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	8MHz
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)	3V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (12x12)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17lc762-08i-pt

TABLE 3-1: PINOUT DESCRIPTIONS (CONTINUED)

Name	PIC17C75X			PIC17C76X		I/O/P Type	Buffer Type	Description
	DIP No.	PLCC No.	TQFP No.	PLCC No.	QFP No.			
								PORTG is a bi-directional I/O Port.
RG0/AN3	32	34	24	42	30	I/O	ST	RG0 can also be analog input 3.
RG1/AN2	31	33	23	41	29	I/O	ST	RG1 can also be analog input 2.
RG2/AN1/VREF-	30	32	22	40	28	I/O	ST	RG2 can also be analog input 1, or the ground reference voltage.
RG3/AN0/VREF+	29	31	21	39	27	I/O	ST	RG3 can also be analog input 0, or the positive reference voltage.
RG4/CAP3	35	38	27	46	33	I/O	ST	RG4 can also be the Capture3 input pin.
RG5/PWM3	36	39	28	47	34	I/O	ST	RG5 can also be the PWM3 output pin.
RG6/RX2/DT2	38	41	30	49	36	I/O	ST	RG6 can also be selected as the USART2 (SCI) Asynchronous Receive or USART2 (SCI) Synchronous Data.
RG7/TX2/CK2	37	40	29	48	35	I/O	ST	RG7 can also be selected as the USART2 (SCI) Asynchronous Transmit or USART2 (SCI) Synchronous Clock.
								PORTH is a bi-directional I/O Port. PORTH is only available on the PIC17C76X devices.
RH0	—	—	—	10	79	I/O	ST	
RH1	—	—	—	11	80	I/O	ST	
RH2	—	—	—	12	1	I/O	ST	
RH3	—	—	—	13	2	I/O	ST	
RH4/AN12	—	—	—	31	19	I/O	ST	RH4 can also be analog input 12.
RH5/AN13	—	—	—	32	20	I/O	ST	RH5 can also be analog input 13.
RH6/AN14	—	—	—	33	21	I/O	ST	RH6 can also be analog input 14.
RH7/AN15	—	—	—	34	22	I/O	ST	RH7 can also be analog input 15.
								PORTJ is a bi-directional I/O Port. PORTJ is only available on the PIC17C76X devices.
RJ0	—	—	—	52	39	I/O	ST	
RJ1	—	—	—	53	40	I/O	ST	
RJ2	—	—	—	54	41	I/O	ST	
RJ3	—	—	—	55	42	I/O	ST	
RJ4	—	—	—	73	59	I/O	ST	
RJ5	—	—	—	74	60	I/O	ST	
RJ6	—	—	—	75	61	I/O	ST	
RJ7	—	—	—	76	62	I/O	ST	
TEST	16	17	8	21	10	I	ST	Test mode selection control input. Always tie to VSS for normal operation.
VSS	17, 33, 49, 64	19, 36, 53, 68	9, 25, 41, 56	23, 44, 65, 84	11, 31, 51, 70	P		Ground reference for logic and I/O pins.
VDD	1, 18, 34, 46	2, 20, 37, 49,	10, 26, 38, 57	24, 45, 61, 2	12, 32, 48, 71	P		Positive supply for logic and I/O pins.
AVSS	28	30	20	38	26	P		Ground reference for A/D converter. This pin **MUST** be at the same potential as VSS.
AVDD	27	29	19	37	25	P		Positive supply for A/D converter. This pin **MUST** be at the same potential as VDD.
NC	—	1, 18, 35, 52	—	1, 22, 43, 64	—			No Connect. Leave these pins unconnected.

Legend: I = Input only; O = Output only; I/O = Input/Output;
P = Power; — = Not Used; TTL = TTL input; ST = Schmitt Trigger input

Note 1: The output is only available by the peripheral operation.

2: Open drain input/output pin. Pin forced to input upon any device RESET.

6.2 Peripheral Interrupt Enable Register1 (PIE1) and Register2 (PIE2)

These registers contains the individual enable bits for the peripheral interrupts.

REGISTER 6-2: PIE1 REGISTER (ADDRESS: 17h, BANK 1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE
bit 7							bit 0

bit 7 **RBIE**: PORTB Interrupt-on-Change Enable bit
1 = Enable PORTB interrupt-on-change
0 = Disable PORTB interrupt-on-change

bit 6 **TMR3IE**: TMR3 Interrupt Enable bit
1 = Enable TMR3 interrupt
0 = Disable TMR3 interrupt

bit 5 **TMR2IE**: TMR2 Interrupt Enable bit
1 = Enable TMR2 interrupt
0 = Disable TMR2 interrupt

bit 4 **TMR1IE**: TMR1 Interrupt Enable bit
1 = Enable TMR1 interrupt
0 = Disable TMR1 interrupt

bit 3 **CA2IE**: Capture2 Interrupt Enable bit
1 = Enable Capture2 interrupt
0 = Disable Capture2 interrupt

bit 2 **CA1IE**: Capture1 Interrupt Enable bit
1 = Enable Capture1 interrupt
0 = Disable Capture1 interrupt

bit 1 **TX1IE**: USART1 Transmit Interrupt Enable bit
1 = Enable USART1 Transmit buffer empty interrupt
0 = Disable USART1 Transmit buffer empty interrupt

bit 0 **RC1IE**: USART1 Receive Interrupt Enable bit
1 = Enable USART1 Receive buffer full interrupt
0 = Disable USART1 Receive buffer full interrupt

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 6-2: INT PIN/T0CKI PIN INTERRUPT TIMING

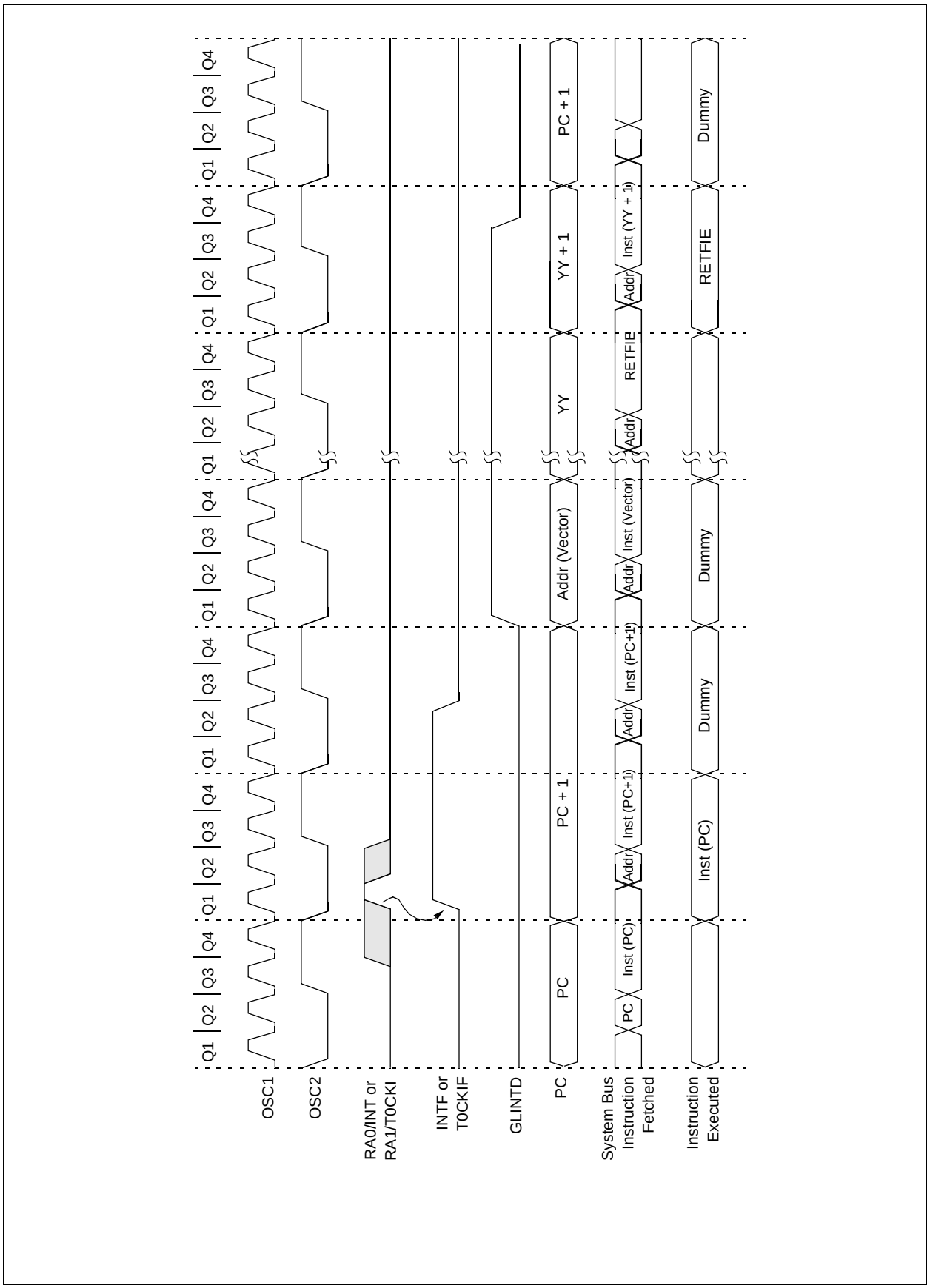


FIGURE 8-6: CONSECUTIVE TABLWT**WRITE TIMING (EXTERNAL MEMORY)**

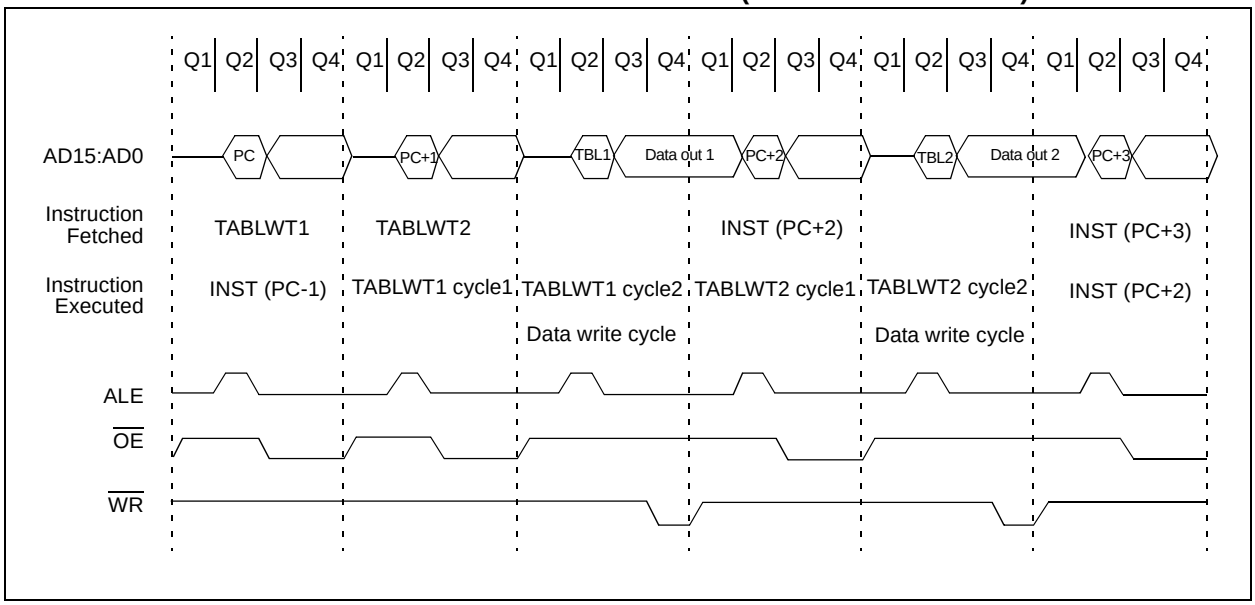
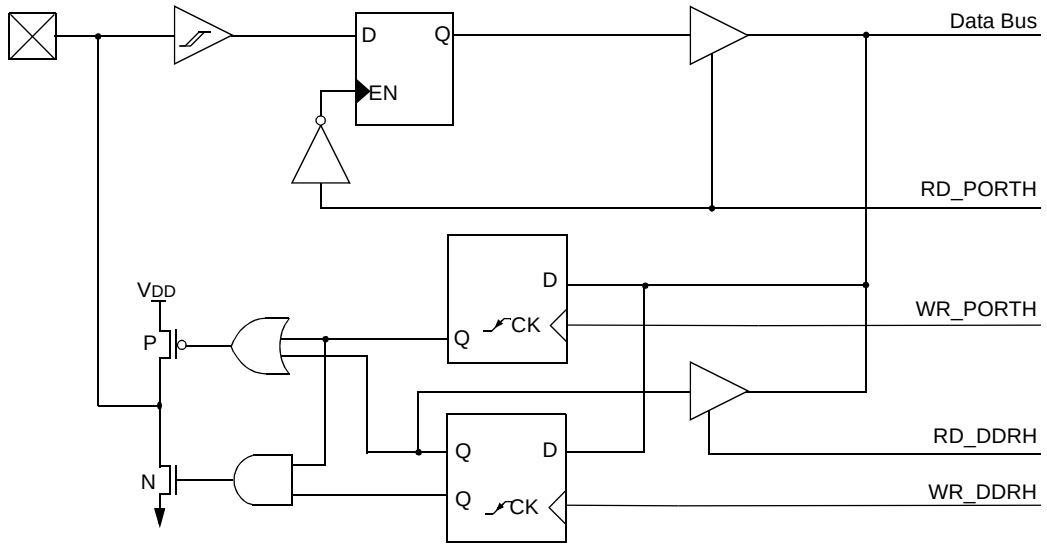


FIGURE 10-18: RH3:RH0 BLOCK DIAGRAM



Note: I/O pins have protection diodes to V_{DD} and V_{SS}.

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	$\overline{MCLR}$, 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

10.10.2 SUCCESSIVE OPERATIONS ON I/O PORTS

The actual write to an I/O port happens at the end of an instruction cycle, whereas for reading, the data must be valid at the beginning of the instruction cycle (Figure 10-20). Therefore, care must be exercised if a write followed by a read operation is carried out on the same I/O port. The sequence of instructions should be such to allow the pin voltage to stabilize (load dependent) before executing the instruction that reads the values on that I/O port. Otherwise, the previous state of that pin may be read into the CPU, rather than the "new" state. When in doubt, it is better to separate these instructions with a NOP, or another instruction not accessing this I/O port.

Figure 10-21 shows the I/O model which causes this situation. As the effective capacitance (C) becomes larger, the rise/fall time of the I/O pin increases. As the device frequency increases, or the effective capacitance increases, the possibility of this subsequent PORTx read-modify-write instruction issue increases. This effective capacitance includes the effects of the board traces.

The best way to address this is to add a series resistor at the I/O pin. This resistor allows the I/O pin to get to the desired level before the next instruction.

The use of NOPinstructions between the subsequent PORTx read-modify-write instructions, is a lower cost solution, but has the issue that the number of NOP instructions is dependent on the effective capacitance C and the frequency of the device.

FIGURE 10-20: SUCCESSIVE I/O OPERATION

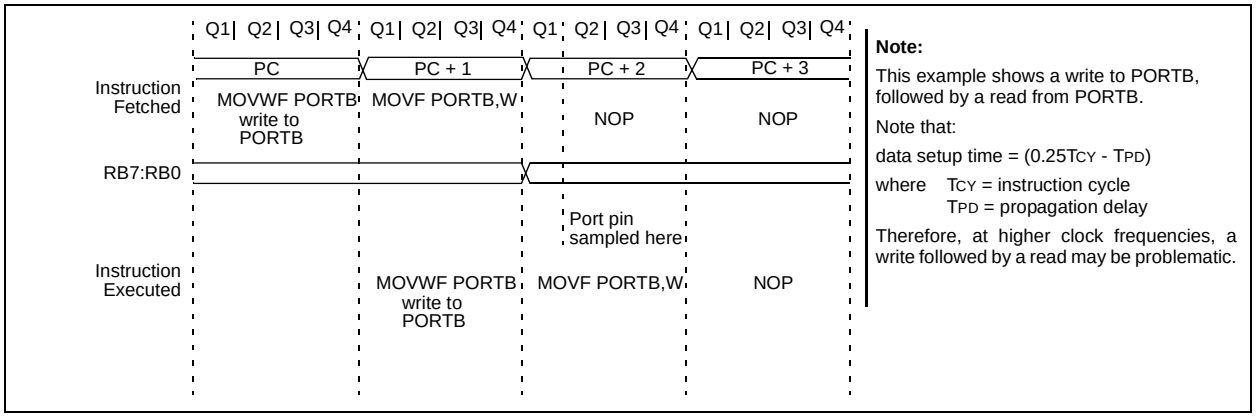
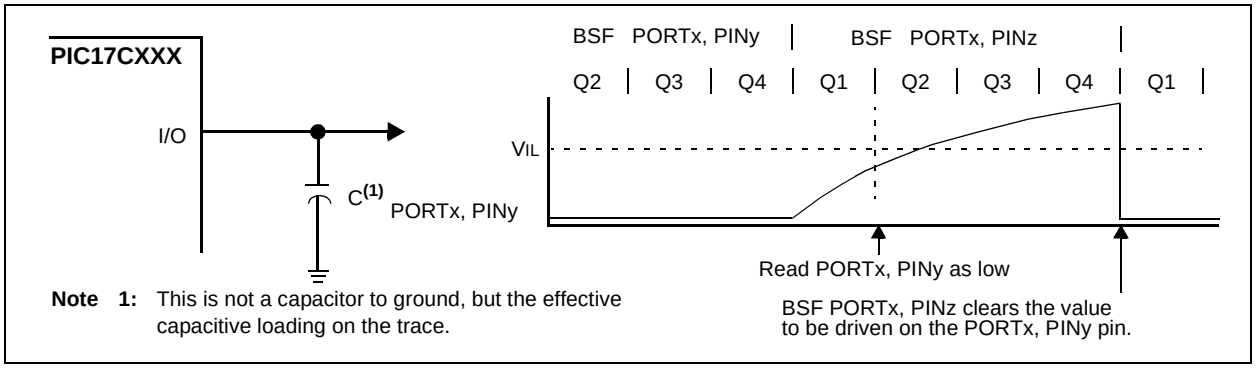


FIGURE 10-21: I/O CONNECTION ISSUES



Note 1: This is not a capacitor to ground, but the effective capacitive loading on the trace.

TABLE 14-4: BAUD RATES FOR SYNCHRONOUS MODE

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

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

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

15.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

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

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

Figure 15-1 shows a block diagram for the SPI mode, while Figure 15-2 and Figure 15-3 show the block diagrams for the two different I^2C modes of operation.

FIGURE 15-1: SPI MODE BLOCK DIAGRAM

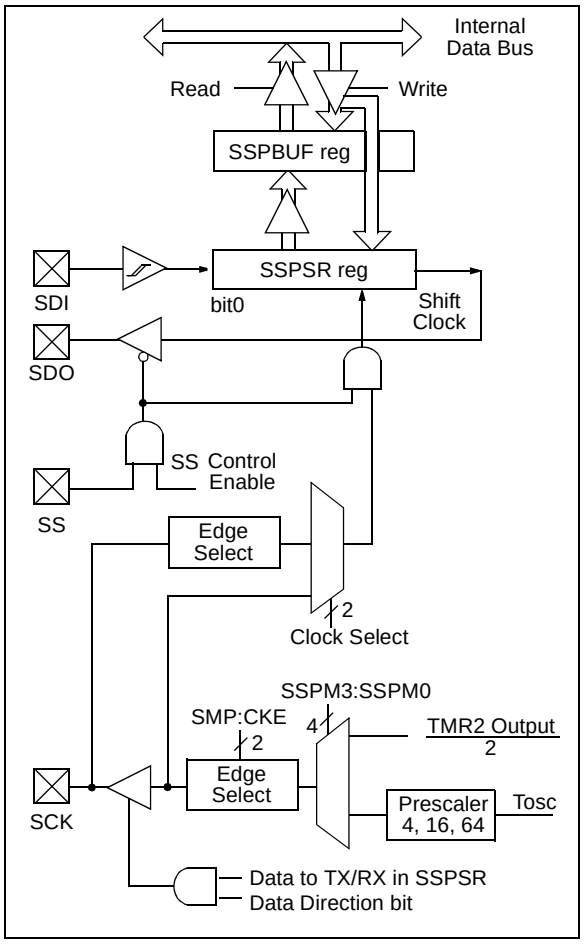


FIGURE 15-2: I^2C SLAVE MODE BLOCK DIAGRAM

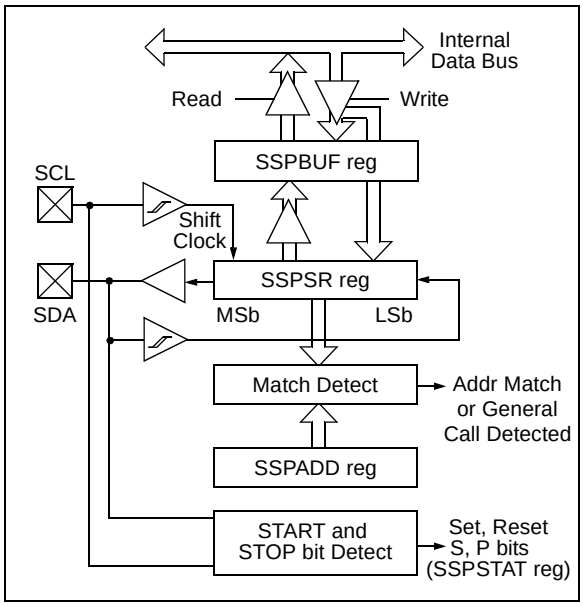
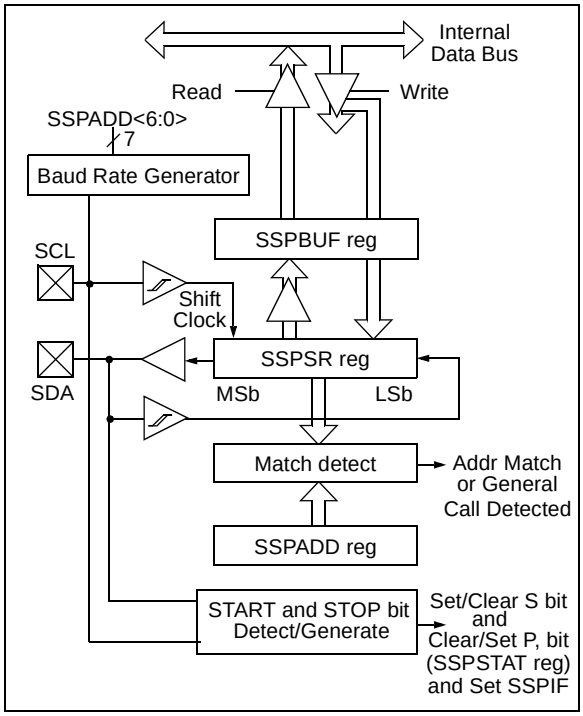


FIGURE 15-3: I^2C MASTER MODE BLOCK DIAGRAM



15.2.1.1 Addressing

Once the MSSP module has been enabled, it waits for a START condition to occur. Following the START condition, the 8-bits are shifted into the SSPSR register. All incoming bits are sampled with the rising edge of the clock (SCL) line. The value of register SSPSR<7:1> is compared to the value of the SSPADD register. The address is compared on the falling edge of the eighth clock (SCL) pulse. If the addresses match and the BF and SSPOV bits are clear, the following events occur:

a) The SSPSR register value is loaded into the SSPBUF register on the falling edge of the 8th SCL pulse.
b) The buffer full bit, BF, is set on the falling edge of the 8th SCL pulse.
c) An $\overline{ACK}$ pulse is generated.
d) SSP interrupt flag bit, SSPIF (PIR2<7>), is set (interrupt is generated if enabled) - on the falling edge of the 9th SCL pulse.

In 10-bit address mode, two address bytes need to be received by the slave. The five Most Significant bits (MSbs) of the first address byte specify if this is a 10-bit address. Bit R/$\overline{W}$ (SSPSTAT<2>) must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '1111 0 A9 A8 0 ', where A9 and A8 are the two MSbs of the address. The sequence of events for a 10-bit address is as follows, with steps 7- 9 for slave-transmitter:

1. Receive first (high) byte of Address (bits SSPIF, BF and bit UA (SSPSTAT<1>) are set).
2. Update the SSPADD register with second (low) byte of Address (clears bit UA and releases the SCL line).
3. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
4. Receive second (low) byte of Address (bits SSPIF, BF and UA are set).
5. Update the SSPADD register with the first (high) byte of Address. This will clear bit UA and release the SCL line.
6. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
7. Receive Repeated Start condition.
8. Receive first (high) byte of Address (bits SSPIF and BF are set).
9. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.

Note:	Following the Repeated Start condition (step 7) in 10-bit mode, the user only needs to match the first 7-bit address. The user does not update the SSPADD for the second half of the address.

15.2.1.2 Slave Reception

When the R/$\overline{W}$ bit of the address byte is clear and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register.

When the address byte overflow condition exists, then no acknowledge ($\overline{ACK}$) pulse is given. An overflow condition is defined as either bit BF (SSPSTAT<0>) is set, or bit SSPOV (SSPCON1<6>) is set.

An SSP interrupt is generated for each data transfer byte. Flag bit SSPIF (PIR2<7>) must be cleared in software. The SSPSTAT register is used to determine the status of the received byte.

Note:	The SSPBUF will be loaded if the SSPOV bit is set and the BF flag is cleared. If a read of the SSPBUF was performed, but the user did not clear the state of the SSPOV bit before the next receive occurred, the $\overline{ACK}$ is not sent and the SSPBUF is updated.

TABLE 15-2: DATA TRANSFER RECEIVED BYTE ACTIONS

Status Bits as Data Transfer is Received		SSPSR → SSPBUF	Generate $\overline{ACK}$ Pulse	Set bit SSPIF (SSP Interrupt occurs if enabled)
BF	SSPOV			
0	0	Yes	Yes	Yes
1	0	No	No	Yes
1	1	No	No	Yes
0	1	Yes	No	Yes

Note 1: Shaded cells show the conditions where the user software did not properly clear the overflow condition.

15.2.10 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit (SSPCON2<1>) is programmed high and the I²C module is in the idle state. When the RSEN bit is set, the SCL pin is asserted low. When the SCL pin is sampled low, the baud rate generator is loaded with the contents of SSPADD<6:0> and begins counting. The SDA pin is released (brought high) for one baud rate generator count (TBRG). When the baud rate generator times out, if SDA is sampled high, the SCL pin will be de-asserted (brought high). When SCL is sampled high the baud rate generator is reloaded with the contents of SSPADD<6:0> and begins counting. SDA and SCL must be sampled high for one TBRG. This action is then followed by assertion of the SDA pin (SDA is low) for one TBRG while SCL is high. Following this, the RSEN bit in the SSPCON2 register will be automatically cleared and the baud rate generator is not reloaded, leaving the SDA pin held low. As soon as a START condition is detected on the SDA and SCL pins, the S bit (SSPSTAT<3>) will be set. The SSPIF bit will not be set until the baud rate generator has timed out.

Note 1: If the RSEN is programmed while any other event is in progress, it will not take effect.

2: A bus collision during the Repeated Start condition occurs if:

- SDA is sampled low when SCL goes from low to high.
- SCL goes low before SDA is asserted low. This may indicate that another master is attempting to transmit a data "1".

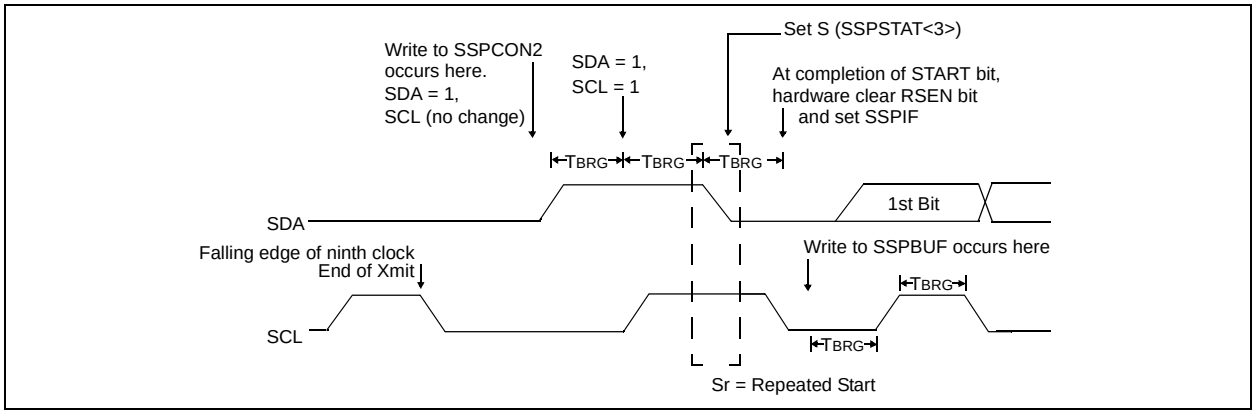
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode), or eight bits of data (7-bit mode).

15.2.10.1 WCOL status flag

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

Note: Because queueing of events is not allowed, writing of the lower 5 bits of SSPCON2 is disabled until the Repeated Start condition is complete.

FIGURE 15-22: REPEAT START CONDITION WAVEFORM



15.2.18 MULTI -MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high and another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin = '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLIF and reset the I^2C port to its IDLE state (Figure 15-34).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDA and SCL lines are de-asserted and the SSPBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I^2C bus is free, the user can resume communication by asserting a START condition.

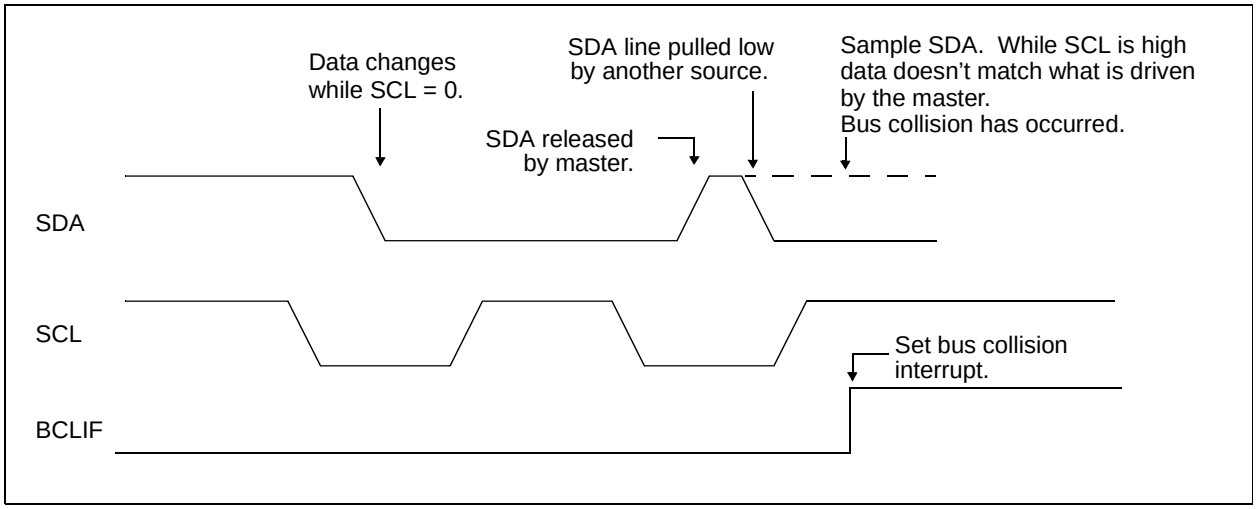
If a START, Repeated Start, STOP, or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are de-asserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine, and if the I^2C bus is free, the user can resume communication by asserting a START condition.

The master will continue to monitor the SDA and SCL pins and if a STOP condition occurs, the SSPIF bit will be set.

A write to the SSPBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of START and STOP conditions allows the determination of when the bus is free. Control of the I^2C bus can be taken when the P bit is set in the SSPSTAT register, or the bus is idle and the S and P bits are cleared.

FIGURE 15-34: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE



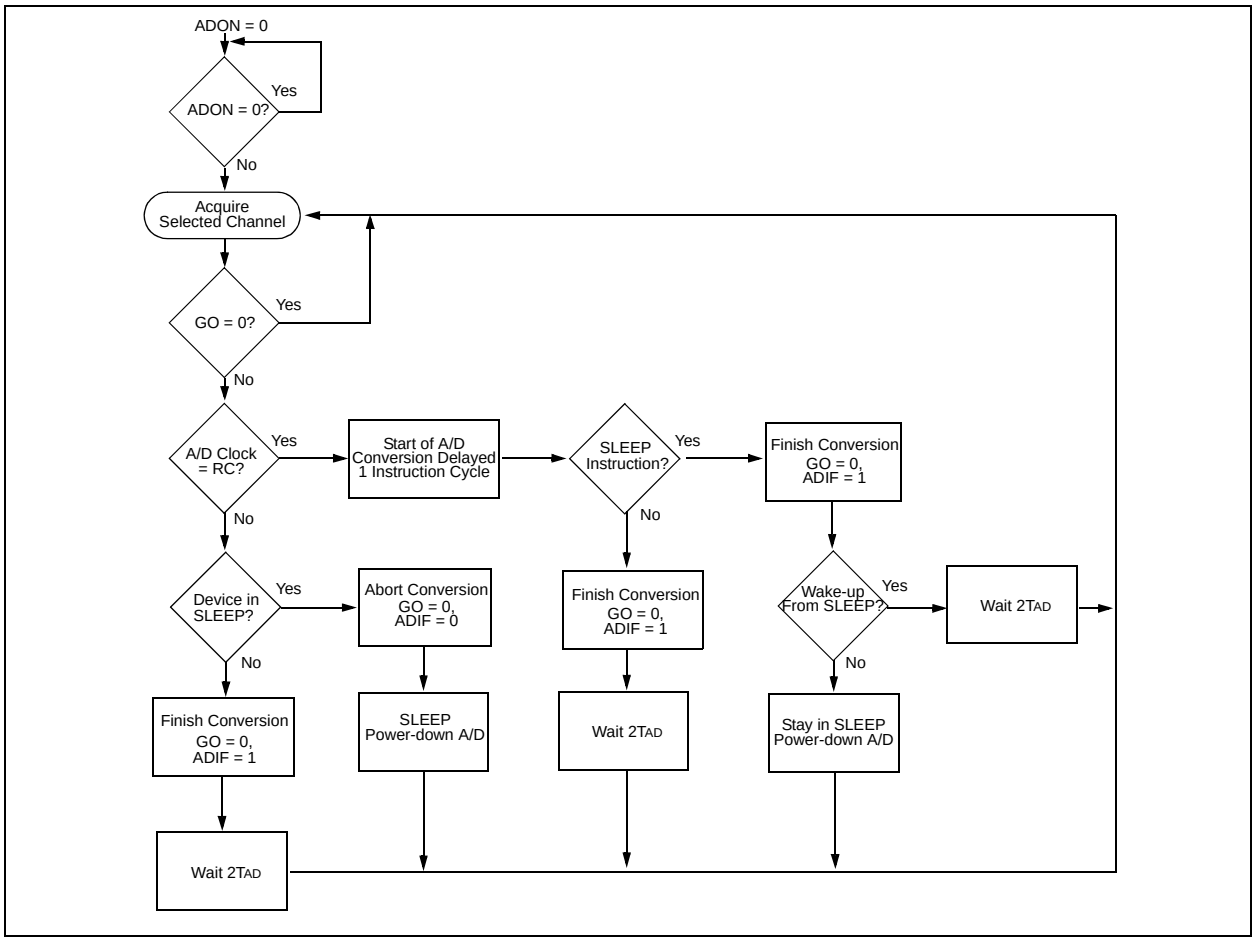
15.4 Example Program

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

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

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

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

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

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

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

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

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

FIGURE 16-5: FLOW CHART OF A/D OPERATION



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/V_{REF+}	RG2/ AN1/V_{REF-}	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.

ADDLW **ADD Literal to WREG**

Syntax: [label] ADDLW k

Operands: $0 \le k \le 255$

Operation: (WREG) + k → (WREG)

Status Affected: OV, C, DC, Z

Encoding:

1011	0001	kkkk	kkkk

Description: The contents of WREG are added to the 8-bit literal 'k' and the result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example: ADDLW 0x15

Before Instruction
WREG = 0x10

After Instruction
WREG = 0x25

ADDWF **ADD WREG to f**

Syntax: [label] ADDWF f,d

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

Operation: (WREG) + (f) → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0000	111d	ffff	ffff

Description: Add WREG to register 'f'. If 'd' is 0 the result is stored in WREG. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: ADDWF REG, 0

Before Instruction
WREG = 0x17
REG = 0xC2

After Instruction
WREG = 0xD9
REG = 0xC2

CLRWDT **Clear Watchdog Timer**

Syntax: [label] CLRWDT

Operands: None

Operation: 00h → WDT
0 → $\overline{\text{WDT}}$ postscaler,
1 → $\overline{\text{TO}}$
1 → $\overline{\text{PD}}$

Status Affected: $\overline{\text{TO}}$, $\overline{\text{PD}}$

Encoding:

0000	0000	0000	0100

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits $\overline{\text{TO}}$ and $\overline{\text{PD}}$ are set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	No operation

Example: CLRWDT

Before Instruction
WDT counter = ?

After Instruction
WDT counter = 0x00
WDT Postscaler = 0
$\overline{\text{TO}}$ = 1
$\overline{\text{PD}}$ = 1

COMF **Complement f**

Syntax: [*label*] COMF f,d

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

Operation: $(\bar{f}) \rightarrow$ (dest)

Status Affected: Z

Encoding:

0001	001d	ffff	ffff

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

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: COMF REG1,0

Before Instruction
REG1 = 0x13

After Instruction
REG1 = 0x13
WREG = 0xEC

NOTES:

19.13 PICDEM 3 Low Cost PIC16CXXX Demonstration Board

The PICDEM 3 demonstration board is a simple demonstration board that supports the PIC16C923 and PIC16C924 in the PLCC package. It will also support future 44-pin PLCC microcontrollers with an LCD Module. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM 3 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer with an adapter socket, and easily test firmware. The MPLAB ICE in-circuit emulator may also be used with the PICDEM 3 demonstration board to test firmware. A prototype area has been provided to the user for adding hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push button switches, a potentiometer for simulated analog input, a thermistor and separate headers for connection to an external LCD module and a keypad. Also provided on the PICDEM 3 demonstration board is a LCD panel, with 4 commons and 12 segments, that is capable of displaying time, temperature and day of the week. The PICDEM 3 demonstration board provides an additional RS-232 interface and Windows software for showing the demultiplexed LCD signals on a PC. A simple serial interface allows the user to construct a hardware demultiplexer for the LCD signals.

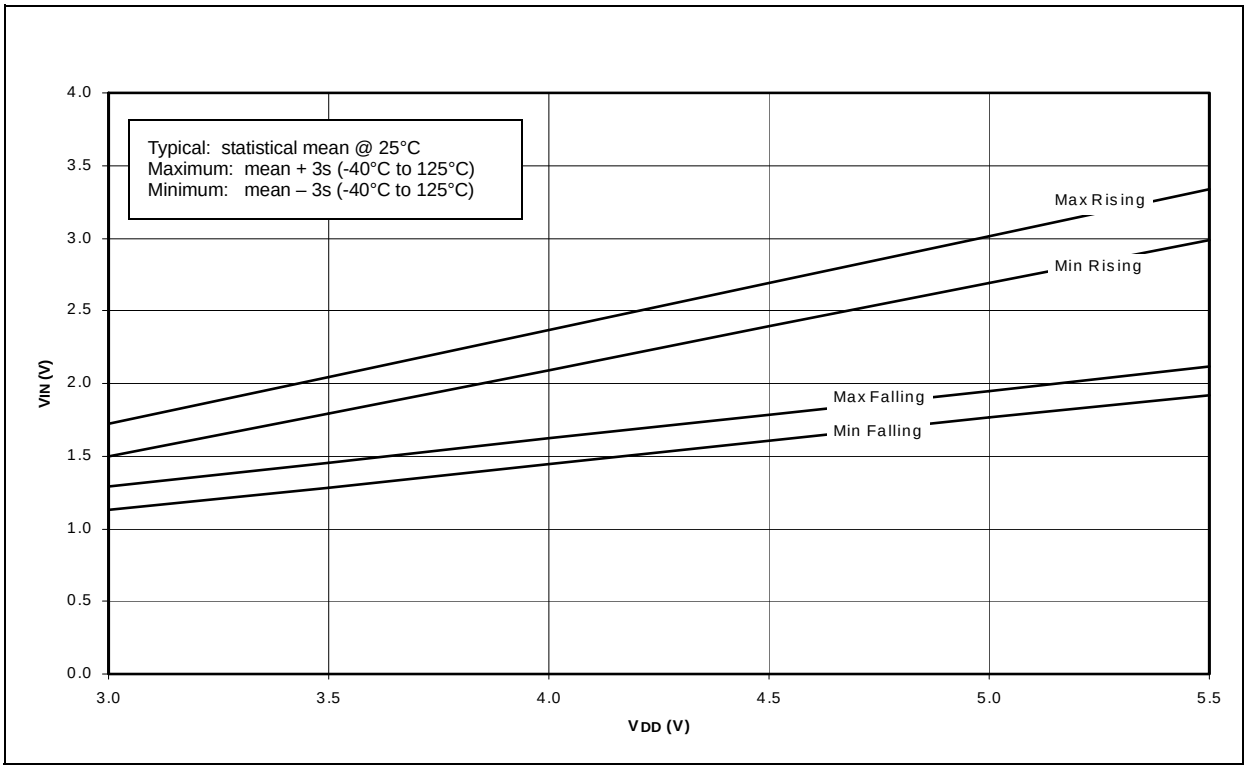
19.14 PICDEM 17 Demonstration Board

The PICDEM 17 demonstration board is an evaluation board that demonstrates the capabilities of several Microchip microcontrollers, including PIC17C752, PIC17C756A, PIC17C762 and PIC17C766. All necessary hardware is included to run basic demo programs, which are supplied on a 3.5-inch disk. A programmed sample is included and the user may erase it and program it with the other sample programs using the PRO MATE II device programmer, or the PICSTART Plus development programmer, and easily debug and test the sample code. In addition, the PICDEM 17 demonstration board supports downloading of programs to and executing out of external FLASH memory on board. The PICDEM 17 demonstration board is also usable with the MPLAB ICE in-circuit emulator, or the 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.

19.15 KEELOQ Evaluation and Programming Tools

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

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



Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

Printed on recycled paper.

ISBN: 9781620769317

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.