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	16MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	50
Program Memory Size	32KB (16K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	902 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	68-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.23x24.23)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c756a-16-l

TABLE 5-4: INITIALIZATION CONDITIONS FOR SPECIAL FUNCTION REGISTERS (CONTINUED)

Register	Address	Power-on Reset Brown-out Reset	$\overline{\text{MCLR}}$ Reset WDT Reset	Wake-up from SLEEP through Interrupt
Bank 7				
PW3DCL	10h	xx0- ----	uu0- ----	uuu- ----
PW3DCH	11h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA3L	12h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA3H	13h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA4L	14h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA4H	15h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TCON3	16h	-000 0000	-000 0000	-uuu uuuu
Unimplemented	17h	---- ----	---- ----	---- ----
Bank 8				
DDRH	10h	1111 1111	1111 1111	uuuu uuuu
PORTH(4)	11h	xxxx xxxx	uuuu uuuu	uuuu uuuu
DDRJ	12h	1111 1111	1111 1111	uuuu uuuu
PORTJ(4)	13h	xxxx xxxx	uuuu uuuu	uuuu uuuu
Unbanked				
PRODL	18h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODH	19h	xxxx xxxx	uuuu uuuu	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented, read as '0', q = value depends on condition

Note 1: One or more bits in INTSTA, PIR1, PIR2 will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GLINTD bit is cleared, the PC is loaded with the interrupt vector.

3: See Table 5-3 for RESET value of specific condition.

4: This is the value that will be in the port output latch.

5: When the device is configured for Microprocessor or Extended Microcontroller mode, the operation of this port does not rely on these registers.

6: On any device RESET, these pins are configured as inputs.

10.0 I/O PORTS

PIC17C75X devices have seven I/O ports, PORTA through PORTG. PIC17C76X devices have nine I/O ports, PORTA through PORTJ. PORTB through PORTJ have a corresponding Data Direction Register (DDR), which is used to configure the port pins as inputs or outputs. Some of these ports pins are multiplexed with alternate functions.

PORTC, PORTD, and PORTE are multiplexed with the system bus. These pins are configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, these pins are general purpose I/O.

PORTA, PORTB, PORTE<3>, PORTF, PORTG and the upper four bits of PORTH are multiplexed with the peripheral features of the device. These peripheral features are:

- Timer Modules
- Capture Modules
- PWM Modules
- USART/SCI Modules
- SSP Module
- A/D Module
- External Interrupt pin

When some of these peripheral modules are turned on, the port pin will automatically configure to the alternate function. The modules that do this are:

- PWM Module
- SSP Module
- USART/SCI Module

When a pin is automatically configured as an output by a peripheral module, the pins data direction (DDR) bit is unknown. After disabling the peripheral module, the user should re-initialize the DDR bit to the desired configuration.

The other peripheral modules (which require an input) must have their data direction bits configured appropriately.

Note:	A pin that is a peripheral input, can be configured as an output (DDRx<y> is cleared). The peripheral events will be determined by the action output on the port pin.

When the device enters the "RESET state", the Data Direction registers (DDR) are forced set, which will make the I/O hi-impedance inputs. The RESET state of some peripheral modules may force the I/O to other operations, such as analog inputs or the system bus.

13.1.3.1 PWM Periods

The period of the PWM1 output is determined by Timer1 and its period register (PR1). The period of the PWM2 and PWM3 outputs can be individually software configured to use either Timer1 or Timer2 as the time-base. For PWM2, when TM2PW2 bit (PW2DCL<5>) is clear, the time base is determined by TMR1 and PR1 and when TM2PW2 is set, the time base is determined by Timer2 and PR2. For PWM3, when TM2PW3 bit (PW3DCL<5>) is clear, the time base is determined by TMR1 and PR1, and when TM2PW3 is set, the time base is determined by Timer2 and PR2.

Running two different PWM outputs on two different timers allows different PWM periods. Running all PWMs from Timer1 allows the best use of resources by freeing Timer2 to operate as an 8-bit timer. Timer1 and Timer2 cannot be used as a 16-bit timer if any PWM is being used.

The PWM periods can be calculated as follows:

$$\text{period of PWM1} = [(PR1) + 1] \times 4T_{OSC}$$

$$\text{period of PWM2} = [(PR1) + 1] \times 4T_{OSC} \text{ or } [(PR2) + 1] \times 4T_{OSC}$$

$$\text{period of PWM3} = [(PR1) + 1] \times 4T_{OSC} \text{ or } [(PR2) + 1] \times 4T_{OSC}$$

The duty cycle of PWMx is determined by the 10-bit value DCx<9:0>. The upper 8-bits are from register PWxDCH and the lower 2-bits are from PWxDCL<7:6> (PWxDCH:PWxDCL<7:6>). Table 13-4 shows the maximum PWM frequency (F_{PWM}), given the value in the period register.

The number of bits of resolution that the PWM can achieve depends on the operation frequency of the device as well as the PWM frequency (F_{PWM}).

Maximum PWM resolution (bits) for a given PWM frequency:

$$= \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

where: F_{PWM} = 1 / period of PWM

The PWMx duty cycle is as follows:

$$\text{PWMx Duty Cycle} = (DCx) \times T_{OSC}$$

where DCx represents the 10-bit value from PWxDCH:PWxDCL.

If DCx = 0, then the duty cycle is zero. If PRx = PWxDCH, then the PWM output will be low for one to four Q-clocks (depending on the state of the PWxDCL<7:6> bits). For a duty cycle to be 100%, the PWxDCH value must be greater then the PRx value.

The duty cycle registers for both PWM outputs are double buffered. When the user writes to these registers, they are stored in master latches. When TMR1 (or TMR2) overflows and a new PWM period begins, the master latch values are transferred to the slave latches and the PWMx pin is forced high.

Note:	For PW1DCH, PW1DCL, PW2DCH, PW2DCL, PW3DCH and PW3DCL registers, a write operation writes to the "master latches", while a read operation reads the "slave latches". As a result, the user may not read back what was just written to the duty cycle registers (until transferred to slave latch).

The user should also avoid any "read-modify-write" operations on the duty cycle registers, such as: `ADDWF PW1DCH`. This may cause duty cycle outputs that are unpredictable.

TABLE 13-4: PWM FREQUENCY vs. RESOLUTION AT 33 MHz

PWM Frequency	Frequency (kHz)				
	32.2	**64.5**	**90.66**	**128.9**	**515.6**
PRx Value	0xFF	0x7F	0x5A	0x3F	0x0F
High Resolution	10-bit	9-bit	8.5-bit	8-bit	6-bit
Standard Resolution	8-bit	7-bit	6.5-bit	6-bit	4-bit

13.1.3.2 PWM INTERRUPTS

The PWM modules make use of the TMR1 and/or TMR2 interrupts. A timer interrupt is generated when TMR1 or TMR2 equals its period register and on the following increment is cleared to zero. This interrupt also marks the beginning of a PWM cycle. The user can write new duty cycle values before the timer rollover. The TMR1 interrupt is latched into the TMR1IF bit and the TMR2 interrupt is latched into the TMR2IF bit. These flags must be cleared in software.

13.2.4 EXTERNAL CLOCK INPUT FOR TIMER3

When TMR3CS is set, the 16-bit TMR3 increments on the falling edge of clock input TCLK3. The input on the RB5/TCLK3 pin is sampled and synchronized by the internal phase clocks, twice every instruction cycle. This causes a delay from the time a falling edge appears on TCLK3 to the time TMR3 is actually incremented. For the external clock input timing requirements, see the Electrical Specification section. Figure 13-7 shows the timing diagram when operating from an external clock.

13.2.5 READING/WRITING TIMER3

Since Timer3 is a 16-bit timer and only 8-bits at a time can be read or written, care should be taken when reading or writing while the timer is running. The best method is to stop the timer, perform any read or write operation and then restart Timer3 (using the TMR3ON bit). However, if it is necessary to keep Timer3 free-running, care must be taken. For writing to the 16-bit TMR3, Example 13-2 may be used. For reading the 16-bit TMR3, Example 13-3 may be used. Interrupts must be disabled during this routine.

EXAMPLE 13-2: WRITING TO TMR3

```
    BSF     CPUSTA, GLINTD      ; Disable interrupts
    MOVFP   RAM_L,  TMR3L       ;
    MOVFP   RAM_H,  TMR3H       ;
    BCF     CPUSTA, GLINTD      ; Done, enable interrupts
```

EXAMPLE 13-3: READING FROM TMR3

```
    MOVPF   TMR3L, TMPLO        ; read low TMR3
    MOVPF   TMR3H, TMPHI        ; read high TMR3
    MOVFP   TMPLO, WREG         ; tmplo –> wreg
    CPFSLT  TMR3L               ; TMR3L < wreg?
    RETURN                      ; no then return
    MOVPF   TMR3L, TMPLO        ; read low TMR3
    MOVPF   TMR3H, TMPHI        ; read high TMR3
    RETURN                      ; return
```

FIGURE 13-7: TIMER1, TIMER2 AND TIMER3 OPERATION (IN COUNTER MODE)

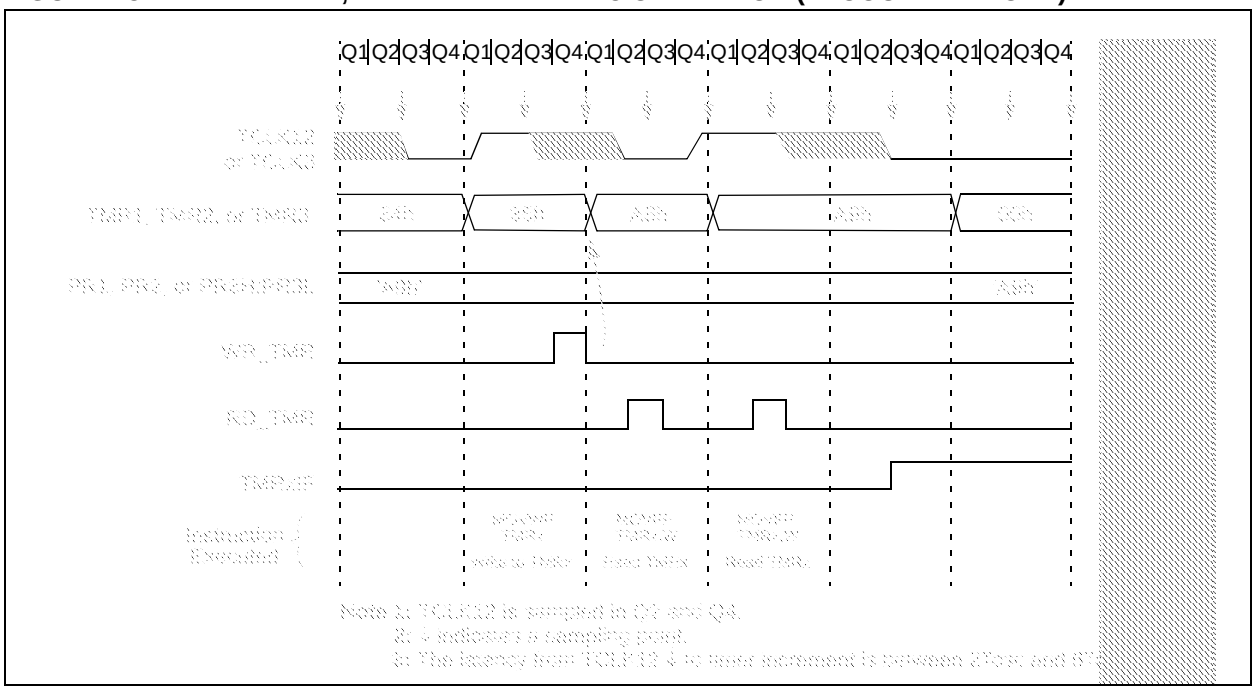


TABLE 14-4: BAUD RATES FOR SYNCHRONOUS MODE

BAUD RATE (K)	FOSC = 33 MHz			FOSC = 25 MHz			FOSC = 20 MHz			FOSC = 16 MHz		
	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)
0.3	NA	—	—	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	NA	—	—	NA	—	—
9.6	NA	—	—	NA	—	—	NA	—	—	NA	—	—
19.2	NA	—	—	NA	—	—	19.53	+1.73	255	19.23	+0.16	207
76.8	77.10	+0.39	106	77.16	+0.47	80	76.92	+0.16	64	76.92	+0.16	51
96	95.93	-0.07	85	96.15	+0.16	64	96.15	+0.16	51	95.24	-0.79	41
300	294.64	-1.79	27	297.62	-0.79	20	294.1	-1.96	16	307.69	+2.56	12
500	485.29	-2.94	16	480.77	-3.85	12	500	0	9	500	0	7
HIGH	8250	—	0	6250	—	0	5000	—	0	4000	—	0
LOW	32.22	—	255	24.41	—	255	19.53	—	255	15.625	—	255

BAUD RATE (K)	FOSC = 10 MHz			FOSC = 7.159 MHz			FOSC = 5.068 MHz		
	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)
0.3	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	NA	—	—
9.6	9.766	+1.73	255	9.622	+0.23	185	9.6	0	131
19.2	19.23	+0.16	129	19.24	+0.23	92	19.2	0	65
76.8	75.76	-1.36	32	77.82	+1.32	22	79.2	+3.13	15
96	96.15	+0.16	25	94.20	-1.88	18	97.48	+1.54	12
300	312.5	+4.17	7	298.3	-0.57	5	316.8	+5.60	3
500	500	0	4	NA	—	—	NA	—	—
HIGH	2500	—	0	1789.8	—	0	1267	—	0
LOW	9.766	—	255	6.991	—	255	4.950	—	255

BAUD RATE (K)	FOSC = 3.579 MHz			FOSC = 1 MHz			FOSC = 32.768 kHz		
	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)
0.3	NA	—	—	NA	—	—	0.303	+1.14	26
1.2	NA	—	—	1.202	+0.16	207	1.170	-2.48	6
2.4	NA	—	—	2.404	+0.16	103	NA	—	—
9.6	9.622	+0.23	92	9.615	+0.16	25	NA	—	—
19.2	19.04	-0.83	46	19.24	+0.16	12	NA	—	—
76.8	74.57	-2.90	11	83.34	+8.51	2	NA	—	—
96	99.43	_3.57	8	NA	—	—	NA	—	—
300	298.3	-0.57	2	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	894.9	—	0	250	—	0	8.192	—	0
LOW	3.496	—	255	0.976	—	255	0.032	—	255

REGISTER 15-3: SSPCON2: SYNC SERIAL PORT CONTROL REGISTER2 (ADDRESS 12h, BANK 6)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

bit 7 **GCEN**: General Call Enable bit (in I^2C Slave mode only)
1 = Enable interrupt when a general call address (0000h) is received in the SSPSR
0 = General call address disabled

bit 6 **ACKSTAT**: Acknowledge Status bit (in I^2C Master mode only)

In Master Transmit mode:
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT**: Acknowledge Data bit (in I^2C Master mode only)

In Master Receive mode:
Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN**: Acknowledge Sequence Enable bit (in I^2C Master mode only)

In Master Receive mode:
1 = Initiate Acknowledge sequence on SDA and SCL pins and transmit AKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence idle

Note: If the I^2C module is not in the IDLE mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

bit 3 **RCEN**: Receive Enable bit (in I^2C Master mode only)
1 = Enables Receive mode for I^2C
0 = Receive idle

Note: If the I^2C module is not in the IDLE mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

bit 2 **PEN**: STOP Condition Enable bit (in I^2C Master mode only)

SCK Release Control:
1 = Initiate STOP condition on SDA and SCL pins. Automatically cleared by hardware.
0 = STOP condition idle

Note: If the I^2C module is not in the IDLE mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

bit 1 **RSEN**: Repeated Start Condition Enabled bit (in I^2C Master mode only)
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition idle

Note: If the I^2C module is not in the IDLE mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

bit 0 **SEN**: START Condition Enabled bit (In I^2C Master mode only)
1 = Initiate START condition on SDA and SCL pins. Automatically cleared by hardware.
0 = START condition idle.

Note: If the I^2C module is not in the IDLE mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

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

15.1.4 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave (Processor 2, Figure 15-5) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a “Line Activity Monitor” mode.

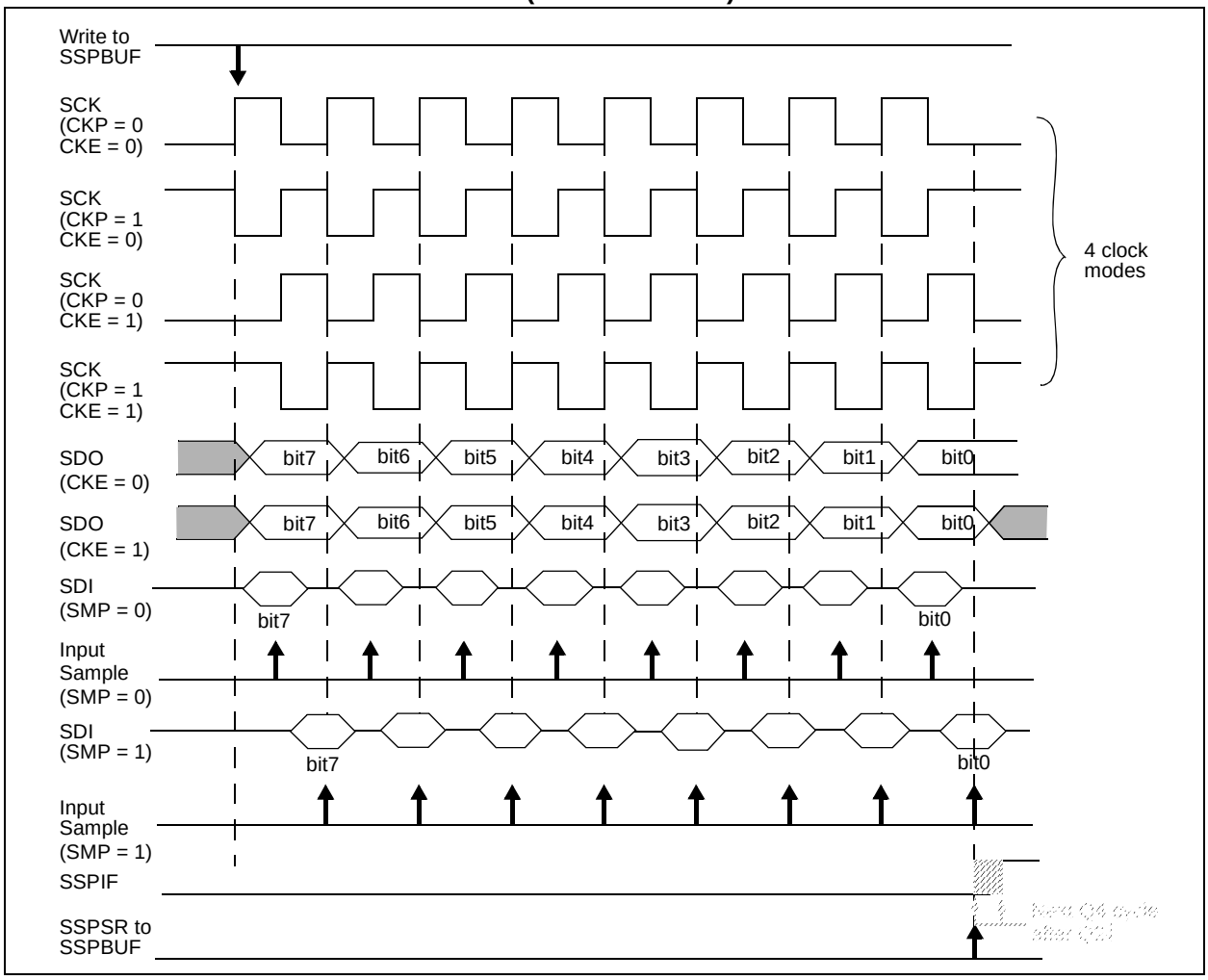
The clock polarity is selected by appropriately programming bit CKP (SSPCON1<4>). This then, would give waveforms for SPI communication as shown in Figure 15-6, Figure 15-8 and Figure 15-9, where the MSb is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- FOSC/4 (or TCY)
- FOSC/16 (or 4 • TCY)
- FOSC/64 (or 16 • TCY)
- Timer2 output/2

This allows a maximum bit clock frequency (at 33 MHz) of 8.25 MHz.

Figure 15-6 shows the waveforms for Master mode. When CKE = 1, the SDO data is valid before there is a clock edge on SCK. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPBUF is loaded with the received data is shown.

FIGURE 15-6: SPI MODE WAVEFORM (MASTER MODE)



EXAMPLE 15-2: INTERFACING TO A 24LC01B SERIAL EEPROM (USING MPLAB C17)

```
// Writes the byte data to 24LC01B at the specified address
void ByteWrite(static unsigned char address, static unsigned char data)
{
    StartI2C();                      // Send start bit
    IdleI2C();                       // Wait for idle condition
    WriteI2C(CONTROL);               // Send control byte
    IdleI2C();                       // Wait for idle condition
    if (!SSPCON2bits.ACKSTAT)        // If 24LC01B ACKs
    {
        WriteI2C(address);           // Send control byte
        IdleI2C();                   // Wait for idle condition

        if (!SSPCON2bits.ACKSTAT)    // If 24LC01B ACKs
            WriteI2C(data);          // Send data
    }
    IdleI2C();                       // Wait for idle condition
    StopI2C();                       // Send stop bit
    IdleI2C();                       // Wait for idle condition
    return;
}


// Reads a byte of data from 24LC01B at the specified address
unsigned char ByteRead(static unsigned char address)
{
    StartI2C();                      // Send start bit
    IdleI2C();                       // Wait for idle condition
    WriteI2C(CONTROL);               // Send control byte
    IdleI2C();                       // Wait for idle condition
    if (!SSPCON2bits.ACKSTAT)        // If the 24LC01B ACKs
    {
        WriteI2C(address);           // Send address
        IdleI2C();                   // Wait for idle condition
        if (!SSPCON2bits.ACKSTAT)    // If the 24LC01B ACKs
        {
            RestartI2C();            // Send restart
            IdleI2C();               // Wait for idle condition
            WriteI2C(CONTROL+1);     // Send control byte with R/W set
            IdleI2C();               // Wait for idle condition
            if (!SSPCON2bits.ACKSTAT)    // If the 24LC01B ACKs
            {
                getcI2C();           // Read a byte of data from 24LC01B
                IdleI2C();           // Wait for idle condition
                NotAckI2C();         // Send a NACK to 24LC01B
                IdleI2C();           // Wait for idle condition
                StopI2C();           // Send stop bit
                IdleI2C();           // Wait for idle condition
             }
        }
    }
    return(SSPBUF);
}
```

16.4.1 A/D RESULT REGISTERS

The ADRESH:ADRESL register pair is the location where the 10-bit A/D result is loaded at the completion of the A/D conversion. This register pair is 16-bits wide. The A/D module gives the flexibility to left or right justify the 10-bit result in the 16-bit result register. The A/D Format Select bit (ADFM) controls this justification. Figure 16-6 shows the operation of the A/D result justification. The extra bits are loaded with '0's'. When an A/D result will not overwrite these locations (A/D disable), these registers may be used as two general purpose 8-bit registers.

16.5 A/D Operation During SLEEP

The A/D module can operate during SLEEP mode. This requires that the A/D clock source be set to RC (ADCS1:ADCS0 = 11). When the RC clock source is selected, the A/D module waits one instruction cycle before starting the conversion. This allows the SLEEP instruction to be executed, which eliminates all digital switching noise from the conversion. When the conversion is completed, the GO/DONE bit will be cleared, and the result loaded into the ADRES register. If the A/D interrupt is enabled, the device will wake-up from SLEEP. If the A/D interrupt is not enabled, the A/D module will then be turned off, although the ADON bit will remain set.

When the A/D clock source is another clock option (not RC), a SLEEP instruction will cause the present conversion to be aborted and the A/D module to be turned off, though the ADON bit will remain set.

Turning off the A/D places the A/D module in its lowest current consumption state.

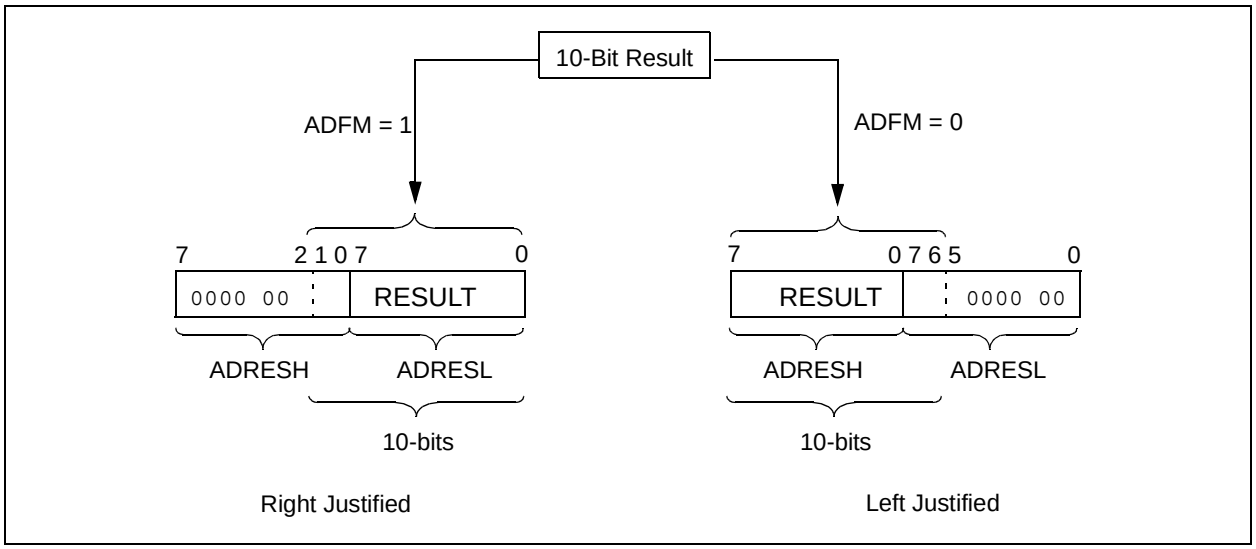
Note:	For the A/D module to operate in SLEEP, the A/D clock source must be set to RC (ADCS1:ADCS0 = 11). To allow the conversion to occur during SLEEP, ensure the SLEEP instruction immediately follows the instruction that sets the GO/DONE bit.

16.6 Effects of a RESET

A device RESET forces all registers to their RESET state. This forces the A/D module to be turned off, and any conversion is aborted.

The value that is in the ADRESH:ADRESL registers is not modified for a Power-on Reset. The ADRESH:ADRESL registers will contain unknown data after a Power-on Reset.

FIGURE 16-6: A/D RESULT JUSTIFICATION



CALL **Subroutine Call**

Syntax: [*label*] CALL k

Operands: $0 \le k \le 8191$

Operation: PC+ 1→ TOS, k → PC<12:0>, k<12:8> → PCLATH<4:0>; PC<15:13> → PCLATH<7:5>

Status Affected: None

Encoding:

111k	kkkk	kkkk	kkkk

Description: Subroutine call within 8K page. First, return address (PC+1) is pushed onto the stack. The 13-bit value is loaded into PC bits<12:0>. Then the upper-eight bits of the PC are copied into PCLATH. CALL is a two-cycle instruction.

See LCALL for calls outside 8K memory space.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>, Push PC to stack	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: HERE CALL THERE

Before Instruction
PC = Address (HERE)

After Instruction
PC = Address (THERE)
TOS = Address (HERE + 1)

CLRF **Clear f**

Syntax: [*label*] CLRF f,s

Operands: $0 \le f \le 255$

Operation: 00h → f, $s \in [0,1]$
00h → dest

Status Affected: None

Encoding:

0010	100s	ffff	ffff

Description: Clears the contents of the specified register(s).
s = 0: Data memory location 'f' and WREG are cleared.
s = 1: Data memory location 'f' is cleared.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f' and if specified WREG

Example: CLRF FLAG_REG, 1

Before Instruction
FLAG_REG = 0x5A
WREG = 0x01

After Instruction
FLAG_REG = 0x00
WREG = 0x01

INCF **Increment f**

Syntax: [*label*] INCF f,d

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

Operation: (f) + 1 → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0001	010d	ffff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `INCF CNT, 1`

Before Instruction
CNT = 0xFF
Z = 0
C = ?

After Instruction
CNT = 0x00
Z = 1
C = 1

INCFSZ **Increment f, skip if 0**

Syntax: [*label*] INCFSZ f,d

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

Operation: (f) + 1 → (dest)
skip if result = 0

Status Affected: None

Encoding:

0001	111d	ffff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f'.
If the result is 0, the next instruction, which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

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

If skip:

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

Example:
```
HERE    INCFSZ  CNT, 1
NZERO   :
ZERO    :
```

Before Instruction
PC = Address (HERE)

After Instruction
CNT = CNT + 1
If CNT = 0;
PC = Address (ZERO)
If CNT ≠ 0;
PC = Address (NZERO)

SUBWF **Subtract WREG from f**

Syntax: [*label*] SUBWF f,d

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

Operation: (f) – (W) → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0000	010d	ffff	ffff

Description: Subtract WREG 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'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1: `SUBWF REG1, 1`

Before Instruction
REG1 = 3
WREG = 2
C = ?

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

Example 2:

Before Instruction
REG1 = 2
WREG = 2
C = ?

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

Example 3:

Before Instruction
REG1 = 1
WREG = 2
C = ?

After Instruction
REG1 = FF
WREG = 2
C = 0 ; result is negative
Z = 0

SUBWFB **Subtract WREG from f with Borrow**

Syntax: [*label*] SUBWFB f,d

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

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

Status Affected: OV, C, DC, Z

Encoding:

0000	001d	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'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1: `SUBWFB REG1, 1`

Before Instruction
REG1 = 0x19 (0001 1001)
WREG = 0x0D (0000 1101)
C = 1

After Instruction
REG1 = 0x0C (0000 1011)
WREG = 0x0D (0000 1101)
C = 1 ; result is positive
Z = 0

Example2: `SUBWFB REG1,0`

Before Instruction
REG1 = 0x1B (0001 1011)
WREG = 0x1A (0001 1010)
C = 0

After Instruction
REG1 = 0x1B (0001 1011)
WREG = 0x00
C = 1 ; result is zero
Z = 1

Example3: `SUBWFB REG1,1`

Before Instruction
REG1 = 0x03 (0000 0011)
WREG = 0x0E (0000 1101)
C = 1

After Instruction
REG1 = 0xF5 (1111 0100) [2's comp]
WREG = 0x0E (0000 1101)
C = 0 ; result is negative
Z = 0

TABLRD **Table Read**

Example1: `TABLRD 1, 1, REG ;`

Before Instruction

REG	=	0x53
TBLATH	=	0xAA
TBLATL	=	0x55
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0x1234

After Instruction (table write completion)

REG	=	0xAA
TBLATH	=	0x12
TBLATL	=	0x34
TBLPTR	=	0xA357
MEMORY(TBLPTR)	=	0x5678

Example2: `TABLRD 0, 0, REG ;`

Before Instruction

REG	=	0x53
TBLATH	=	0xAA
TBLATL	=	0x55
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0x1234

After Instruction (table write completion)

REG	=	0x55
TBLATH	=	0x12
TBLATL	=	0x34
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0x1234

TABLWT **Table Write**

Syntax: [*label*] TABLWT t,i,f

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

Operation: If t = 0,
f → TBLATL;
If t = 1,
f → TBLATH;
TBLAT → Prog Mem (TBLPTR);
If i = 1,
TBLPTR + 1 → TBLPTR
If i = 0,
TBLPTR is unchanged

Status Affected: None

Encoding:

1010	11ti	ffff	ffff

Description:

1. Load value in 'f' into 16-bit table latch (TBLAT)
   If t = 1: load into high byte;
   If t = 0: load into low byte
2. The contents of TBLAT are written to the program memory location pointed to by TBLPTR.
   If TBLPTR points to external program memory location, then the instruction takes two-cycle.
   If TBLPTR points to an internal EPROM location, then the instruction is terminated when an interrupt is received.

Note: The $\overline{MCLR}/V_{PP}$ pin must be at the programming voltage for successful programming of internal memory.
If $\overline{MCLR}/V_{PP} = V_{DD}$
the programming sequence of internal memory will be interrupted. A short write will occur (2 T_{CY}). The internal memory location will not be affected.

3. The TBLPTR can be automatically incremented
   If i = 1; TBLPTR is not incremented
   If i = 0; TBLPTR is incremented

Words: 1

Cycles: 2 (many if write is to on-chip EPROM program memory)

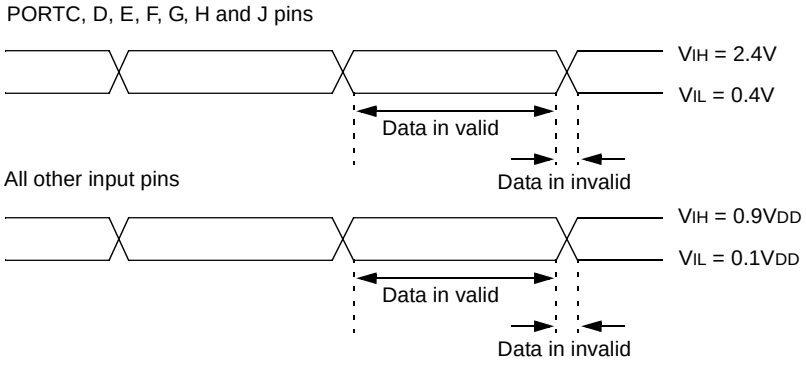
Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register TBLATH or TBLATL
No operation	No operation (Table Pointer on Address bus)	No operation	No operation (Table Latch on Address bus, $\overline{WR}$ goes low)

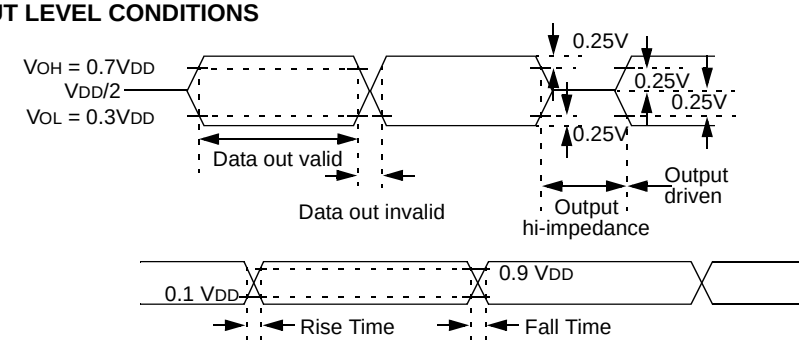
FIGURE 20-5: PARAMETER MEASUREMENT INFORMATION

All timings are measured between high and low measurement points as indicated below.

INPUT LEVEL CONDITIONS

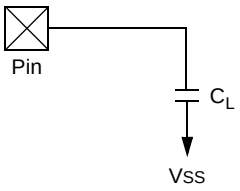


OUTPUT LEVEL CONDITIONS



LOAD CONDITIONS

Load Condition 1



50 pF $\leq C_L$

FIGURE 20-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER, AND BROWN-OUT RESET TIMING

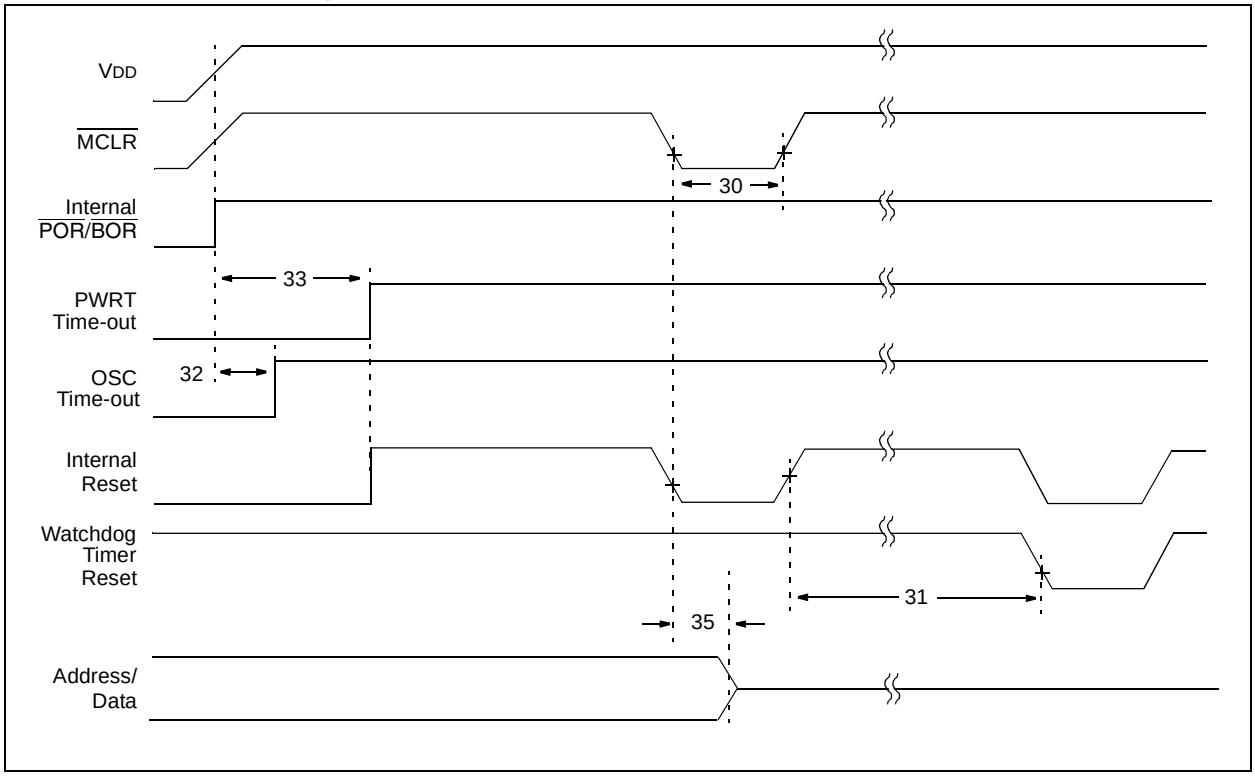


TABLE 20-3: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER, AND BROWN-OUT RESET REQUIREMENTS

Param. No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
30	TmcL	$\overline{\text{MCLR}}$ Pulse Width (low)		100	—	—	ns	VDD = 5V
31	TWDT	Watchdog Timer Time-out Period (Postscale = 1)		5	12	25	ms	VDD = 5V
32	TOST	Oscillation Start-up Timer Period		—	1024TOSC	—	ms	TOSC = OSC1 period
33	TPWRT	Power-up Timer Period		40	96	200	ms	VDD = 5V
34	TIOZ	$\overline{\text{MCLR}}$ to I/O hi-impedance		100	—	—	ns	Depends on pin load
35	TmcL2adI	$\overline{\text{MCLR}}$ to System Interface bus (AD15:AD0>) invalid	PIC17**C**7XX	—	—	100	ns	
			PIC17**LC**7XX	—	—	120	ns	
36	TBOR	Brown-out Reset Pulse Width (low)		100	—	—	ns	VDD within VBOR limits (parameter D005)

† Data in “Typ” column is at 5V, 25°C unless otherwise stated.

.

FIGURE 20-25: MEMORY INTERFACE READ TIMING

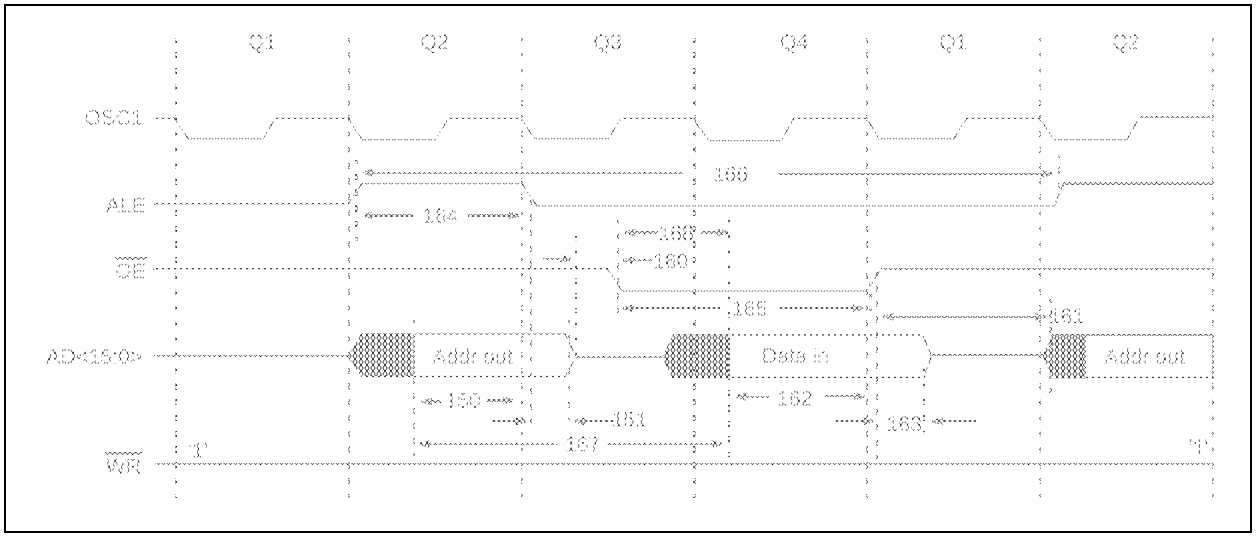


TABLE 20-21: MEMORY INTERFACE READ REQUIREMENTS

Param. No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
150	TadV2alL	AD15:AD0 (address) valid to ALE↓ (address setup time)	PIC17**C**XXX	0.25TCY - 10	—	—	ns	
			PIC17**LC**XXX	0.25TCY - 10	—	—		
151	TalL2adI	ALE↓ to address out invalid (address hold time)	PIC17**C**XXX	5	—	—	ns	
			PIC17**LC**XXX	5	—	—		
160	TadZ2oeL	AD15:AD0 hi-impedance to $\overline{OE}$↓	PIC17**C**XXX	0	—	—	ns	
			PIC17**LC**XXX	0	—	—		
161	ToeH2adD	$\overline{OE}$↑ to AD15:AD0 driven	PIC17**C**XXX	0.25TCY - 15	—	—	ns	
			PIC17**LC**XXX	0.25TCY - 15	—	—		
162	TadV2oeH	Data in valid before $\overline{OE}$↑ (data setup time)	PIC17**C**XXX	35	—	—	ns	
			PIC17**LC**XXX	45	—	—		
163	ToeH2adI	$\overline{OE}$↑to data in invalid (data hold time)	PIC17**C**XXX	0	—	—	ns	
			PIC17**LC**XXX	0	—	—		
164	TalH	ALE pulse width	PIC17**C**XXX	—	0.25TCY	—	ns	
			PIC17**LC**XXX	—	0.25TCY	—		
165	ToeL	$\overline{OE}$ pulse width	PIC17**C**XXX	0.5TCY - 35	—	—	ns	
			PIC17**LC**XXX	0.5TCY - 35	—	—		
166	TalH2alH	ALE↑ to ALE↑(cycle time)	PIC17**C**XXX	—	TCY	—	ns	
			PIC17**LC**XXX	—	TCY	—		
167	Tacc	Address access time	PIC17**C**XXX	—	—	0.75TCY - 30	ns	
			PIC17**LC**XXX	—	—	0.75TCY - 45		
168	Toe	Output enable access time ($\overline{OE}$ low to data valid)	PIC17**C**XXX	—	—	0.5TCY - 45	ns	
			PIC17**LC**XXX	—	—	0.5TCY - 75		

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

FIGURE 21-4: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD

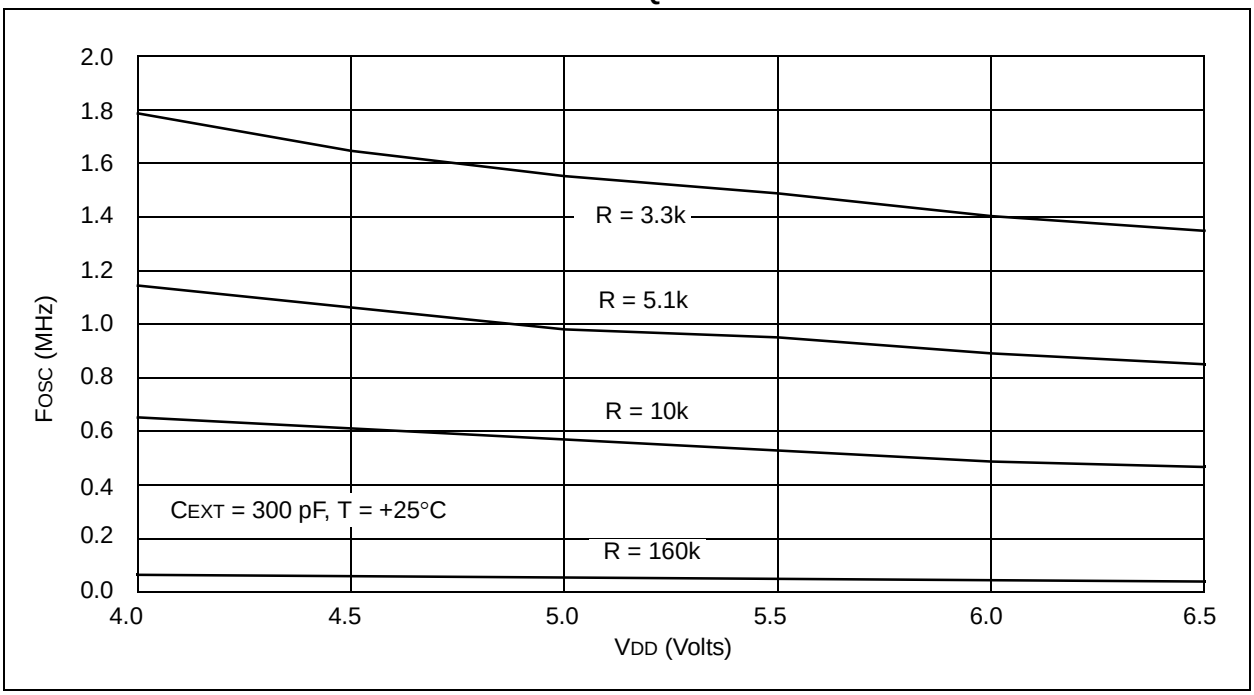


TABLE 21-2: RC OSCILLATOR FREQUENCIES

CEXT	REXT	Average FOSC @ 5V, +25°C	
22 pF	10k	3.33 MHz	± 12%
	100k	353 kHz	± 13%
100 pF	3.3k	3.54 MHz	± 10%
	5.1k	2.43 MHz	± 14%
	10k	1.30 MHz	± 17%
	100k	129 kHz	± 10%
300 pF	3.3k	1.54 MHz	± 14%
	5.1k	980 kHz	± 12%
	10k	564 kHz	± 16%
	160k	35 kHz	± 18%

FIGURE 21-21: TYPICAL, MAXIMUM AND MINIMUM VIN vs. VDD (TTL INPUT, -40°C to 125°C)

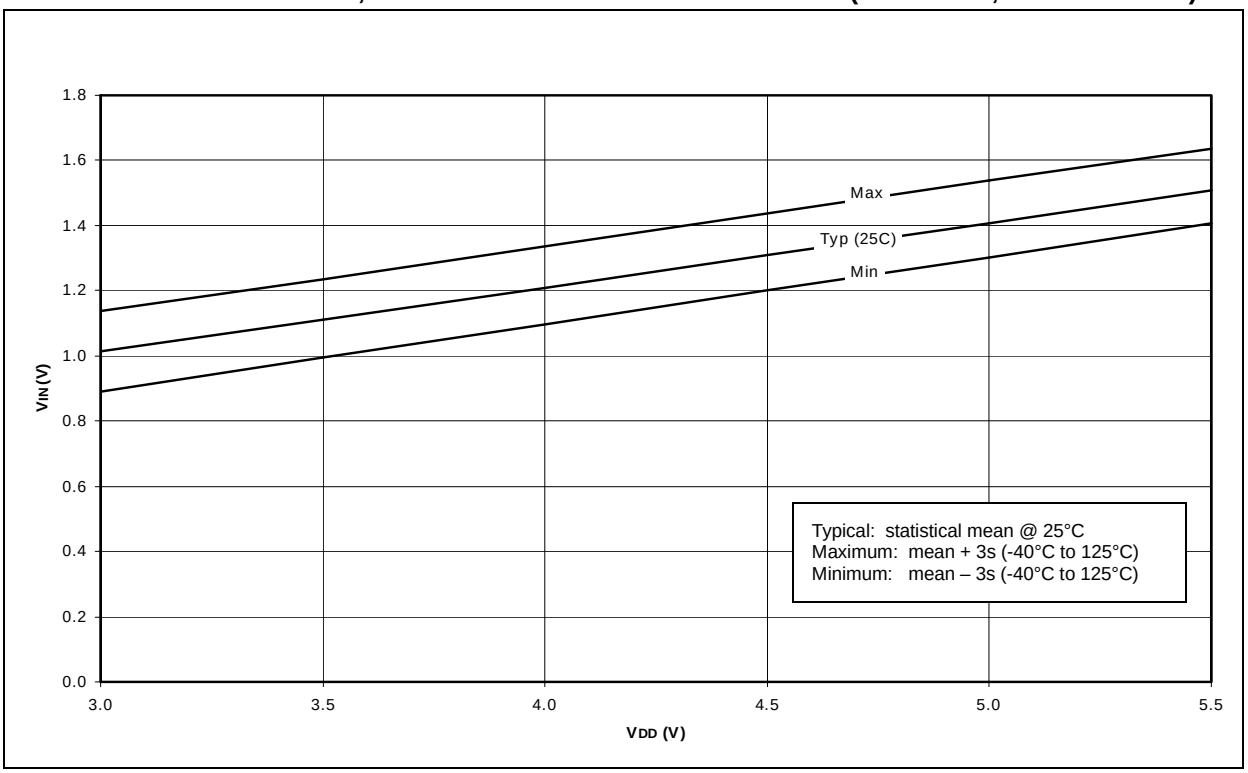


FIGURE 21-22: MAXIMUM AND MINIMUM VIN vs. VDD (ST Input, -40° C to +125°C)

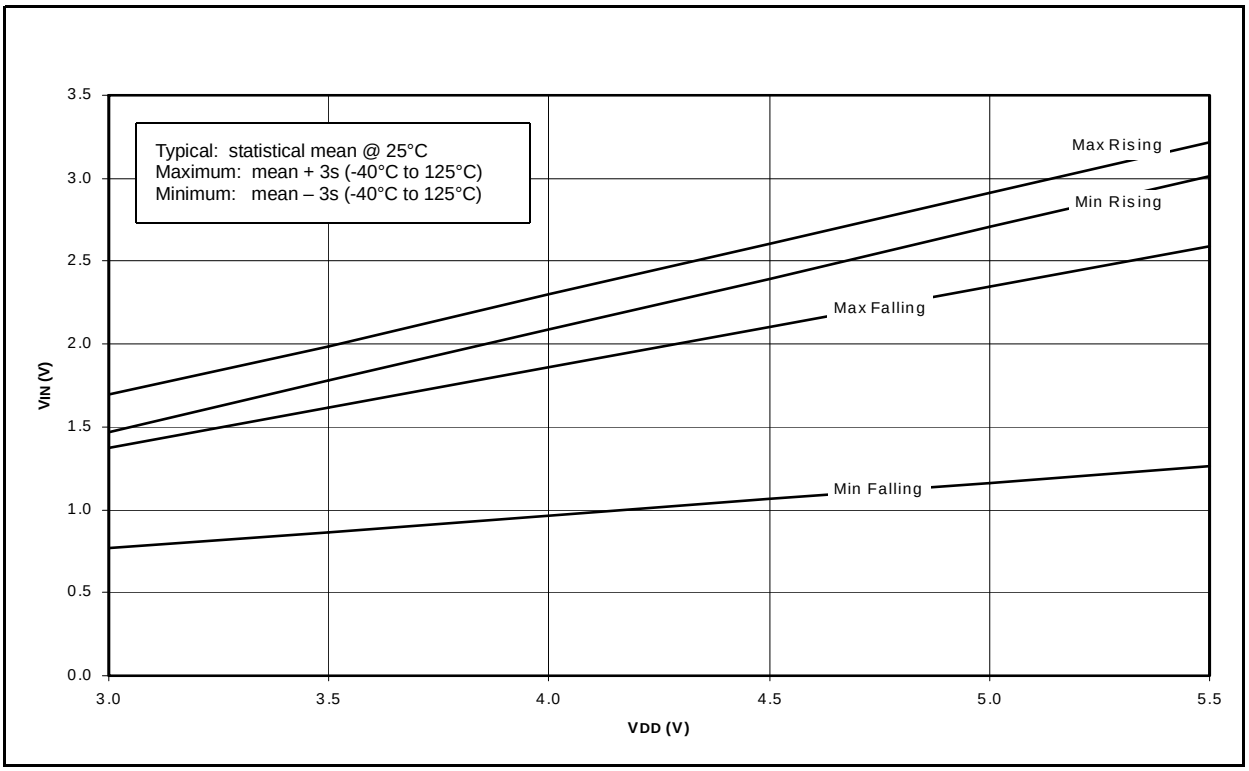
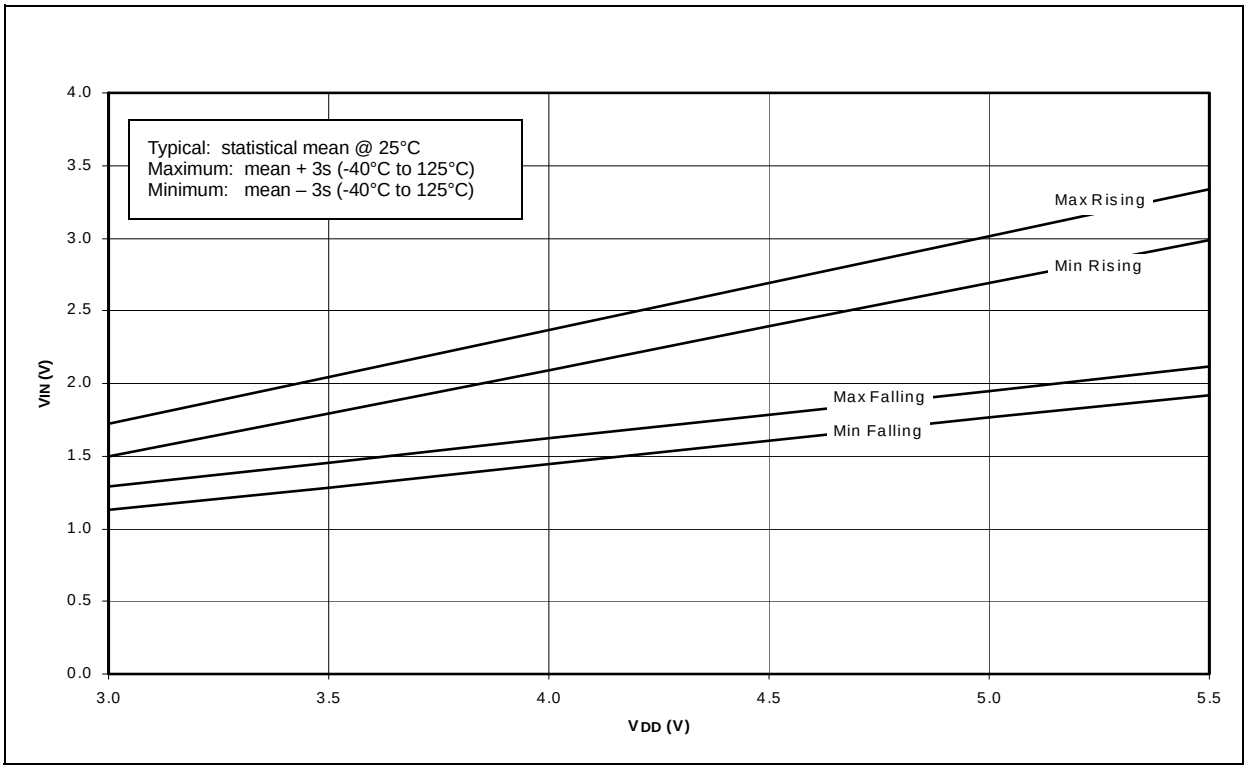


FIGURE 21-23: MAXIMUM AND MINIMUM VIN vs. VDD (I^2C Input, -40°C to +125°C)



NOTES: