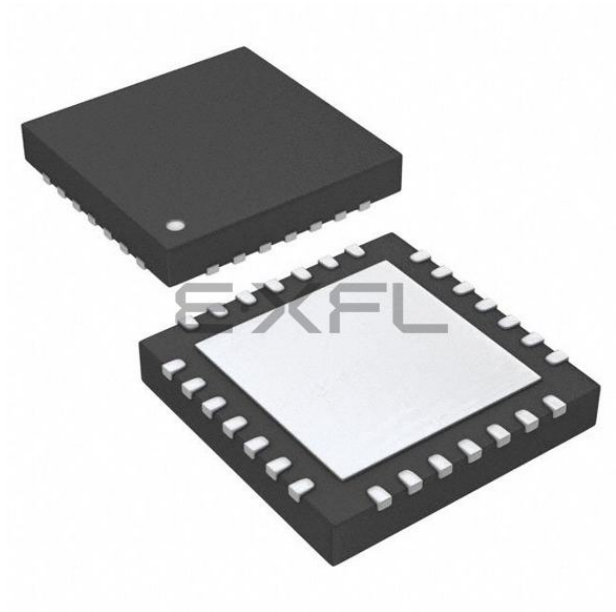


Welcome to E-XFL.COM

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	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 24x10b; 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/pic18f24k40-i-ml

3.0 DEVICE CONFIGURATION

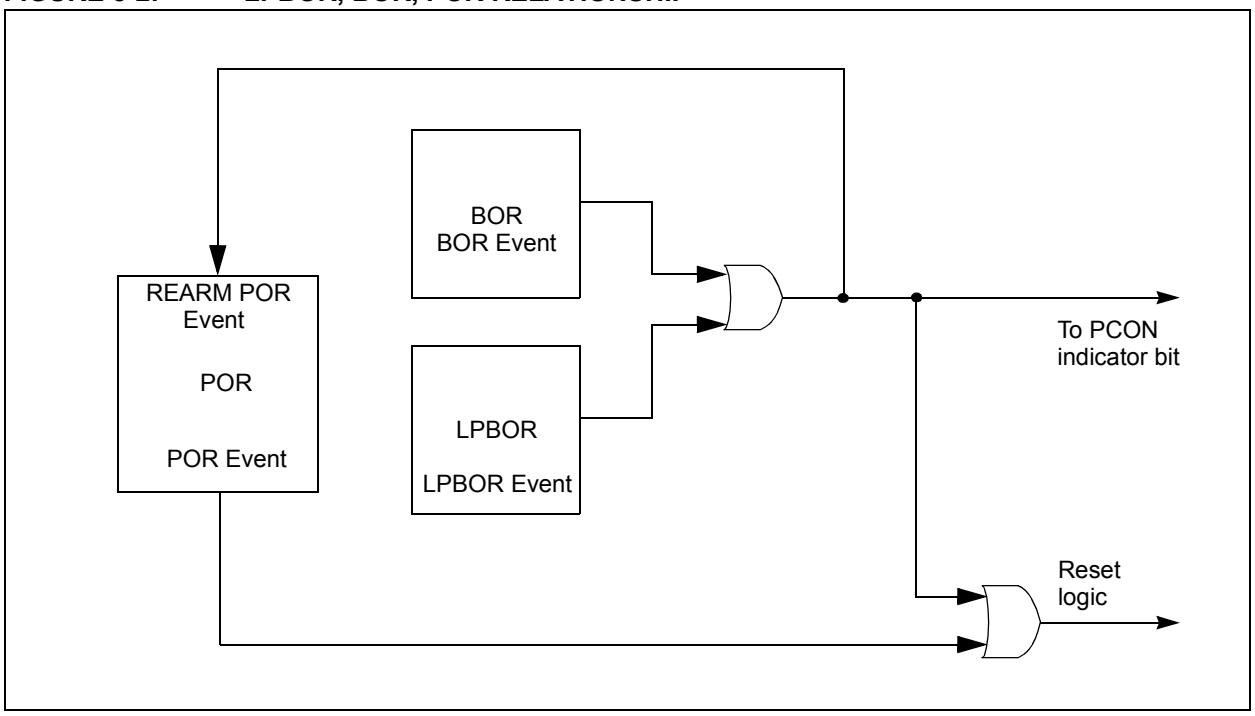
Device configuration consists of Configuration Words, Code Protection, Device ID and Rev ID.

3.1 Configuration Words

There are six Configuration Word bits that allow the user to setup the device with several choices of oscillators, Resets and memory protection options. These are implemented as Configuration Word 1 through Configuration Word 6 at 300000h through 30000Bh.

Note:	The $\overline{\text{DEBUG}}$ bit in Configuration Words is managed automatically by device development tools including debuggers and programmers. For normal device operation, this bit should be maintained as a '1'.

FIGURE 8-2: LPBOR, BOR, POR RELATIONSHIP



8.1 Register Definitions: BOR Control

REGISTER 8-1: BORCON: BROWN-OUT RESET CONTROL REGISTER

R/W-1/u	U-0	U-0	U-0	U-0	U-0	U-0	R-q/u
SBOREN	—	—	—	—	—	—	BORRDY
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	q = Value depends on condition

bit 7 **SBOREN:** Software Brown-out Reset Enable bit
If BOREN ≠ 01:
SBOREN is read/write, but has no effect on the BOR.
If BOREN = 01:
1 = BOR Enabled
0 = BOR Disabled

bit 6-1 **Unimplemented:** Read as '0'

bit 0 **BORRDY:** Brown-out Reset Circuit Ready Status bit
1 = The Brown-out Reset Circuit is active and armed
0 = The Brown-out Reset Circuit is disabled or is warming up

10.2.3 LOOK-UP TABLES IN PROGRAM MEMORY

There may be programming situations that require the creation of data structures, or look-up tables, in program memory. For PIC18 devices, look-up tables can be implemented in two ways:

- Computed GOTO
- Table Reads

10.2.3.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the program counter. An example is shown in Example 10-2.

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW nn instructions. The W register is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW nn instructions that returns the value 'nn' to the calling function.

The offset value (in WREG) specifies the number of bytes that the program counter should advance and should be multiples of two (LSb = 0).

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

EXAMPLE 10-2: COMPUTED GOTO USING AN OFFSET VALUE

```
        MOVF    OFFSET, W
        CALL    TABLE
ORG     nn00h
TABLE   ADDWF   PCL
        RETLW   nnh
        RETLW   nnh
        RETLW   nnh
        .
        .
        .
```

10.2.3.2 Table Reads and Table Writes

A better method of storing data in program memory allows two bytes of data to be stored in each instruction location.

Look-up table data may be stored two bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) register specifies the byte address and the Table Latch (TABLAT) register contains the data that is read from or written to program memory. Data is transferred to or from program memory one byte at a time.

Table read and table write operations are discussed further in **Section 11.1.1 "Table Reads and Table Writes"**.

19.7 Timer1/3/5 16-Bit Read/Write Mode

Timer1/3/5 can be configured to read and write all 16 bits of data, to and from, the 8-bit TMRxL and TMRxH registers, simultaneously. The 16-bit read and write operations are enabled by setting the RD16 bit of the TxCON register.

To accomplish this function, the TMRxH register value is mapped to a buffer register called the TMRxH buffer register. While in 16-Bit mode, the TMRxH register is not directly readable or writable and all read and write operations take place through the use of this TMRxH buffer register.

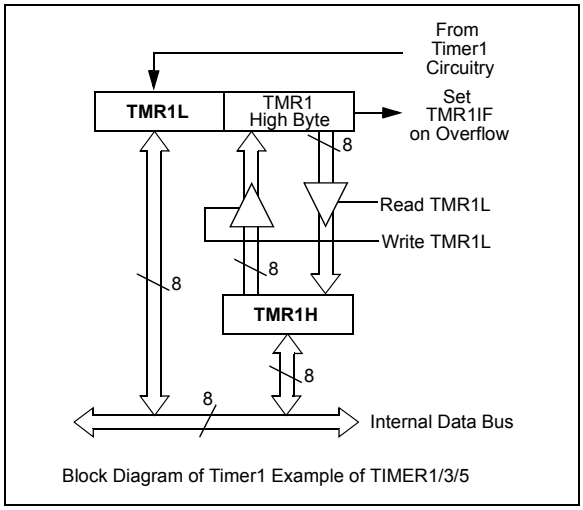
When a read from the TMRxL register is requested, the value of the TMRxH register is simultaneously loaded into the TMRxH buffer register. When a read from the TMRxH register is requested, the value is provided from the TMRxH buffer register instead. This provides the user with the ability to accurately read all 16 bits of the Timer1/3/5 value from a single instance in time. Reference the block diagram in Figure 19-2 for more details.

In contrast, when not in 16-Bit mode, the user must read each register separately and determine if the values have become invalid due to a rollover that may have occurred between the read operations.

When a write request of the TMRxL register is requested, the TMRxH buffer register is simultaneously updated with the contents of the TMRxH register. The value of TMRxH must be preloaded into the TMRxH buffer register prior to the write request for the TMRxL register. This provides the user with the ability to write all 16 bits to the TMRxL:TMRxH register pair at the same time.

Any requests to write to the TMRxH directly does not clear the Timer1/3/5 prescaler value. The prescaler value is only cleared through write requests to the TMRxL register.

FIGURE 19-2: TIMER1/3/5 16-BIT READ/WRITE MODE BLOCK DIAGRAM



Block Diagram of Timer1 Example of TIMER1/3/5

19.8 Timer1/3/5 Gate

Timer1/3/5 can be configured to count freely or the count can be enabled and disabled using Timer1/3/5 gate circuitry. This is also referred to as Timer1/3/5 gate enable.

Timer1/3/5 gate can also be driven by multiple selectable sources.

19.8.1 TIMER1/3/5 GATE ENABLE

The Timer1/3/5 Gate Enable mode is enabled by setting the TMRxGE bit of the TxGCON register. The polarity of the Timer1/3/5 Gate Enable mode is configured using the TxGPOL bit of the TxGCON register.

When Timer1/3/5 Gate Enable mode is enabled, Timer1/3/5 will increment on the rising edge of the Timer1/3/5 clock source. When Timer1/3/5 Gate signal is inactive, the timer will not increment and hold the current count. Enable mode is disabled, no incrementing will occur and Timer1/3/5 will hold the current count. See Figure 19-4 for timing details.

TABLE 19-3: TIMER1/3/5 GATE ENABLE SELECTIONS

TMRxCLK	TxGPOL	TxG	Timer1/3/5 Operation
↑	1	1	Counts
↑	1	0	Holds Count
↑	0	1	Holds Count
↑	0	0	Counts

TABLE 20-1: TIMER2 OPERATING MODES

Mode	MODE<4:0>		Output Operation	Operation	Timer Control		
	<4:3>	<2:0>			Start	Reset	Stop
Free Running Period	00	000	Period Pulse	Software gate (Figure 20-4)	ON = 1	—	ON = 0
		001		Hardware gate, active-high (Figure 20-5)	ON = 1 and TMRx_ers = 1	—	ON = 0 or TMRx_ers = 0
		010		Hardware gate, active-low	ON = 1 and TMRx_ers = 0	—	ON = 0 or TMRx_ers = 1
		011	Period Pulse with Hardware Reset	Rising or falling edge Reset	ON = 1	TMRx_ers ↕	ON = 0
		100		Rising edge Reset (Figure 20-6)		TMRx_ers ↑	
		101		Falling edge Reset		TMRx_ers ↓	
		110		Low level Reset		TMRx_ers = 0	ON = 0 or TMRx_ers = 0
		111		High level Reset (Figure 20-7)		TMRx_ers = 1	ON = 0 or TMRx_ers = 1
One-shot	01	000	One-shot	Software start (Figure 20-8)	ON = 1	—	ON = 0 or Next clock after TMRx = PRx (**Note 2**)
		001	Edge triggered start (**Note 1**)	Rising edge start (Figure 20-9)	ON = 1 and TMRx_ers ↑	—	
		010		Falling edge start	ON = 1 and TMRx_ers ↓	—	
		011		Any edge start	ON = 1 and TMRx_ers ↕	—	
		100	Edge triggered start and hardware Reset (**Note 1**)	Rising edge start and Rising edge Reset (Figure 20-10)	ON = 1 and TMRx_ers ↑	TMRx_ers ↑	
		101		Falling edge start and Falling edge Reset	ON = 1 and TMRx_ers ↓	TMRx_ers ↓	
		110		Rising edge start and Low level Reset (Figure 20-11)	ON = 1 and TMRx_ers ↑	TMRx_ers = 0	
		111		Falling edge start and High level Reset	ON = 1 and TMRx_ers ↓	TMRx_ers = 1	
Mono-stable	10	000	Reserved				
		001	Edge triggered start (**Note 1**)	Rising edge start (Figure 20-12)	ON = 1 and TMRx_ers ↑	—	ON = 0 or Next clock after TMRx = PRx (**Note 3**)
		010		Falling edge start	ON = 1 and TMRx_ers ↓	—	
		011		Any edge start	ON = 1 and TMRx_ers ↕	—	
Reserved		100	Reserved				
Reserved		101	Reserved				
One-shot		110	Level triggered start and hardware Reset	High level start and Low level Reset (Figure 20-13)	ON = 1 and TMRx_ers = 1	TMRx_ers = 0	ON = 0 or Held in Reset (**Note 2**)
		111		Low level start & High level Reset	ON = 1 and TMRx_ers = 0	TMRx_ers = 1	
Reserved	11	xxx	Reserved				

Note 1: If ON = 0 then an edge is required to restart the timer after ON = 1.

2: When TMRx = PRx then the next clock clears ON and stops TMRx at 00h.

3: When TMRx = PRx then the next clock stops TMRx at 00h but does not clear ON.

20.5 Operation Examples

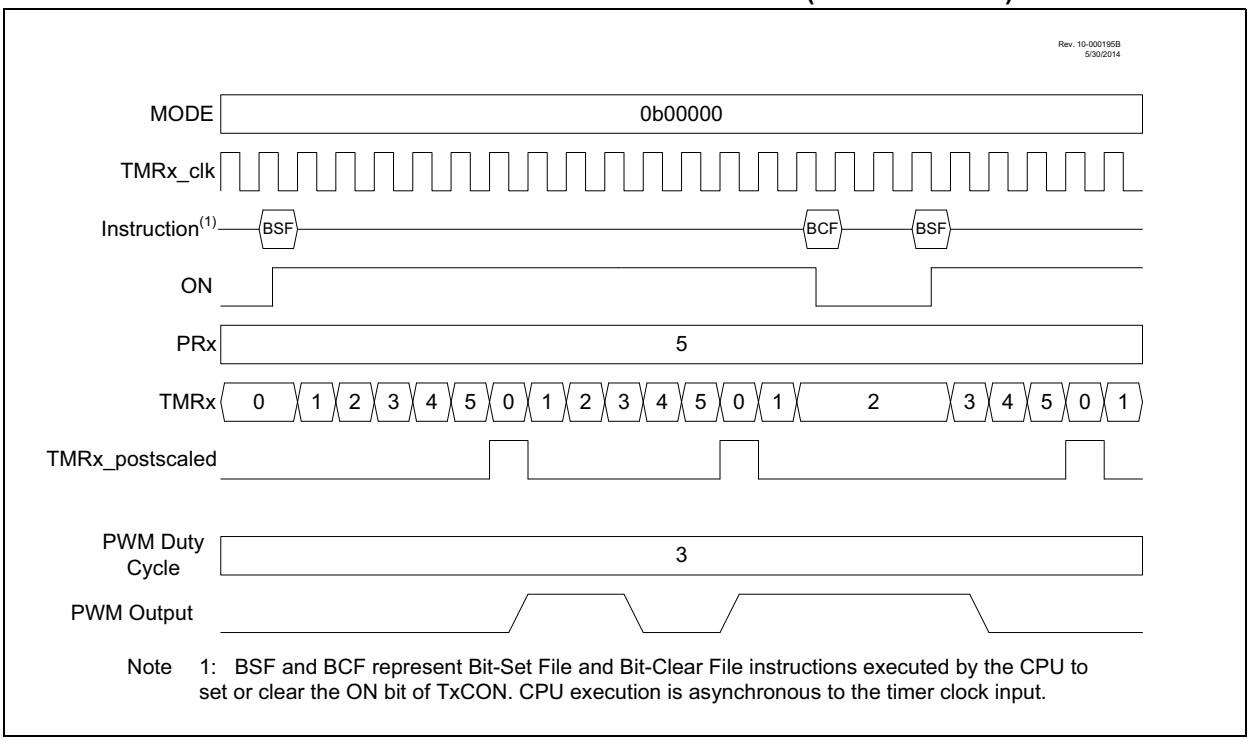
Unless otherwise specified, the following notes apply to the following timing diagrams:

- Both the prescaler and postscaler are set to 1:1 (both the CKPS and OUTPS bits in the TxCON register are cleared).
- The diagrams illustrate any clock except Fosc/4 and show clock-sync delays of at least two full cycles for both ON and Timer2_ers. When using Fosc/4, the clock-sync delay is at least one instruction period for Timer2_ers; ON applies in the next instruction period.
- The PWM Duty Cycle and PWM output are illustrated assuming that the timer is used for the PWM function of the CCP module as described in **Section 21.0 "Capture/Compare/PWM Module"**. The signals are not a part of the Timer2 module.

20.5.1 SOFTWARE GATE MODE

This mode corresponds to legacy Timer2 operation. The timer increments with each clock input when ON = 1 and does not increment when ON = 0. When the TMRx count equals the PRx period count the timer resets on the next clock and continues counting from 0. Operation with the ON bit software controlled is illustrated in Figure 20-4. With PRx = 5, the counter advances until TMRx = 5, and goes to zero with the next clock.

FIGURE 20-4: SOFTWARE GATE MODE TIMING DIAGRAM (MODE = 00000)



Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

REGISTER 20-4: TxRST: TIMER2 EXTERNAL RESET SIGNAL SELECTION REGISTER

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	RSEL<3:0>			
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 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **RSEL<3:0>:** Timer2 External Reset Signal Source Selection bits

RSEL<3:0>	TMR2	TMR4	TMR6
	Reset Source	Reset Source	Reset Source
1011-1111	Reserved	Reserved	Reserved
1010	ZCD_OUT	ZCD_OUT	ZCD_OUT
1001	CMP2OUT	CMP2OUT	CMP2OUT
1000	CMP1OUT	CMP1OUT	CMP1OUT
0111	PWM4OUT	PWM4OUT	PWM4OUT
0110	PWM3OUT	PWM3OUT	PWM3OUT
0101	CCP2OUT	CCP2OUT	CCP2OUT
0100	CCP1OUT	CCP1OUT	CCP1OUT
0011	TMR6 post-scaled	TMR6 post-scaled	Reserved
0010	TMR4 post-scaled	Reserved	TMR4 post-scaled
0001	Reserved	TMR2 post-scaled	TMR2 post-scaled
0000	Pin selected by T2INPPS	Pin selected by T4INPPS	Pin selected by T6INPPS

REGISTER 21-3: CCPxCAP: CAPTURE INPUT SELECTION MULTIPLEXER REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/x	R/W-0/x
—	—	—	—	—	—	CTS<1:0>	
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 7-2 **Unimplemented:** Read as '0'

bit 1-0 **CTS<1:0>:** Capture Trigger Input Selection bits

CTS<1:0>	**Connection**	
	CCP1	**CCP2**
11	IOC_Interrupt	
10	CMP2_output	
01	CMP1_output	
00	Pin selected by CCP1PPS	Pin selected by CCP2PPS

REGISTER 21-4: CCPRxL: CCPx REGISTER LOW BYTE

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
CCPRx<7:0>							
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 7-0 MODE = Capture Mode:
CCPRxL<7:0>: LSB of captured TMR1 value
MODE = Compare Mode:
CCPRxL<7:0>: LSB compared to TMR1 value
MODE = PWM Mode && FMT = 0:
CCPRxL<7:0>: CCPW<7:0> – Pulse-Width LS 8 bits
MODE = PWM Mode && FMT = 1:
CCPRxL<7:6>: CCPW<1:0> – Pulse-Width LS 2 bits
CCPRxL<5:0>: Not used

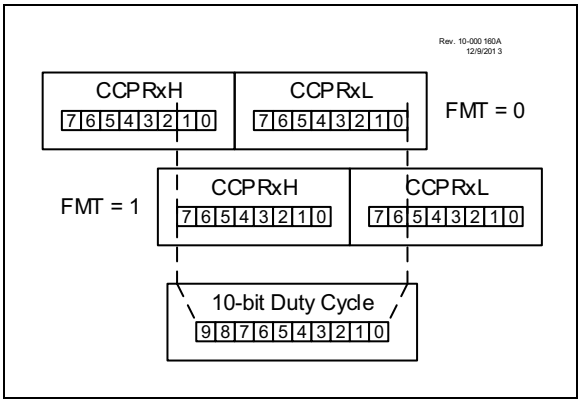
21.5.5 PWM DUTY CYCLE

The PWM duty cycle is specified by writing a 10-bit value to the CCPRxH:CCPRxL register pair. The alignment of the 10-bit value is determined by the FMT bit of the CCPxCON register (see Figure 21-5). The CCPRxH:CCPRxL register pair can be written to at any time; however the duty cycle value is not latched into the 10-bit buffer until after a match between PR2 and TMR2.

Equation 21-2 is used to calculate the PWM pulse width.

Equation 21-3 is used to calculate the PWM duty cycle ratio.

FIGURE 21-5: PWM 10-BIT ALIGNMENT



EQUATION 21-2: PULSE WIDTH

$$Pulse\ Width = (CCPRxH{:}CCPRxL\ register\ pair) \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

EQUATION 21-3: DUTY CYCLE RATIO

$$Duty\ Cycle\ Ratio = \frac{(CCPRxH{:}CCPRxL\ register\ pair)}{4(PR2 + 1)}$$

CCPRxH:CCPRxL register pair are used to double buffer the PWM duty cycle. This double buffering is essential for glitchless PWM operation.

The 8-bit timer TMR2 register is concatenated with either the 2-bit internal system clock (FOSC), or two bits of the prescaler, to create the 10-bit time base. The system clock is used if the Timer2 prescaler is set to 1:1.

When the 10-bit time base matches the CCPRxH:CCPRxL register pair, then the CCPx pin is cleared (see Figure 21-4).

21.5.6 PWM RESOLUTION

The resolution determines the number of available duty cycles for a given period. For example, a 10-bit resolution will result in 1024 discrete duty cycles, whereas an 8-bit resolution will result in 256 discrete duty cycles.

The maximum PWM resolution is ten bits when PR2 is 255. The resolution is a function of the PR2 register value as shown by Equation 21-4.

EQUATION 21-4: PWM RESOLUTION

$$Resolution = \frac{log[4(PR2 + 1)]}{log(2)}\ bits$$

Note: If the pulse-width value is greater than the period the assigned PWM pin(s) will remain unchanged.

REGISTER 26-4: SSPxBUF: MSSP DATA BUFFER REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
BUF<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 **BUF<7:0>:** MSSP Buffer bits

REGISTER 26-5: SSPxADD: MSSP ADDRESS 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
ADD<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	

Master mode: SPI mode

bit 7-0 Baud Rate Clock Divider bits
SCK/SCL pin clock period = ((SSPxADD<7:0> + 1) *4)/FOSC

REGISTER 27-2: RCxSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R-0/0	R/HC-0/0	R/HC-0/0
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	HC = Bit is cleared by hardware
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 **SPEN:** Serial Port Enable bit
1 = Serial port enabled
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-Bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave
Don't care

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

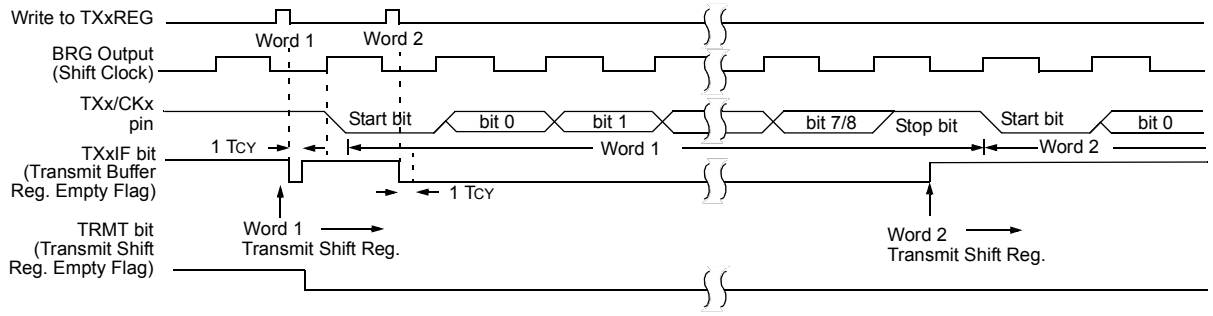
bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enable interrupt and load the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 8-bit (RX9 = 0):
Don't care

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be updated by reading RCxREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** Ninth bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.

FIGURE 27-4: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)



Note: This timing diagram shows two consecutive transmissions.

TABLE 27-1: SUMMARY OF REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
BAUDxCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	389
INTCON	GIE/GIEH	PEIE/GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG	166
PIE3	—	—	RC1IE	TX1IE	—	—	BCL1IE	SSP1IE	178
PIR3	—	—	RC1IF	TX1IF	—	—	BCL1IF	SSP1IF	170
IPR3	—	—	RC1IP	TX1IP	—	—	BCL1IP	SSP1IP	186
RCxSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	388
RxyPPS	—	—	—	RxyPPS<4:0>					213
TXxPPS	—	—	—	TXPPS<4:0>					211
SPxBRGH	EUSARTx Baud Rate Generator, High Byte								398*
SPxBRGL	EUSARTx Baud Rate Generator, Low Byte								398*
TXxREG	EUSARTx Transmit Register								390*
TXxSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	387

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for asynchronous transmission.
* Page provides register information.

31.4.2 PRECHARGE CONTROL

The Precharge stage is an optional period of time that brings the external channel and internal sample and hold capacitor to known voltage levels. Precharge is enabled by writing a non-zero value to the ADPRE register. This stage is initiated when an ADC conversion begins, either from setting the ADGO bit, a special event trigger, or a conversion restart from the computation functionality. If the ADPRE register is cleared when an ADC conversion begins, this stage is skipped.

During the precharge time, CHOLD is disconnected from the outer portion of the sample path that leads to the external capacitive sensor and is connected to either VDD or VSS, depending on the value of the ADPPOL bit of ADCON1. At the same time, the port pin logic of the selected analog channel is overridden to drive a digital high or low out, in order to precharge the outer portion of the ADC's sample path, which includes the external sensor. The output polarity of this override is also determined by the ADPPOL bit of ADCON1. The amount of time that this charging needs is controlled by the ADPRE register.

Note:	The external charging overrides the TRIS setting of the respective I/O pin. If there is a device attached to this pin, Precharge should not be used.

31.4.3 ACQUISITION CONTROL

The Acquisition stage is an optional time for the voltage on the internal sample and hold capacitor to charge or discharge from the selected analog channel.This acquisition time is controlled by the ADACQ register. If ADPRE = 0, acquisition starts at the beginning of conversion. When ADPRE = 1, the acquisition stage begins when precharge ends.

At the start of the acquisition stage, the port pin logic of the selected analog channel is overridden to turn off the digital high/low output drivers so they do not affect the final result of the charge averaging. Also, the selected ADC channel is connected to CHOLD. This allows charge averaging to proceed between the precharged channel and the CHOLD capacitor.

Note:	When ADPRE! = 0, acquisition time cannot be '0'. In this case, setting ADACQ to '0' will set a maximum acquisition time (256 ADC clock cycles). When precharge is disabled, setting ADACQ to '0' will disable hardware acquisition time control.

31.4.4 GUARD RING OUTPUTS

Figure 31-8 shows a typical guard ring circuit. CGUARD represents the capacitance of the guard ring trace placed on the PCB board. The user selects values for RA and RB that will create a voltage profile on CGUARD, which will match the selected acquisition channel.

The purpose of the guard ring is to generate a signal in phase with the CVD sensing signal to minimize the effects of the parasitic capacitance on sensing electrodes. It also can be used as a mutual drive for mutual capacitive sensing. For more information about active guard and mutual drive, see Application Note AN1478, "*mTouch™ Sensing Solution Acquisition Methods Capacitive Voltage Divider*" (DS01478).

The ADC has two guard ring drive outputs, ADGRDA and ADGRDB. These outputs can be routed through PPS controls to I/O pins (see **Section 17.0 "Peripheral Pin Select (PPS) Module"** for details) and the polarity of these outputs are controlled by the ADGPOL and ADIPEN bits of ADCON1.

At the start of the first precharge stage, both outputs are set to match the ADGPOL bit of ADCON1. Once the acquisition stage begins, ADGRDA changes polarity, while ADGRDB remains unchanged. When performing a double sample conversion, setting the ADIPEN bit of ADCON1 causes both guard ring outputs to transition to the opposite polarity of ADGPOL at the start of the second precharge stage, and ADGRDA toggles again for the second acquisition. For more information on the timing of the guard ring output, refer to Figure 31-8 and Figure 31-9.

FIGURE 31-8: GUARD RING CIRCUIT

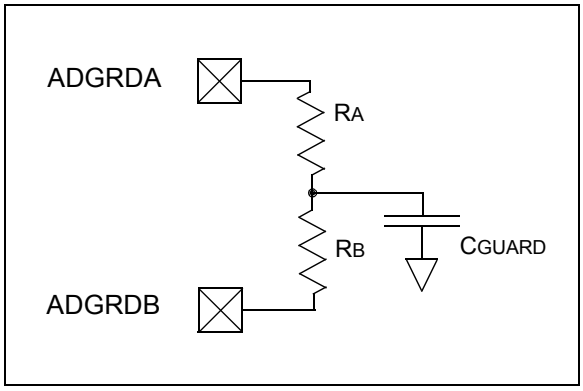
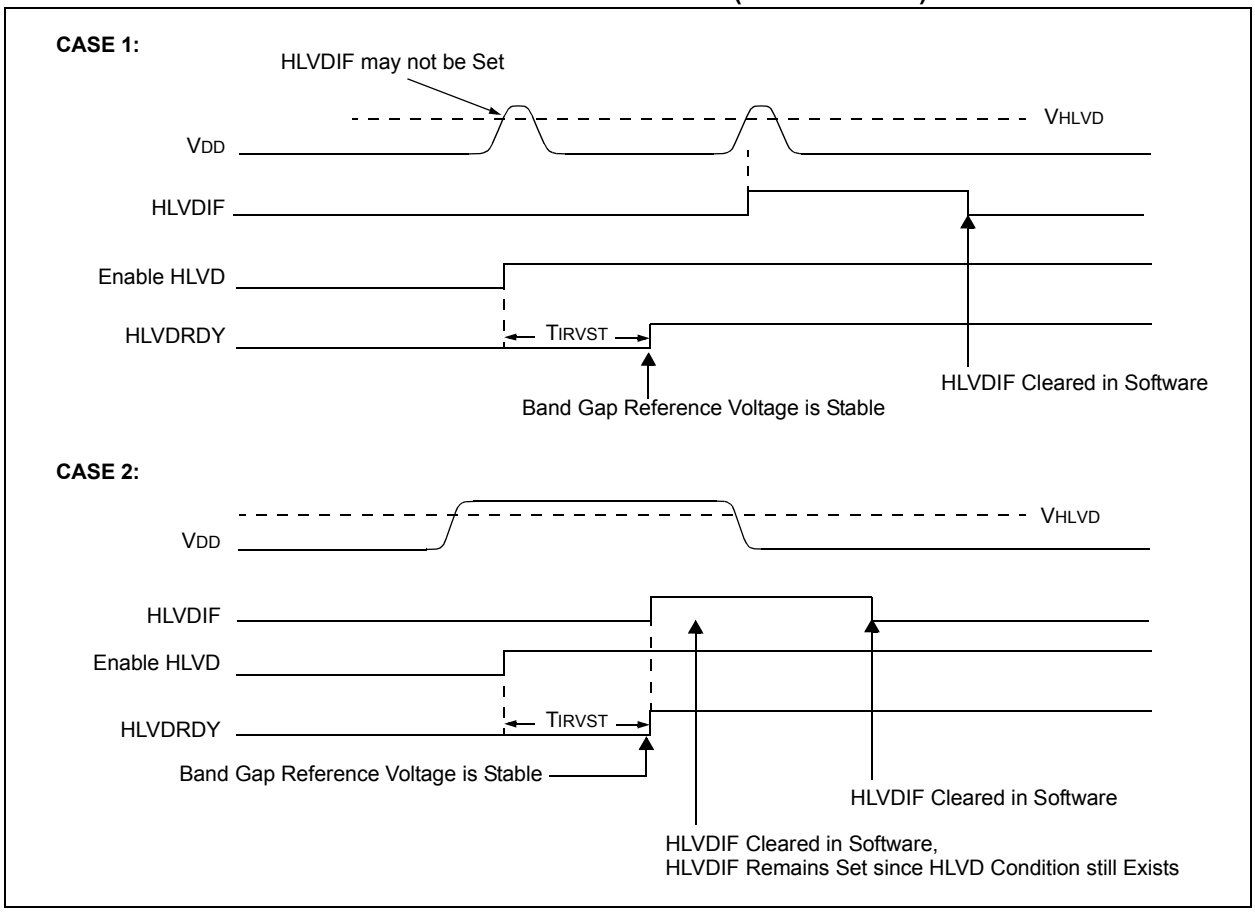


FIGURE 33-3: HIGH-VOLTAGE DETECT OPERATION (HLVDINTH = 1)



33.5 Applications

In many applications, it is desirable to detect a drop below, or rise above, a particular voltage threshold. For example, the HLVD module could be periodically enabled to detect Universal Serial Bus (USB) attach or detach. This assumes the device is powered by a lower voltage source than the USB when detached. An attach would indicate a High-Voltage Detect from, for example, 3.3V to 5V (the voltage on USB) and vice versa for a detach. This feature could save a design a few extra components and an attach signal (input pin).

For general battery applications, Figure 33-4 shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage, VA, the HLVD logic generates an interrupt at time, TA. The interrupt could cause the execution of an Interrupt Service Routine (ISR), which would allow the application to perform “housekeeping tasks” and a controlled shutdown before the device voltage exits the valid operating range at TB. This would give the application a time window, represented by the difference between TA and TB, to safely exit.

FIGURE 33-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION

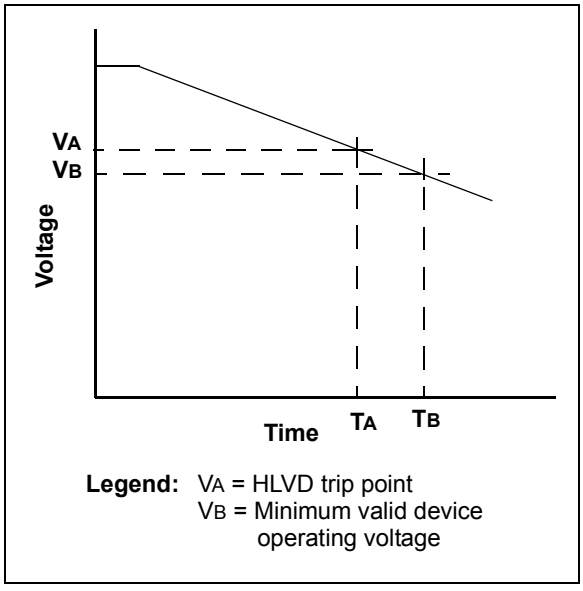


TABLE 35-2: INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BIT-ORIENTED OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, b, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2
CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BNZ	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	k, s	Call subroutine 1st word	2	1110	110s	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C	
GOTO	k	Go to address 1st word	2	1110	1111	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	4
POP	—	Pop top of return stack (TOS)	1	0000	0000	0000	0110	None	
PUSH	—	Push top of return stack (TOS)	1	0000	0000	0000	0101	None	
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None	
RESET		Software device Reset	1	0000	0000	1111	1111	All	
RETFIE	s	Return from interrupt enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	
SLEEP	—	Go into Standby mode	1	0000	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	

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.

MULLW **Multiply literal with W**

Syntax: MULLW k

Operands: $0 \le k \le 255$

Operation: (W) x k → PRODH:PRODL

Status Affected: None

Encoding:

0000	1101	kkkk	kkkk

Description: An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged.
None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL

Example: `MULLW 0C4h`

Before Instruction
W = E2h
PRODH = ?
PRODL = ?
After Instruction
W = E2h
PRODH = ADh
PRODL = 08h

MULWF **Multiply W with f**

Syntax: MULWF f {,a}

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

Operation: (W) x (f) → PRODH:PRODL

Status Affected: None

Encoding:

0000	001a	ffff	ffff

Description: An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged.
None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
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 35.2.3 "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 registers PRODH: PRODL

Example: `MULWF REG, 1`

Before Instruction
W = C4h
REG = B5h
PRODH = ?
PRODL = ?
After Instruction
W = C4h
REG = B5h
PRODH = 8Ah
PRODL = 94h

ADDWF **ADD W to Indexed (Indexed Literal Offset mode)**

Syntax: ADDWF [k] {,d}

Operands: $0 \le k \le 95$
$d \in [0,1]$

Operation: (W) + ((FSR2) + k) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

0010	01d0	kkkk	kkkk

Description: The contents of W are added to the contents of the register indicated by FSR2, offset by the value 'k'.
If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: ADDWF [OFST], 0

Before Instruction
W = 17h
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 20h
After Instruction
W = 37h
Contents of 0A2Ch = 20h

BSF **Bit Set Indexed (Indexed Literal Offset mode)**

Syntax: BSF [k], b

Operands: $0 \le f \le 95$
$0 \le b \le 7$

Operation: 1 → ((FSR2) + k)<b>

Status Affected: None

Encoding:

1000	bbb0	kkkk	kkkk

Description: Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: BSF [FLAG_OFST], 7

Before Instruction
FLAG_OFST = 0Ah
FSR2 = 0A00h
Contents of 0A0Ah = 55h
After Instruction
Contents of 0A0Ah = D5h

SETF **Set Indexed (Indexed Literal Offset mode)**

Syntax: SETF [k]

Operands: $0 \le k \le 95$

Operation: FFh → ((FSR2) + k)

Status Affected: None

Encoding:

0110	1000	kkkk	kkkk

Description: The contents of the register indicated by FSR2, offset by 'k', are set to FFh.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read 'k'	Process Data	Write register

Example: SETF [OFST]

Before Instruction
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 00h
After Instruction
Contents of 0A2Ch = FFh

36.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

36.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

36.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

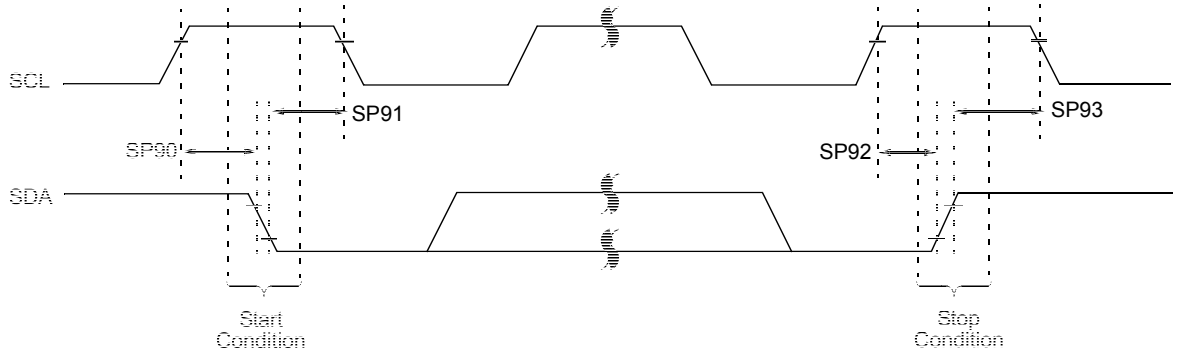
- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

36.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

FIGURE 37-20: I²C BUS START/STOP BITS TIMING



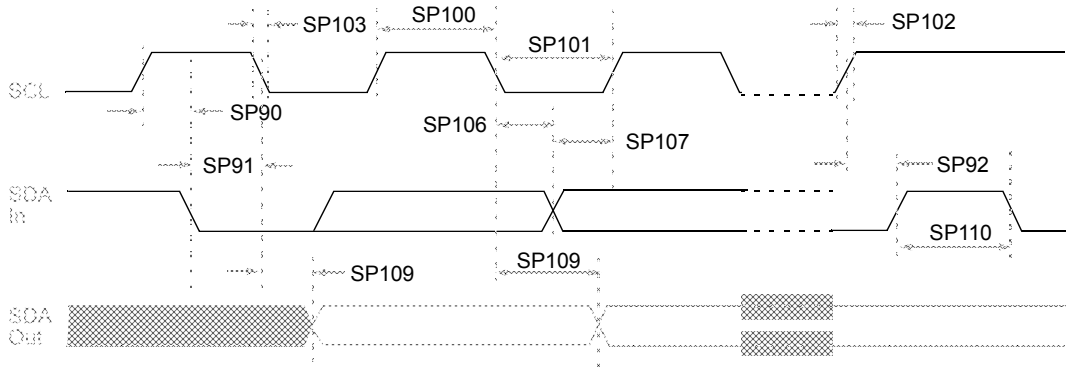
Note: Refer to Figure 37-4 for load conditions.

TABLE 37-24: I²C BUS START/STOP BITS REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)								
Param No.	**Symbol**	**Characteristic**		**Min.**	**Typ**	**Max.**	**Units**	**Conditions**
SP90*	TSU:STA	Start condition	100 kHz mode	4700	—	—	ns	Only relevant for Repeated Start condition
		Setup time	400 kHz mode	600	—	—		
SP91*	THD:STA	Start condition	100 kHz mode	4000	—	—	ns	After this period, the first clock pulse is generated
		Hold time	400 kHz mode	600	—	—		
SP92*	TSU:STO	Stop condition	100 kHz mode	4700	—	—	ns	
		Setup time	400 kHz mode	600	—	—		
SP93	THD:STO	Stop condition	100 kHz mode	4000	—	—	ns	
		Hold time	400 kHz mode	600	—	—		

* These parameters are characterized but not tested.

FIGURE 37-21: I²C BUS DATA TIMING

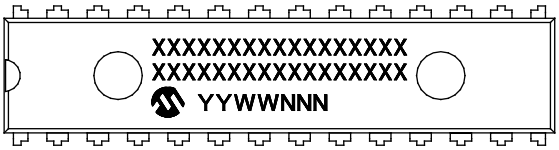


Note: Refer to Figure 37-4 for load conditions.

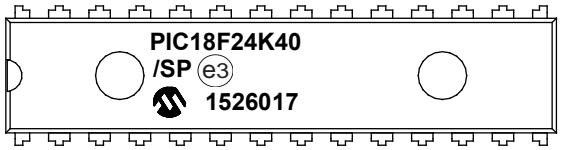
39.0 PACKAGING INFORMATION

Package Marking Information

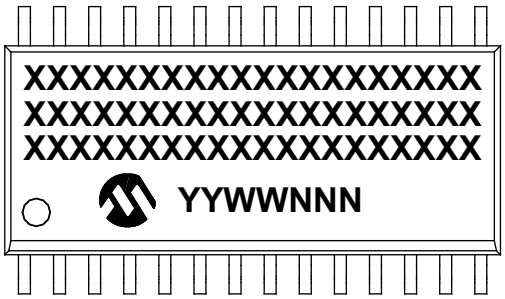
28-Lead SPDIP (.300")



Example

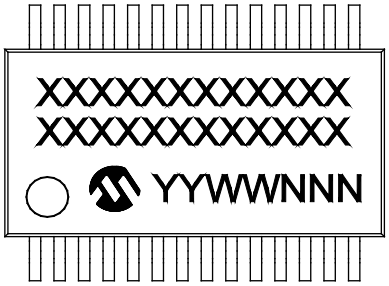


28-Lead SOIC (7.50 mm)

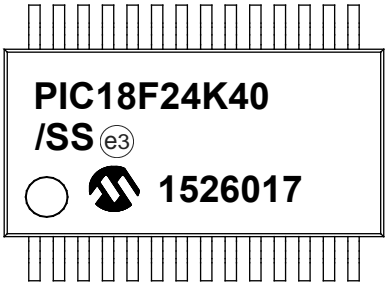


Example

28-Lead SSOP (5.30 mm)



Example



Legend:	XX...X	Customer-specific information or Microchip part number
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week '01')
	NNN	Alphanumeric traceability code
	(e3)	Pb-free JEDEC® designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package.

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.