interrupts are disabled
// assume the DMA controller is suspended
// assume the device is locked

    // starting critical sequence
    SYSKEY = 0xaa996655;      // write first unlock key to SYSKEY
    SYSKEY = 0x556699aa;      // write second unlock key to SYSKEY
    RTCCONSET = 0x8;          // set RTCWREN in RTCCONSET
    // end critical sequence

    SYSKEY = 0x33333333;      // perform device re-lock

// can resume the DMA controller activity
// can re-enable interrupts
```

Note:	To avoid accidental writes to the RTCC time values, it is recommended that the RTCWREN bit (RTCCON<3>) is kept clear at any other time.

 Advance Information

21.3 Alarm Mode

The PIC32MX RTCC module provides alarm functions with the following features:

- One-time alarm
- Repeat alarms
- Indefinite alarm repetition
- Configurable from half-second to one year

The RTCC alarm generates an alarm event when the RTCC timer matches the masked alarm value.

The RTCC alarm functions are configurable from a half-second to one year and can repeat the alarm at preconfigured intervals. The chime feature provides indefinite repetition of the alarm.

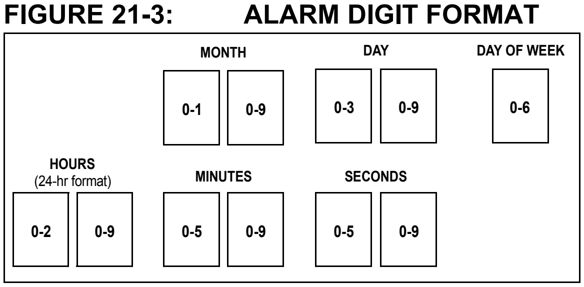
To enable the alarm feature, configure the ALRMEN bit (RTCALRM<15>) = `1`. To disble the alarm feature, configure the ALRMEN bit = `0`. An alarm event is generated when the RTCC timer matches the masked alarm registers.

Note 1:	Once the timer value reaches the alarm setting, one RTCC clock period will elapse prior to setting the alarm interrupt.
2:	IF RTCC is off (RTCCON<15> = `0`) the writable fields in the RTCALRM register can be safely modified. If RTCC is ON, the write of the RTCALRM register has to be done while ALRMSYNC = `0`. Not following the above steps can result in a false alarm event.
3:	The same applies to the ALRMTIME and ALRMDATE registers: They can be safely modified only when ALRMSYNC = `0`.

21.3.1 ALARM CONFIGURATION

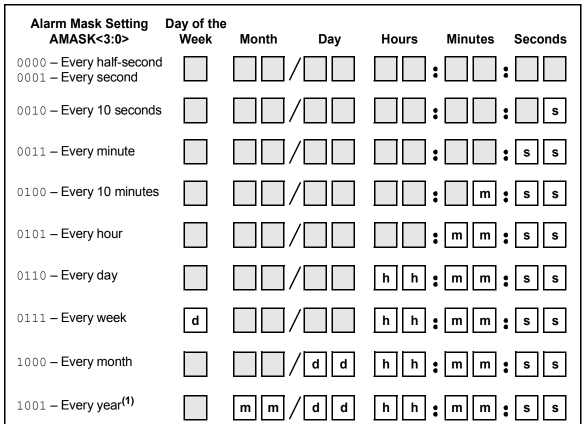
The ALRMTIME and ALRMDATE registers can be programmed with the desired time and date numeric values expressed in Binary Coded Decimal (BCD) format. This simplifies users' firmware as each of the digit values is contained within its own 4-bit value (see Figure 21-3).

FIGURE 21-3: ALARM DIGIT FORMAT



The alarm interval selection is based on the settings of the alarm mask, AMASK (RTCALRM<11:8>). The AMASK bits determine which and how many digits of the alarm must match the RTCC clock value for the alarm event to occur (see Figure 21-4).

FIGURE 21-4: ALARM MASK SETTINGS



Note 1: Annually, except when configured for February 29.

21.3.2 ONE-TIME ALARM

A single, one-time alarm can be generated by configuring the Alarm Repeat Counter bits, ARPT (RTCALRM<7:0>) = `0`, and the CHIME bit, (RTCALRM<14>) = `0`. Once the alarm event occurs, the ALRMEN bit is automatically cleared in hardware, disabling future alarms. The user must re-enable this bit for any new alarm configuration.

It is suggested to read and verify the Alarm Sync bit, ALRMSYNC (RTCALRM<12>) = `0`, before performing the following configuration:

- Disable Alarm – ALRMEN (RTCALRM<15>) = `0`.
- Disable Chime – CHIME (RTCALRM<14>) = `0`.
- Clear Alarm Repeat Counter – ARPT (RTCALRM<7:0>) = `0`.

The remaining bits are shown with example configurations and may be configured as desired:

- Configure alarm date and time – Load ALRMDATE and ALRMTIME registers with the desired alarm date/time values.
- Configure mask – Load the desired AMASK value.
- Enable Alarm – ALRMEN (RTCALRM<15>) = `0`.

Refer to Example 21-4

EXAMPLE 21-4: CONFIGURING THE RTCC FOR A ONE-TIME ALARM

```
/*
    The following code example will update the RTCC one-time alarm.
    Assumes the interrupts are disabled.
*/


    unsigned long alTime=0x16153300;// set time to 04 hr, 15 min, 33 sec
    unsigned long alDate=0x06102705;// set date to Friday 27 Oct 2006

                                    // turn off the alarm, chime and alarm repeats; clear
                                    // the alarm mask

    while(RTCALRM&0x1000);          // wait ALRMSYNC to be off
    RTCALRMCLR=0xCFFF;              // clear ALRMEN, CHIME, AMASK and ARPT;
    ALRMTIME=alTime;
    ALRMDATE=alDate;                // update the alarm time and date

    RTCALRMSET=0x8000|0x00000600;   // re-enable the alarm, set alarm mask at once per day
```

21.3.3 REPEAT ALARM

A repeat alarm can be generated by configuring the Alarm Repeat Counter bits, ARPT (RTCALRM<7:0>) = 0x00 to 0xFF (0 to 255), and the CHIME bit (RTCALRM<14>) = 0. Once the the alarm is enabled and an alarm event occurs, the ARPT count is decremented by one. Once the register reaches 0, the alarm will be generated one last time; after which point, ALRMEN bit is cleared automatically and the alarm will turn off. The user must re-enable this bit for any new alarm configuration.

Note:	An alarm event is generated when ARPT bits are = 0x00.

It is recommended to read and verify the Alarm Sync bit ALRMSYNC (RTCALRM<12>) = 0, before performing the following configuration steps:

- Disable alarm – ALRMEN (RTCALRM<15>) = 0.
- Disable chime – CHIME (RTCALRM<14>) = 0.
- Configure alarm repeat counter – ARPT (RTCALRM<7:0>) = 0x00 to 0xFF.
- Configure alarm date and time – Load ALRMDATE and ALRMTIME registers with the desired alarm date/time values.
- Configure mask – Load the desired AMASK value.
- Enable alarm – ALRMEN (RTCALRM<15>) = 0.

Refer to Example 21-5.

EXAMPLE 21-5: CONFIGURING THE RTCC FOR A TEN TIMES PER HOUR ALARM

```
/*
    The following code example will update the RTCC repeat alarm.
    Assumes the interrupts are disabled.
*/


    unsigned long alTime=0x23352300;  // set time to 23hr, 35 min, 23 sec
    unsigned long alDate=0x06111301;  // set date to Monday 13 Nov 2006

                                      // turn off the alarm, chime and alarm repeats; clear
                                      // the alarm mask
    while(RTCALRM&0x1000);            // wait ALRMSYNC to be off
    RTCALRMCLR=0xCFFF;                // clear the ALRMEN, CHIME, AMASK and ARPT;
    ALRMTIME=alTime;
    ALRMDATE=alDate;                  // update the alarm time and date
    RTCALRMSET=0x8000|0x0509;         // re-enable the alarm, set alarm mask at once per hour
                                      // for 10 times repeat
```


21.3.4 INDEFINITE ALARM

An indefinite alarm can be generated by configuring the CHIME bit (RTCALRM<14>) = 1; ARPT can be any value. Once the the alarm is enabled and an alarm event occurs, the ARPT count is decremented by one. ARPT rolls over from 0x00 to 0xFF and continues to decrement on each alarm event indefinitely. The ALRMEN bit is never automatically cleared in hardware. The user must clear this bit to disable the indefinite alarm.

> **Note:** An alarm event is generated when the ARPT are = 0x00.

It is recommended to read and verify the Alarm Sync bit, ALRMSYNC (RTCALRM<12>) = 0, before performing the following configuration:

- Disable alarm – ALRMEN (RTCALRM<15>) = 0.
- Enable chime – CHIME (RTCALRM<14>) = 1.
- Configure alarm repeat counter – ARPT (RTCALRM<7:0>) = 0 to 256.
- Configure alarm date and time – Load ALRMDATE and ALRMTIME registers with the desired alarm date/time values.
- Configure mask – Load the desired AMASK value.
- Enable Alarm – ALRMEN (RTCALRM<15>) = 0.

Refer to Example 21-6.

EXAMPLE 21-6: CONFIGURING THE RTCC FOR INDEFINITE ALARM

```
/*
    The following code example will update the RTCC indefinite alarm.
    Assumes the interrupts are disabled.
*/


    unsigned long alTime=0x23352300;  // set time to 23hr, 35 min, 23 sec
    unsigned long alDate=0x06111301;  // set date to Monday 13 Nov 2006

                                      // turn off the alarm, chime and alarm repeats; clear
                                      // the alarm mask
    while(RTCALRM&0x1000);            // wait ALRMSYNC to be off
    RTCALRMCLR=0xCFFF;                // clear ALRMEN, CHIME, AMASK, ARPT;
    ALRMTIME=alTime;
    ALRMDATE=alDate;                  // update the alarm time and date
    RTCALRMSET=0xC600;                // re-enable the alarm, set alarm mask at once per
                                      // hour, enable CHIME
```

21.4 RTCC Clock Source

The RTCC module is intended to be clocked by an external Real-Time Clock crystal that is oscillating at 32.768 kHz. To allow the RTCC to be clocked by an external 32.768 kHz crystal, the SOSCEN bit (OSCCON<1>) must be set (see **Section 10.0 "Oscillators"**) or the FSOSCEN (DEVCFG1<5>) Configuration bit must be programmed to '1'. This is the only bit outside of the RTCC module with which the user must be concerned of for enabling the RTCC. The status bit, SOSCRDY (OSCCON<22>), can be used to check that the secondary oscillator is running.

> **Note:** The RTCC does not have an exclusive access to use the SOSC oscillator. This oscillator may be used by other peripherals, such as the CPU as a low-power clock source or Timer1. Refer to the "*PIC32MX Family Reference Manual*" (DS61132) regarding the operation of the Secondary Low-Power Oscillator.

21.4.1 CALIBRATION

The real-time crystal input can be calibrated using the periodic auto-adjust feature. When properly calibrated, the RTCC can provide an error of less than 0.66 seconds per month. Calibration has the ability to eliminate an error of up to 260 ppm.

The calibration is accomplished by finding the number of error clock pulses and writing this value into the CAL field of the RTCCCON register (RTCCON<9:0>). This 10-bit signed value will either be added or subtracted from the RTCC timer, once every minute. Refer to the steps below for RTCC calibration:

1. Using another timer resource on the device, the user must find the error of the 32.768 kHz crystal.
2. Once the error is known, it must be converted to the number of error clock pulses per minute.

EQUATION 21-1: ERROR CLOCKS PER MINUTE

(Ideal Frequency (32,758) – Measured Frequency)
* 60 = Error Clocks per Minute

3. a) If the oscillator is *faster* than ideal (negative result from step 2), the CAL bits register value needs to be negative. This causes the specified number of clock pulses to be subtracted from the timer counter, once every minute.

   b) If the oscillator is *slower* than ideal (positive result from step 2), the CAL bits register value needs to be positive. This causes the specified number of clock pulses to be added to the timer counter, once every minute.

4. Load the CAL bits (RTCCON<9:0>) with the correct value.

Writes to the CAL bits should only occur when the timer is turned off, or immediately after the rising edge of the seconds pulse (except when the seconds (RTCTIME<15:8>) field is ‘00’ due to the possibility of the auto-adjust event).

> **Note:** It is up to the user, to include in the error value, the initial error of the crystal drift, due to temperature and drift due to crystal aging.
>
> A write to the seconds bits resets the state of calibration (not its value). If an adjustment just occurred, it will occur again because of the minute roll over.

EXAMPLE 21-7: UPDATING THE RTCC CALIBRATION VALUE

```
/*
    The following code example will update the RTCC calibration.
*/

    int cal=0x3FD;                  // 10 bits adjustment, -3 in value

    if(RTCCON&0x8000)
    {                               // RTCC is ON
        unsigned intt0, t1;
        do
        {
            t0=RTCTIME;
            t1=RTCTIME;
        }while(t0!=t1);             // read valid time value
        if((t0&0xFF)==00)
        {                           // we're at second 00, wait auto-adjust to be performed
            while(!(RTCCON&0x2));   // wait until second half...
        }
    }

    RTCCONCLR=0x03FF0000;           // clear the calibration
    RTCCONSET=cal;
```

21.5 RTCC Interrupts

The RTCC alarm can be configured to generate an interrupt at every alarm event. Refer to **Section 21.3 “Alarm Mode”** for details regarding the various alarm events.

The RTCC module is enabled as a source of interrupts via the respective RTCC interrupt enable bit:

- RTCCIE (IEC1<15>).

The alarm interrupt is signalled by the corresponding RTCC interrupt flag bit:

- RTCCIF (IFS1<15>).

This interrupt flag must be cleared in software.

The interrupt priority level bits and interrupt subpriority level bits must be also be configured:

- RTCCIP<2:0> (IPC8<28:26>)
- RTCCIS<1:0> (IPC8<25:24>)

In addition to enabling the RTCC interrupt, an Interrupt Service Routine, ISR, is required (see Example 21-9)..

Note:	It is the user’s responsibility to clear the corresponding interrupt flag bit before returning from an ISR.

EXAMPLE 21-8: RTCC INITIALIZATION WITH INTERRUPTS

```
/*
    The following code example illustrates an RTCC initialization with interrupts enabled.
    When the RTCC alarm interrupt is generated, the cpu will jump to the vector assigned to
    RTCC interrupt.
*/
                                  // assume RTCC write is enabled i.e. RTCWREN (RTCCON<3>) =1;
    IEC1CLR=0x00008000;           // disable RTCC interrupts

    RTCCONCLR=0x8000;             // turn off the RTCC
    while(RTCCON&0x40);           // wait for clock to be turned off

    IFS1CLR=0x00008000;           // clear RTCC existing event
    IPC8CLR=0x1f000000;           // clear the priority
    IPC8SET=0x0d000000;           // Set IPL=3, subpriority 1
    IEC1SET=0x00008000;           // Enable RTCC interrupts

    RTCTIME=0x16153300;           // safe to update time to 16 hr, 15 min, 33 sec
    RTCDATE=0x06102705;           // update the date to Friday 27 Oct 2006

    RTCALRMCLR=0xCFFF;            // clear ALRMEN, CHIME, AMASK and ARPT;
    ALRMTIME=0x16154300;          // set alarm time to 16 hr, 15 min, 43 sec
    ALRMDATE=0x06102705;          // set alarm date to Friday 27 Oct 2006

    RTCALRMSET=0x8000|0x00000600; // re-enable the alarm, set alarm mask at once per day

    RTCCONSET=0x8000;             // turn on the RTCC
    while(!(RTCCON&0x40));        // wait for clock to be turned on
```

EXAMPLE 21-9: RTCC ISR

```
/*
    The following code example demonstrates a simple interrupt service routine for RTCC
    interrupts. The user's code at this vector should perform any application specific
    operations and must clear the RTCC interrupt flag before exiting.
*/

void__ISR(_RTCC_VECTOR, ipl7) __RTCCInterrupt(void)
{
                                        // ... perform application specific operations
                                        // in response to the interrupt

    IFS1CLR=0x00008000;                 // be sure to clear RTCC interrupt flag
                                        // before exiting the service routine.
}
```

Note: The RTCC ISR code example shows MPLAB® C32 C compiler specific syntax. Refer to your compiler manual regarding support for ISRs.

21.6 I/O Pin Control

Enabling the RTCC modules configures the I/O pin direction. When the RTCC module is enabled, configured and the output enabled, the I/O pin direction is properly configured as a digital output.

The RTCC pin can be configured to toggle at every alarm or "seconds" event. To enable the RTCC pin output, set the RTCOE bit (RTCCON<0>) = `1`. To select the output to toggle on an alarm event, configure RTSECSEL bit (RTCCON<7>) = `0`. To select the output to toggle on every "seconds" update, configure RTSECSEL bit = `1`.

TABLE 21-3: I/O PIN CONFIGURATION FOR USE WITH RTCC MODULE

Required Settings for Module Pin Control							
IO Pin Name	**Required**	**Module Control**	**Bit Field**	**TRIS(4)**	**Pin Type**	**Buffer Type**	**Description**
RTCC	Yes(1)	ON and RTCOE(2)	RTSECSEL = 1	X	O	CMOS	RTCC Seconds Clock
RTCC	Yes(1)	ON and RTCOE(2)	RTSECSEL = 0 and ALRMEN and PIV(3)	X	O	CMOS	RTCC Alarm Pulse

Legend: CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output

Note 1: The RTCC pin is only required when seconds clock or alarm pulse output is needed. Otherwise, this pin can be used for general purpose IO and require the user to set the corresponding TRIS control register bit.

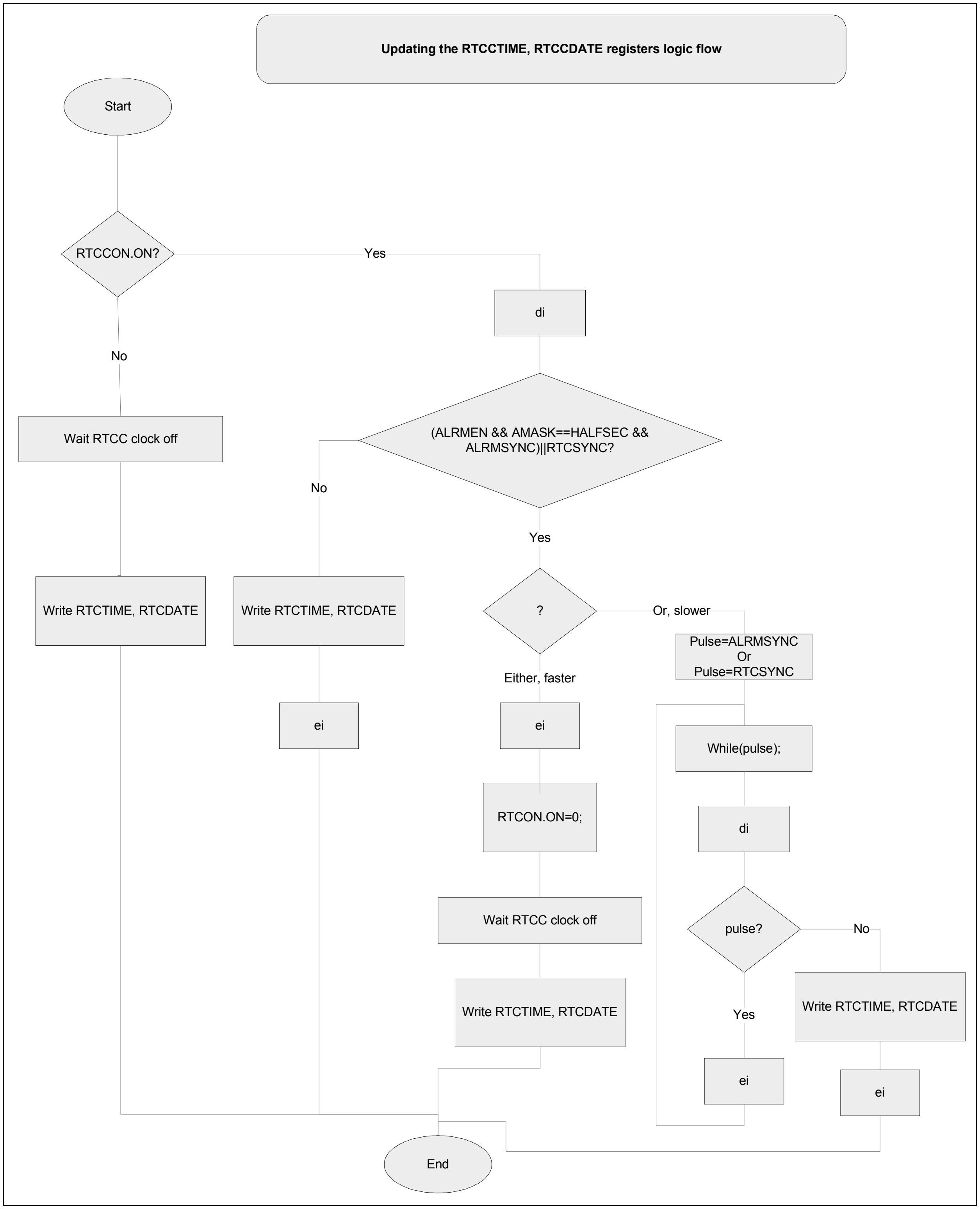
2: The ON (RTCCON<15>) and RTCOE (RTCCON<0>) bits are always required to validate the output function of the RTCC pin, either seconds clock or alarm pulse.

3: When RTSECSEL (RTCCON<7>) = `0`, the RTCC pin output is the alarm pulse. If the ALRMEN (RTCALRM<15>) = `0`, PIV (RTCALRM<13>) selects the value at the RTCC pin. When the ALRMEN = `1`, the RTCC pin reflects the state of the alarm pulse.

4: The setting of the TRIS bit is irrelevant.

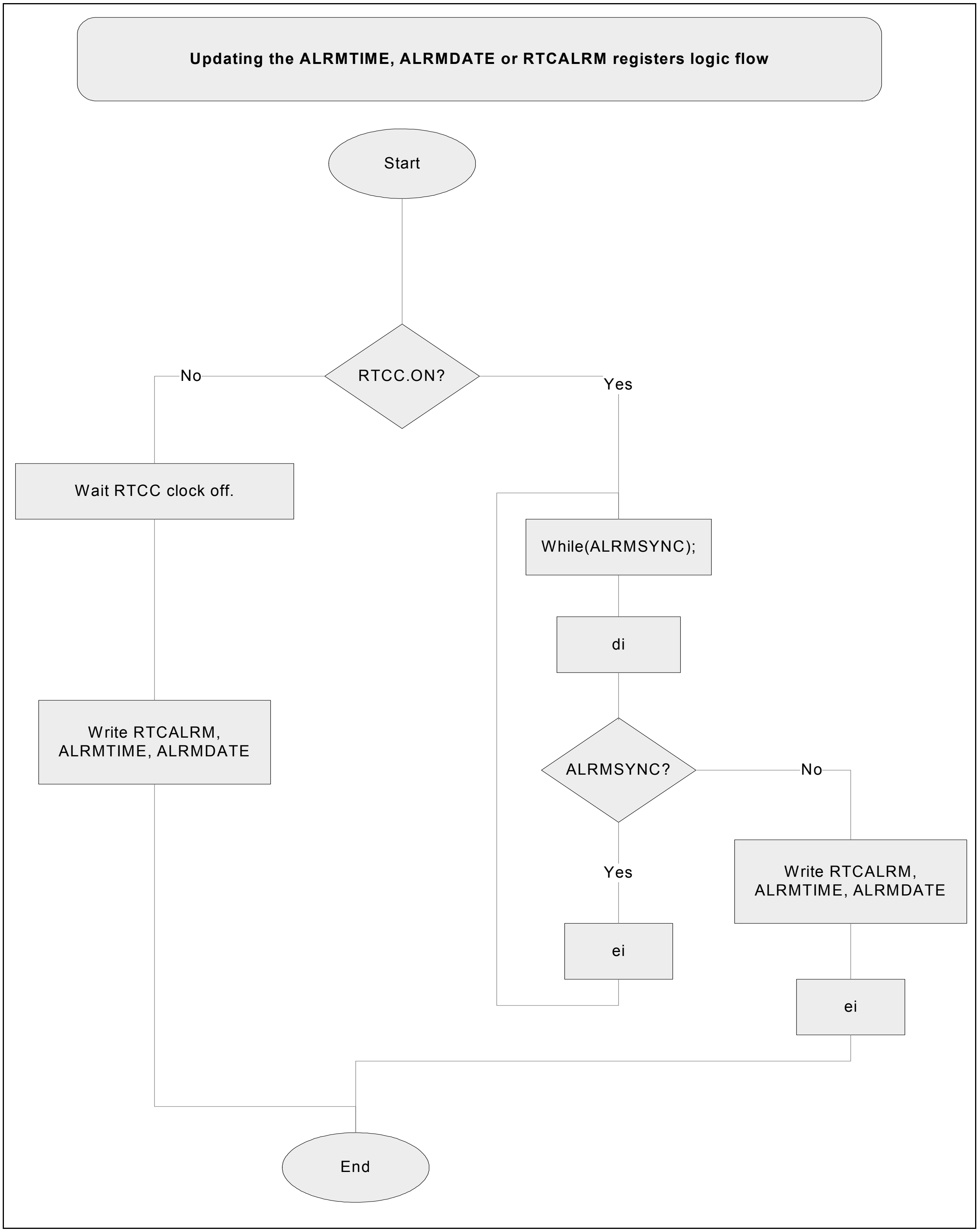
21.7 Updating the Time and Date Registers

The following flowchart explains in detail the steps that have to be performed in order to update the RTCTIME and RTCDATE registers.



21.8 Updating the Alarm Registers

The following flowchart explains in detail the steps that have to be performed in order to update the ALRMTIME, ALRMDATE and RTCALRM registers.



 Advance Information

22.0 ANALOG-DIGITAL CONVERTER

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *“PIC32MX Family Reference Manual”* (DS61132) for a detailed description of this peripheral.

The PIC32MX Family 10-bit Analog-to-Digital (A/D) converter (or ADC) includes the following features:

- Successive Approximation Register (SAR) conversion
- Up to 400 kilo samples per second (ksps) conversion speed
- Up to 16 analog input pins
- External voltage reference input pins
- One unipolar, differential Sample-and-Hold Amplifier (SHA)
- Automatic Channel Scan mode
- Selectable conversion trigger source
- 16-word conversion result buffer
- Selectable Buffer Fill modes
- Eight conversion result format options
- Operation during CPU SLEEP and IDLE modes

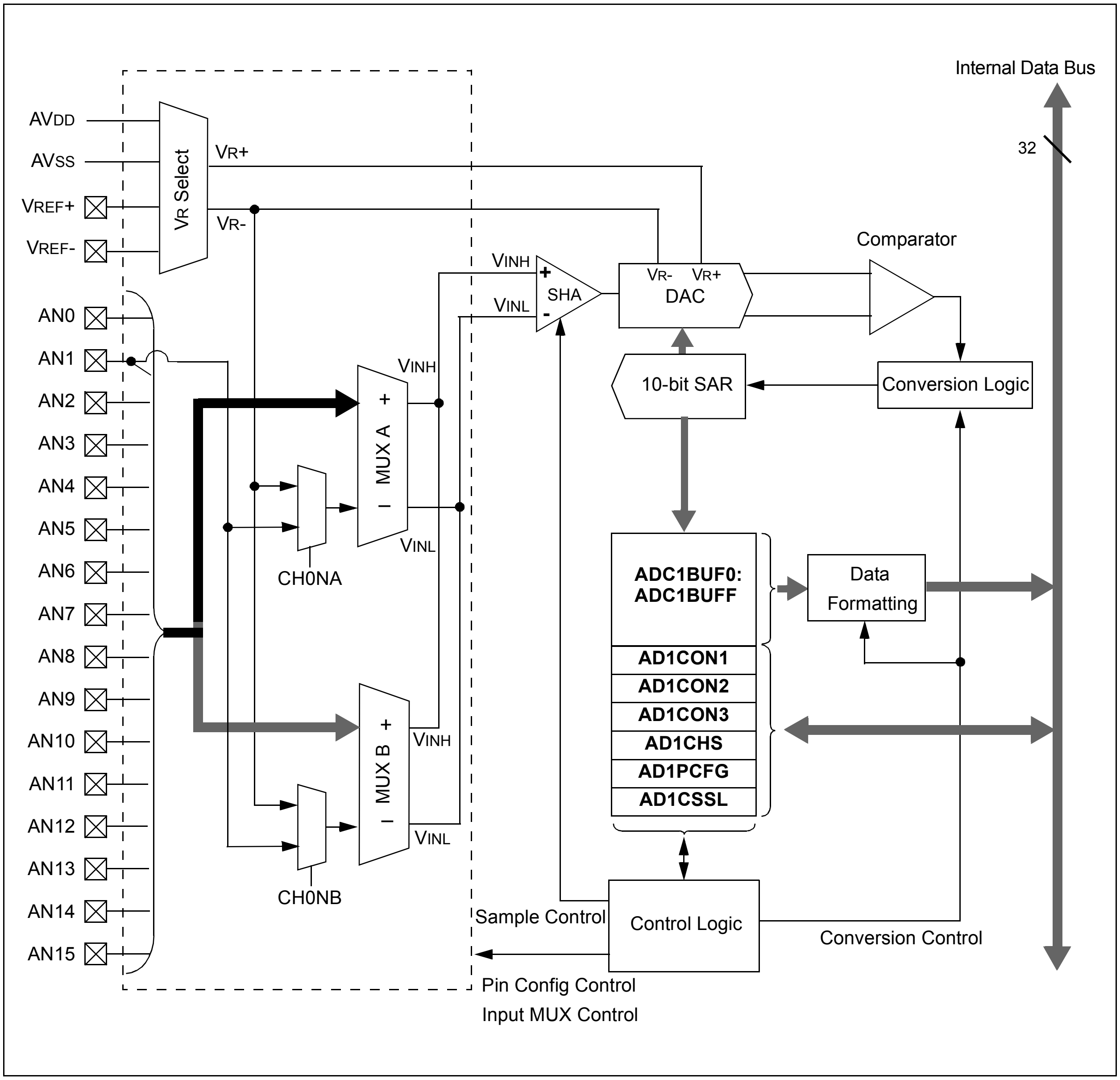
A block diagram of the 10-bit ADC is shown in Figure 22-1. The 10-bit ADC can have up to 16 analog input pins, designated AN0-AN15. In addition, there are two analog input pins for external voltage reference connections. These voltage reference inputs may be shared with other analog input pins and may be common to other analog module references. The actual number of analog input pins and external voltage reference input configuration will depend on the specific PIC32MX device. Refer to the device data sheet for further details.

The analog inputs are connected through two multiplexers (MUXs) to one SHA. The analog input MUXs can be switched between two sets of analog inputs between conversions. Unipolar differential conversions are possible on all channels, other than the pin used as the reference, using a reference input pin (see Figure 22-1).

The Analog Input Scan mode sequentially converts user-specified channels. A control register specifies which analog input channels will be included in the scanning sequence.

The 10-bit ADC is connected to a 16-word result buffer. Each 10-bit result is converted to one of eight, 32-bit output formats when it is read from the result buffer.

FIGURE 22-1: 10-BIT HIGH-SPEED A/D CONVERTER BLOCK DIAGRAM



 Advance Information

22.1 Control Registers

The ADC module includes the following Special Function Registers (SFRs):

The AD1CON1, AD1CON2 and AD1CON3 registers control the operation of the ADC module.

- AD1CON1: ADC Control Register 1
  AD1CON1CLR, AD1CON1SET, AD1CON1INV: Atomic Bit Manipulation, Write-only Registers for AD1CON1.
- AD1CON2: ADC Control Register 2
  AD1CON2CLR, AD1CON2SET, AD1CON2INV: Atomic Bit Manipulation, Write-only Registers for AD1CON2.
- AD1CON3: ADC Control Register 3
  AD1CON3CLR, AD1CON3SET, AD1CON3INV: Atomic Bit Manipulation, Write-only Registers for AD1CON3.

The AD1CHS register selects the input pins to be connected to the SHA.

- AD1CHS: ADC Input Channel Select Register
  AD1CHSCLR, AD1CHSSET, AD1CHSINV: Atomic Bit Manipulation, Write-only Registers for AD1CHS.

The AD1PCFG register configures the analog input pins as analog inputs or as digital I/O.

- AD1PCFG: ADC Port Configuration Register
  AD1PCFGCLR, AD1PCFGSET, AD1PCFGINV: Atomic Bit Manipulation, Write-only Registers for AD1PCFG.

The AD1CSSL register selects inputs to be sequentially scanned.

- AD1CSSL: ADC Input Scan Selection Register
  AD1CSSLCLR, AD1CSSLSET, AD1CSSLINV: Atomic Bit Manipulation, Write-only Registers for AD1CSSL.

The ADC module also has the following associated bits for interrupt control:

- Interrupt Request Flag Status bit (AD1IF) in IFS1: Interrupt Flag Status Register 1
- Interrupt Enable Control bit (AD1IE) in IEC1: Interrupt Enable Control Register 1
- Interrupt Priority Control bits (AD1IP<2:0>) and (AD1IS<1:0>) in IPC6: Interrupt Priority Control Register 6

22.1.1 SPECIAL FUNCTION REGISTERS ASSOCIATED WITH THE 10-BIT ADC

Table 22-1 provides a summary of all ADC-related registers, including their addresses and formats. Corresponding registers appear after the summary, followed by a detailed description of each register. All unimplemented registers and/or bits within a register read as zeros.

TABLE 22-1: ADC SFR SUMMARY

Virtual Address	Name		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
BF80_9000	AD1CON1	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL	—	—	FORM2	FORM1	FORM0
		7:0	SSRC2	SSRC1	SSRC0	CLRASAM	—	ASAM	SAMP	DONE
BF80_9004	AD1CON1CLR	31:0	Write clears selected bits in AD1CON1, read yields undefined value							
BF80_9008	AD1CON1SET	31:0	Write sets selected bits in AD1CON1, read yields undefined value							
BF80_900C	AD1CON1INV	31:0	Write inverts selected bits in AD1CON1, read yields undefined value							
BF80_9010	AD1CON2	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	VCFG2	VCFG1	VCFG0	OFFCAL	—	CSCNA	—	—
		7:0	BUFS	—	SMPI3	SMPI2	SMPI1	SMPI0	BUFM	ALTS
BF80_9014	AD1CON2CLR	31:0	Write clears selected bits in AD1CON2, read yields undefined value							
BF80_9018	AD1CON2SET	31:0	Write sets selected bits in AD1CON2, read yields undefined value							
BF80_901C	AD1CON2INV	31:0	Write inverts selected bits in AD1CON2, read yields undefined value							
BF80_9020	AD1CON3	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ADRC	—	—	SAMC4	SAMC3	SAMC2	SAMC1	SAMC0
		7:0	ADCS7	ADCS6	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0
BF80_9024	AD1CON3CLR	31:0	Write clears selected bits in AD1CON3, read yields undefined value							
BF80_9028	AD1CON3SET	31:0	Write sets selected bits in AD1CON3, read yields undefined value							
BF80_902C	AD1CON3INV	31:0	Write inverts selected bits in AD1CON3, read yields undefined value							
BF80_9040	AD1CHS	31:24	CH0NB	—	—	—	CH0SB3	CH0SB2	CH0SB1	CH0SB0
		23:16	CH0NA	—	—	—	CH0SA3	CH0SA2	CH0SA1	CH0SA0
		15:8	—	—	—	—	—	—	—	—
		7:0	—	—	—	—	—	—	—	—
BF80_9044	AD1CHSCLR	31:0	Write clears selected bits in AD1CHS, read yields undefined value							
BF80_9048	AD1CHSSET	31:0	Write sets selected bits in AD1CHS, read yields undefined value							
BF80_904C	AD1CHS1INV	31:0	Write inverts selected bits in AD1CHS, read yields undefined value							
BF80_9060	AD1PCFG	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
		7:0	PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
BF80_9064	AD1PCFGCLR	31:0	Write clears selected bits in AD1PCFG, read yields undefined value							
BF80_9068	AD1PCFGSET	31:0	Write sets selected bits in AD1PCFG, read yields undefined value							
BF80_906C	AD1PCFGINV	31:0	Write inverts selected bits in AD1PCFG, read yields undefined value							
BF80_9050	AD1CSSL	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8
		7:0	CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0

TABLE 22-1: ADC SFR SUMMARY (CONTINUED)

Virtual Address	Name		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
BF80_9050	AD1CSSLCLR	31:0	Write clears selected bits in AD1CSSL, read yields undefined value							
BF80_9050	AD1CSSLSET	31:0	Write sets selected bits in AD1CSSL, read yields undefined value							
BF80_9050	AD1CSSLINV	31:0	Write inverts selected bits in AD1CSSL, read yields undefined value							
BF80_9070	ADC1BUF0	31:0	ADC Result Word 0 (ADC1BUF0<31:0>)							
BF80_9080	ADC1BUF1	31:0	ADC Result Word 1 (ADC1BUF1<31:0>)							
BF80_9090	ADC1BUF2	31:0	ADC Result Word 2 (ADC1BUF2<31:0>)							
BF80_90A0	ADC1BUF3	31:0	ADC Result Word 3 (ADC1BUF3<31:0>)							
BF80_9070	ADC1BUF4	31:0	ADC Result Word 4 (ADC1BUF4<31:0>)							
BF80_90C0	ADC1BUF5	31:0	ADC Result Word 5 (ADC1BUF5<31:0>)							
BF80_90D0	ADC1BUF6	31:0	ADC Result Word 6 (ADC1BUF6<31:0>)							
BF80_90E0	ADC1BUF7	31:0	ADC Result Word 7 (ADC1BUF7<31:0>)							
BF80_90F0	ADC1BUF8	31:0	ADC Result Word 8 (ADC1BUF8<31:0>)							
BF80_910	ADC1BUF9	31:0	ADC Result Word 9 (ADC1BUF9<31:0>)							
BF80_9110	ADC1BUFA	31:0	ADC Result Word A (ADC1BUFA<31:0>)							
BF80_9120	ADC1BUFB	31:0	ADC Result Word B (ADC1BUFB<31:0>)							
BF80_9130	ADC1BUFC	31:0	ADC Result Word C (ADC1BUFC<31:0>)							
BF80_9140	ADC1BUFD	31:0	ADC Result Word D (ADC1BUFD<31:0>)							
BF80_9150	ADC1BUFE	31:0	ADC Result Word E (ADC1BUFE<31:0>)							
BF80_9160	ADC1BUFF	31:0	ADC Result Word F (ADC1BUFF<31:0>)							

REGISTER 22-1: AD1CON1: ADC CONTROL REGISTER 1

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
ON	FRZ	SIDL	—	—	FORM<2:0>		
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/C-0
SSRC<2:0>			CLRASAM	—	ASAM	SAMP	DONE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** ADC Operating Mode bit
1 = A/D converter module is operating
0 = A/D converter is off

bit 14 **FRZ:** Freeze in Debug Exception Mode bit
1 = Freeze operation when CPU enters Debug Exception mode
0 = Continue operation when CPU enters Debug Exception mode
Note: FRZ is writable in Debug Exception mode only. It reads ‘0’ in Normal mode.

bit 13 **SIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 12-11 **Unimplemented:** Read as ‘0’

bit 10-8 **FORM<2:0>:** Data Output Format bits
011 = Signed Fractional 16-bit (DOUT = 0000 0000 0000 0000 sddd dddd dd00 0000)
010 = Fractional 16-bit (DOUT = 0000 0000 0000 0000 dddd dddd dd00 0000)
001 = Signed Integer 16-bit (DOUT = 0000 0000 0000 0000 ssss sssd dddd dddd)
000 = Integer 16-bit (DOUT = 0000 0000 0000 0000 0000 00dd dddd dddd)
111 = Signed Fractional 32-bit (DOUT = sddd dddd dd00 0000 0000 0000 0000)
110 = Fractional 32-bit (DOUT = dddd dddd dd00 0000 0000 0000 0000 0000)
101 = Signed Integer 32-bit (DOUT = ssss ssss ssss ssss ssss sssd dddd dddd)
100 = Integer 32-bit (DOUT = 0000 0000 0000 0000 0000 00dd dddd dddd)

REGISTER 22-1: AD1CON1: ADC CONTROL REGISTER 1 (CONTINUED)

bit 7-5	**SSRC<2:0>:** Conversion Trigger Source Select bits 111 = Internal counter ends sampling and starts conversion (auto-convert) 110 = Reserved 101 = Reserved 100 = Reserved 011 = Reserved 010 = Timer 3 period match ends sampling and starts conversion 001 = Active transition on INT0 pin ends sampling and starts conversion 000 = Clearing SAMP bit ends sampling and starts conversion
bit 4	**CLRASAM:** Stop Conversion Sequence bit (when the first A/D converter interrupt is generated) 1 = Stop conversions when the first ADC interrupt is generated. Hardware clears the ASAM bit when the ADC interrupt is generated. 0 = Normal operation, buffer contents will be overwritten by the next conversion sequence
bit 3	**Unimplemented:** Read as ‘0’
bit 2	**ASAM:** ADC Sample Auto-Start bit 1 = Sampling begins immediately after last conversion completes; SAMP bit is automatically set. 0 = Sampling begins when SAMP bit is set
bit 1	**SAMP:** ADC Sample Enable bit 1 = The ADC SHA is sampling 0 = The ADC sample/hold amplifier is holding When ASAM = 0, writing ‘1’ to this bit starts sampling. When SSRC = 000, writing ‘0’ to this bit will end sampling and start conversion.
bit 0	**DONE:** A/D Conversion Status bit 1 = A/D conversion is done 0 = A/D conversion is not done or has not started Clearing this bit will not affect any operation in progress. **Note:** Bit is cleared by software, or by hardware, at the start of a new conversion.

REGISTER 22-2: AD1CON2: ADC CONTROL REGISTER 2

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0	U-0
VCFG<2:0>			OFFCAL	—	CSCNA	—	—
bit 15							bit 8

R-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS	—	SMPI<3:0>				BUFM	ALTS
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-13 **VCFG<2:0>:** Voltage Reference Configuration bits

	ADC VR+	ADC VR-
000	AVDD	AVSS
001	External VREF+ pin	AVSS
010	AVDD	External VREF- pin
011	External VREF+ pin	External VREF- pin
1xx	AVDD	AVSS

bit 12 **OFFCAL:** Input Offset Calibration Mode Select bit

1 = Enable Offset Calibration mode
VINH and VINL of the SHA are connected to VR-
0 = Disable Offset Calibration mode
The inputs to the SHA are controlled by AD1CHS or AD1CSSL

bit 11 **Unimplemented:** Read as ‘0’

bit 10 **CSCNA:** Scan Input Selections for CH0+ SHA Input for MUX A Input Multiplexer Setting bit
1 = Scan inputs
0 = Do not scan inputs

bit 9-8 **Unimplemented:** Read as ‘0’

bit 7 **BUFS:** Buffer Fill Status bit
Only valid when BUFM = 1 (ADRES split into 2 x 8-word buffers).
1 = ADC is currently filling buffer 0x8-0xF, user should access data in 0x0-0x7
0 = ADC is currently filling buffer 0x0-0x7, user should access data in 0x8-0xF

bit 6 **Unimplemented:** Read as ‘0’

REGISTER 22-2: AD1CON2: ADC CONTROL REGISTER 2 (CONTINUED)

bit 5-2	**SMPI<3:0>:** Sample/Convert Sequences Per Interrupt Selection bits `1111` = Interrupts at the completion of conversion for each 16^{th} sample/convert sequence `1110` = Interrupts at the completion of conversion for each 15^{th} sample/convert sequence . . . . . `0001` = Interrupts at the completion of conversion for each 2^{nd} sample/convert sequence `0000` = Interrupts at the completion of conversion for each sample/convert sequence
bit 1	**BUFM:** ADC Result Buffer Mode Select bit `1` = Buffer configured as two 8-word buffers, ADC1BUF(7...0), ADC1BUF(15...8) `0` = Buffer configured as one 16-word buffer ADC1BUF(15...0.)
bit 0	**ALTS:** Alternate Input Sample Mode Select bit `1` = Uses MUX A input multiplexer settings for first sample, then alternates between MUX B and MUX A input multiplexer settings for all subsequent samples `0` = Always use MUX A input multiplexer settings

REGISTER 22-3: AD1CON3: ADC CONTROL REGISTER 3

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADRC	—	—	SAMC<4:0>				
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W	R/W-0
ADCS<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ADRC:** ADC Conversion Clock Source bit
1 = ADC internal RC clock
0 = Clock derived from Peripheral Bus Clock (PBClock)

bit 14-13 **Unimplemented:** Read as ‘0’

bit 12-8 **SAMC<4:0>:** Auto-Sample Time bits
11111 = 31 TAD
· · · · ·
00001 = 1 TAD
00000 = 0 TAD (Not allowed)

bit 7-0 **ADCS<7:0>:** ADC Conversion Clock Select bits
11111111 = TPB • (ADCS<7:0> + 1) • 2 = 512 • TPB = TAD
· · · · · ·
00000001 = TPB • (ADCS<7:0> + 1) • 2 = 4 • TPB = TAD
00000000 = TPB • (ADCS<7:0> + 1) • 2 = 2 • TPB = TAD

REGISTER 22-4: AD1CHS: ADC INPUT SELECT REGISTER

R/W-0	U-0	U-0	r-0	R/W-0	R/W-0	R/W-0	R/W-0
CH0NB	—	—	—	CH0SB<3:0>			
bit 31							bit 24

R/W-0	U-0	U-0	r-0	R/W-0	R/W-0	R/W-0	R/W-0
CH0NA	—	—	—	CH0SA<3:0>			
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31 **CH0NB:** Negative Input Select for MUX B bit
1 = Channel 0 negative input is AN1
0 = Channel 0 negative input is VR-

bit 30-29 **Unimplemented:** Read as ‘0’

bit 28 **Reserved:** Reserved for future use, maintain as ‘0’

bit 27-24 **CH0SB<3:0>:** Positive Input Select for MUX B bits
1111 = Channel 0 positive input is AN15
1110 = Channel 0 positive input is AN14
1101 = Channel 0 positive input is AN13
||
||
||
0001 = Channel 0 positive input is AN1
0000 = Channel 0 positive input is AN0

bit 23 **CH0NA:** Negative Input Select for MUX A Multiplexer Setting bit(2)
1 = Channel 0 negative input is AN1
0 = Channel 0 negative input is VR-

bit 22-21 **Unimplemented:** Read as ‘0’

bit 20 **Reserved:** Reserved for future use, maintain as ‘0’

bit 19-16 **CH0SA<3:0>:** Positive Input Select for MUX A Multiplexer Setting bits
1111 = Channel 0 positive input is AN15
1110 = Channel 0 positive input is AN14
1101 = Channel 0 positive input is AN13
||
||
||
0001 = Channel 0 positive input is AN1
0000 = Channel 0 positive input is AN0

bit 15-0 **Unimplemented:** Read as ‘0’

REGISTER 22-5: AD1PCFG: ADC PORT CONFIGURATION REGISTER

r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
—	—	—	—	—	—	—	—
bit 31							bit 24

r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W	R/W-0
PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: ('0', '1', x = Unknown)		

bit 31-16 **Reserved:** Reserved for future use, maintain as '0'

bit 15-0 **PCFG<15:0>:** Analog Input Pin Configuration Control bits

1 = Analog input pin in Digital mode, port read input enabled, ADC input multiplexer input for this analog input connected to AVss

0 = Analog input pin in Analog mode, digital port read will return as a '1' without regard to the voltage on the pin, ADC samples pin voltage

Note: The AD1PCFG register functionality will vary depending on the number of ADC inputs available on the selected device. Please refer to the specific device data sheet for additional details on this register.

REGISTER 22-6: AD1CSSL: ADC INPUT SCAN SELECT REGISTER

r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
—	—	—	—	—	—	—	—
bit 31							bit 24

r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W	R/W-0
CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Reserved for future use, maintain as ‘0’

bit 15-0 **CSSL<15:0>:** ADC Input Pin Scan Selection bits
1 = Select ANx for input scan
0 = Skip ANx for input scan

Note: The AD1CSSL register functionality will vary depending on the number of ADC inputs available on the selected device. Please refer to the specific device data sheet for additional details on this register.

22.2 ADC Operation, Terminology and Conversion Sequence

This section will describe the operation the A/D converter, the steps required to configure the converter, describe the special feature of the module, and provide examples of ADC configuration with timing diagrams and charts showing the expected output of the converter.

22.2.1 OVERVIEW OF OPERATION

Analog sampling consists of two steps: acquisition and conversion (see Figure 22-2). During acquisition the analog input pin is connected to the Sample and Hold Amplifier (SHA). After the pin has been sampled for a sufficient period, the sample voltage is equivalent to the input, the pin is disconnected from the SHA to provide a stable input voltage for the conversion process. The conversion process then converts the analog sample voltage to a binary representation.

An overview of the ADC is presented in Figure 22-1. The 10-bit A/D converter has a single SHA. The SHA is connected to the analog input pins via the analog input MUXs, MUX A and MUX B. The analog input MUXs are controlled by the AD1CHS register. There are two sets of MUX control bits in the AD1CHS register. These two sets of control bits allow the two different analog input to be independently controlled. The A/D converter can optionally switch between MUX A and MUX B configurations between conversions. The A/D converter can also optionally scan through a series of analog inputs using a single MUX.

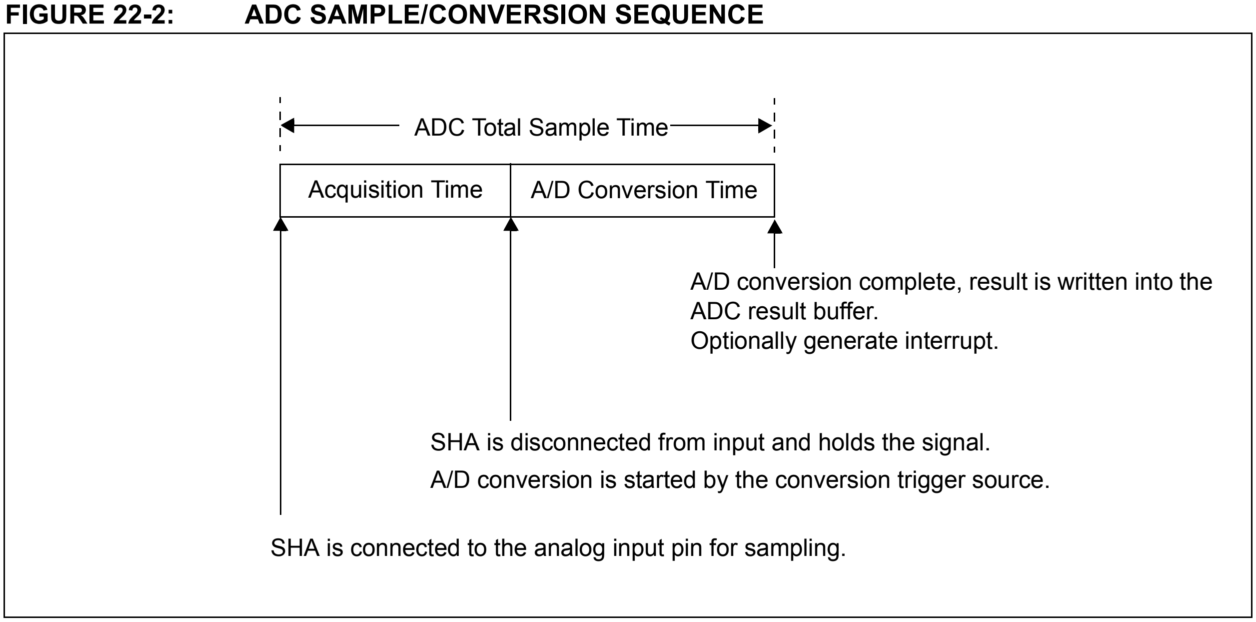
Acquisition time can be controlled manually or automatically. The acquisition time may be started manually by setting the SAMP bit (AD1CON1<1>), and ended manually by clearing the SAMP in the user software. The acquisition time may be started automatically by the A/D converter hardware and ended automatically by a conversion trigger source. The acquisition time is set by the SAMC bits (AD1CON3<12:8>). The SHA has a minimum acquisition period. Refer to the device data sheet for acquisition time specifications

Conversion time is the time required for the A/D converter to convert the voltage held by the SHA. The A/D converter requires one ADC clock cycle (T_{AD}) to convert each bit of the result, plus two additional clock cycles. Therefore, a total of 12 T_{AD} cycles are required to perform the complete conversion. When the conversion time is complete, the result is written into one of the 16 ADC result registers (ADC1BUF0...ADC1BUFF).

The sum of the acquisition time and the A/D conversion time provides the total sample time (refer to Figure 22-2). There are multiple input clock options for the A/D converter that are used to create the T_{AD} clock. The user must select an input clock option that does not violate the minimum T_{AD} specification.

The sampling process can be performed once, periodically, or based on a trigger as defined by the module configuration.

FIGURE 22-2: ADC SAMPLE/CONVERSION SEQUENCE



 Advance Information

The start time for sampling can be controlled in software by setting the SAMP control bit. The start of the sampling time can also be controlled automatically by the hardware. When the A/D converter operates in the Auto-Sample mode, the SHA is reconnected to the analog input pin at the end of the conversion in the sample/convert sequence. The auto-sample function is controlled by the ASAM control bit (AD1CON1<2>).

The conversion trigger source ends the sampling time and begins an A/D conversion or a sample/convert sequence. The conversion trigger source is selected by the control bits SSRC<2:0> (AD1CON1<7:5>). The conversion trigger can be taken from a variety of hardware sources, or can be controlled manually in software by clearing the SAMP control bit. One of the conversion trigger sources is an auto-conversion. The time between auto-conversions is set by a counter and the ADC clock. The Auto-Sample mode and auto-conversion trigger can be used together to provide endless automatic conversions without software intervention.

An interrupt may be generated at the end of each sample sequence or multiple sample sequences as determined by the value of the SMPI<3:0> (AD1CON2<5:2>). The number of sample sequences between interrupts can vary between 1 and 16. The user should note that the A/D conversion buffer holds the results of a single conversion sequence. The next sequence starts filling the buffer from the top even if the number of samples in the previous sequence was less than 16. The total number of conversion results between interrupts is the SMPI value. The total number of conversions between interrupts cannot exceed the physical buffer length.

22.3 ADC Module Configuration

Operation of the ADC module is directed through bit settings in the appropriate registers. The following instructions summarize the actions and the settings. Options and details for each configuration step are provided in subsequent sections.

1. To configure the ADC module, perform the following steps:

   A-1. Configure analog port pins in AD1PCFG<15:0>, as described in **Section 22.3.1 “Configuring Analog Port Pins”**.

   B-1. Select the analog inputs to the ADC MUXs in AD1CHS<32:0>, as described in **Section 22.3.2 “Selecting the Analog Inputs to the ADC MUXs”**.

   C-1. Select the format of the ADC result using FORM<2:0> (AD1CON1<10:8>), as described in **Section “The data in the ADC Result register can be read as one of eight formats. The format is controlled by FORM<2:0> (AD1CON1<10:8>). The user can select from integer, signed integer, fractional or signed fractional as a 16-bit or 32-bit result.”**.

   C-2. Select the sample clock source using SSRC<2:0> (AD1CON1<7:5>), as described in **Section 22.3.3.1 “Selecting the Sample Clock Source”**.

   D-1. Select the voltage reference source using VCFG<2:0> (AD1CON2<15:13>), as described in **Section 22.3.6 “Selecting the Voltage Reference Source”**.

   D-2. Select the Scan mode using CSCNA (AD1CON2<10>), as described in **Section 22.3.7 “Selecting the Scan Mode”**.

   D-3. Set the number of conversions per interrupt SMPI<3:0> (AD1CON2<5:2>), if interrupts are to be used, as described in **Section 22.3.8 “Setting the Number of Conversions per Interrupt”**.

   D-4. Set Buffer Fill mode using BUFM (AD1CON2<1>), as described in **Section 22.3.9 “Buffer Fill Mode”**.

   D-5. Select the MUX to be connected to the ADC in ALTS (AD1CON2<0>), as described in **Section 22.3.10 “Selecting the MUX to be Connected to the ADC (Alternating Sample Mode)”**.

   E-1. Select the ADC clock source using ADRC (AD1CON3<15>), as described in **Section 22.3.11 “Selecting the ADC Conversion Clock Source and Prescaler”**.

   E-2. Select the sample time using SAMC<4:0> (AD1CON3<12:8>), if auto-convert is to be used, as described in **Section 22.3.12 “Acquisition Time Considerations”**.

   E-3. Select the ADC clock prescaler using ADCS<7:0> (AD1CON3<7:0>), as described in **Section 22.3.11 “Selecting the ADC Conversion Clock Source and Prescaler”**.

   F. Turn on ADC module using AD1CON1<15>, as described in **Section 22.3.13 “Turning the ADC On”**.

Note:	Steps A through E, above, can be performed in any order, but Step F must be the final step in every case.

2. To configure ADC interrupt (if required).

   A-1. Clear AD1IF bit (IFS1<1>), as described in **Section 9.0 “Interrupts”**.

   A-2. Select ADC interrupt priority AD1IP<2:0> (IPC<28:26>) and sub priority AD1IS<1:0> (IPC<24:24>), as described in **Section 9.0 “Interrupts”**, if interrupts are to be used.

3. Start the conversion sequence by initiating

sampling, as described in **Section 22.3.14 “Initiating Sampling”**.

22.3.1 CONFIGURING ANALOG PORT PINS

The AD1PCFG register and the TRISB register control the operation of the ADC port pins.

AD1PCFG specifies the configuration of device pins to be used as analog inputs. A pin is configured as an analog input when the corresponding PCFGn bit (AD1PCFG<n>) = `0`. When the bit = `1`, the pin is set to digital control. When configured for analog input, the associated port I/O digital input buffer is disabled so it does not consume current. The AD1PCFG register is cleared at Reset, causing the ADC input pins to be configured for analog input by default at Reset.

TRIS registers control the digital function of the port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bit set, specifying the pin as an input. If the I/O pin associated with an ADC input is configured as an output, the TRIS bit is cleared and the ports digital output level (V_{OH} or V_{OL}) will be converted. After a device Reset, all TRIS bits are set.

Notes:	When reading a PORT register that shares pins with the ADC, any pin configured as an analog input reads as a ‘`0`’ when the PORT latch is read. Analog levels on any pin that is defined as a digital input (including the AN15:AN0 pins), but is not configured as an analog input, may cause the input buffer to consume current that is out of the device’s specification.

22.3.2 SELECTING THE ANALOG INPUTS TO THE ADC MUXS

The AD1CHS register is used to select which analog input pin is connected to MUX A and MUX B. Each MUX has two inputs referred to as the positive and the negative input. The positive input to MUX A is controlled by CH0SA<4:0> and the negative input is controlled by CH0NA. The positive input for MUX B is controlled by CH0SB<4:0> and the negative input is controlled by CH0NB.

The positive input can be selected from any one of the available analog input pins. The negative input can be selected as the ADC negative reference or AN0. The use of AN0 as the negative input allows the ADC to be used in a Unipolar Differential mode. Refer to the device data sheet for AN0 input voltage restrictions when used as a negative reference.

22.3.3 SELECTING THE FORMAT OF THE ADC RESULT

The data in the ADC Result register can be read as one of eight formats. The format is controlled by FORM<2:0> (AD1CON1<10:8>). The user can select from integer, signed integer, fractional or signed fractional as a 16-bit or 32-bit result.

22.3.3.1 Selecting the Sample Clock Source

It is often desirable to synchronize the end of sampling and the start of conversion with some other time event. The ADC module may use one of four sources as a conversion trigger. The selection of the conversion trigger source is controlled by the SSRC<2:0> (AD1CON1<7:5>) bits.

22.3.3.2 Manual Conversion

To configure the ADC to end sampling and start a conversion when SAMP is cleared (= `0`), SSRC is set to ‘`000`’.

22.3.3.3 Timer Compare Trigger

The ADC is configured for this Trigger mode by setting SSRC<2:0> = `010`. When a period match occurs for the 32-bit timer, TMR3/TMR2, or the 16-bit Timer3, a special A/D converter trigger event signal is generated by Timer3.

22.3.3.3.1 External INT0 Pin Trigger

To configure the ADC to begin a conversion on an active transition on the INT0 pin, SSRC<2:0> is set to '001'. The INT0 pin may be programmed for either a rising edge input or a falling edge input to trigger the conversion process.

22.3.3.3.2 Auto-Convert

The ADC can be configured to automatically perform conversions at the rate selected by the Auto-Sample Time bits, SAMC<4:0>. The ADC is configured for this Trigger mode by setting SSRC<2:0> = 111. In this mode, the ADC will perform continuous conversions on the selected channels.

22.3.4 SYNCHRONIZING ADC OPERATIONS TO INTERNAL OR EXTERNAL EVENTS

The modes where an external event trigger pulse ends sampling and starts conversion (SSRC2:SSRC0 = 001, 010 or 011) may be used in combination with auto-sampling (ASAM = 1) to cause the ADC to synchronize the sample conversion events to the trigger pulse source. For example, where SSRC = 010 and ASAM = 1, the ADC will always end sampling and start conversions synchronously with the timer compare trigger event. The ADC will have a sample conversion rate that corresponds to the timer comparison event rate.

22.3.5 SELECTING AUTOMATIC OR MANUAL SAMPLING

Sampling can be started manually or automatically when the previous conversion is complete.

22.3.5.1 Manual

Clearing the ASAM (AD1CON1<2>) bit disables the Auto-Sample mode. Acquisition will begin when the SAMP (AD1CON1<1>) bit is set by software. Acquisition will not resume until the SAMP bit is once again set.

22.3.5.2 Automatic

Setting the ASAM (AD1CON1<2>) bit enables the Auto-Sample mode. In this mode, the sampling will start automatically after the pervious sample has been converted.

22.3.6 SELECTING THE VOLTAGE REFERENCE SOURCE

The user can select the voltage reference for the ADC module. The reference can be internal or external.

The VCFG<2:0> control bits (AD1CON2<15:13>) select the voltage reference for A/D conversions. The upper voltage reference (VR+) and the lower voltage reference (VR-) may be the internal AVDD and AVSS voltage rails, or the VREF+ and VREF- input pins.

22.3.7 SELECTING THE SCAN MODE

The ADC module has the ability to scan through a selected vector of inputs. The CSCNA bit (AD1CON2<10>) enables the MUX A input to be scanned across a selected number of analog inputs.

22.3.7.1 Scan Mode Enable

Scan mode is enabled by setting CSCNA (AD1CON2<10>). When Scan mode is enabled, the positive input of MUX A is controlled by the contents of the AD1CSSL register. Each bit in the AD1CSSL register corresponds to an analog input. Bit 0 corresponds to AN0, bit 1 corresponds to AN1 and so on. If a particular bit in the AD1CSSL register is '1', the corresponding input is part of the scan sequence.

22.3.7.2 Using Scan and Alternate Modes Together

The Scan and Alternate modes may be combined to allow a vector of inputs to be scanned and a single input to be converted every other sample.

This mode is enabled by setting the CSCNA bit = 1, and setting the ALTS (AD1CON2<0>) bit = 1.

The CSCNA bit enables the scan for MUX A, and the CH0SB<3:0> (AD1CHS<27:24>) and CH0NB (AD1CHS<31>) are used to configure the inputs to MUX B. Scanning only applies to the MUX A input selection. The MUX B input selection, as specified by CH0SB<3:0>, will still select a single input.

22.3.8 SETTING THE NUMBER OF CONVERSIONS PER INTERRUPT

The SMPI<3:0> bits (AD1CON2<5:2>) select how many A/D conversions will take place before a CPU interrupt is generated. This also defines the number of locations that will be written in the result buffer stating with ADC1BUF0 (ADC1BUF0 or ADC1BUF8 for Dual Buffer mode). This can vary from 1 sample to 16 samples (1 to 8 samples for Dual Buffer mode). After the interrupt is generated, the sampling sequence restarts; with the result of the first sample being written to the first buffer location.

The data in the result registers will be overwritten by the next sampling sequence. The data in the result buffer must be read before the completion of the first sample after the interrupt is generated.

22.3.9 BUFFER FILL MODE

The Buffer Fill mode allows the output buffer to be used as a single, 16-word buffer or two, 8-word buffers.

When BUFM is '0', the complete 16-word buffer is used for all conversion sequences. Conversion results will be written sequentially in the buffer, starting at ADC1BUF0 until the number of samples as defined by SMPI<3:0> (AD1CON2<5:2>) is reached. The next conversion result will be written to ADC1BUF0 and the process repeats. If the ADC interrupt is enabled, an interrupt will be generated when the number of samples in the buffer equals SMPI<3:0>.

When the BUFM bit (AD1CON2<1>) is '1', the 16-word results buffer (ADRES) will be split into two 8-word groups. Conversion results will be written sequentially into the first buffer starting at ADC1BUF0, BUFS (AD1CON2<7>) will be cleared, until the number of samples as defined by SMPI<3:0> (AD1CON2<5:2>) is reached. The ADC interrupt flag will then be set.

After the ADC interrupt flag is set, the following result will be written sequentially to the second buffer, starting at ADC1BUF8 The next conversion result will be written to the second buffer; starting at ADC1BUF8, BUFS (AD1CON2<7>) will be set until the number of samples as defined by SMPI<3:0> (AD1CON2<5:2>) is reached. The ADC interrupt flag will then be set.

The process then restarts with BUFS = 0 and the results being written to the first buffer.

22.3.10 SELECTING THE MUX TO BE CONNECTED TO THE ADC (ALTERNATING SAMPLE MODE)

The ADC has two input MUXs that connect to the SHA. These MUXs are used to select which analog input is to be sampled. Each of the MUXs have a positive and a negative input.

22.3.10.1 Single Input Selection

The user may select one of up to 16 analog inputs, as determined by the number of analog channels on the device, as the positive input of the SHA. The CH0SA<3:0> bits (AD1CHS<19:16>) select the positive analog input.

The user may select either V_{R-} or AN1 as the negative input. The CH0NA bit (AD1CHS<23>) selects the analog input for the negative input of channel 0. Using AN1 as the negative input allows unipolar differential measurements.

The ALTS bit (AD1CON2<0>) must be clear for this mode of operation.

22.3.10.2 Alternating Input Selections

The ALTS bit causes the module to alternate between the two input MUXs.

The inputs specified by CH0SA<3:0> and CH0NA are called the MUX A inputs. The inputs specified by CH0SB<3:0> and CH0NB are called the MUX B inputs.

When ALTS is '1', the module will alternate between the MUX A inputs on one sample and the MUX B inputs on the subsequent sample. When ALTS is '0', only the inputs specified by CH0SA<3:0> and CH0NA are selected for sampling.

22.3.11 SELECTING THE ADC CONVERSION CLOCK SOURCE AND PRESCALER

The ADC module can use the internal RC oscillator or the PBCLK as the conversion clock source.

When the internal RC oscillator is used as the clock source, ADRC (AD1CON3<15>) = 1, the T_{AD} is the period of the oscillator, no prescaler are used. When using the internal oscillator the ADC can continue to function in SLEEP and in IDLE.

When the PBCLK is used as the conversion clock source, ADRC = 0, the T_{AD} is the period of the PBCLK after the prescaler ADCS<7:0> (AD1CON3<7:0>) is applied.

The A/D converter has a maximum rate at which conversions may be completed. An analog module clock, T_{AD}, controls the conversion timing. The A/D conversion requires 12 clock periods (12 T_{AD}).

The period of the ADC conversion clock is software selected using a 8-bit counter. There are 256 possible options for T_{AD}, specified by the ADCS<7:0> bits (AD1CON3<7:0>).

Equation 22-3 gives the T_{AD} value as a function of the ADCS control bits and the device instruction cycle clock period, T_{CY}.

EQUATION 22-3: ADC CONVERSION CLOCK PERIOD

$$T_{AD} = \frac{T_{PB}(ADCS + 1)}{2}$$

$$ADCS = \frac{2 \cdot T_{AD}}{T_{PB}} - 1$$

For correct A/D conversions, the ADC conversion clock (T_{AD}) must be selected to meet the minimum T_{AD} time.

EQUATION 22-4: AVAILABLE SAMPLING TIME, SEQUENTIAL SAMPLING

$$T_{SMP} = \text{Trigger Pulse Interval } (T_{SEQ}) - \text{Conversion Time } (T_{CONV})$$

$$T_{SMP} = T_{SEQ} - T_{CONV}$$

Note: T_{SEQ} is the trigger pulse interval time.

22.3.12 ACQUISITION TIME CONSIDERATIONS

Different acquisition/conversion sequences provide different times for the sample-and-hold channel to acquire the analog signal. The user must ensure the acquisition time meets the sampling requirements.

When SSRC<2:0> (AD1CON1<7:5>) = `111`, the conversion trigger is under ADC clock control. The SAMC<4:0> bits (AD1CON3<12:8>) select the number of T_{AD} clock cycles between the start of acquisition and the start of conversion. This trigger option provides the fastest conversion rates on multiple channels. After the start of acquisition, the module will count a number of T_{AD} clocks specified by the SAMC bits.

22.3.13 TURNING THE ADC ON

When the ON bit (AD1CON1<15>) is '`1`', the module is in Active mode and is fully powered and functional.

When ON is '`0`', the module is disabled. The digital and analog portions of the circuit are turned off for maximum current savings.

In order to return to the Active mode from the Off mode, the user must wait for the analog stages to stabilize. For the stabilization time, refer to the Electrical Characteristics section of the device data sheet.

Note: Writing to ADC control bits other than ON (AD1CON1<15>), SAMP (AD1CON1<1>), and DONE (AD1CON1<0>) is not recommended while the A/D converter is running.

22.3.14 INITIATING SAMPLING

22.3.14.1 Manual Mode

In manual sampling, a acquisition is started by writing a '`1`' to the SAMP (AD1CON1<1>) bit. Software must manually manage the start and end of the acquisition period by setting SAMP and then clearing SAMP after the desired acquisition period has elapsed.

22.3.14.2 Auto-Sample Mode

In Auto-Sample mode, the sampling process is started by writing a '`1`' to the ASAM (AD1CON1<2>) bit. In Auto-Sample mode, the acquisition period is defined by ADCS<7:0> (AD1CON3<7:0>). Acquisition is automatically started after a conversion is completed. Auto-Sample mode can be used with any trigger source other than manual.

22.4 Miscellaneous ADC Functions

The following section describes bits not covered in the previous section.

22.4.1 Aborting Sampling

Clearing the SAMP (AD1CON1<1>) bit while in Manual Sample mode will terminate sampling, but may also start a conversion if SSRC (AD1CON1<7:5>) = `000`.

Clearing the ASAM (AD1CON1<2>) bit while in Auto-Sample mode will not terminate an ongoing acquire/convert sequence, however, sampling will not automatically resume after the current sample is converted.

22.4.2 ABORTING A CONVERSION

Clearing the ON (AD1CON1<15>) bit during a conversion will abort the current conversion. The ADC Result register will NOT be updated with the partially completed A/D conversion sample. That is, the corresponding result buffer location will continue to contain the value of the last completed conversion (or the last value written to the buffer).

22.4.3 BUFFER FILL STATUS

When the conversion result buffer is split using the BUFM control bit, the BUFS Status bit (AD1CON2<7>) indicates which half of the buffer the A/D converter is currently filling. If BUFS = `0`, then the A/D converter is filling ADC1BUF0-ADC1BUF7 and the user software should read conversion values from ADC1BUF8-ADC1BUFF. If BUFS = `1`, the situation is reversed and the user software should read conversion values from ADC1BUF0-ADC1BUF7.

22.4.4 OFFSET CALIBRATION

The ADC module provides a method of measuring the internal offset error. After this offset error is measured, it can be subtracted, in software, from the result of an A/D conversion. Use the following steps to perform an offset measurement:

1. Configure the A/D converter in the same manner as it will be used in the application.
2. Set the OFFCAL bit (AD1CON2<12>). This overrides the input selections and connects the sample and hold inputs to AVss.
3. If auto-sample is used set the CLRASAM bit (AD1CON1<4>) to force conversions.
4. Enable the A/D converter and perform a conversion. The result that is written to the ADC result buffer is the internal offset error.
5. Clear the OFFCAL (AD2CON<12>) bit to return the A/D converter to normal operation.

Note: Only positive ADC offsets can be measured with this method.

22.4.5 TERMINATE CONVERSION SEQUENCE AFTER AN INTERRUPT

The CLRASAM bit provides a method to terminate auto-sample after the first sequence is completed. Setting the CLRASAM and starting an auto-sample sequence will cause the A/D converter to complete one auto-sample sequence (the number of samples as defined by SMPI<3:0> (AD1CON2<5:2>)). Hardware will clear ASAM (AD1CON1<2>) and set the interrupt flag. This will stop the sampling process to allow inspection of the result buffer without results being overwritten by the next automatic conversion sequence. The CLRASAM must be cleared by software to disable this mode.

Note:	Disabling interrupts or masking the ADC interrupt has no effect on the operation of the CLRASAM bit.

22.4.6 CONVERSION SEQUENCE EXAMPLES

The following configuration examples show the ADC operation in different sampling and buffering configurations. In each example, setting the ASAM bit starts automatic sampling. A conversion trigger ends sampling and starts conversion.

22.4.7 MANUAL CONVERSION CONTROL

When SSRC<2:0> = `000`, the conversion trigger is under software control. Clearing the SAMP bit (AD1CON1<1>) starts the conversion sequence. See Example 22-1 for sample code to manually control the sampling of a single channel.

EXAMPLE 22-1: CONVERTING 1 CHANNEL, MANUAL SAMPLE START, MANUAL CONVERSION START CODE

```
AD1PCFG = 0xFFFB;              // PORTB = Digital; RB2 = analog
AD1CON1 = 0x0000;              // SAMP bit = 0 ends sampling ...
                               // and starts converting
AD1CHS = 0x00020000;           // Connect RB2/AN2 as CH0 input  ..
                               // in this example RB2/AN2 is the input
AD1CSSL = 0;
AD1CON3 = 0x0002;              // Manual Sample, Tad = internal 2 Tcy
AD1CON2 = 0;


AD1CON1SET = 0x8000;           // turn ADC ON
while (1)                      // repeat continuously
{
  AD1CON1SET = 0x0002;         // start sampling ...
  DelayNmSec(100);             //  for 100 mS
  AD1CON1CLR = 0x0002;         // start Converting
  while (!(AD1CON1 & 0x0001));// conversion done?
  ADCValue = ADC1BUF0;         // yes then get ADC value
}                              // repeat
```

22.4.8 AUTOMATIC ACQUISITION

Automatic acquisition control is enabled by setting the ASAM (AD1CON1<2>) bit. Setting the ASAM bit initiates automatic acquisition, and clearing the SAMP (AD1CON1<1>) bit terminates sampling and starts conversion. After the conversion completes, the module will automatically return to an acquisition state. The SAMP bit is automatically set at the start of the acquisition interval. The user software must time the clearing of the SAMP bit to ensure adequate acquisition time of the input signal, understanding that the time between clearing of the SAMP bit includes the conversion time as well as the acquisition time. See Example 22-2 for a code example.

EXAMPLE 22-2: CONVERTING 1 CHANNEL, AUTOMATIC SAMPLE START, MANUAL CONVERSION START CODE

```
AD1PCFG = 0xFF7F;            // all PORTB = Digital but RB7 = analog
AD1CON1 = 0x0004;            // ASAM bit = 1 implies acquisition ..
                             // starts immediately after last
                             // conversion is done
AD1CHS = 0x00070000;         // Connect RB7/AN7 as CH0 input  ..
                             // in this example RB7/AN7 is the input
AD1CSSL = 0;
AD1CON3 = 0x0002;            // Sample time manual, Tad = internal 2 Tcy
AD1CON2 = 0;


AD1CON1SET = 0x8000;         // turn ADC ON
while (1)                    // repeat continuously
{
 DelayNmSec(100);            // sample for 100 mS
 AD1CON1SET = 0x0002;        // start Converting
 while (!(AD1CON1 & 0x0001));// conversion done?
 ADCValue = ADC1BUF0;        // yes then get ADC value
}                            // repeat
```

22.4.9 CLOCKED CONVERSION TRIGGER

When SSRC<2:0> = `111`, the conversion trigger is under ADC clock control. The SAMC bits (AD1CON3<4:0>) select the number of T_{AD} clock cycles between the start of acquisition and the start of conversion. This trigger option provides the fastest conversion rates on multiple channels. After the start of acquisition, the module will count a number of T_{AD} clocks specified by the SAMC bits.

EQUATION 22-1: CLOCKED CONVERSION TRIGGER TIME

$$T_{SMP} = SAMC<4:0> * T_{AD}$$

SAMC must always be programmed for at least one clock cycle. See Example 22-1 for a code example.

Example 22-1: Converting 1 Channel, Manual Sample Start, TAD Based Conversion Start Code

```
AD1PCFG = 0xEFFF;                     // all PORTB = Digital; RB12 = analog
AD1CON1 = 0x00E0;                     // SSRC bit = 111 implies internal
                                      // counter ends sampling and starts
                                      // converting.
AD1CHS = 0x000C0000;                  // Connect RB12/AN12 as CH0 input  ..
                                      // in this example RB12/AN12 is the input
AD1CSSL = 0;
AD1CON3 = 0x1F02;                     // Sample time = 31Tad
AD1CON2 = 0;


AD1CON1SET = 0x8000;                  // turn ADC ON
while (1)                             // repeat continuously
{
   AD1CON1CLR = 0x0002;               // start sampling then ...
                                      // after 31Tad go to conversion
   while (!(AD1CON1 & 0x0001));       // conversion done?
   ADCValue = ADC1BUF0;               // yes then get ADC value
}                                     // repeat
```


22.4.10 Free-Running Sample Conversion Sequence

The Auto-Convert Conversion Trigger mode (SSRC = 111) in combination with the Automatic Sampling Start mode (ASAM = 1), allows the ADC module to schedule acquisition/conversion sequences with no intervention by the user or other device resources. This "Clocked" mode allows continuous data collection after module initialization. See Example 22-3 for a code example.

EXAMPLE 22-3: CONVERTING 1 CHANNEL, AUTO-SAMPLE START, AUTO-CONVERT CODE

```
AD1PCFG = 0xFFFB;                     // all PORTB = Digital; RB2 = analog
AD1CON1 = 0x00E0;                     // SSRC bit = 111 internal
                                      // counter ends sampling and starts
                                      // converting.
AD1CHS  = 0x00020000;                 // Connect RB2/AN2 as CH0 input  ..
                                      // in this example RB2/AN2 is the input
AD1CSSL = 0;
AD1CON3 = 0x0F00;                     // Sample time = 15Tad
AD1CON2 = 0x0004;                     // Interrupt after every 2 samples



AD1CON1SET = 0x8000;                  // turn ADC ON
AD1CON1SET = 0x0004;                  // auto start sampling

while (1)                             // repeat continuously
{  IFS1CLR = 0x0002;                  // clear ADC interrupt flag
                                        // for 31Tad then go to conversion
   while (!IFS1 & 0x0002);            // poll for conversion done\

                                      // result of conversions is available in ADC1BUF0
                                      // and ADC1BUF1
```

22.4.11 SAMPLING A SINGLE CHANNEL MULTIPLE TIMES

In this case, one ADC input, AN0, will be acquired and converted. The results are stored in the ADC1BUF buffer. This process repeats 15 times until the buffer is full, and then the module generates an interrupt. Then entire process repeats.

With ALTS (AD1CON2<0>) clear, only the MUX A inputs are active. The CH0SA (AD1CHS<19:16>) bits and CH0NA (AD1CHS<23>) bit are specified (AN0-VREF-) as the input to the sample/hold channel. Other input selection bits are not used.

22.4.12 EXAMPLE: A/D CONVERSIONS WHILE SCANNING THROUGH ANALOG INPUTS

A typical setup might include all available analog input channels to be sampled and converted. The CSCNA (AD1CON2<10>) bit specifies scanning of the ADC inputs. Other conditions are similar to the previous example (see **Section 22.4.11 "Sampling a Single Channel Multiple Times"**).

Initially, the AN0 input is acquired and converted. The result is stored in the ADC1BUF buffer. Then the AN1 input is acquired and converted. This process of scanning the inputs repeats 16 times until the buffer is full and then the module generates an interrupt. Then the entire process repeats.

22.4.12.1 Example: Using Dual 8-Word Buffers

To enable the dual 8-word buffers and alternating the buffer fill, set the BUFM (AD1CON2<1>) bit. The BUFM setting does not affect other operational parameters. First, the conversion sequence starts filling the buffer at ADC1BUF0 (buffer location 0 x 0). After the first interrupt occurs, the buffer begins to fill at ADC1BUF8 (buffer location 0 x 8). The BUFS (AD1CON2<7>) bit is alternately set and cleared after each interrupt to show which buffer is being filled. In this example, three analog inputs are sampled and an interrupt occurs after every third sample.

22.4.12.2 Example: Using Alternating MUX A, MUX B Input Selections

Setting the ALTS (AD1CON2<0>) bit enables alternating input selections. The first sample uses the MUX A inputs specified by the CH0SA (AD1CHS<19:16>) and CH0NA (AD1CHS<23>) bits. The next sample uses the MUX B inputs specified by the CH0SB (AD1CHS<27:24>) and CH0NB (AD1CHS<31>) bits.

In the following example, one of the MUX B input specifications uses 2 analog inputs as a differential source to the sample/hold.

This example also demonstrates use of the dual 8-word buffers. An interrupt occurs after every 4th sample, which results in filling 4-words into the buffer on each interrupt.

22.4.12.3 Example: Converting Three Analog Inputs Using Alternating Sample Mode and a Scan List

It is possible to sample by scanning through the input channels and alternate between MUX A and MUX B. When the Alternating Sample mode is selected, the first input to be sampled will be the input selected for MUX A, the second sample will be the input selected for MUX B. Then the process repeats. When scanning is combined with Alternating Input mode, the positive input to MUX A is selected by the contents of the AD1CSSL register, not CH0SA. For each sample that MUX A is selected the next item in the scan list is sampled. The positive input to MUX B is selected by CH0SB (AD1CHS<27:24>).

When ASAM (AD1CON1<2>) is clear, sampling will not resume after conversion completion, but will occur when setting the SAMP (AD1CON1<1>) bit.

22.5 Initialization

A simple initialization code example for the ADC module is provided in Example 22-4.

In this particular configuration, all 16 analog input pins, AN0-AN15, are set up as analog inputs. Operation in IDLE mode is disabled, output data is in unsigned fractional format, and AV_{DD} and AV_{SS} are used for V_{R+} and V_{R-}. The start of acquisition, as well as start of conversion (conversion trigger), are performed manually in software. The CH0 SHA is used for conversions. Scanning of inputs is disabled, and an interrupt occurs after every acquisition/convert sequence (1 conversion result). The ADC conversion clock is $T_{PB}/2$.

Since acquisition is started manually by setting the SAMP bit (AD1CON1<1>) after each conversion is complete, the auto-sample time bits, SAMC<4:0> (AD1CON3<12:8>), are ignored. Moreover, since the start of conversion (i.e., end of acquisition) is also triggered manually, the SAMP bit needs to be cleared each time a new sample needs to be converted.

EXAMPLE 22-4: ADC INITIALIZATION CODE EXAMPLE

```
AD1PCFG = 0x0000;              /* Configure ADC port
                                  all input pins are analog */

AD1CON1 = 0x2208;              /* Configure sample clock source and Conversion Trigger mode.
                                 Unsigned Fractional format, Manual conversion trigger,
                                 Manual start of sampling, Simultaneous sampling,
                                 No operation in IDLE mode.  */

AD1CON2 = 0x0000;              /* Configure ADC voltage reference
                                  and buffer fill modes.
                                  VREF from AVDD and AVSS,
                                  Inputs are not scanned,
                                  Interrupt every sample */

AD1CON3 = 0x0000;              /* Configure ADC conversion clock */

AD1CHS = 0x0000;               /* Configure input channels,
                                  CH0+ input is AN0.
                                  CHO- input is VREFL (AVss)

AD1CSSL = 0x0000;              /* No inputs are scanned.
                                  Note: Contents of AD1CSSL are ignored when CSCNA = 0 */

IFS1CLR = 2;                   /*Clear ADC conversion interrupt*/

// Configure ADC interrupt priority bits (AD1IP<2:0>) here, if
// required. (default priority level is 4)

IEC1SET = 2;                   /* Enable ADC conversion interrupt*/

AD1CON1SET = 0x8000;           /* Turn on the ADC module */
AD1CON1SET = 0x0002;           /* Start sampling the input */
DelayNmSec(100);               /* Ensure the correct sampling time has elapsed before
                                  starting a conversion.*/

AD1CON1CLR = 0x0002;           /* End Sampling and start Conversion*/
      :                        /* The DONE bit is set by hardware when the convert sequence
                                  is finished. */
      :                        /* The ADIF bit will be set. */
```

22.6 I/O Pin Control

The pins used for analog input can also be used for digital I/O. Configuring a pin for analog input requires three steps. Any digital peripherals that share the desired pin must be disabled. The pin must be configured as a digital input, by setting the corresponding TRIS bit to a '1' to disable the output driver. Then, the pin must be placed in Analog mode by setting the corresponding bit in the AD1PCFG register.

TABLE 22-2: PINS ASSOCIATED WITH THE ADC MODULE

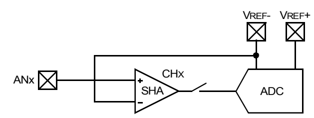
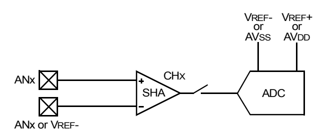
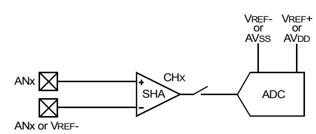
Pin Name	Module Control	Controlling Bit Field	Pin Type	Buffer Type	TRIS	Description
AN0	ON	AD1PCFG<0>	A	—	Input	Analog Input
AN1	ON	AD1PCFG<1>	A	—	Input	Analog Input
AN2	ON	AD1PCFG<2>	A	—	Input	Analog Input
AN3	ON	AD1PCFG<3>	A	—	Input	Analog Input
AN4	ON	AD1PCFG<4>	A	—	Input	Analog Input
AN5	ON	AD1PCFG<5>	A	—	Input	Analog Input
AN6	ON	AD1PCFG<6>	A	—	Input	Analog Input
AN7	ON	AD1PCFG<7>	A	—	Input	Analog Input
AN8	ON	AD1PCFG<8>	A	—	Input	Analog Input
AN9	ON	AD1PCFG<9>	A	—	Input	Analog Input
AN10	ON	AD1PCFG<10>	A	—	Input	Analog Input
AN11	ON	AD1PCFG<11>	A	—	Input	Analog Input
AN12	ON	AD1PCFG<12>	A	—	Input	Analog Input
AN13	ON	AD1PCFG<13>	A	—	Input	Analog Input
AN14	ON	AD1PCFG<14>	A	—	Input	Analog Input
AN15	ON	AD1PCFG<15>	A	—	Input	Analog Input
VREF+	ON	AD1CON2<15:13>	P	—	—	Positive Voltage Reference
VREF-	ON	AD1CON2<15:13>	P	—	—	Negative Voltage Reference

Legend: ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
A = Analog
P = Power

22.6.1 ADC CONVERSION SPEEDS

The PIC32MX 10-bit A/D converter specifications permit a maximum 400 ksps sampling rate. Table 22-3 summarizes the conversion speeds for the PIC32MX 10-bit A/D converter and the required operating conditions..

TABLE 22-3: 10-BIT CONVERSION RATE PARAMETERS

PIC32MX 10-Bit A/D Converter Conversion Rates						
ADC Speed	**TAD Minimum**	**Sampling Time Min**	**RS Max**	**VDD**	**Temperature**	**ADC Channels Configuration**
400 ksps(1)	200 ns	2 TAD	500Ω	4.5V to 5.5V	-40°C to +85°C	
Up to 400 ksps	200 ns	1 TAD	5.0 kΩ	4.5V to 5.5V	-40°C to +125°C	
Up to 300 ksps	256.41 ns	1 TAD	5.0 kΩ	3.0V to 5.5V	-40°C to +125°C	

Note 1: External VREF- and VREF+ pins must be used for correct operation.

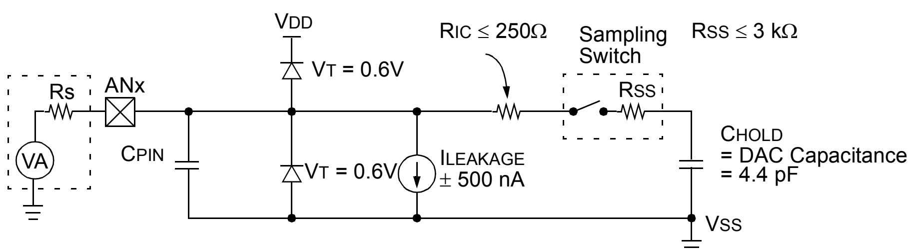
22.6.2 ADC SAMPLING REQUIREMENTS

The analog input model of the 10-bit A/D converter is shown in Figure 22-3. The total acquisition time for the A/D conversion is a function of the internal amplifier settling time and the holding capacitor charge time.

For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the voltage level on the analog input pin. The analog output source impedance (RS), the interconnect impedance (RIC) and the internal sampling switch (RSS) impedance combine to directly affect the time required to charge the CHOLD. The combined impedance of the analog sources must therefore be small enough to fully charge the holding capacitor within the chosen sample time. To minimize the effects of pin leakage currents on the accuracy of the A/D converter, the maximum recommended source impedance, RS, is 5 kΩ for the conversion rates of up to 300 ksps and a maximum of 500Ω for conversion rates of up to 400 ksps). After the analog input channel is selected (changed), this acquisition function must be completed prior to starting the conversion. The internal holding capacitor will be in a discharged state prior to each sample operation.

At least 1 TAD time period should be allowed between conversions for the acquisition time. For more details, see the device electrical specifications.

FIGURE 22-3: 10-BIT A/D CONVERTER ANALOG INPUT MODEL



Note: CPIN value depends on device package and is not tested. Effect of CPIN negligible if Rs ≤ 5 kΩ.

Legend:

CPIN = input capacitance
VT = threshold voltage
RSS = sampling switch resistance
RIC = interconnect resistance
RS = source resistance
CHOLD = sample/hold capacitance (from DAC)
ILEAKAGE = leakage current at the pin due to various junctions

 Advance Information

23.0 COMPARATOR

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

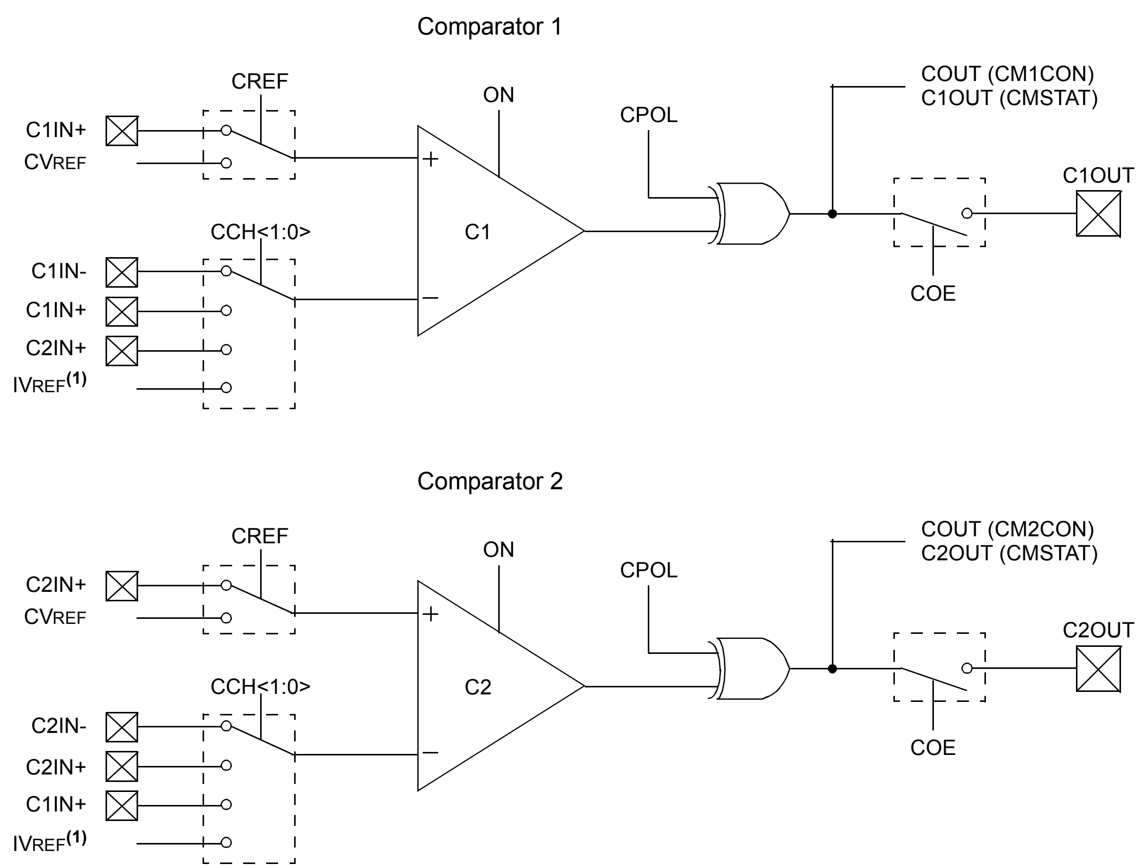
The PIC32MX Family Analog Comparator module contains one or more comparator(s) that can be configured in a variety of ways.

Following are some of the key features of this module:

- Selectable inputs available include:
  - Analog inputs multiplexed with I/O pins
  - On-chip internal absolute voltage reference (IVREF)
  - Comparator voltage reference (CVREF)
- Outputs can be inverted
- Selectable interrupt generation

A block diagram of the comparator module is shown in Figure 23-1.

FIGURE 23-1: COMPARATOR BLOCK DIAGRAM



Note 1: IVref is the internal 1.2V reference.

23.1 Comparator Control Registers

Note:	Each PIC32MX device variant may have one or more Comparator modules. An 'x' used in the names of pins, control/status bits and registers denotes the particular module. Refer to the specific device data sheets for more details.

A Comparator module consists of the following Special Function Registers (SFRs):

- CMxCON: Comparator Control Register
- CMxCONCLR, CMxCONSET, CMxCONINV: Atomic Bit Manipulation Registers for CMxCON
- CMSTAT: Comparator Status Registers
- CMSTATCLR, CMSTATSET, CMSTATINV: Atomic Bit Manipulation Registers for CMSTAT

The comparator module also has the following interrupt control registers:

- IFS1: Interrupt Flag Status Register
- IEC: Interrupt Enable Control Register
- IPC7: Interrupt Priority Control Register

Table 23-1 provides brief summaries of all comparator related registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

TABLE 23-1: COMPARATOR SFRS SUMMARY

Virtual Address	Name		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
BF80_A000	CM1CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	COE	CPOL	—	—	—	—	COUT
		7:0	EVPOL<1:0>		—	CREF	—	—	CCH<1:0>	
BF80_A004	CM1CONCLR	31:0	Write clears selected bits in CM1CON, read yields undefined value							
BF80_A008	CM1CONSET	31:0	Write sets selected bits in CM1CON, read yields undefined value							
BF80_A00C	CM1CONINV	31:0	Write inverts selected bits in CM1CON, read yields undefined value							
BF80_A010	CM2CON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	COE	CPOL	—	—	—	—	COUT
		7:0	EVPOL<1:0>		—	CREF	—	—	CCH<1:0>	
BF80_A004	CM2CONCLR	31:0	Write clears selected bits in CM2CON, read yields undefined value							
BF90_A008	CM2CONSET	31:0	Write sets selected bits in CM2CON, read yields undefined value							
BF80_A00C	CM2CONINV	31:0	Write inverts selected bits in CM2CON, read yields undefined value							
BF80_A060	CMSTAT	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	FRZ	SIDL	—	—	—	—	—
		7:0	—	—	—	—	—	—	C2OUT	C1OUT
BF80_A064	CMSTATCLR	31:0	Write clears selected bits in CMSTAT, read yields undefined value							
BF80_A068	CMSTATSET	31:0	Write sets selected bits in CMSTAT, read yields undefined value							
BF80_A06C	CMSTATINV	31:0	Write inverts selected bits in CMSTAT, read yields undefined value							
BF88_1010	IFS1	7:0	SPI2RXIF	SPI2TXIF	SPI2EIF	CMP2IF	CMP1IF	PMPIF	AD1IF	CNIF
BF88_1040	IEC1	7:0	SPI2RXIE	SPI2TXIE	SPI2EIE	CMP2IE	CMP1IE	PMPIE	AD1IE	CNIE
BF88_10D0	IPC7	23:16	—	—	—	CMP2IP<2:0>			CMP2IS<1:0>	
	IPC7	15:8	—	—	—	CMP1IP<2:0>			CMP1IS<1:0>	

REGISTER 23-1: CM1CON: COMPARATOR 1 CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	r-0	U-0	U-0	U-0	R-0
ON	COE	CPOL	—	—	—	—	COUT
bit 15							bit 8

R/W-1	R/W-1	U-0	R/W-0	U-0	U-0	R/W-1	R/W-1
EVPOL<1:0>		—	CREF	—	—	CCH<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** Comparator ON bit
1 = Module is enabled. Setting this bit does not affect the other bits in this register.
0 = Module is disabled and does not consume current. Clearing this bit does not affect the other bits in this register.

bit 14 **COE:** Comparator Output Enable bit
1 = Comparator output is driven on the output C1OUT pin
0 = Comparator output is not driven on the output C1OUT pin

bit 13 **CPOL:** Comparator Output Inversion bit
1 = Output is inverted
0 = Output is not inverted
Note: Setting this bit will invert the signal to the to the comparator interrupt generator as well. This will result in an interrupt being generated on the opposite edge from the one selected by EVPOL<1:0>.

bit 12 **Reserved:** Maintain as ‘0’

bit 11-9 **Unimplemented:** Read as ‘0’

bit 8 **COUT:** Comparator Output bit
1 = Output of the comparator is a ‘1’
0 = Output of the comparator is a ‘0’

bit 7-6 **EVPOL<1:0>:** Interrupt Event Polarity Select bits
11 = Comparator interrupt is generated on a low-to-high or high-to-low transition of the comparator output
10 = Comparator interrupt is generated on a high-to-low transition of the comparator output
01 = Comparator interrupt is generated on a low-to-high transition of the comparator output
00 = Comparator interrupt generation is disabled

bit 5 **Unimplemented:** Read as ‘0’

REGISTER 23-1: CM1CON: COMPARATOR 1 CONTROL REGISTER (CONTINUED)

bit 4	**CREF:** Comparator 1 Positive Input Configure bit 1 = Comparator non-inverting input is connected to the internal CVREF 0 = Comparator non-inverting input is connected to the C1IN+ pin
bit 3-2	**Unimplemented:** Read as ‘0’
bit 1-0	**CCH<1:0>:** Comparator Negative Input Select bits for Comparator 1 11 = Comparator inverting input is connected to the IVREF 10 = Comparator inverting input is connected to the C2IN+ pin 01 = Comparator inverting input is connected to the C1IN+ pin 00 = Comparator inverting input is connected to the C1IN- pin

REGISTER 23-2: CM2CON: COMPARATOR 2 CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	r-0	U-0	U-0	U-0	R-0
ON	COE	CPOL	—	—	—	—	COUT
bit 15							bit 8

R/W-1	R/W-1	U-0	R/W-0	U-0	U-0	R/W-1	R/W-1
EVPOL<1:0>		—	CREF	—	—	CCH<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: ('0', '1', x = Unknown)		

bit 31-16 **Unimplemented:** Read as '0'

bit 15 **ON:** Comparator ON bit
1 = Module is enabled. Setting this bit does not affect the other bits in this register.
0 = Module is disabled and does not consume current. Clearing this bit does not affect the other bits in this register.

bit 14 **COE:** Comparator Output Enable bit
1 = Comparator output is driven on the output C2OUT pin
0 = Comparator output is not driven on the output C2OUT pin

bit 13 **CPOL:** Comparator Output Inversion bit
1 = Output is inverted
0 = Output is not inverted
Note: Setting this bit will invert the signal to the to the comparator interrupt generator as well. This will result in an interrupt being generated on the opposite edge from the one selected by EVPOL<1:0>.

bit 12 **Reserved:** Maintain as '0'

bit 11-9 **Unimplemented:** Read as '0'

bit 8 **COUT:** Comparator Output bit
1 = Output of the comparator is a '1'
0 = Output of the comparator is a '0'

bit 7-6 **EVPOL<1:0>:** Interrupt Event Polarity Select bits
11 = Comparator interrupt is generated on a low-to-high or high-to-low transition of the comparator output
10 = Comparator interrupt is generated on a high-to-low transition of the comparator output
01 = Comparator interrupt is generated on a low-to-high transition of the comparator output
00 = Comparator interrupt generation is disabled

bit 5 **Unimplemented:** Read as '0'

REGISTER 23-2: CM2CON: COMPARATOR 2 CONTROL REGISTER (CONTINUED)

bit 4	**CREF:** Comparator 1 Positive Input Configure bit 1 = Comparator non-inverting input is connected to the internal CVREF 0 = Comparator non-inverting input is connected to the C2IN+ pin
bit 3-2	**Unimplemented:** Read as ‘0’
bit 1-0	**CCH<1:0>:** Comparator Negative Input Select bits for Comparator 2 11 = Comparator inverting input is connected to the IVREF 10 = Comparator inverting input is connected to the C1IN+ pin 01 = Comparator inverting input is connected to the C2IN+ pin 00 = Comparator inverting input is connected to the C2IN- pin

REGISTER 23-3: CMSTAT: COMPARATOR CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
—	FRZ	SIDL	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	R-0	R-0
—	—	—	—	—	—	C2OUT	C1OUT
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-15 **Unimplemented:** Read as ‘0’

bit 14 **FRZ:** Freeze Control bit
1 = Freeze operation when CPU enters Debug Exception mode
0 = Continue operation when CPU enters Debug Exception mode
Note: FRZ is writable in Debug Exception mode only. It always reads ‘0’ in normal mode.

bit 13 **SIDL:** Stop in Idle Control bit
1 = All comparator modules are disabled in IDLE mode
0 = All comparator modules continue to operate in IDLE mode.

bit 12-2 **Unimplemented:** Read as ‘0’

bit 1 **C2OUT:** Comparator Output bit
1 = Output of comparator 2 is a ‘1’
0 = Output of comparator 2 is a ‘0’

bit 0 **C1OUT:** Comparator Output bit
1 = Output of comparator 1 is a ‘1’
0 = Output of comparator 1 is a ‘0’

REGISTER 23-4: IPC7- INTERRUPT PRIORITY CONTROL REGISTER 7

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	SPI2IP<2:0>			SPI2IS<1:0>	
bit 31							bit 24

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CMP2IP<2:0>			CMP2IS<1:0>	
bit 23							bit 16

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CMP1IP<2:0>			CMP1IS<1:0>	
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	PMPIP<2:0>			PMPIS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 20-18 **CMP2IP<2:0>:** Comparator 2 Interrupt Priority bits

111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 17-6 **CMP2IS<1:0>:** Comparator 2 Interrupt Sub Priority bits

11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

bit 12-10 **CMP1IP<2:0>:** Comparator 1 Interrupt Priority bits

111 = Interrupt priority is 7
110 = Interrupt priority is 6
101 = Interrupt priority is 5
100 = Interrupt priority is 4
011 = Interrupt priority is 3
010 = Interrupt priority is 2
001 = Interrupt priority is 1
000 = Interrupt is disabled

bit 9-8 **CMP1IS<1:0>:** Comparator 1 Interrupt Sub Priority bits

11 = Interrupt subpriority is 3
10 = Interrupt subpriority is 2
01 = Interrupt subpriority is 1
00 = Interrupt subpriority is 0

23.2 Comparator Operation

23.2.1 COMPARATOR CONFIGURATION

The Comparator module has a flexible input and output configuration to allow the module to be tailored to the needs of the application. The PIC32MX Family comparator module has individual control over the enables, output inversion, output on I/O pin and input selections. The VIN+ pin of each comparator can select from an input pin or the CVREF. The VIN- input of the comparator can select from one of 3 input pins or the IVREF. In addition, the module has two individual comparator event generation control bits. These control bits can be used for detecting when the output of an individual comparator changes to a desired state or changes states.

If the Comparator mode is changed, the comparator output level may not be valid for the specified mode change delay (refer to the device data sheet for more information).

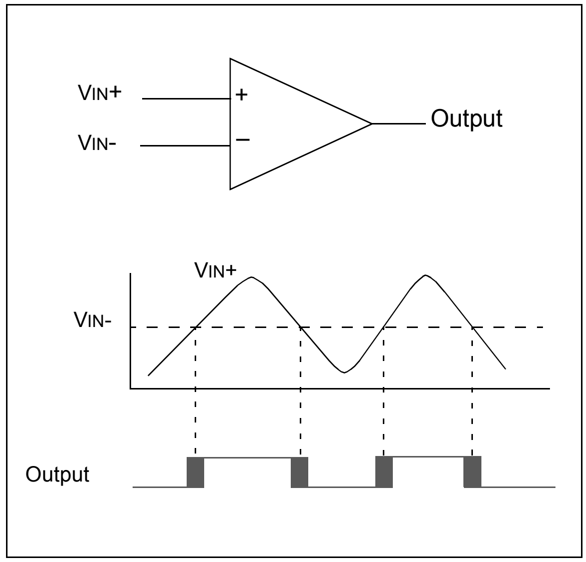
Note:	Comparator interrupts should be disabled during a Comparator mode change; otherwise, a false interrupt may be generated.

A single comparator is shown in the upper portion of Figure 23-2. The lower portion represents the relationship between the analog input levels and the digital output. When the analog input at VIN+ is less than the analog input at VIN-, the output of the comparator is a digital low level. When the analog input at VIN+ is greater than the analog input VIN-, the output of the comparator is a digital high level. The shaded areas of the output of the comparator in the lower portion of Figure 23-2 demonstrate the uncertainty that is due to input offsets and the response time of the comparator.

23.3 Comparator Inputs

Depending on the Comparator Operating mode, the inputs to the comparators may be from two input pins or a combination of an input pin and one of two internal voltage references. The analog signal present at VIN- is compared to the signal at VIN+ and the digital output of the comparator is set or cleared according to the result of the comparison (see Figure 23-2).

FIGURE 23-2: SINGLE COMPARATOR



23.3.0.1 External Reference Signal

An external voltage reference may be used with the comparator by using the output of the reference as an input to the comparator. Refer to the device data sheet for input voltage limits.

23.3.0.2 Internal Reference Signals

The CVREF module and the IVREF can be used as inputs to the comparator (see Figure 23-1). The CVREF provides a user-selectable voltage for use as a comparator reference. Refer to **24.0 "Comparator Reference"** of this manual for more information on this module. The IVREF has a fixed, 1.2V output that does not change with the device supply voltage. Refer to the device data sheet for specific details and accuracy of this reference.

23.4 Comparator Outputs

The comparator output is read through the CMSTAT register and the COUT bit (CM2CON<8> or CM1CON<8>). This bit is read-only. The comparator output may also be directed to an I/O pin via the CxOUT bit; however, the COUT bit is still valid when the signal is routed to a pin. For the comparator output to be available on the CxOut pin, the associated TRIS bit for the output pin must be configured as an output. When the COUT signal is routed to a pin the signal is the unsynchronized output of the comparator.

The output of the comparator has a degree of uncertainty. The uncertainty of each of the comparators is related to the input offset voltage and the response time, as stated in the specifications. The lower portion of Figure 23-2 provides a graphical representation of this uncertainty.

The comparator output bit, COUT, provides the latched sampled value of the comparator's output- when the register was read. There are two common methods used to detect a change in the comparator output:

- Software polling
- Interrupt generation

23.4.1 CHANGING THE POLARITY OF COMPARATOR OUTPUTS

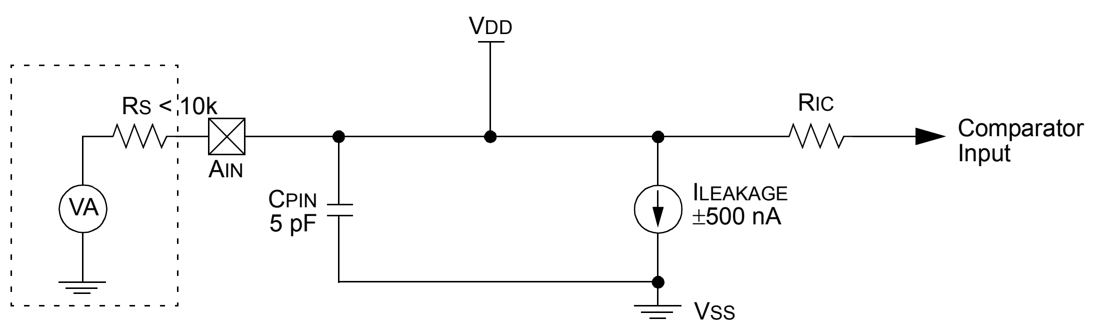
The polarity of the comparator outputs can be changed using the CPOL bit (CMxCON<13>). CPOL appears below the comparator Cx on the left side of Figure 23-1.

23.5 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 23-3. A maximum source impedance of 10 kΩ is recommended for the analog sources. Any external component connected to an analog input pin, such as a capacitor or a zener diode, should have very little leakage current. See the device data sheet for input voltage limits. If a pin is to be shared by two or more analog inputs that are to be used simultaneously, the loading effects of all the modules involved must be taken into consideration. This loading may reduce the accuracy of one or more of the modules connected to the common pin. This may also require a lower source impedance than is stated for a single module with exclusive use of a pin in Analog mode.

Notes:	When reading the PORT register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert an analog input according to the Schmitt Trigger input specification. Analog levels on any pin defined as a digital input may cause the input buffer to consume more current than is specified.

FIGURE 23-3: COMPARATOR ANALOG INPUT MODEL



Legend:
C_{PIN} = Input Capacitance
$I_{LEAKAGE}$ = Leakage Current at the pin due to various junctions
R_{IC} = Interconnect Resistance
R_S = Source Impedance
VA = Analog Voltage

23.6 Interrupts

EXAMPLE 23-1: COMPARATOR INITIALIZATION WITH INTERRUPTS ENABLED CODE EXAMPLE

```
                          // Configure both comparators to generate an interrupt on any
                          // output transition
CM1CON  = 0xC0D0;         // Initialize Comparator 1
                          // Comparator enabled, output enabled, interrupt on any output
                          // change, inputs: CVref, C1IN-
CM2CON  = 0xA0C2;         // Initialize Comparator 2
                          // Comparator enabled, output enabled, interrupt on any output
                          // change, inputs: C2IN+, C1IN+

                          // Enable interrupts for Comparator modules and set priorities
                          // Set priority to 7 & sub priority to 3
IPC7SET = 0x00000700;     // Set CMP1 interrupt sub priority
IFS1CLR = 0x00000008;     // Clear the CMP1 interrupt flag
IEC1SET = 0x00000008;     // Enable CMP1 interrupt

IPC7SET = 0x00070000;     // Set CMP2 interrupt sub priority
IFS1CLR = 0x000000010;    // Clear the CMP2 interrupt flag
IEC1SET = 0x000000010;    // Enable CMP2 interrupt
```

EXAMPLE 23-2: COMPARATOR ISR CODE EXAMPLE

```
                          // Insert user code here

#pragma interrupt CmpIntHandler ipl4 vector 29
void CmpIntHandler(void)
{
                          // Insert user code here
    IFS1CLR = 0x00000010; // Clear the CMP2 interrupt flag
}

#pragma interrupt CmpIntHandler ipl4 vector 30
void CmpIntHandler(void)
{
                          // Insert code user here
    IFS1CLR = 0x00000008; // Clear the CMP1 interrupt flag
}
```

23.7 I/O Pin Control

TABLE 23-2: PINS ASSOCIATED WITH A COMPARATOR

Pin Name	Module Control	Controlling Bit Field	Required TRIS Bit Setting	Pin Type	Buffer Type	Description
C1IN+	ON	CVREF[1], CCH<1:0>[1], CCH<1:0>[2], AD1PCFG	Input	A, I	—	Analog Input for C1IN+
C1IN-	ON	CCH<1:0>[1], AD1PCFG	Input	A, I	—	Analog Input for C1IN-
C2IN+	ON	CVREF[2], CCH<1:0>[1], CCH<1:0>[2], AD1PCFG	Input	A, I	—	Analog Input for C2IN+
C2IN-	ON	CCH<1:0>[2], AD1PCFG	Input	A, I	—	Analog Input for C2IN-
C1OUT	ON	COE[1]	Output	D, O	—	Digital Output of the C1
C2OUT	ON	COE[2]	Output	D, O	—	Digital Output of the C2

Legend: ST = Schmitt Trigger input with CMOS levels, I = Input, O = Output, A = Analog, D = Digital

Note 1: In CM1CON register.

2: In CM2CON register.

24.0 COMPARATOR REFERENCE

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *“PIC32MX Family Reference Manual”* (DS61132) for a detailed description of this peripheral.

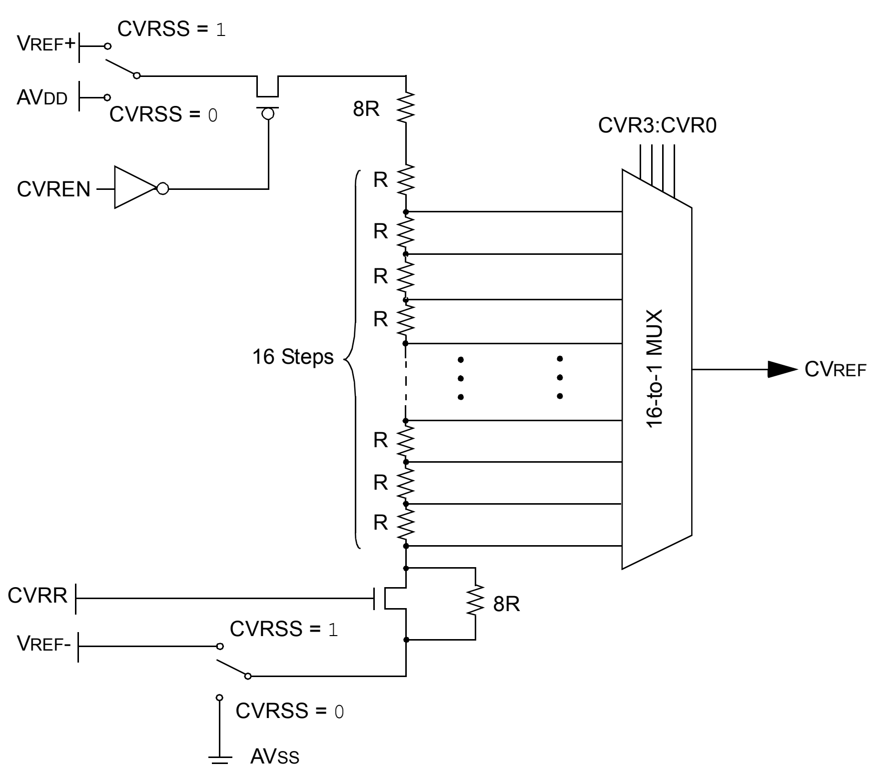
The Comparator Voltage Reference (CVREF) is a 16-tap, resistor ladder network that provides a selectable reference voltage. Although its primary purpose is to provide a reference for the analog comparators, it also may be used independently of them.

A block diagram of the module is shown in Figure 24-1. The resistor ladder is segmented to provide two ranges of voltage reference values and has a power-down function to conserve power when the reference is not being used. The module’s supply reference can be provided from either device VDD/VSS or an external voltage reference. The CVREF output is available for the comparators and typically available for pin output. Please see the specific device data sheet for information.

The comparator voltage reference has the following features:

- High and low range selection
- Sixteen output levels available for each range
- Internally connected to comparators to conserve device pins
- Output can be connected to a pin

FIGURE 24-1: COMPARATOR VOLTAGE REFERENCE BLOCK DIAGRAM



24.1 Comparator Voltage Reference Control Registers

The CVREF module consists of the following Special Function Registers (SFRs):

- CVRCON: Control Register for the Module
- CVRCONCLR, CVRCONSET, CVRCONINV: atomic Bit Manipulation Registers for CVRCON

Table 24-1 provides a brief summary of all CVREF module related registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

TABLE 24-1: COMPARATOR VOLTAGE REFERENCE SFR SUMMARY

Virtual Address	Name		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
BF80_9800	CVRCON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	—	—	—	—	—		
		7:0	—	CVROE	CVRR	CVRSS	CVR<3:0>			
BF80_9804	CVRCONCLR	31:0	Write clears selected bits in CVRCON, read yields undefined value							
BF80_9808	CVRCONSET	31:0	Write sets selected bits in CVRCON, read yields undefined value							
BF80_980C	CVRCONINV	31:0	Write inverts selected bits in CVRCON, read yields undefined value							

REGISTER 24-1: CVRCON: COMPARATOR VOLTAGE REFERENCE CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
ON	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	CVROE	CVRR	CVRSS	CVR<3:0>			
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** CVREF Peripheral On bit
1 = Module is enabled; setting this bit does not affect the other bits in the register
0 = Module is disabled and does not consume current; clearing this bit does not affect the other bits in the register

bit 14-7 **Unimplemented:** Read as ‘0’

bit 6 **CVROE:** CVREF Output Enable bit
1 = Voltage level is output on CVREF pin
0 = Voltage level is disconnected from CVREF pin
Note: CVROE overrides the TRIS bit setting; see **Section 12.0 “I/O Ports”** for more information.

bit 5 **CVRR:** CVREF Range Selection bit
1 = 0 to 0.67 CVRSRC, with CVRSRC/24 step size
0 = 0.25 CVRSRC to 0.75 CVRSRC, with CVRSRC/32 step size

bit 4 **CVRSS:** CVREF Source Selection bit
1 = Comparator voltage reference source, CVRSRC = (VREF+) – (VREF-)
0 = Comparator voltage reference source, CVRSRC = AVDD – AVSS

bit 3-0 **CVR<3:0>:** CVREF Value Selection 0 ≤ CVR3:CVR0 ≤ 15 bits
When CVRR = 1:
CVREF = (CVR<3:0>/24) • (CVRSRC)
When CVRR = 0:
CVREF = 1/4 • (CVRSRC) + (CVR<3:0>/32) • (CVRSRC)

24.2 Operation

The CVREF module is controlled through the CVRCON register (Register 24-1). The CVREF provides two ranges of output voltage, each with 16 distinct levels. The range to be used is selected by the CVRR bit (CVRCON<5>). The primary difference between the ranges is the size of the steps selected by the CVREF Value Selection bits, CVR3:CVR0, with one range offering finer resolution and the other offering a wider range of output voltage. The typical output voltages are listed in Table 24-2.

The equations used to calculate the CVREF output are as follows:

If CVRR = 1:
Voltage Reference = ((CVR3:CVR0)/24) x (CVRSRC)

If CVRR = 0:
Voltage Reference = (CVRSRC/4) + ((CVR3:CVR0)/32) x (CVRSRC)

The CVREF Source Voltage (CVRSRC) can come from either VDD and VSS, or the external VREF+ and VREF- pins that are multiplexed with I/O pins. The voltage source is selected by the CVRSS bit (CVRCON<4>). The voltage reference is output to the CVREF pin by setting the CVROE (CVRCON<6>) bit; this will override the corresponding TRIS bit setting.

The settling time of the CVREF must be considered when changing the CVREF output (refer to the data sheet for your device).

TABLE 24-2: TYPICAL VOLTAGE REFERENCE WITH CVRSRC = 3.3

CVR<3:0>	Voltage Reference	
	CVRR = 0 (CVRCON <5>)	CVRR = 1 (CVRCON <5>
0	0.83V	0.00V
1	0.93V	0.14V
2	1.03V	0.28V
3	1.13V	0.41V
4	1.24V	0.55V
5	1.34V	0.69V
6	1.44V	0.83V
7	1.55V	0.96V
8	1.65V	1.10V
9	1.75V	1.24V
10	1.86V	1.38V
11	1.96V	1.51V
12	2.06V	1.65V
13	2.17V	1.79V
14	2.27V	1.93V
15	2.37V	2.06V

24.2.1 CVREF OUTPUT CONSIDERATIONS

The full range of voltage reference cannot be realized due to the construction of the module. The transistors on the top and bottom of the resistor ladder network (Figure 24-1) keep the voltage reference from approaching the reference source rails. The voltage reference is derived from the reference source; therefore, the voltage reference output changes with fluctuations in that source. Refer to the product data sheet for the electrical specifications. Table 24-3 contains the typical output impedances for the CVREF module.

TABLE 24-3: TYPICAL CVREF OUTPUT IMPEDANCE IN OHMS

CVR<3:0>	Voltage Reference	
	CVRR = 0 (CVRCON <5>)	CVRR = 1 (CVRCON <5>
0	12k	500
1	13k	1.9k
2	13.8k	3.7k
3	14.4k	5.3k
4	15k	6.7k
5	15.4k	7.9k
6	15.8k	9k
7	15.9k	9.9k
8	16k	10.7k
9	15.9k	11.3k
10	15.8k	11.7k
11	15.4k	11.9k
12	15k	12k
13	14.4k	11.9k
14	13.8k	11.7k
15	12.9k	11.3k

24.2.2 INITIALIZATION

This initialization sequence, shown in Example 24-1, configures the CVREF module for: module enabled, output enabled, high range, and set output for maximum (2.37V).

EXAMPLE 24-1: VOLTAGE REFERENCE CONFIGURATION

```
CVRCON = 0x804F;   //Initialize Voltage Reference Module
                   //enable module, enable output, set
                   // range to high, set output to maximum
```

24.3 Interrupts

There are no Interrupt configuration registers or bits for the CVREF module. The CVREF module does not generate interrupts.

24.4 I/O Pin Control

The CVREF module has the ability to output to a pin. When the CVREF module is enabled and CVROE (CVRCON<6>) is '1', the output driver for the CVREF pin is disabled and the CVREF voltage is available at the pin. For proper operation, the TRIS bit corresponding to the CVREF pin must be a '1' when CVREF is to be output to a pin. This disables the Digital Input mode for the pin and prevents undesired current draw resulting from applying an analog voltage to a digital input pin. The output buffer has very limited drive capability. An external buffer amplifier is recommended for any application that uses the CVREF voltage externally. An output capacitor may be used to reduce output noise. Use of an output capacitor will increase settling time.

TABLE 24-4: PINS ASSOCIATED WITH A COMPARATOR

Pin Name	Module Control	Controlling Bit Field	Required TRIS Bit Setting	Pin Type	Buffer Type	Description
CVREF	ON	CVROE	Input	A, O	—	CVREF Output

Legend: ST = Schmitt Trigger input with CMOS levels, I = Input, O = Output, A = Analog, D = Digital

25.0 SPECIAL FEATURES

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

PIC32MX Family devices include several features intended to maximize application flexibility and reliability, and minimize cost through elimination of external components. These are:

- Flexible Device Configuration
- Code Protection
- Internal Voltage Regulator

TABLE 25-1: DEVCFG: DEVICE CONFIGURATION WORD SUMMARY

Virtual Address	Name		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
BFC0_2FF0	DEVCFG3	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	USERID15	USERID14	USERID13	USERID12	USERID11	USERID10	USERID9	USERID8
		7:0	USERID7	USERID6	USERID5	USERID4	USERID3	USERID2	USERID1	USERID0
BFC0_2FF4	DEVCFG2	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	FPLLODIV<2:0>		
		15:8	—	—	—	—	—	—	—	—
		7:0	—	FPLLMULT<2:0>			—	FPLLIDIV<2:0>		
BFC0_2FF8	DEVCFG1	31:24	—	—	—	—	—	—	—	—
		23:16	FWDTEN	—	—	WDTPS<4:0>				
		15:8	FCKSM<1:0>		FPBDIV<1:0>		—	OSCIOFNC	POSCMD<1:0>	
		7:0	IESO	—	FSOSCEN	—	—	FNOSC<2:0>		
BFC0_2FFC	DEVCFG0	31:24	—	—	—	CP	—	—	—	BWP
		23:16	—	—	—	—	PWP19	PWP18	PWP17	PWP16
		15:8	PWP15	PWP14	PWP13	PWP12	—	—	—	—
		7:0	—	—	—	—	ICESEL	—	DEBUG<1:0>	

TABLE 25-2: DEVID SUMMARY

Virtual Address	Name		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
BF80_F220	DEVID	31:24	VER3	VER2	VER1	VER0	—	—	—	—
		23:16	—	—	—	—	DEV7	DEV6	DEV5	DEV4
		15:8	DEV3	DEV2	DEV1	DEV0	MANID11	MANID10	MANID9	MANID8
		7:0	MANID7	MANID6	MANID5	MANID4	MANID3	MANID2	MANID1	1

REGISTER 25-1: DEVCFG0: DEVICE CONFIGURATION WORD 0

r-0	r-1	r-1	R/P-1	r-1	r-1	r-1	R/P-1
—	—	—	CP	—	—	—	BWP
bit 31							bit 24

r-1	r-1	r-1	r-1	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	—	PWP19	PWP18	PWP17	PWP16
bit 23							bit 16

R/P-1	R/P-1	R/P-1	R/P-1	r-1	r-1	r-1	r-1
PWP15	PWP14	PWP13	PWP12	—	—	—	—
bit 15							bit 8

r-1	r-1	r-1	r-1	R/P-1	r-1	R/P-1	R/P-1
—	—	—	—	ICESEL	—	DEBUG1	DEBUG0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31 **Reserved:** Maintain as ‘0’.

bit 30-29 **Reserved:** Maintain as ‘1’

bit 28 **CP:** Code-Protect bit

Prevents boot and program Flash memory from being read or modified by an external programming device.

1 = Protection disabled
0 = Protection enabled

bit 27-25 **Reserved:** Maintain as ‘1’

bit 24 **BWP:** Boot Flash Write-Protect bit

Prevents boot Flash memory from being modified during code execution.

1 = Boot Flash is writable
0 = Boot Flash is not writable

bit 23-20 **Reserved:** Maintain as ‘1’

(Continued)

bit 19-12 **PWP<19:12>:** Program Flash Write-Protect bits

Prevents selected program Flash memory pages from being modified during code execution.
The PWP bits represent the one's compliment of the number of write protected program Flash memory pages.

11111111 = Disabled
11111110 = 0xBD00_0FFF
11111101 = 0xBD00_1FFF
11111100 = 0xBD00_2FFF
11111011 = 0xBD00_3FFF
11111010 = 0xBD00_4FFF
11111001 = 0xBD00_5FFF
11111000 = 0xBD00_6FFF
11110111 = 0xBD00_7FFF
11110110 = 0xBD00_8FFF
11110101 = 0xBD00_9FFF
11110100 = 0xBD00_AFFF
11110011 = 0xBD00_BFFF
11110010 = 0xBD00_CFFF
11110001 = 0xBD00_DFFF
11110000 = 0xBD00_EFFF
11101111 = 0xBD00_FFFF
...
01111111 = 0xBD07_FFFF

bit 11-4 **Reserved:** Maintain as '1'

bit 3 **ICESEL:** ICE/ICD Communication Channel Select bit

1 = ICE uses PGC2/PGD2 pins
0 = ICE uses PGC1/PGD1 pins

bit 2 **Reserved:** Maintain as '1'

bit 1-0 **DEBUG<1:0>:** Background Debugger Enable bits

11 = Debugger disabled (forced if device is code-protected)
10 = ICE debugger enabled
01 = Reserved
00 = Reserved

REGISTER 25-2: DEVCFG1: DEVICE CONFIGURATION WORD 1

r-1	r-1	r-1	r-1	r-1	r-1	r-1	r-1
—	—	—	—	—	—	—	—
bit 31							bit 24

R/P-1	r-1	r-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
FWDTEN	—	—	WDTPS<4:0>				
bit 23							bit 16

R/P-1	R/P-1	R/P-1	R/P-1	r-1	R/P-1	R/P-1	R/P-1
FCKSM<1:0>		FPBDIV<1:0>		—	OSCIOFNC	POSCMD<1:0>	
bit 15							bit 8

R/P-1	r-1	R/P-1	r-1	r-1	R/P-1	R/P-1	R/P-1
IESO	—	FSOSCEN	—	—	FNOSC<2:0>		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-24 **Reserved:** Maintain as ‘1’

bit 23 **FWDTEN:** Watchdog Timer Enable bit

1 = The WDT is enabled and cannot be disabled by software
0 = The WDT is not enabled; it can be enabled in software

bit 22-21 **Reserved:** Maintain as ‘1’

bit 20-16 **WDTPS<4:0>:** Watchdog Timer Postscale Select bits

10100 = 1:1048576
10011 = 1:524288
10010 = 1:262144
10001 = 1:131072
10000 = 1:65536
01111 = 1:32768
01110 = 1:16384
01101 = 1:8192
01100 = 1:4096
01011 = 1:2048
01010 = 1:1024
01001 = 1:512
01000 = 1:256
00111 = 1:128
00110 = 1:64
00101 = 1:32
00100 = 1:16
00011 = 1:8
00010 = 1:4
00001 = 1:2
00000 = 1:1
All other combinations not shown result in operation = ‘10100’

bit 15-14 **FCKSM<1:0>:** Clock Switching and Monitor Selection Configuration bits

1x = Clock switching is disabled, Fail-Safe Clock Monitor is disabled
01 = Clock switching is enabled, Fail-Safe Clock Monitor is disabled
00 = Clock switching is enabled, Fail-Safe Clock Monitor is enabled

(Continued)

bit 13-12	**FPBDIV<1:0>:** Peripheral Bus Clock Divisor Default Value bits 11 = PBCLK is SYSCLK divided by 8 10 = PBCLK is SYSCLK divided by 4 01 = PBCLK is SYSCLK divided by 2 00 = PBCLK is SYSCLK divided by 1
bit 11	**Reserved:** Maintain as '1'
bit 10	**OSCIOFNC:** CLKO Enable Configuration bit 1 = CLKO output signal active on the OSCO pin; primary oscillator must be disabled or configured for the External Clock mode (EC) for the CLKO to be active (POSCMD<1:0> = 11 OR 00) 0 = CLKO output disabled
bit 9-8	**POSCMD<1:0>:** Primary Oscillator Configuration bits 11 = Primary oscillator disabled 10 = HS oscillator mode selected 01 = XT oscillator mode selected 00 = External clock mode selected
bit 7	**IESO:** Internal External Switchover bit 1 = Internal External Switchover mode enabled (Two-Speed Start-up enabled) 0 = Internal External Switchover mode disabled (Two-Speed Start-up disabled)
bit 6	**Reserved:** Maintain as '1'
bit 5	**FSOSCEN:** Secondary Oscillator Enable bit 1 = Enable Secondary Oscillator 0 = Disable Secondary Oscillator
bit 4-3	**Reserved:** Maintain as '1'
bit 2-0	**FNOSC<2:0>:** Oscillator Selection bits 000 = Fast RC Oscillator (FRC) 001 = Fast RC Oscillator with divide-by-N with PLL module (FRCDIV+PLL) 010 = Primary Oscillator (XT, HS, EC)(1) 011 = Primary Oscillator with PLL module (XT+PLL, HS+PLL, EC+PLL) 100 = Secondary Oscillator (SOSC) 101 = Low-Power RC Oscillator (LPRC) 110 = FRCDIV16 Fast RC Oscillator with fixed divide-by-16 postscaler 111 = Fast RC Oscillator with divide-by-N (FRCDIV)

Note 1: Do not disable POSC (POSCMD = 00) when using this oscillator source.

REGISTER 25-3: DEVCFG2: DEVICE CONFIGURATION WORD 2

r-1	r-1	r-1	r-1	r-1	r-1	r-1	r-1
—	—	—	—	—	—	—	—
bit 31							bit 24

r-1	r-1	r-1	r-1	r-1	R/P-1	R/P-1	R/P-1
—	—	—	—	—	FPLLODIV<2:0>		
bit 23							bit 16

r-1	r-1	r-1	r-1	r-1	r-1	r-1	r-1
—	—	—	—	—	—	—	—
bit 15							bit 8

r-1	R/P-1	R/P-1	R/P-1	r-1	R/P-1	R/P-1	R/P-1
—	FPLLMULT<2:0>			—	FPLLIDIV<2:0>		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-19 **Reserved:** Maintain as ‘1’

bit 18-16 **FPLLODIV<2:0>:** Default Postscaler for PLL bits
- 111 = PLL output divided by 256
- 110 = PLL output divided by 64
- 101 = PLL output divided by 32
- 100 = PLL output divided by 16
- 011 = PLL output divided by 8
- 010 = PLL output divided by 4
- 001 = PLL output divided by 2
- 000 = PLL output divided by 1

bit 15-7 **Reserved:** Maintain as ‘1’

bit 6-4 **FPLLMULT<2:0>:** PLL Multiplier bits
- 111 = 24x multiplier
- 110 = 21x multiplier
- 101 = 20x multiplier
- 100 = 19x multiplier
- 011 = 18x multiplier
- 010 = 17x multiplier
- 001 = 16x multiplier
- 000 = 15x multiplier

bit 3 **Reserved:** Maintain as ‘1’

bit 2-0 **FPLLIDIV<2:0>:** PLL Input Divider bits
- 111 = 12x divider
- 110 = 10x divider
- 101 = 6x divider
- 100 = 5x divider
- 011 = 4x divider
- 010 = 3x divider
- 001 = 2x divider
- 000 = 1x divider

REGISTER 25-4: DEVCFG3: DEVICE CONFIGURATION WORD 3

r-1	r-1	r-1	r-1	r-1	r-1	r-1	r-1
—	—	—	—	—	—	—	—
bit 31							bit 24

r-1	r-1	r-1	r-1	r-1	r-1	r-1	r-1
—	—	—	—	—	—	—	—
bit 23							bit 16

R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x
USERID15	USERID14	USERID13	USERID12	USERID11	USERID10	USERID9	USERID8
bit 15							bit 8

R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x
USERID7	USERID6	USERID5	USERID4	USERID3	USERID2	USERID1	USERID0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Maintain as ‘1’

bit 15-0 **USERID<15:0>:** This is a 16-bit value that is user defined and is readable via ICSP™ and JTAG.

REGISTER 25-5: DEVID: DEVICE ID REGISTER

R	R	R	R	r	r	r	r
VER3	VER2	VER1	VER0	—	—	—	—
bit 31							bit 24

r	r	r	r	R	R	R	R
—	—	—	—	DEV7	DEV6	DEV5	DEV4
bit 23							bit 16

R	R	R	R	R-0	R-0	R-0	R-0
DEV3	DEV2	DEV1	DEV0	MANID11	MANID10	MANID09	MANID08
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-1
MANID07	MANID06	MANID05	MANID04	MANID03	MANID02	MANID01	1
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-28 **VER<3:0>:** Revision Identifier bits

bit 27-20 **Reserved:** For factory use only

bit 19-12 **DEVID<7:0>:** Device ID
38h = PIC32MX360F512L
34h = PIC32MX360F256L
2Ah = PIC32MX320F128L
12h = PIC32MX340F256H
0Ah = PIC32MX320F128H
06h = PIC32MX320F064H
00h = PIC32MX300F032H

bit 11-1 **MANID<11:0>:** JEDEC Manufacturer’s Identification Code for Microchip Technology Inc.

bit 0 **Fixed Value:** Read as ‘1’

25.1 Device Configuration

In PIC32MX Family devices, the Configuration Words select various device configurations. These Configuration Words are implemented as volatile memory registers and must be loaded from the nonvolatile programmed configuration data mapped in the last four words (32-bit x 4 words) of boot Flash memory, DEVCFG0-DEVCFG3. These are the four locations an external programming device programs with the appropriate configuration data (see Table 25-3).

TABLE 25-3: DEVCFG LOCATIONS

Configuration Word	Address
DEVCFG0	0xBFC0_2FFC
DEVCFG1	0xBFC0_2FF8
DEVCFG2	0xBFC0_2FF4
DEVCFG3	0xBFC0_2FF0

On Power-on Reset (POR) or any Reset, the Configuration Words are copied from boot FLASH memory to their corresponding Configuration registers. A Configuration bit can only be programmed = `0` (unprogrammed state = `1`). During programming, a Configuration Word can be programmed a maximum of two times before a page erase must be performed.

After programming the Configuration Words, the user should reset the device to ensure the Configuration registers are reloaded with the new programmed data.

25.1.1 CONFIGURATION REGISTER PROTECTION

To prevent inadvertent Configuration bit changes during code execution, all programmable Configuration bits are write-once. After a bit is initially programmed during a power cycle, it cannot be written to again. Changing a device configuration requires changes to the configuration data in the boot Flash memory and power to the device be cycled.

To ensure the 128-bit data integrity, a comparison is continuously made between each Configuration bit and its stored complement. If a mismatch is detected, a Configuration Mismatch Reset is generated, causing a device Reset.

25.2 Device Code Protection

The PIC32MX Family features a single device code protection bit, CP that when programmed = `0`, protects boot Flash and program Flash from being read or modified by an external programming device. When code protection is enabled, only the Device ID and User ID registers are available to be read by an external programmer. Boot Flash and program Flash memory are not protected from self-programming during program execution when code protection is enabled. See **Section 25.3 “Program Write Protection (PWP)”**.

25.3 Program Write Protection (PWP)

In addition to a device code protection bit, the PIC32MX Family also features write protection bits to prevent boot Flash and program Flash memory regions from being written during code execution.

Boot Flash memory is write protected with a single Configuration bit, BWP (DEVCFG0<24>), when programmed = `0`.

Program Flash memory can be write-protected entirely or in selectable page sizes using Configuration bits PWP<7:0> (DEVCFG0<19:12>). A page of Program Flash memory is 4096 bytes (1024 words). The PWP bits represent the one’s complement of the number of protected pages. For example, programming PWP bits = 0xFF selects 0 pages to be write-protected, effectively disabling the program Flash write protection. Programming PWP bits = 0xFE selects the first page to be write protected. When enabled, the write-protected memory range is inclusive from the beginning of program Flash memory (0xBD00_0000) up through the selected page. Refer to Table 25-4.

> **Note:** The PWP bits represent the one’s complement of the number of protected pages.

TABLE 25-4: FLASH PROGRAM MEMORY WRITE-PROTECT RANGES

PWP Bit Value	Range Size (Kbytes)	Write Protected Memory Ranges(1)
0xFF	0	Disabled
0xFE	4	0xBD00_0FFF
0xFD	8	0xBD00_1FFF
0xFC	12	0xBD00_2FFF
0xFB	16	0xBD00_3FFF
0xFA	20	0xBD00_4FFF
0xF9	24	0xBD00_5FFF
0xF8	28	0xBD00_6FFF
0xF7	32	0xBD00_7FFF
0xF6	36	0xBD00_8FFF
0xF5	40	0xBD00_9FFF
0xF4	44	0xBD00_AFFF
0xF3	48	0xBD00_BFFF
0xF2	52	0xBD00_CFFF
0xF1	56	0xBD00_DFFF
0xF0	60	0xBD00_EFFF
0xEF	64	0xBD00_FFFF
...		
0x7F	512	0xBD07_FFFF

Note 1: Write-protected memory range is inclusive from 0xBD00_0000.

The amount of program Flash memory available for write protection depends on the family device variant.

25.4 On-Chip Voltage Regulator

All PIC32MX Family device's core and digital logic are designed to operate at a nominal 1.8V. To simplify system designs, most devices in the PIC32MX Family incorporate an on-chip regulator providing the required core logic voltage from V_{DD}.

The internal 1.8V regulator is controlled by the ENVREG pin. Tying this pin to V_{DD} enables the regulator, which in turn provides power to the core. A low ESR capacitor (such as tantalum) must be connected to the V_{DDCORE}/V_{CAP} pin (Figure 25-1). This helps to maintain the stability of the regulator. The recommended value for the filer capacitor is provided in **Section 29.1 "DC Characteristics"**.

Tying the ENVREG pin to V_{SS} disables the regulator. In this case, separate power for the core logic at a nominal 1.8V must be supplied to the device on the V_{DDCORE}/V_{CAP} pin.

Alternatively, the V_{DDCORE}/V_{CAP} and V_{DD} pins can be tied together to operate at a lower nominal voltage. Refer to Figure 25-1 for possible configurations.

25.4.1 ON-CHIP REGULATOR AND POR

When the voltage regulator is enabled, it takes approximately 10 µs for it to generate output. During this time, designated as $T_{STARTUP}$, code execution is disabled. $T_{STARTUP}$ is applied every time the device resumes operation after any power-down, including Sleep mode.

If the regulator is disabled, a separate Power-up Timer (PWRT) is automatically enabled. The PWRT adds a fixed delay of 64 ms nominal delay at device start-up.

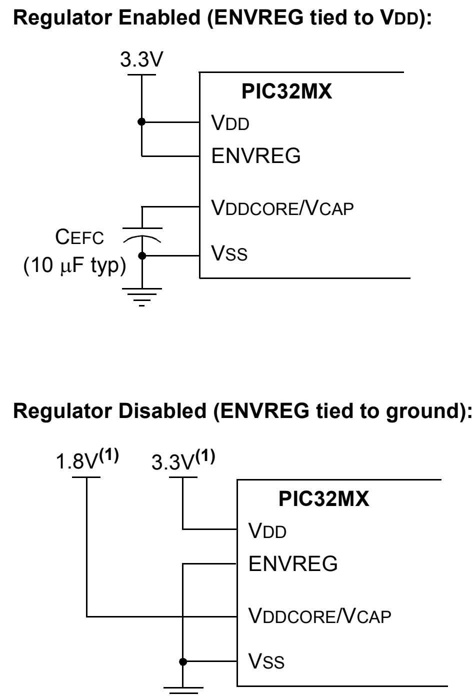
25.4.2 ON-CHIP REGULATOR AND BOR

When the on-chip regulator is enabled, PIC32MX Family devices also have a simple brown-out capability. If the voltage supplied to the regulator is inadequate to maintain a regulated level, the regulator Reset circuitry will generate a Brown-out Reset. This event is captured by the BOR flag bit (RCON<1>). The brown-out voltage levels are specific in **Section 29.1 "DC Characteristics"**.

25.4.3 POWER-UP REQUIREMENTS

The on-chip regulator is designed to meet the power-up requirements for the device. If the application does not use the regulator, then strict power-up conditions must be adhered to. While powering up, V_{DDCORE} must never exceed V_{DD} by 0.3 volts.

FIGURE 25-1: CONNECTIONS FOR THE ON-CHIP REGULATOR



Note 1: These are typical operating voltages. Refer to **Section 29.1 "DC Characteristics"** for the full operating ranges of V_{DD} and V_{DDCORE}.

NOTES:

26.0 WATCHDOG TIMER

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

This section describes the operation of the Watchdog Timer (WDT) and Power-up Timer of the PIC32MX.

The WDT, when enabled, operates from the internal Low-Power Oscillator (LPRC) clock source and can be used to detect system software malfunctions by resetting the device if the WDT is not cleared periodically in software. 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. Refer to Figure 26-1.

The following are some of the key features of the WDT module:

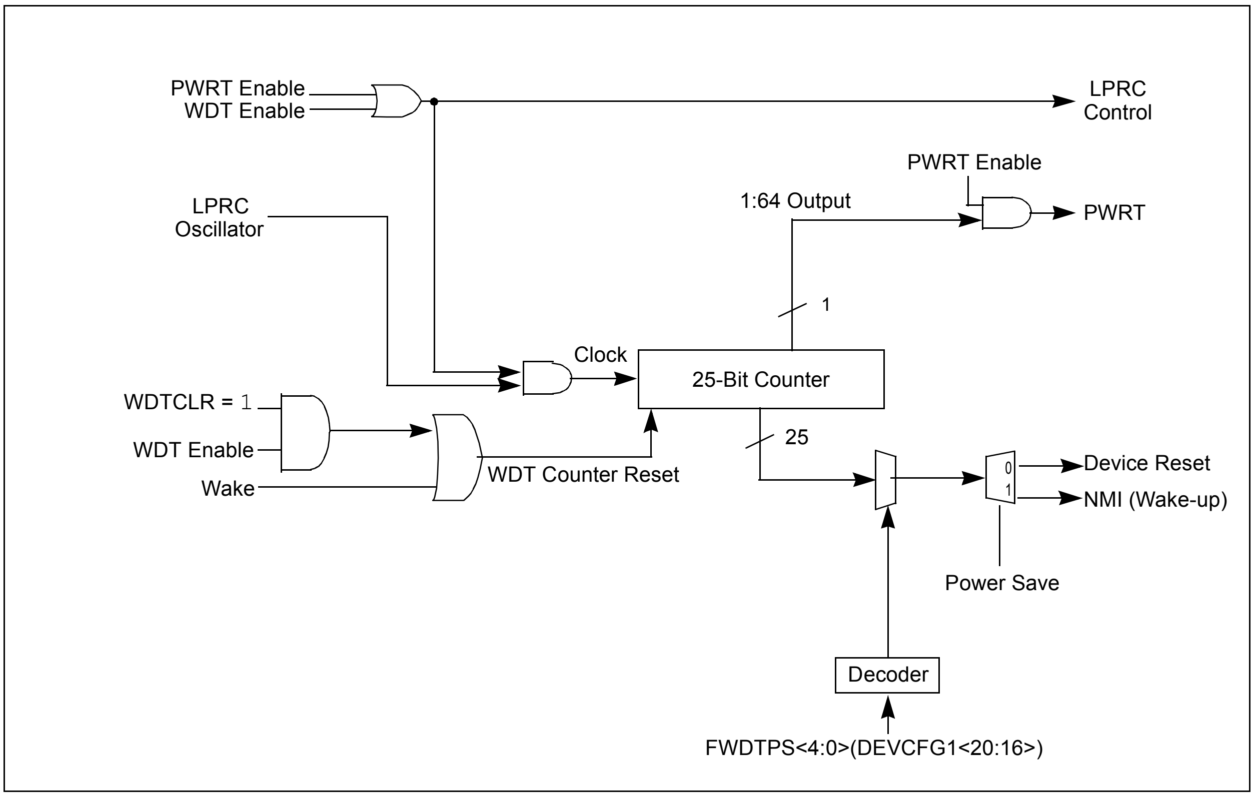
- Configuration or software controlled
- User-configurable time-out period
- Can wake the device from Sleep or Idle

TABLE 26-1: RESULTS OF A WDT TIME-OUT EVENT FOR AVAILABLE MODES OF DEVICE OPERATION

Device Mode	Device Reset Generated	Non-Maskable Interrupt Generated	WDTO[(1)] Bit Set	SLEEP[(1)] Bit Set	IDLE[(1)] Bit Set	Device Registers Reset
Awake	Yes	No	Yes	No	No	Yes
Sleep	No	Yes	Yes	Yes	No	No
Idle	No	Yes	Yes	No	Yes	No

Note 1: Status bits are in the RCON register.

FIGURE 26-1: WATCHDOG AND POWER-UP TIMER BLOCK DIAGRAM



26.1 Watchdog Timer Registers

TABLE 26-2: WDT SFR SUMMARY

Virtual Address	Name		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
BF80_0000	WDTCON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	—	—	—	—	—	—	—
		7:0	—	SWDTPS					—	WDTCLR
BF80_0004	WDTCONCLR	31:0	Write clears selected bits in WDTCON, Read yields an undefined value							
BF80_0008	WDTCONSET	31:0	Write sets selected bits in WDTCON, Read yields an undefined value							
BF80_000C	WDTCONINV	31:0	Write inverts selected bits in WDTCON, Read yields an undefined value							
BF80_F600	RCON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	TRAPR	—	—	—	—	—	CM	VREGS
		7:0	EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR
BF80_F604	RCONCLR	31:0	Write clears selected bits in RCON, Read yields an undefined value							
BF80_F608	RCONSET	31:0	Write sets selected bits in RCON, Read yields an undefined value							
BF80_F60C	RCONINV	31:0	Write inverts selected bits in RCON, Read yields an undefined value							
BFC0_2FF8	DEVCFG1	31:24	—	—	—	—	—	—	—	—
		23:16	FWDTEN	—	—	WDTPS4	WDTPS3	WDTPS2	WDTPS1	WDTPS0
		15:8	FCKSM1	FCKSM0	FPBDIV1	FPBDIV0	—	OSCIOFNC	POSCMD1	POSCMD0
		7:0	IESO	—	FSOSCEN	—	—	FNOSC2	FNOSC1	FNOSC0

REGISTER 26-1: WDTCON: WATCHDOG TIMER CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	U-0	U-0	U-0	U-0	r-1	r-1	r-0
ON	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R-x	R-x	R-x	R-x	R-x	r-0	R/W-0
—	SWDTPS					—	WDTCLR
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **ON:** Watchdog Timer Enable bit(1)
1 = Enables the WDT if it is not enabled by the device configuration
0 = Disable the WDT if is was enabled in software.

bit 14-7 **Unimplemented:** Read as ‘0’

bit 6-2 **SWDTPS<4:0>:** Shadow Copy of Watchdog Timer Post-Scaler Value from Device Configuration bits

bit 1 **Reserved:** Maintain as ‘0’

bit 0 **WDTCLR:** Watchdog Timer Reset bit
1 = Writing a ‘1’ will reset the WDT.
0 = Software cannot force this bit to a ‘0’.

Note 1: A read of this bit will result in a ‘1’ if the WDT is enabled by the device configuration or by software.

Register 26-1: RCON: RESETS CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	U-0	U-0	U-0	U-0	R-0	R/W-0	R/W-0
TRAPR	—	—	—	—	—	CM	VREGS
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 4 **WDTO:** Watchdog Time-Out bit

1 = A WDT time-out has occurred since the device was powered up
0 = A WDT time-out has not occurred since the WDTO bit was cleared by software

bit 3 **SLEEP:** Sleep Mode Status bit

1 = The device has been in Sleep mode since the device was powered up
0 = The device has not been in Sleep mode since the SLEEP bit was cleared by software

bit 2 **IDLE:** Idle Mode Status bit

1 = The device has been in Idle mode since the device was powered up
0 = The device has not been in Idle mode since the Idle bit was cleared by software

REGISTER 26-2: DEVCFG1: DEVICE CONFIGURATION WORD 1

U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1
—	—	—	—	—	—	—	—
bit 31							bit 24

R/P-1	r-1	U-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
FWDTEN	—	—	WDTPS<4:0>				
bit 23							bit 16

R/P-1	R/P-1	U-1	U-1	U-1	R/P-1	R/P-1	R/P-1
FCKSM<1:0>		FPBDIV1	FPBDIV0	—	OSCIOFNC	POSCMD<1:0>	
bit 15							bit 8

R/P-1	U-1	U-1	U-1	U-1	R/P-1	R/P-1	R/P-1
IESO	—	FSOSCEN	—	—	FNOSC<2:0>		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-24 **Reserved:** Maintain as ‘1’

bit 23 **FWDTEN:** WatchDog Timer Hardware Enable bit.
1 = The WDT is enabled and cannot be disabled by software
0 = The WDT is not enabled. It can be enabled in software

bit 22 **Reserved:** Maintain as ‘1’

bit 20-16 **WDTPS<4:0>:** Watchdog Timer Postscaler Selection bits(1)
These bits are used to set the WDT time-out period.
10100 = 1:1,045,876
10011 = 1:524,288
10010 = 1:262,144
10001 = 1:131,072
10000 = 1:65,536
01111 = 1:32,768
01110 = 1:16,384
01101 = 1:8,192
01100 = 1:4,096
01011 = 1:2,048
01010 = 1:1,024
01001 = 1:512
01000 = 1:256
00111 = 1:128
00110 = 1:64
00101 = 1:32
00100 = 1:16
00011 = 1:8
00010 = 1:4
00001 = 1:2
00000 = 1:1

Note 1: All combinations not listed result in operation as if the selection was 10100 .
2: Do not disable POSC (POSCMD = 00) when using this oscillator source.

bit 15-14 **FCKSM<1:0>:** Clock Switching and Monitor Selection Configuration bits
1x = Clock switching is disabled, Fail-Safe Clock Monitor is disabled
01 = Clock switching is enabled, Fail-Safe Clock Monitor is disabled
00 = Clock switching is enabled, Fail-Safe Clock Monitor is enabled

bit 13-12 **FPBDIV<1:0>:** Peripheral Bus Clock Divisor Default Value bits
11 = PBCLK is SYSCLK divided by 8
10 = PBCLK is SYSCLK divided by 4
01 = PBCLK is SYSCLK divided by 2
00 = PBCLK is SYSCLK divided by 1

bit 11 **Reserved:** Maintain as '1'

bit 10 **OSCIOFNC:** CLKO Enable Configuration bit
1 = CLKO output signal active on the OSCO pin; primary oscillator must be disabled or configured for the External Clock (EC) mode for the CLKO to be active (POSCMD<1:0> = 11 or 00)
0 = CLKO output disabled

bit 9-8 **POSCMD<1:0>:** Primary Oscillator Configuration bits
11 = Primary oscillator is disabled
10 = HS Oscillator mode selected
01 = XT Oscillator mode selected
00 = External Clock mode selected

bit 7 **IESO:** Internal External Switchover bit
1 = Internal External Switchover mode enabled (Two-Speed Start-Up enabled)
0 = Internal External Switchover mode disabled (Two-Speed Start-Up disabled)

bit 6 **Reserved:** Maintain as '1'

bit 5 **FSOSCEN:** Secondary Oscillator Enable bits
1 = Enable secondary oscillator
0 = Disable secondary oscillator

bit 4-3 **Reserved:** Maintain as '1'

bit 2-0 **FNOSC<2:0>:** Oscillator Selection bits
000 = Fast RC Oscillator (FRC)
001 = Fast RC Oscillator with divide-by-N with PLL module (FRCDIV + PLL)
010 = Primary Oscillator (XT, HS, EC)[(2)]
011 = Primary Oscillator with PLL module (XT + PLL, HS + PLL, EC + PLL)[(2)]
100 = Secondary Oscillator
101 = Low-Power RC Oscillator (LPRC)
110 = FRCDIVIG Fast RC Oscillator with fixed divide-by-16 postscaler
111 = Fast RC Oscillator with divide-by-N (FRCDIV)

Note 1: All combinations not listed result in operation as if the selection was 10100 .
2: Do not disable POSC (POSCMD = 00) when using this oscillator source.

26.2 Watchdog Timer and Power-Up Timer Operation

This describes the operation of the Watchdog Timer operation and the Power-up Timer

26.2.1 WATCHDOG TIMER OPERATION

If enabled, the WDT will increment until it overflows or “times out”. A WDT time-out will force a device Reset, except during Sleep or Idle modes. To prevent a WDT time-out Reset, the user must periodically clear the Watchdog Timer by setting the WDTCLR (WDTCON<0>) bit.

The WDT uses the LPRC oscillator for reliability.

Note:	The LPRC is enabled whenever the WDT is enabled.

26.2.2 ENABLING AND DISABLING THE WDT

The WDT is enabled or disabled by the device configuration or controlled via software by writing to the WDTCON register.

26.2.3 DEVICE CONFIGURATION CONTROLLED WDT

If the FWDTEN Configuration bit is set, then the WDT is always enabled. The WDT ON control bit (WDTCON<15>) will reflect this by reading a ‘1’. In this mode, the ON bit cannot be cleared in software. This bit will not be cleared by any form of Reset. To disable the WDT in this mode, the configuration must be rewritten to the device.

Note:	The default state for the WDT on an unprogrammed device is WDT enabled.

26.2.4 SOFTWARE CONTROLLED WDT

If the FWDTEN Configuration bit is a ‘0’, then the WDT can be enabled or disabled (the default condition) by software. In this mode, the ON (WDTCON<15>) bit reflects the status of the WDT under software control. A ‘1’ indicates the WDT is enabled and a ‘0’ indicates it is disabled.

The WDT is enabled in software by setting the WDT ON control bit. The WDT ON control bit is cleared on any device Reset, The bit is not cleared upon a wake from Sleep or exit from Idle mode. The software WDT option allows the user to enable the WDT for critical code segments and disable the WDT during noncritical segments for maximum power savings. This bit can also be used to disable the WDT while the part is awake to eliminate the need for WDT servicing, and then re-enable it before the device is put into Idle or Sleep to wake the part at a later time.

26.2.5 WDT OPERATION IN POWER SAVE MODES

The WDT, if enabled, will continue operation in Sleep or Idle modes. The WDT may be used to wake the device from Sleep or Idle. When the WDT times out in a Power Save mode, a Non-Maskable Interrupt (NMI) is generated and the WDTO (RCON<4>) bit is set. The NMI vectors execution to the CPU start-up address but does not reset registers or peripherals. If the device was in Sleep, the SLEEP (RCON<3>) status bit will also be set. If the device was in Idle, the IDLE (RCON<2>) status bit will also be set. These bits allow the start-up code to determine the cause of the wake-up.

26.2.6 TIME DELAYS ON WAKE

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 and the Power-up Timer delay, if it is enabled.

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.

26.2.7 RESETTING THE WDT TIMER

The WDT is reset by any of the following:

- On ANY device Reset
- By a `WDTCONSET = 0x01` or equivalent instruction during normal execution.
- Execution of a DEBUG command
- Exiting from Idle or Sleep due to an interrupt

Note:	The WDT timer is not reset when the device enters a Power Save mode. The WDT should be serviced prior to entering a Power Save mode.

26.2.8 WDT TIMER PERIOD SELECTION

The WDT clock source is the internal LPRC oscillator, which has a nominal frequency of 32 kHz. This creates a nominal time-out period for the WDT (T_{WDT}) of 1 millisecond when no postscaler is used.

Note:	The WDT time-out period is directly related to the frequency of the LPRC oscillator. The frequency of the LPRC oscillator will vary as a function of device operating voltage and temperature. Please refer to the specific PIC32MX device data sheet for LPRC clock frequency specifications.

26.2.9 WDT POSTSCALERS

The WDT has a 5-bit postscaler to create a wide variety of time-out periods. This postscaler provides 1:1 through 1: 1048576 divider ratios. Time-out periods that range between 1 ms and 1048.576 seconds (nominal) can be achieved using the postscaler.

The postscaler settings are selected using the WDTPS bits in the DEVCFG1 Configuration register. The time-out period of the WDT is calculated as follows:

EQUATION 26-1: WDT TIME-OUT PERIOD CALCULATIONS

$$\text{WDT Period} = 1\text{ ms} \bullet 2^{\text{Prescaler}}$$

TABLE 26-3: WDT TIME-OUT PERIOD VS. POSTSCALER SETTINGS

FWDTPS<4:0>	Postscaler Ratio	Time-out Period
00000	1:1	1 ms
00001	1:2	2 ms
00010	1:4	4 ms
00011	1:8	8 ms
00100	1:16	16 ms
00101	1:32	32 ms
00110	1:64	64 ms
00111	1:128	128 ms
01000	1:256	256 ms
01001	1:512	512 ms
01010	1:1024	1.024 s
01011	1:2048	2.048 s
01100	1:4096	4.096 s
01101	1:8192	8.192 s
01110	1:16384	16.384 s
01111	1:32768	32.768 s
10000	1:65536	65.536 s
10001	1:131072	131.072 s
10010	1:262144	262.144 s
10011	1:524288	524.288 s
10100	1:1045876	1048.576 s

Note 1: All other combinations will result in an operation as if the prescaler was set to 10100.

2: The periods listed are based on a 32 kHz (nominal) input clock.

26.3 Interrupts and Resets

The WDT will cause an NMI or a device Reset when it expires. The Power Save mode of the device determines which event occurs. The PWRT does not generate interrupts or Resets.

26.3.1 WATCHDOG TIMER RESET

When the WDT expires and the device is not in Sleep or Idle, a device Reset is generated. The CPU code execution jumps to the device Reset vector and the Registers and Peripherals are forced to their Reset values.

To detect a WDT Reset, the WDTO (RCON<4>), SLEEP (RCON<3>) and IDLE (WDTCON<2>) bits must be tested. If the WDTO bit is a '1', the event was do to a WDT time-out. The SLEEP and IDLE bits can then be tested to determine if the WDT event occurred while the device was awake or if it was in Sleep or Idle.

26.3.2 WATCHDOG TIMER NMI

When the WDT expires in Sleep or Idle, a NMI is generated. The NMI causes the CPU code execution to jump to the device Reset vector. Though the NMI share the same vector as a device Reset, registers and peripherals are not reset.

To detect a wake from a Power Save mode by WDT, the WDTO (RCON<4>), SLEEP (RCON<3>) and IDLE (WDTCON<2>) bits must be tested. If the WDTO bit is a '1' the event was caused by a WDT time-out. The SLEEP and IDLE bits can then be tested to determine if the WDT event occurred in Sleep or Idle.

To cause a WDT time-out in Sleep to act like an interrupt, a return from interrupt instruction may be used in the start-up code after the event was determined to be a WDT wake-up. This will cause code execution to continue with the opcode following the `WAIT` instruction that put the device into Power Save mode. See Example 26-1.

EXAMPLE 26-1: SAMPLE WDT INITIALIZATION AND SERVICING

```
//This code fragment assumes the WDT was not enabled by the device configuration
   // The Postscaler value must be set with the device configuration

   WDTCONSET = 0x8000;// Turn on the WDT

   main
   {
      WDTCONSET = 0x01;// Service the WDT

      ... User code goes here ...
   }
```

EXAMPLE 26-2: SAMPLE CODE TO DETERMINE THE CAUSE OF A WDT EVENT

```
// sample code to determine the cause of a WDT event

// Unlock the OSCCON register
asm (“la $t3, SYSREG”);//load the address of SYSREG into t3
asm (“li $t0,0xaa996655”);// load Key value into t0
asm (”nor $t1, $0, $t0”);// complement Key1 to form Key2
// the following writes must be performed back to back
asm (”sw $t0, 0($t3)”);         //write Key1 to SYSREG
asm (”sw $t1, 0($t3)”);         //write Key2 to SYSREG
// OSCCON is now unlocked

OSCCONSET = 0x10;// set power save mode to Sleep

// Alternate relock code in ‘C’
SYSREG = 0x33333333;
// OSCCON is relocked

    WDTCONSET = 0x8000;//Enable WDT

    while (1)
    {
        ... user code ...

        WDTCONSET = 0x01;// service the WDT
        asm ( “wait” );// put device is selected power save mode

        // code execution will resume here after wake

        ... user code ...
    }


// The following code fragment is at the top of the device start-up code

    if ( RCON & 0x18 )
    {
        // The WDT caused a wake from sleep
        asm ( “eret” );// return from interrupt
    }

    if ( RCON & 0x14 )
    {
        // The WDT caused a wake from idle
        asm ( “eret” );// return from interrupt
    }

    if ( RCON & 0x10 )
    {
        // The WDT timed-out while the device was awake
    }
```


27.0 PROGRAMMING AND DIAGNOSTICS

Note:	This data sheet summarizes the features of the PIC32MX family of devices. It is not intended to be a comprehensive reference source. Refer to the *"PIC32MX Family Reference Manual"* (DS61132) for a detailed description of this peripheral.

PIC32MX Family devices provide a complete range of programming and diagnostic features that can increase the flexibility of any application using them. These features allow system designers to include:

- Simplified field programmability using two-wire In-Circuit Serial Programming™ (ICSP™) interfaces
- Debugging using ICSP
- Programming and debugging capabilities using the EJTAG extension of JTAG
- JTAG boundary scan testing for device and board diagnostics

PIC32MX Family devices incorporate two programming and diagnostic modules, and a trace controller, that provide a range of functions to the application developer. They are summarized in Table 27-1.

FIGURE 27-1: BLOCK DIAGRAM OF PROGRAMMING, DEBUGGING, AND TRACE PORTS

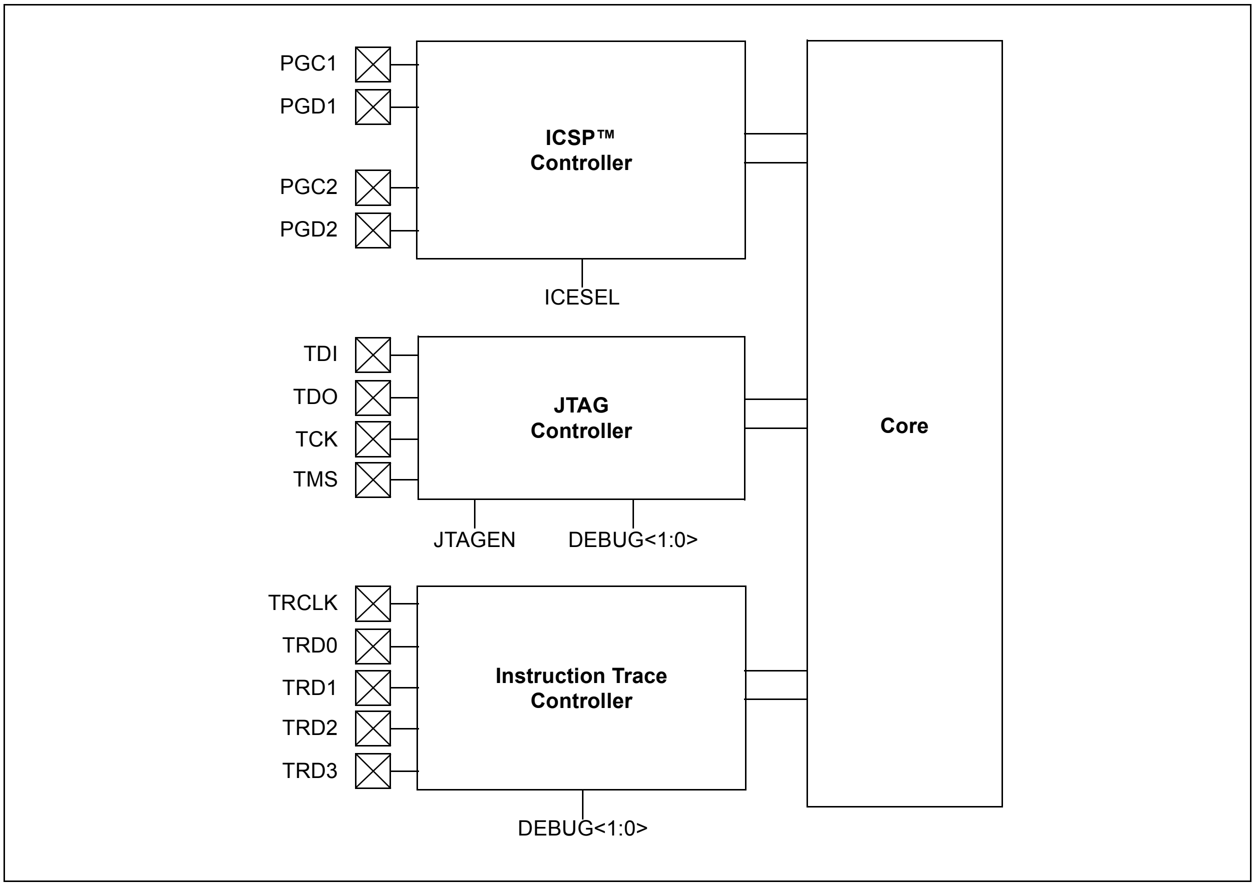


TABLE 27-1: COMPARISON OF PIC32MX FAMILY PROGRAMMING AND DIAGNOSTIC FEATURES

Functions	Pins Used	Interface
Boundary Scan	TDI, TDO, TMS and TCK pins	JTAG
Programming and Debugging	TDI, TDO, TMS and TCK pins	EJTAG
Programming and Debugging	PGCx and PGDx pins	ICSP™

27.1 Control Registers

The programming and diagnostics module consists of the following Special Function Registers (SFRs):

- DDPCON: Control Register for the Diagnostic Module

  DDPCONCLR, DDPCONSET, DDPCONINV: Atomic Bit Manipulation Write-Only Registers for DDPCON
- DEVCFG0: Device Configuration Register

The following table summarizes all programming and diagnostics related registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

TABLE 27-2: PROGRAMMING AND DIAGNOSTICS SFR SUMMARY

Virtual Address	Name		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
BF80_F200	DDPCON	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	—	DDPU1	DDPU2	DDPSPI1	JTAGEN	TROEN	—	—
BF80_F204	DDPCONCLR	31:0	Write clears selected bits in DDPCON, read yields undefined value							
BF80_F208	DDPCONSET	31:0	Write sets selected bits in DDPCON, read yields undefined value							
BF80_F20C	DDPCONINV	31:0	Write inverts selected bits in DDPCON, read yields undefined value							
BFC0_2FFC	DEVCFG0	31:24	—	—	—	CP	—	—	—	BWP
		23:16	—	—	—	—	PWP7	PWP6	PWP5	PWP4
		15:8	PWP3	PWP2	PWP1	PWP0	—	—	—	—
		7:0	—	—	—	—	ICESEL	—	DEBUG1	DEBUG0

REGISTER 27-1: DDPCON: DEBUG DATA PORT CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-r	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	U-0	U-0
—	DDPU1	DDPU2	DDPSPI1	JTAGEN	TROEN	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-8 **Unimplemented:** Read as ‘0’

bit 7 **Reserved:** Maintain as ‘0’

bit 6 **DDPU1:** Debug Data Port Enable for UART1 bit
1 = UART1 peripheral ignores FRZ (U1MODE<14>) setting
0 = UART1 peripheral follows FRZ setting

bit 5 **DDPU2:** Debug Data Port Enable for UART2 bit
1 = UART2 peripheral ignores FRZ (U2MODE<14) setting
0 = UART2 peripheral follows FRZ setting

bit 4 **DDPSPI1:** Debug Data Port Enable for SPI1 bit
1 = SPI1 peripheral ignores FRZ (SPI1CON<14>) setting
0 = SPI1 peripheral follows FRZ setting

bit 3 **JTAGEN:** JTAG Port Enable bit
1 = Enable JTAG Port
0 = Disable JTAG Port

bit 2 **TROEN:** Trace Output Enable bit
1 = Enable Trace Port
0 = Disable Trace Port

bit 1-0 **Unimplemented:** Read as ‘0’

REGISTER 27-2: DEVCFG0: DEVICE CONFIGURATION REGISTER

r-1	U-1	U-1	R/P-1	U-1	U-1	U-1	R/P-1
—	—	—	CP	—	—	—	BWP
bit 31							bit 24

U-1	U-1	U-1	U-1	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	—	PWP7	PWP6	PWP5	PWP4
bit 23							bit 16

R/P-1	R/P-1	R/P-1	R/P-1	U-1	U-1	U-1	U-1
PWP3	PWP2	PWP1	PWP0	—	—	—	—
bit 15							bit 8

U-1	U-1	U-1	U-1	R/P-1	U-1	R/P-1	R/P-1
—	—	—	—	ICESEL	—	DEBUG1	DEBUG0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 3 **ICESEL:** ICE Debugger Port Select bit
1 = ICE debugger uses PGC2/PGD2
0 = ICE debugger uses PGC1/PGD1

bit 1-0 **DEBUG<1:0>:** Background Debugger Enable bits (forced to ‘11’ if code-protect is enabled)
11 = ICE debugger disabled
10 = ICE debugger enabled
01 = Reserved (same as ‘11’ setting)
00 = Reserved (same as ‘11’ setting)

27.2 Operation

The PIC32MX Family of devices has multiple programming and Debugging options including:

- In-Circuit Serial Programming via ICSP
- In-Circuit Programming EJTAG
- Debugging via ICSP
- Debugging via EJTAG
- Special Debug modes for Select Communication Peripherals
- Boundary Scan

27.2.1 DEVICE PROGRAMMING OPTIONS

Note:	The following sections provide a brief overview of each programming option. For more detailed information, refer to "*PIC32MX Flash Programming Specification*" (DS61145).

27.2.1.1 In-Circuit Serial Programming

ICSP is Microchip's proprietary solution to providing microcontroller programming in the target application. ICSP is also the most direct method to program the device, whether the controller is embedded in a system or loaded into a device programmer.

27.2.1.2 ICSP Interface

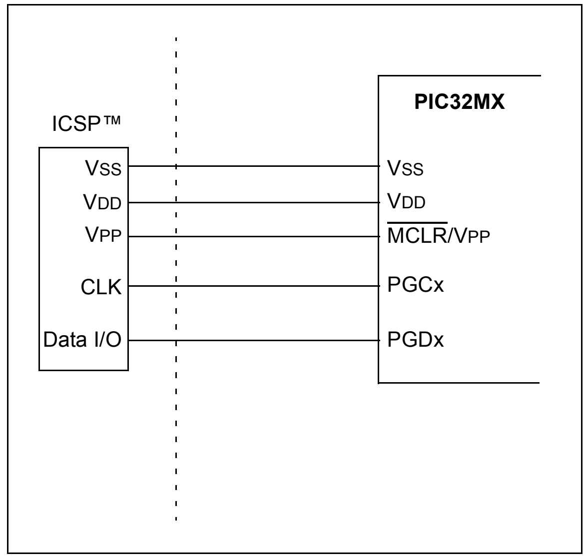
ICSP uses two pins as the core of its interface. The Programming Data (PGD) line functions as both an input and an output, allowing programming data to be read in and device information to be read out on command. The Programming Clock (PGC) line is used to clock in data and control the overall process.

PIC32MX Family devices have more than one pair of PGC and PGD pins; these are multiplexed with other I/O or peripheral functions. Individual ICSP pin pairs are indicated by number (e.g., PGC1/PGD1, etc.), and are generically referred to as 'PGCx' and 'PGDx'. The multiple PGCx/PGDx pairs provide additional flexibility in system design by allowing users to incorporate ICSP on the pair of pins that is least constrained by the circuit design. All PGCx and PGDx pins are functionally tied together and behave identically, and any one pair can be used for successful device programming. The only limitation is that both pins from the same pair must be used.

In addition to the PGCx and PGDx pins, ICSP requires that all voltage supply (including voltage regulator pin ENVREG) and ground pins on the device must be connected. The $\overline{MCLR}$ pin, which is used with PGCx to enter and control the programming process, must also be connected to the programmer.

A typical In-Circuit Serial Programming connection is shown in Figure 27-2.

FIGURE 27-2: TYPICAL IN-CIRCUIT SERIAL PROGRAMMING™ CONNECTION



27.2.1.3 ICSP Operation

ICSP uses a combination of internal hardware and external control to program the target device. Programming data and instructions are provided on PGD. ICSP uses a special set of commands to control the overall process, combined with standard PIC32MX Family instructions to execute the actual writing of the program memory. PGD also returns data to the external programmer when responding to queries.

Users who are interested in a more detailed description, or who are considering designing their own programming interface for PIC32MX Family devices, should consult the appropriate PIC32MX Family device programming specification.

27.2.1.4 Enhanced In-Circuit Serial Programming

The Enhanced In-Circuit Serial Programming (ICSP) protocol is an extension of the original ICSP. It uses the same physical interface as the original, but changes the location and execution of programming control to a software application written to the PIC32MX Family device. Use of Enhanced ICSP results in significant decrease in overall programming time.

For additional information on Enhanced ICSP and the program executive, refer to the appropriate PIC32MX Family device programming specification.

27.2.1.5 EJTAG Device Programming Using the JTAG Interface

The JTAG interface can also be used to program PIC32MX Family devices in their target applications. Using EJTAG with the JTAG interface allows application designers to include a dedicated test and programming port into their applications, with a single 4-pin interface, without imposing the circuit constraints that the ICSP interface may require.

27.2.1.6 Enhanced EJTAG Programming Using the JTAG Interface

Enhanced EJTAG programming uses the standard JTAG interface but uses a programming executive written to RAM. Use of the programming executive with the JTAG interface provides a significant improvement in programming speed.

27.2.2 DEBUGGING

27.2.2.1 ICSP and In-Circuit Debugging

ICSP also provides a hardware channel for the In-Circuit Debugger (ICD) which allows externally controlled debugging of software. Using the appropriate hardware interface and software environment, users can force the device to single step through its code, track the actual content of multiple registers and set software breakpoints.

The active ICSP debugger port is selected by the ICS Configuration bit.

27.2.2.2 EJTAG Debugging

The industry standard EJTAG interface allows third party EJTAG tools to be used for debugging. Using the EJTAG interface, memory and registers can be viewed and modified. Breakpoints can be set and the program execution may be stopped, started or single stepped.

27.2.3 SPECIAL DEBUG MODES FOR SELECT COMMUNICATIONS PERIPHERALS

To aid in debugging applications certain I/O peripherals have a user-controllable bit to override the Freeze function in the peripheral. This allows the module to continue to send any data, buffered within the peripheral, even when a debugger attempts to halt the peripheral. The Debug mode control bits for these peripherals are contained in the DDPCON register.

27.2.4 JTAG BOUNDARY SCAN

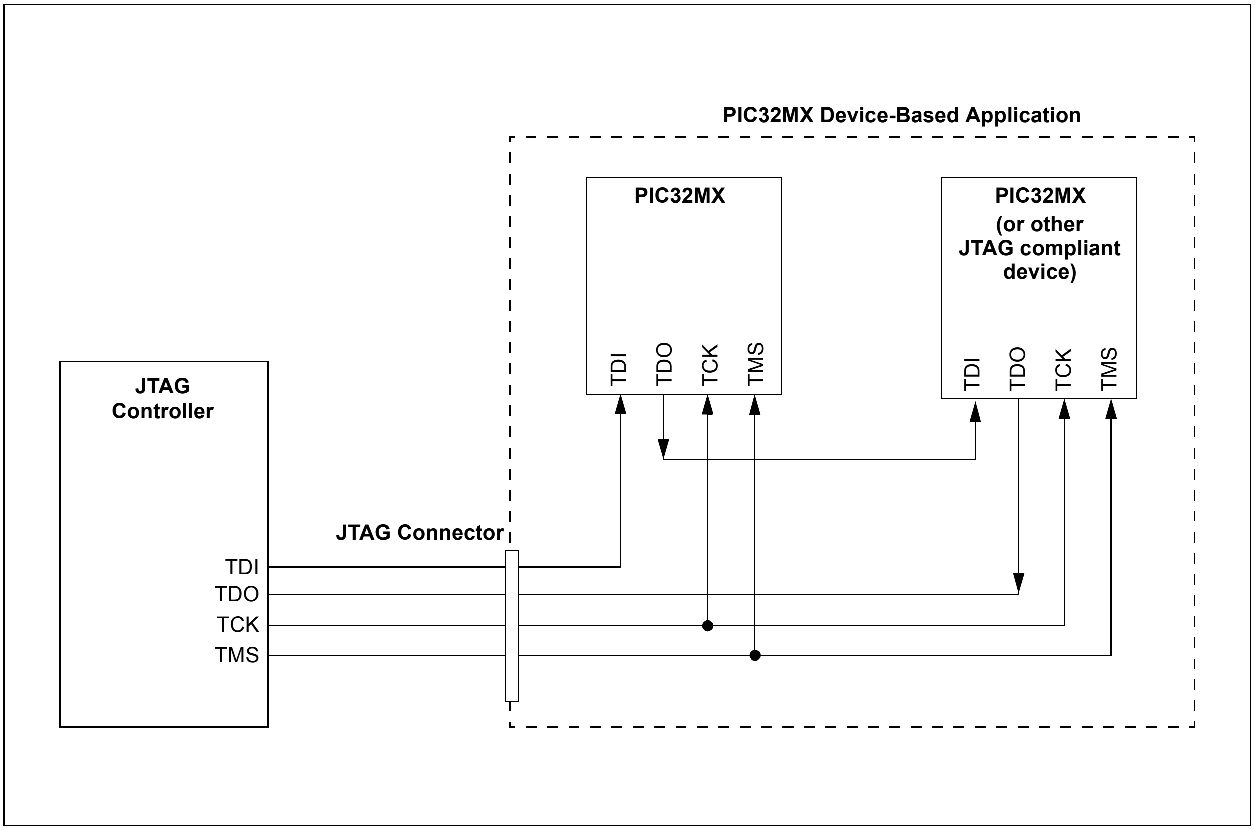
The JTAG boundary scan method is the process of adding a Shift register stage adjacent to each of the component's I/O pins. This permits signals at the component boundaries to be controlled and observed, using a defined set of scan test principles. An external tester or controller provides instructions and reads the results in a serial fashion.

The external device also provides common clock and control signals. Depending on the implementation, access to all test signals is provided through a standardized, 4-pin interface.

A typical application incorporating the JTAG boundary scan interface is shown in Figure 27-3. In this example, a PIC32MX Family microcontroller is daisy-chained to a second JTAG compliant device. Note that the TDI line from the external tester supplies data to the TDI pin of the first device in the chain (in this case, the microcontroller). The resulting test data for this two-device chain is provided from the TDO pin of the second device to the TDO line of the tester.

This section describes the JTAG module and its general use. Users interested in using the JTAG interface for device programming should refer to the appropriate PIC32MX Family device programming specification for more information.

FIGURE 27-3: OVERVIEW OF PIC32MX FAMILY-BASED JTAG COMPLIANT APPLICATION SHOWING DAISY-CHAINING OF COMPONENTS



In PIC32MX Family devices, the hardware for the JTAG boundary scan is implemented as a peripheral module (i.e., outside of the CPU core) with additional integrated logic in all I/O ports. A logical block diagram of the JTAG module is shown in Figure 27-1. It consists of the following key elements:

- TAP Interface Pins (TDI, TMS, TCK and TDO)
- TAP Controller
- Instruction Shift register and Instruction Register (IR)
- Data Registers (DR)

27.2.4.1 Test Access Port (TAP) and TAP Controller

The Test Access Port (TAP) on the PIC32MX Family device is a general purpose port that provides test access to many built-in support functions and test logic defined in IEEE 1149.1. The TAP is enabled by the JTAGEN bit in the DDPCON register. The TAP is enabled, JTAGEN = `1`, by default when the device exits Power-on-Reset (POR) or any device Reset. Once enabled, the designated I/O pins become dedicated TAP pins.

The PIC32MX Family implements a 4-pin JTAG interface with these pins:

- TCK (Test Clock Input): Provides the clock for test logic.
- TMS (Test Mode Select Input): Used by the TAP to control test operations.
- TDI (Test Data Input): Serial input for test instructions and data.
- TDO (Test Data Output): Serial output for test instructions and data.

27.2.4.2 JTAG Registers

The JTAG module uses a number of registers of various sizes as part of its operation. In terms of bit count, most of the JTAG registers are single bit register cells, integrated into the I/O ports. Regardless of their location within the module, none of the JTAG registers are located within the device data memory space, and cannot be directly accessed by the user in normal operating modes.

27.2.4.3 Instruction Shift Register and Instruction Register

The Instruction Shift register is a 4-bit Shift register used for selecting the actions to be performed and/or what data registers to be accessed. Instructions are shifted in, Least Significant bit first, and then decoded.

A list and description of implemented instructions is given in **Section 27.2.4.6 "JTAG Instructions"**.

27.2.4.4 Data Registers

Once an instruction is shifted in and updated into the Instruction Register, the TAP controller places certain data registers between the TDI and TDO pins. Additional data values can then be shifted into these data registers as needed.

The PIC32MX Family device supports three data registers:

- BYPASS Register: A single bit register which allows the boundary scan test data to pass through the selected device to adjacent devices. The BYPASS register is placed between the TDI and TDO pins when the `BYPASS` instruction is active.
- DEVID Register: A 32-bit part identifier. It consists of an 11-bit manufacturer ID assigned by the IEEE (29h for Microchip Technology), device part number and device revision identifier. When the `IDCODE` instruction is active, the device ID register is placed between the TDI and TDO pins. The device data ID is then shifted out on to the TDO pin, on the next 32 falling edges of TCK, after the TAP controller is in the Shift_DR.
- MCHP Command Shift Register: An 8-bit Shift register that is placed between the TDI and TDO pins when the `MCHP_CMD` instruction is active. This Shift register is used to shift in Microchip commands.

27.2.4.5 Boundary Scan Register (BSR)

The BSR is a large Shift register that is comprised of all the I/O Boundary Scan Cells (BSCs), daisy-chained together. Each I/O pin has one BSC, each containing 3 BSC registers, an input cell, an output cell and a control cell. When the `SAMPLE/PRELOAD` or `EXTEST` instructions are active, the BSR is placed between the TDI and TDO pins, with the TDI pin as the input and the TDO pin as the output.

The size of the BSR depends on the number of I/O pins on the device. For example, the 100-pin PIC32MX general purpose parts have 82 I/O pins. With 3 BSC registers for each of the 82 I/Os, this yields a Boundary Scan register length of 244 bits. This is due to the $\overline{\text{MCLR}}$ pin being an input only BSR cell. Information on the I/O port pin count of other PIC32MX Family devices can be found in their specific device data sheets.

27.2.4.6 JTAG Instructions

PIC32MX Family devices support the mandatory instruction set specified by IEEE 1149.1, as well as several optional public instructions defined in the specification. These devices also implement instructions that are specific to Microchip devices.

The mandatory JTAG instructions are:

- `BYPASS` (0x1F): Used for bypassing a device in a test chain; this allows the testing of off-chip circuitry and board level interconnections.
- `SAMPLE/PRELOAD` (0x02): Captures the I/O states of the component, providing a snapshot of its operation.
- `EXTEST` (0x06): Allows the external circuitry and interconnections to be tested, by either forcing various test patterns on the output pins, or capturing test results from the input pins.

Microchip has implemented optional JTAG instructions and manufacturer-specific JTAG commands in PIC32MX Family devices. Please refer to Table 27-3, Table 27-4, Table 27-5 and Table 27-6.

TABLE 27-3: JTAG COMMANDS

Opcode	Name	Device Integration
0x1F	Bypass	Bypasses device in test chain
0x00	HIGHZ	Places device in a high-impedance state, all pins are forced to inputs
0x01	ID Code	Shifts out the device's ID code
0x02	Sample/Preload	Samples all pins or loads a specific value into output latch
0x06	EXTEST	Boundry scan

TABLE 27-4: MICROCHIP TAP IR COMMANDS

Opcode	Name	Device Integration
0x01	MTAP_IDCODE	Shifts out the device's ID code
0x07	MTAP_COMMAND	Configures Microchip TAP controller for DR commands
0x04	MTAP_SW_MTAP	Selects Microchip TAP controller

Opcode	Name	Device Integration
0x05	MTAP_SW_ETAP	Selects EJTAG TAP controller

TABLE 27-5: MICROCHIP TAP 8-BIT DR COMMANDS

Opcode	Name	Device Integration
0x00	MCHP_STATUS	Performs NOP and returns status
0xD1	MCHP_ASERT_RST	Requests Assert Device Reset
0xD0	MCHP_DE_ASSERT_RST	Requests Deassert Device Reset
0xFC	MCHP_ERASE	Performs a chip erase
0xFE	MCHP_FLASH_ENABLE	Enables fetches and loads to the Flash from the CPU
0xFD	MCHP_FLASH_DISABLE	Disables fetches and loads to the Flash from the CPU
0xFF	MCHP_READ_CONFIG	Forces device to reread the configuration settings and initialize accordingly

TABLE 27-6: EJTAG COMMANDS

Opcode	Name	Device Integration	Data Length for the Following DR
0x00		Not used	
0x01	IDCODE	Selects the device's ID Code register	32 bits
0x02		Not used	
0x03	IMPCODE	Selects Implementation register	
0x04(2)	MTAP_SW_MTAP	Selects Microchip TAP controller	
0x05(2)	MTAP_SW_ETAP	Selects EJTAG TAP controller(1)	
0x06-0x07		Not used	
0x08	ADDRESS	Selects the Address register	32 bits
0x09	DATA	Selects the Data register	32 bits
0x0A	CONTROL	Selects the EJTAG Control register(1)	32 bits
0x0B	ALL	Selects the Address, Data, EJTAG Control register(1)	96 bits
0x0C	EJTAGBOOT	Forces the CPU to take a debug exception after boot	1 bit
0x0D	NORMALBOOT	Makes the CPU execute the reset handler after a boot	1 bit
0x0E	FASTDATA	Selects the Data and Fast Data registers	1 bit
0x0F-0x1B		Reserved	
0x1C-0xFE		Not used	
0xFF		Selects the Bypass register	

Note 1: For complete information about EJTAG commands and protocol, refer to the EJTAG Specification available on MIPS Technologies web site, www.mips.com.

2: Not EJTAG commands but are recognized by the Microchip implementation.

27.2.5 BOUNDARY SCAN TESTING (BST)

Boundary Scan Testing (BST) is the method of controlling and observing the boundary pins of the JTAG compliant device, like those of the PIC32MX Family, utilizing software control. BST can be used to test connectivity between devices by daisy-chaining JTAG compliant devices to form a single scan chain. Several scan chains can exist on a PCB to form multiple scan chains. These multiple scan chains can then be driven simultaneously to test many components in parallel. Scan chains can contain both JTAG compliant devices and non-JTAG compliant devices.

A key advantage of BST is that it can be implemented without physical test probes; all that is needed is a 4-wire interface and an appropriate test platform. Since JTAG boundary scan has been available for many years, many software tools exist for testing scan chains without the need for extensive physical probing. The main drawback to BST is that it can only evaluate digital signals and circuit continuity; it cannot measure input or output voltage levels or currents.

27.2.5.1 Related JTAG Files

To implement BST, all JTAG test tools will require a Boundary Scan Description Language (BSDL) file. BSDL is a subset of VHDL (VHSIC Hardware Description Language), and is described as part of IEEE. 1149.1. The device-specific BSDL file describes how the standard is implemented on a particular device and how it operates.

The BSDL file for a particular device includes the following:

- The pinout and package configuration for the particular device
- The physical location of the TAP pins
- The Device ID register and the device ID
- The length of the Instruction Register
- The supported BST instructions and their binary codes
- The length and structure of the Boundary Scan register
- The boundary scan cell definition

Device-specific BSDL files are available at Microchip's web site, www.microchip.com.

The name for each BSDL file is the device name and silicon revision–for example, PIC32MX Family 320F128L_A2.BSD is the BSDL file for PIC32MX Family 320F128L, silicon revision A2.

27.3 Interrupts

Programming and debugging operations are not performed during code execution and are therefore not affected by interrupts. Trace operations will report the change in code execution when a interrupt occurs but the trace controller is not affected by interrupts.

27.4 I/O Pins

In order to interface the numerous programming and debugging option available and still provide peripheral access to the pins, the pins are multiplexed with peripherals. Table 27-7 describes the function of the programming and debug related pins.

TABLE 27-7: PROGRAMMING AND DEBUGGING PIN FUNCTIONS

Pin Name	Function				Description
	Program Mode	Debug Mode	Trace Mode	Boundary Scan Mode	
$\overline{MCLR}$	$\overline{MCLR}$	$\overline{MCLR}$	$\overline{MCLR}$	$\overline{MCLR}$	Master Clear, used to enter ICSP™ mode and to override JTAGEN (DDPCON<3>)
PGC1	PGC1/ Alternate	PGC1/ Alternate	PGC1/ Alternate	Alternate	ICSP clock, determined by ICESEL Configuration bit (DEVCFG0<3>)
PGD1	PGD1/ Alternate	PGD1/ Alternate	PGD1/ Alternate	Alternate	ICSP data, determined by ICESEL (DEVCFG0<3>) and DEBUG Configuration bits (DEVCFG0<1:0>)
PGC2	PGC2/ Alternate	PGC2/ Alternate	PGC2/ Alternate	Alternate	Alternate ICSP clock, determined by ICESEL (DEVCFG0<3>) and DEBUG Configuration bits (DEVCFG0<1:0>)
PGD2	PGD2/ Alternate	PGD2/ Alternate	PGD2/ Alternate	Alternate	Alternate ICSP data, determined by ICESEL (DEVCFG0<3>) and DEBUG Configuration bits (DEVCFG0<1:0>)
TCK	TCK	TCK	TCK	Alternate	JTAG clock, determined by JTAGEN control bit (DDPCON<3>)
TDO	TDO	TDO	TDO	Alternate	JTAG data out, determined by JTAGEN control bit (DDPCON<3>)
TDI	TDI	TDI	TDI	Alternate	JTAG data in, determined by JTAGEN control bit (DDPCON<3>)
TMS	TMS	TMS	TMS	Alternate	JTAG test mode select, determined by JTAGEN control bit (DDPCON<3>)
TRCLK	Alternate	Alternate	TRCLK	Alternate	Trace clock, determined by TROEN control bit (DDPCON<2>)
TRD0	Alternate	Alternate	TRD0	Alternate	Trace data, determined by TROEN control bit (DDPCON<2>)
TRD1	Alternate	Alternate	TRD1	Alternate	Trace data, determined by TROEN control bit (DDPCON<2>)
TRD2	Alternate	Alternate	TRD2	Alternate	Trace data, determined by TROEN control bit (DDPCON<2>)
TRD3	Alternate	Alternate	TRD3	Alternate	Trace data, determined by TROEN control bit (DDPCON<2>)

NOTES:

28.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C18 and MPLAB C30 C Compilers
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB ASM30 Assembler/Linker/Library
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD 2
- Device Programmers
  - PICSTART® Plus Development Programmer
  - MPLAB PM3 Device Programmer
  - PICkit™ 2 Development Programmer
- Low-Cost Demonstration and Development Boards and Evaluation Kits

28.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16-bit microcontroller market. The MPLAB IDE is a Windows® operating system-based application that contains:

- A single graphical interface to all debugging tools
  - Simulator
  - Programmer (sold separately)
  - Emulator (sold separately)
  - In-Circuit Debugger (sold separately)
- A full-featured editor with color-coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High-level source code debugging
- Visual device initializer for easy register initialization
- Mouse over variable inspection
- Drag and drop variables from source to watch windows
- Extensive on-line help
- Integration of select third party tools, such as HI-TECH Software C Compilers and IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either assembly or C)
- One touch assemble (or compile) and download to PIC MCU emulator and simulator tools (automatically updates all project information)
- Debug using:
  - Source files (assembly or C)
  - Mixed assembly and C
  - Machine code

MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost-effective simulators, through low-cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increased flexibility and power.

28.2 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for all PIC MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multi-purpose source files
- Directives that allow complete control over the assembly process

28.3 MPLAB C18 and MPLAB C30 C Compilers

The MPLAB C18 and MPLAB C30 Code Development Systems are complete ANSI C compilers for Microchip's PIC18 and PIC24 families of microcontrollers and the dsPIC30 and dsPIC33 family of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use not found with other compilers.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

28.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler and the MPLAB C18 C Compiler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

28.5 MPLAB ASM30 Assembler, Linker and Librarian

MPLAB ASM30 Assembler produces relocatable machine code from symbolic assembly language for dsPIC30F devices. MPLAB C30 C Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire dsPIC30F instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

28.6 MPLAB SIM Software Simulator

The MPLAB SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC® DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB SIM Software Simulator fully supports symbolic debugging using the MPLAB C18 and MPLAB C30 C Compilers, and the MPASM and MPLAB ASM30 Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

28.7 MPLAB ICE 2000 High-Performance In-Circuit Emulator

The MPLAB ICE 2000 In-Circuit Emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PIC microcontrollers. Software control of the MPLAB ICE 2000 In-Circuit Emulator is advanced by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The architecture of the MPLAB ICE 2000 In-Circuit Emulator allows expansion to support new PIC microcontrollers.

The MPLAB ICE 2000 In-Circuit Emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft® Windows® 32-bit operating system were chosen to best make these features available in a simple, unified application.

28.8 MPLAB REAL ICE In-Circuit Emulator System

MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs PIC® Flash MCUs and dsPIC® Flash DSCs with the easy-to-use, powerful graphical user interface of the MPLAB Integrated Development Environment (IDE), included with each kit.

The MPLAB REAL ICE probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with the popular MPLAB ICD 2 system (RJ11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

MPLAB REAL ICE is field upgradeable through future firmware downloads in MPLAB IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added, such as software breakpoints and assembly code trace. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, real-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

28.9 MPLAB ICD 2 In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD 2, is a powerful, low-cost, run-time development tool, connecting to the host PC via an RS-232 or high-speed USB interface. This tool is based on the Flash PIC MCUs and can be used to develop for these and other PIC MCUs and dsPIC DSCs. The MPLAB ICD 2 utilizes the in-circuit debugging capability built into the Flash devices. This feature, along with Microchip's In-Circuit Serial Programming™ (ICSP™) protocol, offers cost-effective, in-circuit Flash debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by setting breakpoints, single stepping and watching variables, and CPU status and peripheral registers. Running at full speed enables testing hardware and applications in real time. MPLAB ICD 2 also serves as a development programmer for selected PIC devices.

28.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at V_{DDMIN} and V_{DDMAX} for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages and a modular, detachable socket assembly to support various package types. The ICSP™ cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices and incorporates an SD/MMC card for file storage and secure data applications.

28.11 PICSTART Plus Development Programmer

The PICSTART Plus Development Programmer is an easy-to-use, low-cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient. The PICSTART Plus Development Programmer supports most PIC devices in DIP packages up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus Development Programmer is CE compliant.

28.12 PICkit 2 Development Programmer

The PICkit™ 2 Development Programmer is a low-cost programmer and selected Flash device debugger with an easy-to-use interface for programming many of Microchip's baseline, mid-range and PIC18F families of Flash memory microcontrollers. The PICkit 2 Starter Kit includes a prototyping development board, twelve sequential lessons, software and HI-TECH's PICC™ Lite C compiler, and is designed to help get up to speed quickly using PIC® microcontrollers. The kit provides everything needed to program, evaluate and develop applications using Microchip's powerful, mid-range Flash memory family of microcontrollers.

28.13 Demonstration, Development and Evaluation Boards

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

29.0 ELECTRICAL CHARACTERISTICS

This section provides an overview of PIC32MX Family electrical characteristics. Additional information will be provided in future revisions of this document as it becomes available.

Absolute maximum ratings for the PIC32MX 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.

Absolute Maximum Ratings(1)

Ambient temperature under bias...-40°C to +85°C
Storage temperature ....-65°C to +150°C
Voltage on VDD with respect to VSS ....-0.3V to +4.0V
Voltage on any combined analog and digital pin and $\overline{\text{MCLR}}$, with respect to VSS ....-0.3V to (VDD + 0.3V)
Voltage on any digital only pin with respect to VSS ....-0.3V to +5.5V
Voltage on VDDCORE with respect to VSS ....-0.3V to 2.0V
Maximum current out of VSS pin ....TBD mA
Maximum current into VDD pin **(Note 2)**...TBD mA
Maximum output current sunk by any I/O pin **(Note 3)** ....TBD mA
Maximum output current sourced by any I/O pin **(Note 3)** ....TBD mA
Maximum current sunk by all ports ....TBD mA
Maximum current sourced by all ports **(Note 2)**......................................TBD mA

Legend: TBD = To Be Determined

Note 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 Table 29-2).

3: Exceptions are CLKOUT, which is able to sink/source 25 mA, and the VREF+, VREF-, SCLx, SDAx, PGCx and PGDx pins, which are able to sink/source 12 mA.

29.1 DC Characteristics

TABLE 29-1: OPERATING MIPS VS. VOLTAGE

Characteristic	V_{DD} Range (in Volts)	Temp Range (in °C)	Max Frequency
			PIC32MX Family
DC5	2.5-3.6V	-40°C to +85°C	72 MHz[1]

Note 1: 20 MHz maximum for PIC32MX300 family variants.

TABLE 29-2: THERMAL OPERATING CONDITIONS

Rating	Symbol	Min	Typ	Max	Unit
PIC32MX Family					
Operating Junction Temperature Range	T_J	-40	—	+125	°C
Operating Ambient Temperature Range	T_A	-40	—	+85	°C
Power Dissipation: Internal Chip Power Dissipation: $P_{INT} = V_{DD} \times (I_{DD} – S\ I_{OH})$ I/O Pin Power Dissipation: $I/O = S\ (\{V_{DD} – V_{OH}\} \times I_{OH}) + S\ (V_{OL} \times I_{OL})$	P_D	$P_{INT} + P_{I/O}$			W
Maximum Allowed Power Dissipation	P_{DMAX}	$(T_J – T_A)/\theta_{JA}$			W

TABLE 29-3: THERMAL PACKAGING CHARACTERISTICS

Characteristic	Symbol	Typ	Max	Unit	Notes
Package Thermal Resistance, 100-Pin TQFP (12x12x1 mm)	θ_{JA}	52.3	—	°C/W	**1**
Package Thermal Resistance, 64-Pin TQFP (10x10x1 mm)	θ_{JA}	38.3	—	°C/W	**1**

Note 1: Junction to ambient thermal resistance, Theta-JA (θ_{JA}) numbers are achieved by package simulations.

TABLE 29-4: DC TEMPERATURE AND VOLTAGE SPECIFICATIONS

DC CHARACTERISTICS			**Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ[1]**	**Max**	**Units**	**Conditions**
Operating Voltage							
DC10	**Supply Voltage**						
	V_{DD}		2.5	—	3.6	V	
DC12	V_{DR}	**RAM Data Retention Voltage[2]**	—	TBD	—	V	
DC16	V_{POR}	**V_{DD} Start Voltage** to Ensure Internal Power-on Reset Signal	—	V_{SS}	—	V	
DC17	S_{VDD}	**V_{DD} Rise Rate** to Ensure Internal Power-on Reset Signal	0.05	—	—	V/ms	

Legend: TBD = To Be Determined

Note 1: Data in "Typ" column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

2: This is the limit to which V_{DD} can be lowered without losing RAM data.

TABLE 29-5: DC CHARACTERISTICS: OPERATING CURRENT (I_{DD})(1)

DC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial			
Parameter No.	Typical(2)	Max	Units	Conditions		
Operating Current (I_{DD})(3)						
DC20	—	TBD	mA		2.5V	4 MHz
DC20a	11	—	mA		—	
DC20b	—	TBD	mA		3.6V	
DC21	—	TBD	mA		2.5V	20 MHz
DC21a	25	—	mA		—	
DC21b	—	TBD	mA		3.6V	
DC22	—	TBD	mA		2.5V	60 MHz
DC22a	56	—	mA		—	
DC22b	—	TBD	mA		3.6V	
DC23	—	TBD	mA		2.5V	72 MHz
DC23a	64	—	mA		—	
DC23b	—	TBD	mA		3.6V	
DC24	—	TBD	µA	-40°C	2.5V	LPRC (31 kHz)
DC24a	—	TBD	µA	+25°C		
DC24b	—	TBD	µA	+85°C		
DC25	TBD	—	µA	-40°C	3.3V	
DC25a	230	—	µA	+25°C		
DC25b	TBD	—	µA	+85°C		
DC26	—	TBD	µA	-40°C	3.6V	
DC26a	—	TBD	µA	+25°C		
DC26b	—	TBD	µA	+85°C		

Legend: TBD = To Be Determined

Note 1: A device's I_{DD} supply current is mainly a function of the operating voltage and frequency. Other factors, such as PBCLK (Peripheral Bus Clock) frequency, I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.

2: Data in "Typical" column is at 3.3V, 25°C at specified operating frequency unless otherwise stated. Parameters are for design guidance only and are not tested.

3: The test conditions for I_{DD} measurements are as follows: Oscillator mode = EC+PLL with OSC1 driven by external square wave from rail to rail and PBCLK divisor = 1:8. CPU, SRAM, program memory and data memory are operational with CPU Wait states disabled, Flash memory Wait states = 7 and program cache disabled. Only digital peripheral modules are enabled (ON bit = `1`) and being clocked; however, not accessed. WDT and FSCM are disabled. All I/O pins are configured as inputs and pulled to V_{SS}. $\overline{MCLR}$ = V_{DD}.

TABLE 29-6: DC CHARACTERISTICS: IDLE CURRENT (IIDLE)

DC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for Industrial			
Parameter No.	**Typical(1)**	**Max**	**Units**	**Conditions**		
Idle Current (IIDLE): Core OFF Clock ON Base Current(2)						
DC30	—	TBD	mA		2.5V	4 MHz
DC30a	3	—	mA		—	
DC30b	—	TBD	mA		3.6V	
DC31	—	TBD	mA		2.5V	40 MHz
DC31a	11	—	mA		—	
DC31b	—	TBD	mA		3.6V	
DC32	—	TBD	mA		2.5V	60 MHz
DC32a	31	—	mA		—	
DC32b	—	TBD	mA		3.6V	
DC33	—	TBD	mA		2.5V	72 MHz
DC33a	36	—	mA		—	
DC33b	—	TBD	mA		3.6V	
DC34	—	TBD	µA	-40°C	2.5V	LPRC (31 kHz)
DC34a	—	TBD	µA	+25°C		
DC34b	—	TBD	µA	+85°C		
DC35	TBD	—	µA	-40°C	3.3V	
DC35a	200	—	µA	+25°C		
DC35b	TBD	—	µA	+85°C		
DC36	—	TBD	µA	-40°C	3.6V	
DC36a	—	TBD	µA	+25°C		
DC36b	—	TBD	µA	+85°C		

Legend: TBD = To Be Determined

Note 1: Data in “Typical” column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

2: The test conditions for base IIDLE current measurements are as follows: System clock is enabled and PBCLK divisor = 1:8. CPU in IDLE mode (CPU core halted). Only digital peripheral modules are enabled (ON bit = 1) and being clocked. WDT and FSCM are disabled. All I/O pins are configured as inputs and pulled to VSS. $\overline{\text{MCLR}}$ = VDD.

TABLE 29-7: DC CHARACTERISTICS: POWER-DOWN CURRENT (IPD)

DC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for Industrial			
Parameter No.	**Typical(1)**	**Max**	**Units**	**Conditions**		
Power-Down Current (IPD)(2)						
DC40	—	TBD	µA	-40°C	2.5V	Base Power-Down Current
DC40a	—	TBD	µA	+25°C		
DC40b	—	TBD	µA	+85°C		
DC40c	50	—	µA			
DC40d	—	TBD	µA	-40°C	3.6V	
DC40e	—	TBD	µA	+25°C		
DC40f	—	TBD	µA	+85°C		
Module Differential Current						
DC41	—	TBD	µA	-40°C	2.5V	Watchdog Timer Current: ΔIWDT(3)
DC41a	—	TBD	µA	+25°C		
DC41b	—	TBD	µA	+85°C		
DC41c	10	—	µA			
DC41d	—	TBD	µA	-40°C	3.6V	
DC41e	—	TBD	µA	+25°C		
DC41f	—	TBD	µA	+85°C		
DC42	—	TBD	µA	-40°C	2.5V	RTCC + Timer1: ΔIRTCC(3,4)
DC42a	—	TBD	µA	+25°C		
DC42b	—	TBD	µA	+85°C		
DC42c	22.5	—	µA			
DC42e	—	TBD	µA	-40°C	3.6V	
DC42f	—	TBD	µA	+25°C		
DC42g	—	TBD	µA	+85°C		
DC42	—	TBD	µA	-40°C	2.5V	ADC: ΔIADC(3,5)
DC42a	—	TBD	µA	+25°C		
DC42b	—	TBD	µA	+85°C		
DC42c	880	—	µA			
DC42e	—	TBD	µA	-40°C	3.6V	
DC42f	—	TBD	µA	+25°C		
DC42g	—	TBD	µA	+85°C		

Legend: TBD = To Be Determined

Note 1: Data in the Typical column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

2: Base IPD is measured with all peripherals (ON bit = 0) and clocks shut down. All I/Os are configured as outputs and pulled low. WDT, etc., are turned off.

3: The Δ current is the additional current consumed when the module is enabled. This current should be added to the base IPD current.

4: Test conditions for RTCC module differential current are as follows: Timer1 is enabled (T1CON.ON bit = `1`), secondary oscillator is enabled (OSCCON.SLPEN = `1`), pin SOSCI is driven by external square wave from rail to rail.

5: Test conditions for ADC module differential current are as follows: Internal ADC RC oscillator enabled.

TABLE 29-8: DC CHARACTERISTICS: I/O PIN INPUT SPECIFICATIONS

DC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for Industrial				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ(1)**	**Max**	**Units**	**Conditions**
	VIL	**Input Low Voltage**					
DI10		I/O pins:					
		with TTL Buffer	VSS	—	0.15 VDD	VSS	
		with Schmitt Trigger Buffer	VSS	—	0.2 VDD	VSS	
DI15		$\overline{\text{MCLR}}$	VSS	—	0.2 VDD	V	
DI16		OSC1 (XT mode)	VSS	—	0.2 VDD	V	
DI17		OSC1 (HS mode)	VSS	—	0.2 VDD	V	
DI18		SDAx, SCLx	VSS	—	0.3 VDD	V	SMBus disabled
DI19		SDAx, SCLx	VSS	—	0.8	V	SMBus enabled
	VIH	**Input High Voltage**					
DI20		I/O pins: with Analog Functions	0.8 VDD	—	VDD	V	
		Digital Only	0.8 VDD	—		V	
		with TTL Buffer	0.25VDD + 0.8V	—	5.5	VSS	
		with Schmitt Trigger Buffer	0.8 VDD	—	5.5	VSS	
DI25		$\overline{\text{MCLR}}$	0.8 VDD	—	VDD	V	
DI26		OSC1 (XT mode)	0.7 VDD	—	VDD	V	
DI27		OSC1 (HS mode)	0.7 VDD	—	VDD	V	
DI28		SDAx, SCLx	0.7 VDD	—	VDD	V	SMBus disabled
DI29		SDAx, SCLx	2.1	—	VDD	V	SMBus enabled, 2.5V ≤ VPIN ≤ VDD
DI30	ICNPU	**CNxx Pull up Current**	50	250	400	μA	VDD = 3.3V, VPIN = VSS
	IIL	**Input Leakage Current(2,3)**					
DI50		I/O Ports	—	—	±1	μA	VSS ≤ VPIN ≤ VDD, Pin at high-impedance
DI51		Analog Input Pins	—	—	±1	μA	VSS ≤ VPIN ≤ VDD, Pin at high-impedance
DI55		$\overline{\text{MCLR}}$	—	—	±1	μA	VSS ≤ VPIN ≤ VDD
DI56		OSC1	—	—	±1	μA	VSS ≤ VPIN ≤ VDD, XT and HS modes

Note 1: Data in “Typ” column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

2: The leakage current on the $\overline{\text{MCLR}}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as current sourced by the pin.

TABLE 29-9: DC CHARACTERISTICS: I/O PIN OUTPUT SPECIFICATIONS

DC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for Industrial				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ(1)**	**Max**	**Units**	**Conditions**
	VOL	**Output Low Voltage**					
DO10		I/O Ports	—	—	0.4	V	IOL = 9 mA, VDD = 3.6V
			—	—	0.4	V	IOL = TBD mA, VDD = 2.5V
DO16		OSC2/CLKO	—	—	0.4	V	IOL = TBD mA, VDD = 3.6V
			—	—	0.4	V	IOL = TBD mA, VDD = 2.5V
	VOH	**Output High Voltage**					
DO20		I/O Ports	2.4	—	—	V	IOH = -15 mA, VDD = 3.6V
			1.4	—	—	V	IOH = -TBD mA, VDD = 2.5V
DO26		OSC2/CLKO	2.4	—	—	V	IOH = -TBD mA, VDD = 3.6V
			1.4	—	—	V	IOH = -TBD mA, VDD = 2.5V

Legend: TBD = To Be Determined

Note 1: Data in “Typ” column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

TABLE 29-10: DC CHARACTERISTICS: PROGRAM MEMORY

DC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for Industrial				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ(1)**	**Max**	**Units**	**Conditions**
		Program Flash Memory					
D130	EP	Cell Endurance	1000	—	—	E/W	-40°C to +85°C
D131	VPR	VDD for Read	VMIN	—	3.6	V	VMIN = Minimum operating voltage
D132B	VPEW	VDD for Erase or Write	3.0	—	3.6	V	
D134	TRETD	Characteristic Retention	20	—	—	Year	Provided no other specifications are violated
D135	IDDP	Supply Current during Programming	—	10	—	mA	
	TWW	Word Write Cycle Time	20	—	40	µs	
D136	TRW	Row Write Cycle Time (128 words per row)	3	TBD	—	ms	
D137	TPE	Page Erase Cycle Time	20	—	40	ms	
	TCE	Chip Erase Cycle Time	20	—	40	ms	

Legend: TBD = To Be Determined

Note 1: Data in “Typ” column is at 3.3V, 25°C unless otherwise stated.

TABLE 29-11: COMPARATOR SPECIFICATIONS

Operating Conditions: 2.5V < VDD < 3.6V, -40°C < TA < +85°C (unless otherwise stated)							
Param No.	**Sym**	**Characteristics**	**Min**	**Typ**	**Max**	**Units**	**Comments**
D300	VIOFF	Input Offset Voltage	—	±7.5	±15	mV	
D301	VICM	Input Common Mode Voltage*	0	—	VDD	V	
D302	CMRR	Common Mode Rejection Ratio*	55	—	—	dB	
300	TRESP	Response Time*(1)	—	150	400	ns	
301	TMC2OV	Comparator Mode Change to Output Valid*	—	—	10	µs	

* These parameters are characterized but not tested.

Note 1: Response time measured with one comparator input at (VDD – 1.5)/2, while the other input transitions from VSS to VDD.

TABLE 29-12: VOLTAGE REFERENCE SPECIFICATIONS

Operating Conditions: 2.5V < VDD < 3.6V, -40°C < TA < +85°C (unless otherwise stated)							
Param No.	**Sym**	**Characteristics**	**Min**	**Typ**	**Max**	**Units**	**Comments**
D310	VRES	Resolution	VDD/24	—	VDD/32	LSb	
D311	VRAA	Absolute Accuracy	—	—	1/2	LSb	
D312	VROZ	Output Impedance	—	TBD	—	Ω	
310	TSET	Settling Time(1)	—	—	10	µs	

Legend: TBD = To Be Determined

Note 1: Settling time measured while CVRR = 1 and CVR3:CVR0 transitions from '0000' to '1111'.

TABLE 29-13: INTERNAL VOLTAGE REGULATOR SPECIFICATIONS

Operating Conditions: -40°C < TA < +85°C (unless otherwise stated)							
Param No.	**Symbol**	**Characteristics**	**Min**	**Typ**	**Max**	**Units**	**Comments**
	VRGOUT	Regulator Output Voltage	1.62	1.80	1.98	V	
	CEFC	External Filter Capacitor Value	TBD	10	—	µF	Capacitor must be low series resistance
	TPWRT		—	64	—	ms	ENVREG = 0

Legend: TBD = To Be Determined

29.2 AC Characteristics and Timing Parameters

The information contained in this section defines PIC32MX Family AC characteristics and timing parameters.

TABLE 29-14: AC CHARACTERISTICS

AC CHARACTERISTICS	**Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for Industrial Operating voltage VDD range as described in **Section 29.0 "Electrical Characteristics"**.

FIGURE 29-1: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS

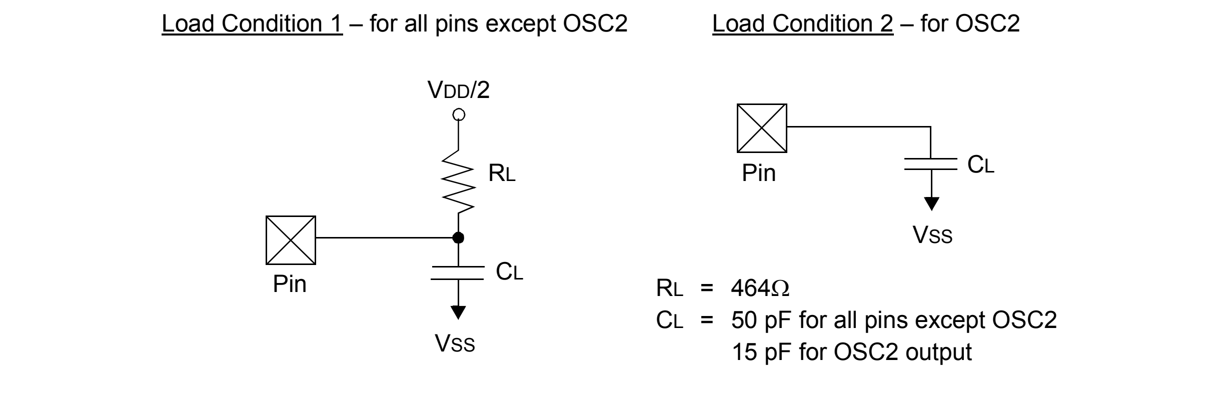


TABLE 29-15: CAPACITIVE LOADING REQUIREMENTS ON OUTPUT PINS

Param No.	Symbol	Characteristic	Min	Typ(1)	Max	Units	Conditions
DO50	COSC2	OSC2/SOSC2 pin	—	—	15	pF	In XT and HS modes when external clock is used to drive OSC1
DO56	CIO	All I/O pins and OSC2	—	—	50	pF	EC mode
DO58	CB	SCLx, SDAx	—	—	400	pF	In I^{2}C™ mode

Note 1: Data in "Typ" column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

FIGURE 29-2: EXTERNAL CLOCK TIMING

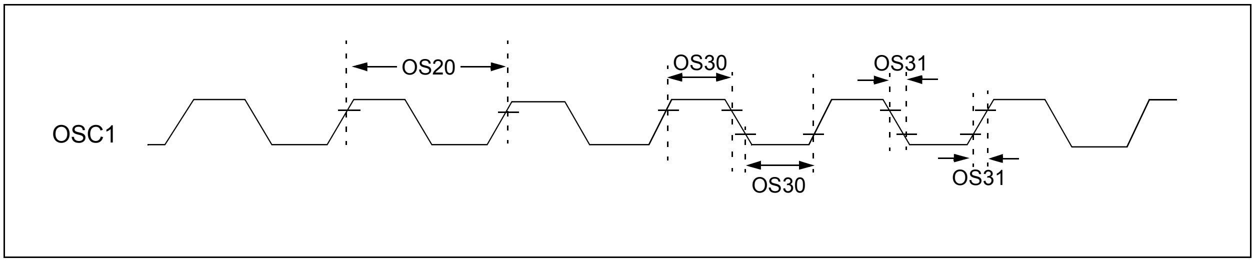


TABLE 29-16: EXTERNAL CLOCK TIMING REQUIREMENTS

AC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for Industrial				
Param No.	Symb	Characteristic	Min	Typ(1)	Max	Units	Conditions
OS10	FOSC	External CLKI Frequency (External clocks allowed only in EC and ECPLL modes)	DC 4	— —	50(3) 50	MHz MHz	EC ECPLL(4)
		Oscillator Crystal Frequency	3 4 10 10 32	— — — — 32.768	10 1 40(3) 40(3) 100	MHz MHz MHz MHz kHz	XT XTPLL(4) HS HSPLL(4) SOSC
OS20	TOSC	TOSC = 1/FOSC = TCY(2)	—	—	—	—	See parameter OS10 for FOSC value
OS30	TosL, TosH	External Clock In (OSC1) High or Low Time	0.45 x TOSC	—	—	ns	EC
OS31	TosR, TosF	External Clock In (OSC1) Rise or Fall Time	—	—	0.05 x TOSC	ns	EC

Note 1: Data in “Typ” column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

2: Instruction cycle period (TCY) equals the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min.” values with an external clock applied to the OSC1/CLKI pin.

3: 20 MHz maximum for PIC32MX300 family devices.

4: PLL input requirements: 4 MHZ <= FOSC <= 5 MHZ (use PLL prescaler to reduce FOSC).

 Advance Information

TABLE 29-17: PLL CLOCK TIMING SPECIFICATIONS (V_{DD} = 2.5V TO 3.6V)

AC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for Industrial				
Param No.	**Symbol**	**Characteristic(1)**	**Min**	**Typ(2)**	**Max**	**Units**	**Conditions**
OS50	FPLLI	PLL Voltage Controlled Oscillator (VCO) Input Frequency Range	4	—	5	MHz	ECPLL, HSPLL, MSPLL, FRCPLL modes
OS51	FSYS	On-Chip VCO System Frequency	60	—	230	MHz	
OS52	TLOC	PLL Start-up Time (Lock Time)	TBD	TBD	24	μs	
OS53	DCLK	CLKO Stability (Jitter)	TBD	+/-1	TBD	%	Measured over 100 ms period

Legend: TBD = To Be Determined

Note 1: These parameters are characterized but not tested in manufacturing.

2: Data in "Typ" column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

TABLE 29-18: INTERNAL FRC ACCURACY

AC CHARACTERISTICS		Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
Param No.	**Characteristic**	**Min**	**Typ**	**Max**	**Units**	**Conditions**	
Internal FRC Accuracy @ 8.00 MHz(1)							
F20	FRC	-2	—	+2	%	+25°C	VDD = 3.0 to 3.6V
		-5.0	—	+5.0	%	-40°C ≤ TA ≤ +85°C	VDD = 3.0 to 3.6V

Note 1: Frequency calibrated at 25°C and 3.3V. TUN bits can be used to compensate for temperature drift.

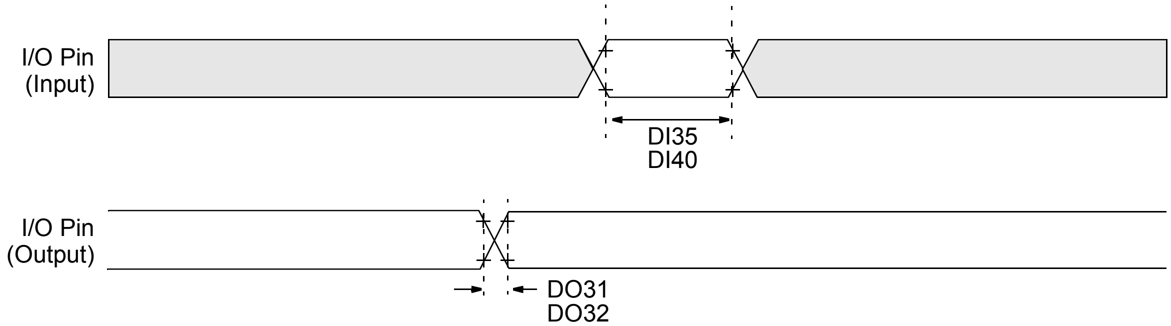
TABLE 29-19: INTERNAL RC ACCURACY

AC CHARACTERISTICS		Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for Industrial					
Param No.	**Characteristic**	**Min**	**Typ**	**Max**	**Units**	**Conditions**	
LPRC @ 31.25 kHz(1)							
F21		TBD	—	TBD	%	+25°C	VDD = 2.5 to 3.6V
		-15	—	+15	%	-40°C ≤ TA ≤ +85°C	VDD = 2.5 to 3.6V

Legend: TBD = To Be Determined

Note 1: Change of LPRC frequency as VDD changes.

FIGURE 29-3: CLKO AND I/O TIMING CHARACTERISTICS



Note: Refer to Figure 29-1 for load conditions.

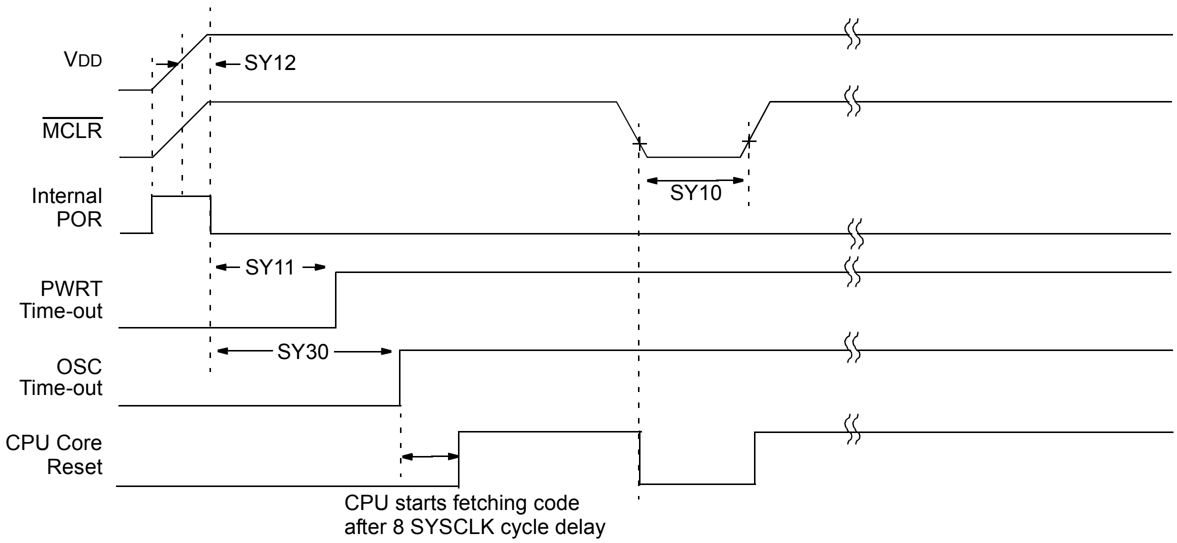
TABLE 29-20: CLKO AND I/O TIMING REQUIREMENTS

AC CHARACTERISTICS			**Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for Industrial				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ(1)**	**Max**	**Units**	**Conditions**
DO31	TIOR	Port Output Rise Time	—	10	20	ns	
DO32	TIOF	Port Output Fall Time	—	10	20	ns	
DI35	TINP	INTx Pin High or Low Time	10	—	—	ns	
DI40	TRBP	CNx High or Low Time (input)	2	—	—	TSYSCLK	

Note 1: Data in “Typ” column is at 3.3V, 25°C unless otherwise stated.

2: Measurements are taken in EC mode. The CLKO signal is measured on the OSC2 pin.

FIGURE 29-4: RESETS TIMING CHARACTERISTICS



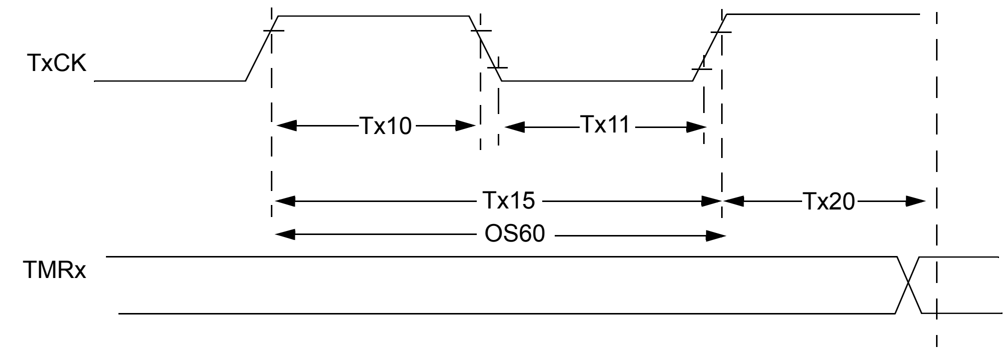
Note: Refer to Figure 29-1 for load conditions.

TABLE 29-21: RESETS TIMING

AC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C				
Param No.	**Symbol**	**Characteristic[1]**	**Min**	**Typ[2]**	**Max**	**Units**	**Conditions**
SY10	TMCL	$\overline{\text{MCLR}}$ Pulse Width (low)	2	—	—	μs	-40°C to +85°C
SY11	TPWRT	Power-up Timer Period	48	64	80	ms	-40°C to +85°C
SY12	TPOR	Power-on Reset Delay	1	5	10	μs	-40°C to +85°C
SY30	TOST	Oscillator Start-up Timer Period	—	1024 TOSC	—	—	TOSC = OSC1 period

Note 1: These parameters are characterized but not tested in manufacturing.
2: Data in “Typ” column is at 3.3V, 25°C unless otherwise stated. Characterized by design but not tested.

FIGURE 29-5: TIMER1, 2, 3, 4, 5 EXTERNAL CLOCK TIMING CHARACTERISTICS



Note: Refer to Figure 29-1 for load conditions.

TABLE 29-22: TIMER1 EXTERNAL CLOCK TIMING REQUIREMENTS[(1)]

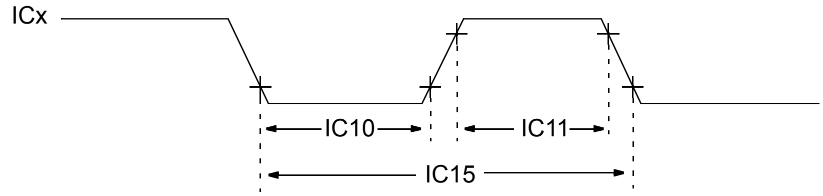
AC CHARACTERISTICS				**Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C				
Param No.	**Symbol**	**Characteristic**		**Min**	**Typ**	**Max**	**Units**	**Conditions**
TA10	TTXH	TxCK High Time	Synchronous, with prescaler	10	—	—	ns	Must also meet parameter TA15
			Asynchronous, with prescaler	10	—	—	ns	
			Asynchronous	10	—	—	ns	
TA11	TTXL	TxCK Low Time	Synchronous, with prescaler	10	—	—	ns	Must also meet parameter TA15
			Asynchronous, with prescaler	10	—	—	ns	
			Asynchronous	10	—	—	ns	
TA15	TTXP	TxCK Input Period	Synchronous, with prescaler	Greater of: 10 ns or (2 * TPB + 10)	—	—	ns	
			Asynchronous, with prescaler	Greater of: 10 ns or (2 * TPB + 10)/N	—	—	—	N = prescale value (1, 8, 64, 256)
			Asynchronous	10	—	—	ns	
OS60	Ft1	SOSC1/T1CK Oscillator Input Frequency Range (oscillator enabled by setting TCS bit (T1CON<1>))		32	—	100	kHz	
TA20	TCKEXTMRL	Delay from External TxCK Clock Edge to Timer Increment		—		10	ns	

Note 1: Timer1 is a Type A.

TABLE 29-23: TIMER2, 3, 4, 5 EXTERNAL CLOCK TIMING REQUIREMENTS

AC CHARACTERISTICS				Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C				
Param No.	**Symbol**	**Characteristic**		**Min**	**Typ**	**Max**	**Units**	**Conditions**
TB10	TtxH	TxCK High Time	Synchronous, no prescaler	Greater of: 10 ns or (TPB + 5)	—	—	ns	Must also meet parameter TB15
			Synchronous, with prescaler	20	—	—	ns	
TB11	TtxL	TxCK Low Time	Synchronous, no prescaler	Greater of: 10 ns or (TPB + 5)	—	—	ns	Must also meet parameter TB15
			Synchronous, with prescaler	10	—	—	ns	
TB15	TtxP	TxCK Input Period	Synchronous, no prescaler	Greater of: 10 ns or (2 * TPB + 10)	—	—	ns	N = prescale value (1, 2, 4, 8, 16, 32, 64, 256)
			Synchronous, with prescaler	Greater of: 10 ns or (2*TPB + 10)/N				
TB20	TCKEXTMRL	Delay from External TxCK Clock Edge to Timer Increment		—	—	5	ns	

FIGURE 29-6: INPUT CAPTURE (CAPx) TIMING CHARACTERISTICS



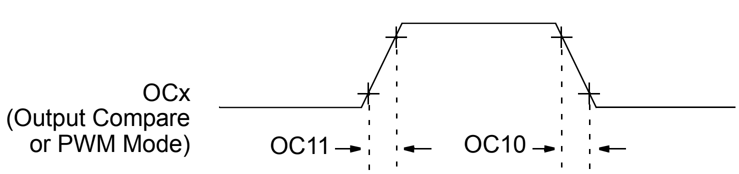
Note: Refer to Figure 29-1 for load conditions.

TABLE 29-24: INPUT CAPTURE MODULE TIMING REQUIREMENTS

AC CHARACTERISTICS				Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C			
Param No.	**Symbol**	**Characteristic[(1)]**		**Min**	**Max**	**Units**	**Conditions**
IC10	TccL	ICx Input Low Time	No prescaler	Greater of: 10 ns or (TPB + 5)	—	ns	
			With prescaler	10	—	ns	
IC11	TccH	ICx Input High Time	No prescaler	Greater of: 20 ns or (TPB + 5)	—	ns	
			With prescaler	10	—	ns	
IC15	TccP	ICx Input Period		Greater of: 10 ns or (2*TPB + 10)/N	—	ns	N = prescale value (1, 4, 16)

Note 1: These parameters are characterized but not tested in manufacturing.

FIGURE 29-7: OUTPUT COMPARE MODULE (OCx) TIMING CHARACTERISTICS



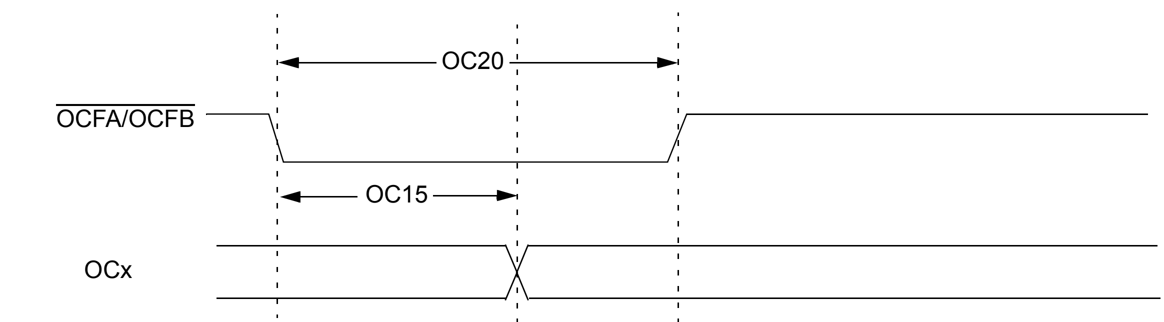
Note: Refer to Figure 29-1 for load conditions.

TABLE 29-25: OUTPUT COMPARE MODULE TIMING REQUIREMENTS

AC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C				
Param No.	Symbol	Characteristic(1)	Min	Typ(2)	Max	Units	Conditions
OC10	TccF	OCx Output Fall Time	—	—	10	ns	
OC11	TccR	OCx Output Rise Time	—	—	10	ns	

Note 1: These parameters are characterized but not tested in manufacturing.
2: Data in “Typ” column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

FIGURE 29-8: OC/PWM MODULE TIMING CHARACTERISTICS



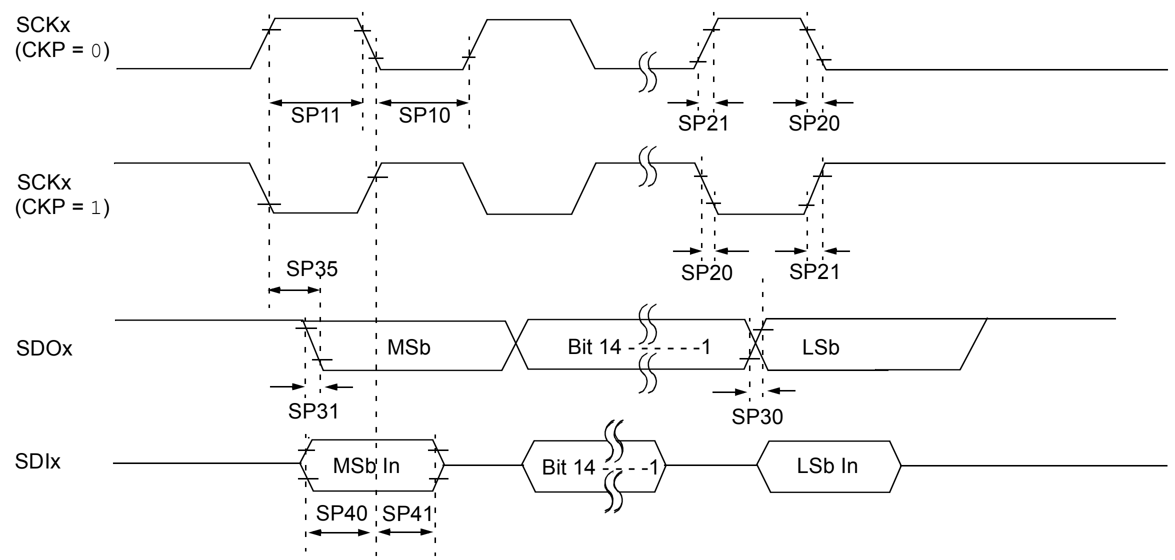
Note: Refer to Figure 29-1 for load conditions.

TABLE 29-26: SIMPLE OC/PWM MODE TIMING REQUIREMENTS

AC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C				
Param No.	Symbol	Characteristic(1)	Min	Typ(2)	Max	Units	Conditions
OC15	TFD	Fault Input to PWM I/O Change	—	—	25	ns	
OC20	TFLT	Fault Input Pulse Width	50	—	—	ns	

Note 1: These parameters are characterized but not tested in manufacturing.
2: Data in “Typ” column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

FIGURE 29-9: SPIx MODULE MASTER MODE (CKE = 0) TIMING CHARACTERISTICS



Note: Refer to Figure 29-1 for load conditions.

TABLE 29-27: SPIx MASTER MODE (CKE = 0) TIMING REQUIREMENTS

AC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C				
Param No.	**Symbol**	**Characteristic(1)**	**Min**	**Typ(2)**	**Max**	**Units**	**Conditions**
SP10	TscL	SCKx Output Low Time(3)	TSCK/2	—	—	ns	
SP11	TscH	SCKx Output High Time(3)	TSCK/2	—	—	ns	
SP20	TscF	SCKx Output Fall Time(4)	—	—	—	ns	
SP21	TscR	SCKx Output Rise Time(4)	—	—	—	ns	
SP30	TdoF	SDOx Data Output Fall Time(4)	—	TBD	TBD	ns	
SP31	TdoR	SDOx Data Output Rise Time(4)	—	TBD	TBD	ns	
SP35	TscH2doV, TscL2doV	SDOx Data Output Valid after SCKx Edge	—	—	TBD	ns	
SP40	TdiV2scH, TdiV2scL	Setup Time of SDIx Data Input to SCKx Edge	TBD	—		ns	
SP41	TscH2diL, TscL2diL	Hold Time of SDIx Data Input to SCKx Edge	TBD	—	—	ns	

Legend: TBD = To Be Determined

Note 1: These parameters are characterized but not tested in manufacturing.

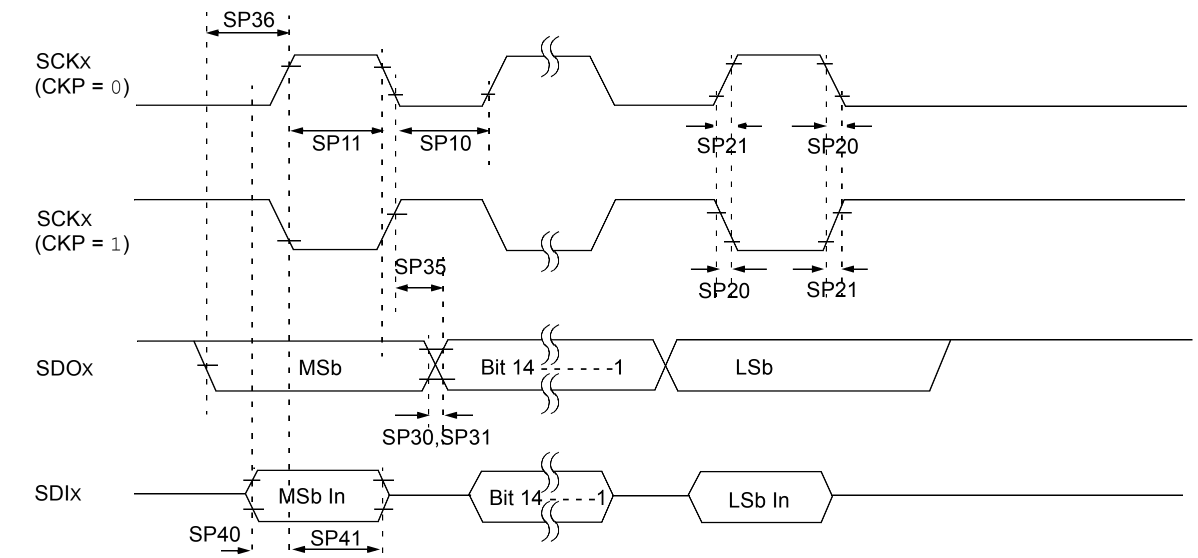
2: Data in “Typ” column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

3: The minimum clock period for SCKx is 20 ns. Therefore, the clock generated in Master mode must not violate this specification.

4: Assumes 50 pF load on all SPIx pins.

 Advance Information

FIGURE 29-10: SPIx MODULE MASTER MODE (CKE = 1) TIMING CHARACTERISTICS



Note: Refer to Figure 29-1 for load conditions.

TABLE 29-28: SPIx MODULE MASTER MODE (CKE = 1) TIMING REQUIREMENTS

AC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C				
Param No.	**Symbol**	**Characteristic(1)**	**Min**	**Typ(2)**	**Max**	**Units**	**Conditions**
SP10	TscL	SCKx Output Low Time(3)	TSCK/2	—	—	ns	
SP11	TscH	SCKx Output High Time(3)	TSCK/2	—	—	ns	
SP20	TscF	SCKx Output Fall Time(4)	—	—	—	ns	
SP21	TscR	SCKx Output Rise Time(4)	—	—	—	ns	
SP30	TdoF	SDOx Data Output Fall Time(4)	—	TBD	TBD	ns	
SP31	TdoR	SDOx Data Output Rise Time(4)	—	TBD	TBD	ns	
SP35	TscH2doV, TscL2doV	SDOx Data Output Valid after SCKx Edge	—	—	TBD	ns	
SP36	TdoV2sc, TdoV2scL	SDOx Data Output Setup to First SCKx Edge	TBD	—	—	ns	
SP40	TdiV2scH, TdiV2scL	Setup Time of SDIx Data Input to SCKx Edge	TBD	—	—	ns	
SP41	TscH2diL, TscL2diL	Hold Time of SDIx Data Input to SCKx Edge	TBD	—	—	ns	

Legend: TBD = To Be Determined

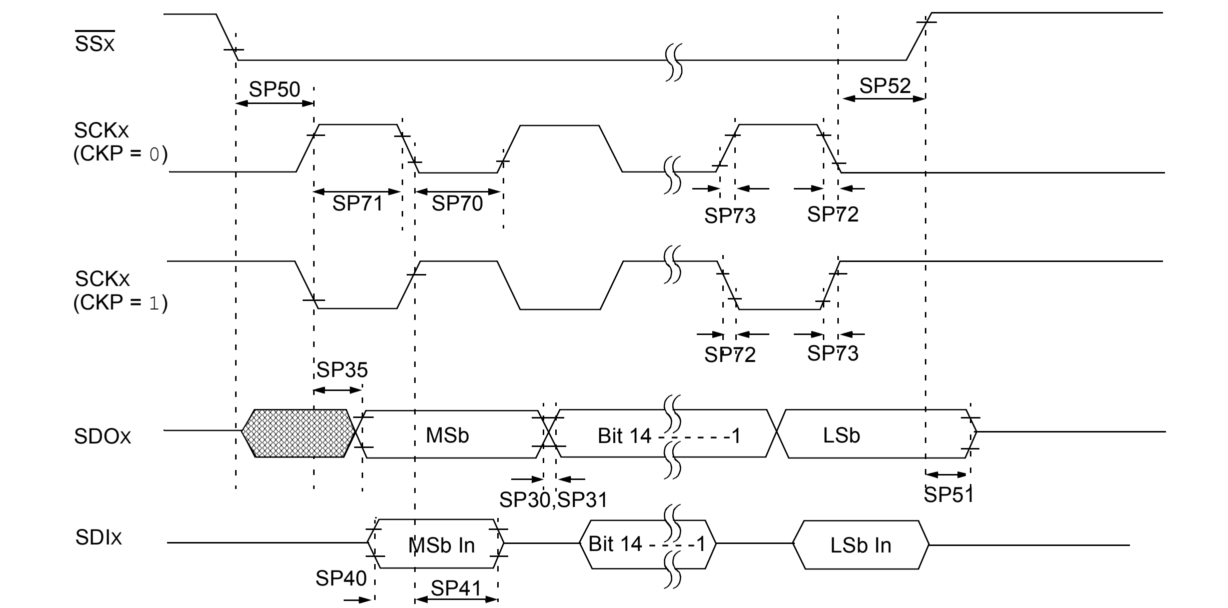
Note 1: These parameters are characterized but not tested in manufacturing.

2: Data in "Typ" column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

3: The minimum clock period for SCKx is 20 ns. Therefore, the clock generated in Master mode must not violate this specification.

4: Assumes 50 pF load on all SPIx pins.

FIGURE 29-11: SPIx MODULE SLAVE MODE (CKE = 0) TIMING CHARACTERISTICS



Note: Refer to Figure 29-1 for load conditions.

TABLE 29-29: SPIx MODULE SLAVE MODE (CKE = 0) TIMING REQUIREMENTS

AC CHARACTERISTICS			**Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C				
Param No.	**Symbol**	**Characteristic(1)**	**Min**	**Typ(2)**	**Max**	**Units**	**Conditions**
SP70	TscL	SCKx Input Low Time	TBD	—	—	ns	
SP71	TscH	SCKx Input High Time	TSCK/2	—	—	ns	
SP72	TscF	SCKx Input Fall Time(3)	—	TBD	TBD	ns	
SP73	TscR	SCKx Input Rise Time(3)	—	TBD	TBD	ns	
SP30	TdoF	SDOx Data Output Fall Time(3)	—	TBD	TBD	ns	
SP31	TdoR	SDOx Data Output Rise Time(3)	—	TBD	TBD	ns	
SP35	TscH2doV, TscL2doV	SDOx Data Output Valid after SCKx Edge	—	—	TBD	ns	
SP40	TdiV2scH, TdiV2scL	Setup Time of SDIx Data Input to SCKx Edge	TBD	—		ns	
SP41	TscH2diL, TscL2diL	Hold Time of SDIx Data Input to SCKx Edge	TBD	—	—	ns	
SP50	TssL2scH, TssL2scL	SSx ↓ to SCKx ↑ or SCKx Input	TBD	—	—	ns	
SP51	TssH2doZ	SSx ↑ to SDOx Output High-Impedance(3)	TBD	—	TBD	ns	
SP52	TscH2ssH TscL2ssH	SSx after SCKx Edge	TBD	—	—	ns	—

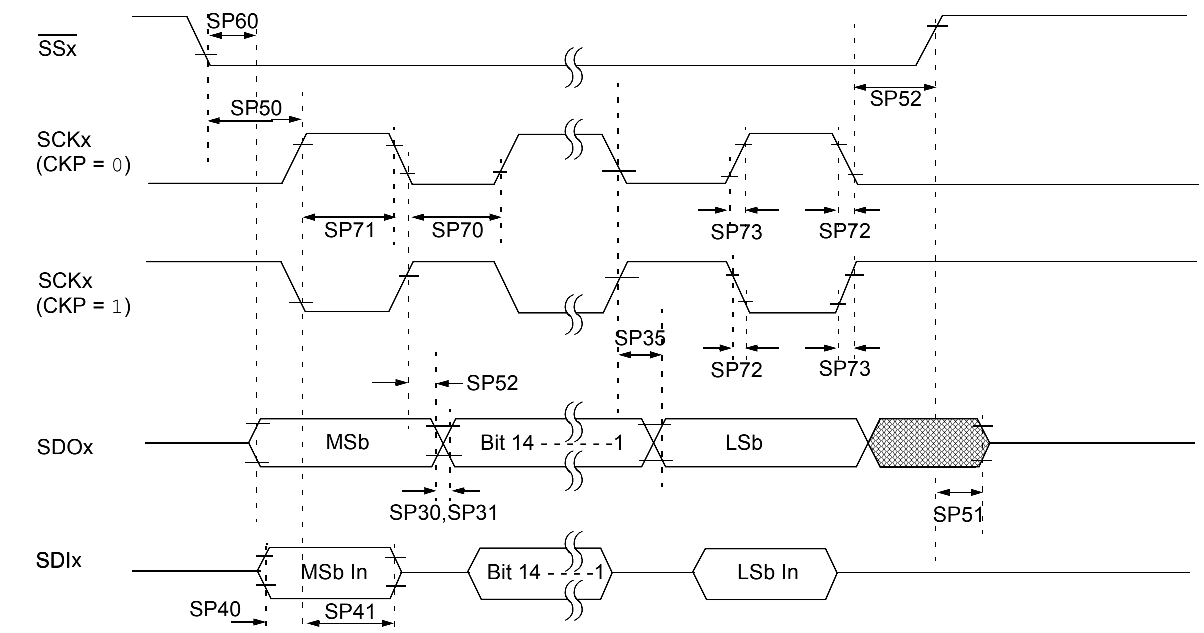
Legend: TBD = To Be Determined

Note 1: These parameters are characterized but not tested in manufacturing.

2: Data in "Typ" column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

3: Assumes 50 pF load on all SPIx pins.

FIGURE 29-12: SPIx MODULE SLAVE MODE (CKE = 1) TIMING CHARACTERISTICS



Note: Refer to Figure 29-1 for load conditions.

TABLE 29-30: SPIx MODULE SLAVE MODE (CKE = 1) TIMING REQUIREMENTS

AC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C				
Param No.	**Symbol**	**Characteristic(1)**	**Min**	**Typ(2)**	**Max**	**Units**	**Conditions**
SP70	TscL	SCKx Input Low Time	TBD	—	—	ns	
SP71	TscH	SCKx Input High Time	TSCK/2	—	—	ns	
SP72	TscF	SCKx Input Fall Time(3)	—	TBD	TBD	ns	
SP73	TscR	SCKx Input Rise Time(3)	—	TBD	TBD	ns	
SP30	TdoF	SDOx Data Output Fall Time(3)	—	TBD	TBD	ns	
SP31	TdoR	SDOx Data Output Rise Time(3)	—	TBD	TBD	ns	
SP35	TscH2doV, TscL2doV	SDOx Data Output Valid after SCKx Edge	—	—	TBD	ns	
SP40	TdiV2scH, TdiV2scL	Setup Time of SDIx Data Input to SCKx Edge	TBD	—		ns	
SP41	TscH2diL, TscL2diL	Hold Time of SDIx Data Input to SCKx Edge	TBD	—	—	ns	
SP50	TssL2scH, TssL2scL	$\overline{SSx}$ ↓ to SCKx ↓ or SCKx ↑ Input	TBD	—	—	ns	
SP51	TssH2doZ	$\overline{SSx}$ ↑ to SDOx Output High-Impedance(4)	TBD	—	TBD	ns	
SP52	TscH2ssH TscL2ssH	$\overline{SSx}$ ↑ after SCKx Edge	TBD	—	—	ns	
SP60	TssL2doV	SDOx Data Output Valid after $\overline{SSx}$ Edge	—	—	TBD	ns	

Legend: TBD = To Be Determined

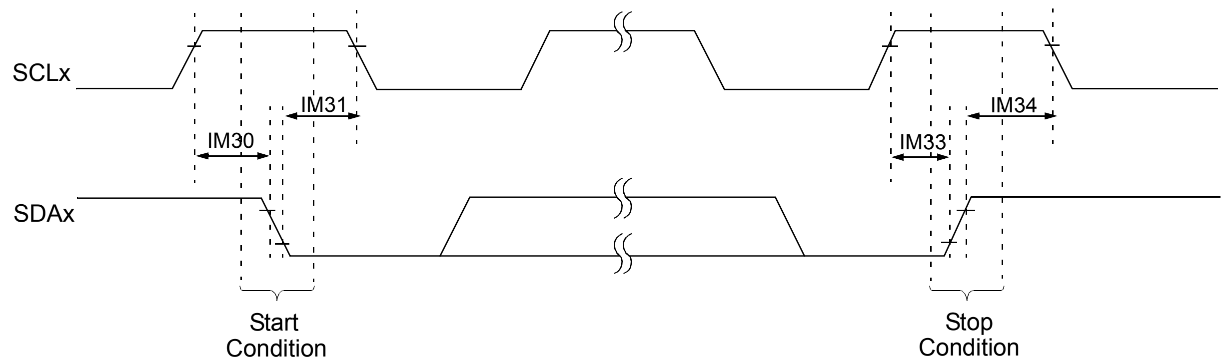
Note 1: These parameters are characterized but not tested in manufacturing.

2: Data in “Typ” column is at 3.3V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

3: The minimum clock period for SCKx is 20 ns. Therefore, the clock generated in Master mode must not violate this specification.

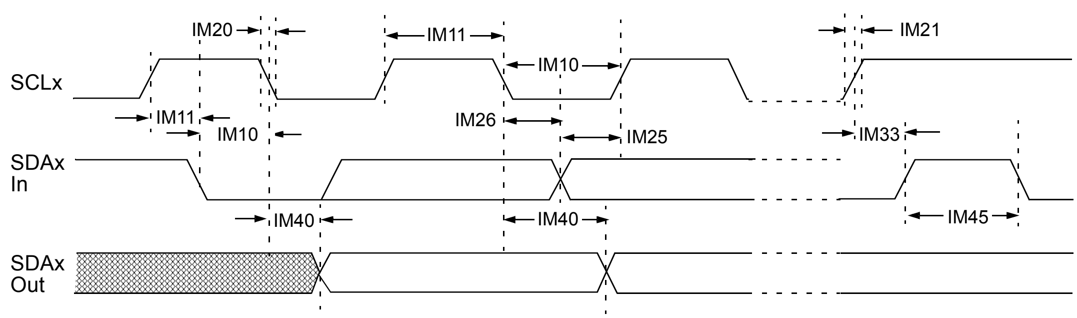
4: Assumes 50 pF load on all SPIx pins.

FIGURE 29-13: I2Cx BUS START/STOP BITS TIMING CHARACTERISTICS (MASTER MODE)



Note: Refer to Figure 29-1 for load conditions.

FIGURE 29-14: I2Cx BUS DATA TIMING CHARACTERISTICS (MASTER MODE)



Note: Refer to Figure 29-1 for load conditions.

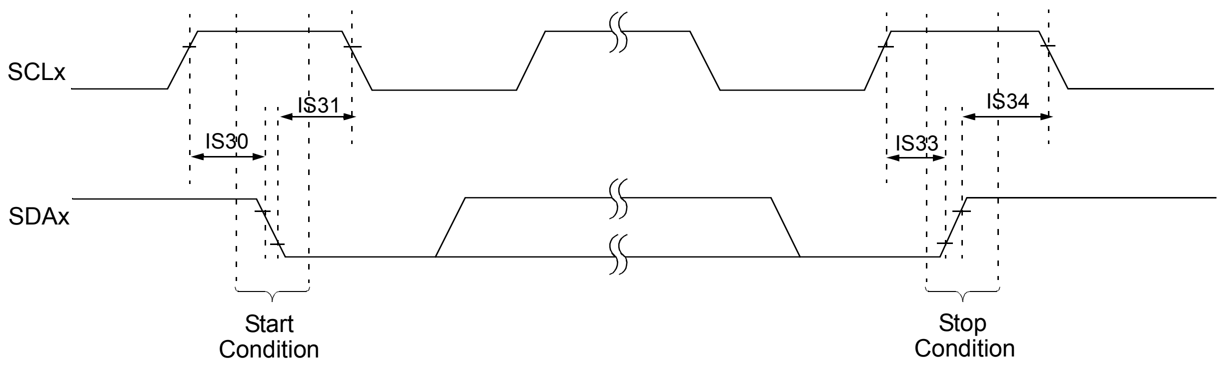
TABLE 29-31: I2Cx BUS DATA TIMING REQUIREMENTS (MASTER MODE)

AC CHARACTERISTICS				Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C			
Param No.	**Symbol**	**Characteristic**		**Min(1)**	**Max**	**Units**	**Conditions**
IM10	TLO:SCL	Clock Low Time	100 kHz mode	TPB/2 (BRG + 1)	—	µs	—
			400 kHz mode	TPB/2 (BRG + 1)	—	µs	—
			1 MHz mode(2)	TPB/2 (BRG + 1)	—	µs	—
IM11	THI:SCL	Clock High Time	100 kHz mode	TPB/2 (BRG + 1)	—	µs	—
			400 kHz mode	TPB/2 (BRG + 1)	—	µs	—
			1 MHz mode(2)	TPB/2 (BRG + 1)	—	µs	—
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 + 0.1 CB	300	ns	
			1 MHz mode(2)	—	100	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)	—	300	ns	
IM25	TSU:DAT	Data Input Setup Time	100 kHz mode	250	—	ns	—
			400 kHz mode	100	—	ns	
			1 MHz mode(2)	100	—	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	TPB/2 (BRG + 1)	—	µs	Only relevant for Repeated Start condition
			400 kHz mode	TPB/2 (BRG + 1)	—	µs	
			1 MHz mode(2)	TPB/2 (BRG + 1)	—	µs	
IM31	THD:STA	Start Condition Hold Time	100 kHz mode	TPB/2 (BRG + 1)	—	µs	After this period, the first clock pulse is generated
			400 kHz mode	TPB/2 (BRG + 1)	—	µs	
			1 MHz mode(2)	TPB/2 (BRG + 1)	—	µs	
IM33	TSU:STO	Stop Condition Setup Time	100 kHz mode	TPB/2 (BRG + 1)	—	µs	—
			400 kHz mode	TPB/2 (BRG + 1)	—	µs	
			1 MHz mode(2)	TPB/2 (BRG + 1)	—	µs	
IM34	THD:STO	Stop Condition Hold Time	100 kHz mode	TPB/2 (BRG + 1)	—	ns	—
			400 kHz mode	TPB/2 (BRG + 1)	—	ns	
			1 MHz mode(2)	TPB/2 (BRG + 1)	—	ns	
IM40	TAA:SCL	Output Valid From Clock	100 kHz mode	—	3500	ns	—
			400 kHz mode	—	1000	ns	—
			1 MHz mode(2)	—	350	ns	—
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	

Note 1: BRG is the value of the I^{2}C™ Baud Rate Generator.

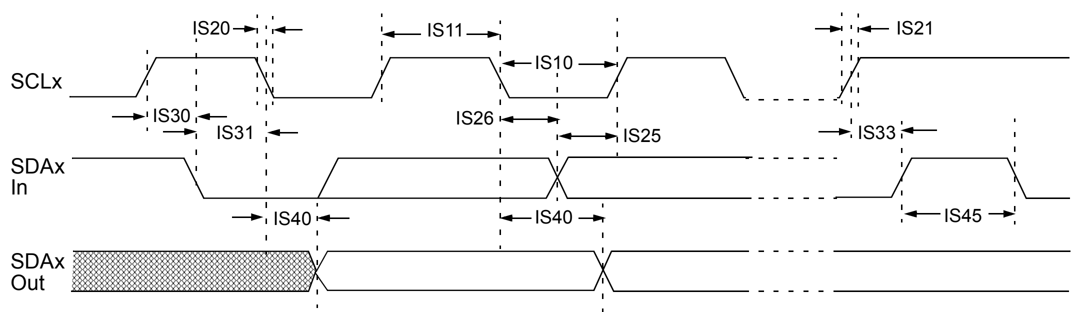
2: Maximum pin capacitance = 10 pF for all I2Cx pins (for 1 MHz mode only).

FIGURE 29-15: I2Cx BUS START/STOP BITS TIMING CHARACTERISTICS (SLAVE MODE)



Note: Refer to Figure 29-1 for load conditions.

FIGURE 29-16: I2Cx BUS DATA TIMING CHARACTERISTICS (SLAVE MODE)



Note: Refer to Figure 29-1 for load conditions.

TABLE 29-32: I2Cx BUS DATA TIMING REQUIREMENTS (SLAVE MODE)

AC CHARACTERISTICS				Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C			
Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
IS10	TLO:SCL	Clock Low Time	100 kHz mode	4.7	—	μs	Device must operate at a minimum of 1.5 MHz
			400 kHz mode	1.3	—	μs	Device must operate at a minimum of 10 MHz
			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)	0.5	—	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 + 0.1 CB	300	ns	
			1 MHz mode(1)	—	100	ns	
IS21	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(1)	—	300	ns	
IS25	TSU:DAT	Data Input Setup Time	100 kHz mode	250	—	ns	
			400 kHz mode	100	—	ns	
			1 MHz mode(1)	100	—	ns	
IS26	THD:DAT	Data Input Hold Time	100 kHz mode	0	—	ns	
			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	4700	—	μs	Only relevant for Repeated Start condition
			400 kHz mode	600	—	μs	
			1 MHz mode(1)	250	—	μs	
IS31	THD:STA	Start Condition Hold Time	100 kHz mode	4000	—	μs	After this period, the first clock pulse is generated
			400 kHz mode	600	—	μs	
			1 MHz mode(1)	250	—	μs	
IS33	TSU:STO	Stop Condition Setup Time	100 kHz mode	4000	—	μs	
			400 kHz mode	600	—	μs	
			1 MHz mode(1)	600	—	μs	
IS34	THD:STO	Stop Condition Hold Time	100 kHz mode	4000	—	ns	
			400 kHz mode	600	—	ns	
			1 MHz mode(1)	250		ns	
IS40	TAA:SCL	Output Valid From Clock	100 kHz mode	0	3500	ns	
			400 kHz mode	0	1000	ns	
			1 MHz mode(1)	0	350	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	CB	Bus Capacitive Loading		—	400	pF	

Note 1: Maximum pin capacitance = 10 pF for all I2Cx pins (for 1 MHz mode only).

TABLE 29-33: ADC MODULE SPECIFICATIONS

AC CHARACTERISTICS			Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for Industrial				
Param No.	**Symbol**	**Characteristic**	**Min.**	**Typ**	**Max.**	**Units**	**Conditions**
Device Supply							
AD01	AVDD	Module VDD Supply	Greater of VDD – 0.3 or 2.5	—	Lesser of VDD + 0.3 or 3.6	V	
AD02	AVSS	Module VSS Supply	VSS – 0.3	—	VSS + 0.3	V	
Reference Inputs							
AD05	VREFH	Reference Voltage High	AVSS + 2.0	—	AVDD	V	See **Note 1**
AD05a			3.0	—	3.6	V	VREFH = AVDD, VREFL = AVSS = 0
AD06	VREFL	Reference Voltage Low	AVSS	—	AVDD – 2.0	V	See **Note 1**
AD06a			0	—	0	V	VREFH = AVDD, VREFL = AVSS = 0
AD07	VREF	Absolute Reference Voltage	AVss – 0.3	—	AVss + 0.3	V	
AD08	IREF	Current Drain	—	200 —	300 3	µA µA	ADC operating ADC off
Analog Input							
AD12	VINH-VINL	Full-Scale Input Span	VREFL	—	VREFH	V	
	VINL	Absolute VINL Input Voltage	AVSS – 0.3		AVDD/2	V	
	VIN	Absolute Input Voltage	AVSS – 0.3		AVDD + 0.3	V	
		Leakage Current	—	+/- 0.001	+/-0.610	µA	VINL = AVSS = VREFL = 0V, AVDD = VREFH = 3.3V Source Impedance = 10KΩ
AD17	RIN	Recommended Impedance of Analog Voltage Source	—	—	10K	Ω	
ADC Accuracy – Measurements with External VREF+/VREF-							
AD20c	Nr	Resolution	10 data bits			bits	
AD21c	INL	Integral Nonlinearity	—	—	<+/-1	LSb	VINL = AVSS = VREFL = 0V, AVDD = VREFH = 3.3V
AD22c	DNL	Differential Nonlinearity	—	—	<+/-1	LSb	VINL = AVSS = VREFL = 0V, AVDD = VREFH = 3.3V
AD23c	GERR	Gain Error	—	—	<+/-2	LSb	VINL = AVSS = VREFL = 0V, AVDD = VREFH = 3.3V
AD24n	EOFF	Offset Error	—	—	<+/-2	LSb	VINL = AVSS = 0V, AVDD = 3.3V
AD25c	—	Monotonicity	—	—	—	—	Guaranteed

Legend: TBD = To Be Determined
Note 1: These parameters are not characterized or tested in manufacturing.

TABLE 29-33: ADC MODULE SPECIFICATIONS (CONTINUED)

AC CHARACTERISTICS			Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for Industrial				
Param No.	Symbol	Characteristic	Min.	Typ	Max.	Units	Conditions
ADC Accuracy – Measurements with Internal VREF+/VREF-							
AD20d	Nr	Resolution	10 data bits			bits	
AD21d	INL	Integral Nonlinearity	—	—	<+/-1	LSb	VINL = AVSS = 0V, AVDD = 3.6V
AD22d	DNL	Differential Nonlinearity	—	—	<+/-1	LSb	VINL = AVSS = 0V, AVDD = 3.6V
AD23d	GERR	Gain Error	—	—	<+/-4	LSb	VINL = AVSS = 0V, AVDD = 3.6V
AD24d	EOFF	Offset Error	—	—	<+/-2	LSb	VINL = AVSS = 0V, AVDD = 3.6V
AD25d	—	Monotonicity	—	—	—	—	Guaranteed

Legend: TBD = To Be Determined
Note 1: These parameters are not characterized or tested in manufacturing.

TABLE 29-34: A/D CONVERSION TIMING REQUIREMENTS

AC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C				
Param No.	Symbol	Characteristic	Min.	Typ(1)	Max.	Units	Conditions
Clock Parameters							
AD50	TAD	A/D Clock Period(2)	75	—	—	ns	TPB = 75 ns, AVDD = 3.0V
AD51	tRC	A/D Internal RC Oscillator Period	—	TBD	—	ns	—
Conversion Rate							
AD55	tCONV	Conversion Time	—	12 TAD	—	—	—
AD56	FCNV	Throughput Rate	—	—	1.1	MSPS	—
AD57	TSAMP	Sample Time	—	1 TAD	—	—	—
Timing Parameters							
AD60	tPCS	Conversion Start from Sample Trigger(3)	—	1.0 TAD	—	—	Auto-Convert Trigger (SSRC<2:0> = 111) not selected
AD61	tPSS	Sample Start from Setting Sample (SAMP) bit	0.5 TAD	—	1.5 TAD	—	—
AD62	tCSS	Conversion Completion to Sample Start (ASAM = 1)(3)	—	0.5 TAD	—	—	—

Legend: TBD = To Be Determined
Note 1: These parameters are characterized but not tested in manufacturing.
2: Because the sample caps will eventually lose charge, clock rates below 10 kHz can affect linearity performance, especially at elevated temperatures.
3: Characterized by design but not tested.

TABLE 29-34: A/D CONVERSION TIMING REQUIREMENTS (CONTINUED)

AC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C				
AD63	t_{DPU}	Time to Stabilize Analog Stage from A/D OFF to A/D ON(3)	—	—	2	µs	—

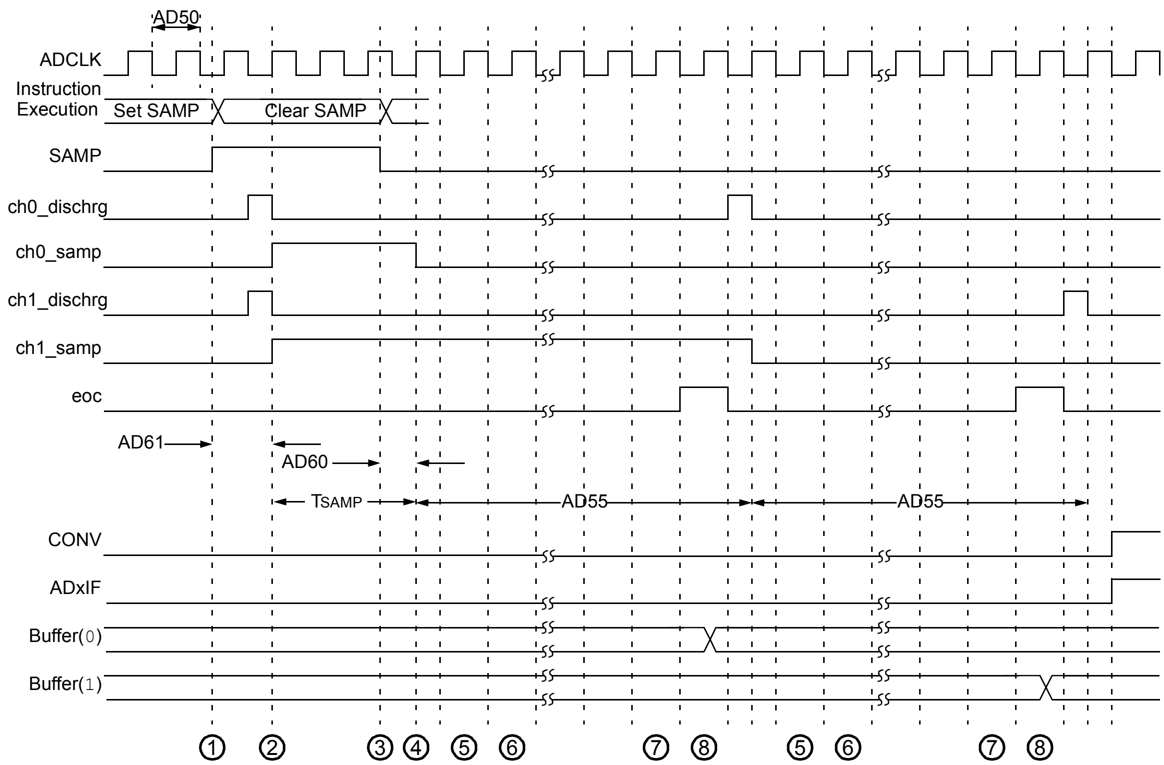
Legend: TBD = To Be Determined

Note 1: These parameters are characterized but not tested in manufacturing.

2: Because the sample caps will eventually lose charge, clock rates below 10 kHz can affect linearity performance, especially at elevated temperatures.

3: Characterized by design but not tested.

FIGURE 29-17: A/D CONVERSION (10-BIT MODE) TIMING CHARACTERISTICS (CHPS<1:0> = 01, SIMSAM = 0, ASAM = 0, SSRC<2:0> = 000)



① – Software sets ADxCON. SAMP to start sampling.

② – Sampling starts after discharge period. TSAMP is described in in the *“PIC32MX Family Reference Manual”* (DS61132).

③ – Software clears ADxCON. SAMP to start conversion.

④ – Sampling ends, conversion sequence starts.

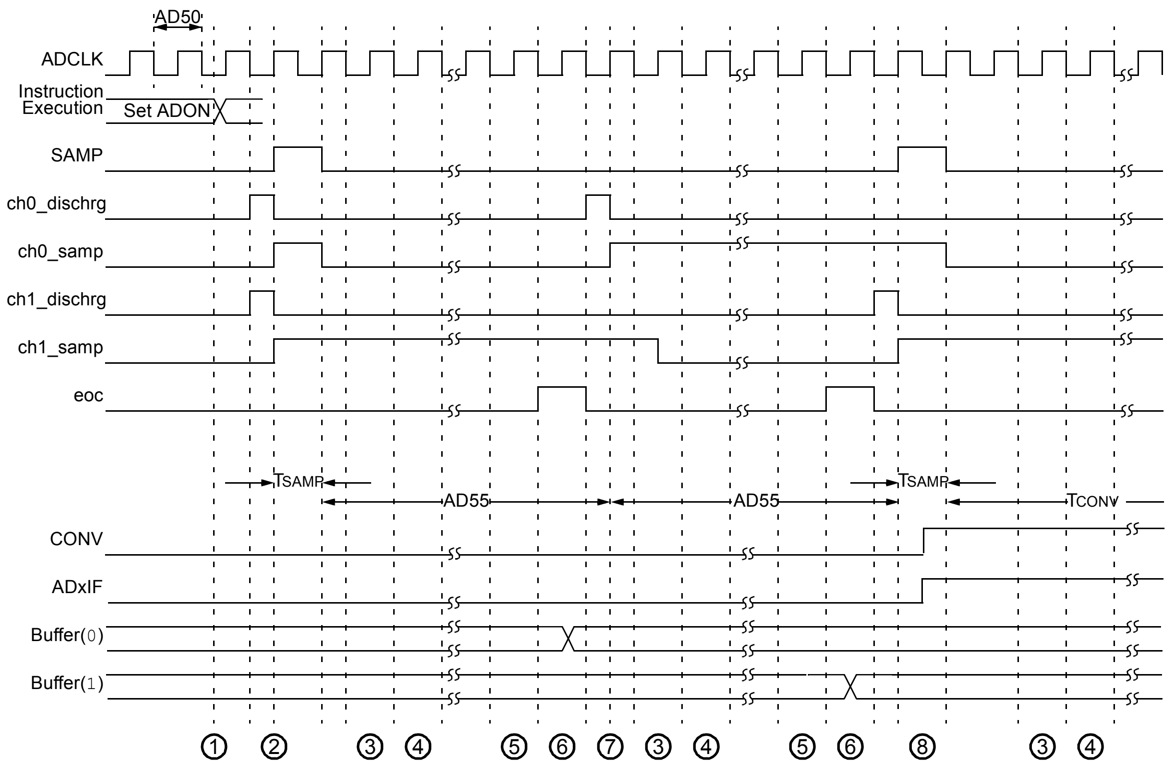
⑤ – Convert bit 9.

⑥ – Convert bit 8.

⑦ – Convert bit 0.

⑧ – One TAD for end of conversion.

FIGURE 29-18: A/D CONVERSION (10-BIT MODE) TIMING CHARACTERISTICS (CHPS<1:0> = `01`, SIMSAM = `0`, ASAM = `1`, SSRC<2:0> = `111`, SAMC<4:0> = `00001`



① – Software sets ADxCON. ADON to start AD operation.

② – Sampling starts after discharge period. TSAMP is described in the *"PIC32MX Family Reference Manual"* (DS61132).

③ – Convert bit 9.

④ – Convert bit 8.

⑤ – Convert bit 0.

⑥ – One TAD for end of conversion.

⑦ – Begin conversion of next channel.

⑧ – Sample for time specified by SAMC<4:0>.

FIGURE 29-19: EJTAG TIMING CHARACTERISTICS

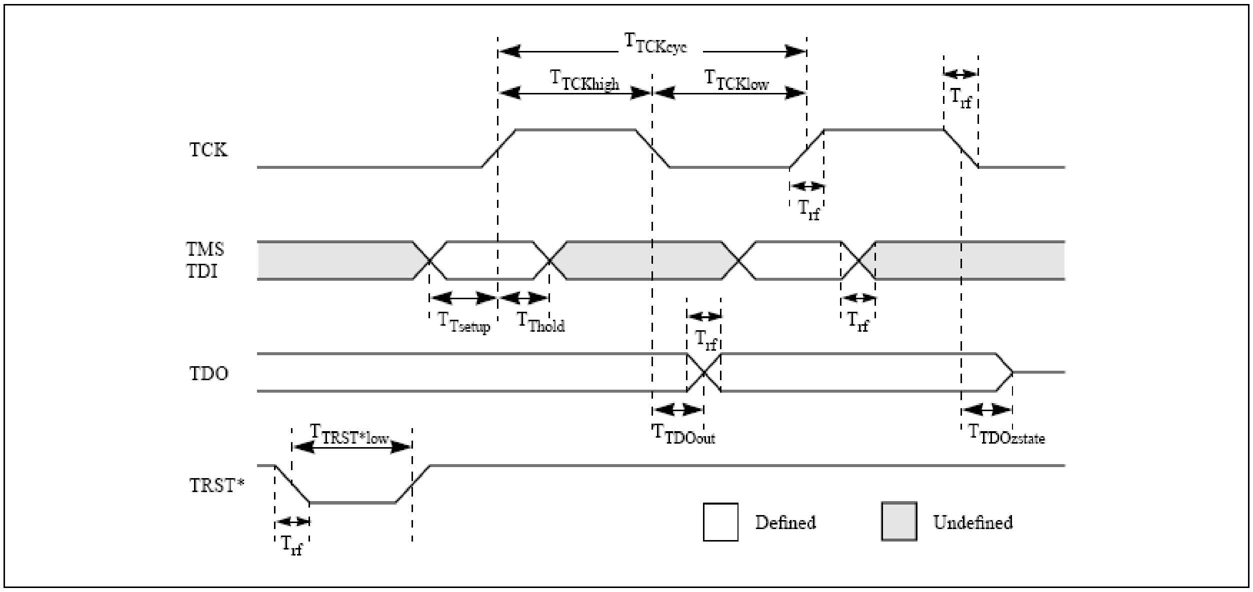


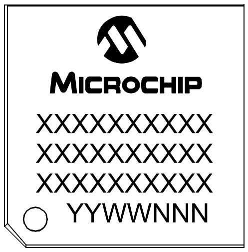
TABLE 29-35: EJTAG TIMING REQUIREMENTS

Symbol	Description	Min	Max	Units
Ttckcyc	TCK Cycle Time	25	—	ns
Ttckhigh	TCK High Time	10	—	ns
Ttcklow	TCK Low Time	10	—	ns
Ttsetup	TAP Signals Setup Time Before Rising TCK	5	—	ns
Tthold	TAP Signals Hold Time After Rising TCK	3	—	ns
Ttdoout	TDO Output Delay Time From Falling TCK	—	5	ns
Ttdozstate	TDO 3-State Delay Time From Falling TCK	—	5	ns
Ttrst*low	TRST* Low Time	25	—	ns
Trf	TAP Signals Rise/Fall Time, All Input and Output	—	—	ns

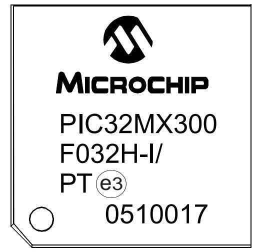
30.0 PACKAGING INFORMATION

30.1 Package Marking Information

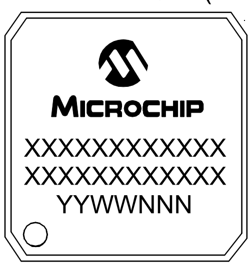
64-Lead TQFP (10x10x1 mm)



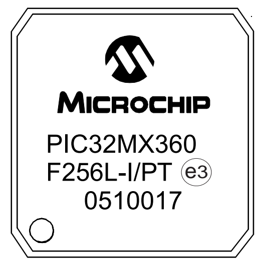
Example



100-Lead TQFP (12x12x1 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
		Pb-free JEDEC designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package.

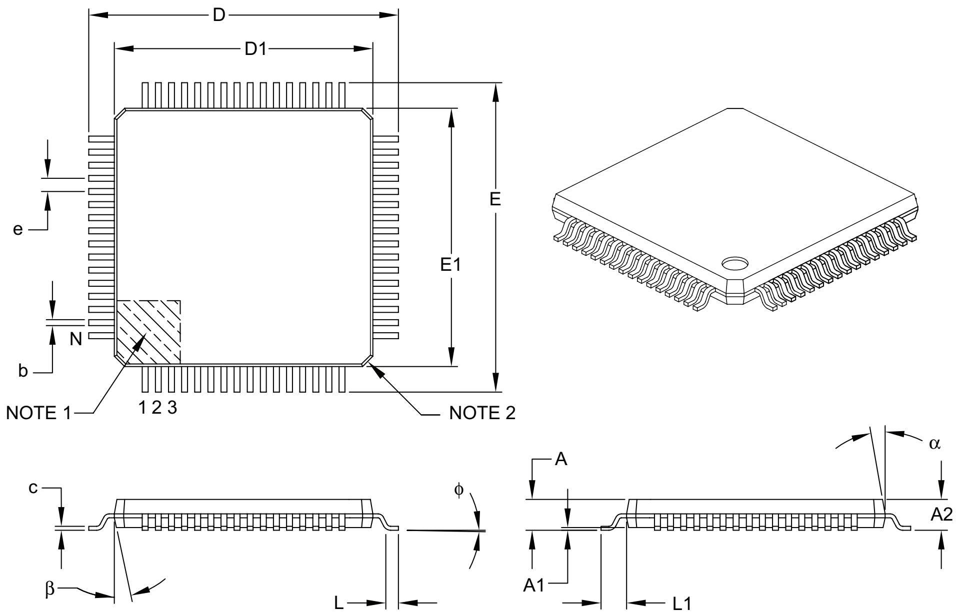
Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

30.2 Package Details

The following sections give the technical details of the packages.

64-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [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 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	φ	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 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

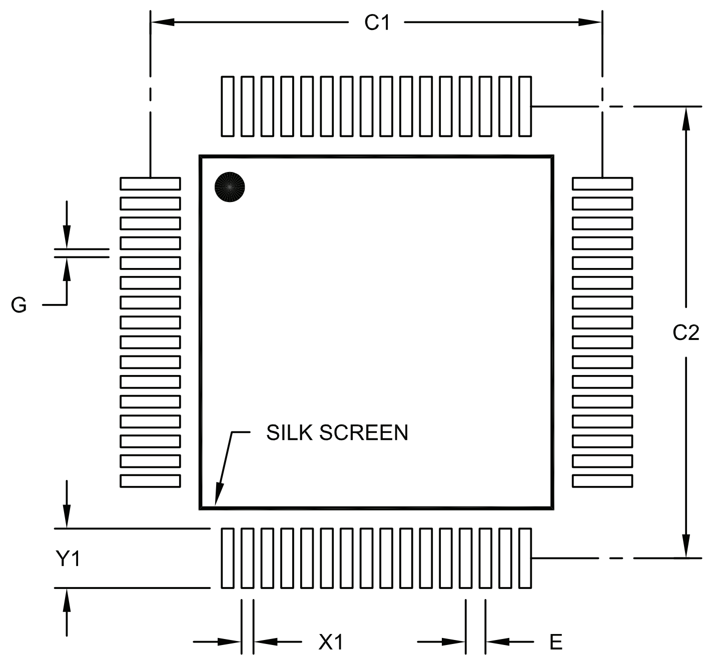
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-085B

64-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [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		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:

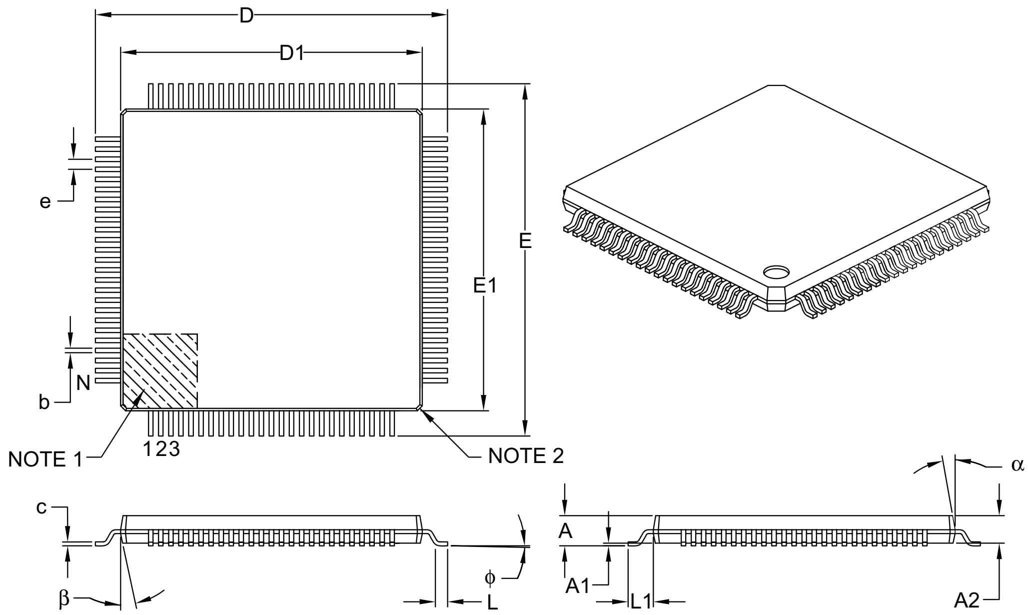
1. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2085A

100-Lead Plastic Thin Quad Flatpack (PT) – 12x12x1 mm Body, 2.00 mm [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 Leads	N	100		
Lead Pitch	e	0.40 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	φ	0°	3.5°	7°
Overall Width	E	14.00 BSC		
Overall Length	D	14.00 BSC		
Molded Package Width	E1	12.00 BSC		
Molded Package Length	D1	12.00 BSC		
Lead Thickness	c	0.09	–	0.20
Lead Width	b	0.13	0.18	0.23
Mold Draft Angle Top	α	11°	12°	13°
Mold Draft Angle Bottom	β	11°	12°	13°

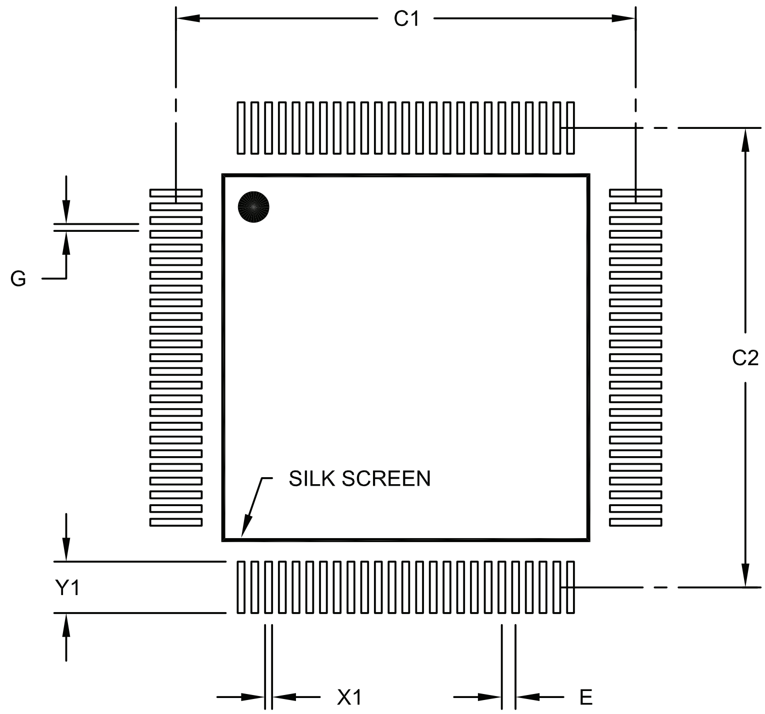
Notes:
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing 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 C04-100B

100-Lead Plastic Thin Quad Flatpack (PT) – 12x12x1 mm Body, 2.00 mm [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		0.40 BSC	
Contact Pad Spacing	C1		13.40	
Contact Pad Spacing	C2		13.40	
Contact Pad Width (X100)	X1			0.20
Contact Pad Length (X100)	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 Drawing No. C04-2100A

NOTES:

Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PIC32 MX 3XX F 512 H T - I / PT - XXX

- **Microchip Brand**
- **Architecture**
- **Product Groups**
- **Flash Memory Family**
- **Program Memory Size (KB)**
- **Pin Count**
- **Tape and Reel Flag (if applicable)**
- **Temperature Range**
- **Package**
- **Pattern**

Flash Memory Family

Architecture	MX = 32-bit RISC MCU core
Product Groups	3xx = General purpose microcontroller family
Flash Memory Family	F = Flash program memory
Program Memory Size	32 = 32K 64 = 64K 128 = 128K 256 = 256K 512 = 512K
Pin Count	H = 64-pin L = 100-pin
Temperature Range	I = -40°C to +85°C (Industrial)
Package	PT = 64-Lead, 100-Lead (12x12x1 mm) TQFP (Thin Quad Flatpack) PF = 100-Lead (14x14x1 mm) TQFP (Thin Quad Flatpack)
Pattern	Three-digit QTP, SQTP, Code or Special Requirements (blank otherwise) ES = Engineering Sample

Examples:

d) PIC32MX300F032H-I/PT:
General purpose PIC32MX, 32 KB program memory, 64-pin, Industrial temp., TQFP package.

e) PIC32MX360F256L-I/PT:
General purpose PIC32MX, 256 KB program memory, 100-pin, Industrial temp., TQFP package

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Fuzhou
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Shunde
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

10/05/07

