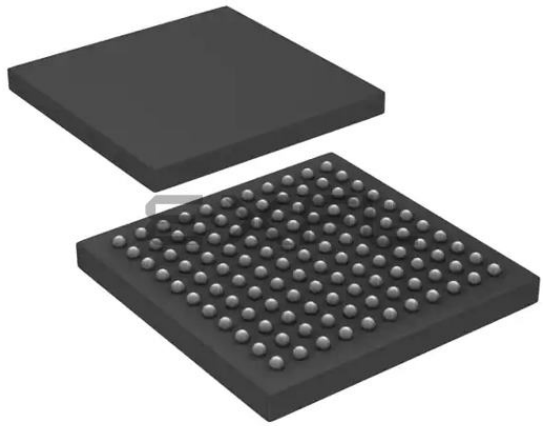
Microchip Technology - PIC24FJ128DA110T-I/BG Datasheet



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	16-Bit
Speed	32MHz
Connectivity	I²C, IrDA, SPI, UART/USART, USB OTG
Peripherals	Brown-out Detect/Reset, GFX, LVD, POR, PWM, WDT
Number of I/O	84
Program Memory Size	128KB (43K x 24)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	24K x 8
Voltage - Supply (Vcc/Vdd)	2.2V ~ 3.6V
Data Converters	A/D 24x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	121-TFBGA
Supplier Device Package	121-TFBGA (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic24fj128da110t-i-bg

TABLE 4-6: INTERRUPT CONTROLLER REGISTER MAP

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
INTCON1	0080	NSTDIS	—	—	—	—	—	—	—	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000
INTCON2	0082	ALTIVT	DISI	—	—	—	—	—	—	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP	0000
IFS0	0084	—	—	AD1IF	U1TXIF	U1RXIF	SPI1IF	SPF1IF	T3IF	T2IF	OC2IF	IC2IF	—	T1IF	OC1IF	IC1IF	INT0IF	0000
IFS1	0086	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	—	IC8IF	IC7IF	—	INT1IF	CNIF	CMIF	MI2C1IF	SI2C1IF	0000
IFS2	0088	—	—	PMPIF(1)	OC8IF	OC7IF	OC6IF	OC5IF	IC6IF	IC5IF	IC4IF	IC3IF	—	—	—	SPI2IF	SPF2IF	0000
IFS3	008A	—	RTCIF	—	—	—	—	—	—	—	INT4IF	INT3IF	—	—	MI2C2IF	SI2C2IF	—	0000
IFS4	008C	—	—	CTMUIF	—	—	—	—	LVDIF	—	—	—	—	CRCIF	U2ERIF	U1ERIF	—	0000
IFS5	008E	—	—	IC9IF	OC9IF	SPI3IF	SPF3IF	U4TXIF	U4RXIF	U4ERIF	USB1IF	MI2C3IF	SI2C3IF	U3TXIF	U3RXIF	U3ERIF	—	0000
IFS6	0090	—	—	—	—	—	—	—	—	—	—	—	GFX1IF	—	—	—	—	0000
IEC0	0094	—	—	AD1IE	U1TXIE	U1RXIE	SPI1IE	SPF1IE	T3IE	T2IE	OC2IE	IC2IE	—	T1IE	OC1IE	IC1IE	INT0IE	0000
IEC1	0096	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	—	IC8IE	IC7IE	—	INT1IE	CNIE	CMIE	MI2C1IE	SI2C1IE	0000
IEC2	0098	—	—	PMPIE(1)	OC8IE	OC7IE	OC6IE	OC5IE	IC6IE	IC5IE	IC4IE	IC3IE	—	—	—	SPI2IE	SPF2IE	0000
IEC3	009A	—	RTCIE	—	—	—	—	—	—	—	INT4IE	INT3IE	—	—	MI2C2IE	SI2C2IE	—	0000
IEC4	009C	—	—	CTMUIE	—	—	—	—	LVDIE	—	—	—	—	CRCIE	U2ERIE	U1ERIE	—	0000
IEC5	009E	—	—	IC9IE	OC9IE	SPI3IE	SPF3IE	U4TXIE	U4RXIE	U4ERIE	USB1IE	MI2C3IE	SI2C3IE	U3TXIE	U3RXIE	U3ERIE	—	0000
IEC6	00A0	—	—	—	—	—	—	—	—	—	—	—	GFX1IE	—	—	—	—	0000
IPC0	00A4	—	T1IP2	T1IP1	T1IP0	—	OC1IP2	OC1IP1	OC1IP0	—	IC1IP2	IC1IP1	IC1IP0	—	INT0IP2	INT0IP1	INT0IP0	4444
IPC1	00A6	—	T2IP2	T2IP1	T2IP0	—	OC2IP2	OC2IP1	OC2IP0	—	IC2IP2	IC2IP1	IC2IP0	—	—	—	—	4440
IPC2	00A8	—	U1RXIP2	U1RXIP1	U1RXIP0	—	SPI1IP2	SPI1IP1	SPI1IP0	—	SPF1IP2	SPF1IP1	SPF1IP0	—	T3IP2	T3IP1	T3IP0	4444
IPC3	00AA	—	—	—	—	—	—	—	—	—	AD1IP2	AD1IP1	AD1IP0	—	U1TXIP2	U1TXIP1	U1TXIP0	0044
IPC4	00AC	—	CNIP2	CNIP1	CNIP0	—	CMIP2	CMIP1	CMIP0	—	MI2C1IP2	MI2C1IP1	MI2C1IP0	—	SI2C1IP2	SI2C1IP1	SI2C1IP0	4444
IPC5	00AE	—	IC8IP2	IC8IP1	IC8IP0	—	IC7IP2	IC7IP1	IC7IP0	—	—	—	—	—	INT1IP2	INT1IP1	INT1IP0	4404
IPC6	00B0	—	T4IP2	T4IP1	T4IP0	—	OC4IP2	OC4IP1	OC4IP0	—	OC3IP2	OC3IP1	OC3IP0	—	—	—	—	4440
IPC7	00B2	—	U2TXIP2	U2TXIP1	U2TXIP0	—	U2RXIP2	U2RXIP1	U2RXIP0	—	INT2IP2	INT2IP1	INT2IP0	—	T5IP2	T5IP1	T5IP0	4444
IPC8	00B4	—	—	—	—	—	—	—	—	—	SPI2IP2	SPI2IP1	SPI2IP0	—	SPF2IP2	SPF2IP1	SPF2IP0	0044
IPC9	00B6	—	IC5IP2	IC5IP1	IC5IP0	—	IC4IP2	IC4IP1	IC4IP0	—	IC3IP2	IC3IP1	IC3IP0	—	—	—	—	4440
IPC10	00B8	—	OC7IP2	OC7IP1	OC7IP0	—	OC6IP2	OC6IP1	OC6IP0	—	OC5IP2	OC5IP1	OC5IP0	—	IC6IP2	IC6IP1	IC6IP0	4444
IPC11	00BA	—	—	—	—	—	—	—	—	—	PMPIP2(1)	PMPIP1(1)	PMPIP0(1)	—	OC8IP2	OC8IP1	OC8IP0	0044(2)
IPC12	00BC	—	—	—	—	—	MI2C2IP2	MI2C2IP1	MI2C2IP0	—	SI2C2IP2	SI2C2IP1	SI2C2IP0	—	—	—	—	0440
IPC13	00BE	—	—	—	—	—	INT4IP2	INT4IP1	INT4IP0	—	INT3IP2	INT3IP1	INT3IP0	—	—	—	—	0440
IPC15	00C2	—	—	—	—	—	RTCIP2	RTCIP1	RTCIP0	—	—	—	—	—	—	—	—	0400

Legend: — = unimplemented, read as '0'. Reset values are shown in hexadecimal.
Note 1: Unimplemented in 64-pin devices, read as '0'.
2: The Reset value in 64-pin devices are '0004'.

4.2.6 SOFTWARE STACK

Apart from its use as a working register, the W15 register in PIC24F devices is also used as a Software Stack Pointer (SSP). The pointer always points to the first available free word and grows from lower to higher addresses. It pre-decrements for stack pops and post-increments for stack pushes, as shown in Figure 4-7. Note that for a PC push during any CALL instruction, the MSB of the PC is zero-extended before the push, ensuring that the MSB is always clear.

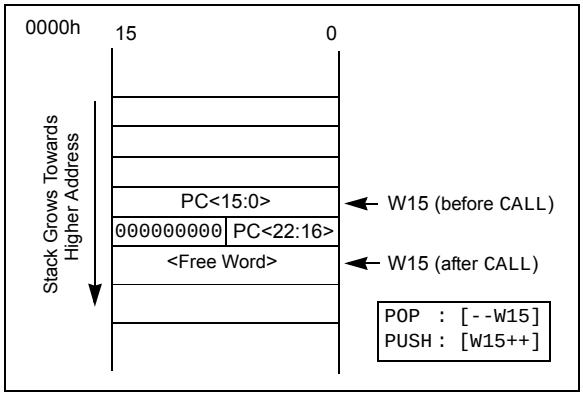
Note:	A PC push during exception processing will concatenate the SRL register to the MSB of the PC prior to the push.

The Stack Pointer Limit Value register (SPLIM), associated with the Stack Pointer, sets an upper address boundary for the stack. SPLIM is uninitialized at Reset. As is the case for the Stack Pointer, SPLIM<0> is forced to '0' as all stack operations must be word-aligned. Whenever an EA is generated using W15 as a source or destination pointer, the resulting address is compared with the value in SPLIM. If the contents of the Stack Pointer (W15) and the SPLIM register are equal, and a push operation is performed, a stack error trap will not occur. The stack error trap will occur on a subsequent push operation. Thus, for example, if it is desirable to cause a stack error trap when the stack grows beyond address 2000h in RAM, initialize the SPLIM with the value, 1FFEh.

Similarly, a Stack Pointer underflow (stack error) trap is generated when the Stack Pointer address is found to be less than 0800h. This prevents the stack from interfering with the SFR space.

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

FIGURE 4-7: CALL STACK FRAME



4.3 Interfacing Program and Data Memory Spaces

The PIC24F architecture uses a 24-bit wide program space and 16-bit wide data space. The architecture is also a modified Harvard scheme, meaning that data can also be present in the program space. To use this data successfully, it must be accessed in a way that preserves the alignment of information in both spaces.

Aside from normal execution, the PIC24F architecture provides two methods by which program space can be accessed during operation:

- Using table instructions to access individual bytes or words anywhere in the program space
- Remapping a portion of the program space into the data space (program space visibility)

Table instructions allow an application to read or write to small areas of the program memory. This makes the method ideal for accessing data tables that need to be updated from time to time. It also allows access to all bytes of the program word. The remapping method allows an application to access a large block of data on a read-only basis, which is ideal for look ups from a large table of static data. It can only access the least significant word of the program word.

4.3.1 ADDRESSING PROGRAM SPACE

Since the address ranges for the data and program spaces are 16 and 24 bits, respectively, a method is needed to create a 23-bit or 24-bit program address from 16-bit data registers. The solution depends on the interface method to be used.

For table operations, the 8-bit Table Memory Page Address register (TBLPAG) is used to define a 32K word region within the program space. This is concatenated with a 16-bit EA to arrive at a full 24-bit program space address. In this format, the MSBs of TBLPAG is used to determine if the operation occurs in the user memory (TBLPAG<7> = 0) or the configuration memory (TBLPAG<7> = 1).

For remapping operations, the 10-bit Extended Data Space Read register (DSRPAG) is used to define a 16K word page in the program space. When the Most Significant bit (MSb) of the EA is '1', and the MSb (bit 9) of DSRPAG is '1', the lower 8 bits of DSRPAG are concatenated with the lower 15 bits of the EA to form a 23-bit program space address. The DSRPAG<8> bit decides whether the lower word (when bit is '0') or the higher word (when bit is '1') of program memory is mapped. Unlike table operations, this strictly limits remapping operations to the user memory area.

Table 4-36 and Figure 4-8 show how the program EA is created for table operations and remapping accesses from the data EA. Here, P<23:0> refers to a program space word, whereas D<15:0> refers to a data space word.

5.6.1 PROGRAMMING ALGORITHM FOR FLASH PROGRAM MEMORY

The user can program one row of Flash program 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 (512 instructions) and store in data RAM.
2. Update the program data in RAM with the desired new data.
3. Erase the block (see Example 5-1):
   a) Set the NVMOP bits (NVMCON<3:0>) to '0010' to configure for block erase. Set the ERASE (NVMCON<6>) and WREN (NVMCON<14>) bits.
   b) Write the starting address of the block to be erased into the TBLPAG and W registers.
   c) Write 55h to NVMKEY.
   d) Write AAh to NVMKEY.
   e) Set the WR bit (NVMCON<15>). The erase cycle begins and the CPU stalls for the duration of the erase cycle. When the erase is done, the WR bit is cleared automatically.
4. Write the first 64 instructions from data RAM into the program memory buffers (see Example 5-3).
5. Write the program block to Flash memory:
   a) Set the NVMOP bits to '0001' to configure for row programming. Clear the ERASE bit and set the WREN bit.
   b) Write 55h to NVMKEY.
   c) Write AAh to NVMKEY.
   d) Set the WR bit. The programming cycle begins and the CPU stalls for the duration of the write cycle. When the write to Flash memory is done, the WR bit is cleared automatically.
6. Repeat steps 4 and 5, using the next available 64 instructions from the block in data RAM by incrementing the value in TBLPAG, until all 512 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. The two instructions following the start of the programming sequence should be NOPs, as shown in Example 5-4.

EXAMPLE 5-1: ERASING A PROGRAM MEMORY BLOCK (ASSEMBLY LANGUAGE CODE)

```
; Set up NVMCON for block erase operation
    MOV #0x4042, W0 ;
    MOV W0, NVMCON                    ; Initialize NVMCON
; Init pointer to row to be ERASED
    MOV #tblpage(PROG_ADDR), W0       ;
    MOV W0, TBLPAG                    ; Initialize Program Memory (PM) Page Boundary SFR
    MOV #tbloffset(PROG_ADDR), W0     ; Initialize in-page EA<15:0> pointer
    TBLWTL W0, [W0]                   ; Set base address of erase block
    DISI #5                           ; Block all interrupts with priority <7
                                      ; for next 5 instructions
    MOV.B #0x55, W0
    MOV W0, NVMKEY                    ; Write the 0x55 key
    MOV.B #0xAA, W1 ;
    MOV W1, NVMKEY                    ; Write the 0xAA key
    BSET NVMCON, #WR                  ; Start the erase sequence
    NOP                               ; Insert two NOPs after the erase
    NOP                               ; command is asserted
```

REGISTER 7-19: IPC0: INTERRUPT PRIORITY CONTROL REGISTER 0

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T1IP2	T1IP1	T1IP0	—	OC1IP2	OC1IP1	OC1IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	IC1IP2	IC1IP1	IC1IP0	—	INT0IP2	INT0IP1	INT0IP0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **Unimplemented:** Read as ‘0’

bit 14-12 **T1IP<2:0>:** Timer1 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **OC1IP<2:0>:** Output Compare Channel 1 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **IC1IP<2:0>:** Input Capture Channel 1 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **INT0IP<2:0>:** External Interrupt 0 Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

REGISTER 7-22: IPC3: INTERRUPT PRIORITY CONTROL REGISTER 3

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	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	AD1IP2	AD1IP1	AD1IP0	—	U1TXIP2	U1TXIP1	U1TXIP0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

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

bit 6-4 **AD1IP<2:0>:** A/D Conversion Complete Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **U1TXIP<2:0>:** UART1 Transmitter Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

REGISTER 7-25: IPC6: INTERRUPT PRIORITY CONTROL REGISTER 6

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T4IP2	T4IP1	T4IP0	—	OC4IP2	OC4IP1	OC4IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	OC3IP2	OC3IP1	OC3IP0	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **Unimplemented:** Read as '0'

bit 14-12 **T4IP<2:0>:** Timer4 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 11 **Unimplemented:** Read as '0'

bit 10-8 **OC4IP<2:0>:** Output Compare Channel 4 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 7 **Unimplemented:** Read as '0'

bit 6-4 **OC3IP<2:0>:** Output Compare Channel 3 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 3-0 **Unimplemented:** Read as '0'

REGISTER 7-29: IPC10: INTERRUPT PRIORITY CONTROL REGISTER 10

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	OC7IP2	OC7IP1	OC7IP0	—	OC6IP2	OC6IP1	OC6IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	OC5IP2	OC5IP1	OC5IP0	—	IC6IP2	IC6IP1	IC6IP0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **Unimplemented:** Read as '0'

bit 14-12 **OC7IP<2:0>:** Output Compare Channel 7 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 11 **Unimplemented:** Read as '0'

bit 10-8 **OC6IP<2:0>:** Output Compare Channel 6 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 7 **Unimplemented:** Read as '0'

bit 6-4 **OC5IP<2:0>:** Output Compare Channel 5 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 3 **Unimplemented:** Read as '0'

bit 2-0 **IC6IP<2:0>:** Input Capture Channel 6 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

REGISTER 7-41: IPC25: INTERRUPT PRIORITY CONTROL REGISTER 25

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	R/W-1	R/W-0	R/W-0
—	—	—	—	—	GFX1IP2	GFX1IP1	GFX1IP0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-3 **Unimplemented:** Read as ‘0’

bit 2-0 **GFX1IP<2:0>:** Graphics 1 Interrupt Priority bits

111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

REGISTER 8-2: CLKDIV: CLOCK DIVIDER REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
ROI	DOZE2	DOZE1	DOZE0	DOZEN(1)	RCDIV2	RCDIV1	RCDIV0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
CPDIV1	CPDIV0	PLLEN	G1CLKSEL	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **ROI:** Recover on Interrupt bit
1 = Interrupts clear the DOZEN bit and reset the CPU peripheral clock ratio to 1:1
0 = Interrupts have no effect on the DOZEN bit

bit 14-12 **DOZE<2:0>:** CPU Peripheral Clock Ratio Select bits
111 = 1:128
110 = 1:64
101 = 1:32
100 = 1:16
011 = 1:8
010 = 1:4
001 = 1:2
000 = 1:1

bit 11 **DOZEN:** DOZE Enable bit(1)
1 = DOZE<2:0> bits specify the CPU peripheral clock ratio
0 = CPU peripheral clock ratio is set to 1:1

bit 10-8 **RCDIV<2:0>:** FRC Postscaler Select bits
111 = 31.25 kHz (divide by 256)
110 = 125 kHz (divide by 64)
101 = 250 kHz (divide by 32)
100 = 500 kHz (divide by 16)
011 = 1 MHz (divide by 8)
010 = 2 MHz (divide by 4)
001 = 4 MHz (divide by 2)
000 = 8 MHz (divide by 1)

bit 7-6 **CPDIV<1:0>:** System Clock Select bits (postscaler select from 32 MHz clock branch)
11 = 4 MHz (divide by 8)(2)
10 = 8 MHz (divide by 4)(2)
01 = 16 MHz (divide by 2)
00 = 32 MHz (divide by 1)

Note 1: This bit is automatically cleared when the ROI bit is set and an interrupt occurs.
2: This setting is not allowed while the USB module is enabled.

10.3 Input Change Notification

The input change notification function of the I/O ports allows the PIC24FJ256DA210 family of devices to generate interrupt requests to the processor in response to a Change-Of-State (COS) on selected input pins. This feature is capable of detecting input Change-Of-States, even in Sleep mode, when the clocks are disabled. Depending on the device pin count, there are up to 84 external inputs that may be selected (enabled) for generating an interrupt request on a Change-Of-State.

Registers, CNEN1 through CNEN6, contain the interrupt enable control bits for each of the CN input pins. Setting any of these bits enables a CN interrupt for the corresponding pins.

Each CN pin has a both a weak pull-up and a weak pull-down connected to it. The pull-ups act as a current source that is connected to the pin, while the pull-downs act as a current sink that is connected to the pin. These eliminate the need for external resistors when push button or keypad devices are connected. The pull-ups and pull-downs are separately enabled using the CNPU1 through CNPU6 registers (for pull-ups), and the CNPD1 through CNPD6 registers (for pull-downs). Each CN pin has individual control bits for its pull-up and pull-down. Setting a control bit enables the weak pull-up or pull-down for the corresponding pin.

When the internal pull-up is selected, the pin pulls up to V_{DD} – 1.1V (typical). When the internal pull-down is selected, the pin pulls down to V_{SS}.

Note:	Pull-ups on change notification pins should always be disabled whenever the port pin is configured as a digital output.

Note:	To use CN83 and CN84, which are on the D+ and D- pins, the UTRDIS bit (U1CNFG2<0>) should be set.

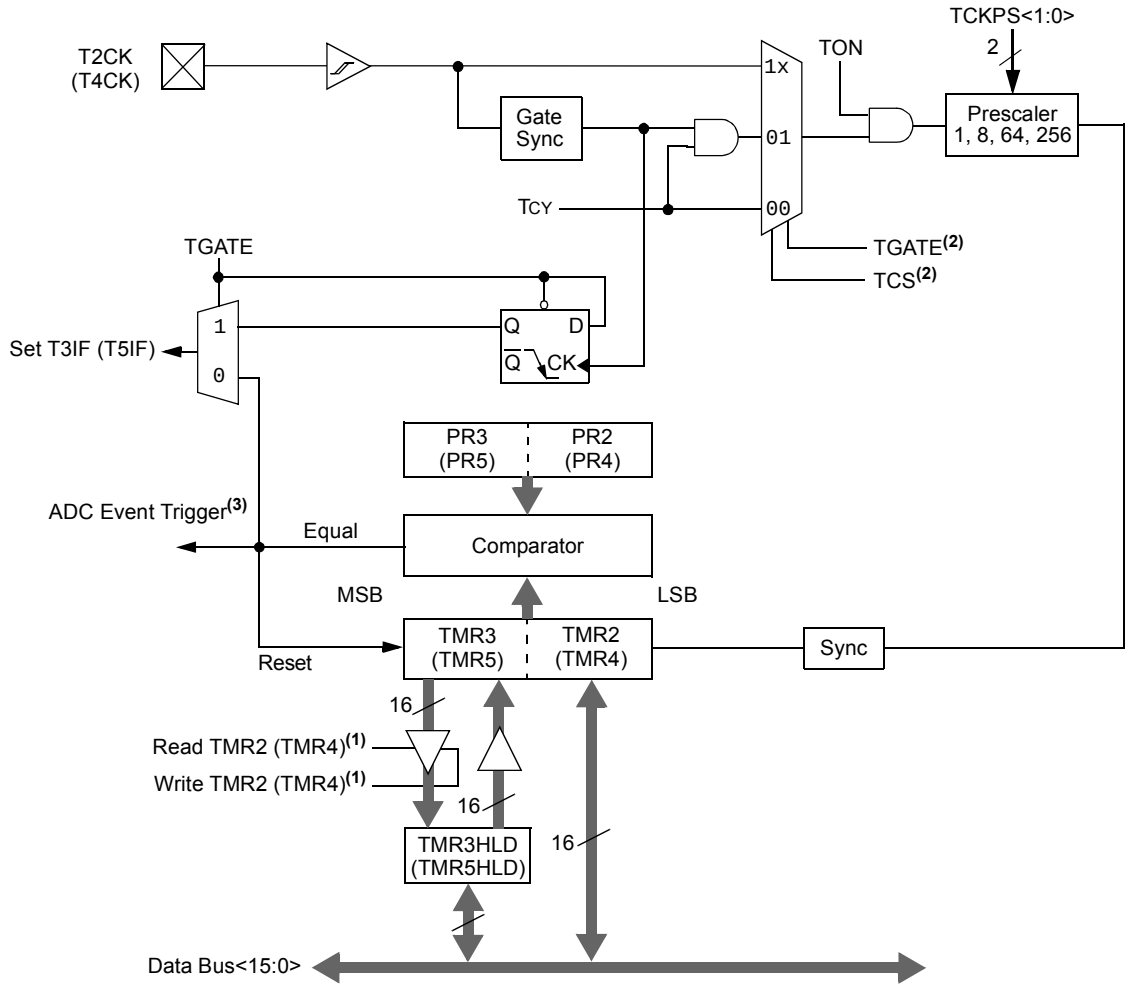
EXAMPLE 10-1: PORT WRITE/READ IN ASSEMBLY

```
MOV   0xFF00, W0   ; Configure PORTB<15:8> as inputs
MOV   W0, TRISB    ; and PORTB<7:0> as outputs
NOP                ; Delay 1 cycle
BTSS  PORTB, #13   ; Next Instruction
```

EXAMPLE 10-2: PORT WRITE/READ IN 'C'

```
TRISB = 0xFF00;               // Configure PORTB<15:8> as inputs and PORTB<7:0> as outputs
Nop();                        // Delay 1 cycle
If (PORTBbits.RB13){ };       // Next Instruction
```

FIGURE 12-1: TIMER2/3 AND TIMER4/5 (32-BIT) BLOCK DIAGRAM



Note 1: The 32-Bit Timer Configuration bit, T32, must be set for 32-bit timer/counter operation. All control bits are respective to the T2CON and T4CON registers.

2: The timer clock input must be assigned to an available RPn/RPIn pin before use. See **Section 10.4 "Peripheral Pin Select (PPS)"** for more information.

3: The ADC event trigger is available only on Timer 2/3 in 32-bit mode and Timer 3 in 16-bit mode.

14.3.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the OCxRS and OCxR registers. The OCxRS and OCxR registers can be written to at any time, but the duty cycle value is not latched until a match between PRy and TMRy occurs (i.e., the period is complete). This provides a double buffer for the PWM duty cycle and is essential for glitchless PWM operation.

Some important boundary parameters of the PWM duty cycle include:

- If OCxR, OCxRS, and PRy are all loaded with 0000h, the OCx pin will remain low (0% duty cycle).
- If OCxRS is greater than PRy, the pin will remain high (100% duty cycle).

See Example 14-1 for PWM mode timing details. Table 14-1 and Table 14-2 show example PWM frequencies and resolutions for a device operating at 4 MIPS and 10 MIPS, respectively.

EQUATION 14-2: CALCULATION FOR MAXIMUM PWM RESOLUTION(1)

$$\text{Maximum PWM Resolution (bits)} = \frac{\log_{10}\left(\frac{FCY}{FPWM \bullet (\text{Timer Prescale Value})}\right)}{\log_{10}(2)} \text{ bits}$$

Note 1: Based on FCY = FOSC/2; Doze mode and PLL are disabled.

EXAMPLE 14-1: PWM PERIOD AND DUTY CYCLE CALCULATIONS(1)

1. Find the Timer Period register value for a desired PWM frequency of 52.08 kHz, where FOSC = 8 MHz with PLL (32 MHz device clock rate) and a Timer2 prescaler setting of 1:1.

   TCY = 2 * TOSC = 62.5 ns

   PWM Period = 1/PWM Frequency = 1/52.08 kHz = 19.2 ms

   PWM Period = (PR2 + 1) • TCY • (Timer2 Prescale Value)

   19.2 ms = PR2 + 1) • 62.5 ns • 1

   PR2 = 306

2. Find the maximum resolution of the duty cycle that can be used with a 52.08 kHz frequency and a 32 MHz device clock rate:

   PWM Resolution = $\log_{10}(FCY/FPWM)/\log_{10}2)$ bits

   = $(\log_{10}(16 \text{ MHz}/52.08 \text{ kHz})/\log_{10}2)$ bits

   = 8.3 bits

Note 1: Based on TCY = 2 * TOSC; Doze mode and PLL are disabled.

TABLE 14-1: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 4 MIPS (FCY = 4 MHz)(1)

PWM Frequency	7.6 Hz	61 Hz	122 Hz	977 Hz	3.9 kHz	31.3 kHz	125 kHz
Timer Prescaler Ratio	8	1	1	1	1	1	1
Period Register Value	FFFFh	FFFFh	7FFFh	0FFFh	03FFh	007Fh	001Fh
Resolution (bits)	16	16	15	12	10	7	5

Note 1: Based on FCY = FOSC/2; Doze mode and PLL are disabled.

TABLE 14-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 16 MIPS (FCY = 16 MHz)(1)

PWM Frequency	30.5 Hz	244 Hz	488 Hz	3.9 kHz	15.6 kHz	125 kHz	500 kHz
Timer Prescaler Ratio	8	1	1	1	1	1	1
Period Register Value	FFFFh	FFFFh	7FFFh	0FFFh	03FFh	007Fh	001Fh
Resolution (bits)	16	16	15	12	10	7	5

Note 1: Based on FCY = FOSC/2; Doze mode and PLL are disabled.

REGISTER 18-9: U1ADDR: USB ADDRESS REGISTER

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
LSPDEN(1)	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-8 **Unimplemented:** Read as '0'

bit 7 **LSPDEN:** Low-Speed Enable Indicator bit(1)
1 = USB module operates at low speed
0 = USB module operates at full speed

bit 6-0 **ADDR<6:0>:** USB Device Address bits

Note 1: Host mode only. In Device mode, this bit is unimplemented and read as '0'.

REGISTER 18-10: U1TOK: USB TOKEN REGISTER (HOST MODE ONLY)

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
PID3	PID2	PID1	PID0	EP3	EP2	EP1	EP0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-8 **Unimplemented:** Read as '0'

bit 7-4 **PID<3:0>:** Token Type Identifier bits
1101 = SETUP (TX) token type transaction(1)
1001 = IN (RX) token type transaction(1)
0001 = OUT (TX) token type transaction(1)

bit 3-0 **EP<3:0>:** Token Command Endpoint Address bits
This value must specify a valid endpoint on the attached device.

Note 1: All other combinations are reserved and are not to be used.

REGISTER 20-3: ALCFGRPT: ALARM CONFIGURATION REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0, HSC	R/W-0, HSC
ALRMEN	CHIME	AMASK3	AMASK2	AMASK1	AMASK0	ALRMPTR1	ALRMPTR0
bit 15							bit 8

R/W-0, HSC	R/W-0, HSC	R/W-0, HSC	R/W-0, HSC	R/W-0, HSC	R/W-0, HSC	R/W-0, HSC	R/W-0, HSC
ARPT7	ARPT6	ARPT5	ARPT4	ARPT3	ARPT2	ARPT1	ARPT0
bit 7							bit 0

Legend:	HSC = Hardware Settable/Clearable bit		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **ALRMEN:** Alarm Enable bit
1 = Alarm is enabled (cleared automatically after an alarm event whenever ARPT<7:0> = 00h and CHIME = 0)
0 = Alarm is disabled

bit 14 **CHIME:** Chime Enable bit
1 = Chime is enabled; ARPT<7:0> bits are allowed to roll over from 00h to FFh
0 = Chime is disabled; ARPT<7:0> bits stop once they reach 00h

bit 13-10 **AMASK<3:0>:** Alarm Mask Configuration bits
11xx = Reserved – do not use
101x = Reserved – do not use
1001 = Once a year (except when configured for February 29th, once every 4 years)
1000 = Once a month
0111 = Once a week
0110 = Once a day
0101 = Every hour
0100 = Every 10 minutes
0011 = Every minute
0010 = Every 10 seconds
0001 = Every second
0000 = Every half second

bit 9-8 **ALRMPTR<1:0>:** Alarm Value Register Window Pointer bits
Points to the corresponding Alarm Value registers when reading the ALRMVALH and ALRMVALL registers. The ALRMPTR<1:0> value decrements on every read or write of ALRMVALH until it reaches '00'.
ALRMVAL<15:8>:
11 = Unimplemented
10 = ALRMMNTH
01 = ALRMWD
00 = ALRMMIN
ALRMVAL<7:0>:
11 = Unimplemented
10 = ALRMDAY
01 = ALRMHR
00 = ALRMSEC

bit 7-0 **ARPT<7:0>:** Alarm Repeat Counter Value bits
11111111 = Alarm will repeat 255 more times
...
00000000 = Alarm will not repeat
The counter decrements on any alarm event. The counter is prevented from rolling over from 00h to FFh unless CHIME = 1.

21.1.3 DATA SHIFT DIRECTION

The LENDIAN bit (CRCCON1<3>) is used to control the shift direction. By default, the CRC will shift data through the engine, MSb first. Setting LENDIAN (= 1) causes the CRC to shift data, LSb first. This setting allows better integration with various communication schemes and removes the overhead of reversing the bit order in software. Note that this only changes the direction the data is shifted into the engine. The result of the CRC calculation will still be a normal CRC result, not a reverse CRC result.

21.1.4 INTERRUPT OPERATION

The module generates an interrupt that is configurable by the user for either of two conditions.

If CRCISEL is '0', an interrupt is generated when the VWORD<4:0> bits make a transition from a value of '1' to '0'. If CRCISEL is '1', an interrupt will be generated after the CRC operation finishes and the module sets the CRCGO bit to '0'. Manually setting CRCGO to '0' will not generate an interrupt. Note that when an interrupt occurs, the CRC calculation would not yet be complete. The module will still need (PLEN + 1)/2 clock cycles after the interrupt is generated until the CRC calculation is finished.

21.1.5 TYPICAL OPERATION

To use the module for a typical CRC calculation:

1. Set the CRCEN bit to enable the module.
2. Configure the module for desired operation:
   a) Program the desired polynomial using the CRCXORL and CRCXORH registers, and the PLEN<4:0> bits.
   b) Configure the data width and shift direction using the DWIDTH and LENDIAN bits.
   c) Select the desired interrupt mode using the CRCISEL bit.
3. Preload the FIFO by writing to the CRCDATL and CRCDATH registers until the CRCFUL bit is set or no data is left.
4. Clear old results by writing 00h to CRCWDATL and CRCWDATH. The CRCWDAT registers can also be left unchanged to resume a previously halted calculation.
5. Set the CRCGO bit to start calculation.
6. Write remaining data into the FIFO as space becomes available.
7. When the calculation completes, CRCGO is automatically cleared. An interrupt will be generated if CRCISEL = 1.
8. Read CRCWDATL and CRCWDATH for the result of the calculation.

There are eight registers used to control programmable CRC operation:

- CRCCON1
- CRCCON2
- CRCXORL
- CRCXORH
- CRCDATL
- CRCDATH
- CRCWDATL
- CRCWDATH

The CRCCON1 and CRCCON2 registers (Register 21-1 and Register 21-2) control the operation of the module and configure the various settings.

The CRCXOR registers (Register 21-3 and Register 21-4) select the polynomial terms to be used in the CRC equation. The CRCDAT and CRCWDAT registers are each register pairs that serve as buffers for the double-word input data, and CRC processed output, respectively.

REGISTER 21-1: CRCCON1: CRC CONTROL 1 REGISTER

R/W-0	U-0	R/W-0	R-0, HSC	R-0, HSC	R-0, HSC	R-0, HSC	R-0, HSC
CRCEN	—	CSIDL	VWORD4	VWORD3	VWORD2	VWORD1	VWORD0
bit 15							bit 8

R-0, HSC	R-1, HSC	R/W-0	R/W-0, HC	R/W-0	U-0	U-0	U-0
CRCFUL	CRCMPT	CRCISEL	CRCGO	LENDIAN	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown
HC = Hardware Cleared	HSC = Hardware Settable/Clearable bit		

bit 15 **CRCEN:** CRC Enable bit
1 = Enables module
0 = Disables module; all state machines, pointers and CRCWDAT/CRCDATH reset; other SFRs are NOT reset

bit 14 **Unimplemented:** Read as '0'

bit 13 **CSIDL:** CRC Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 12-8 **VWORD<4:0>:** Pointer Value bits
Indicates the number of valid words in the FIFO. Has a maximum value of 8 when PLEN<4:0> ≥ 7 or 16 when PLEN<4:0> ≤ 7.

bit 7 **CRCFUL:** FIFO Full bit
1 = FIFO is full
0 = FIFO is not full

bit 6 **CRCMPT:** FIFO Empty bit
1 = FIFO is empty
0 = FIFO is not empty

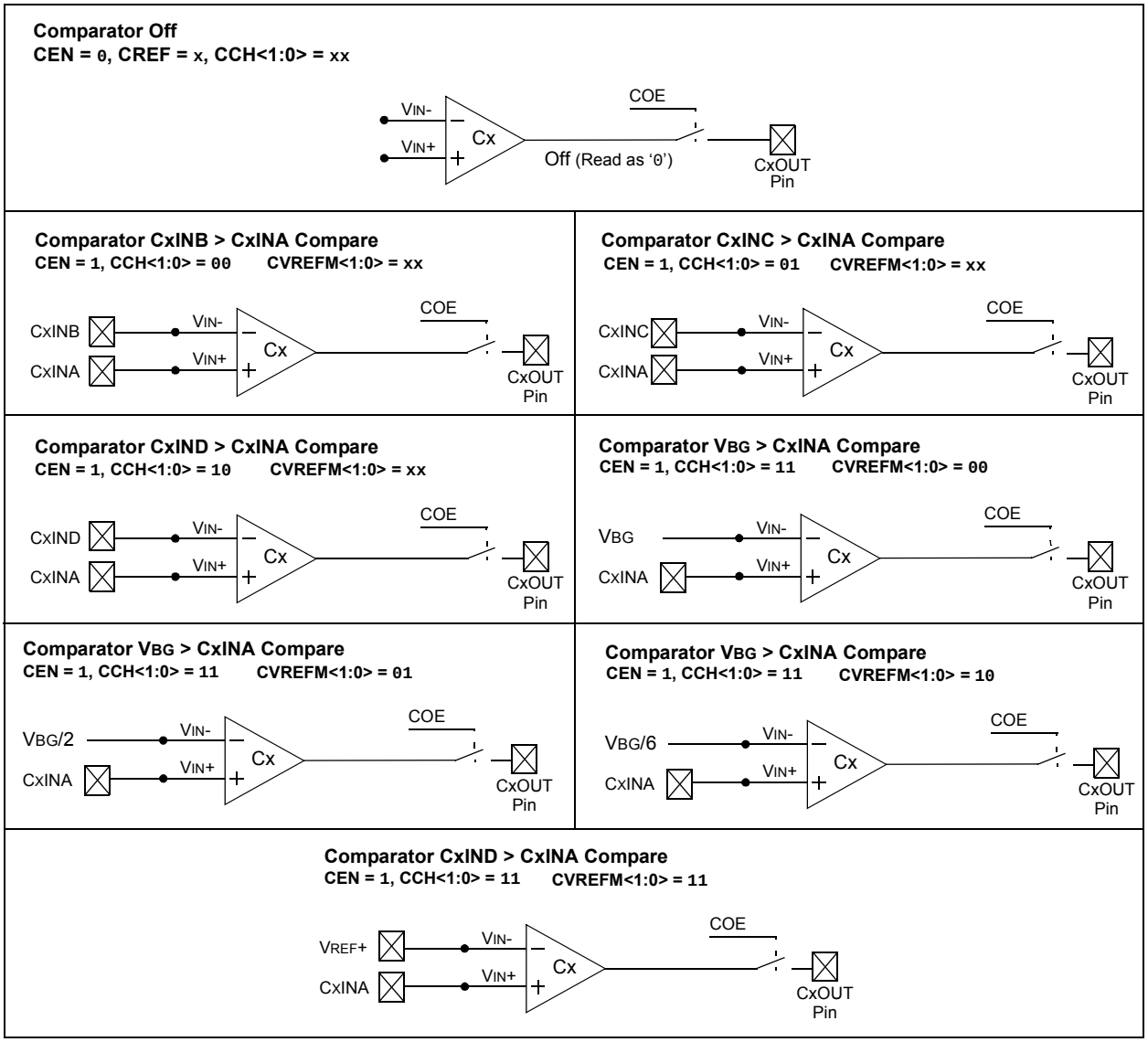
bit 5 **CRCISEL:** CRC Interrupt Selection bit
1 = Interrupt on FIFO is empty; the final word of data is still shifting through the CRC
0 = Interrupt on shift is complete and results are ready

bit 4 **CRCGO:** Start CRC bit
1 = Start CRC serial shifter
0 = CRC serial shifter is turned off

bit 3 **LENDIAN:** Data Shift Direction Select bit
1 = Data word is shifted into the CRC, starting with the LSb (little endian)
0 = Data word is shifted into the CRC, starting with the MSb (big endian)

bit 2-0 **Unimplemented:** Read as '0'

FIGURE 24-2: INDIVIDUAL COMPARATOR CONFIGURATIONS WHEN CREF = 0



REGISTER 26-1: CTMUCON: CTMU CONTROL REGISTER

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CTMUEN	—	CTMUSIDL	TGEN(1)	EDGEN	EDGSEQEN	IDISSEN	CTTRIG
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, HSC	R/W-0, HSC
EDG2POL	EDG2SEL1	EDG2SEL0	EDG1POL	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT
bit 7							bit 0

Legend:	HSC = Hardware Settable/Clearable bit		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **CTMUEN:** CTMU Enable bit
1 = Module is enabled
0 = Module is disabled

bit 14 **Unimplemented:** Read as '0'

bit 13 **CTMUSIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when the device enters Idle mode
0 = Continue module operation in Idle mode

bit 12 **TGEN:** Time Generation Enable bit(1)
1 = Enables edge delay generation
0 = Disables edge delay generation

bit 10 **EDGEN:** Edge Enable bit
1 = Edges are not blocked
0 = Edges are blocked

bit 10 **EDGSEQEN:** Edge Sequence Enable bit
1 = Edge 1 event must occur before Edge 2 event can occur
0 = No edge sequence is needed

bit 9 **IDISSEN:** Analog Current Source Control bit
1 = Analog current source output is grounded
0 = Analog current source output is not grounded

bit 8 **CTTRIG:** Trigger Control bit
1 = Trigger output is enabled
0 = Trigger output is disabled

bit 7 **EDG2POL:** Edge 2 Polarity Select bit
1 = Edge 2 is programmed for a positive edge response
0 = Edge 2 is programmed for a negative edge response

bit 6-5 **EDG2SEL<1:0>:** Edge 2 Source Select bits
11 = CTEDG1 pin
10 = CTEDG2 pin
01 = OC1 module
00 = Timer1 module

bit 4 **EDG1POL:** Edge 1 Polarity Select bit
1 = Edge 1 is programmed for a positive edge response
0 = Edge 1 is programmed for a negative edge response

Note 1: If TGEN = 1, the peripheral inputs and outputs must be configured to an available RPn/RPIn pin. See **Section 10.4 "Peripheral Pin Select (PPS)"** for more information.

28.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers and dsPIC® digital signal controllers are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB C Compiler for Various Device Families
  - HI-TECH C for Various Device Families
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers
  - MPLAB ICD 3
  - PICkit™ 3 Debug Express
- Device Programmers
  - PICkit™ 2 Programmer
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits, and Starter Kits

28.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16/32-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)
  - In-Circuit 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
- 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 IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either C or assembly)
- One-touch compile or assemble, and download to emulator and simulator tools (automatically updates all project information)
- Debug using:
  - Source files (C or assembly)
  - Mixed C and assembly
  - 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.

TABLE 30-13: EXTERNAL CLOCK TIMING REQUIREMENTS

AC CHARACTERISTICS			Standard Operating Conditions: 2.2V 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
OS10	FOSC	External CLKI Frequency (External clocks allowed only in EC mode)	DC 4	— —	32 48	MHz MHz	EC ECPLL
		Oscillator Frequency	3.5 4 10 10 31	— — — — —	10 8 32 32 33	MHz MHz MHz MHz kHz	XT XTPLL HS HSPLL SOSC
OS20	TOSC	TOSC = 1/FOSC	—	—	—	—	See parameter OS10 for FOSC value
OS25	TCY	Instruction Cycle Time(2)	62.5	—	DC	ns	
OS30	TosL, TosH	External Clock in (OSCI) High or Low Time	0.45 x TOSC	—	—	ns	EC
OS31	TosR, TosF	External Clock in (OSCI) Rise or Fall Time	—	—	20	ns	EC
OS40	TckR	CLKO Rise Time(3)	—	6	10	ns	
OS41	TckF	CLKO Fall Time(3)	—	6	10	ns	

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 two times 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 OSCI/CLKI pin. When an external clock input is used, the "Max." cycle time limit is "DC" (no clock) for all devices.

3: Measurements are taken in EC mode. The CLKO signal is measured on the OSCO pin. CLKO is low for the Q1-Q2 period (1/2 TCY) and high for the Q3-Q4 period (1/2 TCY).

TABLE 30-14: PLL CLOCK TIMING SPECIFICATIONS (VDD = 2.2V TO 3.6V)

AC CHARACTERISTICS			Standard Operating Conditions: 2.2V 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 Input Frequency Range(2)	4	—	48	MHz	ECPLL mode
			4		32	MHz	HSPLL mode
			4		8	MHz	XTPLL mode
OS51	FSYS	PLL Output Frequency Range	95.76	—	96.24	MHz	
OS52	TLOCK	PLL Start-up Time (Lock Time)	—	—	200	µs	
OS53	DCLK	CLKO Stability (Jitter)	-0.25	—	0.25	%	

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.