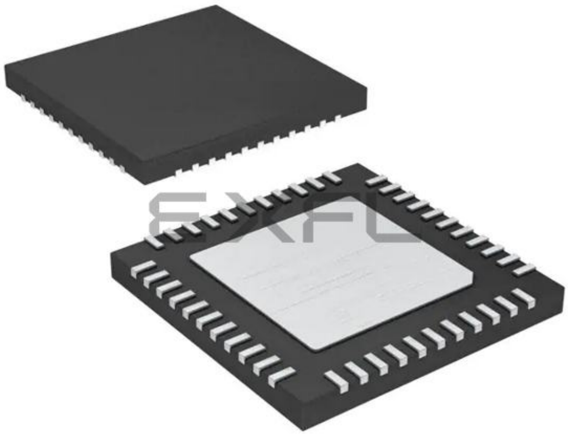


Welcome to E-XFL.COM

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	36
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.6K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 35x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VQFN Exposed Pad
Supplier Device Package	44-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf46k40t-i-ml

TABLE 3-1: SUMMARY OF CONFIGURATION WORDS

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprogrammed Value
30 0000h	CONFIG1L	—	RSTOSC2	RSTOSC1	RSTOSC0	—	FEXTOSC2	FEXTOSC1	FEXTOSC0	1111 1111
30 0001h	CONFIG1H	—	—	FCMEN	—	CSWEN	—	—	$\overline{\text{CLKOUTEN}}$	1111 1111
30 0002h	CONFIG2L	BOREN1	BOREN0	$\overline{\text{LPBOREN}}$	—	—	—	$\overline{\text{PWRTE}}$	MCLRE	1111 1111
30 0003h	CONFIG2H	$\overline{\text{XINST}}$	—	$\overline{\text{DEBUG}}$	STVREN	PPS1WAY	$\overline{\text{ZCD}}$	BORV1	BORV0	1111 1111
30 0004h	CONFIG3L	—	WDTE<1:0>		WDTCPS<4:0>					1111 1111
30 0005h	CONFIG3H	—	—	WDTCCS<2:0>			WDTCWS<2:0>			1111 1111
30 0006h	CONFIG4L	WRT7	WRT6	WRT5	WRT4	WRT3	WRT2	WRT1	WRT0	1111 1111
30 0007h	CONFIG4H	—	—	LVP	SCANE	—	WRTD	WRTB	WRTC	1111 1111
30 0008h	CONFIG5L	—	—	—	—	—	—	$\overline{\text{CPD}}$	$\overline{\text{CP}}$	1111 1111
30 000Ah	CONFIG6L	EBTR7	EBTR6	EBTR5	EBTR4	EBTR3	EBTR2	EBTR1	EBTR0	1111 1111
30 000Bh	CONFIG6H	—	—	—	—	—	—	EBTRB	—	1111 1111

Preliminary

REGISTER 9-2: WDTCON1: WATCHDOG TIMER CONTROL REGISTER 1

U-0	R/W(3)-q/q(1)	R/W(3)-q/q(1)	R/W(3)-q/q(1)	U-0	R/W(4)-q/q(2)	R/W(4)-q/q(2)	R/W(4)-q/q(2)
—	WDTCS<2:0>			—	WINDOW<2:0>		
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	q = Value depends on condition

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **WDTCS<2:0>:** Watchdog Timer Clock Select bits
111 = Reserved
•
•
•
010 = Reserved
001 = MFINTOSC 31.25 kHz
000 = LFINTOSC 31 kHz

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **WINDOW<2:0>:** Watchdog Timer Window Select bits

WINDOW<2:0>	Window delay Percent of time	Window opening Percent of time
111	N/A	100
110	12.5	87.5
101	25	75
100	37.5	62.5
011	50	50
010	62.5	37.5
001	75	25
000	87.5	12.5

Note 1: If WDTCCS <2:0> in CONFIG3H = 111, the Reset value of WDTCS<2:0> is 000.
2: The Reset value of WINDOW<2:0> is determined by the value of WDTCWS<2:0> in the CONFIG3H register.
3: If WDTCCS<2:0> in CONFIG3H ≠ 111, these bits are read-only.
4: If WDTCWS<2:0> in CONFIG3H ≠ 111, these bits are read-only.

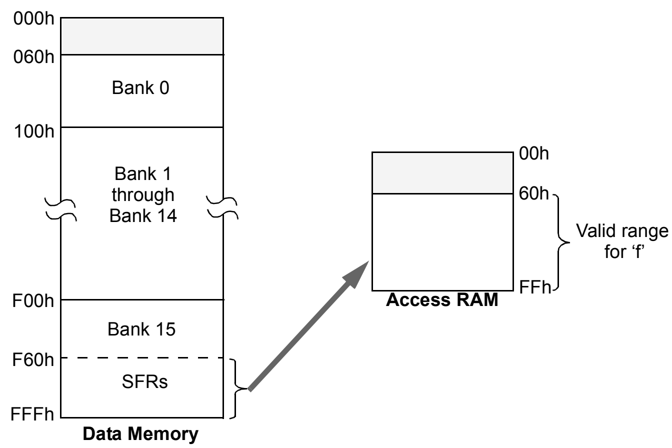
FIGURE 10-7: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)

EXAMPLE INSTRUCTION: `ADDWF, f, d, a` (Opcode: `0010 01da ffff ffff`)

When ‘a’ = `0` and f ≥ 60h:

The instruction executes in Direct Forced mode. ‘f’ is interpreted as a location in the Access RAM between 060h and 0FFh. This is the same as locations F60h to FFFh (Bank 15) of data memory.

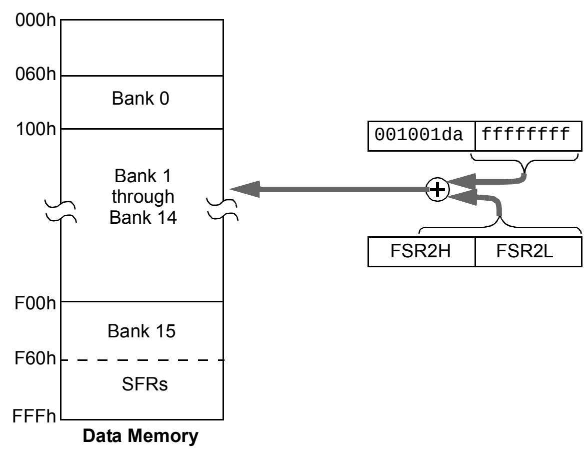
Locations below 60h are not available in this addressing mode.



When ‘a’ = `0` and f ≤ 5Fh:

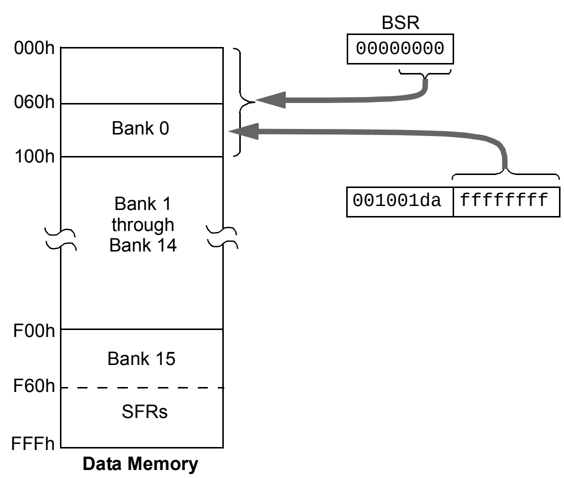
The instruction executes in Indexed Literal Offset mode. ‘f’ is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

Note that in this mode, the correct syntax is now:
`ADDWF [k], d`
where ‘k’ is the same as ‘f’.



When ‘a’ = `1` (all values of f):

The instruction executes in Direct mode (also known as Direct Long mode). ‘f’ is interpreted as a location in one of the 16 banks of the data memory space. The bank is designated by the Bank Select Register (BSR). The address can be in any implemented bank in the data memory space.



11.3.5 WRITE VERIFY

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

EXAMPLE 11-5: DATA EEPROM READ

```
; Data Memory Address to read
             BCF NVMCON1, NVMREG0  ; Setup Data EEPROM Access
             BCF NVMCON1, NVMREG1  ; Setup Data EEPROM Access
             MOVF EE_ADDRL, W      ;
             MOVWF NVMADRL         ; Setup Address low byte
             MOVF EE_ADDRH, W      ;
             MOVWF NVMADRH         ; Setup Address high byte (if applicable)
             BSF NVMCON1, RD       ; Issue EE Read
             MOVF NVMDAT, W        ; W = EE_DATA
```

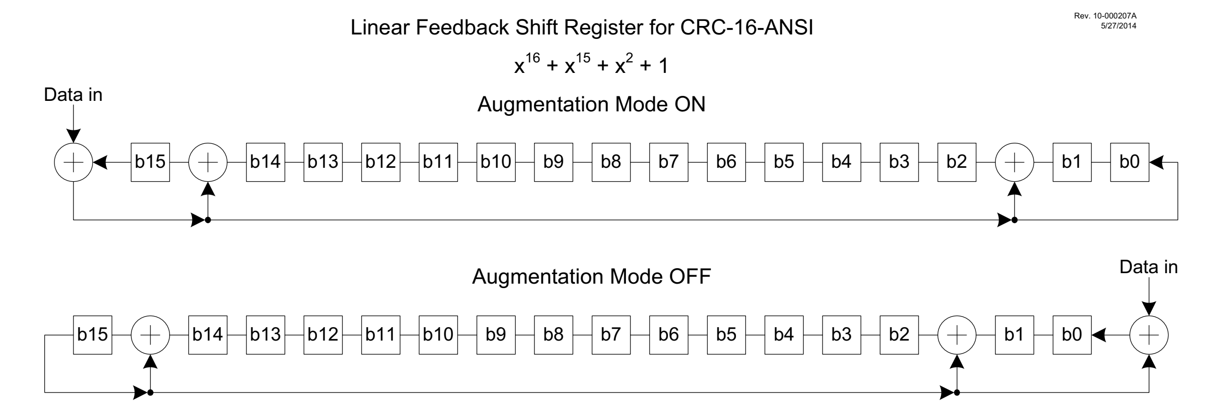
EXAMPLE 11-6: DATA EEPROM WRITE

```
; Data Memory Address to write
           BCF      NVMCON1, NVMREG0 ; Setup Data EEPROM access
           BCF      NVMCON1, NVMREG1 ; Setup Data EEPROM access
           MOVF     EE_ADDRL, W      ;
           MOVWF    NVMADRL          ; Setup Address low byte
           MOVF     EE_ADDRH, W      ;
           MOVWF    NVMADRH          ; Setup Address high byte (if applicable)
; Data Memory Value to write
           MOVF     EE_DATA, W       ;
           MOVWF    NVMDAT           ;
; Enable writes
           BSF      NVMCON1, WREN    ;
; Disable interrupts
           BCF      INTCON, GIE      ;
; Required unlock sequence
           MOVLW    55h              ;
           MOVWF    NVMCON2          ;
           MOVLW    AAh              ;
           MOVWF    NVMCON2          ;
; Set WR bit to begin write
           BSF      NVMCON1, WR      ;
; Wait for write to complete
           BTFSC    NVMCON1, WR
           BRA      $-2
; Enable INT
           BSF      INTCON, GIE      ;
; Disable writes
           BCF      NVMCON1, WREN    ;
```

13.4 CRC Polynomial Implementation

Any polynomial can be used. The polynomial and accumulator sizes are determined by the PLEN<3:0> bits. For an n-bit accumulator, PLEN = n-1 and the corresponding polynomial is n+1 bits. Therefore, the accumulator can be any size up to 16 bits with a corresponding polynomial up to 17 bits. The MSb and LSb of the polynomial are always '1' which is forced by hardware. All polynomial bits between the MSb and LSb are specified by the CRCXOR registers. For example, when using CRC16-ANSI, the polynomial is defined as $X^{16}+X^{15}+X^2+1$. The X^{16} and X^0 = 1 terms are the MSb and LSb controlled by hardware. The X^{15} and X^2 terms are specified by setting the corresponding CRCXOR<15:0> bits with the value of 0x8004. The actual value is 0x8005 because the hardware sets the LSb to 1. However, the LSb of the CRCXORL register is unimplemented and always reads as '0'. Refer to Example 13-1.

EXAMPLE 13-2: CRC LFSR EXAMPLE



13.5 CRC Data Sources

Data can be input to the CRC module in two ways:

- User data using the CRCDATA registers (CRCDATH and CRCDATL)
- Flash using the Program Memory Scanner

To set the number of bits of data, up to 16 bits, the DLEN bits of CRCCON1 must be set accordingly. Only data bits in CRCDATA registers up to DLEN will be used, other data bits in CRCDATA registers will be ignored.

Data is moved into the CRCSHIFT as an intermediate to calculate the check value located in the CRCACC registers.

The SHIFTM bit is used to determine the bit order of the data being shifted into the accumulator. If SHIFTM is not set, the data will be shifted in MSb first (Big Endian). The value of DLEN will determine the MSb. If SHIFTM bit is set, the data will be shifted into the accumulator in reversed order, LSb first (Little Endian).

The CRC module can be seeded with an initial value by setting the CRCACC<15:0> registers to the appropriate value before beginning the CRC.

13.5.1 CRC FROM USER DATA

To use the CRC module on data input from the user, the user must write the data to the CRCDAT registers. The data from the CRCDAT registers will be latched into the shift registers on any write to the CRCDATL register.

13.5.2 CRC FROM FLASH

To use the CRC module on data located in Flash memory, the user can initialize the Program Memory Scanner as defined in **Section 13.9, Program Memory Scan Configuration**.

REGISTER 14-18: IPR0: PERIPHERAL INTERRUPT PRIORITY REGISTER 0

U-0	U-0	R/W-1/1	R/W-1/1	U-0	R/W-1/1	R/W-1/1	R/W-1/1
—	—	TMR0IP	IOCIP	—	INT2IP	INT1IP	INT0IP
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 7-6 **Unimplemented:** Read as ‘0’

bit 5 **TMR0IP:** Timer0 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **IOCIP:** Interrupt-on-Change Priority bit
1 = High priority
0 = Low priority

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **INT2IP:** External Interrupt 2 Priority bit
1 = High priority
0 = Low priority

bit 1 **INT1IP:** External Interrupt 1 Priority bit
1 = High priority
0 = Low priority

bit 0 **INT0IP:** External Interrupt 0 Priority bit
1 = High priority
0 = Low priority

19.8.2 TIMER1/3/5 GATE SOURCE SELECTION

The gate source for Timer1/3/5 can be selected using the GSS<3:0> bits of the TMRxGATE register (Register 19-4). The polarity selection for the gate source is controlled by the TxGPOL bit of the TxGCON register (Register 19-2).

Any of the above mentioned signals can be used to trigger the gate. The output of the CMPx can be synchronized to the Timer1/3/5 clock or left asynchronous. For more information see **Section 32.5.1 “Comparator Output Synchronization”**.

19.8.3 TIMER1/3/5 GATE TOGGLE MODE

When Timer1/3/5 Gate Toggle mode is enabled, it is possible to measure the full-cycle length of a Timer1/3/5 gate signal, as opposed to the duration of a single level pulse.

The Timer1/3/5 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See Figure 19-5 for timing details.

Timer1/3/5 Gate Toggle mode is enabled by setting the GTM bit of the TxGCON register. When the GTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

Note:	Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation.

19.8.4 TIMER1/3/5 GATE SINGLE-PULSE MODE

When Timer1/3/5 Gate Single-Pulse mode is enabled, it is possible to capture a single-pulse gate event. Timer1/3/5 Gate Single-Pulse mode is first enabled by setting the GSPM bit in the TxGCON register. Next, the GGO/$\overline{\text{DONE}}$ bit in the TxGCON register must be set. The Timer1/3/5 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the GGO/$\overline{\text{DONE}}$ bit will automatically be cleared. No other gate events will be allowed to increment Timer1/3/5 until the GGO/$\overline{\text{DONE}}$ bit is once again set in software.

Clearing the TxGSPM bit of the TxGCON register will also clear the GGO/$\overline{\text{DONE}}$ bit. See Figure 19-6 for timing details.

Enabling the Toggle mode and the Single-Pulse mode simultaneously will permit both sections to work together. This allows the cycle times on the Timer1/3/5 gate source to be measured. See Figure 19-7 for timing details.

19.8.5 TIMER1/3/5 GATE VALUE STATUS

When Timer1/3/5 Gate Value Status is utilized, it is possible to read the most current level of the gate control value. The value is stored in the GVAL bit in the TxGCON register. The GVAL bit is valid even when the Timer1/3/5 gate is not enabled (GE bit is cleared).

19.8.6 TIMER1/3/5 GATE EVENT INTERRUPT

When Timer1/3/5 Gate Event Interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of GVAL occurs, the TMRxGIF flag bit in the PIR5 register will be set. If the TMRxGIE bit in the PIE5 register is set, then an interrupt will be recognized.

The TMRxGIF flag bit operates even when the Timer1/3/5 gate is not enabled (GE bit is cleared).

For more information on selecting high or low priority status for the Timer1/3/5 Gate Event Interrupt see **Section 14.0 “Interrupts”**.

20.5.8 LEVEL RESET, EDGE-TRIGGERED HARDWARE LIMIT ONE-SHOT MODES

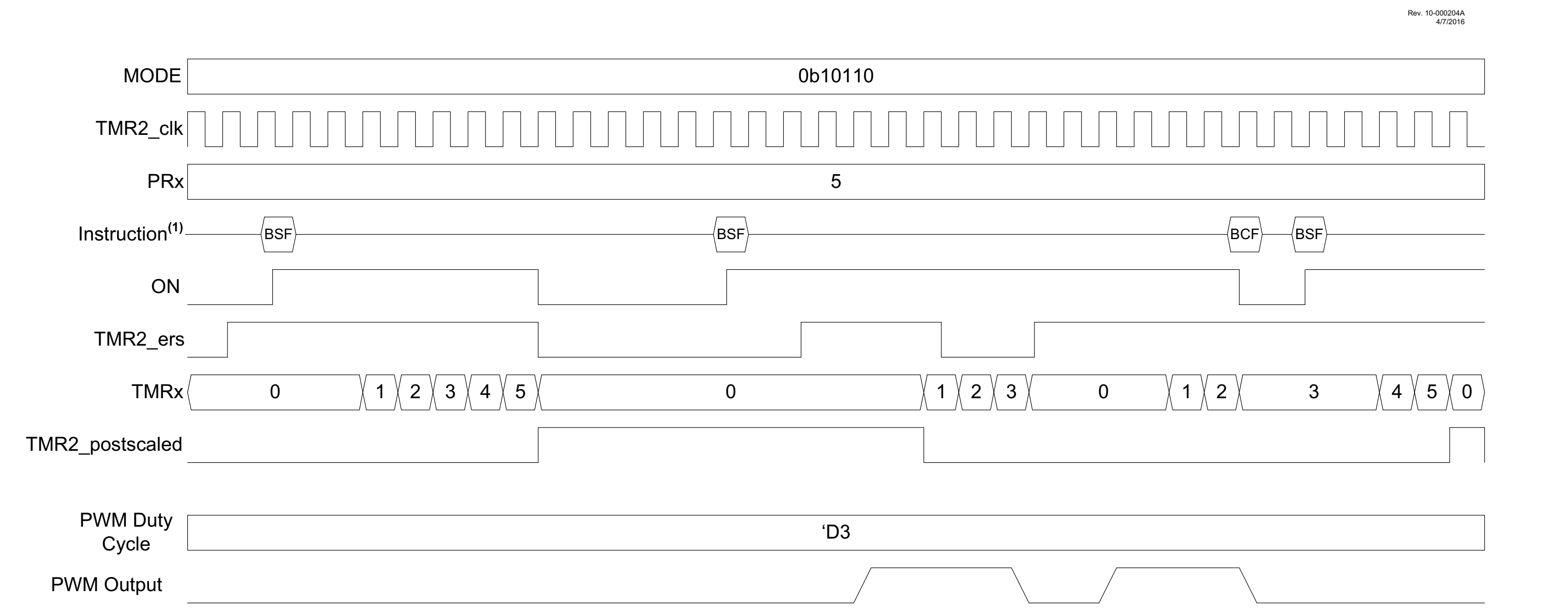
In Level -Triggered One-Shot mode the timer count is reset on the external signal level and starts counting on the rising/falling edge of the transition from Reset level to the active level while the ON bit is set. Reset levels are selected as follows:

- Low Reset level (MODE<4:0> = `01110`)
- High Reset level (MODE<4:0> = `01111`)

When the timer count matches the PRx period count, the timer is reset and the ON bit is cleared. When the ON bit is cleared by either a PRx match or by software control a new external signal edge is required after the ON bit is set to start the counter.

When Level-Triggered Reset One-Shot mode is used in conjunction with the CCP PWM operation the PWM drive goes active with the external signal edge that starts the timer. The PWM drive goes inactive when the timer count equals the CCPRx pulse width count. The PWM drive does not go active when the timer count clears at the PRx period count match.

FIGURE 20-13: LEVEL-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE TIMING DIAGRAM (MODE = `10110`)



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 22-3: PWMxDCH: PWM DUTY CYCLE HIGH BITS

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
DCH<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 **DCh<7:0>:** PWM Duty Cycle Most Significant bits
These bits are the MSbs of the PWM duty cycle. The two LSbs are found in PWMxDCL Register.

REGISTER 22-4: PWMxDCL: PWM DUTY CYCLE LOW BITS

R/W-x/u	R/W-x/u	U-0	U-0	U-0	U-0	U-0	U-0
DCL<7:6>		—	—	—	—	—	—
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 7-6 **DC<8:9>:** PWM Duty Cycle Least Significant bits
These bits are the LSbs of the PWM duty cycle. The MSbs are found in PWMxDCH Register.

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

FIGURE 26-7: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

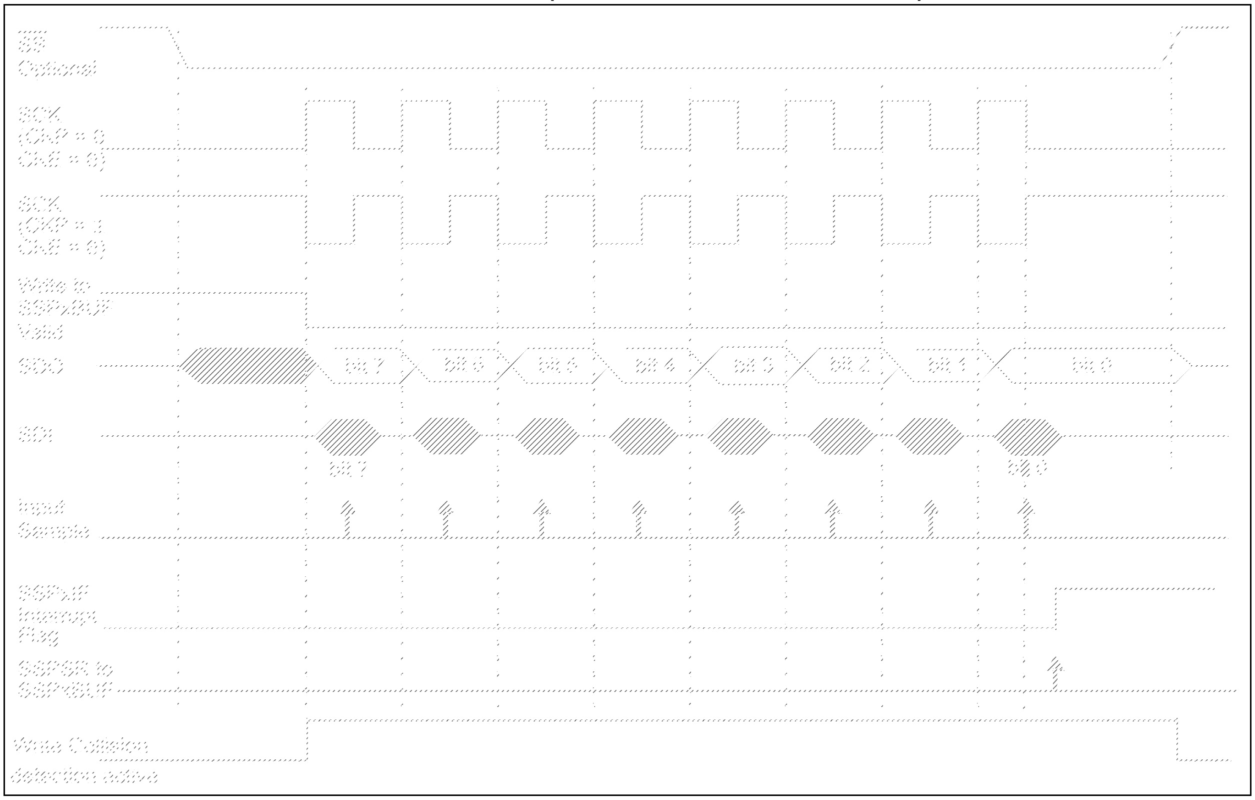
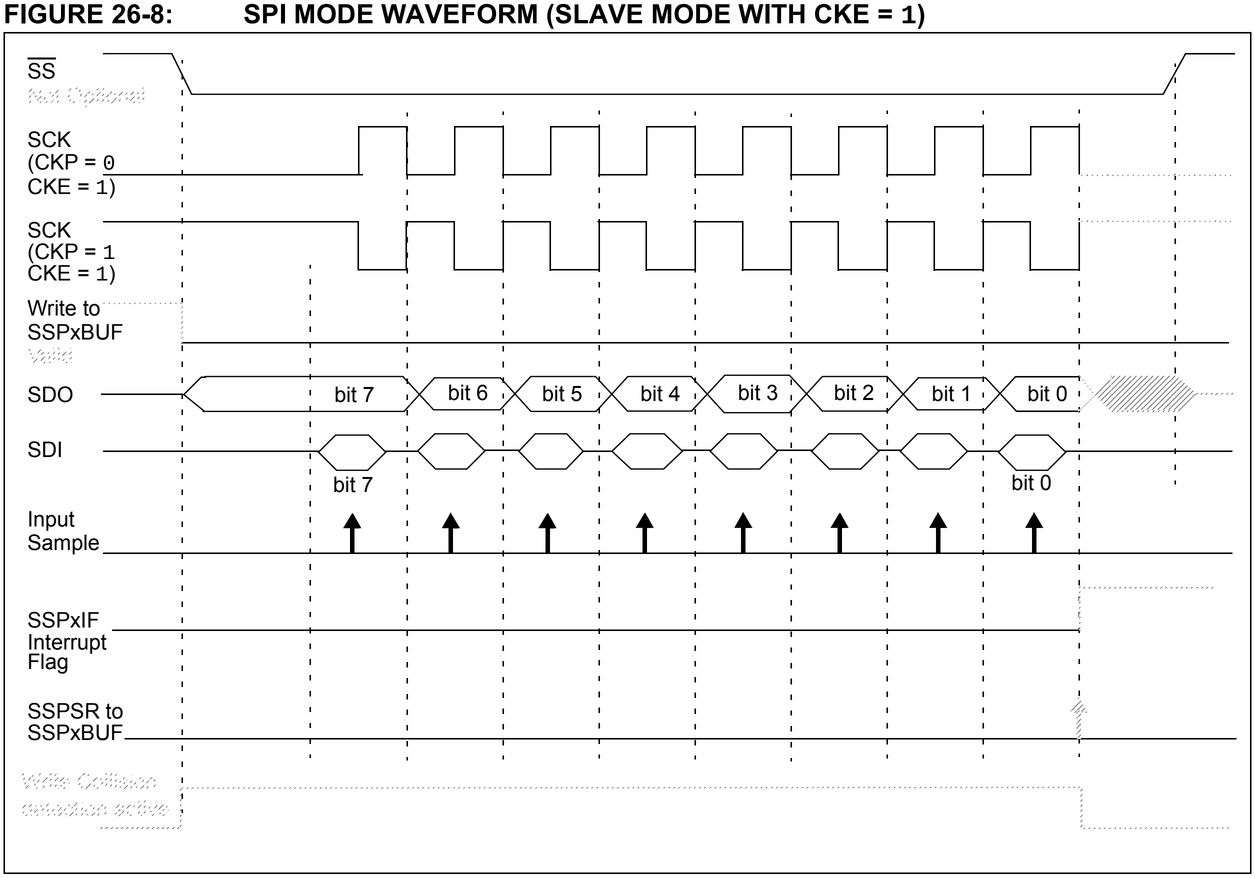


FIGURE 26-8: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)



REGISTER 26-9: SSPxCON3: MSSPx CONTROL REGISTER 3 (I^2C MASTER MODE)

R/HS/HC-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
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	HS/HC = Bit is set/cleared by hardware
x = Bit is unknown	‘0’ = Bit is cleared	

bit 7 **ACKTIM:** Acknowledge Time Status bit
Unused in Master mode.

bit 6 **PCIE:** Stop Condition Interrupt Enable bit(1)
1 = Enable interrupt on detection of Stop condition
0 = Stop detection interrupts are disabled

bit 5 **SCIE:** Start Condition Interrupt Enable bit(1)
1 = Enable interrupt on detection of Start or Restart conditions
0 = Start detection interrupts are disabled

bit 4 **BOEN:** Buffer Overwrite Enable bit
1 = SSPxBUF is updated every time a new data byte is available, ignoring the SSPOV effect on updating the buffer
0 = SSPxBUF is only updated when SSPOV is clear

bit 3 **SDAHT:** SDA Hold Time Selection bit
1 = Minimum of 300ns hold time on SDA after the falling edge of SCL
0 = Minimum of 100ns hold time on SDA after the falling edge of SCL

bit 2 **SBCDE:** Slave Mode Bus Collision Detect Enable bit
Unused in Master mode.

bit 1 **AHEN:** Address Hold Enable bit
Unused in Master mode.

bit 0 **DHEN:** Data Hold Enable bit
Unused in Master mode.

Note 1: This bit has no effect when SSPM<3:0> = 1111 or 1110.In these Slave modes the START and STOP condition interrupts are always enabled.

REGISTER 26-10: SSPxBUF: MSSP DATA BUFFER REGISTER (I^2C MASTER MODE)

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

26.10 I²C Master Mode

Master mode is enabled by setting and clearing the appropriate SSPM bits in the SSPxCON1 register and by setting the SSPEN bit. In Master mode, the SDA and SCK pins must be configured as inputs. The MSSP peripheral hardware will override the output driver TRIS controls when necessary to drive the pins low.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I²C bus may be taken when the P bit is set, or the bus is Idle.

In Firmware Controlled Master mode, user code conducts all I²C bus operations based on Start and Stop bit condition detection. Start and Stop condition detection is the only active circuitry in this mode. All other communication is done by the user software directly manipulating the SDA and SCL lines.

The following events will cause the SSP Interrupt Flag bit, SSPxIF, to be set (SSP interrupt, if enabled):

- Start condition detected
- Stop condition detected
- Data transfer byte transmitted/received
- Acknowledge transmitted/received
- Repeated Start generated

Note 1: The MSSP module, when configured in I²C Master mode, does not allow queuing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPxBUF register to initiate transmission before the Start condition is complete. In this case, the SSPxBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPxBUF did not occur

2: Master mode suspends Start/Stop detection when sending the Start/Stop condition by means of the SEN/PEN control bits. The SSPxIF bit is set at the end of the Start/Stop generation when hardware clears the control bit.

26.10.1 I²C MASTER MODE OPERATION

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I²C bus will not be released.

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/$\overline{\text{Write}}$ (R/$\overline{\text{W}}$) bit. In this case, the R/$\overline{\text{W}}$ bit will be logic ‘0’. Serial data is transmitted eight bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/$\overline{\text{W}}$ bit. In this case, the R/$\overline{\text{W}}$ bit will be logic ‘1’. Thus, the first byte transmitted is a 7-bit slave address followed by a ‘1’ to indicate the receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received eight bits at a time. After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

A Baud Rate Generator is used to set the clock frequency output on SCL. See **Section 26.11 “Baud Rate Generator”** for more detail.

26.10.2 CLOCK ARBITRATION

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, releases the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count in the event that the clock is held low by an external device (Figure 26-25).

26.10.13.2 Bus Collision During a Repeated Start Condition

During a Repeated Start condition, a bus collision occurs if:

a) A low level is sampled on SDA when SCL goes from low level to high level (Case 1).
b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1' (Case 2).

When the user releases SDA and the pin is allowed to float high, the BRG is loaded with SSPxADD and counts down to zero. The SCL pin is then deasserted and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', Figure 26-36). If SDA is sampled high, the BRG is reloaded and begins counting. If SDA goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

If SCL goes from high-to-low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition, see Figure 26-37.

If, at the end of the BRG time out, both SCL and SDA are still high, the SDA pin is driven low and the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCL pin, the SCL pin is driven low and the Repeated Start condition is complete.

FIGURE 26-36: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

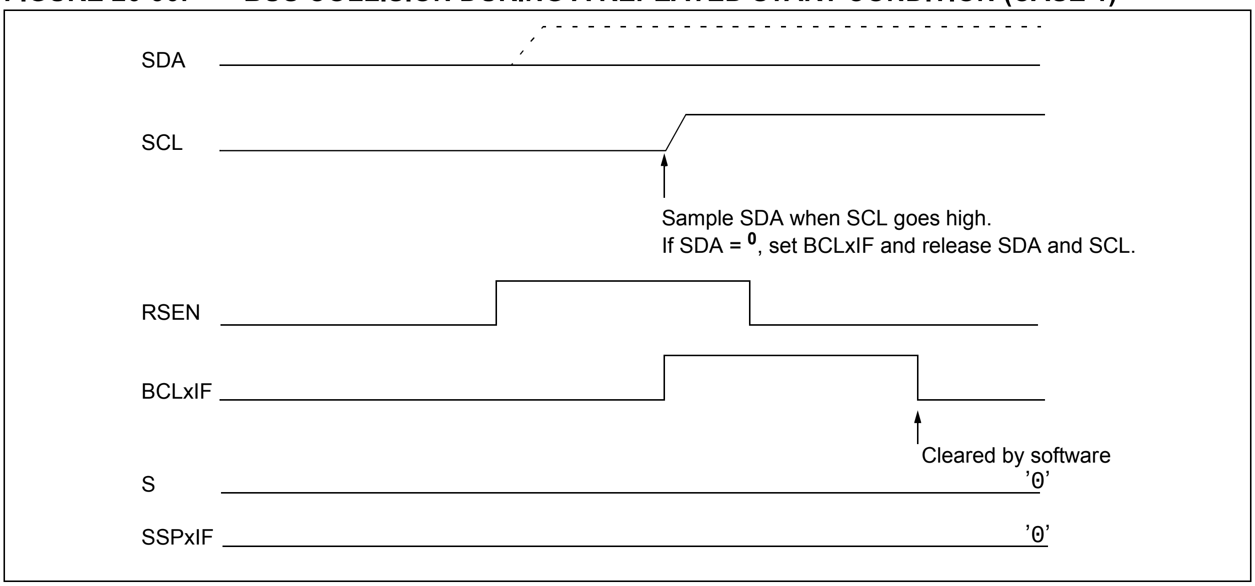
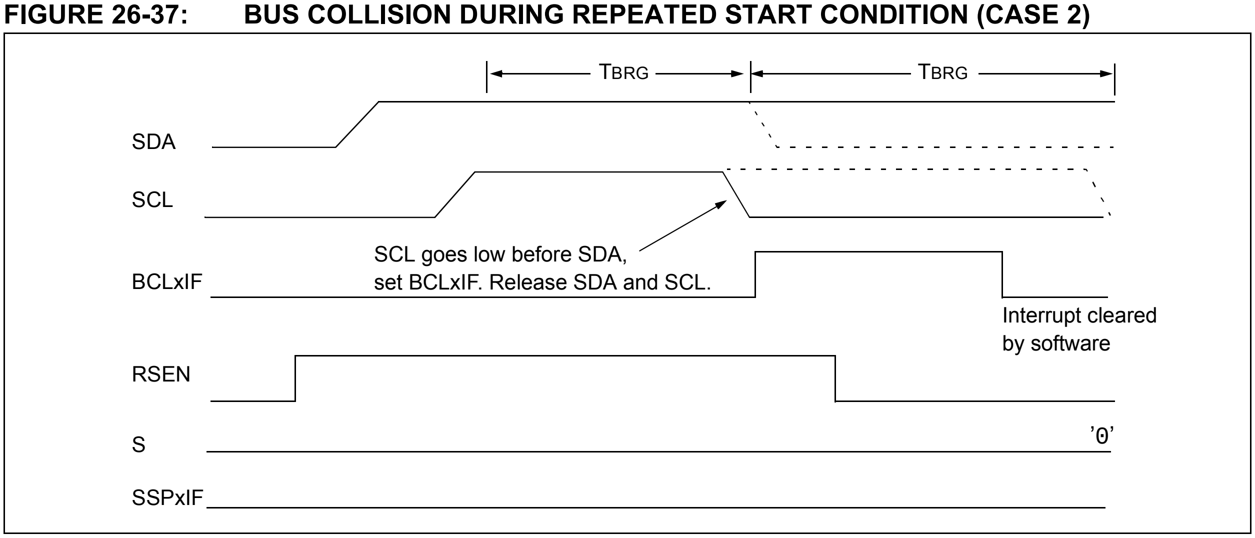


FIGURE 26-37: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



27.2 EUSART Asynchronous Mode

The EUSART transmits and receives data using the standard non-return-to-zero (NRZ) format. NRZ is implemented with two levels: a V_{OH} Mark state which represents a '1' data bit, and a V_{OL} Space state which represents a '0' data bit. NRZ refers to the fact that consecutively transmitted data bits of the same value stay at the output level of that bit without returning to a neutral level between each bit transmission. An NRZ transmission port idles in the Mark state. Each character transmission consists of one Start bit followed by eight or nine data bits and is always terminated by one or more Stop bits. The Start bit is always a space and the Stop bits are always marks. The most common data format is eight bits. Each transmitted bit persists for a period of 1/(Baud Rate). An on-chip dedicated 8-bit/16-bit Baud Rate Generator is used to derive standard baud rate frequencies from the system oscillator. See Table 27-5 for examples of baud rate configurations.

The EUSART transmits and receives the LSb first. The EUSART's transmitter and receiver are functionally independent, but share the same data format and baud rate. Parity is not supported by the hardware, but can be implemented in software and stored as the ninth data bit.

27.2.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 27-1. The heart of the transmitter is the serial Transmit Shift Register (TSR), which is not directly accessible by software. The TSR obtains its data from the transmit buffer, which is the TXxREG register.

27.2.1.1 Enabling the Transmitter

The EUSART transmitter is enabled for asynchronous operations by configuring the following three control bits:

- TXEN = 1
- SYNC = 0
- SPEN = 1

All other EUSART control bits are assumed to be in their default state.

Setting the TXEN bit of the TXxSTA register enables the transmitter circuitry of the EUSART. Clearing the SYNC bit of the TXxSTA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCxSTA register enables the EUSART and automatically configures the TXx/CKx I/O pin as an output. If the TXx/CKx pin is shared with an analog peripheral, the analog I/O function must be disabled by clearing the corresponding ANSEL bit.

Note:	The TXxIF Transmitter Interrupt flag is set when the TXEN enable bit is set.

27.2.1.2 Transmitting Data

A transmission is initiated by writing a character to the TXxREG register. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXxREG is immediately transferred to the TSR register. If the TSR still contains all or part of a previous character, the new character data is held in the TXxREG until the Stop bit of the previous character has been transmitted. The pending character in the TXxREG is then transferred to the TSR in one T_{CY} immediately following the Stop bit transmission. The transmission of the Start bit, data bits and Stop bit sequence commences immediately following the transfer of the data to the TSR from the TXxREG.

27.2.1.3 Transmit Data Polarity

The polarity of the transmit data can be controlled with the SCKP bit of the BAUDxCON register. The default state of this bit is '0' which selects high true transmit idle and data bits. Setting the SCKP bit to '1' will invert the transmit data resulting in low true idle and data bits. The SCKP bit controls transmit data polarity in Asynchronous mode only. In Synchronous mode, the SCKP bit has a different function. See **Section 27.5.1.2 "Clock Polarity"**.

27.2.1.4 Transmit Interrupt Flag

The TXxIF interrupt flag bit of the PIR3 register is set whenever the EUSART transmitter is enabled and no character is being held for transmission in the TXxREG. In other words, the TXxIF bit is only clear when the TSR is busy with a character and a new character has been queued for transmission in the TXxREG. The TXxIF flag bit is not cleared immediately upon writing TXxREG. TXxIF becomes valid in the second instruction cycle following the write execution. Polling TXxIF immediately following the TXxREG write will return invalid results. The TXxIF bit is read-only, it cannot be set or cleared by software.

The TXxIF interrupt can be enabled by setting the TXxIE interrupt enable bit of the PIE3 register. However, the TXxIF flag bit will be set whenever the TXxREG is empty, regardless of the state of TXxIE enable bit.

To use interrupts when transmitting data, set the TXxIE bit only when there is more data to send. Clear the TXxIE interrupt enable bit upon writing the last character of the transmission to the TXxREG.

31.2.6 ADC CONVERSION PROCEDURE (BASIC MODE)

This is an example procedure for using the ADC to perform an Analog-to-Digital conversion:

1. Configure Port:
   - Disable pin output driver (Refer to the TRISx register)
   - Configure pin as analog (Refer to the ANSELx register)
2. Configure the ADC module:
   - Select ADC conversion clock
   - Configure voltage reference
   - Select ADC input channel (precharge+acquisition)
   - Turn on ADC module
3. Configure ADC interrupt (optional):
   - Clear ADC interrupt flag
   - Enable ADC interrupt
   - Enable peripheral interrupt (PEIE bit)
   - Enable global interrupt (GIE bit)[(1)]
4. If ADACQ = 0, software must wait the required acquisition time[(2)].
5. Start conversion by setting the ADGO bit.
6. Wait for ADC conversion to complete by one of the following:
   - Polling the ADGO bit
   - Waiting for the ADC interrupt (interrupts enabled)
7. Read ADC Result.
8. Clear the ADC interrupt flag (required if interrupt is enabled).

Note 1: The global interrupt can be disabled if the user is attempting to wake-up from Sleep and resume in-line code execution.

2: Refer to **Section 31.3 "ADC Acquisition Requirements"**.

EXAMPLE 31-1: ADC CONVERSION

```
;This code block configures the ADC
;for polling, VDD and VSS references, FRC
;oscillator and AN0 input.
;
;Conversion start & polling for completion
;are included.
;
BANKSEL   ADCON1       ;
MOVLW     B'11110000'  ;Right justify,
                       ;FRC oscillator
MOVWF     ADCON1       ;Vdd and Vss Vref
BANKSEL   TRISA        ;
BSF       TRISA,0      ;Set RA0 to input
BANKSEL   ANSEL        ;
BSF       ANSEL,0      ;Set RA0 to analog
BANKSEL   ADCON0       ;
MOVLW     B'00000001'  ;Select channel AN0
MOVWF     ADCON0       ;Turn ADC On
CALL      SampleTime   ;Acquisiton delay
BSF       ADCON0,ADGO  ;Start conversion
BTFSC     ADCON0,ADGO  ;Is conversion done?
GOTO      $-1          ;No, test again
BANKSEL   ADRESH       ;
MOVF      ADRESH,W     ;Read upper 2 bits
MOVWF     RESULTHI     ;store in GPR space
BANKSEL   ADRESL       ;
MOVF      ADRESL,W     ;Read lower 8 bits
MOVWF     RESULTLO     ;Store in GPR space
```

REGISTER 31-16: ADRESH: ADC RESULT REGISTER HIGH, ADFM = 0

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
ADRES<9:2>							
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 **ADRES<9:2>**: ADC Result Register bits
Upper eight bits of 10-bit conversion result.

REGISTER 31-17: ADRESL: ADC RESULT REGISTER LOW, ADFM = 0

R/W-x/u	R/W-x/u	U-0	U-0	U-0	U-0	U-0	U-0
ADRES<1:0>		—	—	—	—	—	—
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 7-6 **ADRES<1:0>**: ADC Result Register bits. Lower two bits of 10-bit conversion result.

bit 5-0 **Reserved**: Do not use.

REGISTER 31-18: ADRESH: ADC RESULT REGISTER HIGH, ADFM = 1

U-0	U-0	U-0	U-0	U-0	U-0	R/W-x/u	R/W-x/u
—	—	—	—	—	—	ADRES<9:8>	
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 7-2 **Reserved**: Do not use.

bit 1-0 **ADRES<9:8>**: ADC Sample Result bits. Upper two bits of 10-bit conversion result.

TABLE 35-1: OPCODE FIELD DESCRIPTIONS

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
bbb	Bit address within an 8-bit file register (0 to 7).
BSR	Bank Select Register. Used to select the current RAM bank.
C, DC, Z, OV, N	ALU Status bits: **C**arry, **D**igit **C**arry, **Z**ero, **Ov**erflow, **N**egative.
d	Destination select bit d = 0: store result in WREG d = 1: store result in file register f
dest	Destination: either the WREG register or the specified register file location.
f	8-bit Register file address (00h to FFh) or 2-bit FSR designator (0h to 3h).
f_s	12-bit Register file address (000h to FFFh). This is the source address.
f_d	12-bit Register file address (000h to FFFh). This is the destination address.
GIE	Global Interrupt Enable bit.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value).
label	Label name.
mm	The mode of the TBLPTR register for the table read and table write instructions. Only used with table read and table write instructions:
*	No change to register (such as TBLPTR with table reads and writes)
*+	Post-Increment register (such as TBLPTR with table reads and writes)
*-	Post-Decrement register (such as TBLPTR with table reads and writes)
+*	Pre-Increment register (such as TBLPTR with table reads and writes)
n	The relative address (2's complement number) for relative branch instructions or the direct address for CALL/BRANCH and RETURN instructions.
PC	Program Counter.
PCL	Program Counter Low Byte.
PCH	Program Counter High Byte.
PCLATH	Program Counter High Byte Latch.
PCLATU	Program Counter Upper Byte Latch.
$\overline{PD}$	Power-down bit.
PRODH	Product of Multiply High Byte.
PRODL	Product of Multiply Low Byte.
s	Fast Call/Return mode select bit s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
TBLPTR	21-bit Table Pointer (points to a Program Memory location).
TABLAT	8-bit Table Latch.
$\overline{TO}$	Time-out bit.
TOS	Top-of-Stack.
u	Unused or unchanged.
WDT	Watchdog Timer.
WREG	Working register (accumulator).
x	Don't care ('0' or '1'). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
z_s	7-bit offset value for indirect addressing of register files (source).
z_d	7-bit offset value for indirect addressing of register files (destination).
{ }	Optional argument.
[text]	Indicates an indexed address.
(text)	The contents of text.
[expr]<n>	Specifies bit n of the register indicated by the pointer expr.
→	Assigned to.
< >	Register bit field.
∈	In the set of.
italics	User defined term (font is Courier).

DAW — Decimal Adjust W Register

Syntax: DAW

Operands: None

Operation: If [W<3:0> > 9] or [DC = 1] then
(W<3:0>) + 6 → W<3:0>;
else
(W<3:0>) → W<3:0>;

If [W<7:4> + DC > 9] or [C = 1] then
(W<7:4>) + 6 + DC → W<7:4> ;
else
(W<7:4>) + DC → W<7:4>

Status Affected: C

Encoding:

`0000`	`0000`	`0000`	`0111`

Description: `DAW` adjusts the 8-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register W	Process Data	Write W

Example1:

`DAW`

Before Instruction
W = A5h
C = 0
DC = 0

After Instruction
W = 05h
C = 1
DC = 0

Example 2:

Before Instruction
W = CEh
C = 0
DC = 0

After Instruction
W = 34h
C = 1
DC = 0

DECF — Decrement f

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

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

Operation: (f) – 1 → dest

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

Encoding:

`0000`	`01da`	`ffff`	`ffff`

Description: Decrement register ‘f’. If ‘d’ is ‘`0`’, the result is stored 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.
If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (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 to destination

Example: `DECF CNT, 1, 0`

Before Instruction
CNT = 01h
Z = 0

After Instruction
CNT = 00h
Z = 1

POP — Pop Top of Return Stack

Syntax:	POP
Operands:	None
Operation:	(TOS) $\rightarrow$ bit bucket
Status Affected:	None
Encoding:	`0000` `0000` `0000` `0110`
Description:	The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	POP TOS value	No operation

Example:

```
POP
GOTO    NEW
```

Before Instruction

TOS	=	0031A2h
Stack (1 level down)	=	014332h

After Instruction

TOS	=	014332h
PC	=	NEW

PUSH — Push Top of Return Stack

Syntax:	PUSH
Operands:	None
Operation:	(PC + 2) $\rightarrow$ TOS
Status Affected:	None
Encoding:	`0000` `0000` `0000` `0101`
Description:	The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	PUSH PC + 2 onto return stack	No operation	No operation

Example:

```
PUSH
```

Before Instruction

TOS	=	345Ah
PC	=	0124h

After Instruction

PC	=	0126h
TOS	=	0126h
Stack (1 level down)	=	345Ah