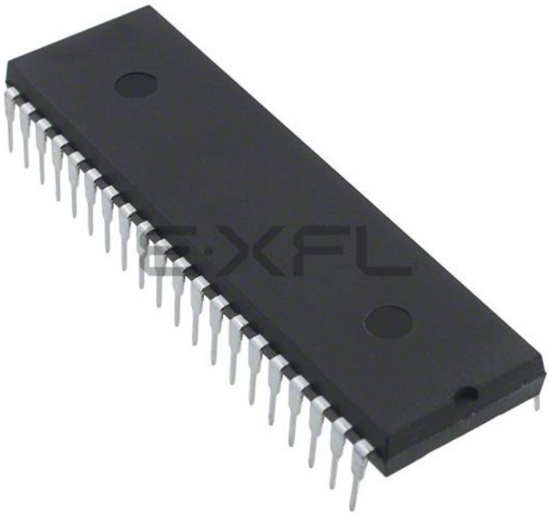


Welcome to E-XFL.COM

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	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	33
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	40-DIP (0.600", 15.24mm)
Supplier Device Package	40-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f448-i-p

FIGURE 3-3: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ TIED TO VDD)

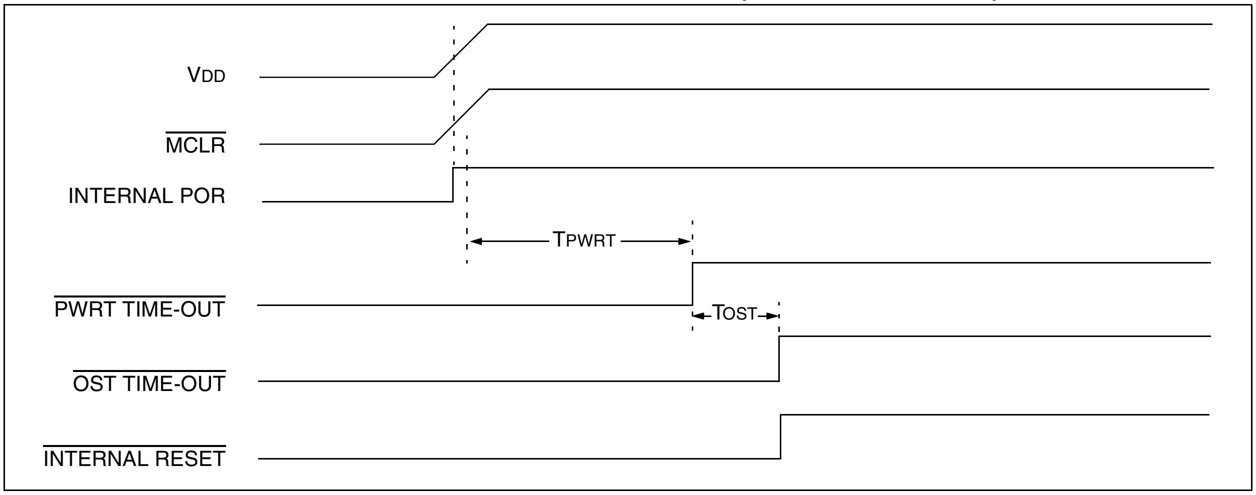


FIGURE 3-4: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO VDD): CASE 1

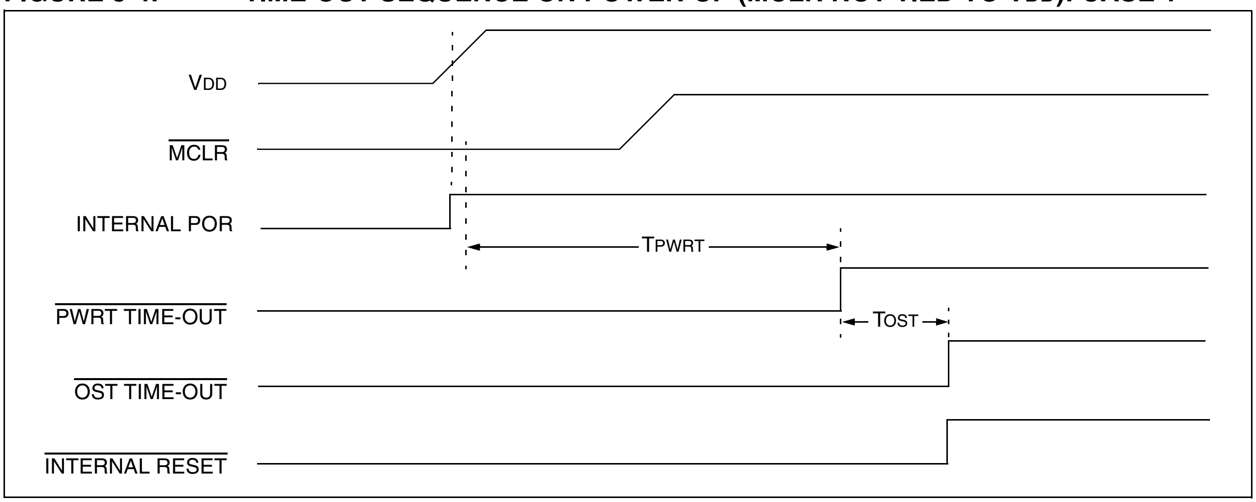


FIGURE 3-5: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO VDD): CASE 2

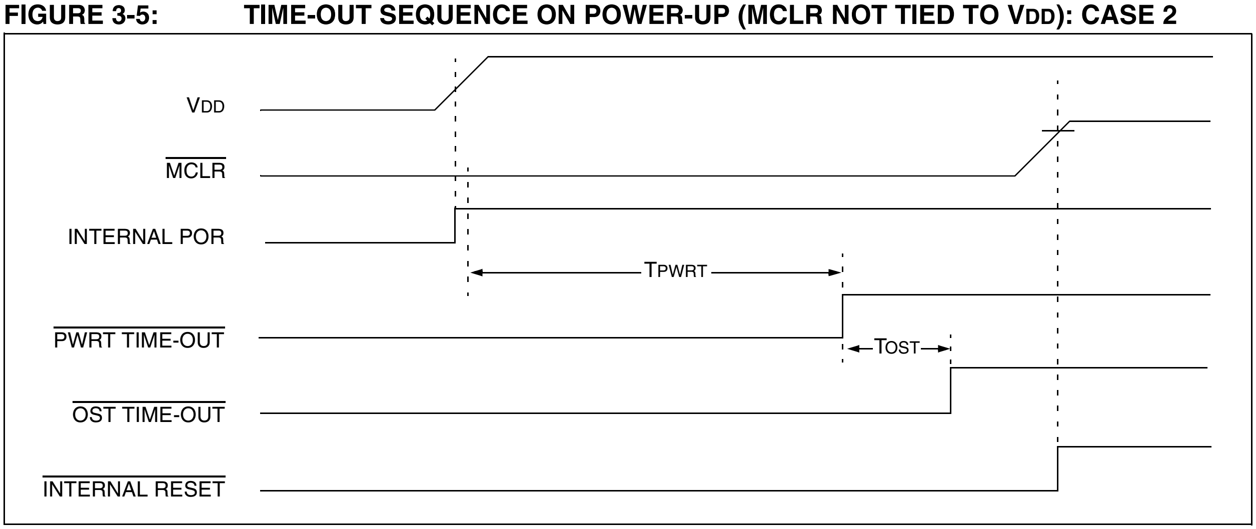


FIGURE 3-6: SLOW RISE TIME ($\overline{MCLR}$ TIED TO V_{DD})

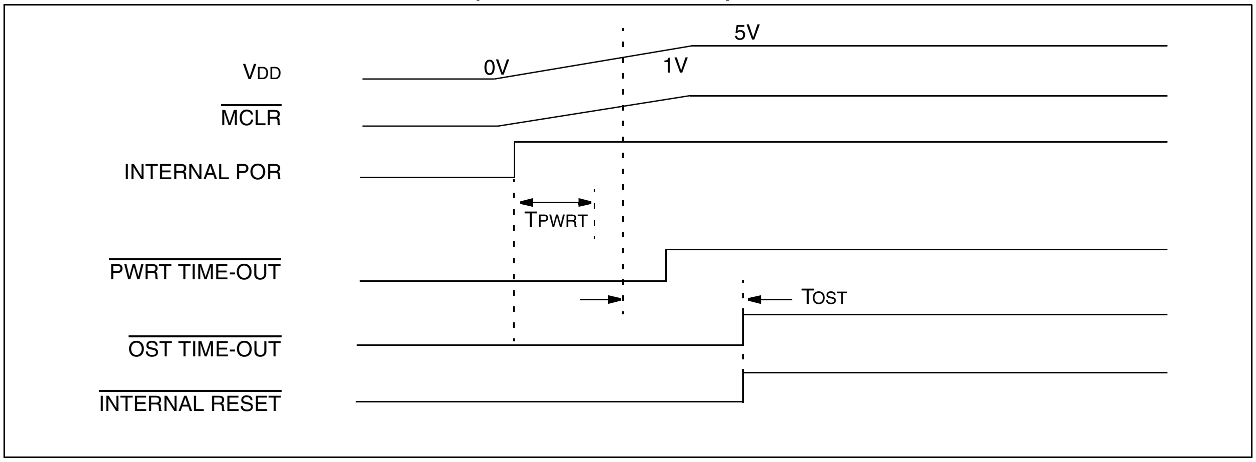
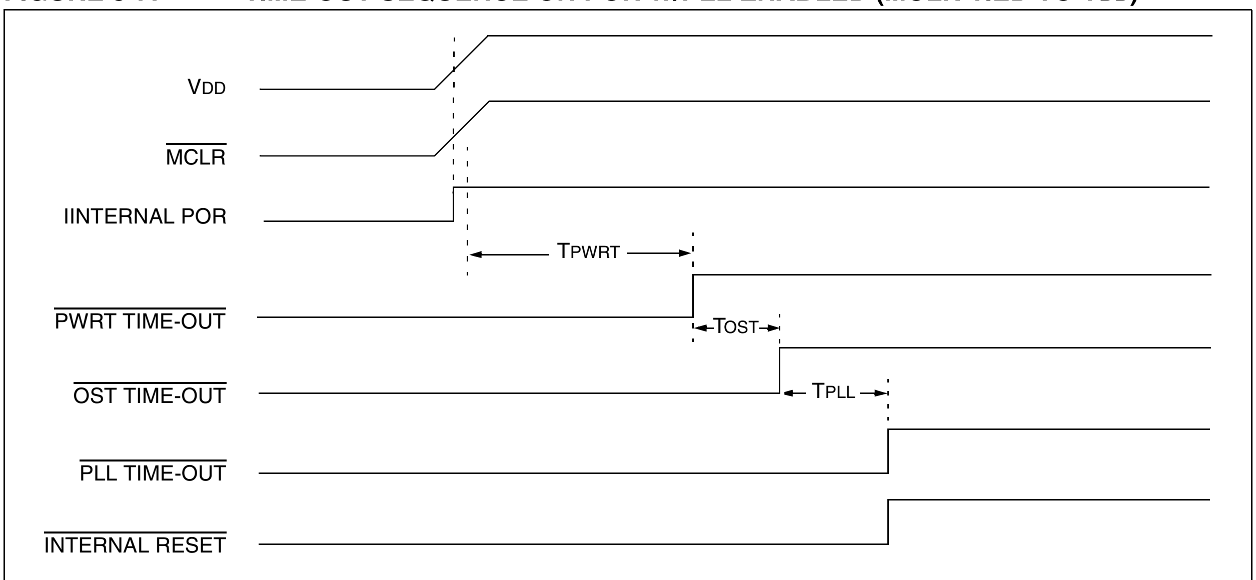


FIGURE 3-7: TIME-OUT SEQUENCE ON POR W/PLL ENABLED ($\overline{MCLR}$ TIED TO V_{DD})



Note: T_{OST} = 1024 clock cycles.
$T_{PLL} \approx$ 2 ms max. First three stages of the PWRT timer.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Reset WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
RXM1SIDL	PIC18F2X8	PIC18F4X8	xxx- --xx	uuu- --uu	uuu- --uu
RXM1SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM0EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM0EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM0SIDL	PIC18F2X8	PIC18F4X8	xxx- --xx	uuu- --uu	uuu- --uu
RXM0SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF5EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF5EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF5SIDL	PIC18F2X8	PIC18F4X8	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF5SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF4EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF4EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF4SIDL	PIC18F2X8	PIC18F4X8	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF4SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF3EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF3EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF3SIDL	PIC18F2X8	PIC18F4X8	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF3SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF2EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF2EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF2SIDL	PIC18F2X8	PIC18F4X8	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF2SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF1EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF1EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF1SIDL	PIC18F2X8	PIC18F4X8	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF1SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF0EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF0EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF0SIDL	PIC18F2X8	PIC18F4X8	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF0SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu

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

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: 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.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).

4.12 Indirect Addressing, INDF and FSR Registers

Indirect addressing is a mode of addressing data memory where the data memory address in the instruction is not fixed. A SFR register is used as a pointer to the data memory location that is to be read or written. Since this pointer is in RAM, the contents can be modified by the program. This can be useful for data tables in the data memory and for software stacks. Figure 4-8 shows the operation of indirect addressing. This shows the moving of the value to the data memory address specified by the value of the FSR register.

Indirect addressing is possible by using one of the INDF registers. Any instruction using the INDF register actually accesses the register indicated by the File Select Register, FSR. Reading the INDF register itself, indirectly (FSR = `0`), will read 00h. Writing to the INDF register indirectly, results in a no operation. The FSR register contains a 12-bit address which is shown in Figure 4-8.

The INDFn ($0 \le n \le 2$) register is not a physical register. Addressing INDFn actually addresses the register whose address is contained in the FSRn register (FSRn is a pointer). This is indirect addressing.

Example 4-5 shows a simple use of indirect addressing to clear the RAM in Bank 1 (locations 100h-1FFh) in a minimum number of instructions.

EXAMPLE 4-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
       LFSR   FSR0, 100h ;
NEXT   CLRF   POSTINC0    ; Clear INDF
                          ; register
                          ; & inc pointer
       BTFSS  FSR0H, 1    ; All done
                          ; w/ Bank1?
       BRA    NEXT        ; NO, clear next
CONTINUE                  ;
   :                      ; YES, continue
```

There are three indirect addressing registers. To address the entire data memory space (4096 bytes), these registers are 12 bits wide. To store the 12 bits of addressing information, two 8-bit registers are required. These indirect addressing registers are:

1. FSR0: composed of FSR0H:FSR0L
2. FSR1: composed of FSR1H:FSR1L
3. FSR2: composed of FSR2H:FSR2L

In addition, there are registers INDF0, INDF1 and INDF2, which are not physically implemented. Reading or writing to these registers activates indirect addressing, with the value in the corresponding FSR register being the address of the data.

If an instruction writes a value to INDF0, the value will be written to the address indicated by FSR0H:FSR0L. A read from INDF1 reads the data from the address indicated by FSR1H:FSR1L. INDFn can be used in code anywhere an operand can be used.

If INDF0, INDF1 or INDF2 are read indirectly via an FSR, all '`0`'s are read (zero bit is set). Similarly, if INDF0, INDF1 or INDF2 are written to indirectly, the operation will be equivalent to a `NOP` instruction and the Status bits are not affected.

4.12.1 INDIRECT ADDRESSING OPERATION

Each FSR register has an INDF register associated with it, plus four additional register addresses. Performing an operation on one of these five registers determines how the FSR will be modified during indirect addressing.

- When data access is done to one of the five INDFn locations, the address selected will configure the FSRn register to:
  - Do nothing to FSRn after an indirect access (no change) – INDFn
  - Auto-decrement FSRn after an indirect access (post-decrement) – POSTDECn
  - Auto-increment FSRn after an indirect access (post-increment) – POSTINCn
  - Auto-increment FSRn before an indirect access (pre-increment) – PREINCn
  - Use the value in the WREG register as an offset to FSRn. Do not modify the value of the WREG or the FSRn register after an indirect access (no change) – PLUSWn

When using the auto-increment or auto-decrement features, the effect on the FSR is not reflected in the Status register. For example, if the indirect address causes the FSR to equal '`0`', the Z bit will not be set.

Incrementing or decrementing an FSR affects all 12 bits. That is, when FSRnL overflows from an increment, FSRnH will be incremented automatically.

Adding these features allows the FSRn to be used as a software stack pointer in addition to its uses for table operations in data memory.

Each FSR has an address associated with it that performs an indexed indirect access. When a data access to this INDFn location (PLUSWn) occurs, the FSRn is configured to add the 2's complement value in the WREG register and the value in FSR to form the address before an indirect access. The FSR value is not changed.

If an FSR register contains a value that indicates one of the INDFn, an indirect read will read 00h (zero bit is set), while an indirect write will be equivalent to a `NOP` (Status bits are not affected).

If an indirect addressing operation is done where the target address is an FSRnH or FSRnL register, the write operation will dominate over the pre- or post-increment/decrement functions.

6.0 FLASH PROGRAM MEMORY

The Flash program memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed on one byte at a time. A write to program memory is executed on blocks of 8 bytes at a time. Program memory is erased in blocks of 64 bytes at a time. A bulk erase operation may not be issued from user code.

Writing or erasing program memory will cease instruction fetches until the operation is complete. The program memory cannot be accessed during the write or erase, therefore, code cannot execute. An internal programming timer terminates program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a `NOP`.

6.1 Table Reads and Table Writes

In order to read and write program memory, there are two operations that allow the processor to move bytes between the program memory space and the data RAM:

- Table Read (`TBLRD`)
- Table Write (`TBLWT`)

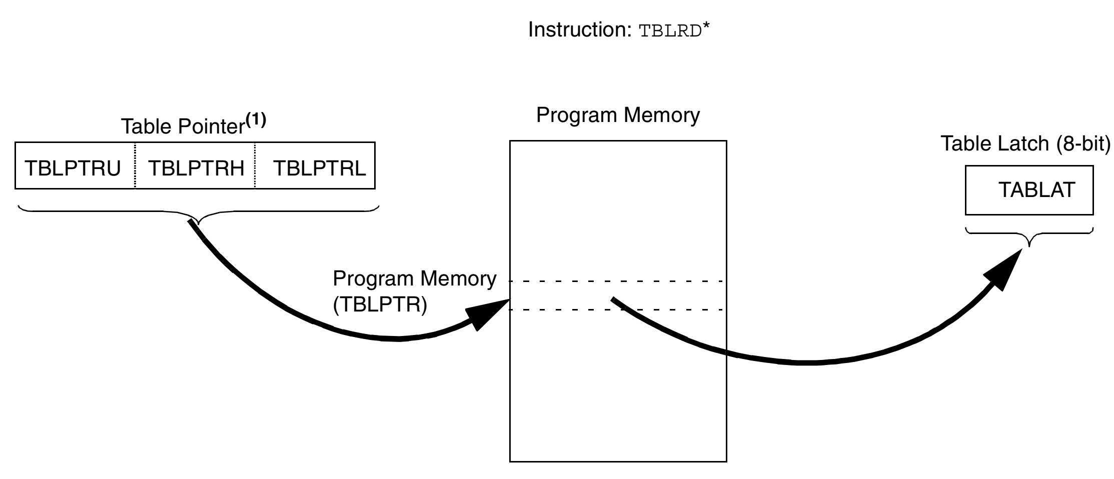
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and place it into the data RAM space. Figure 6-1 shows the operation of a table read with program memory and data RAM.

Table write operations store data from the data memory space into holding registers in program memory. The procedure to write the contents of the holding registers into program memory is detailed in **Section 6.5 "Writing to Flash Program Memory"**. Figure 6-2 shows the operation of a table write with program memory and data RAM.

Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word aligned. Therefore, a table block can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need to be word aligned.

FIGURE 6-1: TABLE READ OPERATION



Note 1: Table Pointer points to a byte in program memory.

REGISTER 8-5: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2

U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	CMIF(1)	—	EEIF	BCLIF	LVDIF	TMR3IF	ECCP1IF(1)
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6 **CMIF:** Comparator Interrupt Flag bit(1)
1 = Comparator input has changed
0 = Comparator input has not changed

bit 5 **Unimplemented:** Read as '0'

bit 4 **EEIF:** EEPROM Write Operation Interrupt Flag bit
1 = Write operation is complete (must be cleared in software)
0 = Write operation is not complete

bit 3 **BCLIF:** Bus Collision Interrupt Flag bit
1 = A bus collision occurred (must be cleared in software)
0 = No bus collision occurred

bit 2 **LVDIF:** Low-Voltage Detect Interrupt Flag bit
1 = A low-voltage condition occurred (must be cleared in software)
0 = The device voltage is above the Low-Voltage Detect trip point

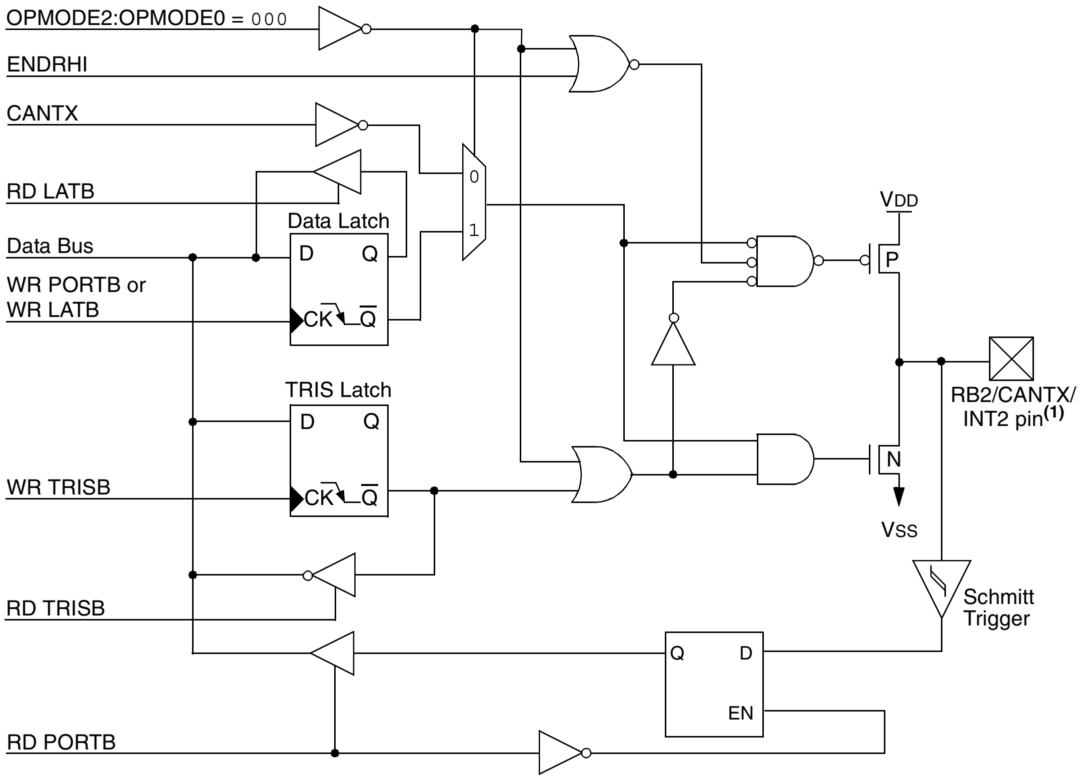
bit 1 **TMR3IF:** TMR3 Overflow Interrupt Flag bit
1 = TMR3 register overflowed (must be cleared in software)
0 = TMR3 register did not overflow

bit 0 **ECCP1IF:** ECCP1 Interrupt Flag bit(1)
Capture mode:
1 = A TMR1 (TMR3) register capture occurred (must be cleared in software)
0 = No TMR1 (TMR3) register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

Note 1: This bit is only available on PIC18F4X8 devices. For PIC18F2X8 devices, this bit is unimplemented and reads as '0'.

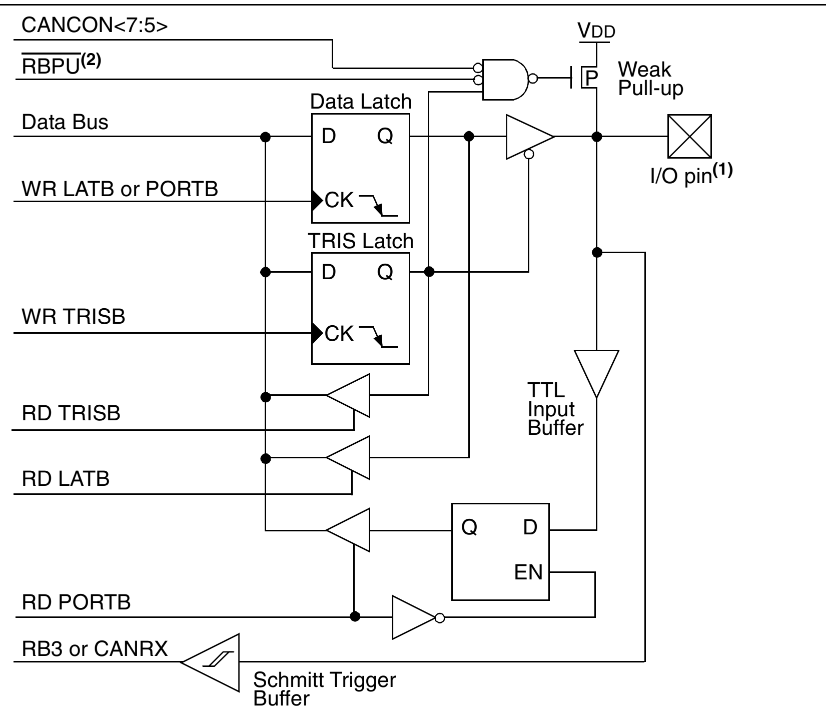
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

FIGURE 9-6: RB2/CANTX/INT2 PIN BLOCK DIAGRAM



Note 1: I/O pin has diode protection to V_{DD} and V_{SS}.

FIGURE 9-7: RB3/CANRX PIN BLOCK DIAGRAM



Note 1: I/O pins have diode protection to V_{DD} and V_{SS}.

2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (INTCON2<7>).

16.0 ENHANCED CAPTURE/ COMPARE/PWM (ECCP) MODULE

Note:	The ECCP (Enhanced Capture/Compare/ PWM) module is only available on PIC18F448 and PIC18F458 devices.

This module contains a 16-bit register which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register.

The operation of the ECCP module differs from the CCP (discussed in detail in **Section 15.0 "Capture/ Compare/PWM (CCP) Modules"**) with the addition of an Enhanced PWM module which allows for up to 4 output channels and user selectable polarity. These features are discussed in detail in **Section 16.5 "Enhanced PWM Mode"**. The module can also be programmed for automatic shutdown in response to various analog or digital events.

The control register for ECCP1 is shown in Register 16-1.

REGISTER 16-1: ECCP1CON: ECCP1 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EPWM1M1	EPWM1M0	EDC1B1	EDC1B0	ECCP1M3	ECCP1M2	ECCP1M1	ECCP1M0
bit 7							bit 0

bit 7-6 **EPWM1M<1:0>:** PWM Output Configuration bits

If ECCP1M<3:2> = `00, 01, 10`:

`xx` = P1A assigned as Capture/Compare input; P1B, P1C, P1D assigned as port pins

If ECCP1M<3:2> = `11`:

`00` = Single output; P1A modulated; P1B, P1C, P1D assigned as port pins
`01` = Full-bridge output forward; P1D modulated; P1A active; P1B, P1C inactive
`10` = Half-bridge output; P1A, P1B modulated with deadband control; P1C, P1D assigned as port pins
`11` = Full-bridge output reverse; P1B modulated; P1C active; P1A, P1D inactive

bit 5-4 **EDC1B<1:0>:** PWM Duty Cycle Least Significant bits

Capture mode:
Unused.

Compare mode:
Unused.

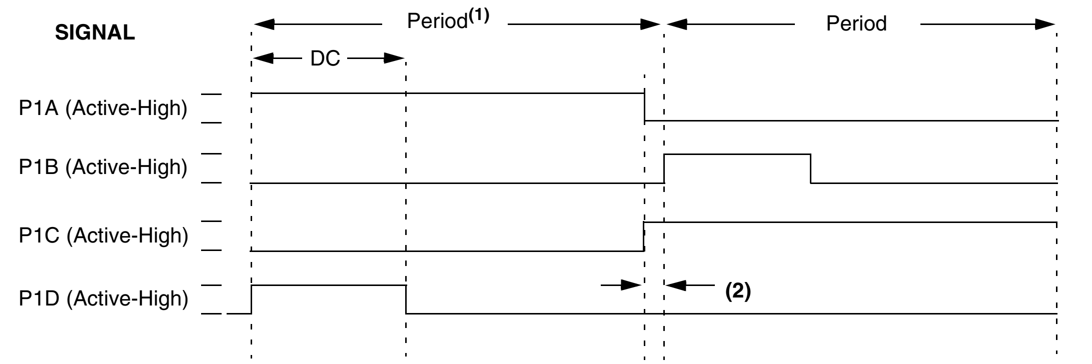
PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in ECCPR1L.

bit 3-0 **ECCP1M<3:0>:** ECCP1 Mode Select bits

`0000` = Capture/Compare/PWM off (resets ECCP module)
`0001` = Unused (reserved)
`0010` = Compare mode, toggle output on match (ECCP1IF bit is set)
`0011` = Unused (reserved)
`0100` = Capture mode, every falling edge
`0101` = Capture mode, every rising edge
`0110` = Capture mode, every 4th rising edge
`0111` = Capture mode, every 16th rising edge
`1000` = Compare mode, set output on match (ECCP1IF bit is set)
`1001` = Compare mode, clear output on match (ECCP1IF bit is set)
`1010` = Compare mode, ECCP1 pin is unaffected (ECCP1IF bit is set)
`1011` = Compare mode, trigger special event (ECCP1IF bit is set; ECCP resets TMR1or TMR3 and starts an A/D conversion if the A/D module is enabled)
`1100` = PWM mode; P1A, P1C active-high; P1B, P1D active-high
`1101` = PWM mode; P1A, P1C active-high; P1B, P1D active-low
`1110` = PWM mode; P1A, P1C active-low; P1B, P1D active-high
`1111` = PWM mode; P1A, P1C active-low; P1B, P1D active-low

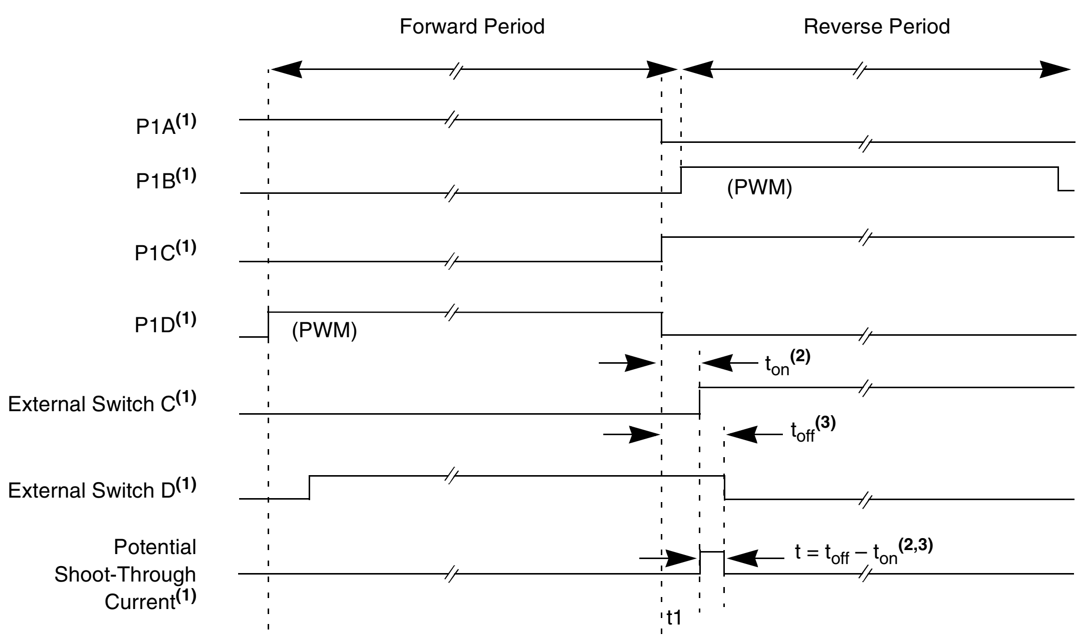
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

FIGURE 16-7: PWM DIRECTION CHANGE



Note 1: The direction bit in the ECCP1 Control Register (ECCP1CON.EPWM1M1) is written any time during the PWM cycle.
2: The P1A and P1C signals switch at intervals of TOSC, 4 TOSC or 16 TOSC, depending on the Timer2 prescaler value earlier when changing direction. The modulated P1B and P1D signals are inactive at this time.

FIGURE 16-8: PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE



Note 1: All signals are shown as active-high.
2: t_{on} is the turn-on delay of power switch and driver.
3: t_{off} is the turn-off delay of power switch and driver.

17.3.6 SLAVE MODE

In Slave mode, the data is transmitted and received as the external clock pulses appear on SCK. When the last bit is latched, the SSPIF interrupt flag bit is set.

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. When a byte is received, the device will wake-up from Sleep. Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCK pin. The Idle state is determined by the CKP bit (SSPCON1<4>).

17.3.7 SLAVE SELECT SYNCHRONIZATION

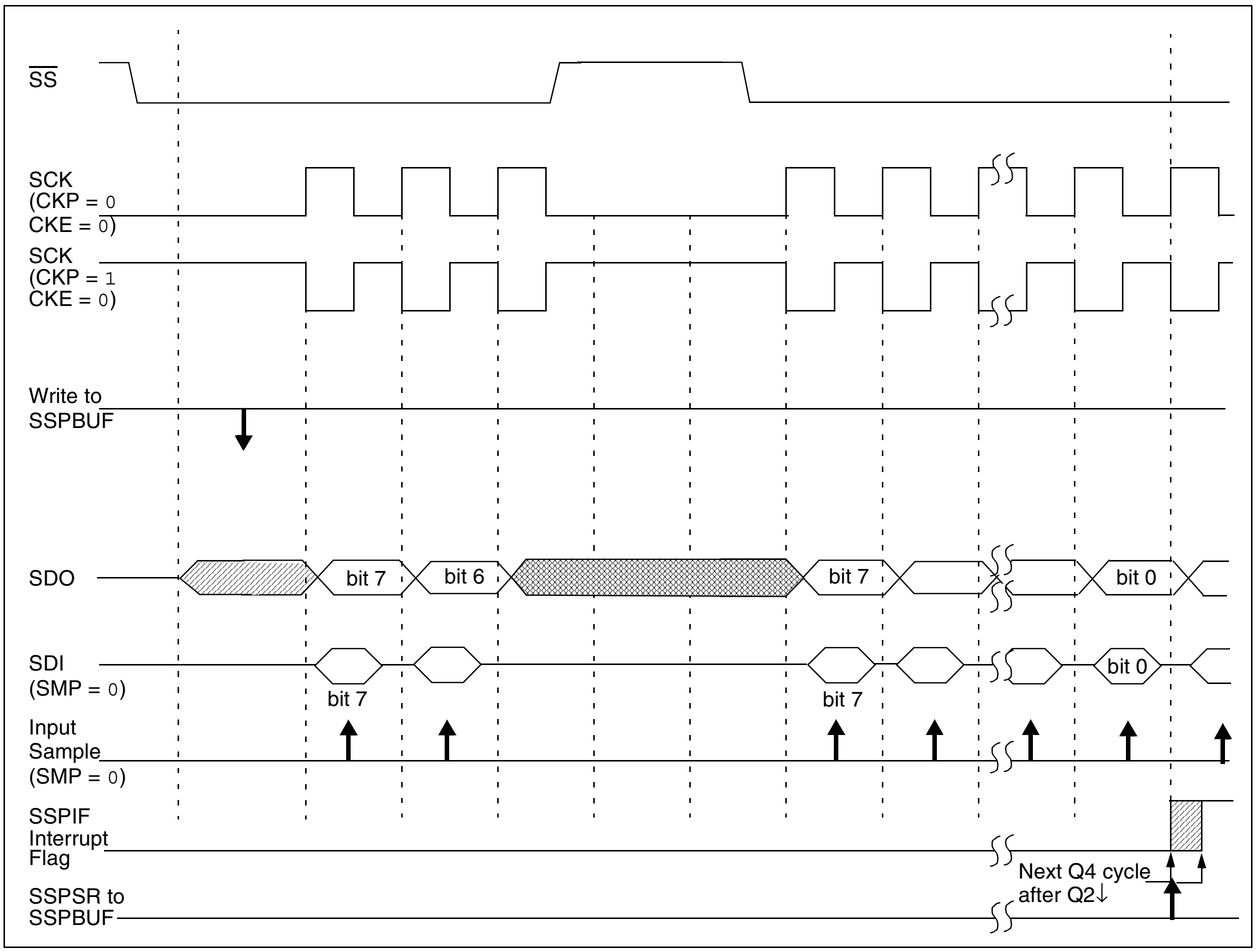
The $\overline{SS}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with $\overline{SS}$ pin control enabled (SSPCON1<3:0> = 04h). The pin must not be driven low for the $\overline{SS}$ pin to function as an input. The data latch must be high. When the $\overline{SS}$ pin is low, transmission and reception are enabled and the SDO pin is driven. When the $\overline{SS}$ pin goes high, the SDO pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

> **Note 1:** When the SPI is in Slave mode with $\overline{SS}$ pin control enabled (SSPCON1<3:0> = `0100`), the SPI module will reset if the $\overline{SS}$ pin is set to VDD.
>
> **2:** If the SPI is used in Slave mode with CKE set, then the $\overline{SS}$ pin control must be enabled.

When the SPI module resets, the bit counter is forced to '0'. This can be done by either forcing the $\overline{SS}$ pin to a high level or clearing the SSPEN bit.

To emulate two-wire communication, the SDO pin can be connected to the SDI pin. When the SPI needs to operate as a receiver, the SDO pin can be configured as an input. This disables transmissions from the SDO. The SDI can always be left as an input (SDI function) since it cannot create a bus conflict.

FIGURE 17-4: SLAVE SYNCHRONIZATION WAVEFORM



17.4.2 OPERATION

The MSSP module functions are enabled by setting MSSP Enable bit, SSPEN (SSPCON1<5>).

The SSPCON1 register allows control of the I^2C operation. Four mode selection bits (SSPCON1<3:0>) allow one of the following I^2C modes to be selected:

- I^2C Master mode, clock = OSC/4 (SSPADD +1)
- I^2C Slave mode (7-bit address)
- I^2C Slave mode (10-bit address)
- I^2C Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- I^2C Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- I^2C Firmware Controlled Master mode, slave is Idle

Selection of any I^2C mode with the SSPEN bit set forces the SCL and SDA pins to be open-drain, provided these pins are programmed to inputs by setting the appropriate TRISC bits. To ensure proper operation of the module, pull-up resistors must be provided externally to the SCL and SDA pins.

17.4.3 SLAVE MODE

In Slave mode, the SCL and SDA pins must be configured as inputs (TRISC<4:3> set). The MSSP module will override the input state with the output data when required (slave-transmitter).

The I^2C Slave mode hardware will always generate an interrupt on an address match. Through the mode select bits, the user can also choose to interrupt on Start and Stop bits.

When an address is matched, or the data transfer after an address match is received, the hardware automatically will generate the Acknowledge ($\overline{\text{ACK}}$) pulse and load the SSPBUF register with the received value currently in the SSPSR register.

Any combination of the following conditions will cause the MSSP module not to give this $\overline{\text{ACK}}$ pulse:

- The Buffer Full bit, BF (SSPSTAT<0>), was set before the transfer was received.
- The overflow bit, SSPOV (SSPCON1<6>), was set before the transfer was received.

In this case, the SSPSR register value is not loaded into the SSPBUF, but bit SSPIF (PIR1<3>) is set. The BF bit is cleared by reading the SSPBUF register, while bit SSPOV is cleared through software.

The SCL clock input must have a minimum high and low for proper operation. The high and low times of the I^2C specification, as well as the requirement of the MSSP module, are shown in timing parameter #100 and parameter #101.

17.4.3.1 Addressing

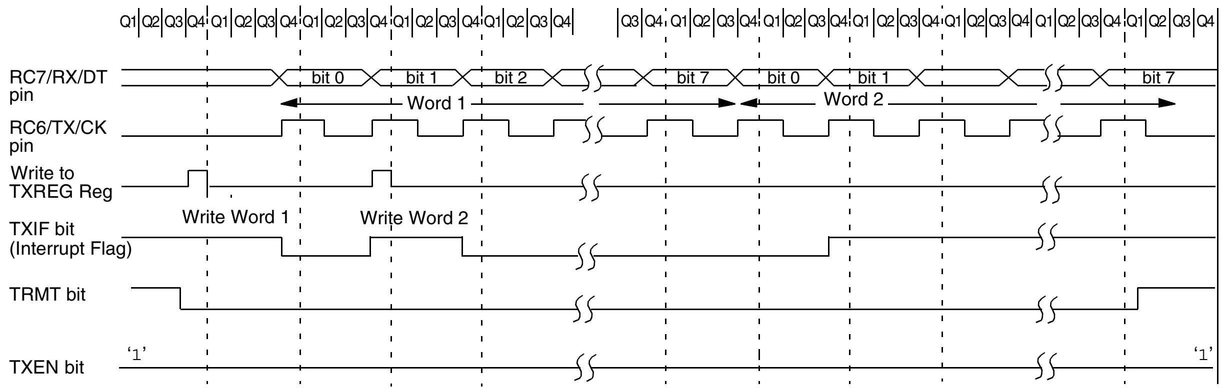
Once the MSSP module has been enabled, it waits for a Start condition to occur. Following the Start condition, the 8 bits are shifted into the SSPSR register. All incoming bits are sampled with the rising edge of the clock (SCL) line. The value of register SSPSR<7:1> is compared to the value of the SSPADD register. The address is compared on the falling edge of the eighth clock (SCL) pulse. If the addresses match and the BF and SSPOV bits are clear, the following events occur:

1. The SSPSR register value is loaded into the SSPBUF register.
2. The Buffer Full bit BF is set.
3. An $\overline{\text{ACK}}$ pulse is generated.
4. MSSP Interrupt Flag bit, SSPIF (PIR1<3>), is set (interrupt is generated if enabled) on the falling edge of the ninth SCL pulse.

In 10-bit Address mode, two address bytes need to be received by the slave. The five Most Significant bits (MSbs) of the first address byte specify if this is a 10-bit address. Bit R/$\overline{\text{W}}$ (SSPSTAT<2>) must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '`11110 A9 A8 0`', where '`A9`' and '`A8`' are the two MSbs of the address. The sequence of events for 10-bit address is as follows, with steps 7 through 9 for the slave-transmitter:

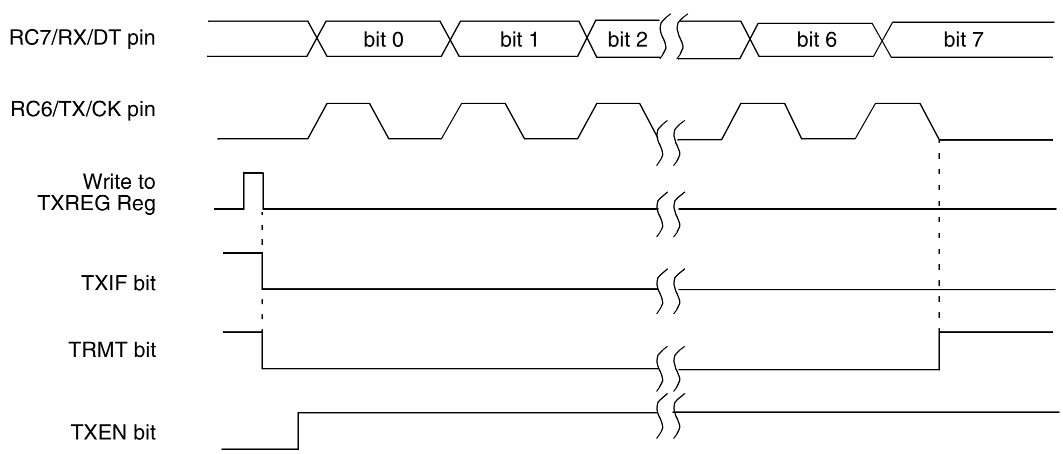
1. Receive first (high) byte of address (bits SSPIF, BF and bit UA (SSPSTAT<1>) are set).
2. Update the SSPADD register with second (low) byte of address (clears bit UA and releases the SCL line).
3. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
4. Receive second (low) byte of address (bits SSPIF, BF and UA are set).
5. Update the SSPADD register with the first (high) byte of address. If match releases SCL line, this will clear bit UA.
6. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
7. Receive Repeated Start condition.
8. Receive first (high) byte of address (bits SSPIF and BF are set).
9. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.

FIGURE 18-6: SYNCHRONOUS TRANSMISSION



Note: Sync Master mode; SPBRG = `0`; continuous transmission of two 8-bit words.

FIGURE 18-7: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)



24.0 SPECIAL FEATURES OF THE CPU

There are several features intended to maximize system reliability, minimize cost through elimination of external components, provide power-saving operating modes and offer code protection. These are:

- Oscillator Selection
- Reset
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- Sleep
- Code Protection
- ID Locations
- In-Circuit Serial Programming

All PIC18FXX8 devices have a Watchdog Timer which is permanently enabled via the configuration bits or software controlled. It runs off its own RC oscillator for added reliability. There are two timers that offer necessary delays on power-up. One is the Oscillator Start-up Timer (OST), intended to keep the chip in Reset until the crystal oscillator is stable. The other is the Power-up Timer (PWRT) which provides a fixed delay on power-up only, designed to keep the part in Reset while the power supply stabilizes. With these two timers on-chip, most applications need no external Reset circuitry.

Sleep mode is designed to offer a very Low-Current Power-Down mode. The user can wake-up from Sleep through external Reset, Watchdog Timer wake-up or through an interrupt. Several oscillator options are also made available to allow the part to fit the application. The RC oscillator option saves system cost while the LP crystal option saves power. A set of configuration bits is used to select various options.

24.1 Configuration Bits

The configuration bits can be programmed (read as '0') or left unprogrammed (read as '1'), to select various device configurations. These bits are mapped starting at program memory location 300000h.

The user will note that address 300000h is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h-3FFFFFh) which can only be accessed using table reads and table writes.

Programming the Configuration registers is done in a manner similar to programming the Flash memory. The EECON1 register $\overline{\text{WR}}$ bit starts a self-timed write to the Configuration register. In normal operation mode, a `TBLWT` instruction, with the TBLPTR pointed to the Configuration register, sets up the address and the data for the Configuration register write. Setting the $\overline{\text{WR}}$ bit starts a long write to the Configuration register. The Configuration registers are written a byte at a time. To write or erase a configuration cell, a `TBLWT` instruction can write a '1' or a '0' into the cell.

TABLE 24-1: CONFIGURATION BITS AND DEVICE IDS

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprogrammed Value
300001h	CONFIG1H	—	—	$\overline{\text{OSCSEN}}$	—	—	FOSC2	FOSC1	FOSC0	`--1- -111`
300002h	CONFIG2L	—	—	—	—	BORV1	BORV0	BOREN	$\overline{\text{PWRTEN}}$	`---- 1111`
300003h	CONFIG2H	—	—	—	—	WDTPS2	WDTPS1	WDTPS0	WDTEN	`---- 1111`
300006h	CONFIG4L	$\overline{\text{DEBUG}}$	—	—	—	—	LVP	—	STVREN	`1--- -1-1`
300008h	CONFIG5L	—	—	—	—	CP3	CP2	CP1	CP0	`---- 1111`
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	`11-- ----`
30000Ah	CONFIG6L	—	—	—	—	WRT3	WRT2	WRT1	WRT0	`---- 1111`
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—	`111- ----`
30000Ch	CONFIG7L	—	—	—	—	EBTR3	EBTR2	EBTR1	EBTR0	`---- 1111`
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	`-1-- ----`
3FFFFEh	DEVID1	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	**(1)**
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	`0000 1000`

Legend: `x` = unknown, `u` = unchanged, - = unimplemented, `q` = value depends on condition. Shaded cells are unimplemented, read as '0'.

Note 1: See Register 24-11 for DEVID1 values.

IORLW **Inclusive OR Literal with W**

Syntax:	[*label*] IORLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .OR. k $\rightarrow$ W
Status Affected:	N, Z
Encoding:	`0000` `1001` `kkkk` `kkkk`
Description:	The contents of W are ORed with 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: `IORLW 0x35`

Before Instruction
W = 0x9A
After Instruction
W = 0xBF

IORWF **Inclusive OR W with f**

Syntax:	[*label*] IORWF f [,d [,a]]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(W) .OR. (f) $\rightarrow$ dest
Status Affected:	N, Z
Encoding:	`0001` `00da` `ffff` `ffff`
Description:	Inclusive OR W with register ‘f’. If ‘d’ is ‘`0`’, the result is placed in W. If ‘d’ is ‘`1`’, the result is placed back in register ‘f’ (default). If ‘a’ is ‘`0`’, the Access Bank will be selected, overriding the BSR value. If ‘a’ = `1`, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `IORWF RESULT, W`

Before Instruction
RESULT = 0x13
W = 0x91
After Instruction
RESULT = 0x13
W = 0x93

TSTFSZ **Test f, Skip if 0**

Syntax:	[*label*] TSTFSZ f [,a]
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$
Operation:	skip if f = 0
Status Affected:	None
Encoding:	`0110` `011a` `ffff` `ffff`
Description:	If 'f' = 0, the next instruction fetched during the current instruction execution is discarded and a `NOP` is executed, making this a two-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

<u>Example:</u>

```
HERE     TSTFSZ CNT
NZERO     :
ZERO     :
```

Before Instruction
PC = Address (HERE)
After Instruction
If CNT = 0x00,
PC = Address (ZERO)
If CNT ≠ 0x00,
PC = Address (NZERO)

XORLW **Exclusive OR Literal with W**

Syntax:	[*label*] XORLW k
Operands:	$0 \leq k \leq 255$
Operation:	(W) .XOR. k → W
Status Affected:	N, Z
Encoding:	`0000` `1010` `kkkk` `kkkk`
Description:	The contents of W are XORed with the 8-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

<u>Example:</u> `XORLW 0xAF`

Before Instruction
W = 0xB5
After Instruction
W = 0x1A

XORWF	Exclusive OR W with f			
Syntax:	[*label*] XORWF f [,d [,a]]			
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$			
Operation:	(W) .XOR. (f) $\rightarrow$ dest			
Status Affected:	N, Z			
Encoding:	0001	10da	ffff	ffff
Description:	Exclusive OR the contents of W with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).			
Words:	1			
Cycles:	1			

Q Cycle Activity:

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

Example: `XORWF REG`

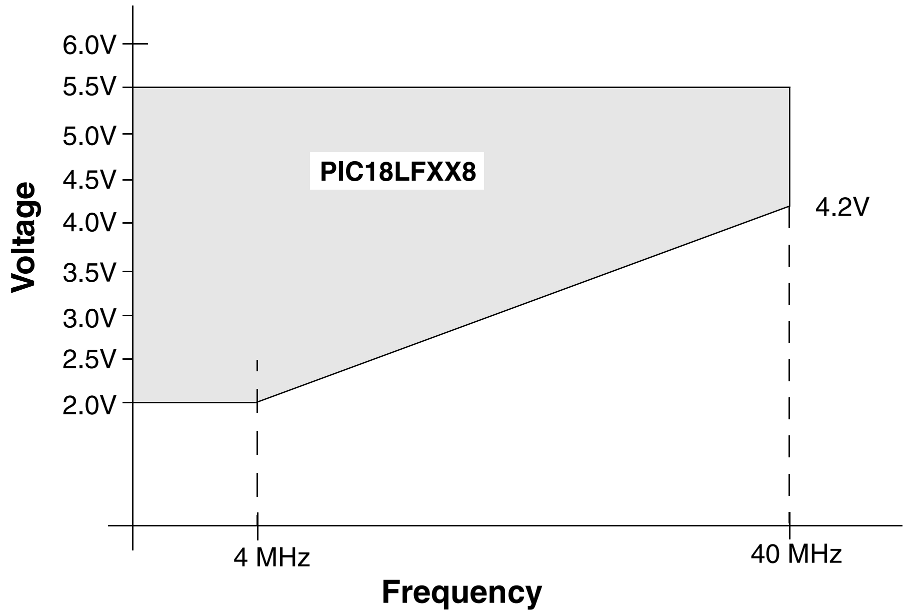
Before Instruction

REG	=	0xAF
W	=	0xB5

After Instruction

REG	=	0x1A
W	=	0xB5

FIGURE 27-3: PIC18LFXX8 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)



FMAX = (16.36 MHz/V) (VDDAPPMIN – 2.0V) + 4 MHz, if VDDAPPMIN ≤ 4.2V = 40 MHz, if VDDAPPMIN > 4.2V

Note: VDDAPPMIN is the minimum voltage of the PICmicro® device in the application.

27.2 DC Characteristics: PIC18FXX8 (Industrial, Extended) PIC18LFXX8 (Industrial)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended			
Param No.	**Symbol**	**Characteristic/ Device**	**Min**	**Max**	**Units**	**Conditions**
	VIL	**Input Low Voltage**				
		I/O ports:				
D030		with TTL buffer	VSS	0.15 VDD	V	VDD < 4.5V
D030A			—	0.8	V	4.5V ≤ VDD ≤ 5.5V
D031		with Schmitt Trigger buffer RC3 and RC4	VSS VSS	0.2 VDD 0.3 VDD	V V	
D032		$\overline{\text{MCLR}}$	VSS	0.2 VDD	V	
D032A		OSC1 (in XT, HS and LP modes) and T1OSI	VSS	0.3 VDD	V	
D033		OSC1 (in RC mode)(1)	VSS	0.2 VDD	V	
	VIH	**Input High Voltage**				
		I/O ports:				
D040		with TTL buffer	0.25 VDD + 0.8V	VDD	V	VDD < 4.5V
D040A			2.0	VDD	V	4.5V ≤ VDD ≤ 5.5V
D041		with Schmitt Trigger buffer RC3 and RC4	0.8 VDD 0.7 VDD	VDD VDD	V V	
D042		$\overline{\text{MCLR}}$	0.8 VDD	VDD	V	
D042A		OSC1 (in XT, HS and LP modes) and T1OSI	0.7 VDD	VDD	V	
D043		OSC1 (RC mode)(1)	0.9 VDD	VDD	V	
	IIL	**Input Leakage Current(2,3)**				
D060		I/O ports	—	±1	μA	VSS ≤ VPIN ≤ VDD, Pin at high-impedance
D061		$\overline{\text{MCLR}}$	—	±5	μA	Vss ≤ VPIN ≤ VDD
D063		OSC1	—	±5	μA	Vss ≤ VPIN ≤ VDD
	IPU	**Weak Pull-up Current**				
D070	IPURB	PORTB weak pull-up current	50	450	μA	VDD = 5V, VPIN = VSS

Note 1: In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PICmicro® device be driven with an external clock while in RC mode.

2: The leakage current on the $\overline{\text{MCLR}}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as current sourced by the pin.

NOTES:

BSF 293
BTFSC 294
BTFSS 294
BTG 295
BZ 296

C

C Compilers
    MPLAB C17 324
    MPLAB C18 324
    MPLAB C30 324
CALL 296
CAN Module
    Aborting Transmission 228
    Acknowledge Error 237
    Baud Rate Registers 218
    Baud Rate Setting 233
    Bit Error 237
    Bit Time Partitioning (diagram) 233
    Bit Timing Configuration Registers 236
        BRGCON1 236
        BRGCON2 236
        BRGCON3 236
    Calculating TQ, Nominal Bit Rate and
        Nominal Bit Time 234
    Configuration Mode 226
    Control and Status Registers 201
    Controller Register Map 225
    CRC Error 237
    Disable Mode 226
    Error Detection 237
    Error Modes and Error Counters 237
    Error Modes State (diagram) 238
    Error Recognition Mode 227
    Error States 237
    Filter/Mask Truth (table) 232
    Form Error 237
    Hard Synchronization 235
    I/O Control Register 221
    Information Processing Time 234
    Initiating Transmission 228
    Internal Message Reception
        Flowchart 231
    Internal Transmit Message
        Flowchart 229
    Interrupt Acknowledge 239
    Interrupt Registers 222
    Interrupts 238
        Bus Activity Wake-up 239
        Bus-Off 239
        Code Bits 238
        Error 239
        Message Error 239
        Receive 238
        Receiver Bus Passive 239
        Receiver Overflow 239
        Receiver Warning 239
        Transmit 238
        Transmitter Bus Passive 239
        Transmitter Warning 239
    Lengthening a Bit Period
        (diagram) 235
    Listen Only Mode 226
    Loopback Mode 227
    Message Acceptance Filters
        and Masks 215, 232
    Message Acceptance Mask and
        Filter Operation (diagram) 232
    Message Reception 230
    Message Time-Stamping 230
    Message Transmission 227
    Modes of Operation 226
    Normal Mode 226
    Oscillator Tolerance 236
    Overview 199
    Phase Buffer Segments 234
    Programming Time Segments 236
    Propagation Segment 234
    Receive Buffer Registers 210
    Receive Buffers 230
    Receive Message Buffering 230
    Receive Priority 230
    Registers 201
    Resynchronization 235
    Sample Point 234
    Shortening a Bit Period (diagram) 236
    Stuff Bit Error 237
    Synchronization 235
    Synchronization Rules 235
    Synchronization Segment 234
    Time Quanta 234
    Transmit Buffer Registers 206
    Transmit Buffers 227
    Transmit Priority 227
    Transmit/Receive Buffers 199
    Values for ICODE (table) 239
Capture (CCP Module) 124
    CAN Message Time-Stamp 125
    CCP Pin Configuration 124
    CCP1 Prescaler 125
    CCPR1H:CCPR1L Registers 124
    Software Interrupt 125
    Timer1/Timer3 Mode Selection 124
Capture (ECCP Module) 133
    CAN Message Time-Stamp 133
Capture/Compare/PWM (CCP) 123
    Capture Mode. *See* Capture
        (CCP Module).
    CCP1 Module 124
        Timer Resources 124
    CCPR1H Register 124
    CCPR1L Register 124
    Compare Mode. *See* Compare
        (CCP Module).
    Interaction of CCP1 and
        ECCP1 Modules 124
    PWM Mode. *See* PWM (CCP Module).
Ceramic Resonators
    Ranges Tested 17
Clocking Scheme 41
CLRF 297
CLRWDT 297