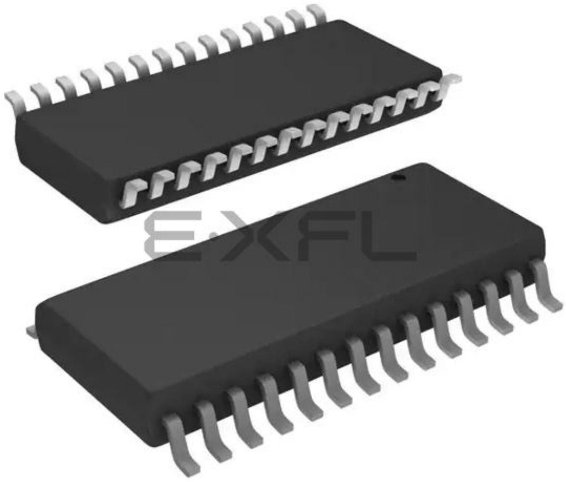


Welcome to E-XFL.COM

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	4MHz
Connectivity	I²C, SPI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	22
Program Memory Size	3.5KB (2K x 14)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	4V ~ 5.5V
Data Converters	A/D 5x8b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16c72at-04i-so

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and Peripheral Modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 2-1.

The Special Function Registers can be classified into two sets; core (CPU) and peripheral. Those registers associated with the core functions are described in detail in this section. Those related to the operation of the peripheral features are described in detail in the peripheral feature section.

TABLE 2-1 SPECIAL FUNCTION REGISTER SUMMARY

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets (4)
Bank 0											
00h	INDF[1]	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	0000 0000
01h	TMR0	Timer0 module's register								xxxx xxxx	uuuu uuuu
02h	PCL[1]	Program Counter's (PC) Least Significant Byte								0000 0000	0000 0000
03h	STATUS[1]	IRP[5]	RP1[5]	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	0001 1xxx	000q quuu
04h	FSR[1]	Indirect data memory address pointer								xxxx xxxx	uuuu uuuu
05h	PORTA[6,7]	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	--0u 0000
06h	PORTB[6,7]	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	uuuu uuuu
07h	PORTC[6,7]	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	uuuu uuuu
08h-09h	—	Unimplemented								—	—
0Ah	PCLATH[1,2]	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	---0 0000
0Bh	INTCON[1]	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	ADIF[3]	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF	-0-- 0000	-0-- 0000
0Dh	—	Unimplemented								—	—
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu
11h	TMR2	Timer2 module's register								0000 0000	0000 0000
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
18h-1Dh	—	Unimplemented								—	—
1Eh	ADRES[3]	A/D Result Register								xxxx xxxx	uuuu uuuu
1Fh	ADCON0[3]	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON	0000 00-0	0000 00-0

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

Note 1: These registers can be addressed from either bank.
2: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for PC<12:8> whose contents are transferred to the upper byte of the program counter.
3: A/D not implemented on the PIC16C62B, maintain as '0'.
4: Other (non power-up) resets include: external reset through $\overline{\text{MCLR}}$ and the Watchdog Timer Reset.
5: The IRP and RP1 bits are reserved. Always maintain these bits clear.
6: On any device reset, these pins are configured as inputs.
7: This is the value that will be in the port output latch.

2.3 PCL and PCLATH

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 13 bits wide. The low byte is called the PCL register and is readable and writable. The high byte is called the PCH register. This register contains the PC<12:8> bits and is not directly accessible. All updates to the PCH register go through the PCLATH register.

2.3.1 STACK

The stack allows any combination of up to 8 program calls and interrupts to occur. The stack contains the return address from this branch in program execution.

Mid-range devices have an 8 level deep hardware stack. The stack space is not part of either program or data space and the stack pointer is not accessible. The PC is PUSHed onto the stack when a `CALL` instruction is executed or an interrupt causes a branch. The stack is POPed in the event of a `RETURN`, `RETLW` or a `RETFIE` instruction execution. PCLATH is not modified when the stack is PUSHed or POPed.

After the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

2.4 Program Memory Paging

The `CALL` and `GOTO` instructions provide 11 bits of address to allow branching within any 2K program memory page. When doing a `CALL` or `GOTO` instruction, the upper bit of the address is provided by PCLATH<3>. The user must ensure that the page select bit is programmed to address the proper program memory page. If a return from a `CALL` instruction (or interrupt) is executed, the entire 13-bit PC is popped from the stack. Therefore, manipulation of the PCLATH<3> bit is not required for the return instructions.

TABLE 3-5 PORTC FUNCTIONS

Name	Bit#	Buffer Type	Function	TRISC Override
RC0/T1OSO/T1CKI	bit0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input	Yes
RC1/T1OSI	bit1	ST	Input/output port pin or Timer1 oscillator input	Yes
RC2/CCP1	bit2	ST	Input/output port pin or Capture1 input/Compare1 output/PWM1 output	No
RC3/SCK/SCL	bit3	ST	RC3 can also be the synchronous serial clock for both SPI and I^2C modes.	No
RC4/SDI/SDA	bit4	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).	No
RC5/SDO	bit5	ST	Input/output port pin or Synchronous Serial Port data output	No
RC6	bit6	ST	Input/output port pin	No
RC7	bit7	ST	Input/output port pin	No

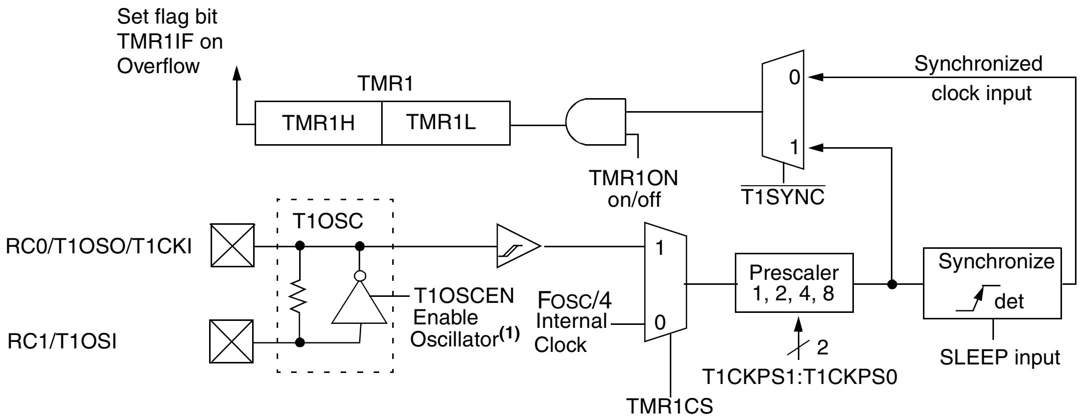
Legend: ST = Schmitt Trigger input

TABLE 3-6 SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged.

FIGURE 5-1: TIMER1 BLOCK DIAGRAM



Note 1: When the T1OSCEN bit is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

 Preliminary

5.2 Timer1 Oscillator

A crystal oscillator circuit is built-in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting control bit T1OSCEN (T1CON<3>). When the Timer1 oscillator is enabled, RC0 and RC1 pins become T1OSO and T1OSI inputs, overriding TRISC<1:0>.

The oscillator is a low power oscillator rated up to 200 kHz. It will continue to run during SLEEP. It is primarily intended for a 32 kHz crystal. Table 5-1 shows the capacitor selection for the Timer1 oscillator.

The Timer1 oscillator is identical to the LP oscillator. The user must provide a software time delay to ensure proper oscillator start-up.

TABLE 5-1 CAPACITOR SELECTION FOR THE TIMER1 OSCILLATOR

Osc Type	Freq	C1	C2
LP	32 kHz	33 pF	33 pF
	100 kHz	15 pF	15 pF
	200 kHz	15 pF	15 pF
These values are for design guidance only.			
Crystals Tested:			
32.768 kHz	Epson C-001R32.768K-A	± 20 PPM	
100 kHz	Epson C-2 100.00 KC-P	± 20 PPM	
200 kHz	STD XTL 200.000 kHz	± 20 PPM	

Note 1: Higher capacitance increases the stability of oscillator but also increases the start-up time.

2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

5.3 Timer1 Interrupt

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

5.4 Resetting Timer1 using a CCP Trigger Output

If the CCP module is configured in compare mode to generate a “special event trigger" (CCP1M3:CCP1M0 = 1011), this signal will reset Timer1 and start an A/D conversion (if the A/D module is enabled).

> **Note:** The special event trigger from the CCP1 module will not set interrupt flag bit TMR1IF (PIR1<0>).

Timer1 must be configured for either timer or synchronized counter mode to take advantage of this feature. If Timer1 is running in asynchronous counter mode, this reset operation may not work.

In the event that a write to Timer1 coincides with a special event trigger from CCP1, the write will take precedence.

In this mode of operation, the CCPR1H:CCPR1L registers pair effectively becomes the period register for Timer1.

TABLE 5-2 REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other resets
0Bh,8Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	ADIF	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF	-0-- 0000	-0-- 0000
8Ch	PIE1	—	ADIE	—	—	SSPIE	CCP1IE	TMR2IE	TMR1IE	-0-- 0000	-0-- 0000
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by the Timer1 module.

7.3 PWM Mode

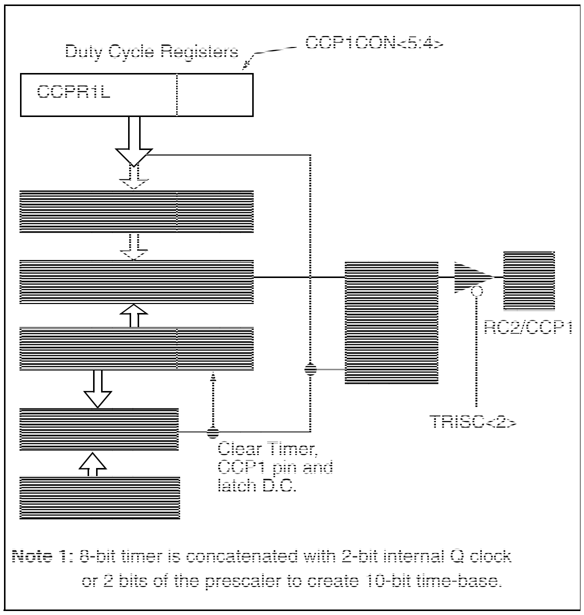
In Pulse Width Modulation (PWM) mode, the CCP1 pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

Note:	Clearing the CCP1CON register will force the CCP1 PWM output latch to the default low level. This is not the PORTC I/O data latch.

Figure 7-3 shows a simplified block diagram of the CCP module in PWM mode.

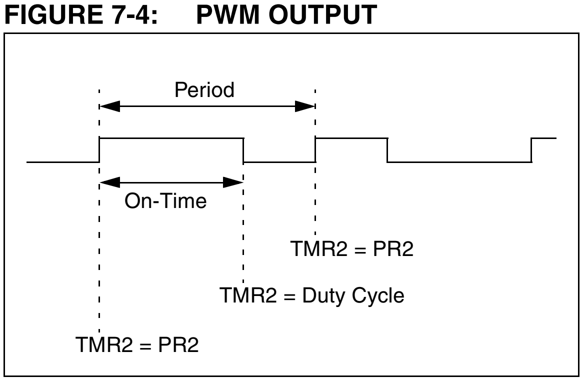
For a step by step procedure on how to set up the CCP module for PWM operation, see Section 7.3.3.

FIGURE 7-3: SIMPLIFIED PWM BLOCK DIAGRAM



A PWM output (Figure 7-4) has a time base (period) and a time that the output stays high (on-time). The frequency of the PWM is the inverse of the period (1/period).

FIGURE 7-4: PWM OUTPUT



7.3.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

$$\text{PWM period} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 prescale value})$$

PWM frequency is defined as 1 / [PWM period].

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H

Note:	The Timer2 postscaler (see Section 6.0) is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

7.3.2 PWM ON-TIME

The PWM on-time is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. CCPR1L contains eight MSbs and CCP1CON<5:4> contains two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

$$\text{PWM on-time} = (\text{CCPR1L:CCP1CON}<5:4>) \cdot T_{OSC} \cdot (\text{TMR2 prescale value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the on-time value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double buffer the PWM on-time. This double buffering is essential for glitchless PWM operation.

When the CCPR1H and 2-bit latch match TMR2 concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.

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

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

Note:	If the PWM on-time value is larger than the PWM period, the CCP1 pin will not be cleared.

For an example PWM period and on-time calculation, see the PIC® MCU Mid-Range Reference Manual, (DS33023).

NOTES:

8.0 SYNCHRONOUS SERIAL PORT (SSP) MODULE

8.1 SSP Module Overview

The Synchronous Serial Port (SSP) 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 SSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I^2C)

For more information on SSP operation (including an I^2C Overview), refer to the PIC® MCU Mid-Range Reference Manual, (DS33023). Also, refer to Application Note AN578, *"Use of the SSP Module in the I^2C Multi-Master Environment."*

8.2 SPI Mode

This section contains register definitions and operational characteristics of the SPI module.

Additional information on SPI operation may be found in the PIC® MCU Mid-Range Reference Manual, (DS33023).

8.2.1 OPERATION OF SSP MODULE IN SPI MODE

A block diagram of the SSP Module in SPI Mode is shown in Figure 8-1.

The SPI mode allows 8-bits of data to be synchronously transmitted and received simultaneously. To accomplish communication, three pins are used:

- Serial Data Out (SDO)RC5/SDO
- Serial Data In (SDI)RC4/SDI/SDA
- Serial Clock (SCK)RC3/SCK/SCL

Additionally, a fourth pin may be used when in a slave mode of operation:

- Slave Select ($\overline{SS}$)RA5/$\overline{SS}$/AN4

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits in the SSPCON register (SSPCON<5:0>) and SSPSTAT<7:6>. These control bits allow the following to be specified:

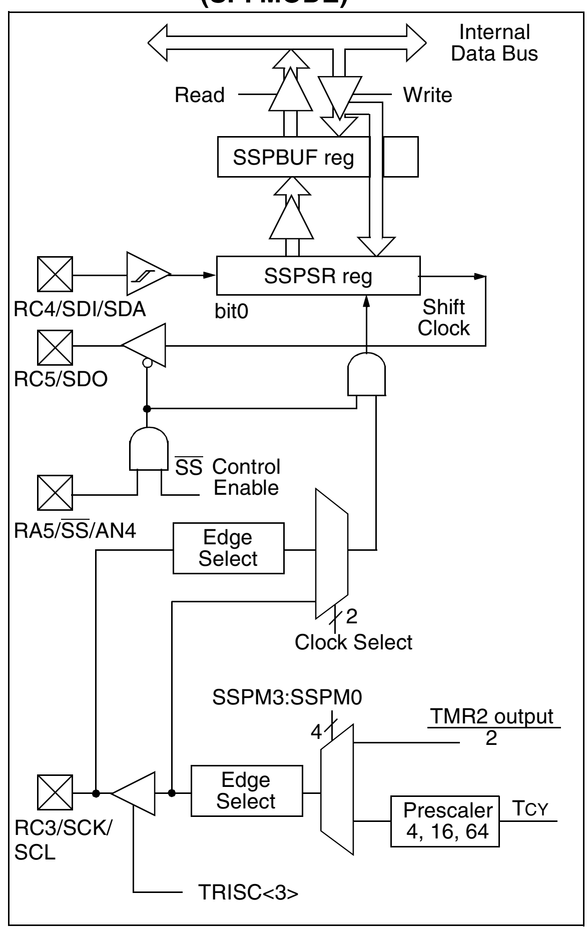
- Master Operation (SCK is the clock output)
- Slave Mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Clock Edge (Output data on rising/falling edge of SCK)
- Clock Rate (master operation only)
- Slave Select Mode (Slave mode only)

To enable the serial port, SSP Enable bit, SSPEN (SSPCON<5>) must be set. To reset or reconfigure SPI mode, clear bit SSPEN, re-initialize the SSPCON register, and then set bit SSPEN. This configures the SDI, SDO, SCK and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, they must have their data direction bits (in the TRISC register) appropriately programmed. That is:

- SDI must have TRISC<4> set
- SDO must have TRISC<5> cleared
- SCK (master operation) must have TRISC<3> cleared
- SCK (Slave mode) must have TRISC<3> set
- $\overline{SS}$ must have TRISA<5> set (if used)

Note:	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}.
Note:	If the SPI is used in Slave Mode with CKE = '1', then the $\overline{SS}$ pin control must be enabled.

FIGURE 8-1: SSP BLOCK DIAGRAM (SPI MODE)



NOTES:

9.4 A/D Conversions

Note:	The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D.

9.5 Use of the CCP Trigger

An A/D conversion can be started by the "special event trigger" of the CCP1 module. This requires that the CCP1M3:CCP1M0 bits (CCP1CON<3:0>) be programmed as `1011` and that the A/D module be enabled (ADON bit is set). When the trigger occurs, the GO/$\overline{\text{DONE}}$ bit will be set, starting the A/D conversion, and the Timer1 counter will be reset to zero. Timer1 is reset to automatically repeat the A/D acquisition period with minimal software overhead. The appropriate analog input channel must be selected and the minimum acquisition time must pass before the "special event trigger" sets the GO/$\overline{\text{DONE}}$ bit (starts a conversion).

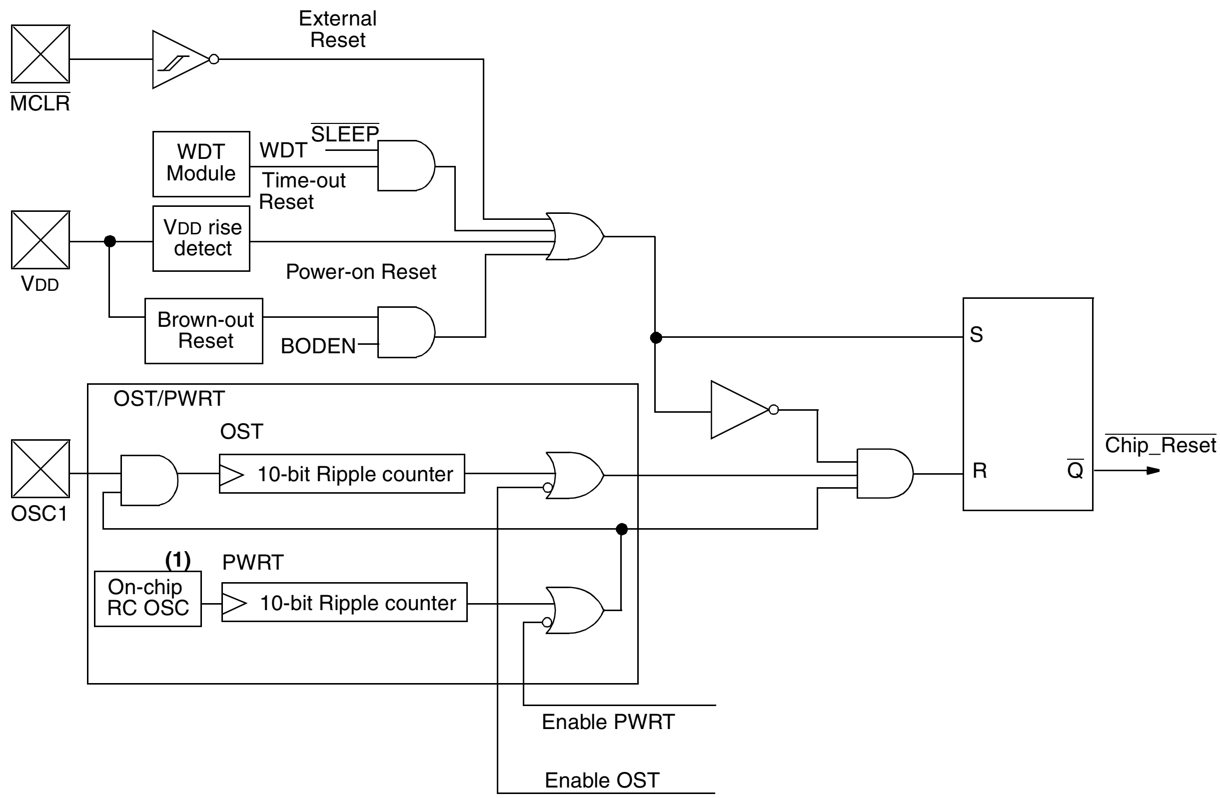
If the A/D module is not enabled (ADON is cleared), then the "special event trigger" will be ignored by the A/D module, but will still reset the Timer1 counter.

TABLE 9-2 SUMMARY OF A/D REGISTERS

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
0Bh,8Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	ADIF	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF	-0-- 0000	-0-- 0000
8Ch	PIE1	—	ADIE	—	—	SSPIE	CCP1IE	TMR2IE	TMR1IE	-0-- 0000	-0-- 0000
1Eh	ADRES	A/D Result Register								xxxx xxxx	uuuu uuuu
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON	0000 00-0	0000 00-0
9Fh	ADCON1	—	—	—	—	—	PCFG2	PCFG1	PCFG0	---- -000	---- -000
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used for A/D conversion.

FIGURE 10-5: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



Note 1: This is a separate oscillator from the RC oscillator of the CLKIN pin.

 Preliminary

and test the sample code. In addition, PICDEM-17 supports down-loading of programs to and executing out of external FLASH memory on board. The PICDEM-17 is also usable with the MPLAB-ICE or PICMASTER emulator, and all of the sample programs can be run and modified using either emulator. Additionally, a generous prototype area is available for user hardware.

12.17 SEEVAL Evaluation and Programming System

The SEEVAL SEEPROM Designer's Kit supports all Microchip 2-wire and 3-wire Serial EEPROMs. The kit includes everything necessary to read, write, erase or program special features of any Microchip SEEPROM product including Smart Serials™ and secure serials. The Total Endurance™ Disk is included to aid in trade-off analysis and reliability calculations. The total kit can significantly reduce time-to-market and result in an optimized system.

12.18 KEELOQ Evaluation and Programming Tools

KEELOQ evaluation and programming tools support Microchips HCS Secure Data Products. The HCS evaluation kit includes an LCD display to show changing codes, a decoder to decode transmissions, and a programming interface to program test transmitters.

NOTES:

FIGURE 13-3: PIC16C62B/72A-04 VOLTAGE-FREQUENCY GRAPH

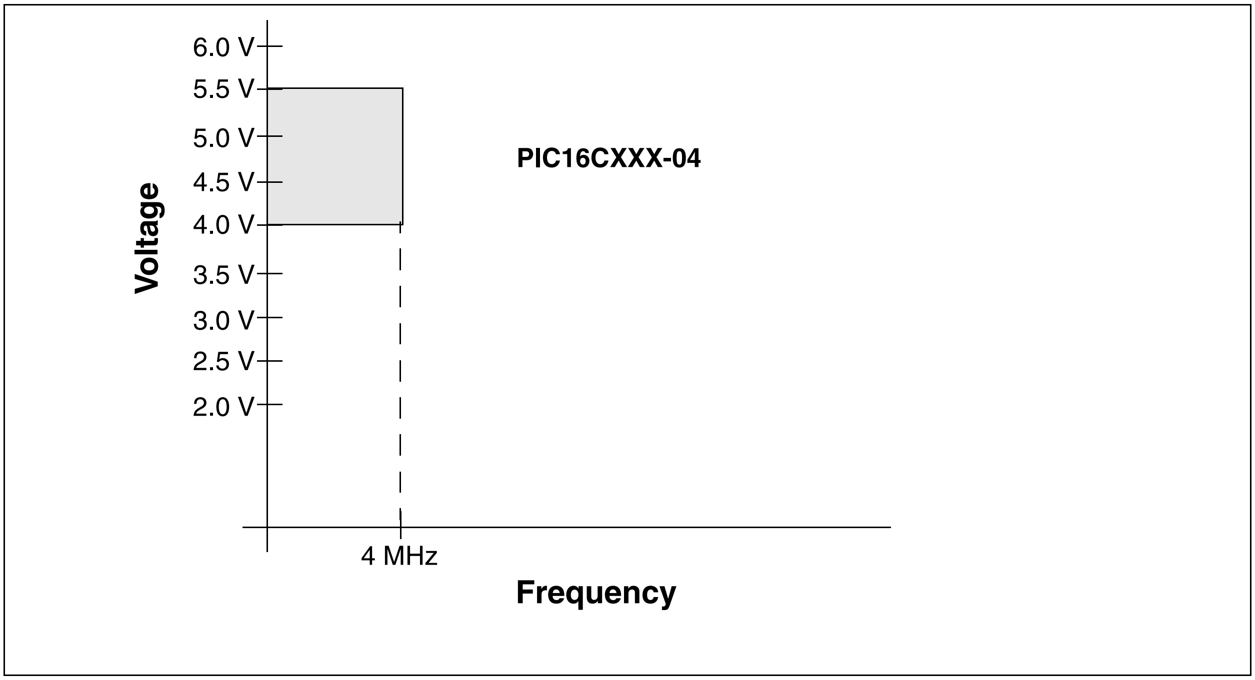


FIGURE 13-11: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)

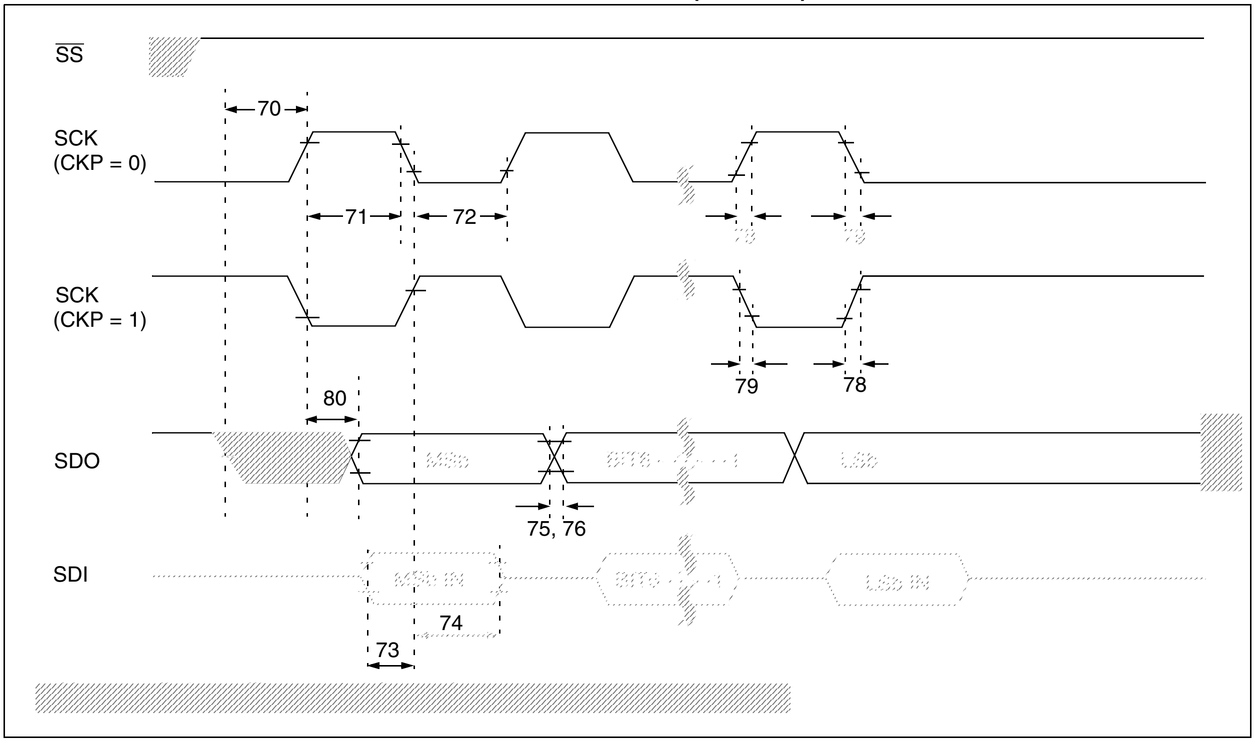


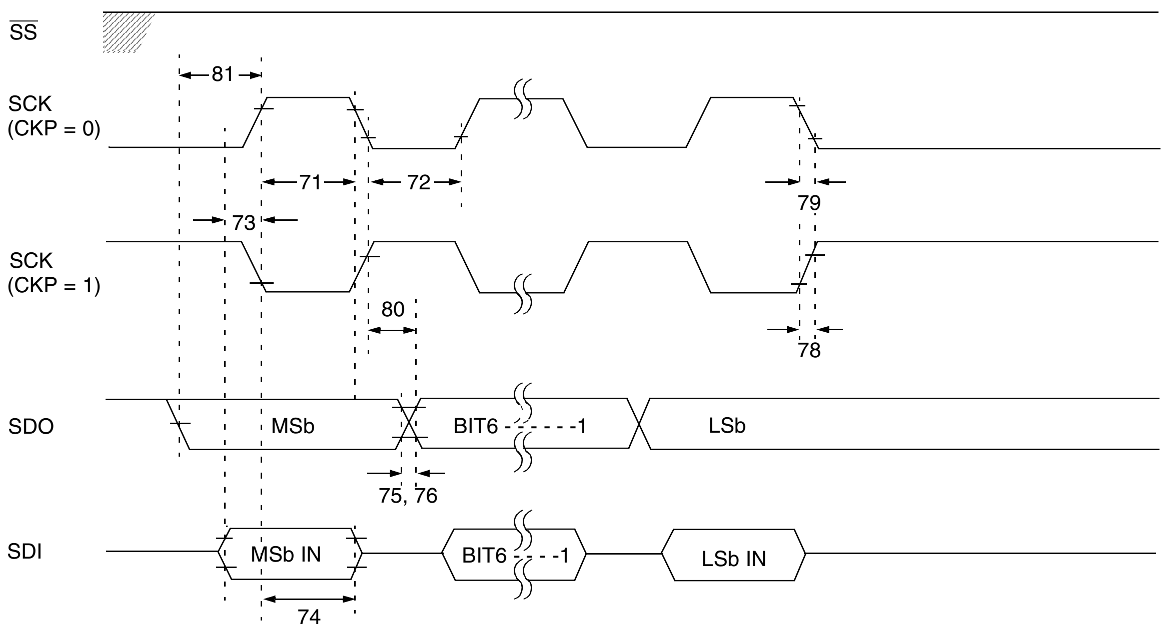
TABLE 13-7: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)

Param. No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS}$↓ to SCK↓ or SCK↑ input		TCY	—	—	ns	
71	TscH	SCK input high time (slave mode)	Continuous	1.25TCY + 30	—	—	ns	
71A			Single Byte	40	—	—	ns	Note 1
72	TscL	SCK input low time (slave mode)	Continuous	1.25TCY + 30	—	—	ns	
72A			Single Byte	40	—	—	ns	Note 1
73	TdiV2scH, TdiV2scL	Setup time of SDI data input to SCK edge		100	—	—	ns	
73A	TB2B	Last clock edge of Byte1 to the 1st clock edge of Byte2		1.5TCY + 40	—	—	ns	Note 1
74	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge		100	—	—	ns	
75	TdoR	SDO data output rise time	**PIC16CXX**	—	10	25	ns	
			PIC16LCXX	—	20	45	ns	
76	TdoF	SDO data output fall time		—	10	25	ns	
78	TscR	SCK output rise time (master mode)	**PIC16CXX**	—	10	25	ns	
			PIC16LCXX	—	20	45	ns	
79	TscF	SCK output fall time (master mode)		—	10	25	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	**PIC16CXX**	—	—	50	ns	
			PIC16LCXX	—	—	100	ns	

† Data in “Typ” column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Specification 73A is only required if specifications 71A and 72A are used.

FIGURE 13-12: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)



Note: Refer to Figure 13-4 for load conditions.

TABLE 13-8: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)

Param. No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
71	TscH	SCK input high time (slave mode)	Continuous	1.25TCY + 30	—	—	ns	
71A			Single Byte	40	—	—	ns	Note 1
72	TscL	SCK input low time (slave mode)	Continuous	1.25TCY + 30	—	—	ns	
72A			Single Byte	40	—	—	ns	Note 1
73	TdiV2scH, TdiV2scL	Setup time of SDI data input to SCK edge		100	—	—	ns	
73A	TB2B	Last clock edge of Byte1 to the 1st clock edge of Byte2		1.5TCY + 40	—	—	ns	Note 1
74	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge		100	—	—	ns	
75	TdoR	SDO data output rise time	**PIC16CXX**	—	10	25	ns	
			PIC16LCXX		20	45	ns	
76	TdoF	SDO data output fall time		—	10	25	ns	
78	TscR	SCK output rise time (master mode)	**PIC16CXX**	—	10	25	ns	
			PIC16LCXX		20	45	ns	
79	TscF	SCK output fall time (master mode)		—	10	25	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	**PIC16CXX**	—	—	50	ns	
			PIC16LCXX		—	100	ns	
81	TdoV2scH, TdoV2scL	SDO data output setup to SCK edge		TCY	—	—	ns	

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Specification 73A is only required if specifications 71A and 72A are used.

FIGURE 13-13: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)

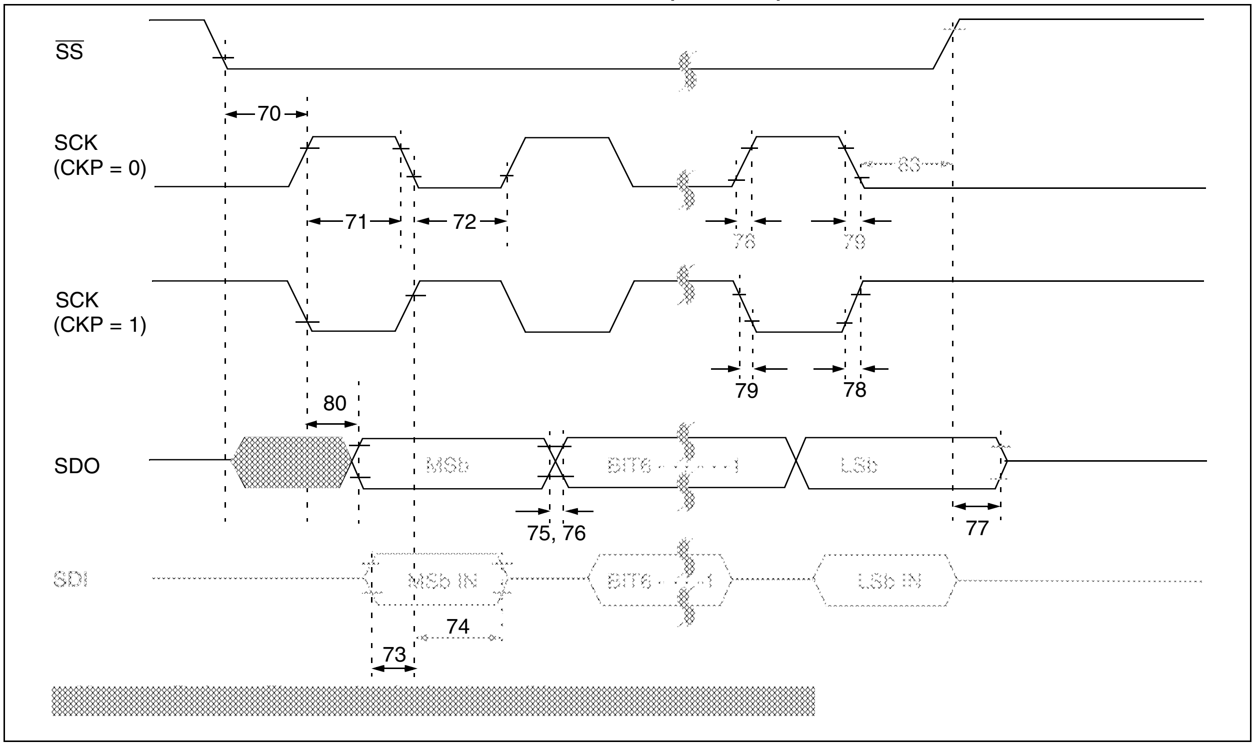


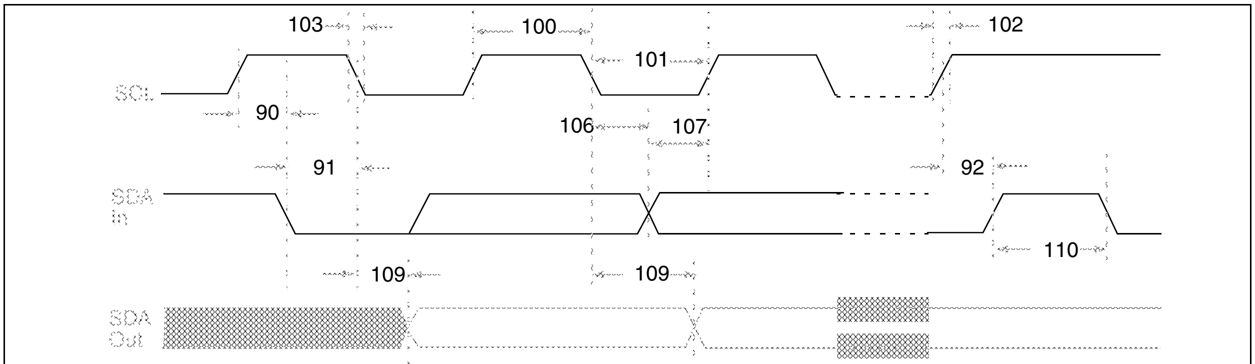
TABLE 13-9: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING (CKE = 0)

Param. No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS}$↓ to SCK↓ or SCK↑ input		TCY	—	—	ns	
71	TscH	SCK input high time (slave mode)	Continuous	1.25TCY + 30	—	—	ns	
71A			Single Byte	40	—	—	ns	Note 1
72	TscL	SCK input low time (slave mode)	Continuous	1.25TCY + 30	—	—	ns	
72A			Single Byte	40	—	—	ns	Note 1
73	TdiV2scH, TdiV2scL	Setup time of SDI data input to SCK edge		100	—	—	ns	
73A	TB2B	Last clock edge of Byte1 to the 1st clock edge of Byte2		1.5TCY + 40	—	—	ns	Note 1
74	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge		100	—	—	ns	
75	TdoR	SDO data output rise time	**PIC16CXX**	—	10	25	ns	
			PIC16LCXX		20	45	ns	
76	TdoF	SDO data output fall time		—	10	25	ns	
77	TssH2doZ	$\overline{SS}$↑ to SDO output hi-impedance		10	—	50	ns	
78	TscR	SCK output rise time (master mode)	**PIC16CXX**	—	10	25	ns	
			PIC16LCXX		20	45	ns	
79	TscF	SCK output fall time (master mode)		—	10	25	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	**PIC16CXX**	—	—	50	ns	
			PIC16LCXX		—	100	ns	
83	TscH2ssH, TscL2ssH	$\overline{SS}$ ↑ after SCK edge		1.5TCY + 40	—	—	ns	

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Specification 73A is only required if specifications 71A and 72A are used.

FIGURE 13-16: I^2C BUS DATA TIMING



Note: Refer to Figure 13-4 for load conditions.

TABLE 13-12: I^2C BUS DATA REQUIREMENTS

Param. No.	Sym	Characteristic		Min	Max	Units	Conditions
100*	THIGH	Clock high time	100 kHz mode	4.0	—	μs	Device must operate at a minimum of 1.5 MHz
			400 kHz mode	0.6	—	μs	Device must operate at a minimum of 10 MHz
			SSP Module	1.5TCY	—		
101*	TLOW	Clock low time	100 kHz mode	4.7	—	μs	Device must operate at a minimum of 1.5 MHz
			400 kHz mode	1.3	—	μs	Device must operate at a minimum of 10 MHz
			SSP Module	1.5TCY	—		
102*	TR	SDA and SCL rise time	100 kHz mode	—	1000	ns	
			400 kHz mode	20 + 0.1Cb	300	ns	Cb is specified to be from 10-400 pF
103*	TF	SDA and SCL fall time	100 kHz mode	—	300	ns	
			400 kHz mode	20 + 0.1Cb	300	ns	Cb is specified to be from 10-400 pF
90*	TSU:STA	START condition setup time	100 kHz mode	4.7	—	μs	Only relevant for repeated START condition
			400 kHz mode	0.6	—	μs	
91*	THD:STA	START condition hold time	100 kHz mode	4.0	—	μs	After this period the first clock pulse is generated
			400 kHz mode	0.6	—	μs	
106*	THD:DAT	Data input hold time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	μs	
107*	TSU:DAT	Data input setup time	100 kHz mode	250	—	ns	Note 2
			400 kHz mode	100	—	ns	
92*	TSU:STO	STOP condition setup time	100 kHz mode	4.7	—	μs	
			400 kHz mode	0.6	—	μs	
109*	TAA	Output valid from clock	100 kHz mode	—	3500	ns	Note 1
			400 kHz mode	—	—	ns	
110*	TBUF	Bus free time	100 kHz mode	4.7	—	μs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	μs	
	Cb	Bus capacitive loading		—	400	pF	

* These parameters are characterized but not tested.

Note 1: As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of START or STOP conditions.

2: A fast-mode (400 kHz) I^2C-bus device can be used in a standard-mode (100 kHz) I^2C-bus system, but the requirement Tsu:DAT ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line TR max.+tsu;DAT = 1000 + 250 = 1250 ns (according to the standard-mode I^2C bus specification) before the SCL line is released.

NOTES:

APPENDIX C: MIGRATION FROM BASE-LINE TO MID-RANGE DEVICES

This section discusses how to migrate from a baseline device (i.e., PIC16C5X) to a mid-range device (i.e., PIC16CXXX).

The following are the list of modifications over the PIC16C5X microcontroller family:

1. Instruction word length is increased to 14-bits. This allows larger page sizes both in program memory (2K now as opposed to 512 before) and register file (128 bytes now versus 32 bytes before).
2. A PC high latch register (PCLATH) is added to handle program memory paging. Bits PA2, PA1, PA0 are removed from STATUS register.
3. Data memory paging is redefined slightly. STATUS register is modified.
4. Four new instructions have been added: `RETURN`, `RETFIE`, `ADDLW`, and `SUBLW`. Two instructions `TRIS` and `OPTION` are being phased out although they are kept for compatibility with PIC16C5X.
5. OPTION_REG and TRIS registers are made addressable.
6. Interrupt capability is added. Interrupt vector is at 0004h.
7. Stack size is increased to 8 deep.
8. Reset vector is changed to 0000h.
9. Reset of all registers is revisited. Five different reset (and wake-up) types are recognized. Registers are reset differently.
10. Wake up from SLEEP through interrupt is added.
11. Two separate timers, Oscillator Start-up Timer (OST) and Power-up Timer (PWRT) are included for more reliable power-up. These timers are invoked selectively to avoid unnecessary delays on power-up and wake-up.
12. PORTB has weak pull-ups and interrupt on change feature.
13. T0CKI pin is also a port pin (RA4) now.
14. FSR is made a full eight bit register.
15. “In-circuit serial programming” is made possible. The user can program PIC16CXX devices using only five pins: V_{DD}, V_{SS}, $\overline{MCLR}/V_{PP}$, RB6 (clock) and RB7 (data in/out).
16. PCON status register is added with a Power-on Reset status bit ($\overline{POR}$).
17. Code protection scheme is enhanced such that portions of the program memory can be protected, while the remainder is unprotected.
18. Brown-out protection circuitry has been added. Controlled by configuration word bit BODEN. Brown-out reset ensures the device is placed in a reset condition if V_{DD} dips below a fixed setpoint.

To convert code written for PIC16C5X to PIC16CXXX, the user should take the following steps:

1. Remove any program memory page select operations (PA2, PA1, PA0 bits) for `CALL`, `GOTO`.
2. Revisit any computed jump operations (write to PC or add to PC, etc.) to make sure page bits are set properly under the new scheme.
3. Eliminate any data memory page switching. Redefine data variables to reallocate them.
4. Verify all writes to STATUS, OPTION, and FSR registers since these have changed.
5. Change reset vector to 0000h.