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	40MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LCD, LVD, POR, PWM, WDT
Number of I/O	67
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (12x12)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f85j90t-i-pt

TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets WDT Reset RESET Instruction Stack Resets CM Resets	Wake-up via WDT or Interrupt
ADRESH	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	PIC18F6XJ90	PIC18F8XJ90	0-00 0000	0-00 0000	u-uu uuuu
ADCON1	PIC18F6XJ90	PIC18F8XJ90	--00 0000	--00 0000	--uu uuuu
ADCON2	PIC18F6XJ90	PIC18F8XJ90	0-00 0000	0-00 0000	u-uu uuuu
LCDDATA4	PIC18F6XJ90	PIC18F8XJ90	---- ---x	---- ---u	---- ---u
LCDDATA4	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA3	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA2	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA1	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA0	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDSE5	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE4	PIC18F6XJ90	PIC18F8XJ90	---- ---0	---- ---u	---- ---u
LCDSE4	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE3	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE2	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE1	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
CVRCON	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
CMCON	PIC18F6XJ90	PIC18F8XJ90	0000 0111	0000 0111	uuuu uuuu
TMR3H	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
T3CON	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
SPBRG1	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
RCREG1	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
TXREG1	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
TXSTA1	PIC18F6XJ90	PIC18F8XJ90	0000 0010	0000 0010	uuuu uuuu
RCSTA1	PIC18F6XJ90	PIC18F8XJ90	0000 000x	0000 000x	uuuu uuuu
LCDPS	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
LCDSE0	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
LCDCON	PIC18F6XJ90	PIC18F8XJ90	000- 0000	000- 0000	uuu- uuuu
EECON2	PIC18F6XJ90	PIC18F8XJ90	---- ----	---- ----	---- ----
EECON1	PIC18F6XJ90	PIC18F8XJ90	---0 x00-	---0 u00-	---0 u00-

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

4: See Table 5-1 for Reset value for specific condition.

5: Bits 6 and 7 of PORTA, LATA and TRISA are enabled depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read as '0'.

6.3 Data Memory Organization

Note:	The operation of some aspects of data memory are changed when the PIC18 extended instruction set is enabled. See **Section 6.6 "Data Memory and the Extended Instruction Set"** for more information.

The data memory in PIC18 devices is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each. The PIC18FX3J90/X4J90 devices, with up to 16 Kbytes of program memory, implement 4 complete banks for a total of 1024 bytes. PIC18FX5J90 devices, with 32 Kbytes of program memory, implement 8 complete banks for a total of 2048 bytes. Figure 6-6 and Figure 6-7 show the data memory organization for the devices.

The data memory contains Special Function Registers (SFRs) and General Purpose Registers (GPRs). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratchpad operations in the user's application. Any read of an unimplemented location will read as '0's.

The instruction set and architecture allow operations across all banks. The entire data memory may be accessed by Direct, Indirect or Indexed Addressing modes. Addressing modes are discussed later in this section.

To ensure that commonly used registers (select SFRs and select GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a 256-byte memory space that provides fast access to select SFRs and the lower portion of GPR Bank 0 without using the BSR. **Section 6.3.2 "Access Bank"** provides a detailed description of the Access RAM.

6.3.1 BANK SELECT REGISTER

Large areas of data memory require an efficient addressing scheme to make rapid access to any address possible. Ideally, this means that an entire address does not need to be provided for each read or write operation. For PIC18 devices, this is accomplished with a RAM banking scheme. This divides the memory space into 16 contiguous banks of 256 bytes. Depending on the instruction, each location can be addressed directly by its full 12-bit address, or an 8-bit low-order address and a 4-bit Bank Pointer.

Most instructions in the PIC18 instruction set make use of the Bank Pointer, known as the Bank Select Register (BSR). This SFR holds the 4 Most Significant bits (MSbs) of a location's address; the instruction itself includes the 8 Least Significant bits (LSbs). Only the four lower bits of the BSR are implemented (BSR<3:0>). The upper four bits are unused; they will always read '0' and cannot be written to. The BSR can be loaded directly by using the MOVLB instruction.

The value of the BSR indicates the bank in data memory. The 8 bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is shown in Figure 6-8.

Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h while the BSR is 0Fh, will end up resetting the program counter.

While any bank can be selected, only those banks that are actually implemented can be read or written to. Writes to unimplemented banks are ignored, while reads from unimplemented banks will return '0's. Even so, the STATUS register will still be affected as if the operation was successful. The data memory map in Figure 6-6 indicates which banks are implemented.

In the core PIC18 instruction set, only the MOVFF instruction fully specifies the 12-bit address of the source and target registers. This instruction ignores the BSR completely when it executes. All other instructions include only the low-order address as an operand and must use either the BSR or the Access Bank to locate their target registers.

10.9 PORTH, LATH and TRISH Registers

Note:	PORTH is available only on 80-pin devices.

PORTH is an 8-bit wide, bidirectional I/O port. The corresponding Data Direction and Data Latch registers are TRISH and LATH. All pins are digital only and tolerate voltages up to 5.5V.

All pins on PORTH are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

All PORTH pins are multiplexed with LCD segment drives controlled by the LCDSE5 register. I/O port functions are only available when the segments are disabled.

EXAMPLE 10-8: INITIALIZING PORTH

```
CLRF    PORTH       ; Initialize PORTH by
                    ; clearing output
                    ; data latches
CLRF    LATH        ; Alternate method
                    ; to clear output
                    ; data latches
MOVLW   0Fh         ; Configure PORTH as
MOVWF   ADCON1      ; digital I/O
MOVLW   0CFh        ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISH       ; Set RH3:RH0 as inputs
                    ; RH5:RH4 as outputs
                    ; RH7:RH6 as inputs
```

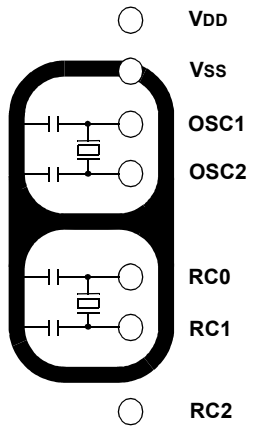
12.3.2 TIMER1 OSCILLATOR LAYOUT CONSIDERATIONS

The Timer1 oscillator circuit draws very little power during operation. Due to the low-power nature of the oscillator, it may also be sensitive to rapidly changing signals in close proximity.

The oscillator circuit, shown in Figure 12-3, should be located as close as possible to the microcontroller. There should be no circuits passing within the oscillator circuit boundaries other than VSS or VDD.

If a high-speed circuit must be located near the oscillator (such as the CCP1 pin in Output Compare or PWM mode, or the primary oscillator using the OSC2 pin), a grounded guard ring around the oscillator circuit, as shown in Figure 12-4, may be helpful when used on a single-sided PCB or in addition to a ground plane.

FIGURE 12-4: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING



Note: Not drawn to scale.

12.4 Timer1 Interrupt

The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The Timer1 interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled or disabled by setting or clearing the Timer1 Interrupt Enable bit, TMR1IE (PIE1<0>).

12.5 Resetting Timer1 Using the CCP Special Event Trigger

If CCP1 or CCP2 is configured to use Timer1 and to generate a Special Event Trigger in Compare mode (CCPxM<3:0> = 1011), this signal will reset Timer3. The trigger from CCP2 will also start an A/D conversion if the A/D module is enabled (see **Section 15.3.4 "Special Event Trigger"** for more information).

The module must be configured as either a timer or a synchronous counter to take advantage of this feature. When used this way, the CCPRxH:CCPRxL register pair effectively becomes a period register for Timer1.

If Timer1 is running in Asynchronous Counter mode, this Reset operation may not work.

In the event that a write to Timer1 coincides with a Special Event Trigger, the write operation will take precedence.

Note:	The Special Event Triggers from the CCPx module will not set the TMR1IF interrupt flag bit (PIR1<0>).

12.6 Using Timer1 as a Real-Time Clock

Adding an external LP oscillator to Timer1 (such as the one described in **Section 12.3 "Timer1 Oscillator"**) gives users the option to include RTC functionality to their applications. This is accomplished with an inexpensive watch crystal to provide an accurate time base and several lines of application code to calculate the time. When operating in Sleep mode and using a battery or supercapacitor as a power source, it can completely eliminate the need for a separate RTC device and battery backup.

The application code routine, RTCisr, shown in Example 12-1, demonstrates a simple method to increment a counter at one-second intervals using an Interrupt Service Routine. Incrementing the TMR1 register pair to overflow triggers the interrupt and calls the routine which increments the seconds counter by one. Additional counters for minutes and hours are incremented as the previous counter overflows.

Since the register pair is 16 bits wide, counting up to overflow the register directly from a 32.768 kHz clock would take 2 seconds. To force the overflow at the required one-second intervals, it is necessary to preload it. The simplest method is to set the MSb of TMR1H with a BSF instruction. Note that the TMR1L register is never preloaded or altered; doing so may introduce cumulative error over many cycles.

For this method to be accurate, Timer1 must operate in Asynchronous mode and the Timer1 overflow interrupt must be enabled (PIE1<0> = 1) as shown in the routine, RTCinit. The Timer1 oscillator must also be enabled and running at all times.

16.1.2 LCD DATA REGISTERS

Once the module is initialized for the LCD panel, the individual bits of the LCDDATA23:LCDDATA0 registers are cleared or set to represent a clear or dark pixel, respectively. Specific sets of LCDDATA registers are used with specific segments and common signals. Each bit represents a unique combination of a specific segment connected to a specific common.

Individual LCDDATA bits are named by the convention "SxxCy", with "xx" as the segment number and "y" as the common number. The relationship is summarized in Table 16-2. The prototype LCDDATA register is shown in Register 16-4.

Note:	In 64-pin devices, writing into the registers LCDDATA5, LCDDATA11, LCDDATA17, and LCDDATA23 will not affect the status of any pixels.

REGISTER 16-4: LCDDATAx: LCD DATA REGISTERS

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
S(n + 7)Cy	S(n + 6)Cy	S(n + 5)Cy	S(n + 4)Cy	S(n + 3)Cy	S(n + 2)Cy	S(n + 1)Cy	S(n)Cy
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 **S(n + 7)Cy:S(n)Cy:** Pixel On bits

For LCDDATA0 through LCDDATA5: n = (8x), y = 0

For LCDDATA6 through LCDDATA11: n = (8(x – 6)), y = 1

For LCDDATA12 through LCDDATA17: n = (8(x – 12)), y = 2

For LCDDATA18 through LCDDATA23: n = (8(x – 18)), y = 3

1 = Pixel on (dark)

0 = Pixel off (clear)

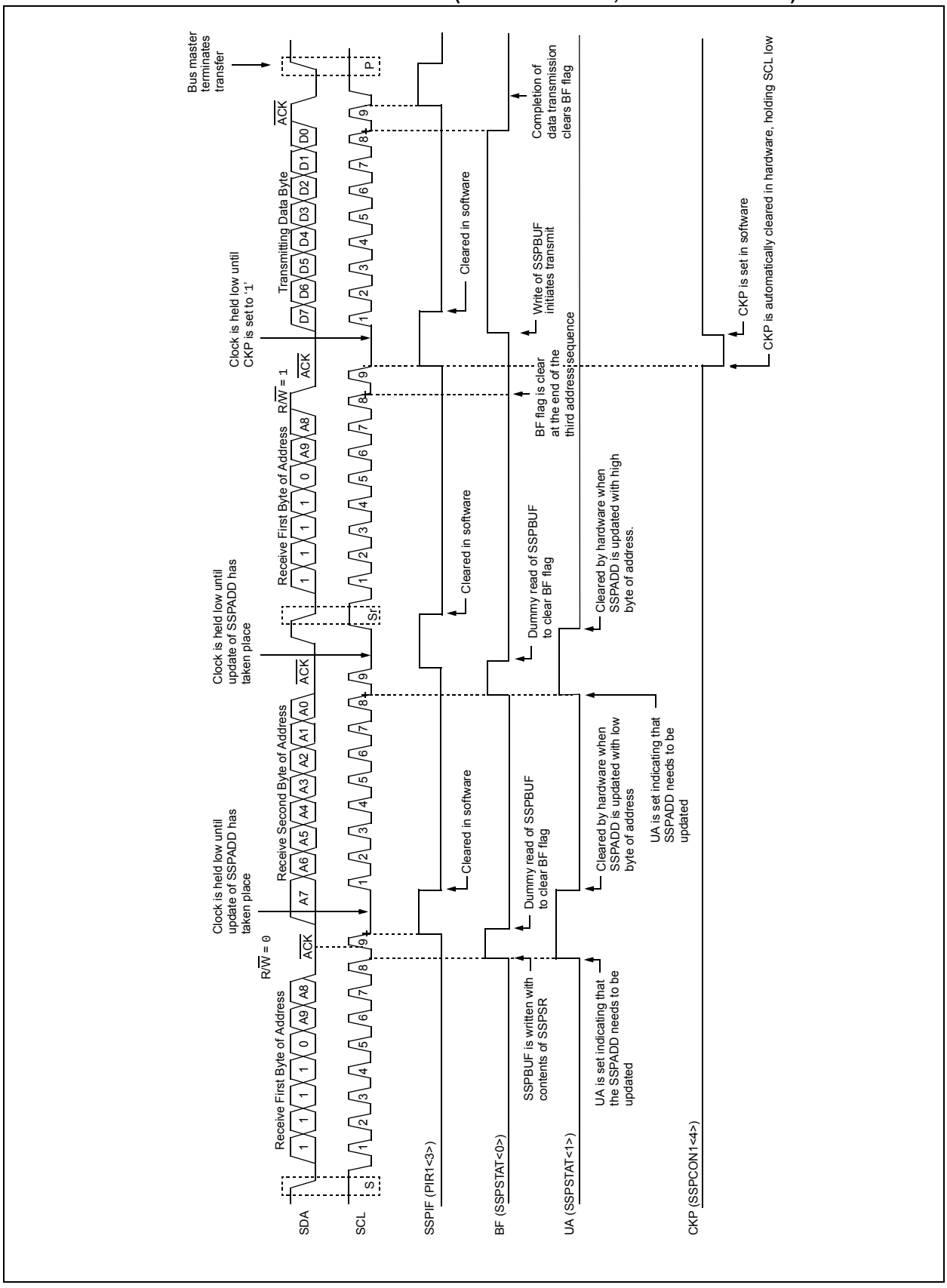
TABLE 16-2: LCDDATA REGISTERS AND BITS FOR SEGMENT AND COM COMBINATIONS

Segments	**COM Lines**			
	0	**1**	**2**	**3**
0 through 7	LCDDATA0	LCDDATA6	LCDDATA12	LCDDATA18
	S00C0:S07C0	S00C1:S07C1	S00C2:S07C2	S00C3:S07C3
8 through 15	LCDDATA1	LCDDATA7	LCDDATA13	LCDDATA19
	S08C0:S15C0	S08C1:S15C1	S08C2:S15C2	S08C0:S15C3
16 through 23	LCDDATA2	LCDDATA8	LCDDATA14	LCDDATA20
	S16C0:S23C0	S16C1:S23C1	S16C2:S23C2	S16C3:S23C3
24 through 31	LCDDATA3	LCDDATA9	LCDDATA15	LCDDATA21
	S24C0:S31C0	S24C1:S31C1	S24C2:S31C2	S24C3:S31C3
32 through 39	LCDDATA4(1)	LCDDATA10(1)	LCDDATA16(1)	LCDDATA22(1)
	S32C0:S39C0	S32C1:S39C1	S32C2:S39C2	S32C3:S39C3
40 through 47	LCDDATA5(2)	LCDDATA11(2)	LCDDATA17(2)	LCDDATA23(2)
	S40C0:S47C0	S40C1:S47C1	S40C2:S47C2	S40C3:S47C3

Note 1: Bits<7:1> of these registers are not implemented in 64-pin devices. Bit 0 of these registers (SEG32Cy) is always implemented.

2: These registers are not implemented on 64-pin devices.

FIGURE 17-13: I²C™ SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)



17.4.10 I²C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address is accomplished by simply writing a value to the SSPBUF register. This action will set the Buffer Full bit, BF, and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted (see data hold time specification parameter 106). SCL is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCL is released high (see data setup time specification parameter 107). When the SCL pin is released high, it is held that way for TBRG. The data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDA. This allows the slave device being addressed to respond with an $\overline{ACK}$ bit during the ninth bit time if an address match occurred, or if data was received properly. The status of $\overline{ACK}$ is written into the ACKDT bit on the falling edge of the ninth clock. If the master receives an Acknowledge, the Acknowledge Status bit, ACKSTAT, is cleared; if not, the bit is set. After the ninth clock, the SSPIF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSPBUF, leaving SCL low and SDA unchanged (Figure 17-23).

After the write to the SSPBUF, each bit of the address will be shifted out on the falling edge of SCL until all seven address bits and the R/$\overline{W}$ bit are completed. On the falling edge of the eighth clock, the master will deassert the SDA pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the $\overline{ACK}$ bit is loaded into the ACKSTAT status bit (SSPCON2<6>). Following the falling edge of the ninth clock transmission of the address, the SSPIF is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPBUF takes place, holding SCL low and allowing SDA to float.

17.4.10.1 BF Status Flag

In Transmit mode, the BF bit (SSPSTAT<0>) is set when the CPU writes to SSPBUF and is cleared when all 8 bits are shifted out.

17.4.10.2 WCOL Status Flag

If the user writes to the SSPBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur) after 2 TCY after the SSPBUF write. If SSPBUF is rewritten within 2 TCY, the WCOL bit is set and SSPBUF is updated. This may result in a corrupted transfer.

The user should verify that the WCOL is clear after each write to SSPBUF to ensure the transfer is correct. In all cases, WCOL must be cleared in software.

17.4.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit (SSPCON2<6>) is cleared when the slave has sent an Acknowledge ($\overline{ACK}$ = 0) and is set when the slave does not Acknowledge ($\overline{ACK}$ = 1). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

17.4.11 I²C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN (SSPCON2<3>).

Note:	The MSSP module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable bit, ACKEN (SSPCON2<4>).

17.4.11.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPBUF from SSPSR. It is cleared when the SSPBUF register is read.

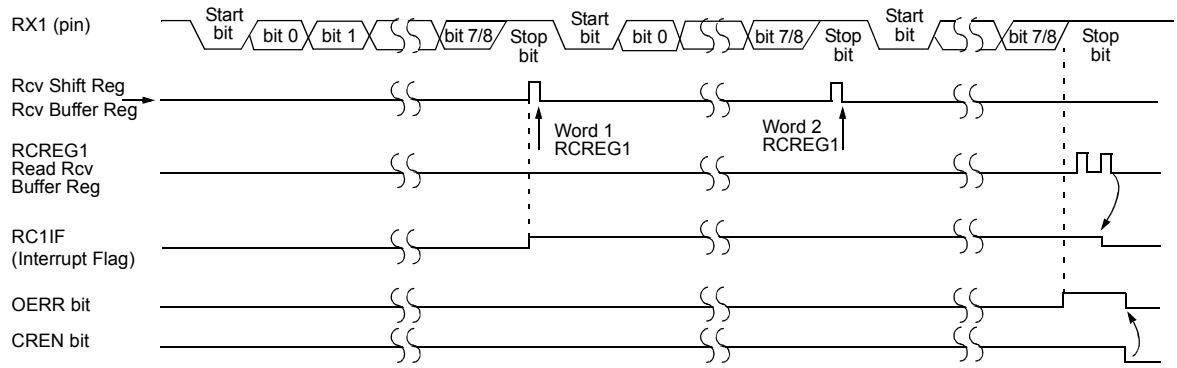
17.4.11.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when 8 bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

17.4.11.3 WCOL Status Flag

If the user writes the SSPBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

FIGURE 18-7: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX1 input. The RCREG1 (Receive Buffer register) is read after the third word causing the OERR (Overrun) bit to be set.

TABLE 18-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	57
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	60
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	60
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	60
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	59
RCREG1	EUSART Receive Register								59
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	59
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	61
SPBRGH1	EUSART Baud Rate Generator Register High Byte								61
SPBRG1	EUSART Baud Rate Generator Register Low Byte								59

Legend: — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.

18.5 EUSART Synchronous Slave Mode

Synchronous Slave mode is entered by clearing bit CSRC (TXSTA<7>). This mode differs from the Synchronous Master mode in that the shift clock is supplied externally at the CK1 pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in any low-power mode.

18.5.1 EUSART SYNCHRONOUS SLAVE TRANSMIT

The operation of the Synchronous Master and Slave modes are identical except in the case of the Sleep mode.

If two words are written to the TXREG1 and then the SLEEP instruction is executed, the following will occur:

a) The first word will immediately transfer to the TSR register and transmit.
b) The second word will remain in the TXREG1 register.
c) Flag bit, TX1IF, will not be set.
d) When the first word has been shifted out of TSR, the TXREG1 register will transfer the second word to the TSR and flag bit, TX1IF, will now be set.
e) If enable bit, TX1IE, is set, the interrupt will wake the chip from Sleep. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Transmission:

1. Enable the synchronous slave serial port by setting bits, SYNC and SPEN, and clearing bit, CSRC.
2. Clear bits, CREN and SREN.
3. If interrupts are desired, set enable bit, TX1IE.
4. If 9-bit transmission is desired, set bit, TX9.
5. Enable the transmission by setting enable bit, TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. Start transmission by loading data to the TXREG1 register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

TABLE 18-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	57
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	60
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	60
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	60
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	59
TXREG1	EUSART Transmit Register								59
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	59
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	61
SPBRGH1	EUSART Baud Rate Generator Register High Byte								61
SPBRG1	EUSART Baud Rate Generator Register Low Byte								59
LATG	U2OD	U1OD	—	LATG4	LATG3	LATG2	LATG1	LATG0	60

Legend: — = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.

19.2 AUSART Baud Rate Generator (BRG)

The BRG is a dedicated, 8-bit generator that supports both the Asynchronous and Synchronous modes of the AUSART.

The SPBRG2 register controls the period of a free-running timer. In Asynchronous mode, bit BRGH (TXSTA<2>) also controls the baud rate. In Synchronous mode, BRGH is ignored. Table 19-1 shows the formula for computation of the baud rate for different AUSART modes, which only apply in Master mode (internally generated clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRG2 register can be calculated using the formulas in Table 19-1. From this, the error in baud rate can be determined. An example calculation is shown in Example 19-1. Typical baud rates and error values for the various Asynchronous modes are shown in Table 19-2. It may be advantageous to use the high baud rate (BRGH = 1) to reduce the baud rate error, or achieve a slow baud rate for a fast oscillator frequency.

Writing a new value to the SPBRG2 register causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

19.2.1 OPERATION IN POWER-MANAGED MODES

The device clock is used to generate the desired baud rate. When one of the power-managed modes is entered, the new clock source may be operating at a different frequency. This may require an adjustment to the value in the SPBRG2 register.

19.2.2 SAMPLING

The data on the RX2 pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX2 pin.

TABLE 19-1: BAUD RATE FORMULAS

Configuration Bits		BRG/AUSART Mode	Baud Rate Formula
SYNC	BRGH		
0	0	Asynchronous	FOSC/[64 (n + 1)]
0	1	Asynchronous	FOSC/[16 (n + 1)]
1	x	Synchronous	FOSC/[4 (n + 1)]

Legend: x = Don't care, n = Value of SPBRG2 register

EXAMPLE 19-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, BRGH = 0:

Desired Baud Rate = FOSC/(64 ([SPBRG2] + 1))

Solving for SPBRG2:

X = ((FOSC/Desired Baud Rate)/64) – 1
= ((16000000/9600)/64) – 1
= [25.042] = 25

Calculated Baud Rate = 16000000/(64 (25 + 1))
= 9615

Error = (Calculated Baud Rate – Desired Baud Rate)/Desired Baud Rate
= (9615 – 9600)/9600 = 0.16%

TABLE 19-2: REGISTERS ASSOCIATED WITH THE BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	62
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	62
SPBRG2	AUSART Baud Rate Generator Register								62

Legend: Shaded cells are not used by the BRG.

19.5.2 AUSART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical except in the case of Sleep or any Idle mode, and bit SREN, which is a “don’t care” in Slave mode.

If receive is enabled by setting the CREN bit prior to entering Sleep, or any Idle mode, then a word may be received while in this low-power mode. Once the word is received, the RSR register will transfer the data to the RCREG2 register; if the RC2IE enable bit is set, the interrupt generated will wake the chip from low-power mode. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting bits, SYNC and SPEN, and clearing bit, CSRC.
2. If interrupts are desired, set enable bit, RC2IE.
3. If 9-bit reception is desired, set bit, RX9.
4. To enable reception, set enable bit, CREN.
5. Flag bit, RC2IF, will be set when reception is complete. An interrupt will be generated if enable bit, RC2IE, was set.
6. Read the RCSTA2 register to get the 9th bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG2 register.
8. If any error occurred, clear the error by clearing bit, CREN.
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

TABLE 19-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	57
PIR3	—	LCDIF	RC2IF	TX2IF	—	CCP2IF	CCP1IF	—	60
PIE3	—	LCDIE	RC2IE	TX2IE	—	CCP2IE	CCP1IE	—	60
IPR3	—	LCDIP	RC2IP	TX2IP	—	CCP2IP	CCP1IP	—	60
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	62
RCREG2	AUSART Receive Register								62
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	62
SPBRG2	AUSART Baud Rate Generator Register								62

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous slave reception.

TABLE 24-2: PIC18F85J90 FAMILY INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb	LSb	Status Affected	Notes
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	n, 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	n	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 the 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 24.2.3 "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

POP **Pop Top of Return Stack**

Syntax: POP

Operands: None

Operation: (TOS) → 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) → 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

SUBLW **Subtract W from Literal**

Syntax: SUBLW k

Operands: $0 \le k \le 255$

Operation: $k - (W) \rightarrow W$

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

Encoding:

0000	1000	kkkk	kkkk

Description: W is subtracted from the eight-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1:
```
SUBLW   02h
Before Instruction
    W   =   01h
    C   =   ?
After Instruction
    W   =   01h
    C   =   1     ; result is positive
    Z   =   0
    N   =   0
```

Example 2:
```
SUBLW   02h
Before Instruction
    W   =   02h
    C   =   ?
After Instruction
    W   =   00h
    C   =   1     ; result is zero
    Z   =   1
    N   =   0
```

Example 3:
```
SUBLW   02h
Before Instruction
    W   =   03h
    C   =   ?
After Instruction
    W   =   FFh   ; (2's complement)
    C   =   0     ; result is negative
    Z   =   0
    N   =   1
```

SUBWF **Subtract W from f**

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

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

Operation: $(f) - (W) \rightarrow dest$

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

Encoding:

0101	11da	ffff	ffff

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

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 24.2.3 "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 1:
```
SUBWF   REG, 1, 0
Before Instruction
    REG   =   3
    W     =   2
    C     =   ?
After Instruction
    REG   =   1
    W     =   2
    C     =   1     ; result is positive
    Z     =   0
    N     =   0
```

Example 2:
```
SUBWF   REG, 0, 0
Before Instruction
    REG   =   2
    W     =   2
    C     =   ?
After Instruction
    REG   =   2
    W     =   0
    C     =   1     ; result is zero
    Z     =   1
    N     =   0
```

Example 3:
```
SUBWF   REG, 1, 0
Before Instruction
    REG   =   1
    W     =   2
    C     =   ?
After Instruction
    REG   =   FFh  ;(2's complement)
    W     =   2
    C     =   0     ; result is negative
    Z     =   0
    N     =   1
```

MOVSS **Move Indexed to Indexed**

Syntax: MOVSS [z_s], [z_d]

Operands: $0 \le z_s \le 127$
$0 \le z_d \le 127$

Operation: ((FSR2) + z_s) → ((FSR2) + z_d)

Status Affected: None

Encoding:

1st word (source)	1110	1011	1zzz	$zzzz_s$
2nd word (dest.)	1111	xxxx	xzzz	$zzzz_d$

Description: The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets, 'z_s' or 'z_d', respectively, to the value of FSR2. Both registers can be located anywhere in the 4096-byte data memory space (000h to FFFh).

The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

If the resultant source address points to an Indirect Addressing register, the value returned will be 00h. If the resultant destination address points to an Indirect Addressing register, the instruction will execute as a NOP.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Determine source addr	Determine source addr	Read source reg
Decode	Determine dest addr	Determine dest addr	Write to dest reg

Example:
```
MOVSS [05h], [06h]
```
Before Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 11h
After Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 33h

PUSHL **Store Literal at FSR2, Decrement FSR2**

Syntax: PUSHL k

Operands: $0 \le k \le 255$

Operation: k → (FSR2),
FSR2 – 1 → FSR2

Status Affected: None

Encoding:

1111	1010	kkkk	kkkk

Description: The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation.

This instruction allows users to push values onto a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example:
```
PUSHL 08h
```
Before Instruction
FSR2H:FSR2L = 01ECh
Memory (01ECh) = 00h

After Instruction
FSR2H:FSR2L = 01EBh
Memory (01ECh) = 08h

25.0 DEVELOPMENT SUPPORT

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

- Integrated Development Environment
  - MPLAB® IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB C Compiler for Various Device Families
  - HI-TECH C for Various Device Families
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers
  - MPLAB ICD 3
  - PICkit™ 3 Debug Express
- Device Programmers
  - PICkit™ 2 Programmer
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits, and Starter Kits

25.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16/32-bit microcontroller market. The MPLAB IDE is a Windows® operating system-based application that contains:

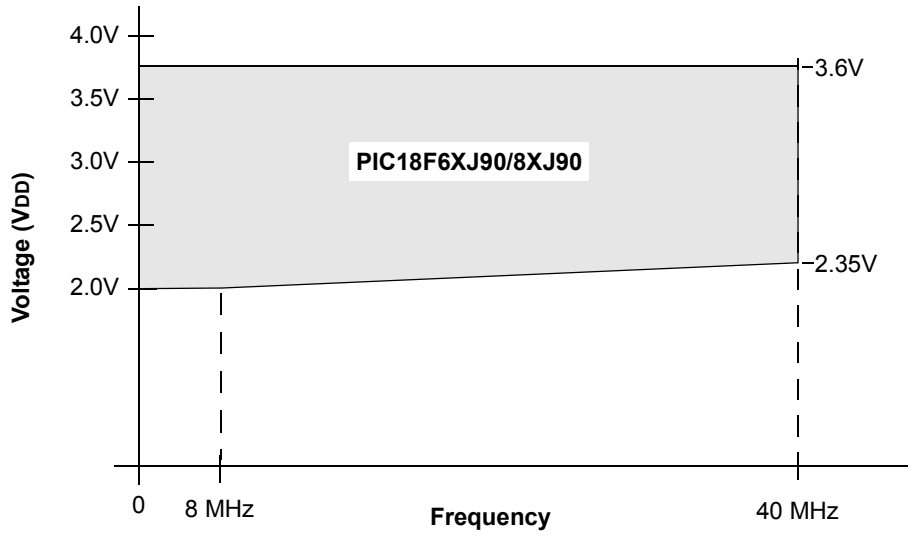
- A single graphical interface to all debugging tools
  - Simulator
  - Programmer (sold separately)
  - In-Circuit Emulator (sold separately)
  - In-Circuit Debugger (sold separately)
- A full-featured editor with color-coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High-level source code debugging
- Mouse over variable inspection
- Drag and drop variables from source to watch windows
- Extensive on-line help
- Integration of select third party tools, such as IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either C or assembly)
- One-touch compile or assemble, and download to emulator and simulator tools (automatically updates all project information)
- Debug using:
  - Source files (C or assembly)
  - Mixed C and assembly
  - Machine code

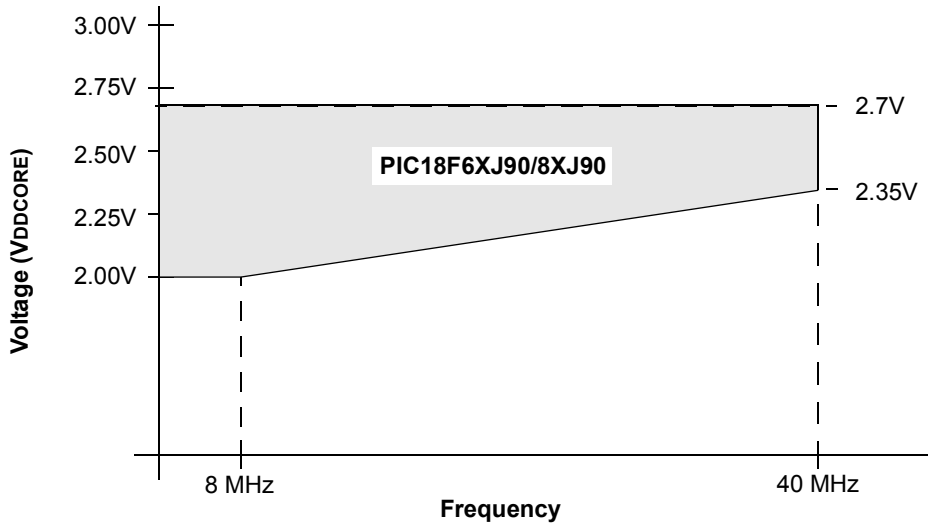
MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost-effective simulators, through low-cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increased flexibility and power.

FIGURE 26-1: PIC18F85J90 FAMILY VOLTAGE-FREQUENCY GRAPH, REGULATOR ENABLED (INDUSTRIAL)(1)



Note 1: When the on-chip regulator is enabled, its BOR circuit will automatically trigger a device Reset before VDD reaches a level at which full-speed operation is not possible.

FIGURE 26-2: PIC18F85J90 FAMILY VOLTAGE-FREQUENCY GRAPH, REGULATOR DISABLED (INDUSTRIAL)(1,2)



Note 1: For frequencies between 4 MHz and 40 MHz, FMAX = (51.42 MHz/V) * (VDDCORE – 2V) + 4 MHz.

2: When the on-chip voltage regulator is disabled, VDD and VDDCORE must be maintained so that VDDCORE ≤ VDD ≤ 3.6V.

26.4 AC (Timing) Characteristics

26.4.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS
2. TppS
3. TCC:ST (I^2C specifications only)
4. Ts (I^2C specifications only)

T			
F	Frequency	T	Time

Lowercase letters (pp) and their meanings:

pp			
cc	CCP1	osc	OSC1
ck	CLKO	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{SS}$
dt	Data in	t0	T0CKI
io	I/O port	t1	T13CKI
mc	$\overline{MCLR}$	wr	$\overline{WR}$

Uppercase letters and their meanings:

S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (High-Impedance)	V	Valid
L	Low	Z	High-Impedance
I^2C only			
AA	output access	High	High
BUF	Bus free	Low	Low

TCC:ST (I^2C specifications only)

CC			
HD	Hold	SU	Setup
ST			
DAT	DATA input hold	STO	Stop condition
STA	Start condition		

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-6578-300
Fax: 886-3-6578-370

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/05/10