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	50
Program Memory Size	16KB (8K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	454 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	68-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.23x24.23)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c752t-33-l

FIGURE 3-1: PIC17C752/756A BLOCK DIAGRAM

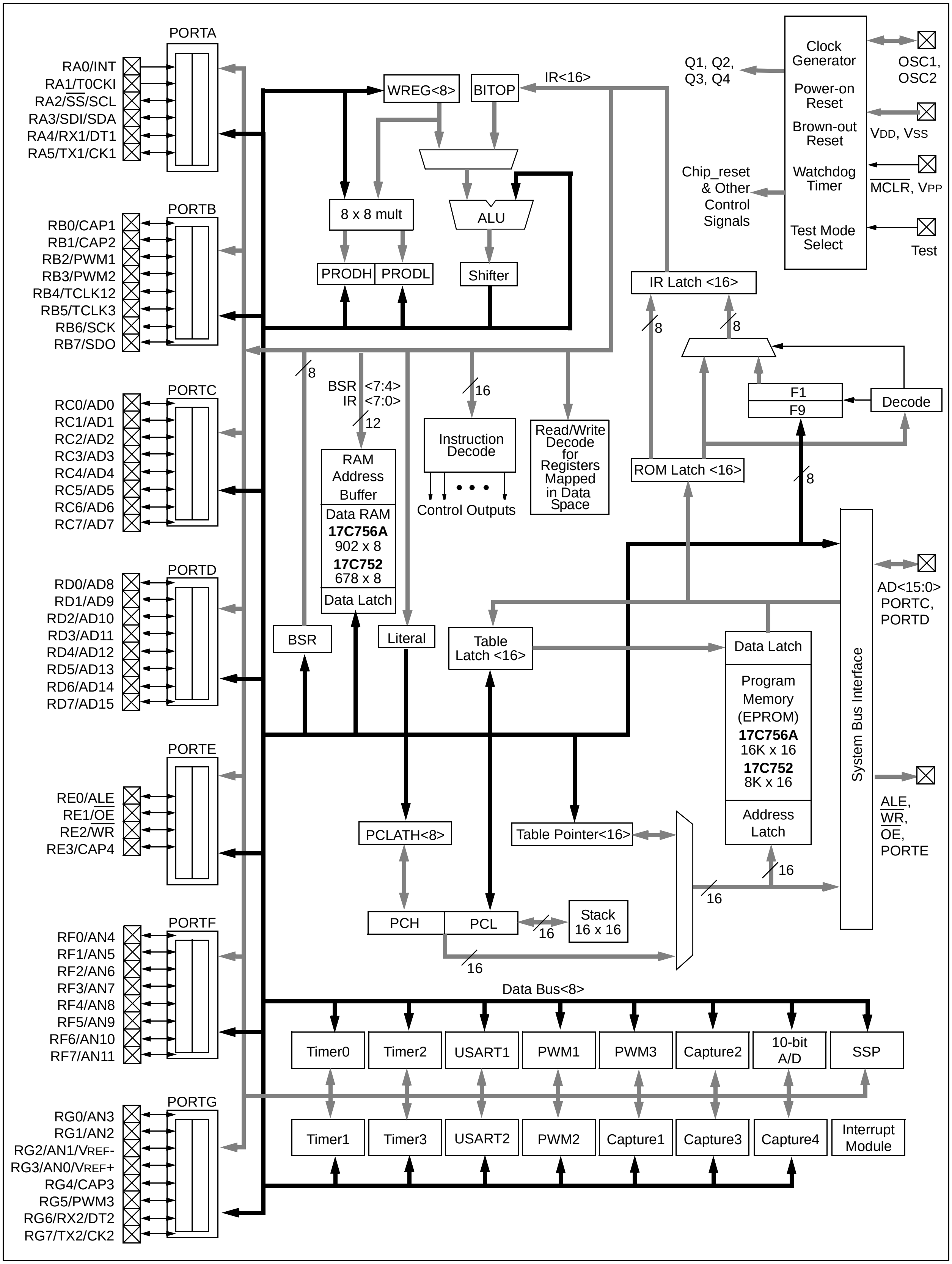


TABLE 7-3: SPECIAL FUNCTION REGISTERS

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	$\overline{MCLR}$, 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	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	--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/$\overline{SS}$/ 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/$\overline{WR}$	RE1/$\overline{OE}$	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 $\overline{TO}$ and $\overline{PD}$ status bits in CPUSTA are not affected by a $\overline{MCLR}$ 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.

7.8 Bank Select Register (BSR)

The BSR is used to switch between banks in the data memory area (Figure 7-9). In the PIC17C7XX devices, the entire byte is implemented. The lower nibble is used to select the peripheral register bank. The upper nibble is used to select the general purpose memory bank.

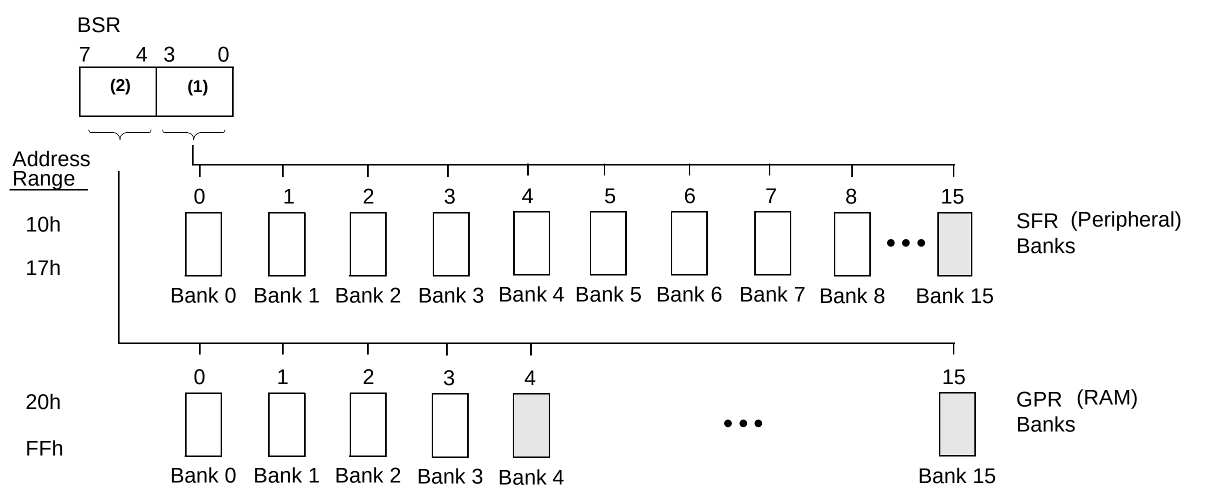
All the Special Function Registers (SFRs) are mapped into the data memory space. In order to accommodate the large number of registers, a banking scheme has been used. A segment of the SFRs, from address 10h to address 17h, is banked. The lower nibble of the bank select register (BSR) selects the currently active "peripheral bank." Effort has been made to group the peripheral registers of related functionality in one bank. However, it will still be necessary to switch from bank to bank in order to address all peripherals related to a single task. To assist this, a `MOVLB bank` instruction has been included in the instruction set.

The need for a large general purpose memory space dictated a general purpose RAM banking scheme. The upper nibble of the BSR selects the currently active general purpose RAM bank. To assist this, a `MOVLR bank` instruction has been provided in the instruction set.

If the currently selected bank is not implemented (such as Bank 13), any read will read all '0's. Any write is completed to the bit bucket and the ALU status bits will be set/cleared as appropriate.

Note:	Registers in Bank 15 in the Special Function Register area, are reserved for Microchip use. Reading of registers in this bank may cause random values to be read.

FIGURE 7-9: BSR OPERATION



Note 1: For the SFRs only Banks 0 through 8 are implemented. Selection of an unimplemented bank is not recommended. **Bank 15 is reserved for Microchip use, reading of registers in this bank may cause random values to be read.**

2: For the GPRs, Bank 3 is unimplemented on the PIC17C752 and the PIC17C762. Selection of an unimplemented bank is not recommended.

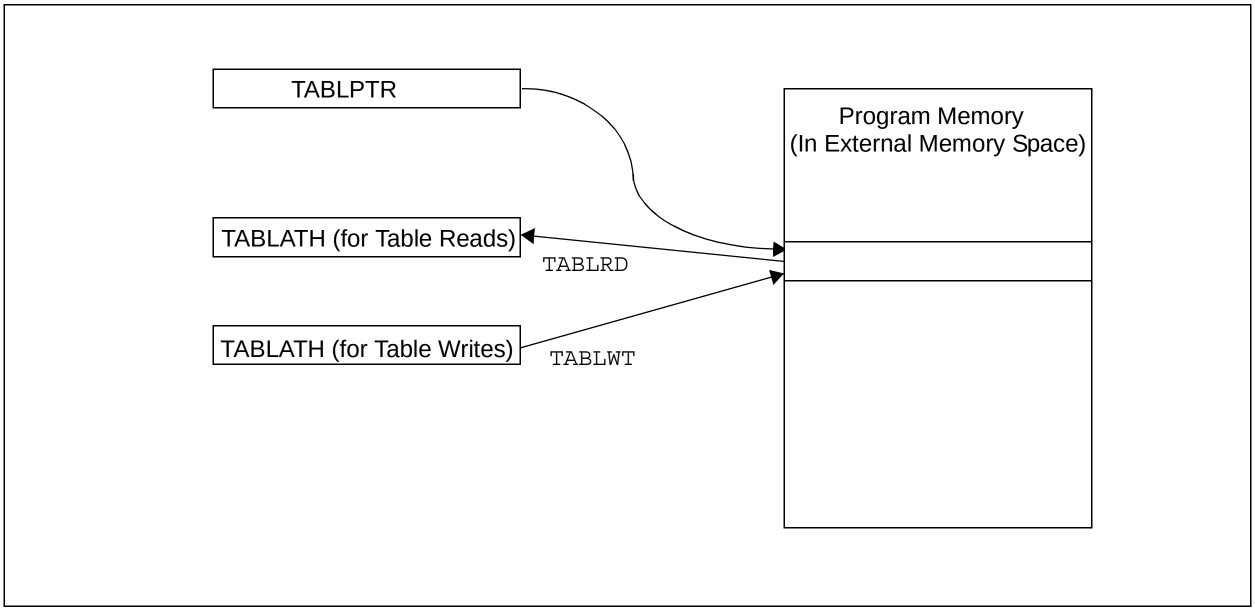
3: SFR Bank 8 is only implemented on the PIC17C76X.

8.4 Operation with External Memory Interface

When the table reads/writes are accessing external memory (via the external system interface bus), the table latch for the table reads is different from the table latch for the table writes (see Figure 8-9).

This means that you cannot do a TABLRD instruction, and use the values that were loaded into the table latches for a TABLWT instruction. Any table write sequence should use both the TLWT and then the TABLWT instructions.

FIGURE 8-9: ACCESSING EXTERNAL MEMORY WITH TABLRD AND TABLWT INSTRUCTIONS



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.

10.5 PORTE and DDRE Register

PORTE is a 4-bit bi-directional port. The corresponding data direction register is DDRE. A '1' in DDRE configures the corresponding port pin as an input. A '0' in the DDRE register configures the corresponding port pin as an output. Reading PORTE reads the status of the pins, whereas writing to PORTE will write to the port latch. PORTE is multiplexed with the system bus. When operating as the system bus, PORTE contains the control signals for the address/data bus (AD15:AD0). These control signals are Address Latch Enable (ALE), Output Enable ($\overline{OE}$) and Write ($\overline{WR}$). The control signals $\overline{OE}$ and $\overline{WR}$ are active low signals. The timing for the system bus is shown in the Electrical Specifications section.

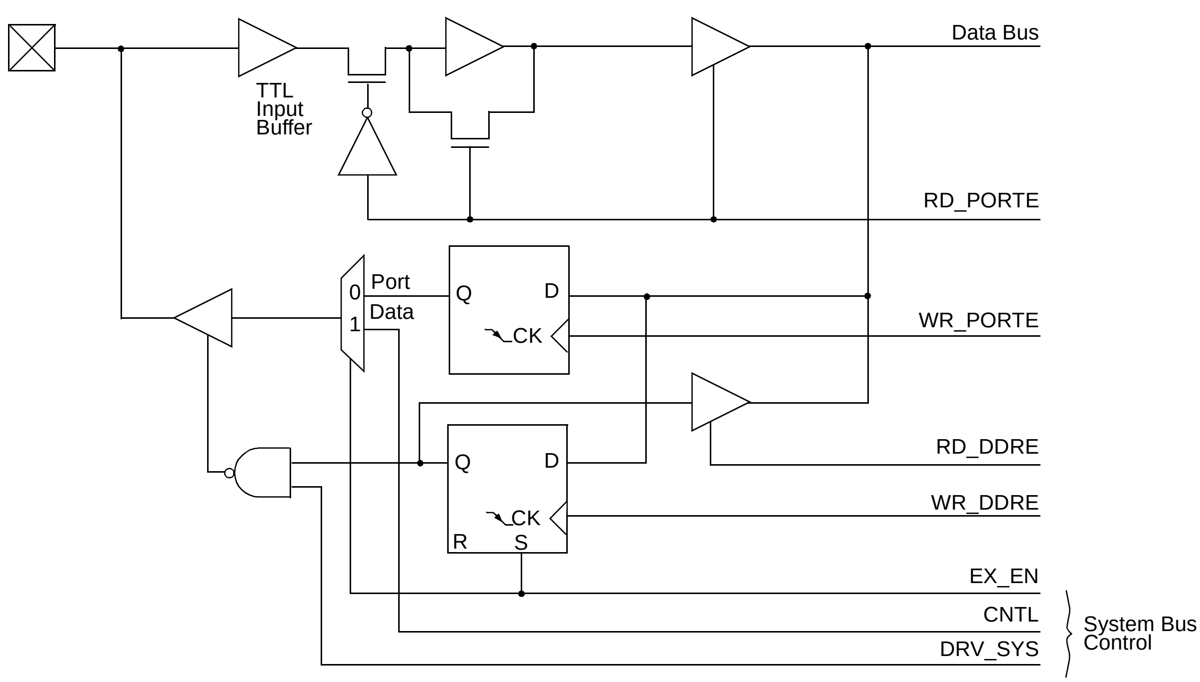
Note:	Three pins of this port are configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. The other pin is a general purpose I/O or Capture4 pin. In the two other microcontroller modes, RE2:RE0 are general purpose I/O pins.

Example 10-5 shows an instruction sequence to initialize PORTE. 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 10-5: INITIALIZING PORTE

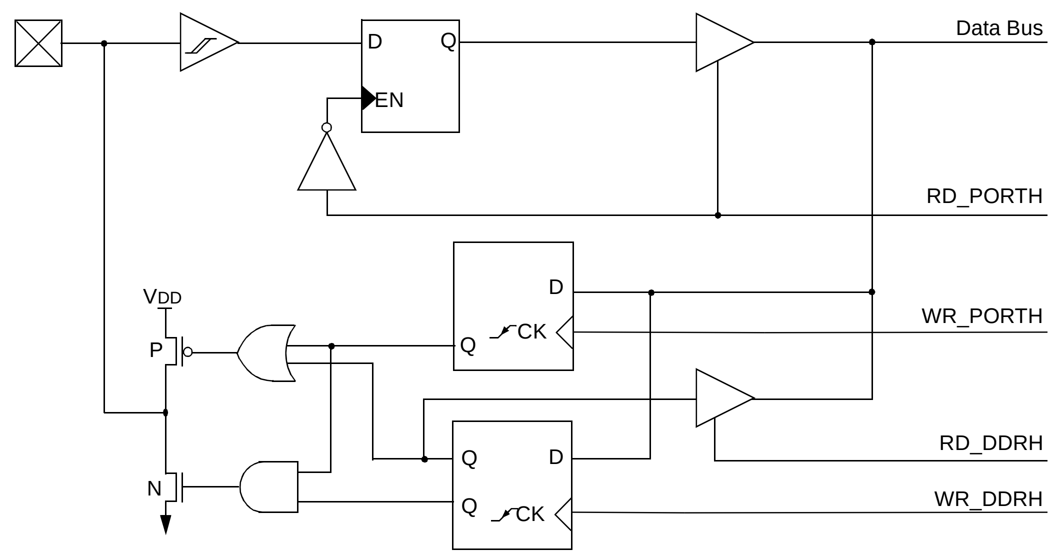
```
MOVLB  1          ; Select Bank 1
CLRF   PORTE, F   ; Initialize PORTE data
                  ; latches before setting
                  ; the data direction
                  ; register
MOVLW   0x03      ; Value used to initialize
                  ; data direction
MOVWF   DDRE      ; Set RE<1:0> as inputs
                  ; RE<3:2> as outputs
                  ; RE<7:4> are always
                  ; read as '0'
```

FIGURE 10-11: BLOCK DIAGRAM OF RE2:RE0 (IN I/O PORT MODE)



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

FIGURE 10-18: RH3:RH0 BLOCK DIAGRAM



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

TABLE 10-15: PORTH FUNCTIONS

Name	Bit	Buffer Type	Function
RH0	bit0	ST	Input/output.
RH1	bit1	ST	Input/output.
RH2	bit2	ST	Input/output.
RH3	bit3	ST	Input/output.
RH4/AN12	bit4	ST	Input/output or analog input 12.
RH5/AN13	bit5	ST	Input/output or analog input 13.
RH6/AN14	bit6	ST	Input/output or analog input 14.
RH7/AN15	bit7	ST	Input/output or analog input 15.

Legend: ST = Schmitt Trigger input

TABLE 10-16: REGISTERS/BITS ASSOCIATED WITH PORTH

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
10h, Bank 8	DDRH	Data Direction Register for PORTH								1111 1111	1111 1111
11h, Bank 8	PORTH	RH7/ AN15	RH6/ AN14	RH5/ AN13	RH4/ AN12	RH3	RH2	RH1	RH0	0000 xxxx	0000 uuuu
15h, Bank 5	ADCON1	ADCS1	ADCS0	ADFM	—	PCFG3	PCFG2	PCFG1	PCFG0	000- 0000	000- 0000

Legend: x = unknown, u = unchanged

NOTES:

14.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. Table 14-2 shows the formula for computation of the baud rate for different USART modes. These only apply when the USART is in Synchronous Master mode (internal clock) and Asynchronous mode.

Given the desired baud rate and Fosc, the nearest integer value between 0 and 255 can be calculated using the formula below. The error in baud rate can then be determined.

TABLE 14-2: BAUD RATE FORMULA

SYNC	Mode	Baud Rate
0	Asynchronous	FOSC/(64(X+1))
1	Synchronous	FOSC/(4(X+1))

X = value in SPBRG (0 to 255)

Example 14-1 shows the calculation of the baud rate error for the following conditions:

FOSC = 16 MHz
Desired Baud Rate = 9600
SYNC = 0

EXAMPLE 14-1: CALCULATING BAUD RATE ERROR

Desired Baud Rate = FOSC / (64 (X + 1))
9600 = 16000000 /(64 (X + 1))
X = 25.042 → 25
Calculated Baud Rate = 16000000 / (64 (25 + 1))
= 9615
Error = $\frac{\text{(Calculated Baud Rate - Desired Baud Rate)}}{\text{Desired Baud Rate}}$
= (9615 - 9600) / 9600
= 0.16%

Writing a new value to the SPBRG, causes the BRG timer to be reset (or cleared). This ensures that the BRG does not wait for a timer overflow before outputting the new baud rate.

Effects of Reset

After any device RESET, the SPBRG register is cleared. The SPBRG register will need to be loaded with the desired value after each RESET.

TABLE 14-3: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

	Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on $\overline{\text{MCLR}}$, WDT
USART1	13h, Bank 0	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
	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
USART2	13h, Bank 4	RCSTA2	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
	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 '0'. Shaded cells are not used by the Baud Rate Generator.

15.2.3 SLEEP OPERATION

While in SLEEP mode, the I^{2}C module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from SLEEP (if the SSP interrupt is enabled).

15.2.4 EFFECTS OF A RESET

A RESET disables the SSP module and terminates the current transfer.

TABLE 15-3: REGISTERS ASSOCIATED WITH I^{2}C OPERATION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR	$\overline{\text{MCLR}}$, WDT
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
10h, Bank 4	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	000- 0000	000- 0000
11h, Bank 4	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	000- 0000	000- 0000
10h. Bank 6	SSPADD	Synchronous Serial Port (I^{2}C mode) Address Register								0000 0000	0000 0000
14h, Bank 6	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
11h, Bank 6	SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
12h, Bank 6	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	0000 0000
13h, Bank 6	SSPSTAT	SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF	0000 0000	0000 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the SSP in I^{2}C mode.

15.2.14 STOP CONDITION TIMING

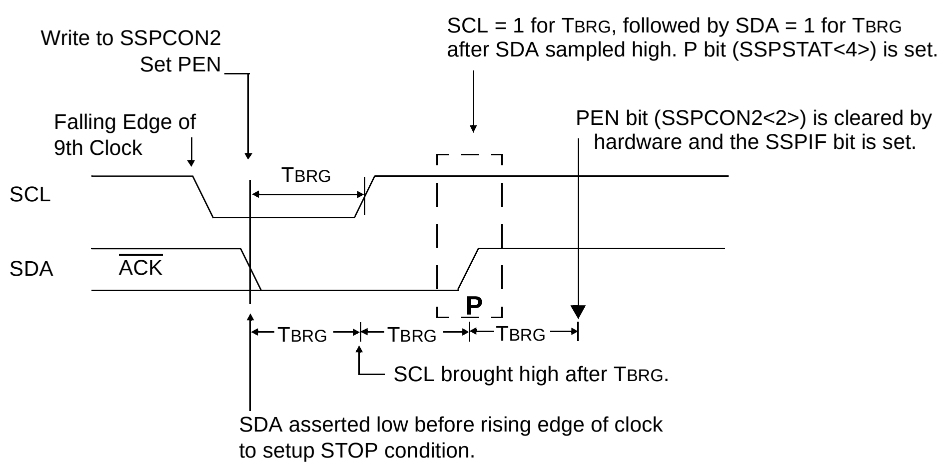
A STOP bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit PEN (SSPCON2<2>). At the end of a receive/transmit the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When 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



Note: TBRG = one baud rate generator period.

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)

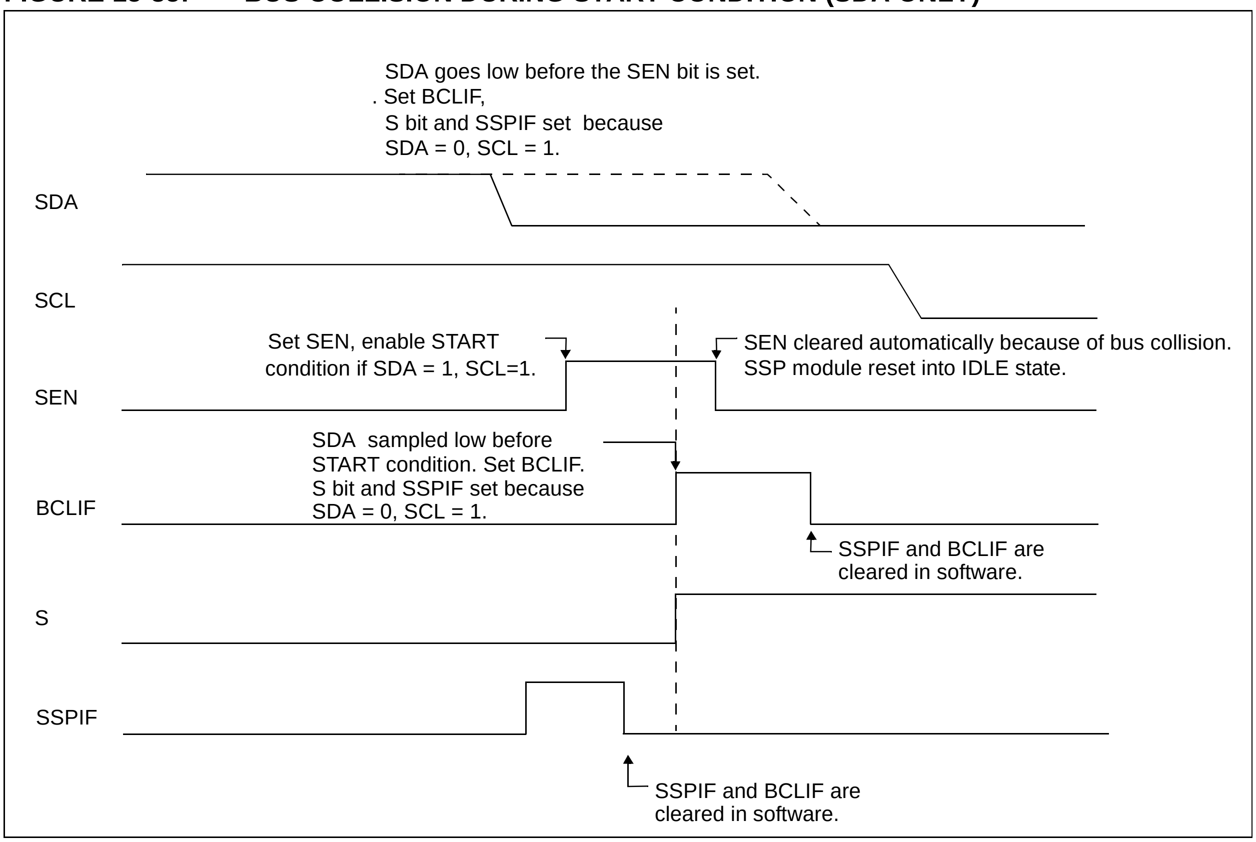
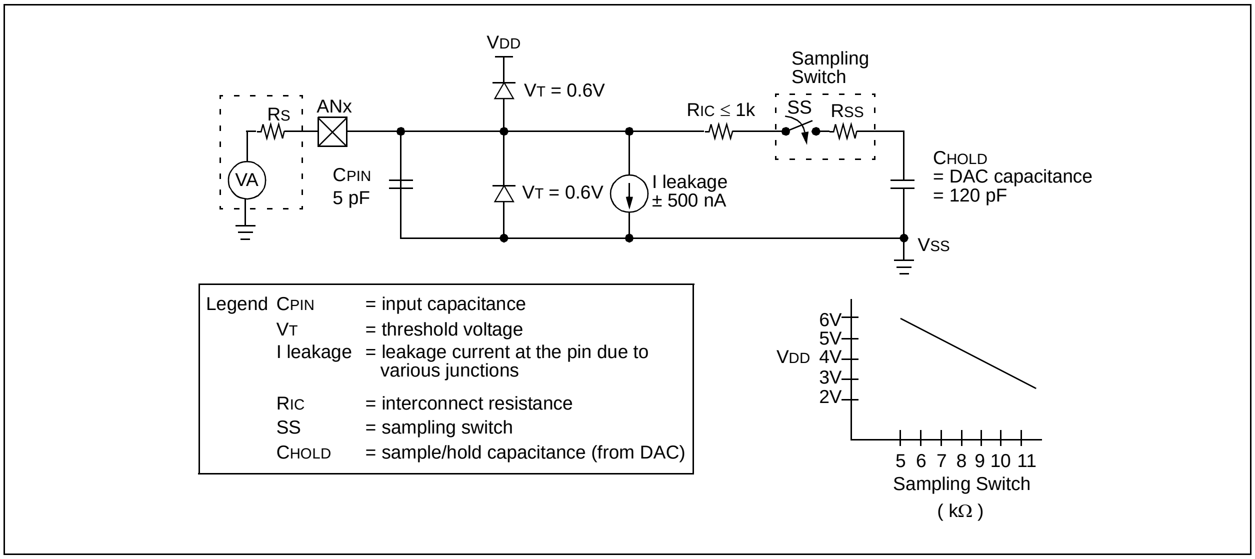


FIGURE 16-3: ANALOG INPUT MODEL



16.10 References

A good reference for understanding A/D converter is the "Analog-Digital Conversion Handbook" third edition, published by Prentice Hall (ISBN 0-13-03-2848-0).

TABLE 16-3: REGISTERS/BITS ASSOCIATED WITH A/D

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR	$\overline{\text{MCLR}}$, WDT
06h, unbanked	CPUSTA	—	—	STAKAV	GLINTD	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	--11 1100	--11 qq11
07h, unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	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
10h, Bank 5	DDRF	Data Direction Register for PORTF								1111 1111	1111 1111
11h, Bank 5	PORTF	RF7/ AN11	RF6/ AN10	RF5/ AN9	RF4/ AN8	RF3/ AN7	RF2/ AN6	RF1/ AN5	RF0/ AN4	0000 0000	0000 0000
12h, Bank 5	DDRG	Data Direction register for PORTG								1111 1111	1111 1111
13h, Bank 5	PORTG	RG7/ TX2/CK2	RG6/ RX2/DT2	RG5/ PWM3	RG4/ CAP3	RG3/ AN0/VREF+	RG2/ AN1/VREF-	RG1/ AN2	RG0/ AN3	xxxx 0000	uuuu 0000
14h, Bank 5	ADCON0	CHS3	CHS2	CHS1	CHS0	—	GO/$\overline{\text{DONE}}$	—	ADON	0000 -0-0	0000 -0-0
15h, Bank 5	ADCON1	ADCS1	ADCS0	ADFM	—	PCFG3	PCFG2	PCFG1	PCFG0	000- 0000	000- 0000
16h, Bank 5	ADRESL	A/D Result Low Register								xxxx xxxx	uuuu uuuu
17h, Bank 5	ADRESH	A/D Result High Register								xxxx xxxx	uuuu uuuu

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

Note: Other (non power-up) RESETS include: external RESET through $\overline{\text{MCLR}}$ and Watchdog Timer Reset.

IORWF — Inclusive OR WREG with f

Syntax:	[*label*] IORWF f,d
Operands:	$0 \le f \le 255$ $d \in [0,1]$
Operation:	(WREG) .OR. (f) $\rightarrow$ (dest)
Status Affected:	$\overline{Z}$
Encoding:	`0000` `100d` `ffff` `ffff`
Description:	Inclusive OR WREG with register 'f'. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f'.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `IORWF RESULT, 0`

Before Instruction
RESULT = 0x13
WREG = 0x91

After Instruction
RESULT = 0x13
WREG = 0x93

LCALL — Long Call

Syntax:	[*label*] LCALL k
Operands:	$0 \le k \le 255$
Operation:	PC + 1 $\rightarrow$ TOS; k $\rightarrow$ PCL, (PCLATH) $\rightarrow$ PCH
Status Affected:	None
Encoding:	`1011` `0111` `kkkk` `kkkk`
Description:	`LCALL` allows an unconditional subroutine call to anywhere within the 64K program memory space. First, the return address (PC + 1) is pushed onto the stack. A 16-bit destination address is then loaded into the program counter. The lower 8-bits of the destination address are embedded in the instruction. The upper 8-bits of PC are loaded from PC high holding latch, PCLATH.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write register PCL
No operation	No operation	No operation	No operation

Example:

```
MOVLW  HIGH(SUBROUTINE)
MOVPF  WREG, PCLATH
LCALL  LOW(SUBROUTINE)
```

Before Instruction
`SUBROUTINE` = 16-bit Address
PC = ?

After Instruction
PC = Address `(SUBROUTINE)`

TABLRD Table Read

Example1: `TABLRD 1, 1, REG ;`

Before Instruction

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

After Instruction (table write completion)

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

Example2: `TABLRD 0, 0, REG ;`

Before Instruction

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

After Instruction (table write completion)

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

TABLWT Table Write

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

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

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

Status Affected: None

Encoding:

1010	11ti	ffff	ffff

Description:

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

> **Note:** The $\overline{MCLR}$/VPP pin must be at the programming voltage for successful programming of internal memory.
> If $\overline{MCLR}$/VPP = VDD
> the programming sequence of internal memory will be interrupted. A short write will occur (2 TCY). The internal memory location will not be affected.

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

Words: 1

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

Q Cycle Activity:

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

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

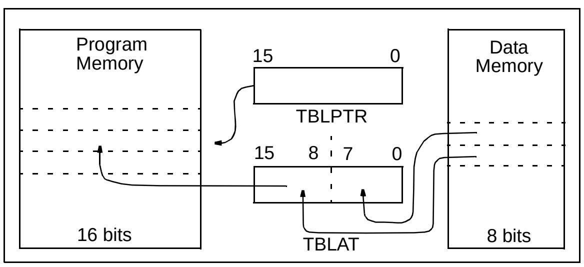
Example 2: `TABLWT 0, 0, REG`

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

Operation: If t = 0,
TBLATL $\rightarrow$ f;
If t = 1,
TBLATH $\rightarrow$ 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

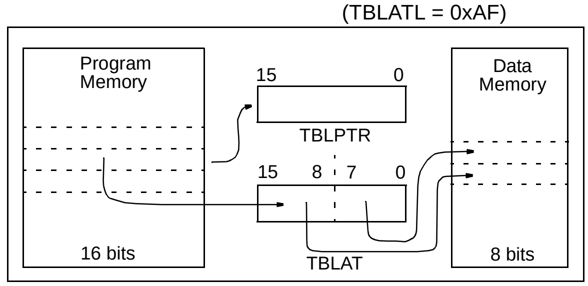
RAM	=	0xAF	
TBLAT	=	0x00AF	(TBLATH = 0x00)
			(TBLATL = 0xAF)

Before Instruction

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

After Instruction

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



19.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C17 and MPLAB C18 C Compilers
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - ICEPIC™ In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD for PIC16F87X
- Device Programmers
  - PRO MATE® II Universal Device Programmer
  - PICSTART® Plus Entry-Level Development Programmer
- Low Cost Demonstration Boards
  - PICDEM™ 1 Demonstration Board
  - PICDEM 2 Demonstration Board
  - PICDEM 3 Demonstration Board
  - PICDEM 17 Demonstration Board
  - KEELOQ® Demonstration Board

19.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8-bit microcontroller market. The MPLAB IDE is a Windows®-based application that contains:

- An interface to debugging tools
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
  - in-circuit debugger (sold separately)
- A full-featured editor
- A project manager
- Customizable toolbar and key mapping
- A status bar
- On-line help

The MPLAB IDE allows you to:

- Edit your source files (either assembly or ‘C’)
- One touch assemble (or compile) and download to PIC MCU emulator and simulator tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
  - machine code

The ability to use MPLAB IDE with multiple debugging tools allows users to easily switch from the cost-effective simulator to a full-featured emulator with minimal retraining.

19.2 MPASM Assembler

The MPASM assembler is a full-featured universal macro assembler for all PIC MCU’s.

The MPASM assembler has a command line interface and a Windows shell. It can be used as a stand-alone application on a Windows 3.x or greater system, or it can be used through MPLAB IDE. The MPASM assembler generates relocatable object files for the MPLINK object linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, an absolute LST file that contains source lines and generated machine code, and a COD file for debugging.

The MPASM assembler features include:

- Integration into MPLAB IDE projects.
- User-defined macros to streamline assembly code.
- Conditional assembly for multi-purpose source files.
- Directives that allow complete control over the assembly process.

19.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI ‘C’ compilers for Microchip’s PIC17CXXX and PIC18CXXX family of microcontrollers, respectively. These compilers provide powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compilers provide symbol information that is compatible with the MPLAB IDE memory display.

20.4 Timing Diagrams and Specifications

FIGURE 20-6: EXTERNAL CLOCK TIMING

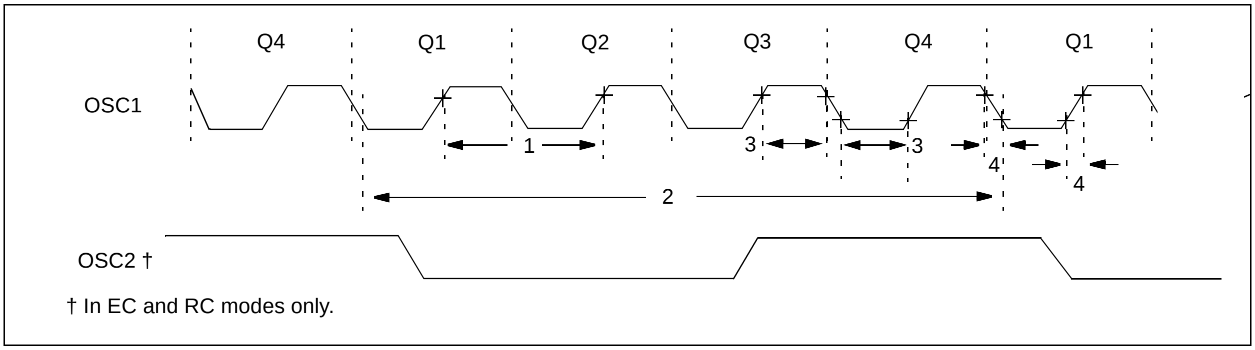


TABLE 20-1: EXTERNAL CLOCK TIMING REQUIREMENTS

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	FOSC	**External CLKIN Frequency (Note 1)**	DC DC DC	— — —	8 16 33	MHz MHz MHz	EC osc mode - 08 devices (8 MHz devices) - 16 devices (16 MHz devices) - 33 devices (33 MHz devices)
		Oscillator Frequency (Note 1)	DC 2 2 2 DC	— — — — —	4 8 16 33 2	MHz MHz MHz MHz MHz	RC osc mode XT osc mode - 08 devices (8 MHz devices) - 16 devices (16 MHz devices) - 33 devices (33 MHz devices) LF osc mode
1	TOSC	**External CLKIN Period (Note 1)**	125 62.5 30.3	— — —	— — —	ns ns ns	EC osc mode - 08 devices (8 MHz devices) - 16 devices (16 MHz devices) - 33 devices (33 MHz devices)
		Oscillator Period (Note 1)	250 125 62.5 30.3 500	— — — — —	— 1,000 1,000 1,000 —	ns ns ns ns ns	RC osc mode XT osc mode - 08 devices (8 MHz devices) - 16 devices (16 MHz devices) - 33 devices (33 MHz devices) LF osc mode
2	TCY	**Instruction Cycle Time (Note 1)**	121.2	4/FOSC	DC	ns	
3	TosL, TosH	**Clock in (OSC1) High or Low Time**	10	—	—	ns	EC oscillator
4	TosR, TosF	**Clock in (OSC1) Rise or Fall Time**	—	—	5	ns	EC oscillator

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

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min." values with an external clock applied to the OSC1/CLKIN pin. When an external clock input is used, the "max." cycle time limit is "DC" (no clock) for all devices.

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

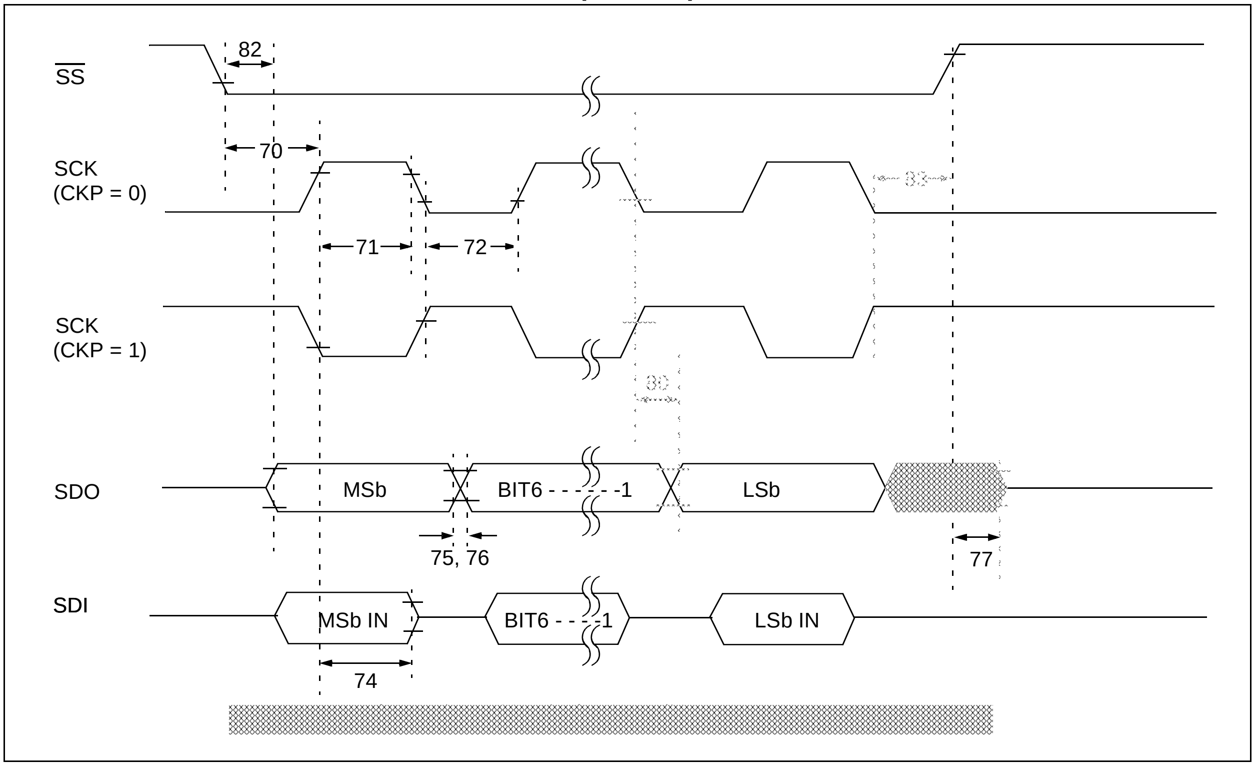


TABLE 20-11: SPI MODE REQUIREMENTS (SLAVE MODE, CKE = 1)

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.

