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	20MHz
Connectivity	I²C, SPI
Peripherals	Brown-out Detect/Reset, LCD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	7KB (4K x 14)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	176 x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	A/D 5x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	68-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.23x24.23)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lc925-i-l

1.0 DEVICE OVERVIEW

This document contains device-specific information for the following devices:

1. PIC16C925
2. PIC16C926

The PIC16C925/926 series is a family of low cost, high performance, CMOS, fully static, 8-bit microcontrollers with an integrated LCD Driver module, in the PIC16CXXX mid-range family.

For the PIC16C925/926 family, there are two device "types" as indicated in the device number:

1. **C**, as in PIC16C926. These devices operate over the standard voltage range.
2. **LC**, as in PIC16**LC**926. These devices operate over an extended voltage range.

These devices come in 64-pin and 68-pin packages, as well as die form. Both configurations offer identical peripheral devices and other features. The only difference between the DSTEMP and DSTEMP is the additional EPROM and data memory offered in the latter. An overview of features is presented in Table 1-1.

A UV-erasable, CERQUAD packaged version (compatible with PLCC) is also available for both the PIC16C925 and PIC16C926. This version is ideal for cost effective code development.

A block diagram for the PIC16C925/926 family architecture is presented in Figure 1-1.

TABLE 1-1: PIC16C925/926 DEVICE FEATURES

Features	PIC16C925	PIC16C926
Operating Frequency	DC-20 MHz	DC-20 MHz
EPROM Program Memory (words)	4K	8K
Data Memory (bytes)	176	336
Timer Module(s)	TMR0,TMR1,TMR2	TMR0,TMR1,TMR2
Capture/Compare/PWM Module(s)	1	1
Serial Port(s) (SPI/I^2C, USART)	SPI/I^2C	SPI/I^2C
Parallel Slave Port	—	—
A/D Converter (10-bit) Channels	5	5
LCD Module	4 Com, 32 Seg	4 Com, 32 Seg
Interrupt Sources	9	9
I/O Pins	25	25
Input Pins	27	27
Voltage Range (V)	2.5-5.5	2.5-5.5
In-Circuit Serial Programming	Yes	Yes
Brown-out Reset	Yes	Yes
Packages	64-pin TQFP 68-pin PLCC 68-pin CLCC (CERQUAD) Die	64-pin TQFP 68-pin PLCC 68-pin CLCC (CERQUAD) Die

2.0 MEMORY ORGANIZATION

2.1 Program Memory Organization

The PIC16C925/926 family has a 13-bit program counter capable of addressing an 8K x 14 program memory space.

For the PIC16C925, only the first 4K x 14 (0000h-0FFFh) are physically implemented. Accessing a location above the physically implemented addresses will cause a wraparound. The RESET vector is at 0000h and the interrupt vector is at 0004h.

FIGURE 2-1: PROGRAM MEMORY MAP AND STACK FOR DSTEMP

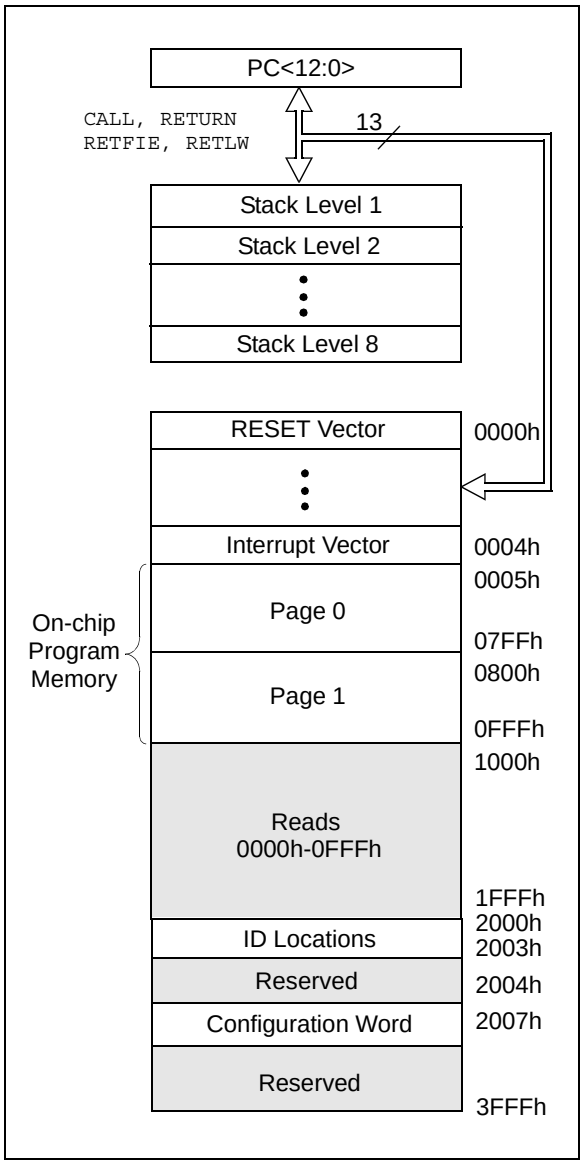
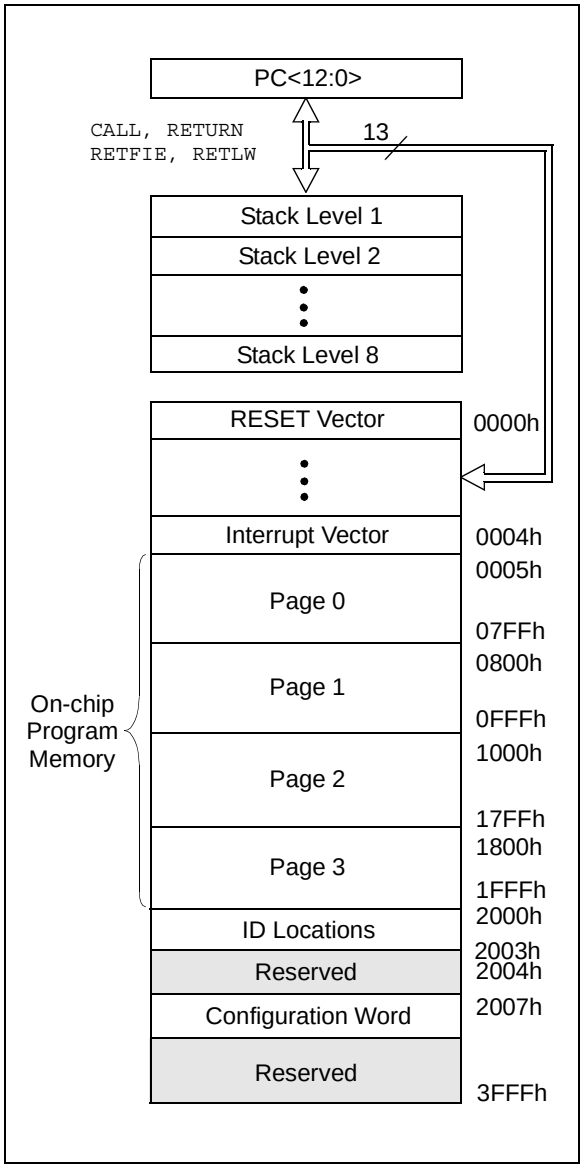


FIGURE 2-2: PROGRAM MEMORY MAP AND STACK FOR DSTEMP



2.3.3 INTCON REGISTER

The INTCON Register is a readable and writable register which contains various enable and flag bits for the TMR0 register overflow, RB Port change and external RB0/INT pin interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>).

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE/GEIL:** Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INTE:** RB0/INT0 External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF:** RB0/INT0 External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

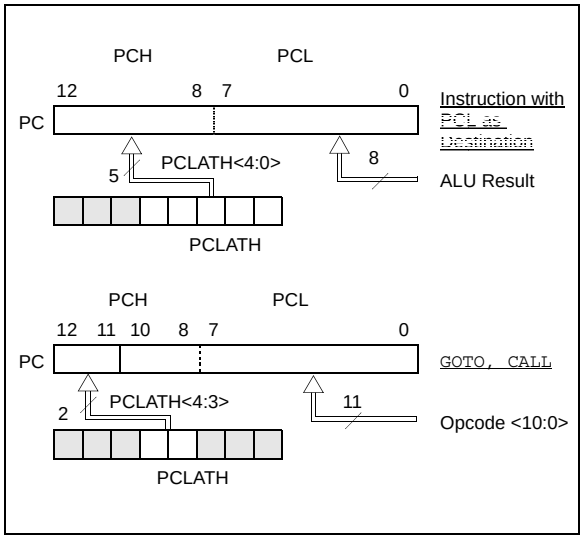
bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

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

2.4 PCL and PCLATH

The program counter (PC) is 13-bits wide. The low byte comes from the PCL register, which is a readable and writable register. The upper bits (PC<12:8>) are not readable, but are indirectly writable through the PCLATH register. On any RESET, the upper bits of the PC will be cleared. Figure 2-5 shows the two situations for the loading of the PC. The upper example in the figure shows how the PC is loaded on a write to PCL (PCLATH<4:0> → PCH). The lower example in the figure shows how the PC is loaded during a CALL or GOTO instruction (PCLATH<4:3> → PCH).

FIGURE 2-5: LOADING OF PC IN DIFFERENT SITUATIONS



2.4.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When doing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256 byte block). Refer to the application note *"Implementing a Table Read"* (AN556).

2.4.2 STACK

The PIC16CXXX family has an 8-level deep x 13-bit wide hardware stack. The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is PUSHed onto the stack when a CALL instruction is executed, or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer. This means that after the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

Note 1: There are no status bits to indicate stack overflow or stack underflow conditions.

2: There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, RETURN, RETLW, and RETFIE instructions, or the vectoring to an interrupt address.

2.5 Program Memory Paging

PIC16C925/926 devices are capable of addressing a continuous 8K word block of program memory. The CALL and GOTO instructions provide only 11-bits of address to allow branching within any 2K program memory page. When doing a CALL or GOTO instruction, the upper 2-bits of the address are provided by PCLATH<4:3>. When doing a CALL or GOTO instruction, the user must ensure that the page select bits are programmed so that the desired program memory page is addressed. If a return from a CALL instruction (or interrupt) is executed, the entire 13-bit PC is pushed onto the stack. Therefore, manipulation of the PCLATH<4:3> bits is not required for the RETURN instructions (which POPs the address from the stack).

Note: The contents of the PCLATH register are unchanged after a RETURN or RETFIE instruction is executed. The user must rewrite the PCLATH for any subsequent CALL or GOTO instructions.

Example 2-1 shows the calling of a subroutine in page 1 of the program memory. This example assumes that PCLATH is saved and restored by the Interrupt Service Routine (if interrupts are used).

EXAMPLE 2-1: CALL OF A SUBROUTINE IN PAGE 1 FROM PAGE 0

```
ORG 0x500
BCF    PCLATH,4
BSF    PCLATH,3  ;Select page 1 (800h-FFFh)
CALL   SUB1_P1   ;Call subroutine in
       :         ;page 1 (800h-FFFh)
       :
       :
ORG 0x900
SUB1_P1:         ;called subroutine
       :         ;page 1 (800h-FFFh)
       :
RETURN           ;return to Call subroutine
                 ;in page 0 (000h-7FFh)
```

2.6 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing.

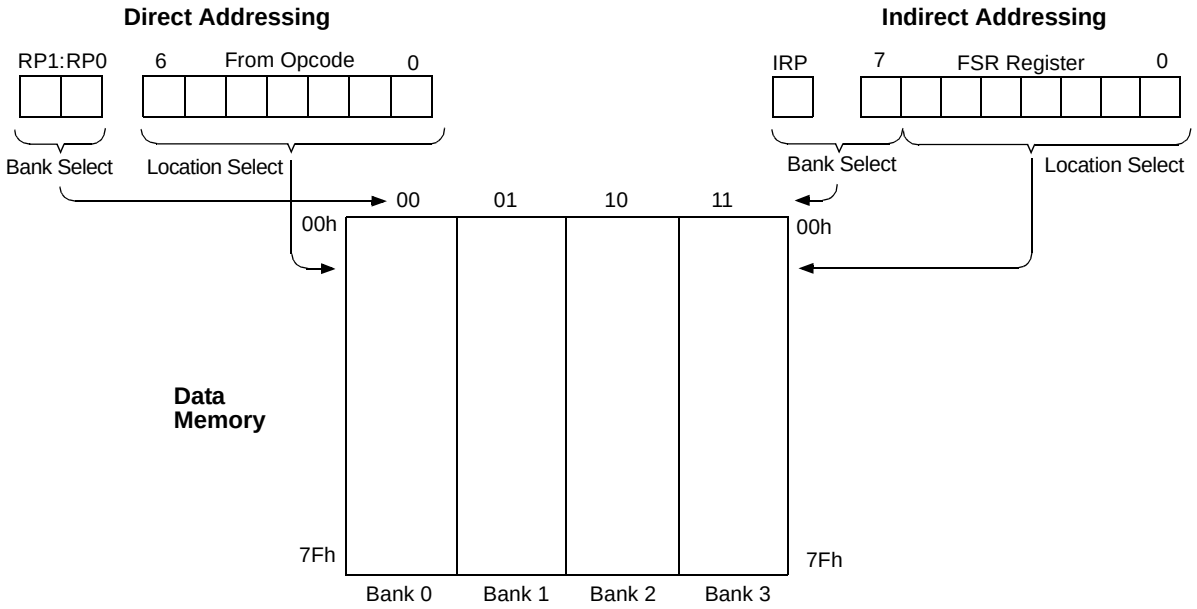
Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses the register pointed to by the File Select Register (FSR). Reading the INDF register itself, indirectly (FSR = '0'), will produce 00h. Writing to the INDF register indirectly results in a no operation (although status bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 2-6.

A simple program to clear RAM locations 20h-2Fh using indirect addressing is shown in Example 2-2.

EXAMPLE 2-2: INDIRECT ADDRESSING

```
            MOVLW  0x20     ;initialize pointer
            MOVWF  FSR      ;to RAM
NEXT        CLRF   INDF     ;clear INDF register
            INCF   FSR,F    ;inc pointer
            BTFSS  FSR,4    ;all done?
            GOTO   NEXT     ;no clear next
CONTINUE
            :               ;yes continue
```

FIGURE 2-6: DIRECT/INDIRECT ADDRESSING



Note: For memory map detail, see Figure 2-3.

TABLE 4-7: PORTD FUNCTIONS

Name	Bit#	Buffer Type	Function
RD0/SEG00	bit0	ST	Input/output port pin or Segment Driver00.
RD1/SEG01	bit1	ST	Input/output port pin or Segment Driver01.
RD2/SEG02	bit2	ST	Input/output port pin or Segment Driver02.
RD3/SEG03	bit3	ST	Input/output port pin or Segment Driver03.
RD4/SEG04	bit4	ST	Input/output port pin or Segment Driver04.
RD5/SEG29/COM3	bit5	ST	Digital input pin or Segment Driver29 or Common Driver3.
RD6/SEG30/COM2	bit6	ST	Digital input pin or Segment Driver30 or Common Driver2.
RD7/SEG31/COM1	bit7	ST	Digital input pin or Segment Driver31 or Common Driver1.

Legend: ST = Schmitt Trigger input

TABLE 4-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	0000 0000	0000 0000
88h	TRISD	PORTD Data Direction Control Register								1111 1111	1111 1111
10Dh	LCDSE	SE29	SE27	SE20	SE16	SE12	SE9	SE5	SE0	1111 1111	1111 1111

Legend: Shaded cells are not used by PORTD.

5.3.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control, i.e., it can be changed “on the fly” during program execution.

Note: To avoid an unintended device RESET, the following instruction sequence (shown in Example 5-1) must be executed when changing the prescaler assignment from Timer0 to the WDT. This precaution must be followed even if the WDT is disabled.

EXAMPLE 5-1: CHANGING PRESCALER (TIMER0→WDT)

Lines 2 and 3 do NOT have to be included if the final desired prescale value is other than 1:1. If 1:1 is final desired value, then a temporary prescale value is set in lines 2 and 3 and the final prescale value will be set in lines 10 and 11.

```
1)  BSF     STATUS, RP0    ;Select Bank1
2)  MOVLW   b'xx0x0xxx'    ;Select clock source and prescale value of
3)  MOVWF   OPTION_REG     ;other than 1:1
4)  BCF     STATUS, RP0    ;Select Bank0
5)  CLRF    TMR0           ;Clear TMR0 and prescaler
6)  BSF     STATUS, RP1    ;Select Bank1
7)  MOVLW   b'xxxx1xxx'    ;Select WDT, do not change prescale value
8)  MOVWF   OPTION_REG     ;
9)  CLRWDT                 ;Clears WDT and prescaler
10) MOVLW   b'xxxx1xxx'    ;Select new prescale value and WDT
11) MOVWF   OPTION_REG     ;
12) BCF     STATUS, RP0    ;Select Bank0
```

To change prescaler from the WDT to the Timer0 module use the precaution shown in Example 5-2.

EXAMPLE 5-2: CHANGING PRESCALER (WDT→TIMER0)

```
CLRWDT              ;Clear WDT and precaler
BSF     STATUS, RP0 ;Select Bank1
MOVLW   b'xxxx0xxx' ;Select TMR0,
                    ;new prescale value and
MOVWF   OPTION_REG  ;clock source
BCF     STATUS, RP0 ;Select Bank0
```

TABLE 5-1: REGISTERS ASSOCIATED WITH TIMER0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
01h, 101h	TMR0	Timer0 Module Register								xxxx xxxx	uuuu uuuu
0Bh, 8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
81h, 181h	OPTION	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	—	—	PORTA Data Direction Control Register						--11 1111	--11 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'.
Shaded cells are not used by Timer0.

6.3 Timer1 Operation in Asynchronous Counter Mode

If control bit $\overline{\text{T1SYNC}}$ (T1CON<2>) is set, the external clock input is not synchronized. The timer continues to increment asynchronous to the internal phase clocks. The timer will continue to run during SLEEP and can generate an interrupt-on-overflow which will wake-up the processor. However, special precautions in software are needed to read from, or write to the Timer1 register pair (TMR1H:TMR1L) (Section 6.3.2).

In Asynchronous Counter mode, Timer1 cannot be used as a time-base for capture or compare operations.

6.3.1 EXTERNAL CLOCK INPUT TIMING WITH UNSYNCHRONIZED CLOCK

If control bit $\overline{\text{T1SYNC}}$ is set, the timer will increment completely asynchronously. The input clock must meet certain minimum high time and low time requirements, as specified in timing parameters 45, 46, and 47.

6.3.2 READING AND WRITING TMR1 IN ASYNCHRONOUS COUNTER MODE

Reading TMR1H or TMR1L, while the timer is running from an external asynchronous clock, will ensure a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself, poses certain problems, since the timer may overflow between the reads.

For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers while the register is incrementing. This may produce an unpredictable value in the timer register.

Reading the 16-bit value requires some care. Example 6-1 is an example routine to read the 16-bit timer value. This is useful if the timer cannot be stopped.

EXAMPLE 6-1: READING A 16-BIT FREE-RUNNING TIMER

```
; All interrupts are disabled
;
            MOVF   TMR1H, W    ;Read high byte
            MOVWF  TMPH        ;
            MOVF   TMR1L, W    ;Read low byte
            MOVWF  TMPL        ;
            MOVF   TMR1H, W    ;Read high byte
            SUBWF  TMPH,  W    ;Sub 1st read with 2nd read
            BTFSC  STATUS,Z    ;Is result = 0
            GOTO   CONTINUE    ;Good 16-bit read
;
; TMR1L may have rolled over between the read of the high and low bytes.
; Reading the high and low bytes now will read a good value.
;
            MOVF   TMR1H, W    ;Read high byte
            MOVWF  TMPH        ;
            MOVF   TMR1L, W    ;Read low byte
            MOVWF  TMPL        ;
; Re-enable the Interrupt (if required)
;
CONTINUE                       ;Continue with your code
```

TABLE 8-3: REGISTERS ASSOCIATED WITH TIMER1, CAPTURE AND COMPARE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
0Bh, 8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	LCDIF	ADIF	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF	00-- 0000	00-- 0000
8Ch	PIE1	LCDIE	ADIE	—	—	SSPIE	CCP1IE	TMR2IE	TMR1IE	00-- 0000	00-- 0000
87h	TRISC	—	—	PORTC Data Direction Control Register						--11 1111	--11 1111
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu
15h	CCPR1L	Capture/Compare/PWM1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used in these modes.

TABLE 8-4: REGISTERS ASSOCIATED WITH PWM AND TIMER2

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
0Bh, 8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	LCDIF	ADIF	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF	00-- 0000	00-- 0000
8Ch	PIE1	LCDIE	ADIE	—	—	SSPIE	CCP1IE	TMR2IE	TMR1IE	00-- 0000	00-- 0000
87h	TRISC	—	—	PORTC Data Direction Control Register						--11 1111	--11 1111
11h	TMR2	Timer2 Module Register								0000 0000	0000 0000
92h	PR2	Timer2 Module Period Register								1111 1111	1111 1111
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
15h	CCPR1L	Capture/Compare/PWM1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used in this mode.

9.1 SPI Mode

The SPI mode allows 8-bits of data to be synchronously transmitted and received simultaneously. To accomplish communication, typically three pins are used:

- Serial Data Out (SDO) RC5/SDO
- Serial Data In (SDI) RC4/SDI
- Serial Clock (SCK) RC3/SCK

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select ($\overline{SS}$) RA5/AN4/$\overline{SS}$

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits in the SSPCON register (SSPCON<5:0>) and SSPSTAT<7:6>. These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Clock Edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

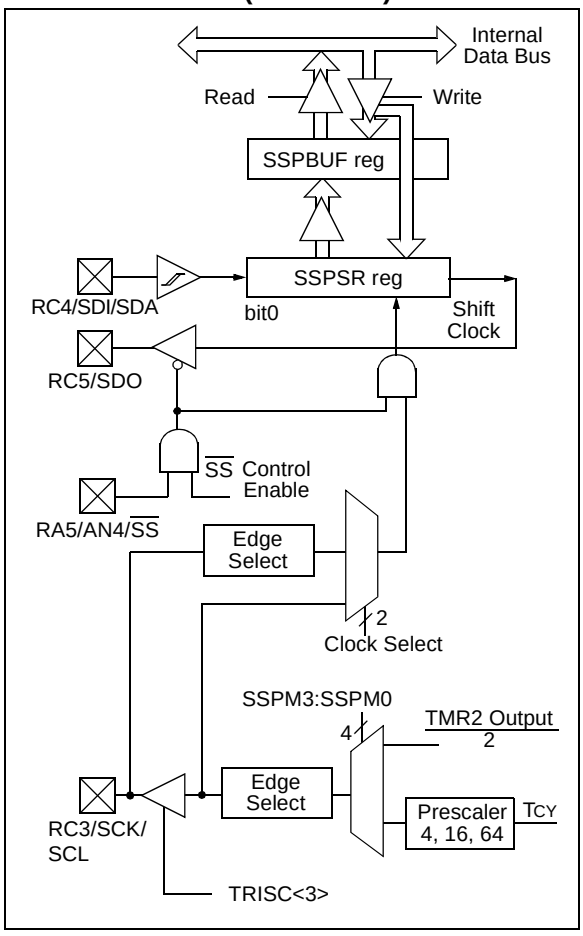
The SSP consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR, until the received data is ready. Once the 8-bits of data have been received, that byte is moved to the SSPBUF register. Then, the buffer full detect bit, BF (SSPSTAT<0>), and interrupt flag bit, SSPIF (PIR1<3>), are set. This double buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored, and the write collision detect bit, WCOL (SSPCON<7>), will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPBUF register completed successfully. When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. Buffer full bit, BF (SSPSTAT<0>), indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, bit BF is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the SSP interrupt is used to determine when the transmission/reception has completed. The SSPBUF must be read and/or written. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur. Example 9-1 shows the loading of the SSPBUF (SSPSR) for data transmission. The `MOVWF RXDATA` instruction (shaded) is only required if the received data is meaningful.

EXAMPLE 9-1: LOADING THE SSPBUF (SSPSR) REGISTER

```
        BCF   STATUS, RP1  ;Select Bank1
        BSF   STATUS, RP0  ;
LOOP    BTFSS SSPSTAT, BF  ;Has data been
                           ;received
                           ;(transmit
                           ;complete)?
        GOTO  LOOP         ;No
        BCF   STATUS, RP0  ;Select Bank0
        MOVF  SSPBUF, W    ;W reg = contents
                           ;of SSPBUF
        MOVWF RXDATA       ;Save in user RAM
        MOVF  TXDATA, W    ;W reg = contents
                           ; of TXDATA
        MOVWF SSPBUF       ;New data to xmit
```

The block diagram of the SSP module, when in SPI mode (Figure 9-1), shows that the SSPSR is not directly readable or writable, and can only be accessed from addressing the SSPBUF register. Additionally, the SSP status register (SSPSTAT) indicates the various status conditions.

FIGURE 9-1: SSP BLOCK DIAGRAM (SPI MODE)



NOTES:

11.1.2 MULTIPLEX TIMING GENERATION

The timing generation circuitry will generate one to four common clocks based on the display mode selected. The mode is specified by bits LMUX1:LMUX0 (LCDCON<1:0>). Table 11-1 shows the formulas for calculating the frame frequency.

TABLE 11-1: FRAME FREQUENCY FORMULAS

Multiplex	Frame Frequency =
Static	Clock source/(128 * (LP3:LP0 + 1))
1/2	Clock source/(128 * (LP3:LP0 + 1))
1/3	Clock source/(96 * (LP3:LP0 + 1))
1/4	Clock source/(128 * (LP3:LP0 + 1))

TABLE 11-2: APPROXIMATE FRAME FREQUENCY (IN Hz) USING TIMER1 @ 32.768 kHz OR FOSC @ 8 MHz

LP3:LP0	Static	1/2	1/3	1/4
2	85	85	114	85
3	64	64	85	64
4	51	51	68	51
5	43	43	57	43
6	37	37	49	37
7	32	32	43	32

TABLE 11-3: APPROXIMATE FRAME FREQUENCY (IN Hz) USING INTERNAL RC OSC @ 14 kHz

LP3:LP0	Static	1/2	1/3	1/4
0	109	109	146	109
1	55	55	73	55
2	36	36	49	36
3	27	27	36	27

TABLE 12-4: STATUS BITS AND THEIR SIGNIFICANCE

$\overline{POR}$	$\overline{BOR}$	$\overline{TO}$	$\overline{PD}$	Condition
0	x	1	1	Power-on Reset
0	x	0	x	Illegal, $\overline{TO}$ is set on $\overline{POR}$
0	x	x	0	Illegal, $\overline{PD}$ is set on $\overline{POR}$
1	0	1	1	Brown-out Reset
1	1	0	1	WDT Reset
1	1	0	0	WDT Wake-up
1	1	u	u	$\overline{MCLR}$ Reset during normal operation
1	1	1	0	$\overline{MCLR}$ Reset during SLEEP or interrupt wake-up from SLEEP

TABLE 12-5: RESET CONDITION FOR SPECIAL REGISTERS

Condition	Program Counter	STATUS Register	PCON Register
Power-on Reset	000h	0001 1xxx	---- --0x
$\overline{MCLR}$ Reset during normal operation	000h	000u uuuu	---- --uu
$\overline{MCLR}$ Reset during SLEEP	000h	0001 0uuu	---- --uu
WDT Reset	000h	0000 1uuu	---- --uu
WDT Wake-up	PC + 1	uuu0 0uuu	---- --uu
Brown-out Reset	000h	0001 1uuu	---- --u0
Interrupt wake-up from SLEEP	PC + 1(1)	uuu1 0uuu	---- --uu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0'.

Note 1: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

RRF **Rotate Right f through Carry**

Syntax: [*label*] RRF f [,d]

Operands: $0 \le f \le 127$
$d \in [0,1]$

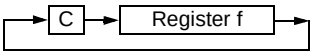
Operation: See description below

Status Affected: C

Encoding:

00	1100	dfff	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 the W register. If 'd' is 1, the result is placed back in register 'f'.



Words: 1

Cycles: 1

Q Cycle Activity:

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

Example
```
RRF             REG1,0
```
Before Instruction:
```
REG1  =  1110 0110
C     =  0
```
After Instruction:
```
REG1  =  1110 0110
W     =  0111 0011
C     =  0
```

SLEEP

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT prescaler,
1 → $\overline{TO}$,
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

00	0000	0110	0011

Description: The power-down status bit, $\overline{PD}$ is cleared. Time-out status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared.
The processor is put into SLEEP mode with the oscillator stopped. See Section 12.8 for more details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No Operation	No Operation	Go to Sleep

Example:
```
SLEEP
```


SUBLW **Subtract W from Literal**

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: k - (W) → (W)

Status Affected: C, DC, Z

Encoding:

11	110x	kkkk	kkkk

Description: The W register is subtracted (2's complement method) from the eight-bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W

Example 1: `SUBLW 0x02`

Before Instruction:
W = 1
C = ?
Z = ?

After Instruction:
W = 1
C = 1; result is positive
Z = 0

Example 2:

Before Instruction:
W = 2
C = ?
Z = ?

After Instruction:
W = 0
C = 1; result is zero
Z = 1

Example 3:

Before Instruction:
W = 3
C = ?
Z = ?

After Instruction:
W = 0xFF
C = 0; result is negative
Z = 0

SUBWF **Subtract W from f**

Syntax: [*label*] SUBWF f [,d]

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - (W) → (destination)

Status Affected: C, DC, Z

Encoding:

00	0010	dfff	ffff

Description: Subtract (2's complement method) W register from register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1: `SUBWF REG1,1`

Before Instruction:
REG1 = 3
W = 2
C = ?
Z = ?

After Instruction:
REG1 = 1
W = 2
C = 1; result is positive
Z = 0

Example 2:

Before Instruction:
REG1 = 2
W = 2
C = ?
Z = ?

After Instruction:
REG1 = 0
W = 2
C = 1; result is zero
Z = 1

Example 3:

Before Instruction:
REG1 = 1
W = 2
C = ?
Z = ?

After Instruction:
REG1 = 0xFF
W = 2
C = 0; result is negative
Z = 0

14.0 DEVELOPMENT SUPPORT

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

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C17 and MPLAB C18 C Compilers
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - ICEPIC™ In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD for PIC16F87X
- Device Programmers
  - PRO MATE® II Universal Device Programmer
  - PICSTART® Plus Entry-Level Development Programmer
- Low Cost Demonstration Boards
  - PICDEM™ 1 Demonstration Board
  - PICDEM 2 Demonstration Board
  - PICDEM 3 Demonstration Board
  - PICDEM 17 Demonstration Board
  - KEELOQ® Demonstration Board

14.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8-bit microcontroller market. The MPLAB IDE is a Windows®-based application that contains:

- An interface to debugging tools
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
  - in-circuit debugger (sold separately)
- A full-featured editor
- A project manager
- Customizable toolbar and key mapping
- A status bar
- On-line help

The MPLAB IDE allows you to:

- Edit your source files (either assembly or ‘C’)
- One touch assemble (or compile) and download to PIC MCU emulator and simulator tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
  - machine code

The ability to use MPLAB IDE with multiple debugging tools allows users to easily switch from the cost-effective simulator to a full-featured emulator with minimal retraining.

14.2 MPASM Assembler

The MPASM assembler is a full-featured universal macro assembler for all PIC MCUs.

The MPASM assembler has a command line interface and a Windows shell. It can be used as a stand-alone application on a Windows 3.x or greater system, or it can be used through MPLAB IDE. 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, an absolute LST file that contains source lines and generated machine code, and a COD file for debugging.

The MPASM assembler features include:

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

14.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI ‘C’ compilers for Microchip’s PIC17CXXX and PIC18CXXX family of microcontrollers, respectively. These compilers provide powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compilers provide symbol information that is compatible with the MPLAB IDE memory display.

15.1 DC Characteristics (Continued)

PIC16LC925/926 (Commercial, Industrial)			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial 0°C ≤ TA ≤ +70°C for commercial				
PIC16C925/926 (Commercial, Industrial)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial 0°C ≤ TA ≤ +70°C for commercial				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	IPD	**Power-down Current (Note 3)**					
D020		PIC16LC925/926	—	0.9	5	µA	VDD = 3.0V
D020		PIC16C925/926	—	1.5	21	µA	VDD = 4.0V
		Module Differential Current (Note 5)					
D021	ΔIWDT	**Watchdog Timer** PIC16LC925/926	—	6.0	20	µA	VDD = 3.0V
D021		**Watchdog Timer** PIC16C925/926	—	9.0	25	µA	VDD = 4.0V
D022	ΔILCDT1	**LCD Voltage Generation** with internal RC osc enabled PIC16LC925/926	—	36	50	µA	VDD = 3.0V **(Note 7)**
D022		**LCD Voltage Generation** with internal RC osc enabled PIC16C925/926	—	40	55	µA	VDD = 4.0V **(Note 7)**
D022A	ΔIBOR	**Brown-out Reset**	—	100	150	µA	BODEN bit set, VDD = 5.0
D024	ΔILCDT1	**LCD Voltage Generation** with Timer1 @ 32.768 kHz PIC16LC925/926	—	15	29	µA	VDD = 3.0V **(Note 7)**
D024		**LCD Voltage Generation** with Timer1 @ 32.768 kHz PIC16C925/926	—	33	60	µA	VDD = 4.0V **(Note 7)**

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail to rail;
all I/O pins tri-stated, pulled to VDD
MCLR = VDD.

3: The power-down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD and VSS.

4: For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kOhm.

5: The Δ current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD or IPD measurement.

6: PWRT must be enabled for slow ramps.

7: ΔILCDT1 and ΔILCDRC includes the current consumed by the LCD Module and the voltage generation circuitry. This does not include current dissipated by the LCD panel.

15.1 DC Characteristics (Continued)

PIC16LC925/926 (Commercial, Industrial)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial 0°C ≤ TA ≤ +70°C for commercial				
PIC16C925/926 (Commercial, Industrial)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial 0°C ≤ TA ≤ +70°C for commercial				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D025	ΔIT1OSC	**Timer1 Oscillator** PIC16LC925/926	—	—	50	μA	VDD = 3.0V
D025		**Timer1 Oscillator** PIC16C925/926	—	—	50	μA	VDD = 4.0V
D026	ΔIAD	**A/D Converter** PIC16LC925/926	—	1.0	—	μA	A/D on, not converting
D026		**A/D Converter** PIC16C925/926	—	1.0	—	μA	A/D on, not converting

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail to rail;
all I/O pins tri-stated, pulled to VDD
MCLR = VDD.

3: The power-down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD and VSS.

4: For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula IR = VDD/2REXT (mA) with REXT in kOhm.

5: The Δ current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD or IPD measurement.

6: PWRT must be enabled for slow ramps.

7: ΔILCDT1 and ΔILCDRC includes the current consumed by the LCD Module and the voltage generation circuitry. This does not include current dissipated by the LCD panel.

NOTES:

THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://microchip.com/support