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	25MHz
Connectivity	EBI/EMI, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	37
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	1.5K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	External
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/pic18lc801-i-pt

2.6 Oscillator Switching Feature

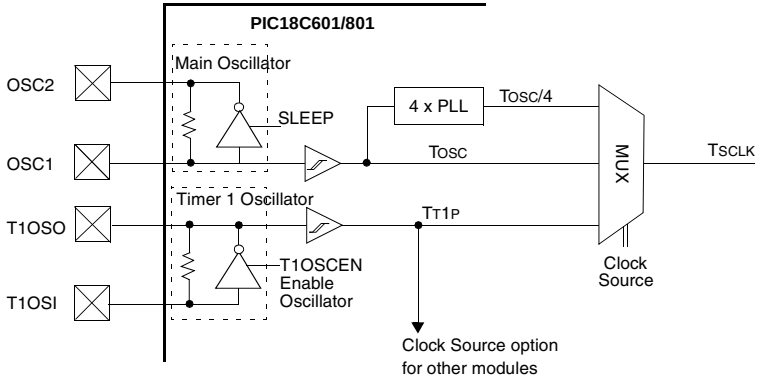
PIC18C601/801 devices include a feature that allows the system clock source to be switched from the main oscillator to an alternate low frequency clock source. For PIC18C601/801 devices, this alternate clock source is the Timer1 oscillator. If a low frequency crystal (32 kHz, for example) has been attached to the Timer1 oscillator pins and the Timer1 oscillator has been enabled, the device can switch to a low power execution mode. Figure 2-5 shows a block diagram of the system clock sources.

2.6.1 SYSTEM CLOCK SWITCH BIT

The system clock source switching is performed under software control. The system clock switch bit, SCS0 (OSCCON register), controls the clock switching. When the SCS0 bit is '0', the system clock source comes from the main oscillator, selected by the FOSC2:FOSC0 configuration bits in CONFIG1H register. When the SCS0 bit is set, the system clock source will come from the Timer1 oscillator. The SCS0 bit is cleared on all forms of RESET.

Note:	The Timer1 oscillator must be enabled to switch the system clock source. The Timer1 oscillator is enabled by setting the T1OSCEN bit in the Timer1 control register (T1CON). If the Timer1 oscillator is not enabled, any write to the SCS0 bit will be ignored (SCS0 bit forced cleared) and the main oscillator will continue to be the system clock source.

FIGURE 2-5: DEVICE CLOCK SOURCES



Note: I/O pins have diode protection to VDD and VSS.

 © 2001-2013 Microchip Technology Inc.

5.0 EXTERNAL MEMORY INTERFACE

The External Memory Interface is a feature of the PIC18C601/801 that allows the processor to access external memory devices, such as FLASH, EPROM, SRAM, etc. Memory mapped peripherals may also be accessed.

The External Memory Interface physical implementation includes up to 26 pins on the PIC18C601 and up to 38 pins on the PIC18C801. These pins are reserved for external address/data bus functions.

These pins are multiplexed with I/O port pins, but the I/O functions are only enabled when program execution takes place in internal Boot RAM and the EBDIS bit in the MEMCON register is set (see Register 5-1).

5.1 Memory Control Register (MEMCON)

Register 5-1 shows the Memory Control Register (MEMCON). This register contains bits used to control the operation of the External Memory Interface.

REGISTER 5-1: MEMCON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
EBDIS	PGRM	WAIT1	WAIT0	—	—	WM1	WM0
bit7							bit0

bit 7 **EBDIS**: External Bus Disable

1 = External system bus disabled, all external bus drivers are mapped as I/O ports
0 = External system bus enabled, and I/O ports are disabled

bit 6 **PGRM**: Program RAM Enable

1 = 512 bytes of internal RAM enabled as internal program memory from location 1FFE00h to 1FFFFFh, external program memory at these locations is unused. Internal GPR memory from 400h to 5FFh is disabled and returns 00h.

0 = Internal RAM enabled as internal GPR memory from 400h to 5FFh. Program memory from location 1FFE00h to 1FFFFFh is configured as external program memory.

bit 5-4 **WAIT<1:0>**: Table Reads and Writes Bus Cycle Wait Count

11 = Table reads and writes will wait 0 TCY
10 = Table reads and writes will wait 1 TCY
01 = Table reads and writes will wait 2 TCY
00 = Table reads and writes will wait 3 TCY

bit 3-2 **Unimplemented**: Read as '0'

bit 1-0 **WM<1:0>**: TABLWT Operation with 16-bit Bus

1X = Word Write mode: TABLAT0 and TABLAT1 word output, $\overline{WRH}$ active when TABLAT1 written
01 = Byte Select mode: TABLAT data copied on both MS and LS Byte, $\overline{WRH}$ and ($\overline{UB}$ or $\overline{LB}$) will activate
00 = Byte Write mode: TABLAT data copied on both MS and LS Byte, $\overline{WRH}$ or $\overline{WRL}$ will activate

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

6.2 Table Read

The TBLRD instruction is used to retrieve data from external program memory and place it into data memory.

TBLPTR points to a byte address in external program memory space. Executing TBLRD places the byte into TABLAT. In addition, TBLPTR can be modified automatically for the next Table Read operation.

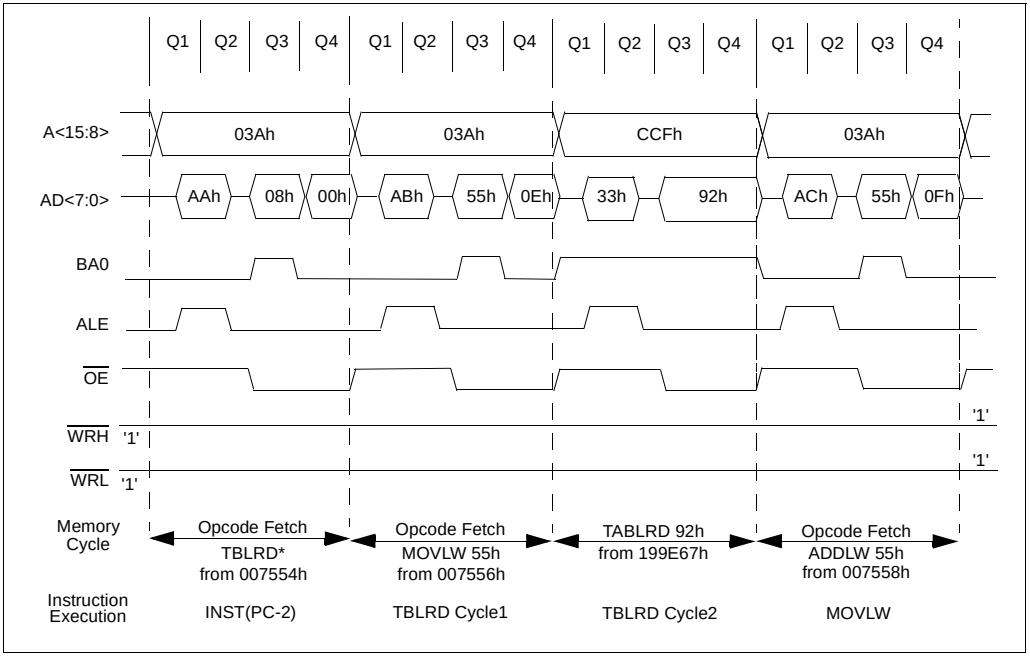
Table Reads from external program memory are performed one byte at a time. If the external interface is 8-bit, the bus interface circuitry in TABLAT will load the external value into TABLAT. If the external interface is 16-bit, interface circuitry in TABLAT will select either the high or low byte of the data from the 16-bit bus, based on the least significant bit of the address.

Example 6-1describes how to use TBLRD. Figure 6-3 and Figure 6-4 show Table Read timings for an 8-bit external interface, and Figure 6-5 describes Table Read timing for a 16-bit interface.

EXAMPLE 6-1: TABLE READ CODE EXAMPLE

```
; Read a byte from location 0020h
CLRF    TBLPTRU             ; clear upper 5 bits of TBLPTR
CLRF    TBLPTRH             ; clear higher 8 bits of TBLPTR
MOVLW   20h                 ; Load 20h into
MOVWF   TBLPTRL             ; TBLPTRL
TBLRD*                      ; Data is in TABLAT
```

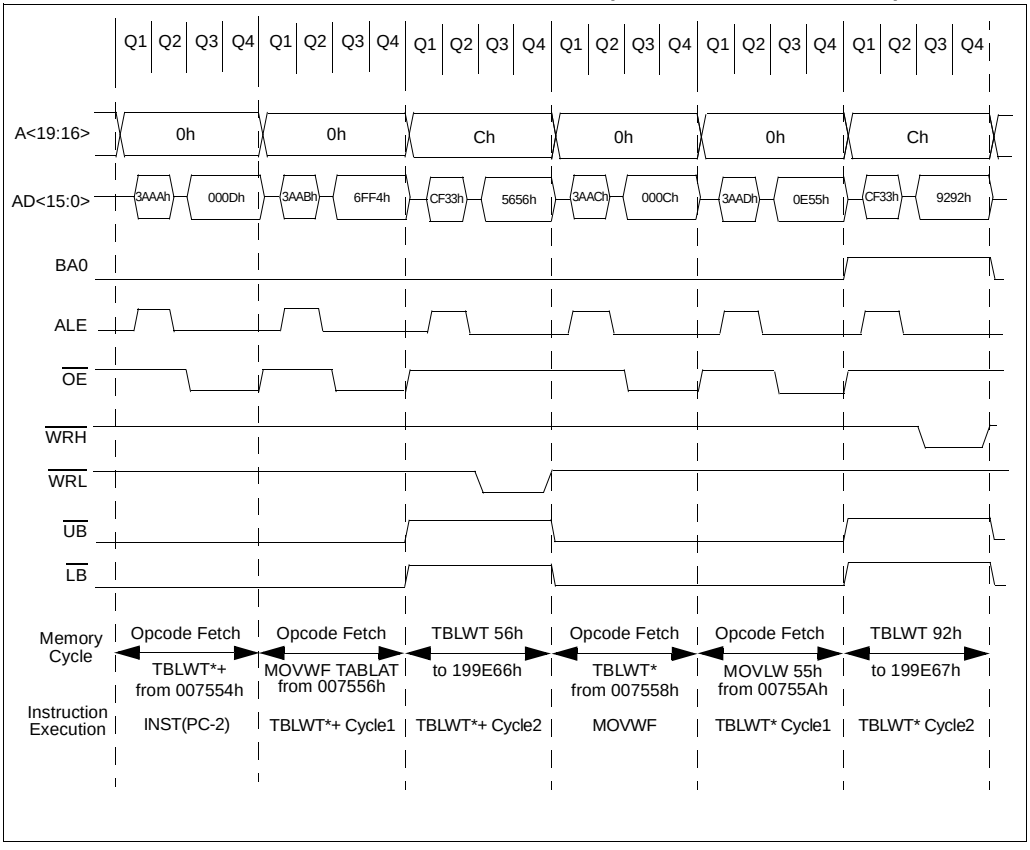
FIGURE 6-3: TBLRD EXTERNAL INTERFACE TIMING (8-BIT MULTIPLEXED MODE)



6.3.2 16-BIT EXTERNAL TABLE WRITE (BYTE WRITE MODE)

This mode allows Table Writes to byte-wide external memories. During a TBLWT cycle, the TABLAT data is presented on the upper and lower byte of the AD<15:0> bus. The appropriate WRH or WRL line is strobed based on the LSb of the TBLPTR. Figure 6-8 shows the timing associated with this mode.

FIGURE 6-8: TBLWT EXTERNAL INTERFACE TIMING (16-BIT BYTE WRITE MODE)



 © 2001-2013 Microchip Technology Inc.

7.1 Operation

Example 7-1 shows the sequence to perform an 8 x 8 unsigned multiply. Only one instruction is required when one argument of the multiply is already loaded in the WREG register.

Example 7-2 shows the sequence to do an 8 x 8 signed multiply. To account for the sign bits of the arguments, each argument's most significant bit (MSb) is tested and the appropriate subtractions are done.

EXAMPLE 7-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

```
MOVFF    ARG1, WREG  ;
MULWF    ARG2        ; ARG1 * ARG2 ->
                     ;   PRODH:PRODL
```

EXAMPLE 7-2: 8 x 8 SIGNED MULTIPLY ROUTINE

```
MOVFF    ARG1, WREG
MULWF    ARG2            ; ARG1 * ARG2 ->
                         ; PRODH:PRODL
BTFSC    ARG2, SB        ; Test Sign Bit
SUBWF    PRODH           ; PRODH = PRODH
                         ;         - ARG1
MOVFF    ARG2, WREG
BTFSC    ARG1, SB        ; Test Sign Bit
SUBWF    PRODH           ; PRODH = PRODH
                         ;         - ARG2
```

Example 7-3 shows the sequence to perform a 16 x 16 unsigned multiply. Equation 7-1 shows the algorithm that is used. The 32-bit result is stored in 4 registers RES3:RES0.

EQUATION 7-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L)
\end{aligned}
$$

EXAMPLE 7-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE

```
   MOVFF    ARG1L, WREG
   MULWF    ARG2L           ; ARG1L * ARG2L ->
                            ; PRODH:PRODL
   MOVFF    PRODH, RES1     ;
   MOVFF    PRODL, RES0     ;
;
   MOVFF    ARG1H, WREG
   MULWF    ARG2H           ; ARG1H * ARG2H ->
                            ; PRODH:PRODL
   MOVFF    PRODH, RES3     ;
   MOVFF    PRODL, RES2     ;
;
   MOVFF    ARG1L, WREG
   MULWF    ARG2H           ; ARG1L * ARG2H ->
                            ; PRODH:PRODL
   MOVF     PRODL, W        ;
   ADDWF    RES1            ; Add cross
   MOVF     PRODH, W        ; products
   ADDWFC   RES2            ;
   CLRF     WREG            ;
   ADDWFC   RES3            ;
;
   MOVFF    ARG1H, WREG     ;
   MULWF    ARG2L           ; ARG1H * ARG2L ->
                            ; PRODH:PRODL
   MOVF     PRODL, W        ;
   ADDWF    RES1            ; Add cross
   MOVF     PRODH, W        ; products
   ADDWFC   RES2            ;
   CLRF     WREG            ;
   ADDWFC   RES3            ;
```

Example 7-4 shows the sequence to perform a 16 x 16 signed multiply. Equation 7-2 shows the algorithm used. The 32-bit result is stored in four registers, RES3:RES0. To account for the sign bits of the arguments, each argument pairs' most significant bit (MSb) is tested and the appropriate subtractions are done.

EQUATION 7-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L) + \\
&\quad (-1 \bullet ARG2H\langle 7\rangle \bullet ARG1H:ARG1L \bullet 2^{16}) + \\
&\quad (-1 \bullet ARG1H\langle 7\rangle \bullet ARG2H:ARG2L \bullet 2^{16})
\end{aligned}
$$

 © 2001-2013 Microchip Technology Inc.

REGISTER 8-10: IPR2 REGISTER

U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	—	BCLIP	LVDIP	TMR3IP	CCP2IP
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **BCLIP**: Bus Collision Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **LVDIP**: Low Voltage Detect Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR3IP**: TMR3 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **CCP2IP**: CCP2 Interrupt Priority bit
1 = High priority
0 = Low priority

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

 © 2001-2013 Microchip Technology Inc.

8.1.6 INT INTERRUPTS

External interrupts on the RB0/INT0, RB1/INT1 and RB2/INT2 pins are edge triggered: either rising, if the corresponding INTEDGx bit is set in the INTCON2 register, or falling, if the INTEDGx bit is clear. When a valid edge appears on the RBx/INTx pin, the corresponding flag bit INTxIF is set. This interrupt can be disabled by clearing the corresponding enable bit INTxIE. Flag bit INTxIF must be cleared in software in the Interrupt Service Routine before re-enabling the interrupt. All external interrupts (INT0, INT1 and INT2) can wake-up the processor from SLEEP, if bit INTxIE was set prior to going into SLEEP. If the global interrupt enable bit GIE is set, the processor will branch to the interrupt vector following wake-up.

Interrupt priority for INT1 and INT2 is determined by the value contained in the interrupt priority bits INT1IP (INTCON3 register) and INT2IP (INTCON3 register). There is no priority bit associated with INT0; it is always a high priority interrupt source.

8.1.7 TMR0 INTERRUPT

In 8-bit mode (which is the default), an overflow (0FFh $\rightarrow$ 00h) in the TMR0 register will set flag bit TMR0IF. In 16-bit mode, an overflow (0FFFFh $\rightarrow$ 0000h) in the TMR0H:TMR0L registers will set flag bit TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit TMR0IE (INTCON register). Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit TMR0IP (INTCON2 register). See Section 10.0 for further details on the Timer0 module.

8.1.8 PORTB INTERRUPT-ON-CHANGE

An input change on PORTB<7:4> sets flag bit RBIF (INTCON register). The interrupt can be enabled/disabled by setting/clearing enable bit RBIE (INTCON register). Interrupt priority for PORTB interrupt-on-change is determined by the value contained in the interrupt priority bit RBIP (INTCON2 register).

8.2 Context Saving During Interrupts

During an interrupt, the return PC value is saved on the stack. Additionally, the WREG, STATUS and BSR registers are saved on the fast return stack. If a fast return from interrupt is not used (See Section 4.3), the user may need to save the WREG, STATUS and BSR registers in software. Depending on the user's application, other registers may also need to be saved. Example 8-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

EXAMPLE 8-1: SAVING STATUS, WREG AND BSR REGISTERS IN RAM

```
MOVWF    W_TEMP                    ; W_TEMP is in Low Access bank
MOVFF    STATUS, STATUS_TEMP       ; STATUS_TEMP located anywhere
MOVFF    BSR, BSR_TEMP             ; BSR located anywhere
;
; USER ISR CODE
;
MOVFF    BSR_TEMP, BSR             ; Restore BSR
MOVF     W_TEMP, W                 ; Restore WREG
MOVFF    STATUS_TEMP, STATUS       ; Restore STATUS
```

9.3 PORTC, TRISC and LATC Registers

PORTC is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATC register, read and write the latched output value for PORTC.

PORTC is multiplexed with several peripheral functions (Table 9-5). PORTC pins have Schmitt Trigger input buffers.

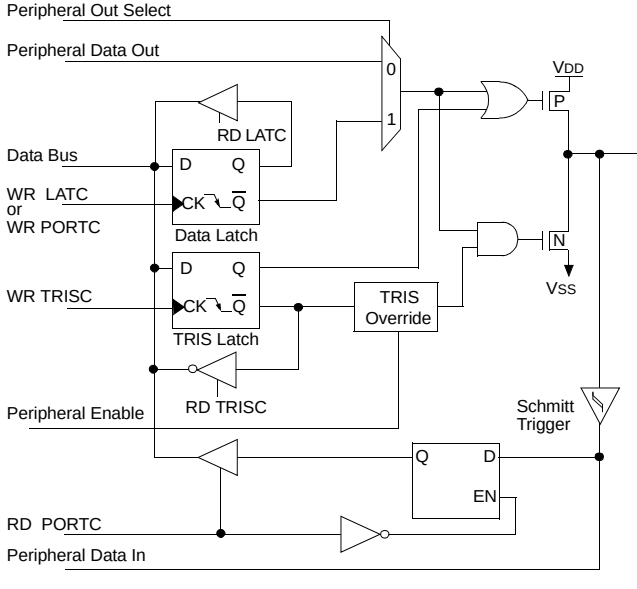
When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

The pin override value is not loaded into the TRIS register. This allows read-modify-write of the TRIS register, without concern due to peripheral overrides.

EXAMPLE 9-3: INITIALIZING PORTC

```
CLRF    PORTC      ; Initialize PORTC by
                   ; clearing output
                   ; data latches
CLRF    LATC       ; Alternate method
                   ; to clear output
                   ; data latches
MOVLW   0CFh       ; Value used to
                   ; initialize data
                   ; direction
MOVWF   TRISC      ; Set RC3:RC0 as inputs
                   ; RC5:RC4 as outputs
                   ; RC7:RC6 as inputs
```

FIGURE 9-6: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE)



TRIS OVERRIDE

Pin	Override	Peripheral
RC0	Yes	Timer1 OSC for Timer1/Timer3
RC1	Yes	Timer1 OSC for Timer1/Timer3
RC2	No	—
RC3	Yes	SPI/I^{2}C Master Clock
RC4	Yes	I^{2}C Data Out
RC5	Yes	SPI Data Out
RC6	Yes	USART Async Xmit, Sync Clock
RC7	Yes	USART Sync Data Out

Note 1: I/O pins have diode protection to VDD and VSS.

 © 2001-2013 Microchip Technology Inc.

9.5 PORTE, TRISE and LATE Registers

PORTE is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISE. Setting a TRISE bit (= 1) will make the corresponding PORTE pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISE bit (= 0) will make the corresponding PORTE pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATE register reads and writes the latched output value for PORTE.

PORTE is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output. PORTE is multiplexed with several peripheral functions (Table 9-9).

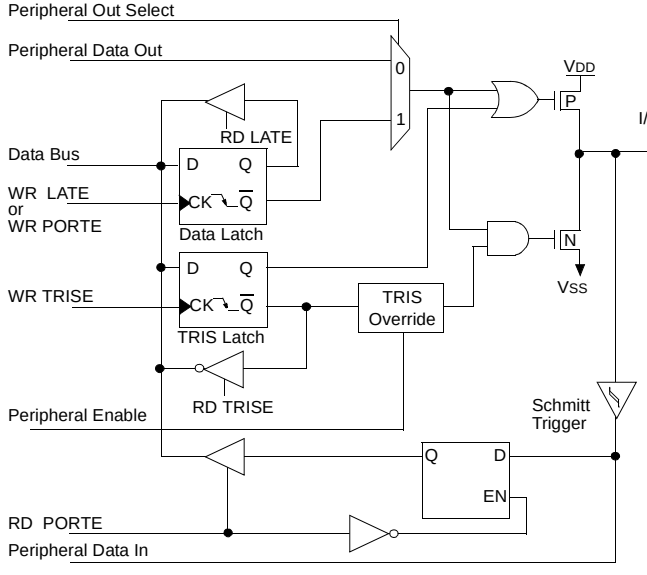
PORTE is multiplexed with the system bus and is available only when the system bus is disabled, by setting EBDIS bit in register MEMCON. When operating as the system bus, PORTE is configured as the high order byte of the address/data bus (AD15:AD8), or as the high order address byte (A15:A8), if address and data buses are de-multiplexed.

Note:	On Power-on Reset, PORTE defaults to the system bus.

EXAMPLE 9-5: INITIALIZING PORTE

```
CLRF    PORTE     ; Initialize PORTE by
                  ; clearing output
                  ; data latches
CLRF    LATE      ; Alternate method
                  ; to clear output
                  ; data latches
MOVLW   03h       ; Value used to
                  ; initialize data
                  ; direction
MOVWF   TRISE     ; Set RE1:RE0 as inputs
                  ; RE7:RE2 as outputs
```

FIGURE 9-9: PORTE BLOCK DIAGRAM IN I/O MODE



TRIS OVERRIDE

Pin	Override	Peripheral
RE0	Yes	External Bus
RE1	Yes	External Bus
RE2	Yes	External Bus
RE3	Yes	External Bus
RE4	Yes	External Bus
RE5	Yes	External Bus
RE6	Yes	External Bus
RE7	Yes	External Bus

Note 1: I/O pins have diode protection to VDD and VSS.

15.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

15.1 Master SSP (MSSP) Module Overview

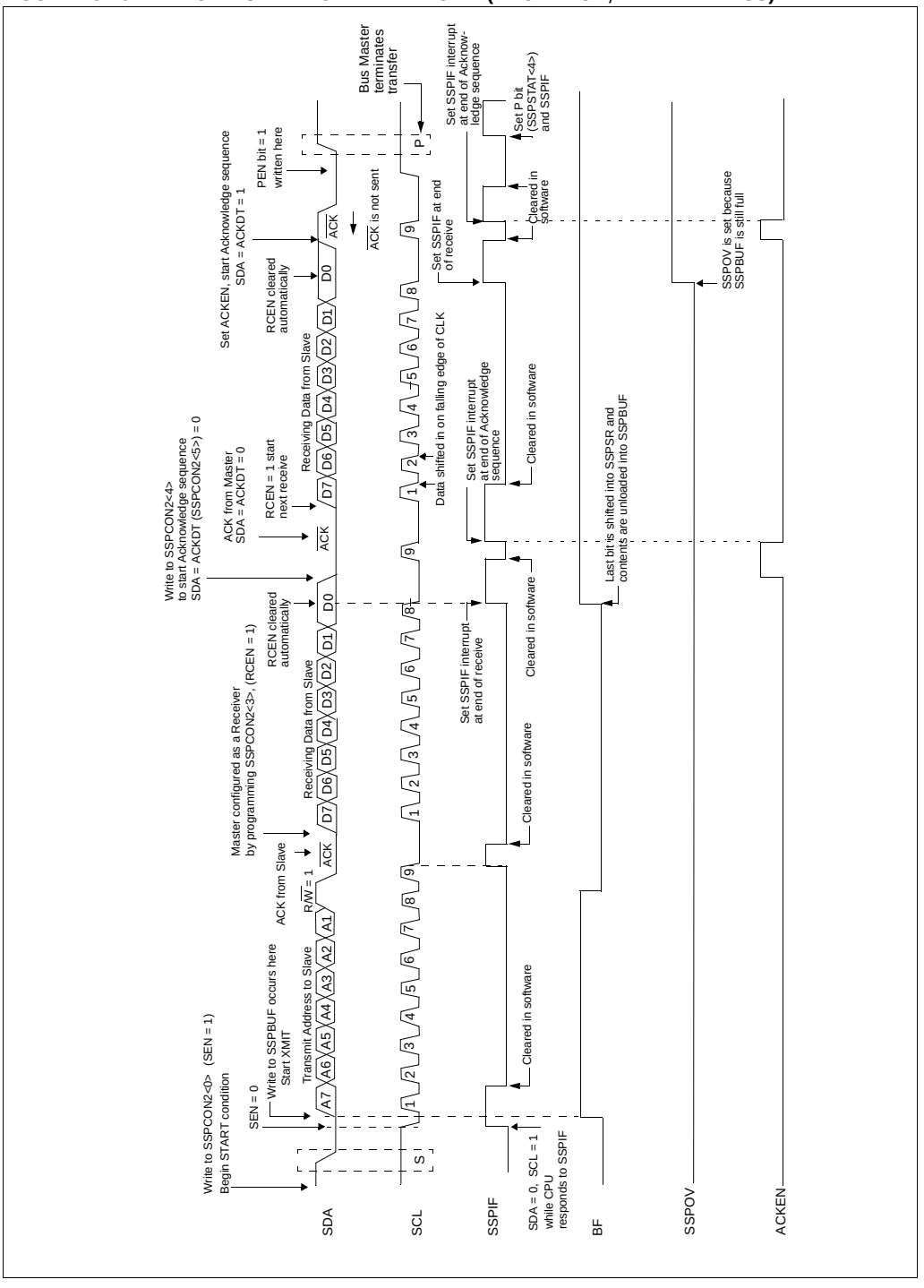
The Master Synchronous Serial Port (MSSP) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be Serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSP module can operate in one of two modes:

- Serial Peripheral Interface™ (SPI)
- Inter-Integrated Circuit™ (I^2C)
  - Full Master mode
  - Slave mode (with general address call)

The I^2C interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode

FIGURE 15-16: I²C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



REGISTER 16-2: RCSTA REGISTER

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

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (Configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled

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

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

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

bit 3 **ADDEN**: Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enable interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit

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

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

bit 0 **RX9D:** 9th bit of Received Data. Can be Address/Data bit or a parity bit.

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

© 2001-2013 Microchip Technology Inc.

17.4 A/D Conversions

Figure 17-3 shows the operation of the A/D converter after the GO bit has been set. Clearing the GO/DONE bit during a conversion will abort the current conversion. The A/D result register pair will NOT be updated with the partially completed A/D conversion sample. That is, the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers). After the A/D conversion is aborted, a 2TAD wait is required before the next acquisition is started. After this 2TAD wait, acquisition on the selected channel is automatically started.

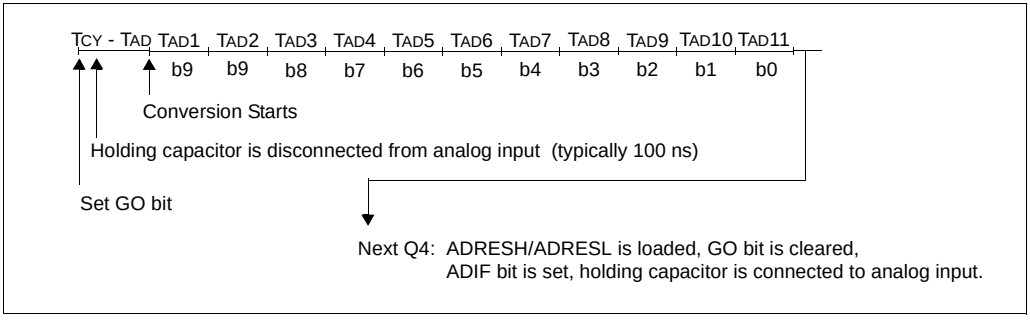
Note:	The GO/DONE bit should **NOT** be set in the same instruction that turns on the A/D.

17.5 Use of the CCP2 Trigger

An A/D conversion can be started by the "special event trigger" of the CCP2 module. This requires that the CCP2M3:CCP2M0 bits (CCP2CON<3:0>) be programmed as 1011, and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/DONE bit will be set, starting the A/D conversion and the Timer1 (or Timer3) counter will be reset to zero. Timer1 (or Timer3) is reset to automatically repeat the A/D acquisition period with minimal software overhead (moving ADRESH/ADRESL to the desired location). The appropriate analog input channel must be selected and the minimum acquisition done before the "special event trigger" sets the GO/DONE bit (starts a conversion).

If the A/D module is not enabled (ADON is cleared), the "special event trigger" will be ignored by the A/D module, but will still reset the Timer1 (or Timer3) counter.

FIGURE 17-3: A/D CONVERSION TAD CYCLES



 © 2001-2013 Microchip Technology Inc.

IORLW **Inclusive OR literal with WREG**

Syntax: [*label*] IORLW k

Operands: $0 \le k \le 255$

Operation: (WREG) .OR. k → WREG

Status Affected: N,Z

Encoding:

0000	1001	kkkk	kkkk

Description: The contents of WREG are OR'ed with the eight bit literal 'k'. The result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `IORLW 35h`

Before Instruction
WREG = 9Ah
N = ?
Z = ?

After Instruction
WREG = 0BFh
N = 1
Z = 0

IORWF **Inclusive OR WREG with f**

Syntax: [*label*] IORWF f [,d [,a]]

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

Operation: (WREG) .OR. (f) → dest

Status Affected: N,Z

Encoding:

0001	00da	ffff	ffff

Description: Inclusive OR WREG with register 'f'. If 'd' is 0, the result is placed in WREG. 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' is 1, the Bank will be selected as per the BSR value.

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 = 13h
WREG = 91h
N = ?
Z = ?

After Instruction
RESULT = 13h
WREG = 93h
N = 1
Z = 0

 © 2001-2013 Microchip Technology Inc.

NEGF **Negate f**

Syntax: [*label*] NEGF f [,a]

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

Operation: $(\overline{f}) + 1 \rightarrow f$

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

Encoding:

0110	110a	ffff	ffff

Description: Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `NEGF REG`

Before Instruction
REG = 0011 1010 [3Ah]
N = ?
OV = ?
C = ?
DC = ?
Z = ?

After Instruction
REG = 1100 0110 [0C6h]
N = 1
OV = 0
C = 0
DC = 0
Z = 0

NOP **No Operation**

Syntax: [*label*] NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding:

0000 1111	0000 xxxx	0000 xxxx	0000 xxxx

Description: No operation.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

Example:

None.

SUBWFB **Subtract WREG from f with Borrow**

Syntax: [*label*] SUBWFB f [,d [,a]]

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

Operation: (f) – (WREG) – ($\overline{C}$) → dest

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

Encoding:

0101	10da	ffff	ffff

Description: Subtract WREG and the carry flag (borrow) from register 'f' (2's complement method). If 'd' is 0, the result is stored in WREG. 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, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

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

SUBWFB (Cont.)

Example 1: `SUBWFB REG`

Before Instruction
REG = 19h (0001 1001)
WREG = 0Dh (0000 1101)
C = 1

After Instruction
REG = 0Ch (0000 1011)
WREG = 0Dh (0000 1101)
C = 1
Z = 0
N = 0 ; result is positive

Example 2: `SUBWFB REG, W`

Before Instruction
REG = 1Bh (0001 1011)
WREG = 1Ah (0001 1010)
C = 0

After Instruction
REG = 1Bh (0001 1011)
WREG = 00h
C = 1
Z = 1 ; result is zero
N = 0

Example 3: `SUBWFB REG`

Before Instruction
REG = 03h (0000 0011)
WREG = 0Eh (0000 1101)
C = 1

After Instruction
REG = 0F5h (1111 0100) [2's comp]
WREG = 0Eh (0000 1101)
C = 0
Z = 0
N = 1 ; result is negative

FIGURE 22-1: PIC18C601/801 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)

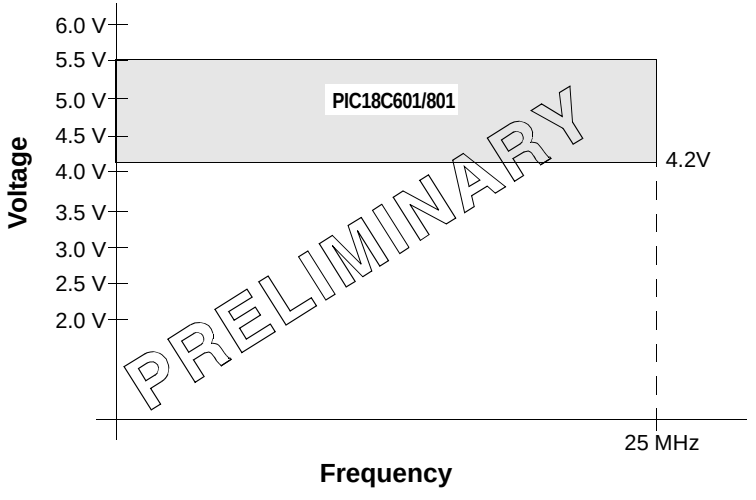
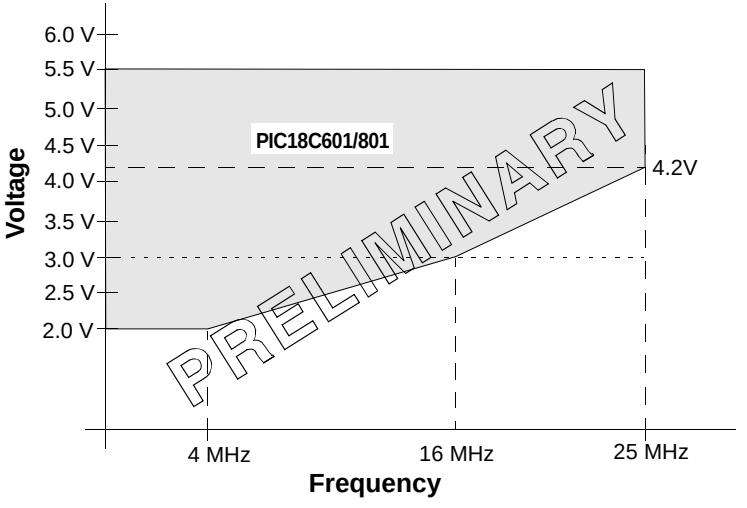


FIGURE 22-2: PIC18C601/801 VOLTAGE-FREQUENCY GRAPH (EXTENDED)



FMAX = (12.0 MHz/V) (VDDAPPMIN - 2.0 V) + 4 MHz where VDDAPPMIN ≤ 3
FMAX = (7.5 MHz/V) (VDDAPPMIN - 3.0 V) + 16 MHz where VDDAPPMIN > 3
Note: VDDAPP is the minimum voltage of the PIC® device in the application.

 © 2001-2013 Microchip Technology Inc.

22.3 AC (Timing) Characteristics

22.3.1 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	T1CKI
mc	$\overline{MCLR}$	wr	$\overline{WR}$

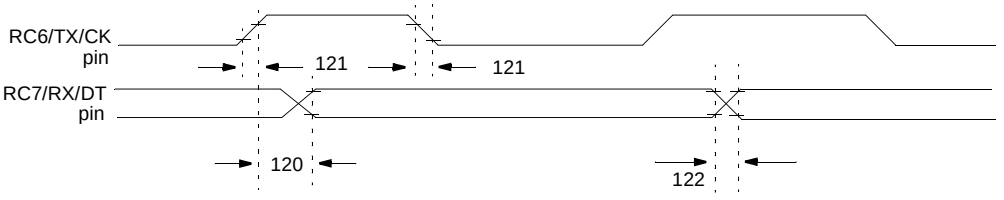
Uppercase letters and their meanings:

S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (Hi-impedance)	V	Valid
L	Low	Z	Hi-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		

FIGURE 22-21: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING



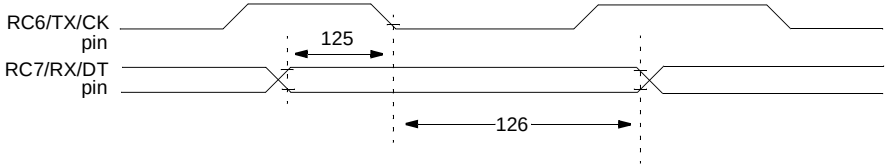
Note: Refer to Figure 22-4 for load conditions.

TABLE 22-20: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
120	TckH2dtV	SYNC XMIT (Master & Slave) Clock high to data-out valid	PIC18**C**601/801	—	40	ns	
			PIC18**LC**601/801	—	100	ns	
121	Tckrf	Clock out rise time and fall time (Master mode)	PIC18**C**601/801	—	20	ns	
			PIC18**LC**601/801	—	50	ns	
122	Tdtrf	Data-out rise time and fall time	PIC18**C**601/801	—	20	ns	
			PIC18**LC**601/801	—	50	ns	

PRELIMINARY

FIGURE 22-22: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING



Note: Refer to Figure 22-4 for load conditions.

TABLE 22-21: USART SYNCHRONOUS RECEIVE REQUIREMENTS

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
125	TdtV2ckl	SYNC RCV (Master & Slave) Data-hold before CK ↓ (DT hold time)	10	—	ns	
126	TckL2dtl	Data-hold after CK ↓ (DT hold time)	15	—	ns	

PRELIMINARY

APPENDIX E: DEVELOPMENT TOOL VERSION REQUIREMENTS

This lists the minimum requirements (software/ firmware) of the specified development tool to support the devices listed in this data sheet.

MPLAB® IDE: TBD

MPLAB® SIMULATOR: TBD

MPLAB® ICE 3000:

PIC18C601/801 Processor Module:
Part Number - TBD

PIC18C601/801 Device Adapter:

Socket	Part Number
64-pin TQFP	TBD
68-pin PLCC	TBD
80-pin TQFP	TBD
84-pin PLCC	TBD

MPLAB® ICD: TBD

PRO MATE® II: TBD

PICSTART® Plus: TBD

MPASM™ Assembler: TBD

MPLAB® C18 C Compiler: TBD

Note:	Please read all associated README.TXT files that are supplied with the development tools. These "read me" files will discuss product support and any known limitations.