Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

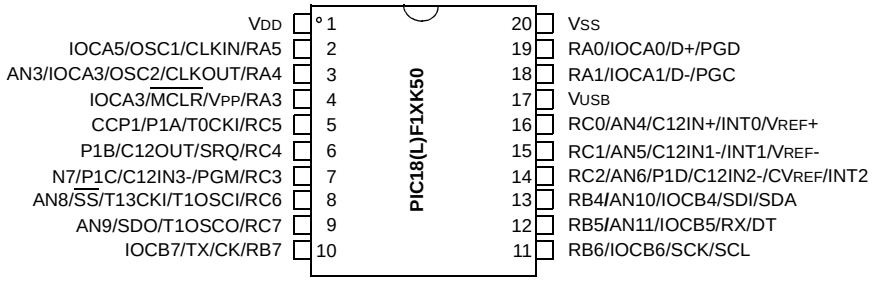
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	48MHz
Connectivity	I²C, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	14
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f14k50-i-so

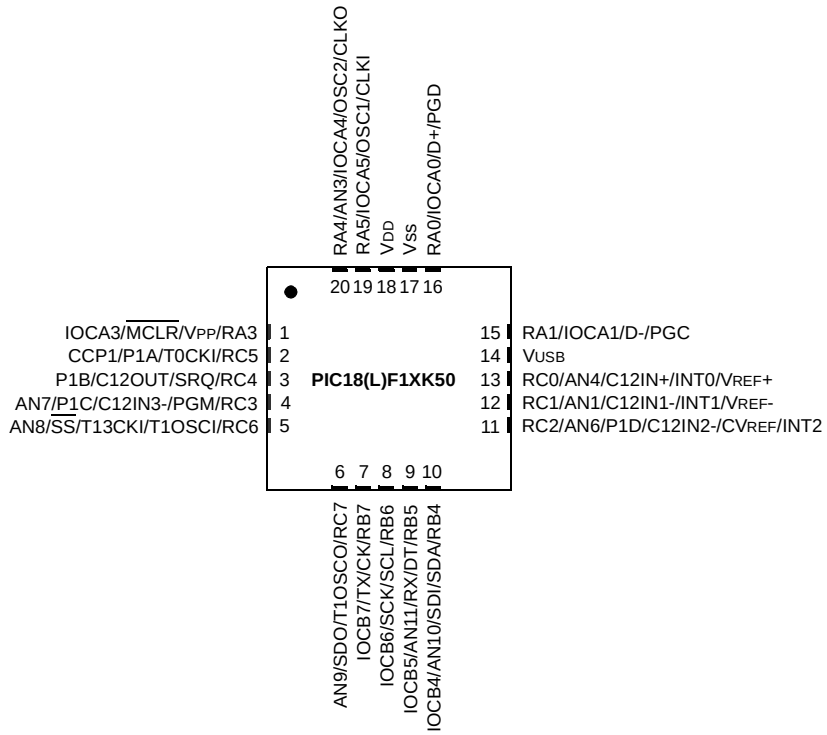
Pin Diagrams

FIGURE 1: 20-PIN PDIP, SSOP, SOIC (300 MIL)



Note: See Table 1 for location of all peripheral functions.

FIGURE 2: 20-PIN QFN (5X5)



Note: See Table 1 for location of all peripheral functions.

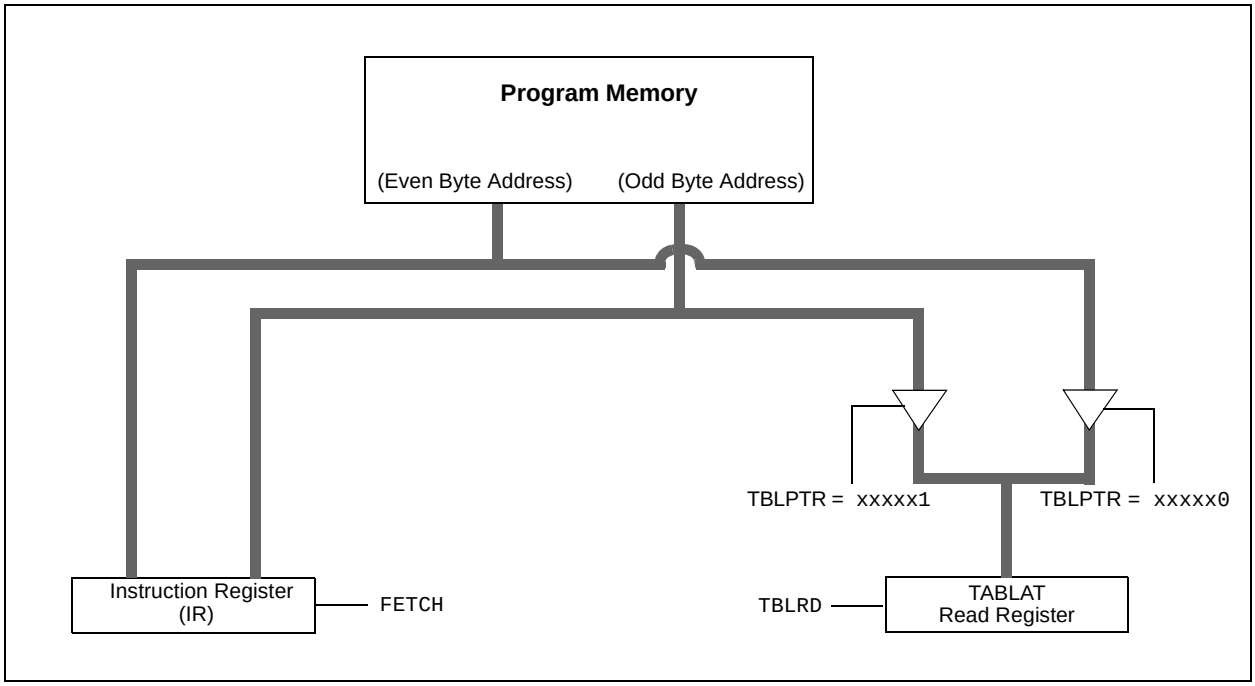
4.3 Reading the Flash Program Memory

The TBLRD instruction retrieves data from program memory and places it into data RAM. Table reads from program memory are performed one byte at a time.

TBLPTR points to a byte address in program space. Executing TBLRD places the byte pointed to into TABLAT. In addition, TBLPTR can be modified automatically for the next table read operation.

The internal program memory is typically organized by words. The Least Significant bit of the address selects between the high and low bytes of the word. Figure 4-4 shows the interface between the internal program memory and the TABLAT.

FIGURE 4-4: READS FROM FLASH PROGRAM MEMORY



EXAMPLE 4-1: READING A FLASH PROGRAM MEMORY WORD

```
            MOVLW     CODE_ADDR_UPPER         ; Load TBLPTR with the base
            MOVWF     TBLPTRU                 ; address of the word
            MOVLW     CODE_ADDR_HIGH
            MOVWF     TBLPTRH
            MOVLW     CODE_ADDR_LOW
            MOVWF     TBLPTRL
READ_WORD
            TBLRD*+                           ; read into TABLAT and increment
            MOVF      TABLAT, W               ; get data
            MOVWF     WORD_EVEN
            TBLRD*+                           ; read into TABLAT and increment
            MOVFW     TABLAT, W               ; get data
            MOVF      WORD_ODD
```

REGISTER 7-7: PIE2: PERIPHERAL INTERRUPT ENABLE (FLAG) REGISTER 2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—
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 7 **OSCFIE:** Oscillator Fail Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **C1IE:** Comparator C1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **C2IE:** Comparator C2 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **EEIE:** Data EEPROM/Flash Write Operation Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **BCLIE:** Bus Collision Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **USBIE:** USB Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **TMR3IE:** TMR3 Overflow Interrupt Enable bit
1 = Enabled
0 = Disabled

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

7.7 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Priority registers (IPR1 and IPR2). Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

REGISTER 7-8: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
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 7 **Unimplemented:** Read as ‘0’

bit 6 **ADIP:** A/D Converter Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **RCIP:** EUSART Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TXIP:** EUSART Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **SSPIP:** Master Synchronous Serial Port Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CCP1IP:** CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR2IP:** TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP:** TMR1 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

9.4 Port Analog Control

Some port pins are multiplexed with analog functions such as the Analog-to-Digital Converter and comparators. When these I/O pins are to be used as analog inputs it is necessary to disable the digital input buffer to avoid excessive current caused by improper biasing of the digital input. Individual control of the digital input buffers on pins which share analog functions is provided by the ANSEL and ANSELH registers. Setting an ANSx bit high will disable the associated digital input buffer and cause all reads of that pin to return '0' while allowing analog functions of that pin to operate correctly.

The state of the ANSx bits has no affect on digital output functions. A pin with the associated TRISx bit clear and ANSx bit set will still operate as a digital output but the Input mode will be analog.

REGISTER 9-15: ANSEL: ANALOG SELECT REGISTER 1

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0	U-0	U-0
ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—
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 7 **ANS7:** RC3 Analog Select Control bit
1 = Digital input buffer of RC3 is disabled
0 = Digital input buffer of RC3 is enabled

bit 6 **ANS6:** RC2 Analog Select Control bit
1 = Digital input buffer of RC2 is disabled
0 = Digital input buffer of RC2 is enabled

bit 5 **ANS5:** RC1 Analog Select Control bit
1 = Digital input buffer of RC1 is disabled
0 = Digital input buffer of RC1 is enabled

bit 4 **ANS4:** RC0 Analog Select Control bit
1 = Digital input buffer of RC0 is disabled
0 = Digital input buffer of RC0 is enabled

bit 3 **ANS3:** RA4 Analog Select Control bit
1 = Digital input buffer of RA4 is disabled
0 = Digital input buffer of RA4 is enabled

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

REGISTER 9-16: ANSELH: ANALOG SELECT REGISTER 2

U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	—	ANS11	ANS10	ANS9	ANS8
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 7-4 **Unimplemented:** Read as ‘0’

bit 3 **ANS11:** RB5 Analog Select Control bit
1 = Digital input buffer of RB5 is disabled
0 = Digital input buffer of RB5 is enabled

bit 2 **ANS10:** RB4 Analog Select Control bit
1 = Digital input buffer of RB4 is disabled
0 = Digital input buffer of RB4 is enabled

bit 1 **ANS9:** RC7 Analog Select Control bit
1 = Digital input buffer of RC7 is disabled
0 = Digital input buffer of RC7 is enabled

bit 0 **ANS8:** RC6 Analog Select Control bit
1 = Digital input buffer of RC6 is disabled
0 = Digital input buffer of RC6 is enabled

13.0 TIMER3 MODULE

The Timer3 module timer/counter incorporates these features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR3H and TMR3L)
- Selectable clock source (internal or external) with device clock or Timer1 oscillator internal options
- Interrupt-on-overflow
- Module Reset on CCP Special Event Trigger

A simplified block diagram of the Timer3 module is shown in Figure 13-1. A block diagram of the module's operation in Read/Write mode is shown in Figure 13-2.

The Timer3 module is controlled through the T3CON register (Register 13-1). It also selects the clock source options for the CCP modules (see **Section 14.1.1 "CCP Module and Timer Resources"** for more information).

REGISTER 13-1: T3CON: TIMER3 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
RD16	—	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
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 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer3 in one 16-bit operation
0 = Enables register read/write of Timer3 in two 8-bit operations

bit 6 **Unimplemented:** Read as '0'

bit 5-4 **T3CKPS<1:0>**: Timer3 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T3CCP1:** Timer3 and Timer1 to CCP1 Enable bits
1 = Timer3 is the clock source for compare/capture of ECCP1
0 = Timer1 is the clock source for compare/capture of ECCP1

bit 2 **T3SYNC:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the device clock comes from Timer1/Timer3.)
When TMR3CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR3CS = 0:
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit
1 = External clock input from Timer1 oscillator or T13CKI (on the rising edge after the first falling edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR3ON:** Timer3 On bit
1 = Enables Timer3
0 = Stops Timer3

13.5 Resetting Timer3 Using the CCP Special Event Trigger

If CCP1 module is configured to use Timer3 and to generate a Special Event Trigger in Compare mode (CCP1M<3:0>), this signal will reset Timer3. It will also start an A/D conversion if the A/D module is enabled (see **Section 17.2.8 "Special Event Trigger"** for more information).

The module must be configured as either a timer or synchronous counter to take advantage of this feature. When used this way, the CCPR1H:CCPR1L register pair effectively becomes a period register for Timer3.

If Timer3 is running in Asynchronous Counter mode, the Reset operation may not work.

In the event that a write to Timer3 coincides with a Special Event Trigger from a CCP module, the write will take precedence.

TABLE 13-1: REGISTERS ASSOCIATED WITH TIMER3 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	275
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	CCP2IF	278
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	CCP2IE	278
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	CCP2IP	278
TMR3L	Timer3 Register, Low Byte								277
TMR3H	Timer3 Register, High Byte								277
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	276
T3CON	RD16	—	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	277
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	278
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	278

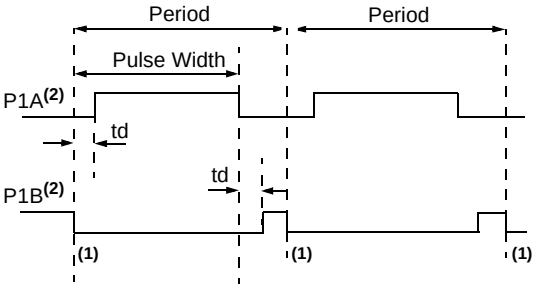
Legend: — = unimplemented, read as '0'. Shaded cells are not used by the Timer3 module.

14.4.6 PROGRAMMABLE DEAD-BAND DELAY MODE

In Half-Bridge applications where all power switches are modulated at the PWM frequency, the power switches normally require more time to turn off than to turn on. If both the upper and lower power switches are switched at the same time (one turned on, and the other turned off), both switches may be on for a short period of time until one switch completely turns off. During this brief interval, a very high current (shoot-through current) will flow through both power switches, shorting the bridge supply. To avoid this potentially destructive shoot-through current from flowing during switching, turning on either of the power switches is normally delayed to allow the other switch to completely turn off.

In Half-Bridge mode, a digitally programmable dead-band delay is available to avoid shoot-through current from destroying the bridge power switches. The delay occurs at the signal transition from the non-active state to the active state. See Figure 14-14 for illustration. The lower seven bits of the associated PWM1CON register (Register 14-3) sets the delay period in terms of microcontroller instruction cycles (T_{CY} or 4 T_{OSC}).

FIGURE 14-14: EXAMPLE OF HALF-BRIDGE PWM OUTPUT

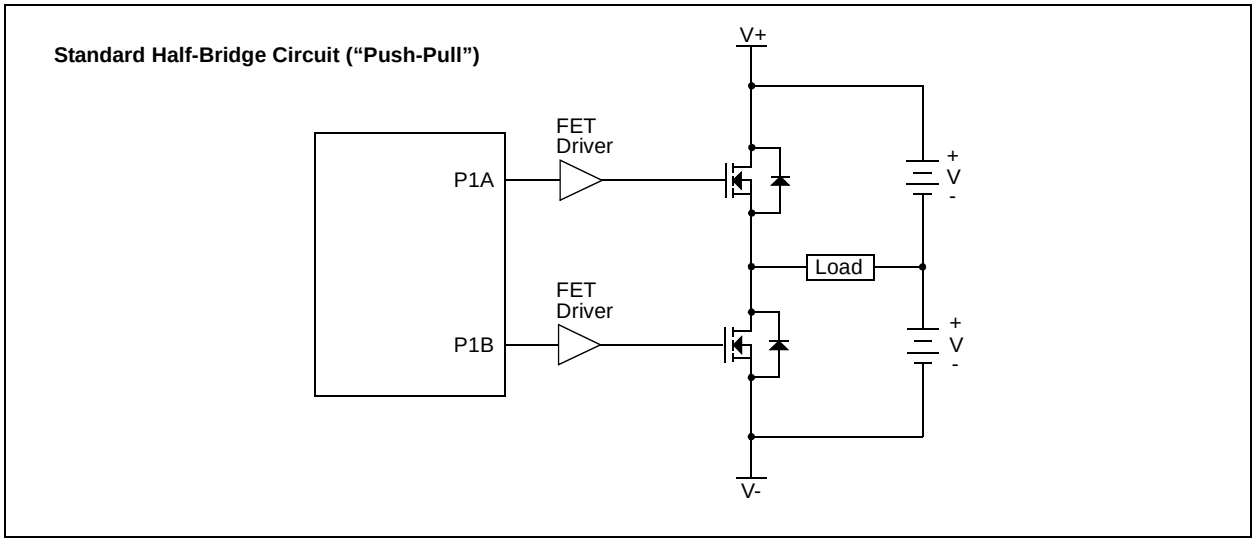


td = Dead-Band Delay

Note 1: At this time, the TMR2 register is equal to the PR2 register.

2: Output signals are shown as active-high.

FIGURE 14-15: EXAMPLE OF HALF-BRIDGE APPLICATIONS



14.4.7 PULSE STEERING MODE

In Single Output mode, pulse steering allows any of the PWM pins to be the modulated signal. Additionally, the same PWM signal can be simultaneously available on multiple pins.

Once the Single Output mode is selected (CCP1M<3:2> = 11 and P1M<1:0> = 00 of the CCP1CON register), the user firmware can bring out the same PWM signal to one, two, three or four output pins by setting the appropriate STR<D:A> bits of the PSTRCON register, as shown in Table 14-2.

Note:	The associated TRIS bits must be set to output ('0') to enable the pin output driver in order to see the PWM signal on the pin.

While the PWM Steering mode is active, CCP1M<1:0> bits of the CCP1CON register select the PWM output polarity for the P1<D:A> pins.

The PWM auto-shutdown operation also applies to PWM Steering mode as described in **Section 14.4.4 "Enhanced PWM Auto-Shutdown mode"**. An auto-shutdown event will only affect pins that have PWM outputs enabled.

REGISTER 14-4: PSTRCON: PULSE STEERING CONTROL REGISTER(1)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
—	—	—	STRSYNC	STRD	STRC	STRB	STRA
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 7-5 **Unimplemented:** Read as '0'

bit 4 **STRSYNC:** Steering Sync bit
1 = Output steering update occurs on next PWM period
0 = Output steering update occurs at the beginning of the instruction cycle boundary

bit 3 **STRD:** Steering Enable bit D
1 = P1D pin has the PWM waveform with polarity control from CCP1M<1:0>
0 = P1D pin is assigned to port pin

bit 2 **STRC:** Steering Enable bit C
1 = P1C pin has the PWM waveform with polarity control from CCP1M<1:0>
0 = P1C pin is assigned to port pin

bit 1 **STRB:** Steering Enable bit B
1 = P1B pin has the PWM waveform with polarity control from CCP1M<1:0>
0 = P1B pin is assigned to port pin

bit 0 **STRA:** Steering Enable bit A
1 = P1A pin has the PWM waveform with polarity control from CCP1M<1:0>
0 = P1A pin is assigned to port pin

Note 1: The PWM Steering mode is available only when the CCP1CON register bits CCP1M<3:2> = 11 and P1M<1:0> = 00.

REGISTER 15-2: SSPCON1: MSSP CONTROL 1 REGISTER (SPI MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
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 7 **WCOL:** Write Collision Detect bit (Transmit mode only)
- 1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared by software)
- 0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit(1)

SPI Slave mode:
- 1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow (must be cleared by software).
- 0 = No overflow

bit 5 **SSPEN:** Synchronous Serial Port Enable bit(2)
- 1 = Enables serial port and configures SCK, SDO, SDI and $\overline{SS}$ as serial port pins
- 0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** Clock Polarity Select bit
- 1 = Idle state for clock is a high level
- 0 = Idle state for clock is a low level

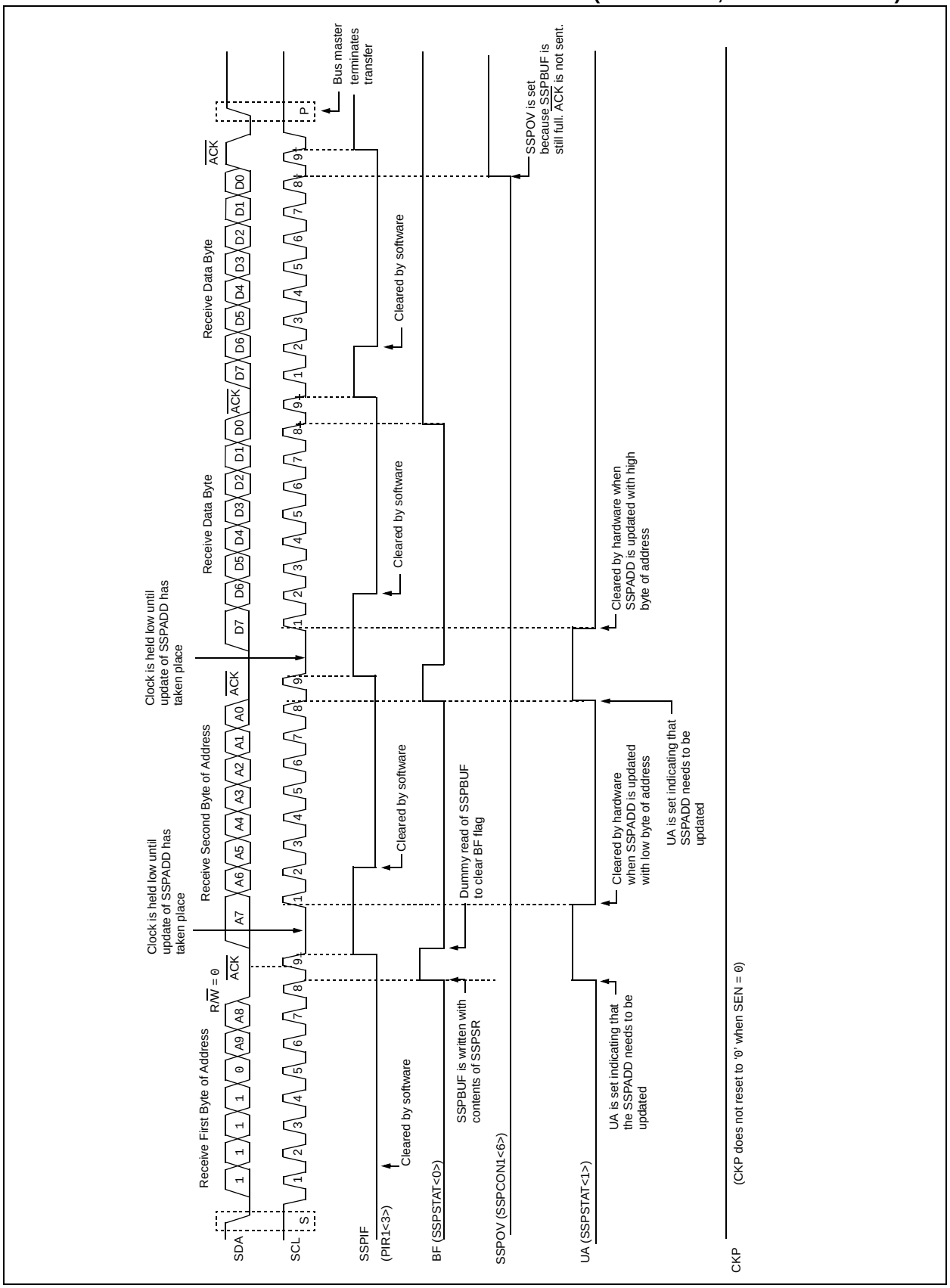
bit 3-0 **SSPM<3:0>:** Synchronous Serial Port Mode Select bits(3)
- 0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin
- 0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled
- 0011 = SPI Master mode, clock = TMR2 output/2
- 0010 = SPI Master mode, clock = FOSC/64
- 0001 = SPI Master mode, clock = FOSC/16
- 0000 = SPI Master mode, clock = FOSC/4

Note 1: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.

2: When enabled, these pins must be properly configured as input or output.

3: Bit combinations not specifically listed here are either reserved or implemented in I^2C mode only.

FIGURE 15-10: I²C™ SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)



16.3.1 AUTO-BAUD DETECT

The EUSART module supports automatic detection and calibration of the baud rate.

In the Auto-Baud Detect (ABD) mode, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RX signal, the RX signal is timing the BRG. The Baud Rate Generator is used to time the period of a received 55h (ASCII “U”) which is the Sync character for the LIN bus. The unique feature of this character is that it has five rising edges including the Stop bit edge.

Setting the ABDEN bit of the BAUDCON register starts the auto-baud calibration sequence (Figure 16-6). While the ABD sequence takes place, the EUSART state machine is held in Idle. On the first rising edge of the receive line, after the Start bit, the SPBRG begins counting up using the BRG counter clock as shown in Table 16-6. The fifth rising edge will occur on the RX pin at the end of the eighth bit period. At that time, an accumulated value totaling the proper BRG period is left in the SPBRGH:SPBRG register pair, the ABDEN bit is automatically cleared, and the RCIF interrupt flag is set. A read operation on the RCREG needs to be performed to clear the RCIF interrupt. RCREG content should be discarded. When calibrating for modes that do not use the SPBRGH register the user can verify that the SPBRG register did not overflow by checking for 00h in the SPBRGH register.

The BRG auto-baud clock is determined by the BRG16 and BRGH bits as shown in Table 16-6. During ABD, both the SPBRGH and SPBRG registers are used as a 16-bit counter, independent of the BRG16 bit setting. While calibrating the baud rate period, the SPBRGH and SPBRG registers are clocked at 1/8th the BRG base clock rate. The resulting byte measurement is the average bit time when clocked at full speed.

Note 1: If the WUE bit is set with the ABDEN bit, auto-baud detection will occur on the byte following the Break character (see **Section 16.3.3 “Auto-Wake-up on Break”**).

2: It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and EUSART baud rates are not possible.

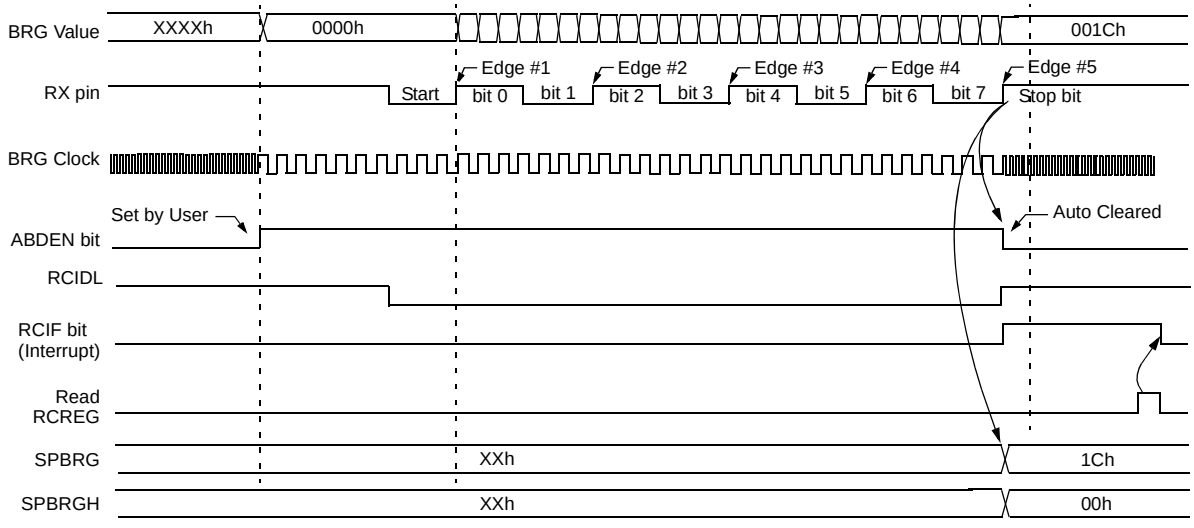
3: During the auto-baud process, the auto-baud counter starts counting at 1. Upon completion of the auto-baud sequence, to achieve maximum accuracy, subtract 1 from the SPBRGH:SPBRG register pair.

TABLE 16-6: BRG COUNTER CLOCK RATES

BRG16	BRGH	BRG Base Clock	BRG ABD Clock
0	0	FOSC/64	FOSC/512
0	1	FOSC/16	FOSC/128
1	0	FOSC/16	FOSC/128
1	1	FOSC/4	FOSC/32

Note: During the ABD sequence, SPBRG and SPBRGH registers are both used as a 16-bit counter, independent of BRG16 setting.

FIGURE 16-6: AUTOMATIC BAUD RATE CALIBRATION



Note 1: The ABD sequence requires the **EUSART** module to be configured in Asynchronous mode.

18.4 Comparator Interrupt Operation

The comparator interrupt flag can be set whenever there is a change in the output value of the comparator. Changes are recognized by means of a mismatch circuit which consists of two latches and an exclusive-or gate (see Figure 18-2 and Figure 18-3). One latch is updated with the comparator output level when the CMxCON0 register is read. This latch retains the value until the next read of the CMxCON0 register or the occurrence of a Reset. The other latch of the mismatch circuit is updated on every Q1 system clock. A mismatch condition will occur when a comparator output change is clocked through the second latch on the Q1 clock cycle. At this point the two mismatch latches have opposite output levels which is detected by the exclusive-or gate and fed to the interrupt circuitry. The mismatch condition persists until either the CMxCON0 register is read or the comparator output returns to the previous state.

Note 1: A write operation to the CMxCON0 register will also clear the mismatch condition because all writes include a read operation at the beginning of the write cycle.

2: Comparator interrupts will operate correctly regardless of the state of CxOE.

The comparator interrupt is set by the mismatch edge and not the mismatch level. This means that the interrupt flag can be reset without the additional step of reading or writing the CMxCON0 register to clear the mismatch registers. When the mismatch registers are cleared, an interrupt will occur upon the comparator's return to the previous state, otherwise no interrupt will be generated.

Software will need to maintain information about the status of the comparator output, as read from the CMxCON0 register, or CM2CON1 register, to determine the actual change that has occurred. See Figures 18-4 and 18-5.

The CxIF bit of the PIR2 register is the comparator interrupt flag. This bit must be reset by software by clearing it to '0'. Since it is also possible to write a '1' to this register, an interrupt can be generated.

In mid-range Compatibility mode the CxIE bit of the PIE2 register and the PEIE and GIE bits of the INTCON register must all be set to enable comparator interrupts. If any of these bits are cleared, the interrupt is not enabled, although the CxIF bit of the PIR2 register will still be set if an interrupt condition occurs.

18.4.1 PRESETTING THE MISMATCH LATCHES

The comparator mismatch latches can be preset to the desired state before the comparators are enabled. When the comparator is off the CxPOL bit controls the CxOUT level. Set the CxPOL bit to the desired CxOUT non-interrupt level while the CxON bit is cleared. Then, configure the desired CxPOL level in the same instruction that the CxON bit is set. Since all register writes are performed as a Read-Modify-Write, the mismatch latches will be cleared during the instruction Read phase and the actual configuration of the CxON and CxPOL bits will be occur in the final Write phase.

FIGURE 18-4: COMPARATOR INTERRUPT TIMING WITHOUT CMxCON0 READ

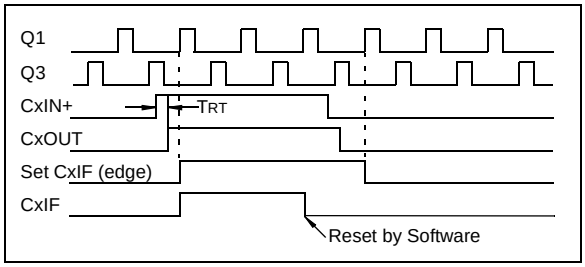
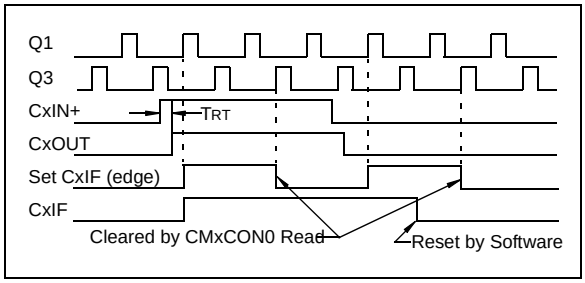


FIGURE 18-5: COMPARATOR INTERRUPT TIMING WITH CMxCON0 READ



Note 1: If a change in the CMxCON0 register (CxOUT) should occur when a read operation is being executed (start of the Q2 cycle), then the CxIF interrupt flag of the PIR2 register may not get set.

2: When either comparator is first enabled, bias circuitry in the Comparator module may cause an invalid output from the comparator until the bias circuitry is stable. Allow about 1 μs for bias settling then clear the mismatch condition and interrupt flags before enabling comparator interrupts.

TABLE 25-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
LITERAL OPERATIONS									
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	Move literal (12-bit) 2nd word	2	1110	1110	00ff	kkkk	None	
		to FSR(f) 1st word		1111	0000	kkkk	kkkk		
MOVLB	k	Move literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS									
TBLRD*		Table Read	2	0000	0000	0000	1000	None	
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None	
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None	
TBLRD+*		Table Read with pre-increment		0000	0000	0000	1011	None	
TBLWT*		Table Write	2	0000	0000	0000	1100	None	
TBLWT*+		Table Write with post-increment		0000	0000	0000	1101	None	
TBLWT*-		Table Write with post-decrement		0000	0000	0000	1110	None	
TBLWT+*		Table Write with pre-increment		0000	0000	0000	1111	None	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a `NOP` unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

BNC **Branch if Not Carry**

Syntax: BNC n

Operands: $-128 \le n \le 127$

Operation: if CARRY bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0011	nnnn	nnnn

Description: If the CARRY bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: HERE BNC Jump

Before Instruction
PC = address (HERE)
After Instruction
If CARRY = 0;
PC = address (Jump)
If CARRY = 1;
PC = address (HERE + 2)

BNN **Branch if Not Negative**

Syntax: BNN n

Operands: $-128 \le n \le 127$

Operation: if NEGATIVE bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0111	nnnn	nnnn

Description: If the NEGATIVE bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: HERE BNN Jump

Before Instruction
PC = address (HERE)
After Instruction
If NEGATIVE = 0;
PC = address (Jump)
If NEGATIVE = 1;
PC = address (HERE + 2)

RLNCF **Rotate Left f (No Carry)**

Syntax: RLNCF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

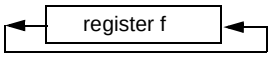
Operation: (f<n>) → dest<n + 1>,
(f<7>) → dest<0>

Status Affected: N, Z

Encoding:

0100	01da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the left. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RLNCF REG, 1, 0

Before Instruction
REG = 1010 1011
After Instruction
REG = 0101 0111

RRCF **Rotate Right f through Carry**

Syntax: RRCF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

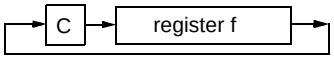
Operation: (f<n>) → dest<n – 1>,
(f<0>) → C,
(C) → dest<7>

Status Affected: C, N, Z

Encoding:

0011	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right through the CARRY flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RRCF REG, 0, 0

Before Instruction
REG = 1110 0110
C = 0
After Instruction
REG = 1110 0110
W = 0111 0011
C = 0

XORWF **Exclusive OR W with f**

Syntax: XORWF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) .XOR. (f) → dest

Status Affected: N, Z

Encoding:

0001	10da	ffff	ffff

Description: Exclusive OR the contents of W with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `XORWF REG, 1, 0`

Before Instruction
REG = AFh
W = B5h
After Instruction
REG = 1Ah
W = B5h

CALLW **Subroutine Call Using WREG**

Syntax: CALLW

Operands: None

Operation: (PC + 2) → TOS,
(W) → PCL,
(PCLATH) → PCH,
(PCLATU) → PCU

Status Affected: None

Encoding:

0000	0000	0001	0100

Description: First, the return address (PC + 2) is pushed onto the return stack. Next, the contents of W are written to PCL; the existing value is discarded. Then, the contents of PCLATH and PCLATU are latched into PCH and PCU, respectively. The second cycle is executed as a NOP instruction while the new next instruction is fetched. Unlike CALL, there is no option to update W, Status or BSR.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read WREG	PUSH PC to stack	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    CALLW
```
Before Instruction
```
PC      =   address (HERE)
PCLATH  =   10h
PCLATU  =   00h
W       =   06h
```
After Instruction
```
PC      =   001006h
TOS     =   address (HERE + 2)
PCLATH  =   10h
PCLATU  =   00h
W       =   06h
```

MOVSF **Move Indexed to f**

Syntax: MOVSF [z_s], f_d

Operands: $0 \le z_s \le 127$
$0 \le f_d \le 4095$

Operation: ((FSR2) + z_s) → f_d

Status Affected: None

Encoding:

1st word (source)	1110	1011	0zzz	$zzzz_s$
2nd word (destin.)	1111	ffff	ffff	$ffff_d$

Description: The contents of the source register are moved to destination register 'f_d'. The actual address of the source register is determined by adding the 7-bit literal offset 'z_s' in the first word to the value of FSR2. The address of the destination register is specified by the 12-bit literal 'f_d' in the second word. Both addresses can be anywhere in the 4096-byte data space (000h to FFFh).
The MOVSF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.
If the resultant source address points to an indirect addressing register, the value returned will be 00h.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Determine source addr	Determine source addr	Read source reg
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example:
```
MOVSF   [05h], REG2
```
Before Instruction
```
FSR2            =   80h
Contents
of 85h          =   33h
REG2            =   11h
```
After Instruction
```
FSR2            =   80h
Contents
of 85h          =   33h
REG2            =   33h
```

Unless otherwise noted, V_{IN} = 5V, F_{OSC} = 300 kHz, C_{IN} = 0.1 µF, T_A = 25°C.

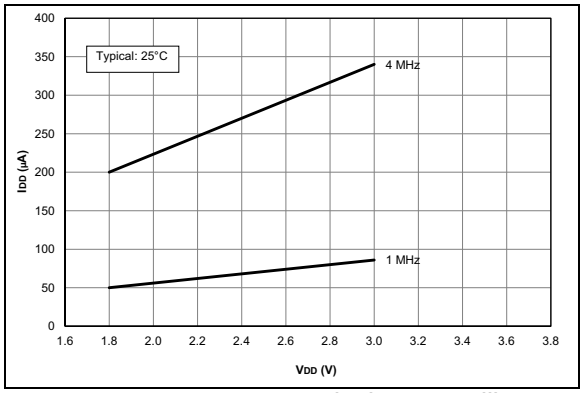


FIGURE 28-7: *IDD Typical, EC Oscillator, Medium-Power Mode, PIC18LF1XK50 Only.*

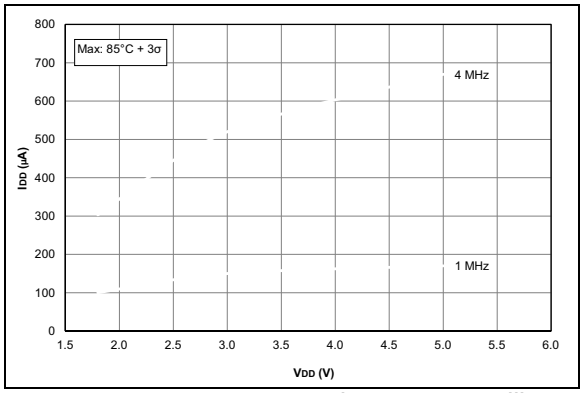


FIGURE 28-10: *IDD Maximum, EC Oscillator, Medium-Power Mode, PIC18F1XK50 Only.*

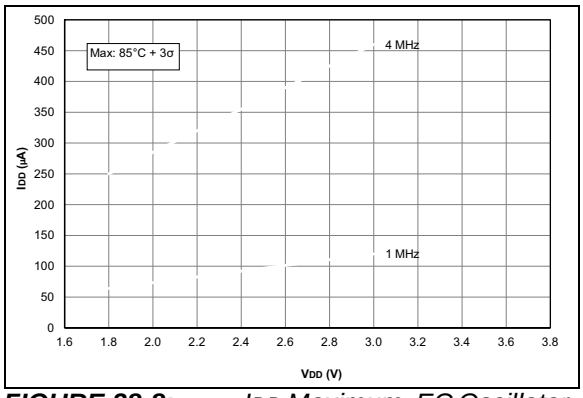


FIGURE 28-8: *IDD Maximum, EC Oscillator, Medium-Power Mode, PIC18LF1XK50 Only.*

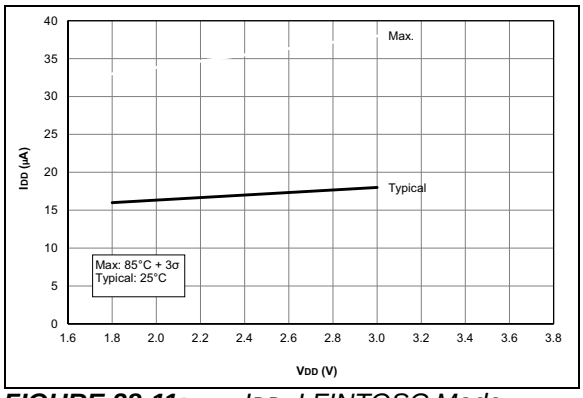


FIGURE 28-11: *IDD, LFINTOSC Mode, FOSC = 31 kHz, PIC18LF1XK50 Only.*

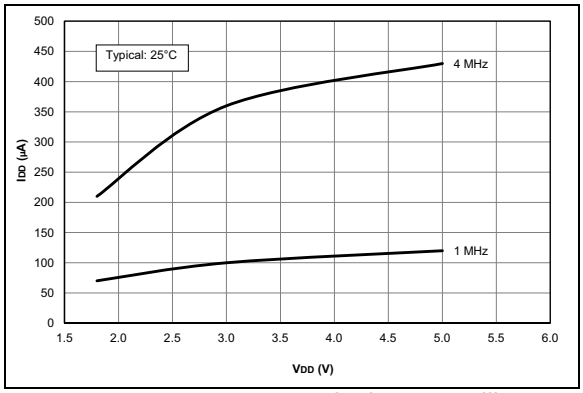


FIGURE 28-9: *IDD Typical, EC Oscillator, Medium-Power Mode, PIC18F1XK50 Only.*

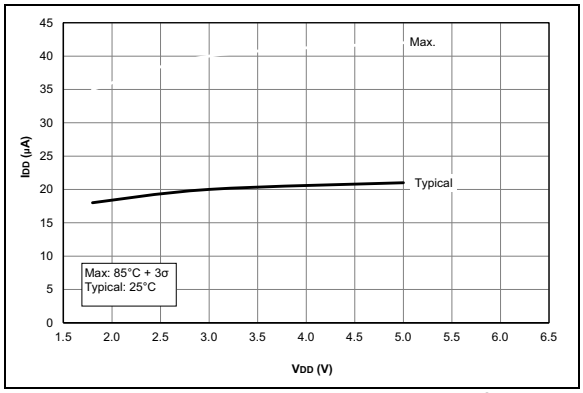


FIGURE 28-12: *IDD, LFINTOSC Mode, FOSC = 31 kHz, PIC18F1XK50 Only.*