Microchip Technology - PIC17C762T-33I/L Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

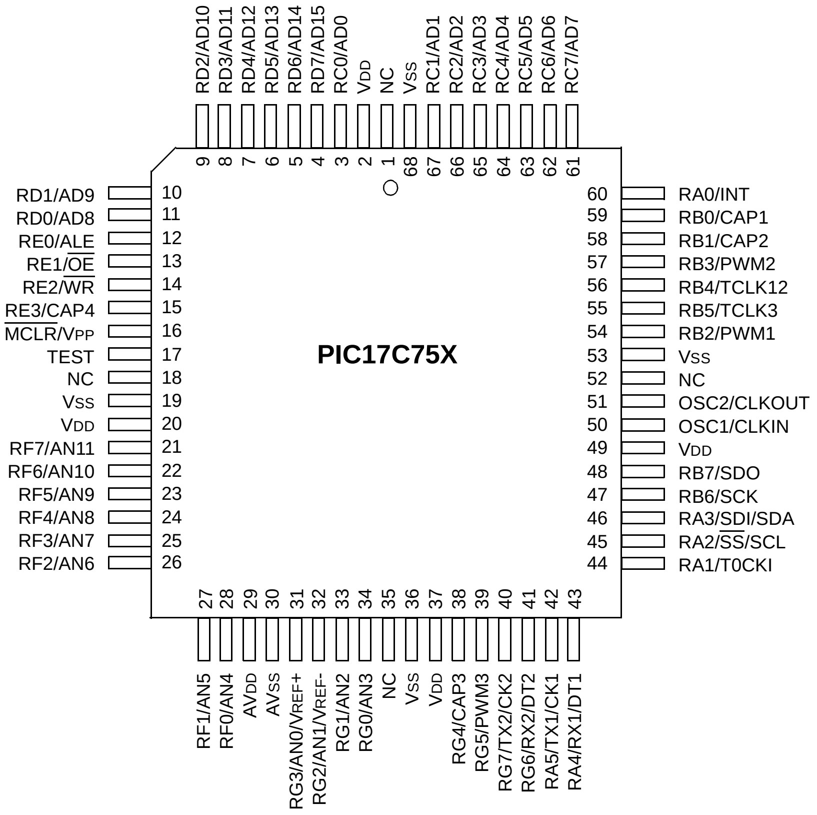
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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"

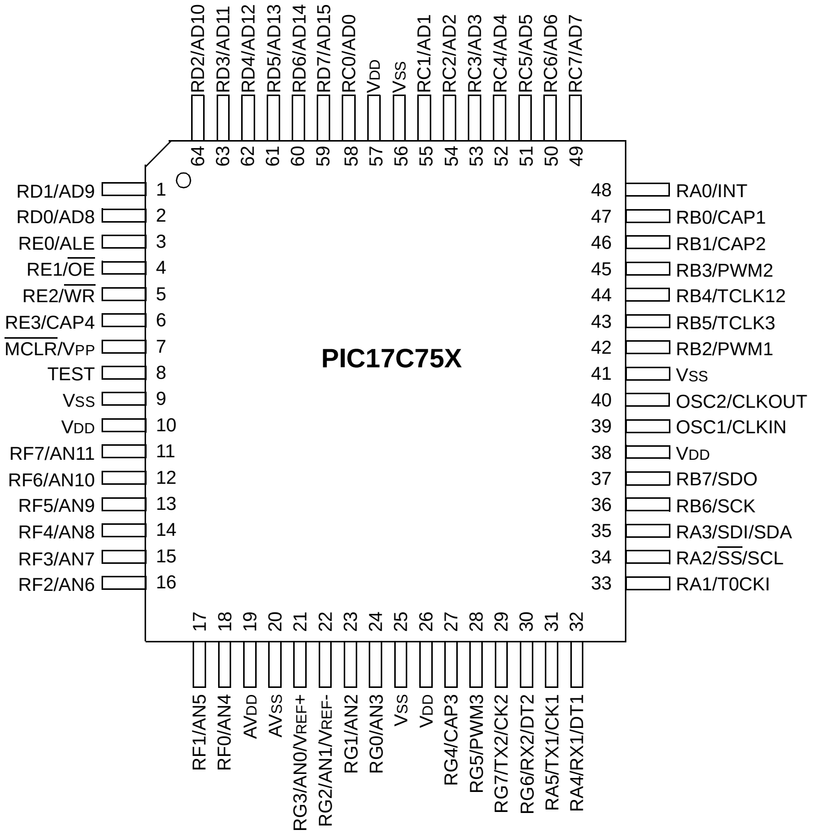
Details	
Product Status	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	33MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	66
Program Memory Size	16KB (8K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	678 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	84-LCC (J-Lead)
Supplier Device Package	84-PLCC (29.31x29.31)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c762t-33i-l

Pin Diagrams cont.’d

68-Pin PLCC



64-Pin TQFP



3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC17CXXX can be attributed to a number of architectural features, commonly found in RISC microprocessors. To begin with, the PIC17CXXX uses a modified Harvard architecture. This architecture has the program and data accessed from separate memories. So, the device has a program memory bus and a data memory bus. This improves bandwidth over traditional von Neumann architecture, where program and data are fetched from the same memory (accesses over the same bus). Separating program and data memory further allows instructions to be sized differently than the 8-bit wide data word. PIC17CXXX opcodes are 16-bits wide, enabling single word instructions. The full 16-bit wide program memory bus fetches a 16-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions execute in a single cycle (121 ns @ 33 MHz), except for program branches and two special instructions that transfer data between program and data memory.

The PIC17CXXX can address up to 64K x 16 of program memory space.

The **PIC17C752** and **PIC17C762** integrate 8K x 16 of EPROM program memory on-chip.

The **PIC17C756A** and **PIC17C766** integrate 16K x 16 EPROM program memory on-chip.

A simplified block diagram is shown in Figure 3-1. The descriptions of the device pins are listed in Table 3-1.

Program execution can be internal only (Microcontroller or Protected Microcontroller mode), external only (Microprocessor mode), or both (Extended Microcontroller mode). Extended Microcontroller mode does not allow code protection.

The PIC17CXXX can directly or indirectly address its register files or data memory. All special function registers, including the Program Counter (PC) and Working Register (WREG), are mapped in data memory. The PIC17CXXX has an orthogonal (symmetrical) instruction set that makes it possible to carry out any operation on any register using any addressing mode. This symmetrical nature and lack of 'special optimal situations' make programming with the PIC17CXXX simple, yet efficient. In addition, the learning curve is reduced significantly.

One of the PIC17CXXX family architectural enhancements from the PIC16CXX family, allows two file registers to be used in some two operand instructions. This allows data to be moved directly between two registers without going through the WREG register, thus increasing performance and decreasing program memory usage.

The PIC17CXXX devices contain an 8-bit ALU and working register. The ALU is a general purpose arithmetic unit. It performs arithmetic and Boolean functions between data in the working register and any register file.

The WREG register is an 8-bit working register used for ALU operations.

All PIC17CXXX devices have an 8 x 8 hardware multiplier. This multiplier generates a 16-bit result in a single cycle.

The ALU is 8-bits wide and capable of addition, subtraction, shift and logical operations. Unless otherwise mentioned, arithmetic operations are two's complement in nature.

Depending on the instruction executed, the ALU may affect the values of the Carry (C), Digit Carry (DC), Zero (Z) and Overflow (OV) bits in the ALUSTA register. The C and DC bits operate as a borrow and digit borrow out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

Signed arithmetic is comprised of a magnitude and a sign bit. The overflow bit indicates if the magnitude overflows and causes the sign bit to change state. That is, if the result of 8-bit signed operations is greater than 127 (7Fh), or less than -128 (80h).

Signed math can have greater than 7-bit values (magnitude), if more than one byte is used. The overflow bit only operates on bit6 (MSb of magnitude) and bit7 (sign bit) of each byte value in the ALU. That is, the overflow bit is not useful if trying to implement signed math where the magnitude, for example, is 11-bits.

If the signed math values are greater than 7-bits (such as 15-, 24-, or 31-bit), the algorithm must ensure that the low order bytes of the signed value ignore the overflow status bit.

Example 3-1 shows two cases of doing signed arithmetic. The Carry (C) bit and the Overflow (OV) bit are the most important status bits for signed math operations.

EXAMPLE 3-1: 8-BIT MATH ADDITION

Hex Value	Signed Values	Unsigned Values
FFh	-1	255
+ 01h	+ 1	+ 1
= 00h	= 0 (FEh)	= 256 → 00h
C bit = 1	C bit = 1	C bit = 1
OV bit = 0	OV bit = 0	OV bit = 0
DC bit = 1	DC bit = 1	DC bit = 1
Z bit = 1	Z bit = 1	Z bit = 1
Hex Value	**Signed Values**	**Unsigned Values**
7Fh	127	127
+ 01h	+ 1	+ 1
= 80h	= 128 → 00h	= 128
C bit = 0	C bit = 0	C bit = 0
OV bit = 1	OV bit = 1	OV bit = 1
DC bit = 1	DC bit = 1	DC bit = 1
Z bit = 0	Z bit = 0	Z bit = 0

TABLE 7-3: SPECIAL FUNCTION 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
Unbanked											
00h	INDF0	Uses contents of FSR0 to address Data Memory (not a physical register)								`---- ----`	`---- ----`
01h	FSR0	Indirect Data Memory Address Pointer 0								`xxxx xxxx`	`uuuu uuuu`
02h	PCL	Low order 8-bits of PC								`0000 0000`	`0000 0000`
03h[1]	PCLATH	Holding Register for upper 8-bits of PC								`0000 0000`	`uuuu uuuu`
04h	ALUSTA	FS3	FS2	FS1	FS0	OV	Z	DC	C	`1111 xxxx`	`1111 uuuu`
05h	T0STA	INTEDG	T0SE	T0CS	T0PS3	T0PS2	T0PS1	T0PS0	—	`0000 000-`	`0000 000-`
06h[2]	CPUSTA	—	—	STKAV	GLINTD	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	`--11 11qq`	`--11 qquu`
07h	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	`0000 0000`	`0000 0000`
08h	INDF1	Uses contents of FSR1 to address Data Memory (not a physical register)								`---- ----`	`---- ----`
09h	FSR1	Indirect Data Memory Address Pointer 1								`xxxx xxxx`	`uuuu uuuu`
0Ah	WREG	Working Register								`xxxx xxxx`	`uuuu uuuu`
0Bh	TMR0L	TMR0 Register; Low Byte								`xxxx xxxx`	`uuuu uuuu`
0Ch	TMR0H	TMR0 Register; High Byte								`xxxx xxxx`	`uuuu uuuu`
0Dh	TBLPTRL	Low Byte of Program Memory Table Pointer								`0000 0000`	`0000 0000`
0Eh	TBLPTRH	High Byte of Program Memory Table Pointer								`0000 0000`	`0000 0000`
0Fh	BSR	Bank Select Register								`0000 0000`	`0000 0000`
Bank 0											
10h	PORTA[4,6]	RBPU	—	RA5/TX1/ CK1	RA4/RX1/ DT1	RA3/SDI/ SDA	RA2/$\overline{\text{SS}}$/ SCL	RA1/T0CKI	RA0/INT	`0-xx 11xx`	`0-uu 11uu`
11h	DDRB	Data Direction Register for PORTB								`1111 1111`	`1111 1111`
12h	PORTB[4]	RB7/ SDO	RB6/ SCK	RB5/ TCLK3	RB4/ TCLK12	RB3/ PWM2	RB2/ PWM1	RB1/ CAP2	RB0/ CAP1	`xxxx xxxx`	`uuuu uuuu`
13h	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	`0000 -00x`	`0000 -00u`
14h	RCREG1	Serial Port Receive Register								`xxxx xxxx`	`uuuu uuuu`
15h	TXSTA1	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	`0000 --1x`	`0000 --1u`
16h	TXREG1	Serial Port Transmit Register (for USART1)								`xxxx xxxx`	`uuuu uuuu`
17h	SPBRG1	Baud Rate Generator Register (for USART1)								`0000 0000`	`0000 0000`
Bank 1											
10h	DDRC[5]	Data Direction Register for PORTC								`1111 1111`	`1111 1111`
11h	PORTC[4,5]	RC7/AD7	RC6/AD6	RC5/AD5	RC4/AD4	RC3/AD3	RC2/AD2	RC1/AD1	RC0/AD0	`xxxx xxxx`	`uuuu uuuu`
12h	DDRD[5]	Data Direction Register for PORTD								`1111 1111`	`1111 1111`
13h	PORTD[4,5]	RD7/ AD15	RD6/ AD14	RD5/ AD13	RD4/ AD12	RD3/ AD11	RD2/ AD10	RD1/AD9	RD0/AD8	`xxxx xxxx`	`uuuu uuuu`
14h	DDRE[5]	Data Direction Register for PORTE								`---- 1111`	`---- 1111`
15h	PORTE[4,5]	—	—	—	—	RE3/ CAP4	RE2/$\overline{\text{WR}}$	RE1/$\overline{\text{OE}}$	RE0/ALE	`---- xxxx`	`---- uuuu`
16h	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	`x000 0010`	`u000 0010`
17h	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	`0000 0000`	`0000 0000`

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:

NOTES:

NOTES:

TABLE 14-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

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 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
13h, Bank 0	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h, Bank 0	RCREG1	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	xxxx xxxx	uuuu uuuu
15h, Bank 0	TXSTA1	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 0	SPBRG1	Baud Rate Generator Register								0000 0000	0000 0000
10h, Bank 4	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	000- 0010	000- 0010
11h, Bank 4	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	000- 0000	000- 0000
13h, Bank 4	RCSTA2	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h, Bank 4	RCREG2	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	xxxx xxxx	uuuu uuuu
15h, Bank 4	TXSTA2	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 4	SPBRG2	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as a '0'. Shaded cells are not used for synchronous master reception.

REGISTER 15-2: SSPCON1: SYNC SERIAL PORT CONTROL REGISTER1 (ADDRESS 11h, 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
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	$\overline{\text{SSPM1}}$	$\overline{\text{SSPM0}}$
bit 7							bit 0

bit 7 **WCOL**: Write Collision Detect bit

Master mode:
1 = A write to the SSPBUF register was attempted while the I^{2}C conditions were not valid for a transmission to be started
0 = No collision

Slave mode:
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPOV**: Receive Overflow Indicator bit

In SPI mode:
1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. In Slave mode, the user must read the SSPBUF, even if only transmitting data, to avoid setting overflow. In Master mode, the overflow bit is not set, since each new reception (and transmission) is initiated by writing to the SSPBUF register. (Must be cleared in software.)
0 = No overflow

In I^{2}C mode:
1 = A byte is received while the SSPBUF register is still holding the previous byte. SSPOV is a “don’t care” in Transmit mode. (Must be cleared in software.)
0 = No overflow

bit 5 **SSPEN**: Synchronous Serial Port Enable bit
In both modes, when enabled, these pins must be properly configured as input or output.

In SPI mode:
1 = Enables serial port and configures SCK, SDO, SDI and $\overline{\text{SS}}$ as the source of the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

In I^{2}C mode:
1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

Note: In SPI mode, these pins must be properly configured as input or output.

bit 4 **CKP**: Clock Polarity Select bit

In SPI mode:
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level

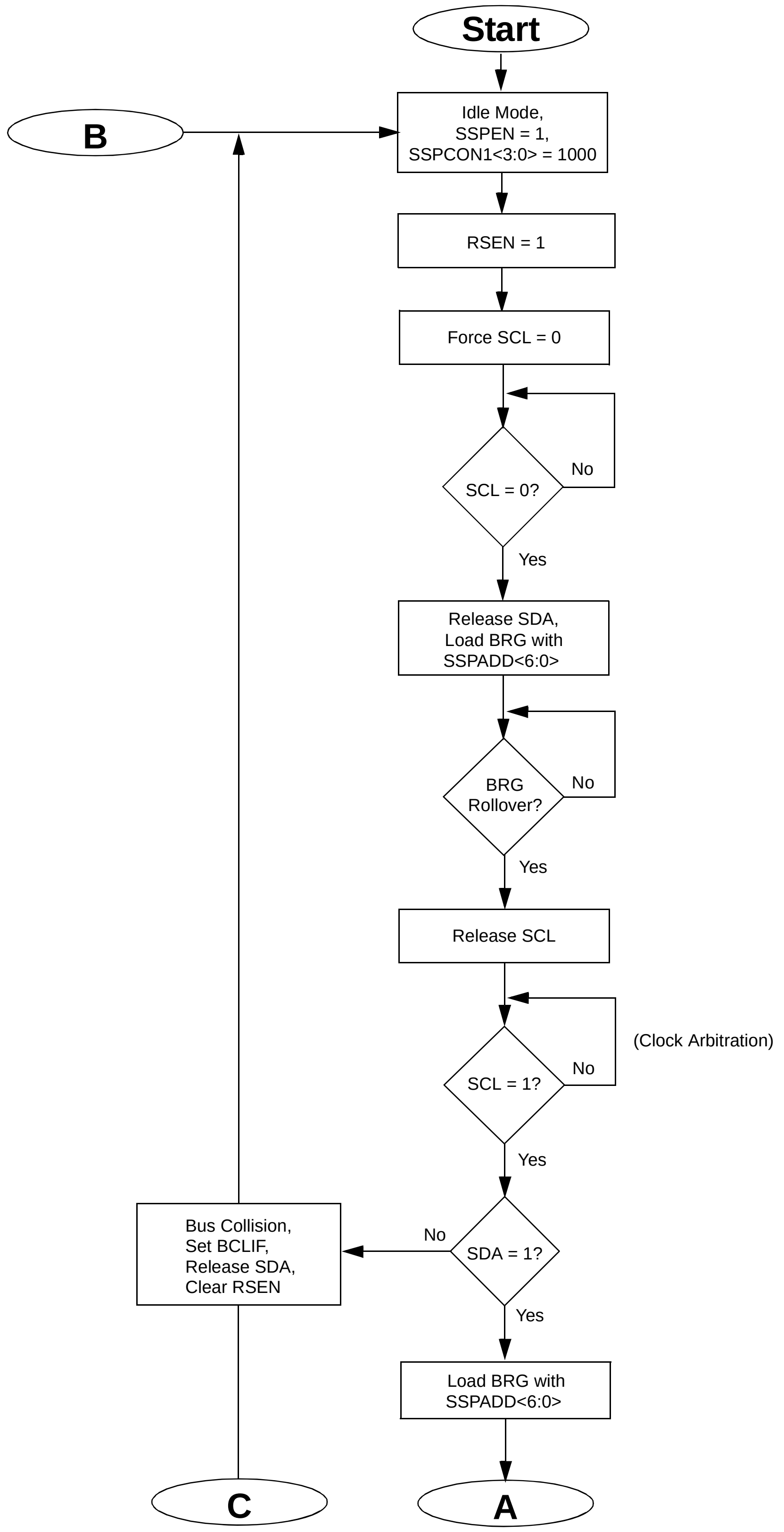
In I^{2}C Slave mode:
SCK release control
1 = Enable clock
0 = Holds clock low (clock stretch). (Used to ensure data setup time.)

In I^{2}C Master mode:
Unused in this mode

bit 3-0 **SSPM3:SSPM0**: Synchronous Serial Port Mode Select bits
0000 = SPI Master mode, clock = FOSC/4
0001 = SPI Master mode, clock = FOSC/16
0010 = SPI Master mode, clock = FOSC/64
0011 = SPI Master mode, clock = TMR2 output/2
0100 = SPI Slave mode, clock = SCK pin, $\overline{\text{SS}}$ pin control enabled
0101 = SPI Slave mode, clock = SCK pin, $\overline{\text{SS}}$ pin control disabled, $\overline{\text{SS}}$ can be used as I/O pin
0110 = I^{2}C Slave mode, 7-bit address
0111 = I^{2}C Slave mode, 10-bit address
1000 = I^{2}C Master mode, clock = FOSC / (4 * (SSPADD+1))
1xx1 = Reserved
1x1x = Reserved

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

FIGURE 15-23: REPEATED START CONDITION FLOW CHART (PAGE 1)



15.2.14 STOP CONDITION TIMING

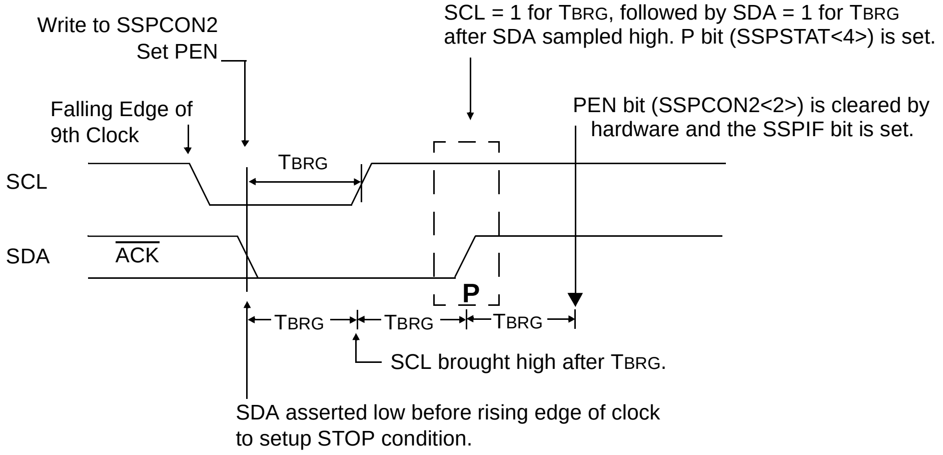
A STOP bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit PEN (SSPCON2<2>). At the end of a receive/transmit the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the baud rate generator is reloaded and counts down to ‘0’. When the baud rate generator times out, the SCL pin will be brought high and one TBRG (baud rate generator rollover count) later, the SDA pin will be de-asserted. When the SDA pin is sampled high while SCL is high, the P bit (SSPSTAT<4>) is set. A TBRG later, the PEN bit is cleared and the SSPIF bit is set (Figure 15-31).

Whenever the firmware decides to take control of the bus, it will first determine if the bus is busy by checking the S and P bits in the SSPSTAT register. If the bus is busy, then the CPU can be interrupted (notified) when a STOP bit is detected (i.e., bus is free).

15.2.14.1 WCOL Status Flag

If the user writes the SSPBUF when a STOP sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write doesn’t occur).

FIGURE 15-31: STOP CONDITION RECEIVE OR TRANSMIT MODE


Write to SSPCON2
Set PEN
SCL = 1 for TBRG, followed by SDA = 1 for TBRG
after SDA sampled high. P bit (SSPSTAT<4>) is set.
PEN bit (SSPCON2<2>) is cleared by
hardware and the SSPIF bit is set.
Falling Edge of
9th Clock
TBRG
SCL
SDA
ACK
P
TBRG
TBRG
TBRG
SCL brought high after TBRG.
SDA asserted low before rising edge of clock
to setup STOP condition.

Note: TBRG = one baud rate generator period.

15.2.18.2 Bus Collision During a Repeated Start Condition

During a Repeated Start condition, a bus collision occurs if:

a) A low level is sampled on SDA when SCL goes from low level to high level.

b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user de-asserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD<6:0> and counts down to '0'. The SCL pin is then de-asserted and when sampled high, the SDA pin is sampled. If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0'). If, however, SDA is sampled high, then the BRG is reloaded and begins counting. If SDA goes from high to low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

If, however, SCL goes from high to low before the BRG times out and SDA has not already been asserted, then a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition.

If, at the end of the BRG time-out, both SCL and SDA are still high, the SDA pin is driven low, the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCL pin, the SCL pin is driven low and the Repeated Start condition is complete (Figure 15-38).

FIGURE 15-38: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

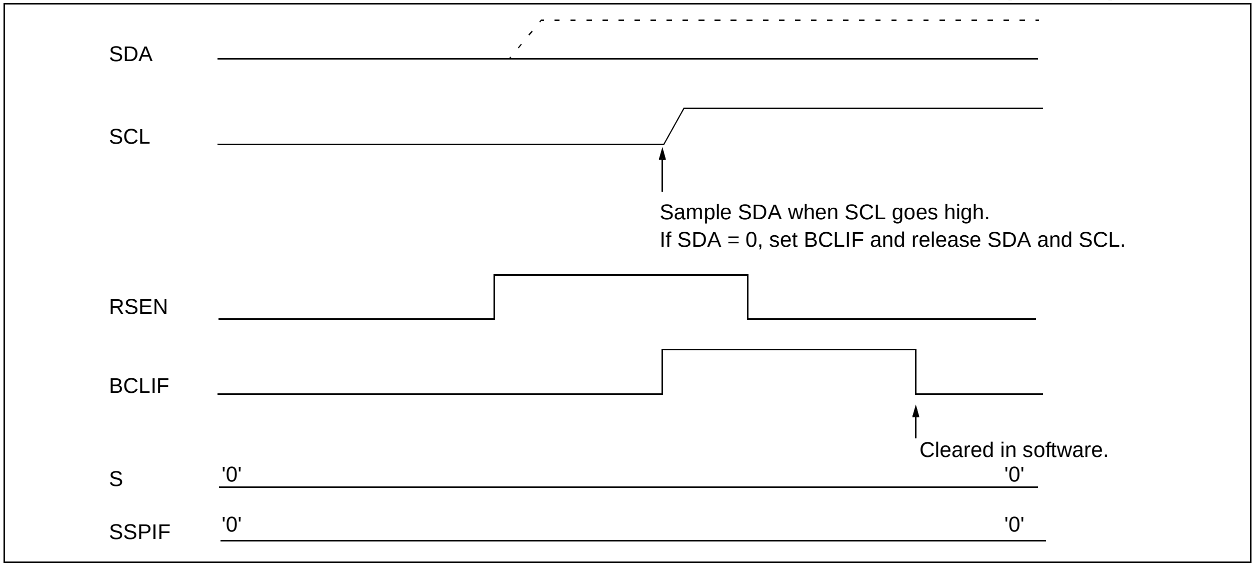
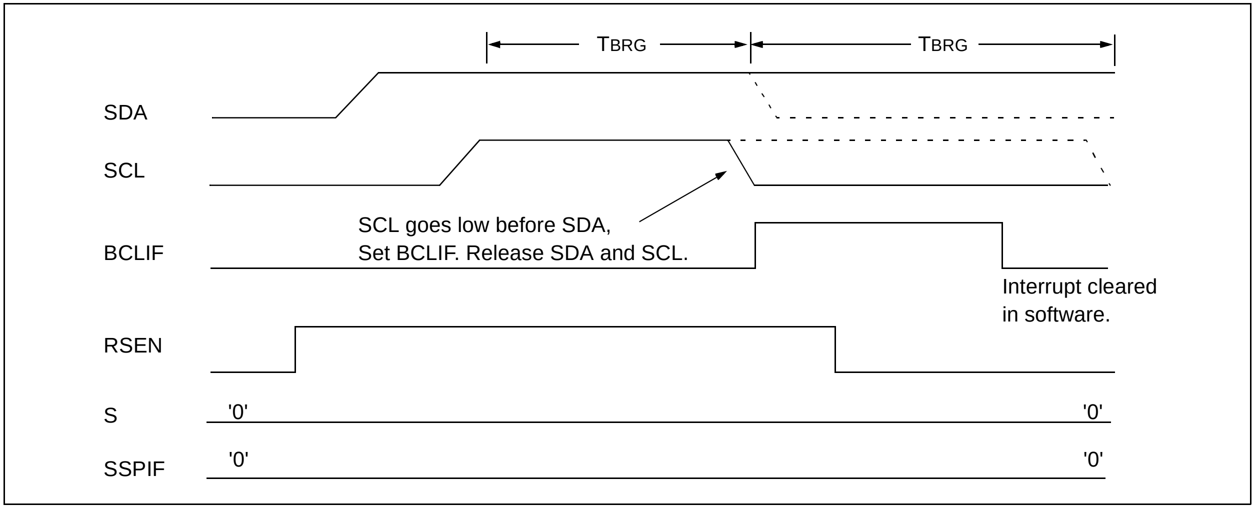

SDA
SCL
Sample SDA when SCL goes high.
If SDA = 0, set BCLIF and release SDA and SCL.
RSEN
BCLIF
Cleared in software.
S
'0'
'0'
SSPIF
'0'
'0'

FIGURE 15-39: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



17.4.2 MINIMIZING CURRENT CONSUMPTION

To minimize current consumption, all I/O pins should be either at VDD, or VSS, with no external circuitry drawing current from the I/O pin. I/O pins that are hi-impedance inputs should be pulled high or low externally to avoid switching currents caused by floating inputs. The T0CKI input should be at VDD or VSS. The contributions from on-chip pull-ups on PORTB should also be considered and disabled, when possible.

17.5 Code Protection

The code in the program memory can be protected by selecting the microcontroller in Code Protected mode (PM2:PM0 = '000').

In this mode, instructions that are in the on-chip program memory space, can continue to read or write the program memory. An instruction that is executed outside of the internal program memory range will be inhibited from writing to, or reading from, program memory.

Note:	Microchip does not recommend code protecting windowed devices.

If the code protection bit(s) have not been programmed, the on-chip program memory can be read out for verification purposes.

17.6 In-Circuit Serial Programming

The PIC17C7XX group of the high-end family (PIC17CXXX) has an added feature that allows serial programming while in the end application circuit. This is simply done with two lines for clock and data and three other lines for power, ground, and the programming voltage. This allows customers to manufacture boards with unprogrammed devices and then program the microcontroller just before shipping the product. This also allows the most recent firmware, or a custom firmware to be programmed.

Devices may be serialized to make the product unique; "special" variants of the product may be offered and code updates are possible. This allows for increased design flexibility.

To place the device into the Serial Programming Test mode, two pins will need to be placed at VIHH. These are the TEST pin and the $\overline{\text{MCLR}}$/VPP pin. Also, a sequence of events must occur as follows:

1. The TEST pin is placed at VIHH.
2. The $\overline{\text{MCLR}}$/VPP pin is placed at VIHH.

There is a setup time between step 1 and step 2 that must be met.

After this sequence, the Program Counter is pointing to program memory address 0xFF60. This location is in the Boot ROM. The code initializes the USART/SCI so that it can receive commands. For this, the device must be clocked. The device clock source in this mode is the RA1/T0CKI pin. After delaying to allow the USART/SCI to initialize, commands can be received. The flow is shown in these 3 steps:

1. The device clock source starts.
2. Wait 80 device clocks for Boot ROM code to configure the USART/SCI.
3. Commands may now be sent.

For complete details of serial programming, please refer to the PIC17C7XX Programming Specification. (Contact your local Microchip Technology Sales Office for availability.)

FIGURE 17-3: TYPICAL IN-CIRCUIT SERIAL PROGRAMMING CONNECTION

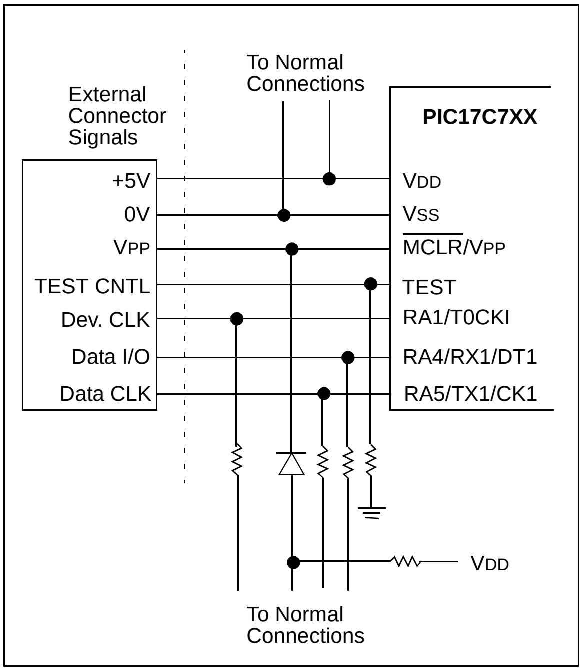


TABLE 17-3: ICSP INTERFACE PINS

Name	During Programming		
	Function	Type	Description
RA4/RX1/DT1	DT	I/O	Serial Data
RA5/TX1/CK1	CK	I	Serial Clock
RA1/T0CKI	OSCI	I	Device Clock Source
TEST	TEST	I	Test mode selection control input, force to VIHH
$\overline{\text{MCLR}}$/VPP	$\overline{\text{MCLR}}$/VPP	P	Master Clear Reset and Device Programming Voltage
VDD	VDD	P	Positive supply for logic and I/O pins
VSS	VSS	P	Ground reference for logic and I/O pins

MULLW Multiply Literal with WREG

Syntax: [*label*] MULLW k

Operands: 0 ≤ k ≤ 255

Operation: (k x WREG) → PRODH:PRODL

Status Affected: None

Encoding:

1011	1100	kkkk	kkkk

Description: An unsigned multiplication is carried out between the contents of WREG and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte.

WREG is unchanged.

None of the status flags are affected.

Note that neither overflow, nor carry is possible in this operation. A zero result is possible, but not detected.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH:PRODL

Example: `MULLW 0xC4`

Before Instruction

WREG	=	0xE2
PRODH	=	?
PRODL	=	?

After Instruction

WREG	=	0xC4
PRODH	=	0xAD
PRODL	=	0x08

MULWF Multiply WREG with f

Syntax: [*label*] MULWF f

Operands: 0 ≤ f ≤ 255

Operation: (WREG x f) → PRODH:PRODL

Status Affected: None

Encoding:

0011	0100	ffff	ffff

Description: An unsigned multiplication is carried out between the contents of WREG and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte.

Both WREG and 'f' are unchanged.

None of the status flags are affected.

Note that neither overflow, nor carry is possible in this operation. A zero result is possible, but not detected.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write registers PRODH:PRODL

Example: `MULWF REG`

Before Instruction

WREG	=	0xC4
REG	=	0xB5
PRODH	=	?
PRODL	=	?

After Instruction

WREG	=	0xC4
REG	=	0xB5
PRODH	=	0x8A
PRODL	=	0x94

RLNCF Rotate Left f (no carry)

Syntax: [*label*] RLNCF f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

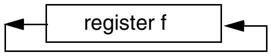
Operation: f<n> → d<n+1>;
f<7> → d<0>

Status Affected: None

Encoding:

`0010`	`001d`	`ffff`	`ffff`

Description: The contents of register 'f' are rotated one bit to the left. If 'd' is 0, the result is placed 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: `RLNCF REG, 1`

Before Instruction

C = `0`
REG = `1110 1011`

After Instruction

C =
REG = `1101 0111`

RRCF Rotate Right f through Carry

Syntax: [*label*] RRCF f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

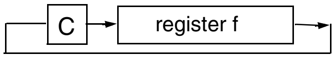
Operation: f<n> → d<n-1>;
f<0> → C;
C → d<7>

Status Affected: C

Encoding:

`0001`	`100d`	`ffff`	`ffff`

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. 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: `RRCF REG1,0`

Before Instruction

REG1 = `1110 0110`
C = `0`

After Instruction

REG1 = `1110 0110`
WREG = `0111 0011`
C = `0`

TABLWT Table Write

Example1: `TABLWT 1, 1, REG`

Before Instruction

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

After Instruction (table write completion)

REG	=	0x53
TBLATH	=	0x53
TBLATL	=	0x55
TBLPTR	=	0xA357
MEMORY(TBLPTR - 1)	=	0x5355

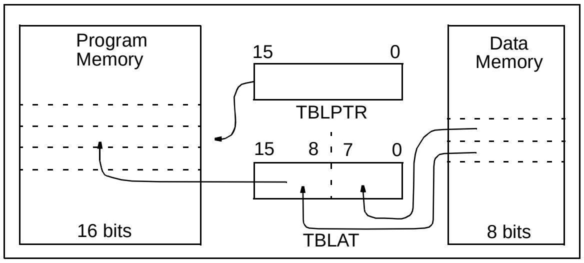
Example 2: `TABLWT 0, 0, REG`

Before Instruction

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

After Instruction (table write completion)

REG	=	0x53
TBLATH	=	0xAA
TBLATL	=	0x53
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0xAA53



TLRD Table Latch Read

Syntax: [*label*] TLRD t,f

Operands: 0 ≤ f ≤ 255
t ∈ [0,1]

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

Status Affected: None

Encoding:

1010	00tx	ffff	ffff

Description: Read data from 16-bit table latch (TBLAT) into file register 'f'. Table Latch is unaffected.

If t = 1; high byte is read

If t = 0; low byte is read

This instruction is used in conjunction with `TABLRD` to transfer data from program memory to data memory.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `TLRD t, RAM`

Before Instruction

t	=	0	
RAM	=	?	
TBLAT	=	0x00AF	(TBLATH = 0x00)
			(TBLATL = 0xAF)

After Instruction

RAM	=	0xAF	
TBLAT	=	0x00AF	(TBLATH = 0x00)
			(TBLATL = 0xAF)

Before Instruction

t	=	1	
RAM	=	?	
TBLAT	=	0x00AF	(TBLATH = 0x00)
			(TBLATL = 0xAF)

After Instruction

RAM	=	0x00	
TBLAT	=	0x00AF	(TBLATH = 0x00)
			(TBLATL = 0xAF)

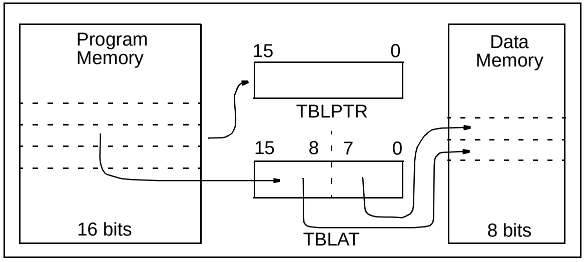


FIGURE 20-13: SPI MASTER MODE TIMING (CKE = 0)

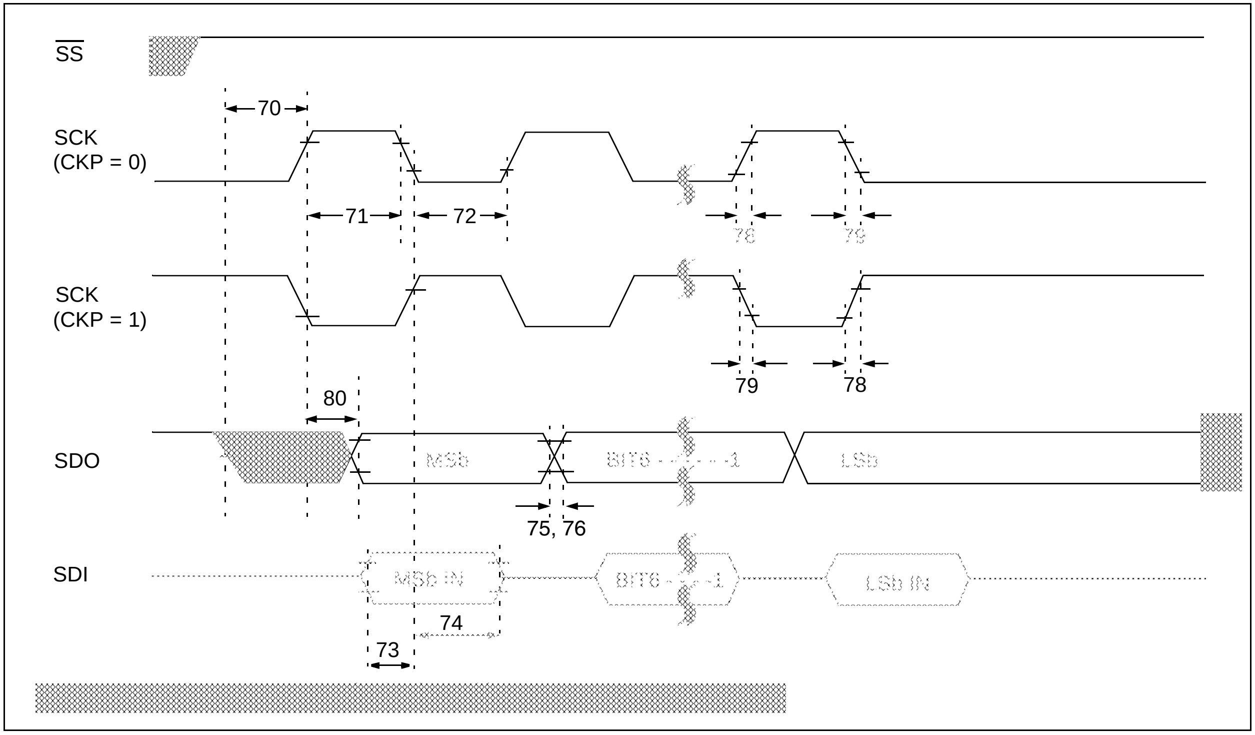


TABLE 20-8: SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)

Param. No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{\text{SS}}$↓ 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		—	10	25	ns	
76	TdoF	SDO data output fall time		—	10	25	ns	
78	TscR	SCK output rise time (Master mode)		—	10	25	ns	
79	TscF	SCK output fall time (Master mode)		—	10	25	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge		—	—	50	ns	

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

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

FIGURE 20-16: SPI SLAVE MODE TIMING (CKE = 1)

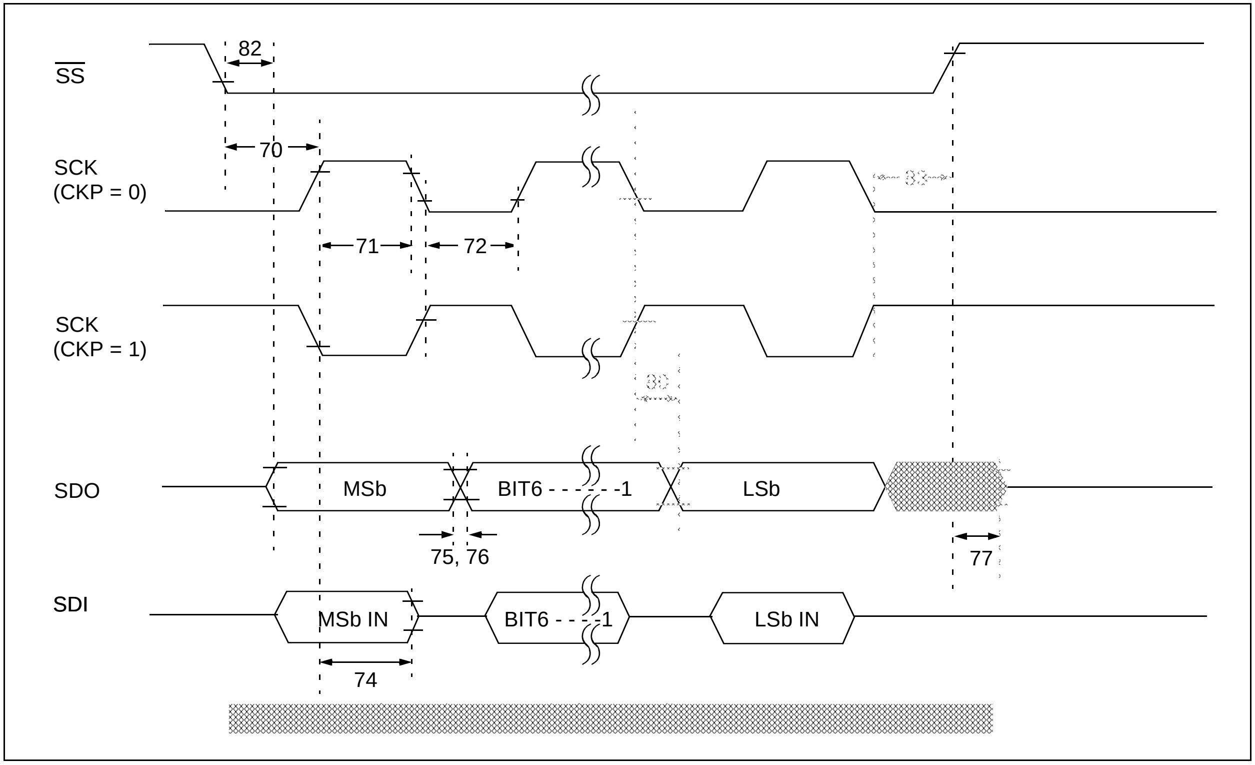


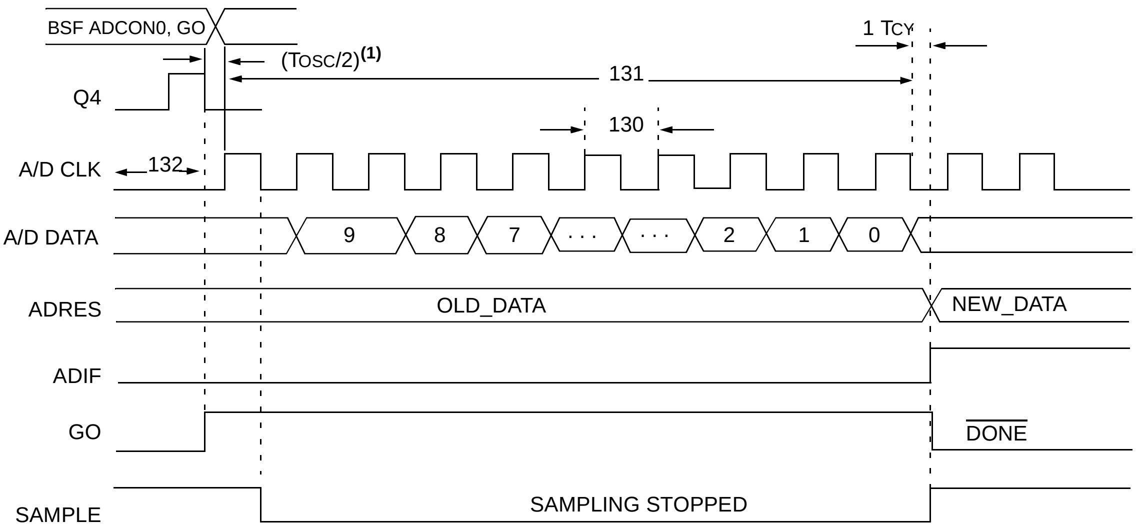
TABLE 20-11: SPI MODE REQUIREMENTS (SLAVE MODE, CKE = 1)

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)**
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		—	10	25	ns	
76	TdoF	SDO data output fall time		—	10	25	ns	
77	TssH2doZ	$\overline{SS}$↑ to SDO output hi-impedance		10	—	50	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge		—	—	50	ns	
82	TssL2doV	SDO data output valid after $\overline{SS}$↓ edge		—	—	50	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.

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

FIGURE 20-23: A/D CONVERSION TIMING



Note 1: If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the `SLEEP` instruction to be executed.

TABLE 20-19: A/D CONVERSION REQUIREMENTS

Param. No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
130	TAD	A/D clock period	PIC17**C**XXX	1.6	—	—	µs	TOSC based, VREF ≥ 3.0V
			PIC17**LC**XXX	3.0	—	—	µs	TOSC based, VREF full range
			PIC17**C**XXX	2.0	4.0	6.0	µs	A/D RC mode
			PIC17**LC**XXX	3.0	6.0	9.0	µs	A/D RC mode
131	TCNV	Conversion time (not including acquisition time) **(Note 1)**		11	—	12	Tad	
132	TACQ	Acquisition time		**(Note 2)**	20	—	µs	
				10	—	—	µs	The minimum time is the amplifier settling time. This may be used if the "new" input voltage has not changed by more than 1LSb (i.e., 5 mV @ 5.12V) from the last sampled voltage (as stated on CHOLD).
134	TGO	Q4 to ADCLK start		—	Tosc/2	—	—	If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the `SLEEP` instruction to be executed.

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

Note 1: ADRES register may be read on the following TCY cycle.

2: See Section 16.1 for minimum conditions when input voltage has changed more than 1 LSb.

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

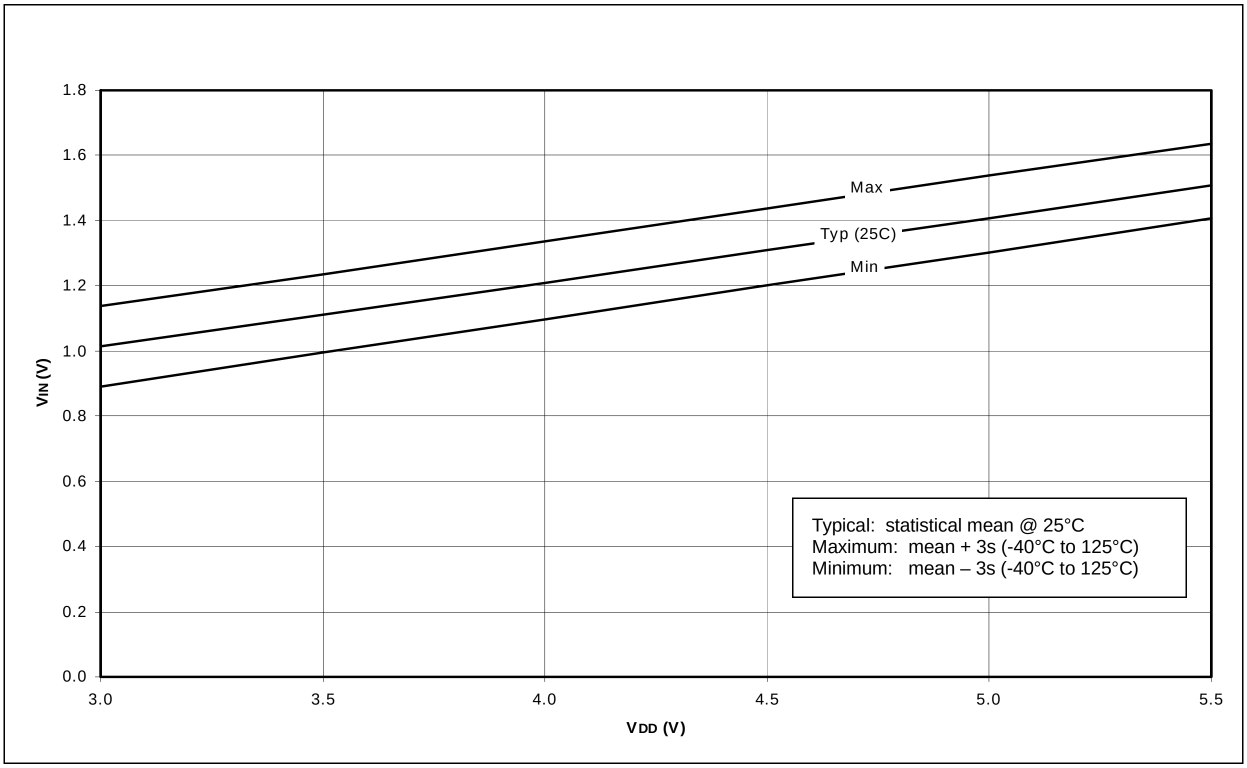
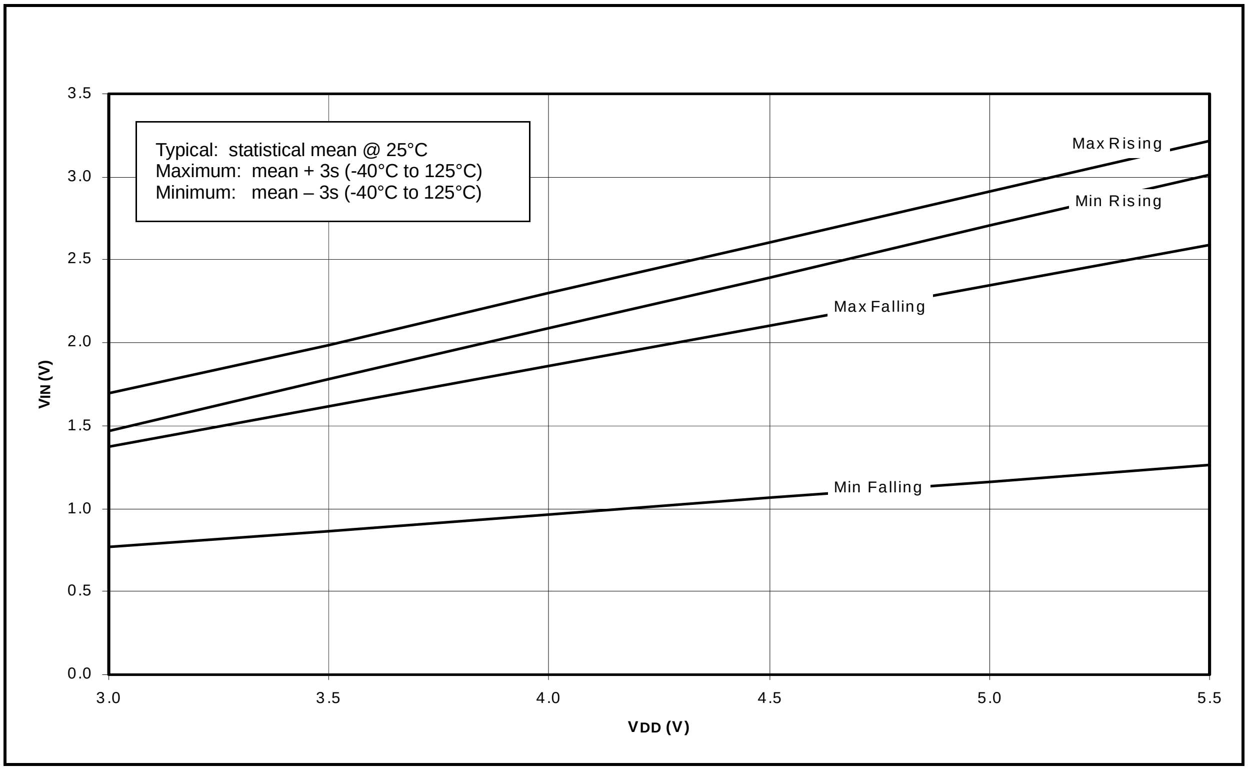


FIGURE 21-22: MAXIMUM AND MINIMUM VIN vs. VDD (ST Input, -40° C to +125°C)


VIN (V)
VDD (V)
Typical: statistical mean @ 25°C
Maximum: mean + 3s (-40°C to 125°C)
Minimum: mean – 3s (-40°C to 125°C)
Max Rising
Min Rising
Max Falling
Min Falling