Microchip Technology - PIC17C762T-16/PT Datasheet

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	Obsolete
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	66
Program Memory Size	16KB (8K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	678 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	0°C ~ 70°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/pic17c762t-16-pt

EXAMPLE 6-1: SAVING STATUS AND WREG IN RAM (SIMPLE)

```
; The addresses that are used to store the CPUSTA and WREG values must be in the data memory
; address range of 1Ah - 1Fh. Up to 6 locations can be saved and restored using the MOVFP
; instruction. This instruction neither affects the status bits, nor corrupts the WREG register.
;
UNBANK1         EQU    0x01A       ; Address for 1st location to save
UNBANK2         EQU    0x01B       ; Address for 2nd location to save
UNBANK3         EQU    0x01C       ; Address for 3rd location to save
UNBANK4         EQU    0x01D       ; Address for 4th location to save
UNBANK5         EQU    0x01E       ; Address for 5th location to save
                                   ;    (Label Not used in program)
UNBANK6         EQU    0x01F       ; Address for 6th location to save
                                   ;    (Label Not used in program)
;
        :                          ; At Interrupt Vector Address
PUSH    MOVFP   ALUSTA, UNBANK1    ; Push ALUSTA value
        MOVFP   BSR, UNBANK2       ; Push BSR value
        MOVFP   WREG, UNBANK3      ; Push WREG value
        MOVFP   PCLATH, UNBANK4    ; Push PCLATH value
        ;
        :                          ; Interrupt Service Routine (ISR) code
        ;
POP     MOVFP   UNBANK4, PCLATH    ; Restore PCLATH value
        MOVFP   UNBANK3, WREG      ; Restore WREG value
        MOVFP   UNBANK2, BSR       ; Restore BSR value
        MOVFP   UNBANK1, ALUSTA    ; Restore ALUSTA value
;
        RETFIE                     ; Return from interrupt (enable interrupts)
```

TABLE 7-3: SPECIAL FUNCTION REGISTERS (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	$\overline{\text{MCLR}}$, WDT
Bank 2											
10h	TMR1	Timer1's Register								xxxx xxxx	uuuu uuuu
11h	TMR2	Timer2's Register								xxxx xxxx	uuuu uuuu
12h	TMR3L	Timer3's Register; Low Byte								xxxx xxxx	uuuu uuuu
13h	TMR3H	Timer3's Register; High Byte								xxxx xxxx	uuuu uuuu
14h	PR1	Timer1's Period Register								xxxx xxxx	uuuu uuuu
15h	PR2	Timer2's Period Register								xxxx xxxx	uuuu uuuu
16h	PR3L/CA1L	Timer3's Period Register - Low Byte/Capture1 Register; Low Byte								xxxx xxxx	uuuu uuuu
17h	PR3H/CA1H	Timer3's Period Register - High Byte/Capture1 Register; High Byte								xxxx xxxx	uuuu uuuu
Bank 3											
10h	PW1DCL	DC1	DC0	—	—	—	—	—	—	xx-- ----	uu-- ----
11h	PW2DCL	DC1	DC0	TM2PW2	—	—	—	—	—	xx0- ----	uu0- ----
12h	PW1DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
13h	PW2DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
14h	CA2L	Capture2 Low Byte								xxxx xxxx	uuuu uuuu
15h	CA2H	Capture2 High Byte								xxxx xxxx	uuuu uuuu
16h	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/$\overline{\text{PR3}}$	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
Bank 4											
10h	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	000- 0010	000- 0010
11h	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	000- 0000	000- 0000
12h	Unimplemented	—	—	—	—	—	—	—	—	---- ----	---- ----
13h	RCSTA2	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h	RCREG2	Serial Port Receive Register for USART2								xxxx xxxx	uuuu uuuu
15h	TXSTA2	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
16h	TXREG2	Serial Port Transmit Register for USART2								xxxx xxxx	uuuu uuuu
17h	SPBRG2	Baud Rate Generator for USART2								0000 0000	0000 0000
Bank 5:											
10h	DDRF	Data Direction Register for PORTF								1111 1111	1111 1111
11h	PORTF(4)	RF7/ AN11	RF6/ AN10	RF5/ AN9	RF4/ AN8	RF3/ AN7	RF2/ AN6	RF1/ AN5	RF0/ AN4	0000 0000	0000 0000
12h	DDRG	Data Direction Register for PORTG								1111 1111	1111 1111
13h	PORTG(4)	RG7/ TX2/CK2	RG6/ RX2/DT2	RG5/ PWM3	RG4/ CAP3	RG3/ AN0	RG2/ AN1	RG1/ AN2	RG0/ AN3	xxxx 0000	uuuu 0000
14h	ADCON0	CHS3	CHS2	CHS1	CHS0	—	GO/$\overline{\text{DONE}}$	—	ADON	0000 -0-0	0000 -0-0
15h	ADCON1	ADCS1	ADCS0	ADFM	—	PCFG3	PCFG2	PCFG1	PCFG0	000- 0000	000- 0000
16h	ADRESL	A/D Result Register Low Byte								xxxx xxxx	uuuu uuuu
17h	ADRESH	A/D Result Register High Byte								xxxx xxxx	uuuu uuuu

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

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for PC<15:8> whose contents are updated from, or transferred to, the upper byte of the program counter.

2: The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ status bits in CPUSTA are not affected by a $\overline{\text{MCLR}}$ Reset.

3: Bank 8 and associated registers are only implemented on the PIC17C76X devices.

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.

NOTES:

Example 9-3 shows the sequence to do a 16 x 16 unsigned multiply. Equation 9-1 shows the algorithm that is used. The 32-bit result is stored in 4 registers, RES3:RES0.

EQUATION 9-1: 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 9-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE

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

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.

10.4 PORTD and DDRD Registers

PORTD is an 8-bit bi-directional port. The corresponding data direction register is DDRD. A '1' in DDRD configures the corresponding port pin as an input. A '0' in the DDRD register configures the corresponding port pin as an output. Reading PORTD reads the status of the pins, whereas writing to PORTD will write to the port latch. PORTD is multiplexed with the system bus. When operating as the system bus, PORTD is the high order byte of the address/data bus (AD15:AD8). The timing for the system bus is shown in the Electrical Specifications section.

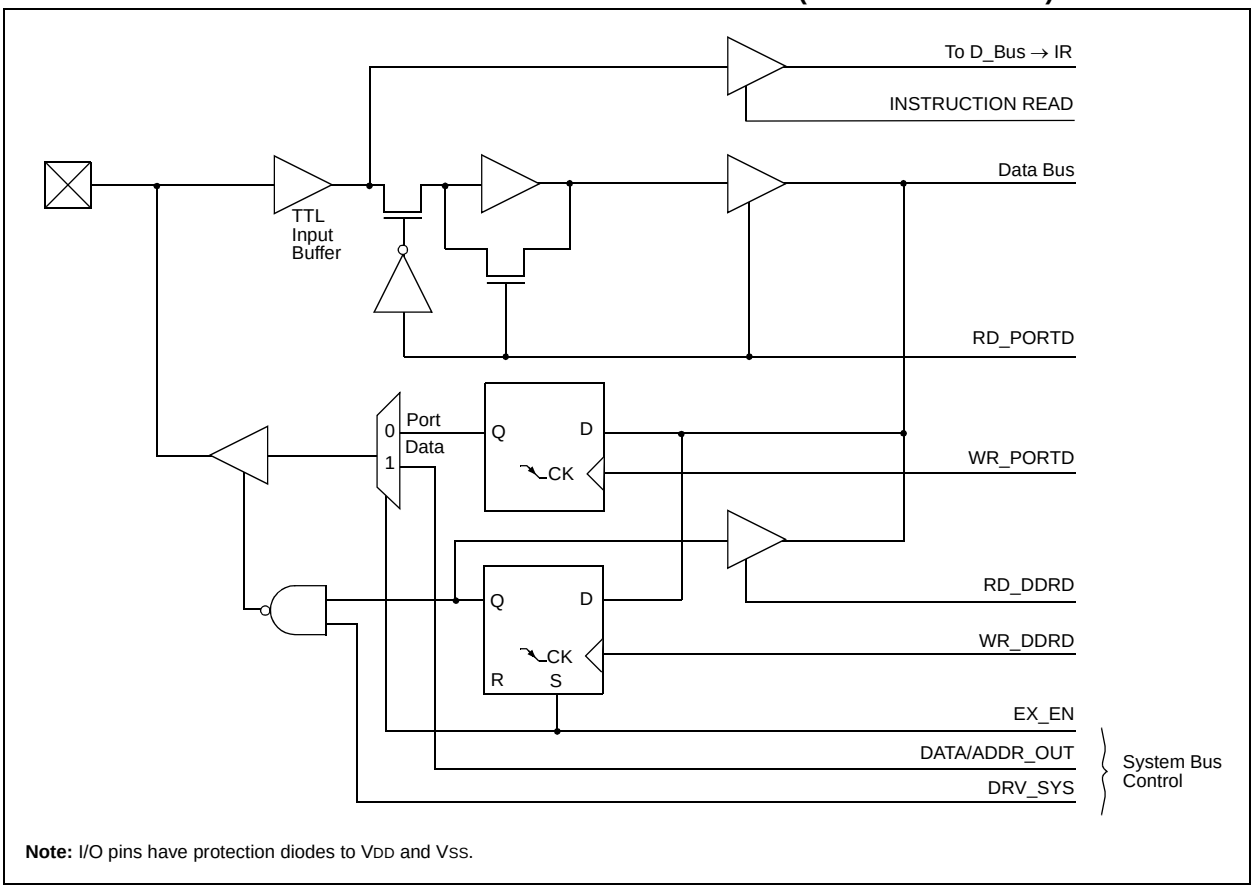
Note:	This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O.

Example 10-4 shows an instruction sequence to initialize PORTD. The Bank Select Register (BSR) must be selected to Bank 1 for the port to be initialized. The following example uses the `MOVLB` instruction to load the BSR register for bank selection.

EXAMPLE 10-4: INITIALIZING PORTD

```
MOVLB   1          ; Select Bank 1
CLRF    PORTD, F   ; Initialize PORTD data
                   ; latches before setting
                   ; the data direction reg
MOVLW    0xCF      ; Value used to initialize
                   ; data direction
MOVWF    DDRD      ; Set RD<3:0> as inputs
                   ; RD<5:4> as outputs
                   ; RD<7:6> as inputs
```

FIGURE 10-10: BLOCK DIAGRAM OF RD7:RD0 PORT PINS (IN I/O PORT MODE)



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

TABLE 13-3: SUMMARY OF TIMER1, TIMER2 AND TIMER3 REGISTERS

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	$\overline{\text{MCLR}}$, WDT
16h, Bank 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/$\overline{\text{PR3}}$	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
16h, Bank 7	TCON3	—	CA4OVF	CA3OVF	CA4ED1	CA4ED0	CA3ED1	CA3ED0	PWM3ON	-000 0000	-000 0000
10h, Bank 2	TMR1	Timer1's Register								xxxx xxxx	uuuu uuuu
11h, Bank 2	TMR2	Timer2's Register								xxxx xxxx	uuuu uuuu
16h, Bank 1	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	x000 0010	u000 0010
17h, Bank 1	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	0000 0000	0000 0000
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	--11 11qq	--11 qquu
14h, Bank 2	PR1	Timer1 Period Register								xxxx xxxx	uuuu uuuu
15h, Bank 2	PR2	Timer2 Period Register								xxxx xxxx	uuuu uuuu
10h, Bank 3	PW1DCL	DC1	DC0	—	—	—	—	—	—	xx-- ----	uu-- ----
11h, Bank 3	PW2DCL	DC1	DC0	TM2PW2	—	—	—	—	—	xx0- ----	uu0- ----
10h, Bank 7	PW3DCL	DC1	DC0	TM2PW3	—	—	—	—	—	xx0- ----	uu0- ----
12h, Bank 3	PW1DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
13h, Bank 3	PW2DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
11h, Bank 7	PW3DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as a '0', q = value depends on condition.
Shaded cells are not used by Timer1 or Timer2.

14.4 USART Synchronous Slave Mode

The Synchronous Slave mode differs from the Master mode, in the fact that the shift clock is supplied externally at the TX/CK pin (instead of being supplied internally in the Master mode). This allows the device to transfer or receive data in the SLEEP mode. The Slave mode is entered by clearing the CSRC (TXSTA<7>) bit.

14.4.1 USART SYNCHRONOUS SLAVE TRANSMIT

The operation of the SYNC Master and Slave modes are identical except in the case of the SLEEP mode.

If two words are written to TXREG and then the SLEEP instruction executes, the following will occur. The first word will immediately transfer to the TSR and will transmit as the shift clock is supplied. The second word will remain in TXREG. TXIF will not be set. When the first word has been shifted out of TSR, TXREG will transfer the second word to the TSR and the TXIF flag will now be set. If TXIE is enabled, the interrupt will wake the chip from SLEEP and if the global interrupt is enabled, then the program will branch to the interrupt vector (0020h).

Steps to follow when setting up a Synchronous Slave Transmission:

1. Enable the synchronous slave serial port by setting the SYNC and SPEN bits and clearing the CSRC bit.
2. Clear the CREN bit.
3. If interrupts are desired, then set the TXIE bit.
4. If 9-bit transmission is desired, then set the TX9 bit.
5. If 9-bit transmission is selected, the ninth bit should be loaded in TX9D.
6. Start transmission by loading data to TXREG.
7. Enable the transmission by setting TXEN.

Writing the transmit data to the TXREG, then enabling the transmit (setting TXEN), allows transmission to start sooner than doing these two events in the reverse order.

Note:	To terminate a transmission, either clear the SPEN bit, or the TXEN bit. This will reset the transmit logic, so that it will be in the proper state when transmit is re-enabled.

14.4.2 USART SYNCHRONOUS SLAVE RECEPTION

Operation of the Synchronous Master and Slave modes are identical except in the case of the SLEEP mode. Also, SREN is a “don't care” in Slave mode.

If receive is enabled (CREN) prior to the SLEEP instruction, then a word may be received during SLEEP. On completely receiving the word, the RSR will transfer the data to RCREG (setting RCIF) and if the RCIE bit is set, the interrupt generated will wake the chip from SLEEP. If the global interrupt is enabled, the program will branch to the interrupt vector (0020h).

Steps to follow when setting up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting the SYNC and SPEN bits and clearing the CSRC bit.
2. If interrupts are desired, then set the RCIE bit.
3. If 9-bit reception is desired, then set the RX9 bit.
4. To enable reception, set the CREN bit.
5. The RCIF bit will be set when reception is complete and an interrupt will be generated if the RCIE bit was set.
6. Read RCSTA to get the ninth bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading RCREG.
8. If any error occurred, clear the error by clearing the CREN bit.

Note:	To abort reception, either clear the SPEN bit, or the CREN bit (when in Continuous Receive mode). This will reset the receive logic, so that it will be in the proper state when receive is re-enabled.

15.1.5 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 interrupt flag bit SSPIF (PIR2<7>) 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.

15.1.6 SLAVE SELECT SYNCHRONIZATION

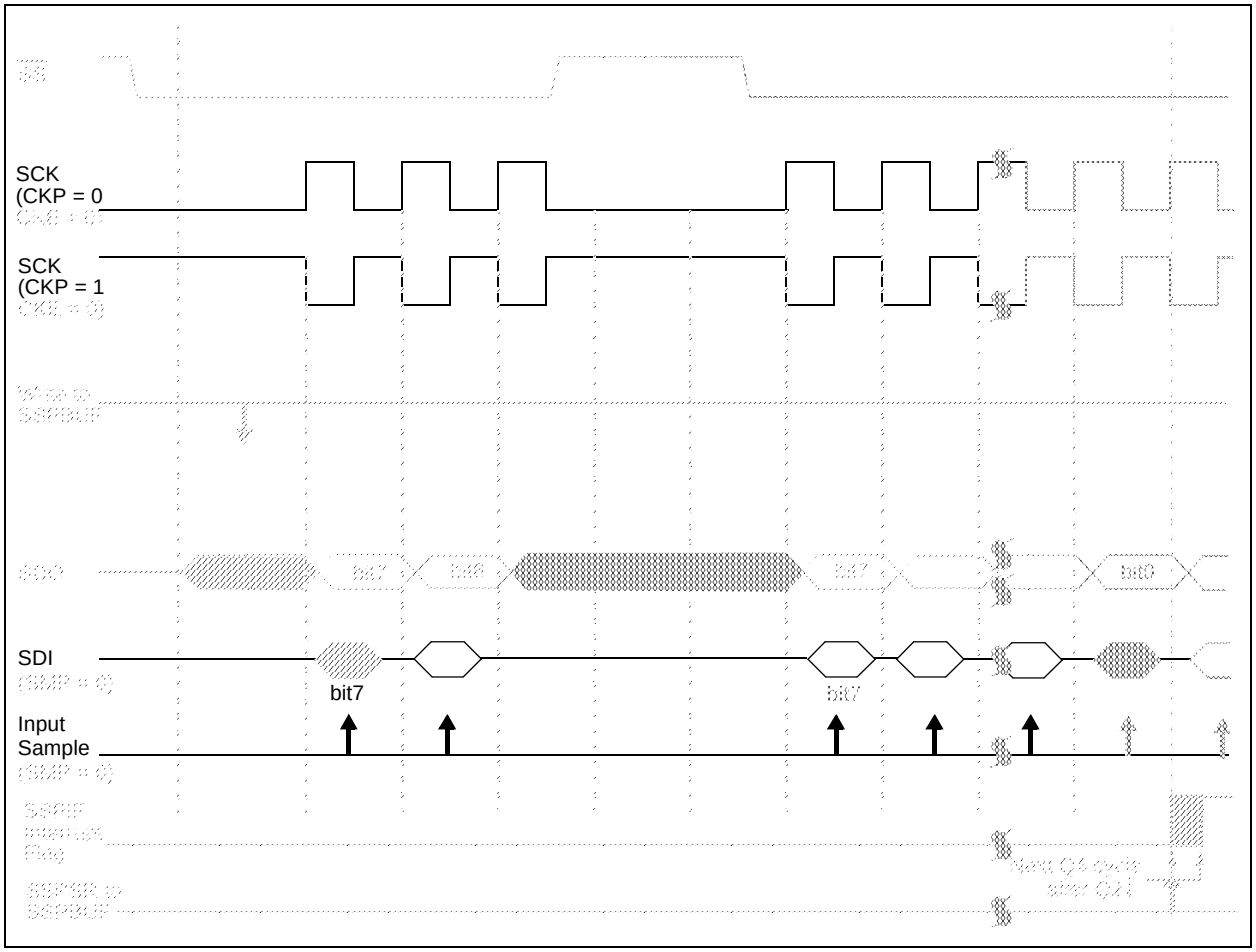
The $\overline{SS}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with $\overline{SS}$ pin control enabled (SSPCON1<3:0> = 04h). The pin must not be driven low for the $\overline{SS}$ pin to function as an input. The RA2 Data Latch must be high. When the $\overline{SS}$ pin is low, 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 (SSPCON<3:0> = 0100), the SPI module will reset if the $\overline{SS}$ pin is set to V_{DD}.
2:	If the SPI is used in Slave mode with CKE = '1', 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 15-7: SLAVE SYNCHRONIZATION WAVEFORM



A typical transmit sequence would go as follows:

a) The user generates a START Condition by setting the START enable bit (SEN) in SSPCON2.
b) SSPIF is set. The module will wait the required START time before any other operation takes place.
c) The user loads the SSPBUF with address to transmit.
d) Address is shifted out the SDA pin until all 8 bits are transmitted.
e) The MSSP Module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
f) The module generates an interrupt at the end of the ninth clock cycle by setting SSPIF.
g) The user loads the SSPBUF with eight bits of data.
h) DATA is shifted out the SDA pin until all 8 bits are transmitted.
i) The MSSP Module shifts in the ACK bit from the slave device, and writes its value into the SSPCON2 register (SSPCON2<6>).
j) The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
k) The user generates a STOP condition by setting the STOP enable bit PEN in SSPCON2.
l) Interrupt is generated once the STOP condition is complete.

15.2.8 BAUD RATE GENERATOR

In I^2C Master mode, the reload value for the BRG is located in the lower 7 bits of the SSPADD register (Figure 15-18). When the BRG is loaded with this value, the BRG counts down to 0 and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (TCY), on the Q2 and Q4 clock.

In I^2C Master mode, the BRG is reloaded automatically. If Clock Arbitration is taking place, for instance, the BRG will be reloaded when the SCL pin is sampled high (Figure 15-19).

FIGURE 15-18: BAUD RATE GENERATOR BLOCK DIAGRAM

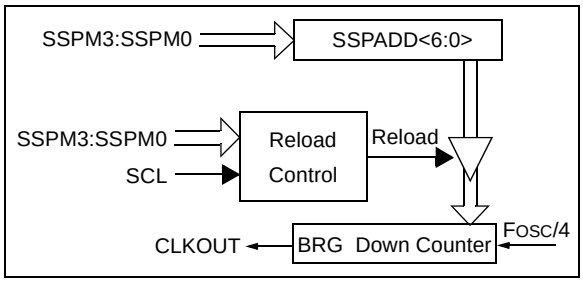
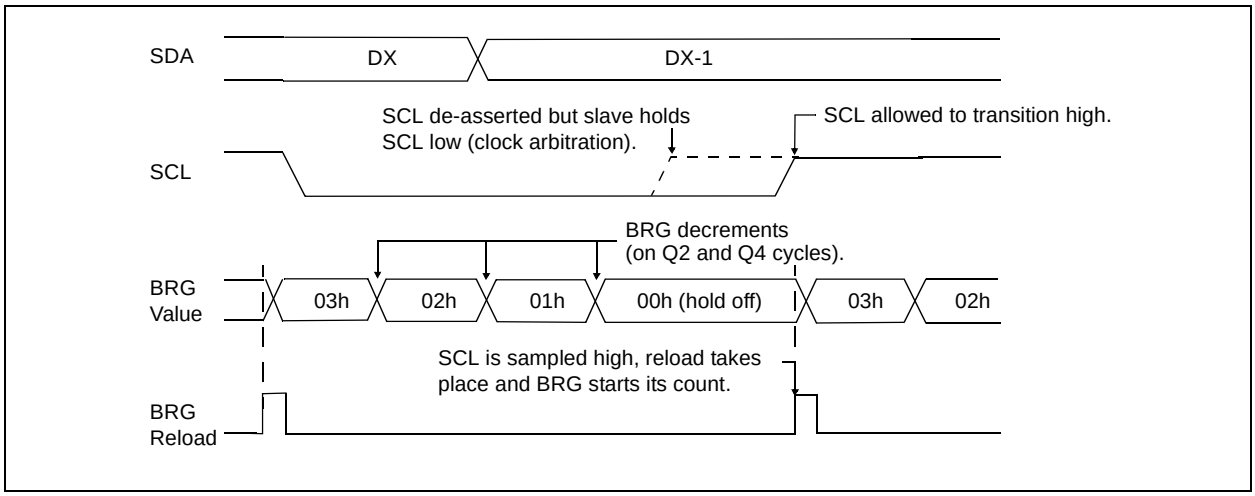


FIGURE 15-19: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION



15.2.18.1 Bus Collision During a START Condition

During a START condition, a bus collision occurs if:

a) SDA or SCL are sampled low at the beginning of the START condition (Figure 15-35).

b) SCL is sampled low before SDA is asserted low (Figure 15-36).

During a START condition, both the SDA and the SCL pins are monitored.

If:

the SDA pin is already low
or the SCL pin is already low,

then:

the START condition is aborted,
and the BCLIF flag is set,
and the SSP module is reset to its IDLE state (Figure 15-35).

The START condition begins with the SDA and SCL pins de-asserted. When the SDA pin is sampled high, the baud rate generator is loaded from SSPADD<6:0> and counts down to '0'. If the SCL pin is sampled low while SDA is high, a bus collision occurs, because it is assumed that another master is attempting to drive a data '1' during the START condition.

If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (Figure 15-37). If, however, a '1' is sampled on the SDA pin, the SDA pin is asserted low at the end of the BRG count. The baud rate generator is then reloaded and counts down to 0 and during this time, if the SCL pin is sampled as '0', a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

Note: The reason that bus collision is not a factor during a START condition is that no two bus masters can assert a START condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the START condition and if the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start, or Stop conditions.

FIGURE 15-35: BUS COLLISION DURING START CONDITION (SDA ONLY)

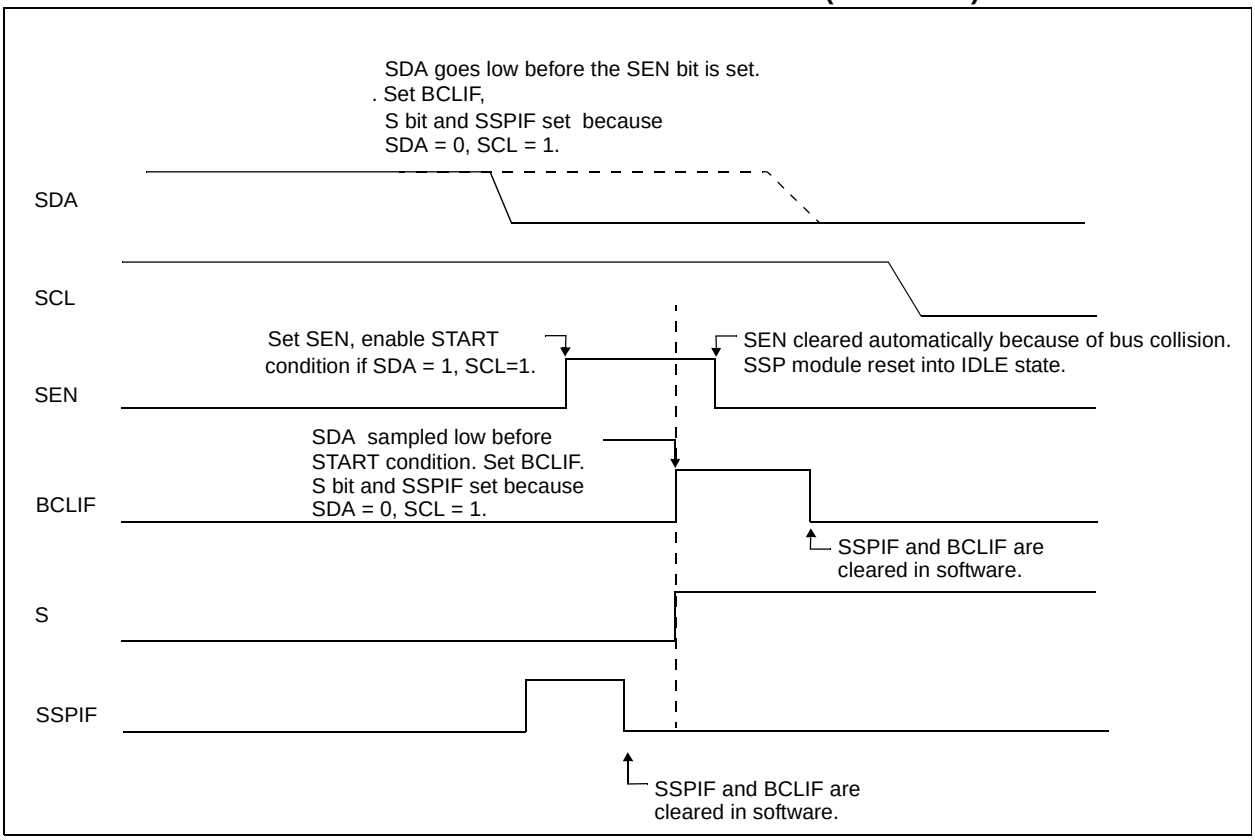


FIGURE 15-36: BUS COLLISION DURING START CONDITION (SCL = 0)

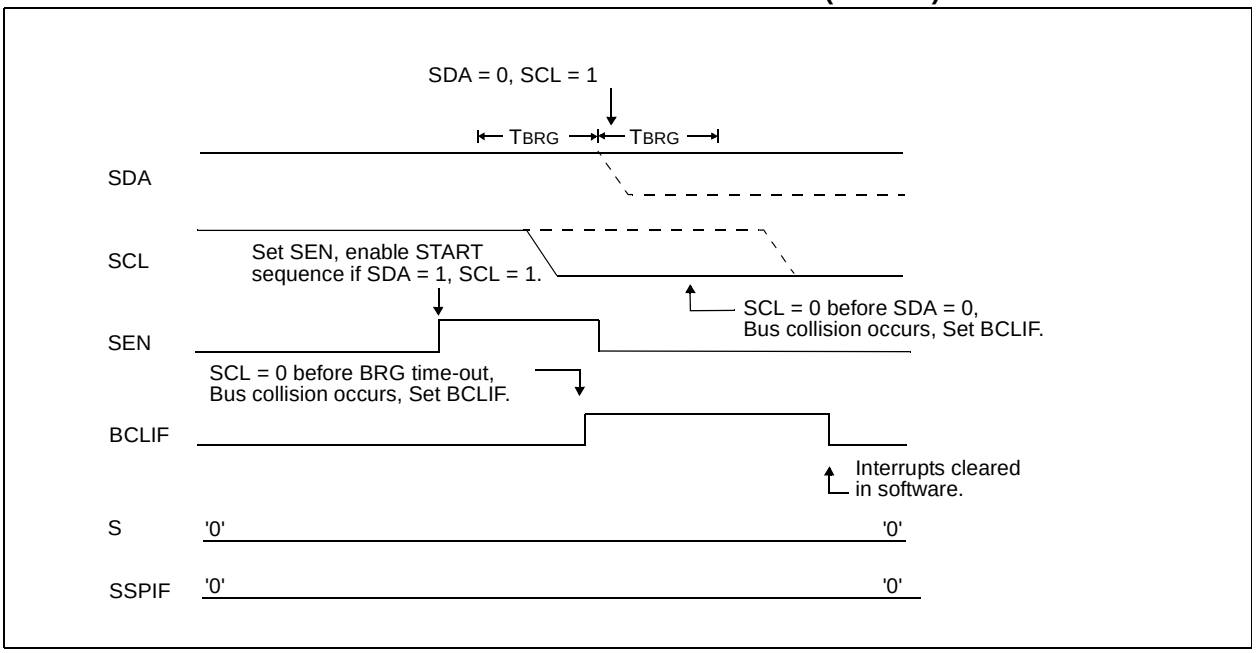
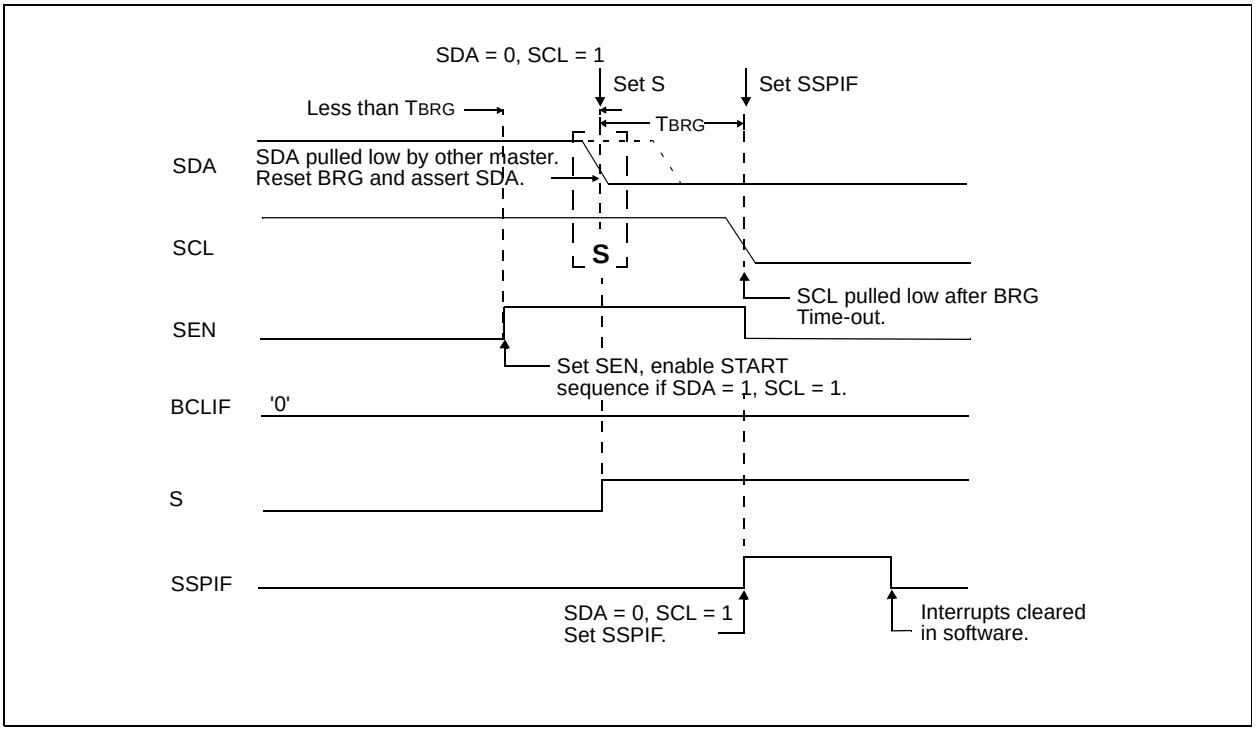


FIGURE 15-37: BRG RESET DUE TO SDA COLLISION DURING START CONDITION



15.4 Example Program

Example 15-2 shows MPLAB® C17 'C' code for using the I^2C module in Master mode to communicate with a 24LC01B serial EEPROM. This example uses the PIC® MCU 'C' libraries included with MPLAB C17.

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

```
// Include necessary header files
#include <p17c756.h>     // Processor header file
#include <delays.h>      // Delay routines header file
#include <stdlib.h>      // Standard Library header file
#include <i2c16.h>       // I2C routines header file

#define CONTROL 0xa0     // Control byte definition for 24LC01B

// Function declarations
void main(void);
void WritePORTD(static unsigned char data);
void ByteWrite(static unsigned char address,static unsigned char data);
unsigned char ByteRead(static unsigned char address);
void ACKPoll(void);

// Main program
void main(void)
{
static unsigned char address;    // I2C address of 24LC01B
static unsigned char datao;      // Data written to 24LC01B
static unsigned char datai;      // Data read from 24LC01B

    address = 0;                 // Preset address to 0
    OpenI2C(MASTER,SLEW_ON);     // Configure I2C Module Master mode, Slew rate control on
    SSPADD = 39;                 // Configure clock for 100KHz

    while(address<128)           // Loop 128 times, 24LC01B is 128x8
    {
        datao = PORTB;
        do
        {
            ByteWrite(address,datao);   // Write data to EEPROM
            ACKPoll();                  // Poll the 24LC01B for state
            datai = ByteRead(address);  // Read data from EEPROM into SSPBUF
        } while(datai != datao);        // Loop as long as data not correctly
                                        //    written to 24LC01B

        address++;                      // Increment address
    }
    while(1)                            // Done writing 128 bytes to 24LC01B, Loop forever
    {
        Nop();
    }
}
```

17.4 Power-down Mode (SLEEP)

The Power-down mode is entered by executing a SLEEP instruction. This clears the Watchdog Timer and postscaler (if enabled). The $\overline{PD}$ bit is cleared and the $\overline{TO}$ bit is set (in the CPUSTA register). In SLEEP mode, the oscillator driver is turned off. The I/O ports maintain their status (driving high,low, or hi-impedance input).

The $\overline{MCLR}/V_{PP}$ pin must be at a logic high level (V_{IHMC}). A WDT time-out RESET does not drive the $\overline{MCLR}/V_{PP}$ pin low.

17.4.1 WAKE-UP FROM SLEEP

The device can wake-up from SLEEP through one of the following events:

- Power-on Reset
- Brown-out Reset
- External RESET input on $\overline{MCLR}/V_{PP}$ pin
- WDT Reset (if WDT was enabled)
- Interrupt from RA0/INT pin, RB port change, T0CKI interrupt, or some peripheral interrupts

The following peripheral interrupts can wake the device from SLEEP:

- Capture interrupts
- USART synchronous slave transmit interrupts
- USART synchronous slave receive interrupts
- A/D conversion complete
- SPI slave transmit/receive complete
- I^2C slave receive

Other peripherals cannot generate interrupts since during SLEEP, no on-chip Q clocks are present.

Any RESET event will cause a device RESET. Any interrupt event is considered a continuation of program execution. The $\overline{TO}$ and $\overline{PD}$ bits in the CPUSTA register can be used to determine the cause of a device RESET. The $\overline{PD}$ bit, which is set on power-up, is cleared when SLEEP is invoked. The $\overline{TO}$ bit is cleared if WDT time-out occurred (and caused a RESET).

When the SLEEP instruction is being executed, the next instruction (PC + 1) is pre-fetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be set (enabled). Wake-up is regardless of the state of the GLINTD bit. If the GLINTD bit is set (disabled), the device continues execution at the instruction after the SLEEP instruction. If the GLINTD bit is clear (enabled), the device executes the instruction after the SLEEP instruction and then branches to the interrupt vector address. In cases where the execution of the instruction following SLEEP is not desirable, the user should have a NOP after the SLEEP instruction.

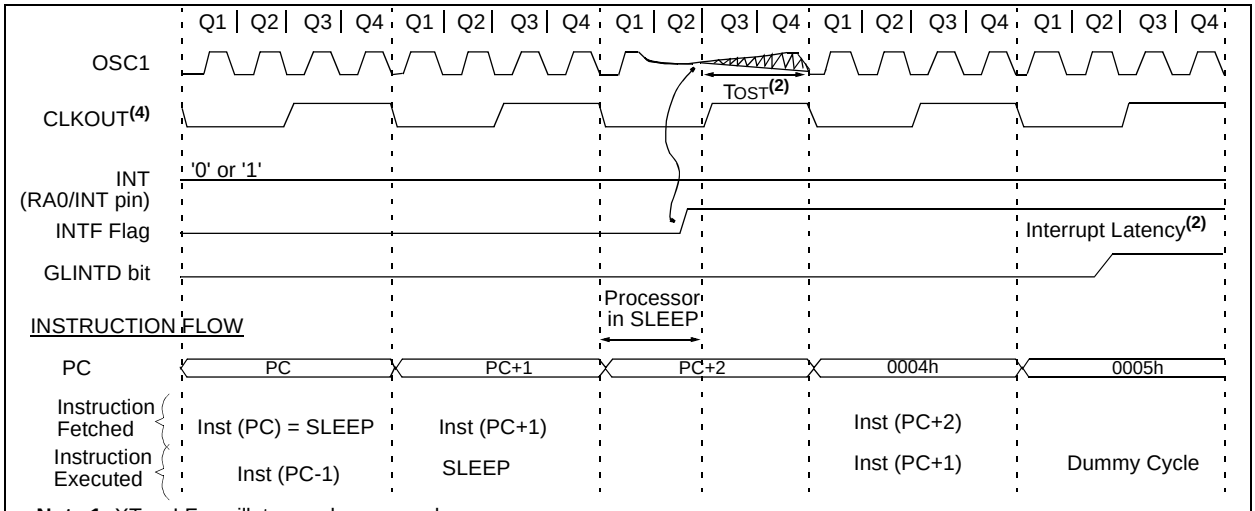
Note:	If the global interrupt is disabled (GLINTD is set), but any interrupt source has both its interrupt enable bit and the corresponding interrupt flag bit set, the device will immediately wake-up from SLEEP. The $\overline{TO}$ bit is set and the $\overline{PD}$ bit is cleared.

The WDT is cleared when the device wakes from SLEEP, regardless of the source of wake-up.

17.4.1.1 Wake-up Delay

When the oscillator type is configured in XT or LF mode, the Oscillator Start-up Timer (OST) is activated on wake-up. The OST will keep the device in RESET for 1024TOSC. This needs to be taken into account when considering the interrupt response time when coming out of SLEEP.

FIGURE 17-2: WAKE-UP FROM SLEEP THROUGH INTERRUPT



Note 1: XT or LF oscillator mode assumed.
2: TOST = 1024TOSC (drawing not to scale). This delay will not be there for RC osc mode.
3: When GLINTD = 0, processor jumps to interrupt routine after wake-up. If GLINTD = 1, execution will continue in line.
4: CLKOUT is not available in these osc modes, but shown here for timing reference.

INFSNZ **Increment f, skip if not 0**

Syntax: [*label*] INFSNZ f,d

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

Operation: (f) + 1 → (dest),
skip if not 0

Status Affected: None

Encoding:

0010	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'.
If the result is not 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    INFSNZ  REG, 1
ZERO
NZERO
```

Before Instruction
REG = REG

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

IORLW **Inclusive OR Literal with WREG**

Syntax: [*label*] IORLW k

Operands: $0 \le k \le 255$

Operation: (WREG) .OR. (k) → (WREG)

Status Affected: Z

Encoding:

1011	0011	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   0x35
```

Before Instruction
WREG = 0x9A

After Instruction
WREG = 0xBF

MOVPF **Move p to f**

Syntax: [*label*] MOVPF p,f

Operands: $0 \le f \le 255$
$0 \le p \le 31$

Operation: (p) → (f)

Status Affected: Z

Encoding:

010p	pppp	ffff	ffff

Description: Move data from data memory location 'p' to data memory location 'f'. Location 'f' can be anywhere in the 256 byte data space (00h to FFh), while 'p' can be 00h to 1Fh.

Either 'p' or 'f' can be WREG (a useful, special situation).

MOVPF is particularly useful for transferring a peripheral register (e.g. the timer or an I/O port) to a data memory location. Both 'f' and 'p' can be indirectly addressed.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example:
```
MOVPF   REG1, REG2
```
Before Instruction
REG1 = 0x11
REG2 = 0x33

After Instruction
REG1 = 0x11
REG2 = 0x11

MOVWF **Move WREG to f**

Syntax: [*label*] MOVWF f

Operands: $0 \le f \le 255$

Operation: (WREG) → (f)

Status Affected: None

Encoding:

0000	0001	ffff	ffff

Description: Move data from WREG to register 'f'. Location 'f' can be anywhere in the 256 byte data space.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example:
```
MOVWF   REG
```
Before Instruction
WREG = 0x4F
REG = 0xFF

After Instruction
WREG = 0x4F
REG = 0x4F

TLWT **Table Latch Write**

Syntax: [*label*] TLWT t,f

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

Operation: If t = 0,
f → TBLATL;
If t = 1,
f → TBLATH

Status Affected: None

Encoding:

1010	01tx	ffff	ffff

Description: Data from file register 'f' is written into the 16-bit table latch (TBLAT).
If t = 1; high byte is written
If t = 0; low byte is written
This instruction is used in conjunction with TABLWT to transfer data from data memory to program memory.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: TLWT t, RAM

Before Instruction
t = 0
RAM = 0xB7
TBLAT = 0x0000 (TBLATH = 0x00) (TBLATL = 0x00)

After Instruction
RAM = 0xB7
TBLAT = 0x00B7 (TBLATH = 0x00) (TBLATL = 0xB7)

Before Instruction
t = 1
RAM = 0xB7
TBLAT = 0x0000 (TBLATH = 0x00) (TBLATL = 0x00)

After Instruction
RAM = 0xB7
TBLAT = 0xB700 (TBLATH = 0xB7) (TBLATL = 0x00)

TSTFSZ **Test f, skip if 0**

Syntax: [*label*] TSTFSZ f

Operands: $0 \le f \le 255$

Operation: skip if f = 0

Status Affected: None

Encoding:

0011	0011	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.

Words: 1

Cycles: 1 (2)

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

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

Before Instruction
PC = Address (HERE)

After Instruction
If CNT = 0x00,
PC = Address (ZERO)
If CNT ¼ 0x00,
PC = Address (NZERO)

20.2 DC Characteristics: PIC17C7XX-16 (Commercial, Industrial, Extended) PIC17C7XX-33 (Commercial, Industrial, Extended) PIC17LC7XX-08 (Commercial, Industrial)

DC CHARACTERISTICS			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +125°C for extended -40°C ≤ TA ≤ +85°C for industrial 0°C ≤ TA ≤ +70°C for commercial Operating voltage VDD range as described in Section 20.1				
Param. No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
		Input Low Voltage					
	VIL	I/O ports					
D030		with TTL buffer **(Note 6)**	Vss	–	0.8	V	4.5V ≤ VDD ≤ 5.5V
			Vss	–	0.2VDD	V	3.0V ≤ VDD ≤ 4.5V
D031		with Schmitt Trigger buffer					
		RA2, RA3	Vss	–	0.3VDD	V	I^2C compliant
		All others	Vss	–	0.2VDD	V	
D032		$\overline{MCLR}$, OSC1 (in EC and RC mode)	Vss	–	0.2VDD	V	**(Note 1)**
D033		OSC1 (in XT, and LF mode)	–	0.5VDD	–	V	
		Input High Voltage					
	VIH	I/O ports					
D040		with TTL buffer **(Note 6)**	2.0	–	VDD	V	4.5V ≤ VDD ≤ 5.5V
			1 + 0.2VDD	–	VDD	V	3.0V ≤ VDD ≤ 4.5V
D041		with Schmitt Trigger buffer					
		RA2, RA3	0.7VDD	–	VDD	V	I^2C compliant
		All others	0.8VDD	–	VDD	V	
D042		$\overline{MCLR}$	0.8VDD	–	VDD	V	**(Note 1)**
D043		OSC1 (XT, and LF mode)	–	0.5VDD	–	V	
D050	VHYS	**Hysteresis of Schmitt Trigger Inputs**	0.15VDD	–	–	V	

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

Note 1: In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC17CXXX devices be driven with external clock in RC mode.

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

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

4: These specifications are for the programming of the on-chip program memory EPROM through the use of the table write instructions. The complete programming specifications can be found in: PIC17C7XX Programming Specifications (Literature number DS TBD).

5: The $\overline{MCLR}$/VPP pin may be kept in this range at times other than programming, but is not recommended.

6: For TTL buffers, the better of the two specifications may be used.

21.0 PIC17C7XX DC AND AC CHARACTERISTICS

The graphs and tables provided in this section are for design guidance and are not tested nor guaranteed. In some graphs or tables the data presented is outside specified operating range (e.g., outside specified VDD range). This is for information only and devices are ensured to operate properly only within the specified range.

The data presented in this section is a statistical summary of data collected on units from different lots over a period of time.

- **Typ** or **Typical** represents the mean of the distribution at 25°C.
- **Max** or **Maximum** represents (mean + 3σ) over the temperature range of -40°C to 85°C.
- **Min** or **Minimum** represents (mean - 3σ) over the temperature range of -40°C to 85°C.

Note: Standard deviation is denoted by sigma (σ).

TABLE 21-1: PIN CAPACITANCE PER PACKAGE TYPE

Pin Name	Typical Capacitance (pF)	
	68-pin PLCC	64-pin TQFP
All pins, except $\overline{MCLR}$, VDD, and VSS	10	10
$\overline{MCLR}$ pin	20	20

FIGURE 21-1: TYPICAL RC OSCILLATOR FREQUENCY vs. TEMPERATURE

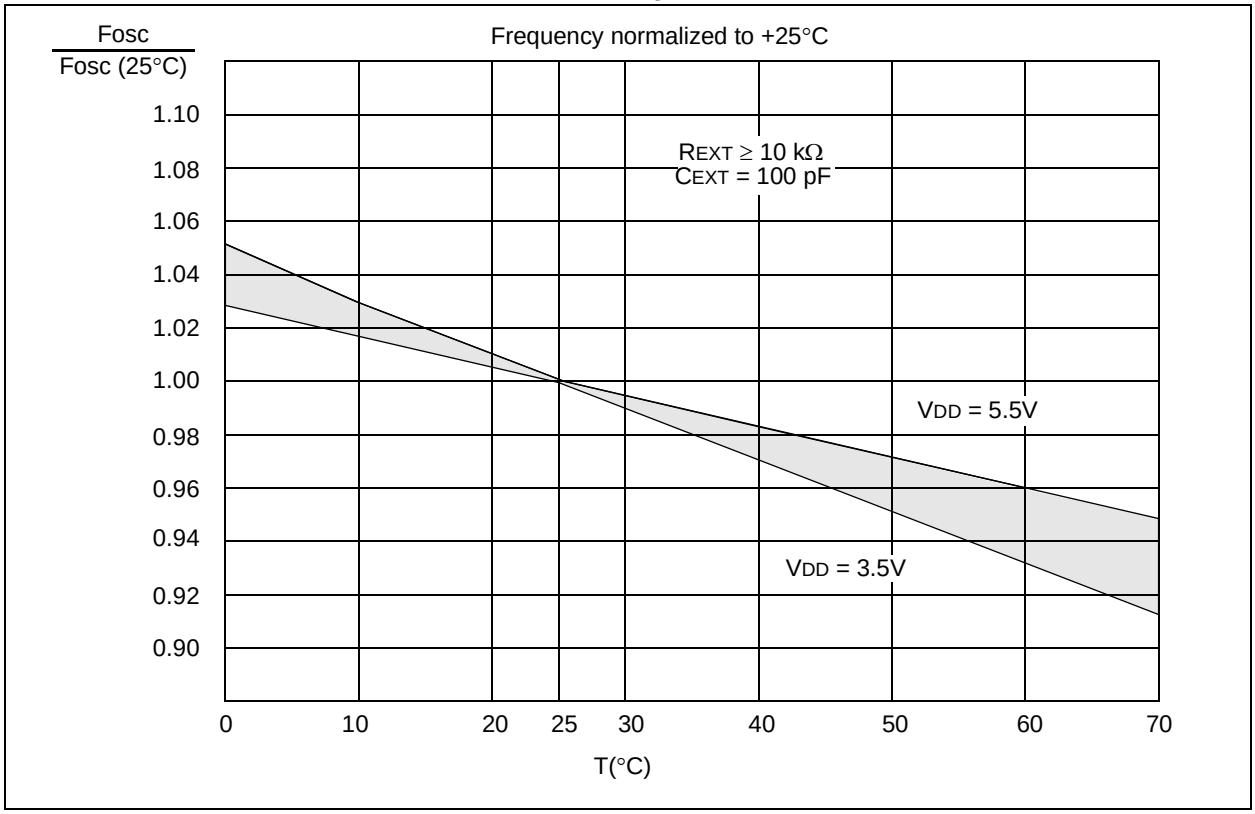


FIGURE 21-15: TYPICAL, MINIMUM AND MAXIMUM WDT PERIOD vs. VDD (-40°C TO +125°C)

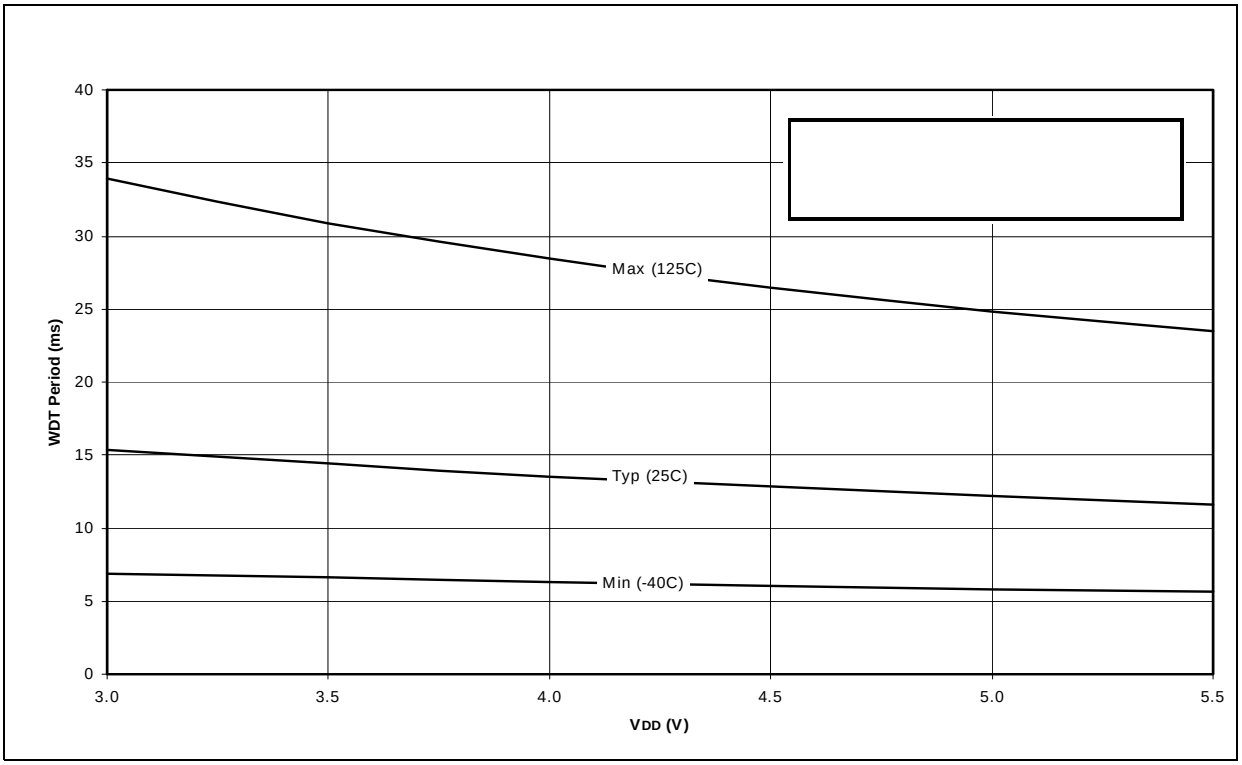


FIGURE 21-16: TYPICAL WDT PERIOD vs. VDD OVER TEMPERATURE (-40°C TO +125°C)