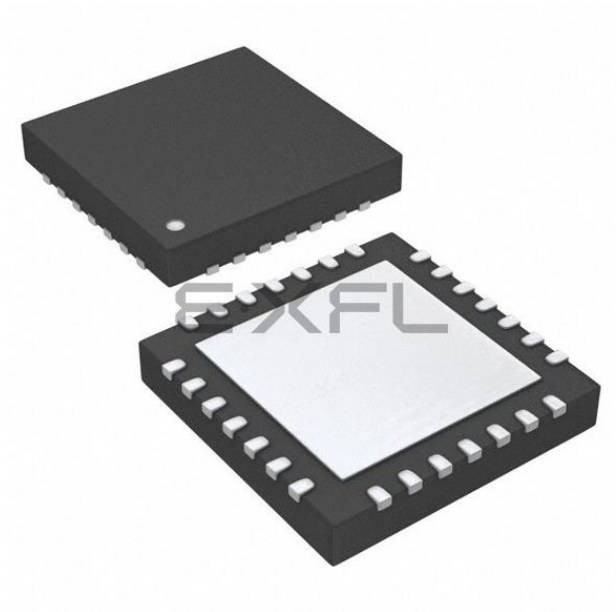


Welcome to E-XFL.COM

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	8-Bit
Speed	64MHz
Connectivity	CANbus, I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	25
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 24x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-VQFN Exposed Pad
Supplier Device Package	28-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf26k83-i-ml

TABLE 4-9: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F25/26K83 DEVICES BANK 57

39FFh	—	39DFh	OSCFRQ	39BFh	—	399Fh	—	397Fh	—	395Fh	WDTU	393Fh	—	391Fh	—
39FEh	—	39DEh	OSCTUNE	39BEh	—	399Eh	—	397Eh	—	395Eh	WDTH	393Eh	—	391Eh	—
39FDh	—	39DDh	OSCEN	39BDh	—	399Dh	—	397Dh	SCANTRIG	395Dh	WDTL	393Dh	—	391Dh	—
39FCh	—	39DCh	OSCSTAT	39BCh	—	399Ch	—	397Ch	SCANCON0	395Ch	WDTCON1	393Ch	—	391Ch	—
39FBh	—	39DBh	OSCCON3	39BBh	—	399Bh	—	397Bh	SCANHADRU	395Bh	WDTCON0	393Bh	—	391Bh	—
39FAh	—	39DAh	OSCCON2	39BAh	—	399Ah	—	397Ah	SCANHADRH	395Ah	—	393Ah	—	391Ah	—
39F9h	—	39D9h	OSCCON1	39B9h	—	3999h	PIE9	3979h	SCANHADRL	3959h	—	3939h	—	3919h	—
39F8h	—	39D8h	CPUDOZE	39B8h	—	3998h	PIE8	3978h	SCANLADRU	3958h	—	3938h	—	3918h	—
39F7h	SCANPR	39D7h	—	39B7h	—	3997h	PIE7	3977h	SCANLADRH	3957h	—	3937h	—	3917h	—
39F6h	—	39D6h	—	39B6h	—	3996h	PIE6	3976h	SCANLADRL	3956h	—	3936h	—	3916h	—
39F5h	—	39D5h	—	39B5h	—	3995h	PIE5	3975h	—	3955h	—	3935h	—	3915h	—
39F4h	DMA2PR	39D4h	—	39B4h	—	3994h	PIE4	3974h	—	3954h	—	3934h	—	3914h	—
39F3h	DMA1PR	39D3h	—	39B3h	—	3993h	PIE3	3973h	—	3953h	—	3933h	—	3913h	—
39F2h	MAINPR	39D2h	—	39B2h	—	3992h	PIE2	3972h	—	3952h	—	3932h	—	3912h	—
39F1h	ISRPR	39D1h	VREGCON(1)	39B1h	—	3991h	PIE1	3971h	—	3951h	—	3931h	—	3911h	—
39F0h	—	39D0h	BORCON	39B0h	—	3990h	PIE0	3970h	—	3950h	—	3930h	—	3910h	—
39EFh	PRLOCK	39CFh	—	39AFh	—	398Fh	—	396Fh	—	394Fh	—	392Fh	—	390Fh	—
39EEh	—	39CEh	—	39AEh	—	398Eh	—	396Eh	—	394Eh	—	392Eh	—	390Eh	—
39EDh	—	39CDh	—	39ADh	—	398Dh	—	396Dh	—	394Dh	—	392Dh	—	390Dh	—
39ECh	—	39CCh	—	39ACh	—	398Ch	—	396Ch	—	394Ch	—	392Ch	—	390Ch	—
39EBh	—	39CBh	—	39ABh	—	398Bh	—	396Bh	—	394Bh	—	392Bh	—	390Bh	—
39EAh	—	39CAh	—	39AAh	—	398Ah	—	396Ah	—	394Ah	—	392Ah	—	390Ah	—
39E9h	—	39C9h	—	39A9h	PIR9	3989h	IPR9	3969h	CRCCON1	3949h	—	3929h	—	3909h	—
39E8h	—	39C8h	—	39A8h	PIR8	3988h	IPR8	3968h	CRCCON0	3948h	—	3928h	—	3908h	—
39E7h	—	39C7h	PMD7	39A7h	PIR7	3987h	IPR7	3967h	CRCXORH	3947h	—	3927h	—	3907h	—
39E6h	NVMCON2	39C6h	PMD6	39A6h	PIR6	3986h	IPR6	3966h	CRCXORL	3946h	—	3926h	—	3906h	—
39E5h	NVMCON1	39C5h	PMD5	39A5h	PIR5	3985h	IPR5	3965h	CRCSHIFTH	3945h	—	3925h	—	3905h	—
39E4h	—	39C4h	PMD4	39A4h	PIR4	3984h	IPR4	3964h	CRCSHIFTL	3944h	—	3924h	—	3904h	—
39E3h	NVMDAT	39C3h	PMD3	39A3h	PIR3	3983h	IPR3	3963h	CRCACCH	3943h	—	3923h	—	3903h	—
39E2h	—	39C2h	PMD2	39A2h	PIR2	3982h	IPR2	3962h	CRCACCL	3942h	—	3922h	—	3902h	—
39E1h	—	39C1h	PMD1	39A1h	PIR1	3981h	IPR1	3961h	CRCDATH	3941h	—	3921h	—	3901h	—
39E0h	NVMADRL	39C0h	PMD0	39A0h	PIR0	3980h	IPR0	3960h	CRCDATL	3940h	—	3920h	—	3900h	—

Legend: Unimplemented data memory locations and registers, read as '0'.

Note 1: Unimplemented in LF devices.

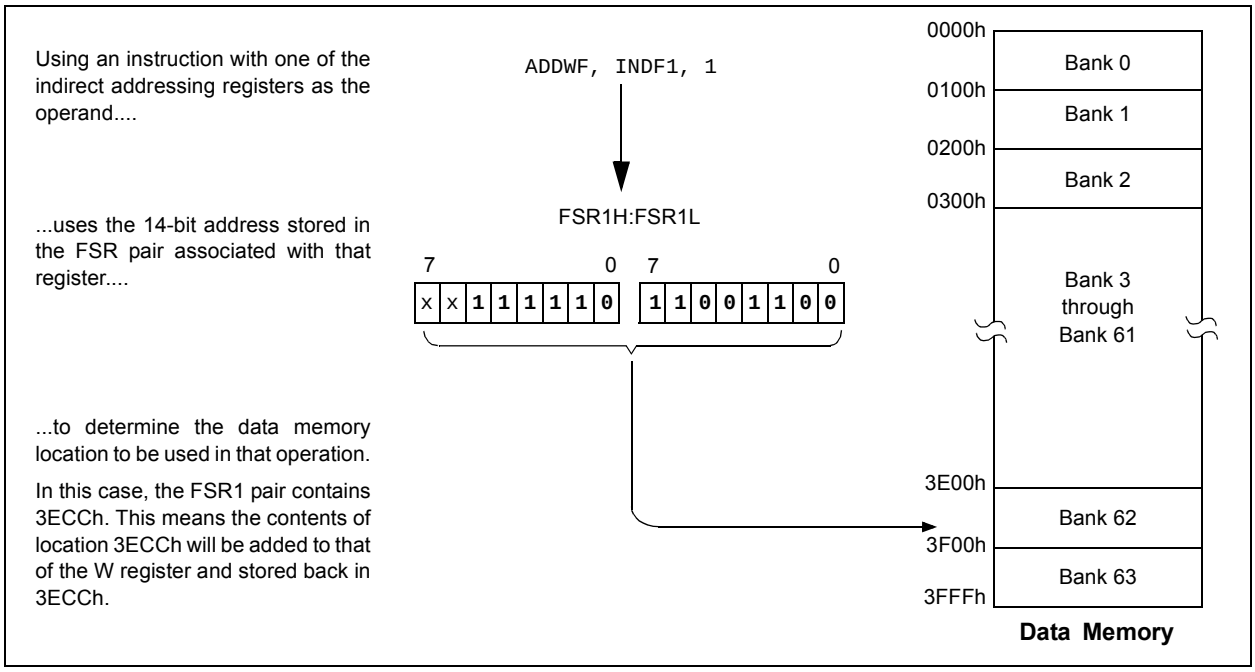
4.7.3.2 FSR Registers, POSTINC, POSTDEC, PREINC and PLUSW

In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are “virtual” registers which cannot be directly read or written. Accessing these registers actually accesses the location to which the associated FSR register pair points, and also performs a specific action on the FSR value. They are:

- POSTDEC: accesses the location to which the FSR points, then automatically decrements the FSR by 1 afterwards
- POSTINC: accesses the location to which the FSR points, then automatically increments the FSR by 1 afterwards
- PREINC: automatically increments the FSR by 1, then uses the location to which the FSR points in the operation
- PLUSW: adds the signed value of the W register (range of -127 to 128) to that of the FSR and uses the location to which the result points in the operation.

In this context, accessing an INDF register uses the value in the associated FSR register without changing it. Similarly, accessing a PLUSW register gives the FSR value an offset by that in the W register; however, neither W nor the FSR is actually changed in the operation. Accessing the other virtual registers changes the value of the FSR register.

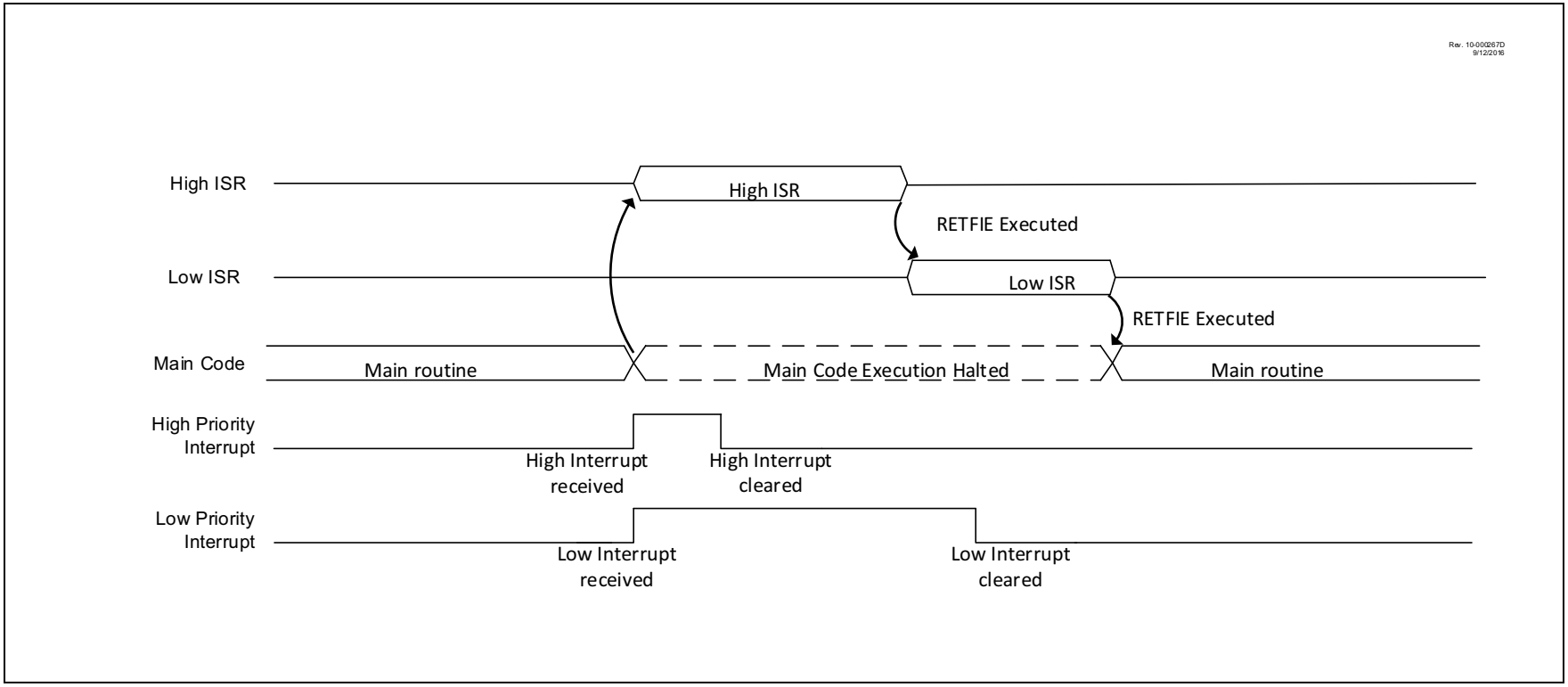
FIGURE 4-6: INDIRECT ADDRESSING



9.4.4 SIMULTANEOUS LOW AND HIGH PRIORITY INTERRUPTS

When both high and low interrupts are active in the same instruction cycle (i.e., simultaneous interrupt events), both the high and the low priority requests are generated. The high priority ISR is serviced first before servicing the low priority interrupt see Figure 9-5.

FIGURE 9-5: INTERRUPT EXECUTION: SIMULTANEOUS LOW AND HIGH PRIORITY INTERRUPTS



9.5 Context Saving

The Interrupt controller supports a two-level deep context saving (Main routine context and Low ISR context). Refer to state machine shown in Figure 9-6 for details.

The Program Counter (PC) is saved on the dedicated device PC stack. CPU registers saved include STATUS, WREG, BSR, FSR0/1/2, PRODL/H and PCLATH/U.

After WREG has been saved to the context registers, the resolved vector number of the interrupt source to be serviced is copied into WREG. Context save and restore operation is completed by the interrupt controller based on current state of the interrupts and the order in which they were sent to the CPU.

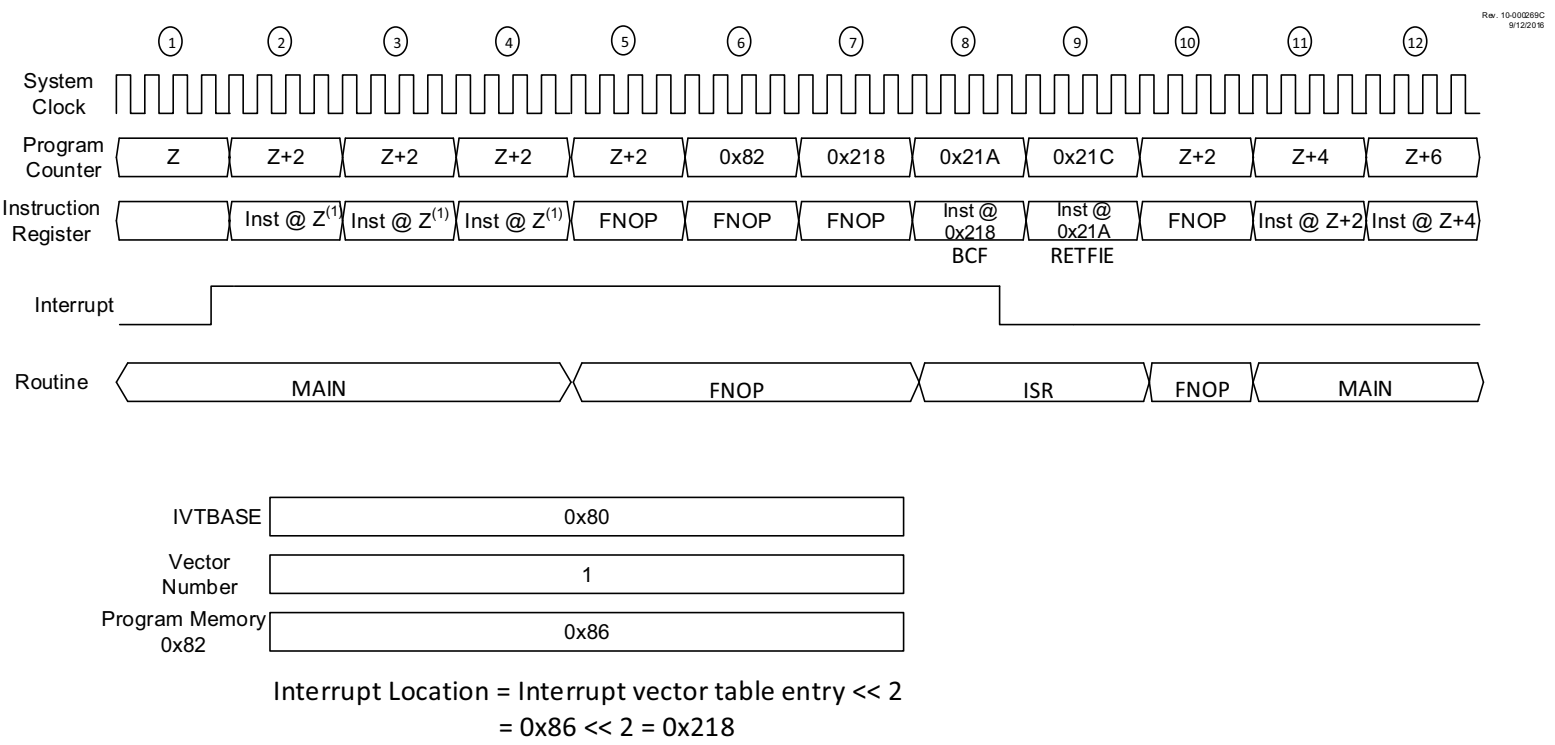
Context save/restore works the same way in both states of MVECEN. When IPEN = 0, there is only one level interrupt active. Hence, only the main context is saved when an interrupt is received.

9.5.1 ACCESSING SHADOW REGISTERS

The Interrupt controller automatically saves the context information in the shadow registers available in Bank 56. Both the saved context values (i.e., main routine and low ISR) can be accessed using the same set of shadow registers. By clearing the SHADLO bit in the SHADCON register (Register 9-40), the CPU register values saved for main routine context can accessed, and by setting the SHADLO bit of the CPU register, values saved for low ISR context can accessed. Low ISR context is automatically restored to the CPU registers upon exiting the high ISR. Similarly, the main context is automatically restored to the CPU registers upon exiting the low ISR.

The Shadow registers in Bank 56 are readable and writable, so if the user desires to modify the context then the corresponding shadow register should be modified and the value will be restored when exiting the ISR. Depending on the user's application, other registers may also need to be saved.

FIGURE 9-9: INTERRUPT TIMING DIAGRAM – THREE CYCLE INSTRUCTION



Note 1: Instruction @ Z is a three-cycle instruction.

REGISTER 9-12: PIR9: PERIPHERAL INTERRUPT REGISTER 9(1)

U-0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0
—	CLC4IF	CCP4IF	CLC3IF	CWG3IF	CCP3IF	TMR6IF	TMR5GIF
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

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

bit 6 **CLC4IF:** CLC4 Interrupt Flag bit
- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

bit 5 **CCP4IF:** CCP4 Interrupt Flag bit
- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

bit 4 **CLC3IF:** CLC3 Interrupt Flag bit
- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

bit 3 **CWG3IF:** CWG3 Interrupt Flag bit
- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

bit 2 **CCP3IF:** CCP3 Interrupt Flag bit
- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

bit 1 **TMR6IF:** TMR6 Interrupt Flag bit
- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

bit 0 **TMR5GIF:** TMR5 Gate Interrupt Flag bit
- 1 = Interrupt has occurred (must be cleared by software)
- 0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

TABLE 9-3: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
INTCON0	GIE/GIEH	GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG	125
INTCON1	STAT<1:0>		—	—	—	—	—	—	126
PIE0	IOCIE	CRCIE	SCANIE	NVMIE	CSWIE	OSFIE	HLVDIE	SWIE	137
PIE1	SMT1PWAIE	SMT1PRAIE	SMT1IE	C1IE	ADTIE	ADIE	ZCDIE	INT0IE	138
PIE2	I2C1RXIE	SPI1IE	SPI1TXIE	SPI1RXIE	DMA1AIE	DMA1ORIE	DMA1DCNTIE	DMA1SCNTIE	139
PIE3	TMR0IE	U1IE	U1EIE	U1TXIE	U1RXIE	I2C1EIE	I2C1IE	I2C1TXIE	140
PIE4	INT1IE	CLC1IE	CWG1IE	NCO1IE	CCP1IE	TMR2IE	TMR1GIE	TMR1IFE	141
PIE5	IRXIE	WAKIE	ERRIE	TXB2IE/TXBnIE	TXB1IE	TXB0IE	RXB1IE/RXBnIE	RXB0IE/FIFOFIE	142
PIE6	DMA2AIE	DMA2ORIE	DMA2DCNTIE	DMA2SCNTIE	SMT2PWAIE	SMT2PRAIE	SMT2IE	C2IE	143
PIE7	U2IE	U2EIE	U2TXIE	U2RXIE	I2C2EIE	I2C2IE	I2C2TXIE	I2C2RXIE	144
PIE8	TMR5IE	INT2IE	CLC2IE	CWG2IE	CCP2IE	TMR4IE	TMR3GIE	TMR3IE	145
PIE9	—	CLC4IE	CCP4IE	CLC3IE	CWG3IE	CCP3IE	TMR6IE	TMR5IE	146
PIR0	IOCIF	CRCIF	SCANIF	NVMIF	CSWIF	OSFIF	HLVDIF	SWIF	127
PIR1	SMT1PWAIF	SMT1PRAIF	SMT1IF	C1IF	ADTIF	ADIF	ZCDIF	INT0IF	128
PIR2	I2C1RXIF	SPI1IF	SPI1TXIF	SPI1RXIF	DMA1AIF	DMA1ORIF	DMA1DCNTIF	DMA1SCNTIF	129
PIR3	TMR0IF	U1IF	U1EIF	U1TXIF	U1RXIF	I2C1EIF	I2C1IF	I2C1TXIF	130
PIR4	INT1IF	CLC1IF	CWG1IF	NCO1IF	CCP1IF	TMR2IF	TMR1GIF	TMR1IF	131
PIR5	IRXIF	WAKIF	ERRIF	TXB2IF/TXBnIF	TXB1IF	TXB0IF	RXB1IF/RXBnIF	RXB0IF/FIFOFIF	132
PIR6	DMA2AIF	DMA2ORIF	DMA2DCNTIF	DMA2SCNTIF	SMT2PWAIF	SMT2PRAIF	SMT2IF	C2IF	133
PIR7	U2IF	U2EIF	U2TXIF	U2RXIF	I2C2EIF	I2C2IF	I2C2TXIF	I2C2RXIF	134
PIR8	TMR5IF	INT2IF	CLC2IF	CWG2IF	CCP2IF	TMR4IF	TMR3GIF	TMR3IF	135
PIR9	—	CLC4IF	CCP4IF	CLC3IF	CWG3IF	CCP3IF	TMR6IF	TMR5IF	136
IPR0	IOCIP	CRCIP	SCANIP	NVMIP	CSWIP	OSFIP	HLVDIP	SWIP	147
IPR1	SMT1PWAIP	SMT1PRAIP	SMT1IP	C1IP	ADTIP	ADIP	ZCDIP	INT0IP	148
IPR2	I2C1RIP	SPI1IP	SPI1TIP	SPI1RIP	DMA1AIP	DMA1ORIP	DMA1DCNTIP	DMA1SCNTIP	149
IPR3	TMR0IP	U1IP	U1EIP	U1TXIP	U1RXIP	I2C1EIP	I2C1IP	I2C1TXIP	150
IPR4	INT1IP	CLC1IP	CWG1IP	NCO1IP	CCP1IP	TMR2IP	TMR1GIP	TMR1IP	151
IPR5	IRXIP	WAKIP	ERRIP	TXB2IP/TXBnIP	TXB1IP	TXB0IP	RXB1IP/RXBnIP	RXB0IP/FIFOFIP	152
IPR6	DMA2AIP	DMA2ORIP	DMA2DCNTIP	DMA2SCNTIP	SMT2PWAIP	SMT2PRAIP	SMT2IP	C2IP	153
IPR7	U2IP	U2EIP	U2TXIP	U2RXIP	I2C2EIP	I2C2IP	I2C2TXIP	I2C2RXIP	154
IPR8	TMR5IP	INT2IP	CLC2IP	CWG2IP	CCP2IP	TMR4IP	TMR3GIP	TMR3IP	155
IPR9	—	CLC4IP	CCP4IP	CLC3IP	CWG3IP	CCP3IP	TMR6IP	TMR5IP	156
IVTBASEU	—	—	—	BASE<20:16>					157
IVTBASEH	BASE<15:8>								157
IVTBASEL	BASE<7:0>								157
IVTADU	—	—	—	AD<20:16>					158
IVTADH	AD<15:8>								158
IVTADL	AD<7:0>								158
IVTLOCK	—	—	—	—	—	—	—	IVTLOCKED	159

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for interrupts.

REGISTER 11-3: WDTPSL: WWDT PRESCALE SELECT LOW BYTE REGISTER (READ-ONLY)

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
PSCNT<7:0>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **PSCNT<7:0>:** Prescale Select Low Byte bits(1)

Note 1: The 18-bit WDT prescale value, PSCNT<17:0> includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT<17:0> is intended for debug operations and should not be read during normal operation.

REGISTER 11-4: WDTPSH: WWDT PRESCALE SELECT HIGH BYTE REGISTER (READ-ONLY)

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
PSCNT<15:8>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **PSCNT<15:8>:** Prescale Select High Byte bits(1)

Note 1: The 18-bit WDT prescale value, PSCNT<17:0> includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT<17:0> is intended for debug operations and should not be read during normal operation.

REGISTER 11-5: WDTTMR: WDT TIMER REGISTER (READ-ONLY)

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
WDTTMR<4:0>					STATE	PSCNT<17:16>	
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-3 **WDTTMR<4:0>:** Watchdog Window Value bits

WINDOW	WDT Window State		Open Percent
	Closed	Open	
111	N/A	00000-11111	100
110	00000-00011	00100-11111	87.5
101	00000-00111	01000-11111	75
100	00000-01011	01100-11111	62.5
011	00000-01111	10000-11111	50
010	00000-10011	10100-11111	37.5
001	00000-10111	11000-11111	25
000	00000-11011	11100-11111	12.5

bit 2 **STATE:** WDT Armed Status bit
1 = WDT is armed
0 = WDT is not armed

bit 1-0 **PSCNT<17:16>:** Prescale Select Upper Byte bits(1)

Note 1: The 18-bit WDT prescale value, PSCNT<17:0> includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT<17:0> is intended for debug operations and should not be read during normal operation.

30.1 DSM Operation

The DSM module can be enabled by setting the EN bit in the MD1CON0 register. Clearing the EN bit in the MD1CON0 register, disables the DSM module output and switches the carrier high and carrier low signals to the default option of MD1CARHPPS and MD1CARLPPS, respectively. The modulator signal source is also switched to the BIT in the MD1CON0 register.

The values used to select the carrier high, carrier low, and modulator sources held by the Modulation Source, Modulation High Carrier, and Modulation Low Carrier control registers are not affected when the EN bit is cleared and the DSM module is disabled. The values inside these registers remain unchanged while the DSM is inactive. The sources for the carrier high, carrier low and modulator signals will once again be selected when the EN bit is set and the DSM module is again enabled and active.

30.2 Modulator Signal Sources

The modulator signal can be supplied from the sources specified in Table 30-3.

The modulator signal is selected by configuring the MS<4:0> bits in the MD1SRC register.

30.3 Carrier Signal Sources

The carrier high signal and carrier low signal can be supplied from the sources specified in Table 30-1.

The carrier high signal is selected by configuring the CH<4:0> bits in the MD1CARH register. The carrier low signal is selected by configuring the CL<4:0> bits in the MD1CARL register.

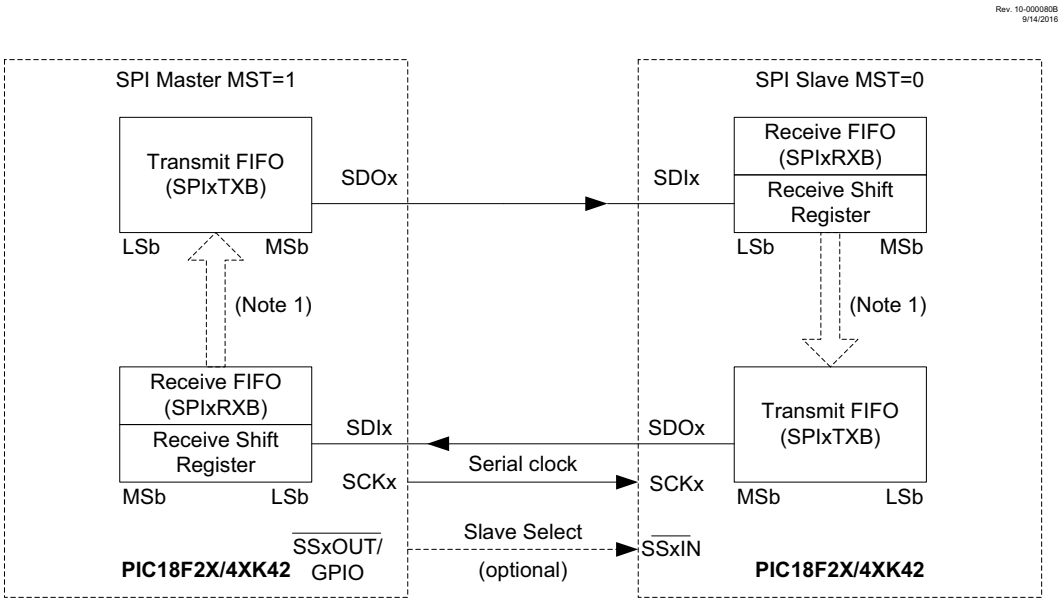
30.4 Carrier Synchronization

During the time when the DSM switches between carrier high and carrier low signal sources, the carrier data in the modulated output signal can become truncated. To prevent this, the carrier signal can be synchronized to the modulator signal. When synchronization is enabled, the carrier pulse that is being mixed at the time of the transition is allowed to transition low before the DSM switches over to the next carrier source.

Synchronization is enabled separately for the carrier high and carrier low signal sources. Synchronization for the carrier high signal is enabled by setting the CHSYNC bit in the MD1CON1 register. Synchronization for the carrier low signal is enabled by setting the CLSYNC bit in the MD1CON1 register.

Figure 30-2 through Figure 30-6 show timing diagrams of using various synchronization methods.

FIGURE 32-2: SPI MASTER/SLAVE CONNECTION WITH FIFOs



Note 1: In some modes, if the Transmit FIFO is empty, the most recently received byte of data will be transmitted

2: This diagram assumes that the LSBF bit is cleared (communications are MSb-first). If LSBF is set, the communications will be LSb-first.

32.3.1 ENABLING AND DISABLING THE SPI MODULE

To enable the serial peripheral, the SPI enable bit (EN in SPIxCON0) must be set. To reset or reconfigure SPI mode, clear the EN bit, re-initialize the SSPxCONx registers and then set the EN bit. Setting the EN bit enables the SPI inputs and outputs: SDI, SDO, SCK(out), SCK(in), SS(out), and SS(in). All of these inputs and outputs are steered by PPS, and thus must have their functions properly mapped to device pins to function (see **Section 17.0 "Peripheral Pin Select (PPS) Module"**). In addition, SS(out) and SCK(out) must have the pins they are steered to set as outputs (TRIS bits must be '0') in order to properly output. Clearing the TRIS bit of the SDO pin will cause the SPI module to always control that pin, but is not necessary for SDO functionality. (see **Section 32.3.5 "Input and Output Polarity Bits"**). Configurations selected by the following registers should not be changed while the EN bit is set:

- SPIxBAUD
- SPIxCON1
- SPIxCON0 (except to clear the EN bit)

Clearing the EN bit aborts any transmissions in progress, disables the setting of interrupt flags by hardware, and resets the FIFO occupancy (see **Section 32.3.3 "Transmit and Receive FIFOs"** for more FIFO details).

32.3.2 BUSY BIT

While a data transfer is in progress, the SPI module sets the BUSY bit of SPIxCON2. This bit can be polled by the user to determine the current status of the SPI module, and to know when a communication is complete. The following registers/bits should not be written by software while the BUSY bit is set:

- SPIxTCNTH/L
- SPIxTWIDTH
- SPIxCON2
- The CLRBF bit of SPIxSTATUS

Note:	It is also not recommended to read SPIx-TCNTH/L while the BUSY bit is set, as the value in the registers may not be a reliable indicator of the Transfer Counter. Use the Transfer Count Zero Interrupt Flag (the TCZIF bit of SPIxINTF) to accurately determine that the Transfer Counter has reached zero.

32.3.3 TRANSMIT AND RECEIVE FIFOS

The transmission and reception of data from the SPI module is handled by two FIFOs, one for reception and one for transmission (addressed by the SFRs SPIxRXB and SPIxTXB, respectively.). The TXFIFO is written by software and is read by the SPI module to shift the data onto the SDO pin. The RXFIFO is written by the SPI module as it shifts in the data from the SDI pin and is read by software. Setting the CLRBF bit of SPIxSTATUS resets the occupancy for both FIFOs, emptying both buffers. The FIFOs are also reset by disabling the SPI module.

Note:	TXFIFO occupancy and RXFIFO occupancy simply refer to the number of bytes that are currently being stored in each FIFO. These values are used in this chapter to illustrate the function of these FIFOs and are not directly accessible through software.

The SPIxRXB register addresses the receive FIFO and is read-only. Reading from this register will read from the first FIFO location that was written to by hardware and decrease the RXFIFO occupancy. If the FIFO is empty, reading from this register will instead return a value of zero and set the RXRE (Receive Buffer Read Error) bit of the SPIxSTATUS register. The RXRE bit must then be cleared in software in order to properly reflect the status of the read error. When RXFIFO is full, the RXBF bit of the SPIxSTATUS register will be set. When the device receives data on the SDI pin, the receive FIFO may be written to by hardware and the occupancy increased, depending on the mode and receiver settings, as summarized in Table 32-1.

The SPIxTXB register addresses the transmit FIFO and is write-only. Writing to the register will write to the first empty FIFO location and increase the occupancy. If the FIFO is full, writing to this register will not affect the data and will set the TXWE bit of the SPIxSTATUS register. When the TXFIFO is empty, the TXBE bit of SPIxSTATUS will be set. When a data transfer occurs, data may be read from the first FIFO location written to and the occupancy decreases, depending on mode and transmitter settings, as summarized in Table 32-1 and **Section 32.6.1 "Slave Mode Transmit options"**.

32.3.4 LSB VS. MSB-FIRST OPERATION

Typically, SPI communication is output Most-Significant bit first, but some devices/buses may not conform to this standard. In this case, the LSBF bit may be used to alter the order in which bits are shifted out during the data exchange. In both Master and Slave mode, the LSBF bit of SPIxCON0 controls if data is shifted MSb or LSb first. Clearing the bit (default) configures the data to transfer MSb first, which is traditional SPI operation, while setting the bit configures the data to transfer LSb first.

REGISTER 32-4: SPIxTCNTH: SPI TRANSFER COUNTER MSB REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	TCNT10	TCNT9	TCNT8
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'

bit 7-3 **Unimplemented**: Read as '0'

bit 2-0 **TCNT<10:8>**:

BMODE = 0

Bits 13-11 of the Transfer Counter, counting the total number of bits to transfer

BMODE = 1

Bits 10-8 of the Transfer Counter, counting the total number of bytes to transfer

Note: This register should not be written to while a transfer is in progress (BUSY bit of SPIxCON2 is set).

REGISTER 32-5: SPIxTWIDTH: SPI TRANSFER WIDTH REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	TWIDTH2	TWIDTH1	TWIDTH0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'

bit 7-3 **Unimplemented**: Read as '0'

bit 2-0 **TWIDTH<2:0>**:

BMODE = 0

Bits 2-0 of the Transfer Counter, counting the total number of bits to transfer

BMODE = 1

Size (in bits) of each transfer counted by the transfer counter

111 = 7 bits

110 = 6 bits

101 = 5 bits

100 = 4 bits

011 = 3 bits

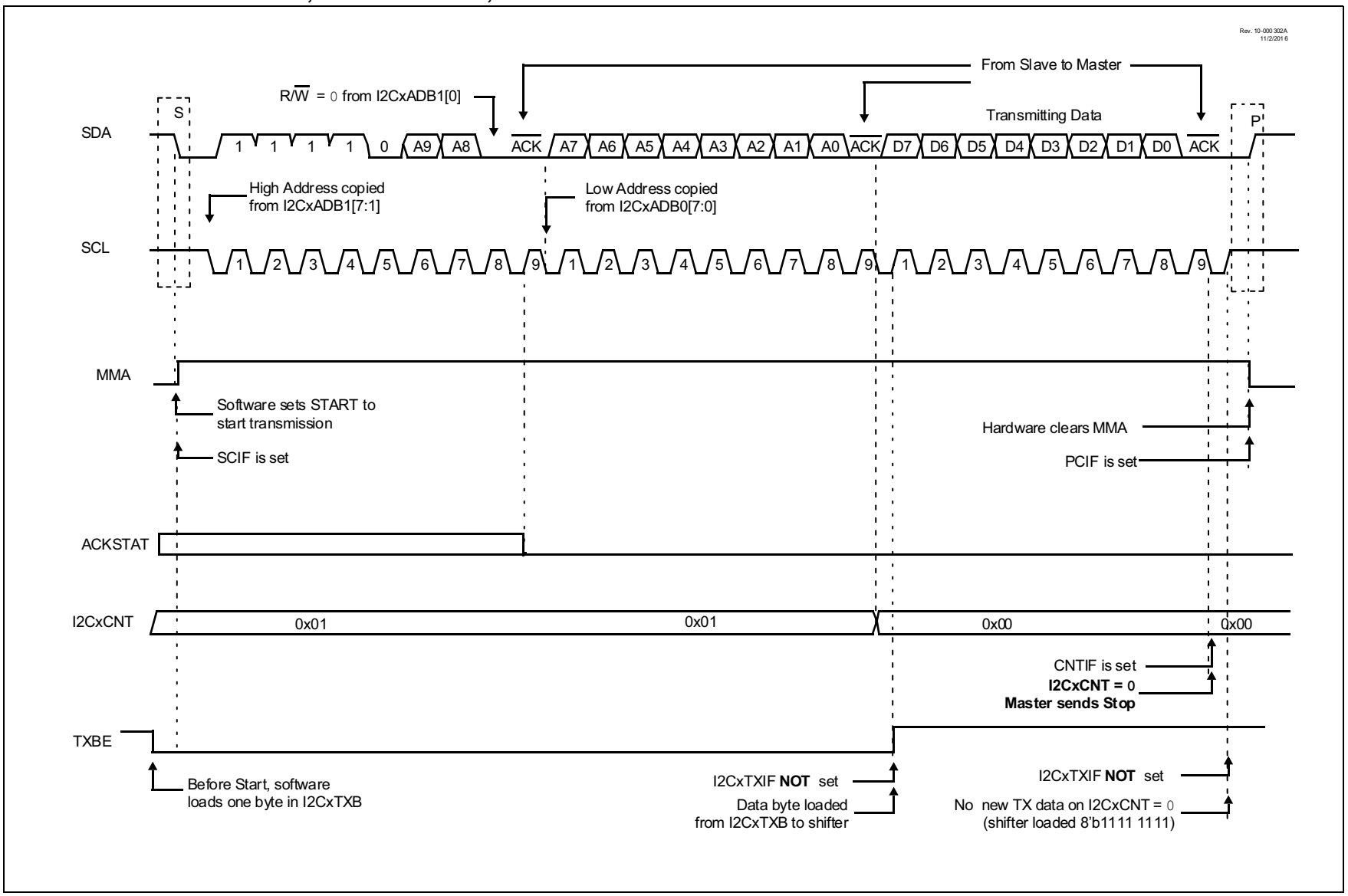
010 = 2 bits

001 = 1 bit

000 = 8 bits

Note: This register should not be written to while a transfer is in progress (BUSY bit of SPIxCON2 is set).

FIGURE 33-21: I²C MASTER, 10-BIT ADDRESS, TRANSMISSION WITH STOP



REGISTER 33-11: I2CxPIE: I2CxIE INTERRUPT AND HOLD ENABLE REGISTER

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNTIE	ACKTIE	—	WRIE	ADRIE	PCIE	RSCIE	SCIE
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	HS = Hardware set HC = Hardware clear

bit 7 **CNTIE:** Byte Count Interrupt Enable bit
1 = When CNTIF is set
0 = Byte count interrupts are disabled

bit 6 **ACKTIE:** Acknowledge Interrupt and Hold Enable bit
1 = When ACKTIF is set
If ACK is generated, CSTR is also set.
If NACK is generated, CSTR is unchanged
0 = Acknowledge holding and interrupt is disabled

bit 5 **Unimplemented:** Read as '0'

bit 4 **WRIE:** Data Write Interrupt and Hold Enable bit
1 = When WRIF is set; CSTR is set
0 = Data Write holding and interrupt is disabled

bit 3 **ADRIE:** Address Interrupt and Hold Enable bit
1 = When ADRIF is set; CSTR is set
0 = Address holding and interrupt is disabled

bit 2 **PCIE:** Stop Condition Interrupt Enable bit
1 = Enable interrupt on detection of Stop condition
0 = Stop detection interrupts are disabled

bit 1 **RSCIE:** Restart Condition Interrupt Enable bit
1 = Enable interrupt on detection of Restart condition
0 = Start detection interrupts are disabled

bit 0 **SCIE:** Start Condition Interrupt Enable bit
1 = Enable interrupt on detection of Start condition
0 = Start detection interrupts are disabled

Note 1: Enabled interrupt flags are OR'd to produce the PIRx<I2CxIF> bit.

REGISTER 33-13: I2CxADR1: I²C ADDRESS 1 REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0
ADR14	ADR13	ADR12	ADR11	ADR10	ADR9	ADR8	—
bit 7							bit 0
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0
ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	—
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	HS = Hardware set HC = Hardware clear

bit 7-1 **ADR[7-1]:** Address or Divider bits

MODE<2:0> = 000 | 110 - 7-bit Slave/Multi-Master Modes

ADR<7:1>:7-bit Slave Address

ADR<0>: Unused in this mode; bit state is a "don't care"

MODE<2:0> = 001 | 111 - 7-bit Slave/Multi-Master modes w/Masking

MSK0<7:1>:7-bit Slave Address

MSK0<0>: Unused in this mode; bit state is a "don't care"

MODE<2:0> = 01x - 10-bit Slave Modes

ADR<14-10>: Bit pattern sent by master is fixed by I²C specification and must be equal to '11110'. However, these bit values are compared by hardware to the received data to determine a match. It is up to the user to set these bits as '11110'.

ADR<9-8>: Two Most Significant bits of 10-bit address

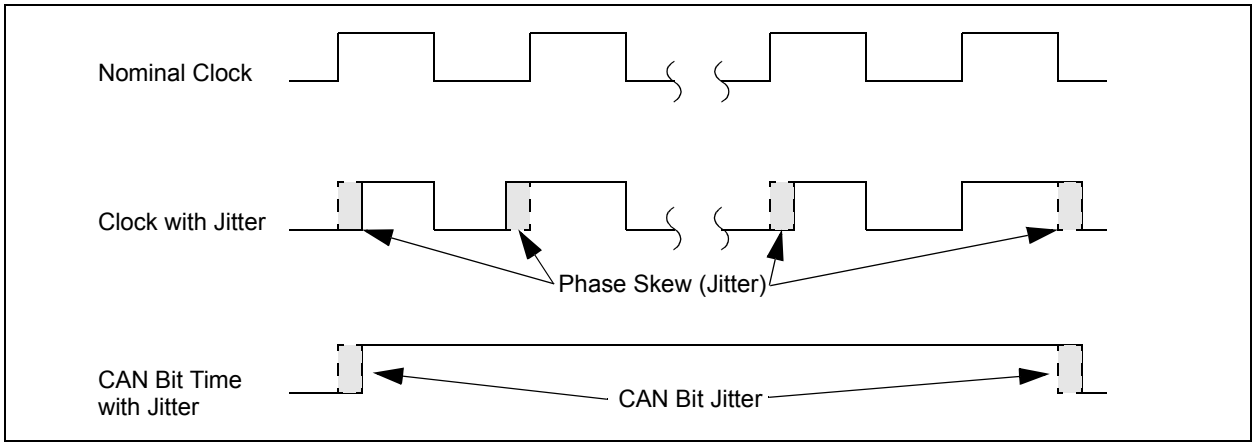
bit 0 **Unimplemented:** Read as '0'.

TABLE 33-18: SUMMARY OF REGISTERS FOR I²C 8-BIT MACRO

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
I2CxBTO	—	—	—	—	—	BTO<2:0>			567
I2CxCLK	—	—	—	—	—	CLK<2:0>			566
I2CxPIE	CNTIE	ACKTIE	—	WRIE	ADRIE	PCIE	RSCIE	SCIE	573
I2CxPIR	CNTIF	ACKTIF	—	WRIF	ADRIF	PCIF	RSCIF	SCIF	572
I2CxERR	—	BTOIF	BCLIF	NACKIF	—	BTOIE	BCLIE	NACKIE	570
I2CxSTAT0	BFRE	SMA	MMA	R	D	—	—	—	568
I2CxSTAT1	TXWE	—	TXBE	—	RXRE	CLRBF	—	RXBF	569
I2CxCON0	EN	RSEN	S	CSTR	MDR	MODE<2:0>			562
I2CxCON1	ACKCNT	ACKDT	ACKSTAT	ACKT	—	RXOV	TXU	CSD	564
I2CxCON2	ACNT	GCEN	FME	ADB	SDAHT<3:2>		BFRET<1:0>		565
I2CxADR0	ADR<7:0>								574
I2CxADR1	ADR<7:1>							—	575
I2CxADR2	ADR<7:0>								576
I2CxADR3	ADR<7:1>							—	577
I2CxADB0	ADB<7:0>								578
I2CxADB1	ADB<7:0>								579
I2CxCNT	CNT<7:0>								571
I2CxRXB	RXB<7:0>								—
I2CxTXB	TXB<7:0>								—

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the I²C module.

FIGURE 34-5: EFFECTS OF PHASE JITTER ON THE MICROCONTROLLER CLOCK AND CAN BIT TIME



Once these considerations are taken into account, it is possible to show that the relation between the jitter and the total frequency error can be defined as:

EQUATION 34-4: JITTER AND TOTAL FREQUENCY ERROR

$$\Delta f = \frac{T_{jitter}}{10 \times NBT} = \frac{2 \times P_{jitter}}{10 \times NBT}$$

where jitter is expressed in terms of time and NBT is the Nominal Bit Time.

For example, assume a CAN bit rate of 125 Kb/s, which gives an NBT of 8 µs. For a 16 MHz clock generated from a 4x PLL, the jitter at this clock frequency is:

EQUATION 34-5: 16 MHz CLOCK FROM 4x PLL JITTER:

$$2\% \times \frac{1}{16\ MHz} = \frac{0.02}{16 \times 10^6} = 1.25 ns$$

and resultant frequency error is:

EQUATION 34-6: RESULTANT FREQUENCY ERROR:

$$\frac{2 \times (1.25 \times 10^{-9})}{10 \times (8 \times 10^{-6})} = 3.125 \times 10^{-5} = 0.0031\%$$

Table 34-2 shows the relation between the clock generated by the PLL and the frequency error from jitter (measured jitter-induced error of 2%, Gaussian distribution, within three standard deviations), as a percentage of the nominal clock frequency.

This is clearly smaller than the expected drift of a crystal oscillator, typically specified at 100 ppm or 0.01%. If we add jitter to oscillator drift, we have a total frequency drift of 0.0132%. The total oscillator frequency errors for common clock frequencies and bit rates, including both drift and jitter, are shown in Table 34-3.

TABLE 34-2: FREQUENCY ERROR FROM JITTER AT VARIOUS PLL GENERATED CLOCK SPEEDS

PLL Output	P_{jitter}	T_{jitter}	Frequency Error at Various Nominal Bit Times (Bit Rates)			
			8 µs (125 Kb/s)	4 µs (250 Kb/s)	2 µs (500 Kb/s)	1 µs (1 Mb/s)
40 MHz	0.5 ns	1 ns	0.00125%	0.00250%	0.005%	0.01%
24 MHz	0.83 ns	1.67 ns	0.00209%	0.00418%	0.008%	0.017%
16 MHz	1.25 ns	2.5 ns	0.00313%	0.00625%	0.013%	0.025%

44.0 DEVELOPMENT SUPPORT

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

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

44.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

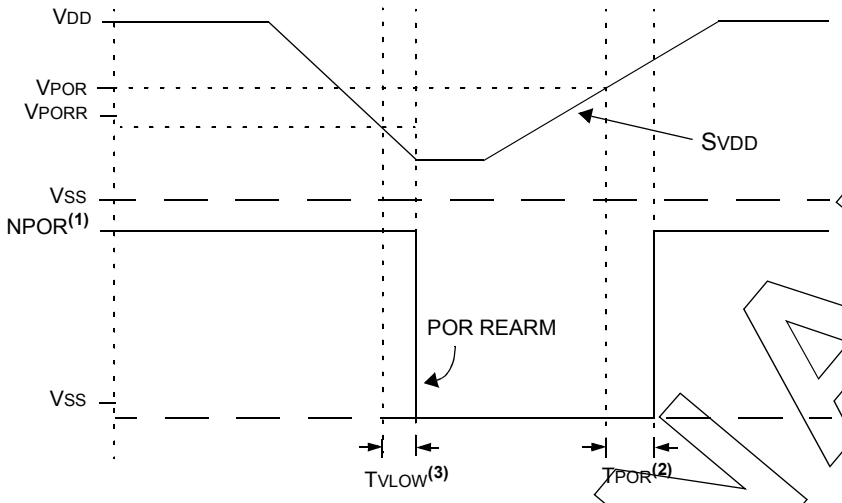
Project-Based Workspaces:

- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker

FIGURE 45-3: POR AND POR REARM WITH SLOW RISING VDD



Note 1: When NPOR is low, the device is held in Reset.
2: TPOR 1 μs typical.
3: TVLOW 2.7 μs typical.