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	0°C ~ 70°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/pic17lc762t-08-l

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

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

9.0 HARDWARE MULTIPLIER

All PIC17C7XX devices have an 8 x 8 hardware multiplier included in the ALU of the device. By making the multiply a hardware operation, it completes in a single instruction cycle. This is an unsigned multiply that gives a 16-bit result. The result is stored into the 16-bit Product register (PRODH:PRODL). The multiplier does not affect any flags in the ALUSTA register.

Making the 8 x 8 multiplier execute in a single cycle gives the following advantages:

- Higher computational throughput
- Reduces code size requirements for multiply algorithms

The performance increase allows the device to be used in applications previously reserved for Digital Signal Processors.

Table 9-1 shows a performance comparison between PIC17CXXX devices using the single cycle hardware multiply and performing the same function without the hardware multiply.

Example 9-1 shows the sequence to do an 8 x 8 unsigned multiply. Only one instruction is required when one argument of the multiply is already loaded in the WREG register.

Example 9-2 shows the sequence to do an 8 x 8 signed multiply. To account for the sign bits of the arguments, each argument's most significant bit (MSb) is tested and the appropriate subtractions are done.

EXAMPLE 9-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

```
MOVFP    ARG1, WREG  ;
MULWF    ARG2        ; ARG1 * ARG2 ->
                     ;   PRODH:PRODL
```

EXAMPLE 9-2: 8 x 8 SIGNED MULTIPLY ROUTINE

```
MOVFP    ARG1, WREG
MULWF    ARG2        ; ARG1 * ARG2 ->
                     ;   PRODH:PRODL
BTFSC    ARG2, SB    ; Test Sign Bit
SUBWF    PRODH, F    ; PRODH = PRODH
                     ;         - ARG1
MOVFP    ARG2, WREG
BTFSC    ARG1, SB    ; Test Sign Bit
SUBWF    PRODH, F    ; PRODH = PRODH
                     ;         - ARG2
```

TABLE 9-1: PERFORMANCE COMPARISON

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time		
				@ 33 MHz	@ 16 MHz	@ 8 MHz
8 x 8 unsigned	Without hardware multiply	13	69	8.364 μs	17.25 μs	34.50 μs
	Hardware multiply	1	1	0.121 μs	0.25 μs	0.50 μs
8 x 8 signed	Without hardware multiply	—	—	—	—	—
	Hardware multiply	6	6	0.727 μs	1.50 μs	3.0 μs
16 x 16 unsigned	Without hardware multiply	21	242	29.333 μs	60.50 μs	121.0 μs
	Hardware multiply	24	24	2.91 μs	6.0 μs	12.0 μs
16 x 16 signed	Without hardware multiply	52	254	30.788 μs	63.50 μs	127.0 μs
	Hardware multiply	36	36	4.36 μs	9.0 μs	18.0 μs

© 1998-2013 Microchip Technology Inc.

TABLE 10-5: PORTC FUNCTIONS

Name	Bit	Buffer Type	Function
RC0/AD0	bit0	TTL	Input/output or system bus address/data pin.
RC1/AD1	bit1	TTL	Input/output or system bus address/data pin.
RC2/AD2	bit2	TTL	Input/output or system bus address/data pin.
RC3/AD3	bit3	TTL	Input/output or system bus address/data pin.
RC4/AD4	bit4	TTL	Input/output or system bus address/data pin.
RC5/AD5	bit5	TTL	Input/output or system bus address/data pin.
RC6/AD6	bit6	TTL	Input/output or system bus address/data pin.
RC7/AD7	bit7	TTL	Input/output or system bus address/data pin.

Legend: TTL = TTL input

TABLE 10-6: REGISTERS/BITS ASSOCIATED WITH PORTC

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	$\overline{\text{MCLR}}$, WDT
11h, Bank 1	PORTC	RC7/ AD7	RC6/ AD6	RC5/ AD5	RC4/ AD4	RC3/ AD3	RC2/ AD2	RC1/ AD1	RC0/ AD0	xxxx xxxx	uuuu uuuu
10h, Bank 1	DDRC	Data Direction Register for PORTC								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

TABLE 10-7: PORTD FUNCTIONS

Name	Bit	Buffer Type	Function
RD0/AD8	bit0	TTL	Input/output or system bus address/data pin.
RD1/AD9	bit1	TTL	Input/output or system bus address/data pin.
RD2/AD10	bit2	TTL	Input/output or system bus address/data pin.
RD3/AD11	bit3	TTL	Input/output or system bus address/data pin.
RD4/AD12	bit4	TTL	Input/output or system bus address/data pin.
RD5/AD13	bit5	TTL	Input/output or system bus address/data pin.
RD6/AD14	bit6	TTL	Input/output or system bus address/data pin.
RD7/AD15	bit7	TTL	Input/output or system bus address/data pin.

Legend: TTL = TTL input

TABLE 10-8: REGISTERS/BITS ASSOCIATED WITH PORTD

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
13h, Bank 1	PORTD	RD7/ AD15	RD6/ AD14	RD5/ AD13	RD4/ AD12	RD3/ AD11	RD2/ AD10	RD1/ AD9	RD0/ AD8	xxxx xxxx	uuuu uuuu
12h, Bank 1	DDRD	Data Direction Register for PORTD								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

10.6 PORTF and DDRF Registers

PORTF is an 8-bit wide bi-directional port. The corresponding data direction register is DDRF. A '1' in DDRF configures the corresponding port pin as an input. A '0' in the DDRF register configures the corresponding port pin as an output. Reading PORTF reads the status of the pins, whereas writing to PORTF will write to the respective port latch.

All eight bits of PORTF are multiplexed with 8 channels of the 10-bit A/D converter.

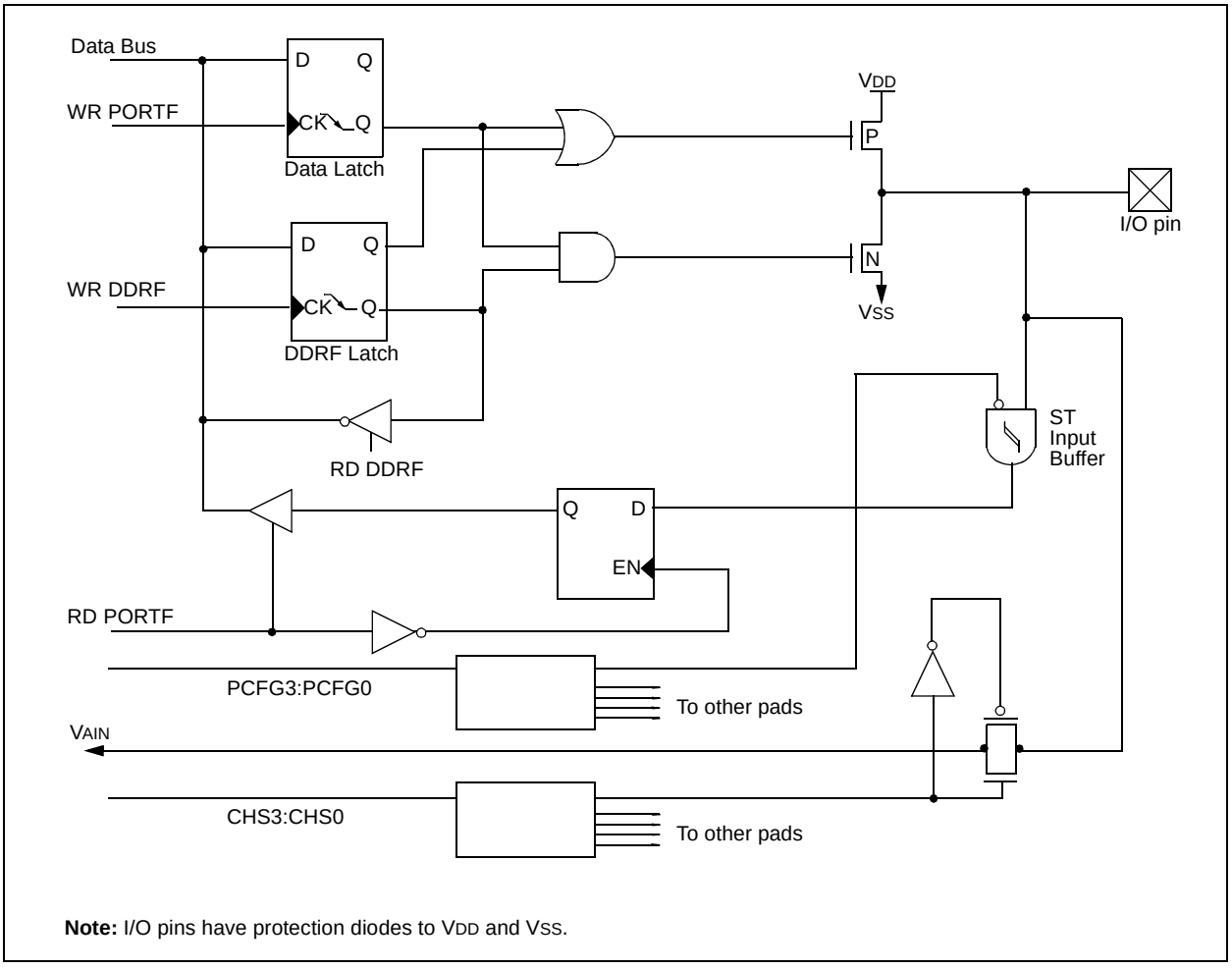
Upon RESET, the entire Port is automatically configured as analog inputs and must be configured in software to be a digital I/O.

Example 10-6 shows an instruction sequence to initialize PORTF. The Bank Select Register (BSR) must be selected to Bank 5 for the port to be initialized. The following example uses the `MOVLB` instruction to load the BSR register for bank selection.

EXAMPLE 10-6: INITIALIZING PORTF

```
MOVLB   5          ; Select Bank 5
MOVWF   0x0E       ; Configure PORTF as
MOVWF   ADCON1     ; Digital
CLRF    PORTF, F   ; Initialize PORTF data
                   ;   latches before
                   ;   the data direction
                   ;   register
MOVLW    0x03      ; Value used to init
                   ;   data direction
MOVWF    DDRF      ; Set RF<1:0> as inputs
                   ;   RF<7:2> as outputs
```

FIGURE 10-13: BLOCK DIAGRAM OF RF7:RF0



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

© 1998-2013 Microchip Technology Inc.

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

$$\text{Error} = \frac{\text{(Calculated Baud Rate - Desired Baud Rate)}}{\text{Desired Baud Rate}}$$

= (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	$\overline{\text{MCLR}}$, 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.

© 1998-2013 Microchip Technology Inc.

14.3 USART Synchronous Master Mode

In Master Synchronous mode, the data is transmitted in a half-duplex manner; i.e., transmission and reception do not occur at the same time: when transmitting data, the reception is inhibited and vice versa. The synchronous mode is entered by setting the SYNC (TXSTA<4>) bit. In addition, the SPEN (RCSTA<7>) bit is set in order to configure the I/O pins to CK (clock) and DT (data) lines, respectively. The Master mode indicates that the processor transmits the master clock on the CK line. The Master mode is entered by setting the CSRC (TXSTA<7>) bit.

14.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 14-1. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer TXREG. TXREG is loaded with data in software. The TSR is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from TXREG (if available). Once TXREG transfers the data to the TSR (occurs in one T_{CY} at the end of the current BRG cycle), TXREG is empty and the TXIF bit is set. This interrupt can be enabled/disabled by setting/clearing the TXIE bit. TXIF will be set regardless of the state of bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into TXREG. While TXIF indicates the status of TXREG, TRMT (TXSTA<1>) shows the status of the TSR. TRMT is a read only bit which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR is empty. The TSR is not mapped in data memory, so it is not available to the user.

Transmission is enabled by setting the TXEN (TXSTA<5>) bit. The actual transmission will not occur until TXREG has been loaded with data. The first data bit will be shifted out on the next available rising edge of the clock on the TX/CK pin. Data out is stable around the falling edge of the synchronous clock (Figure 14-9). The transmission can also be started by first loading TXREG and then setting TXEN. This is advantageous when slow baud rates are selected, since BRG is kept in RESET when the TXEN, CREN, and SREN bits are clear. Setting the TXEN bit will start the BRG, creating a shift clock immediately. Normally when transmission is first started, the TSR is empty, so a transfer to TXREG will result in an immediate transfer to the TSR, resulting in an empty TXREG. Back-to-back transfers are possible.

Clearing TXEN during a transmission will cause the transmission to be aborted and will reset the transmitter. The RX/DT and TX/CK pins will revert to hi-impedance. If either CREN or SREN are set during a transmission, the transmission is aborted and the RX/DT pin reverts to a hi-impedance state (for a reception). The TX/CK pin will remain an output if the CSRC bit is set (internal clock). The transmitter logic is not reset, although it is disconnected from the pins. In order to reset the transmitter, the user has to clear the TXEN bit. If the SREN bit is set (to interrupt an ongoing transmission and receive a single word), then after the single word is received, SREN will be cleared and the serial port will revert back to transmitting, since the TXEN bit is still set. The DT line will immediately switch from hi-impedance Receive mode to transmit and start driving. To avoid this, TXEN should be cleared.

In order to select 9-bit transmission, the TX9 (TXSTA<6>) bit should be set and the ninth bit should be written to TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to TXREG. This is because a data write to TXREG can result in an immediate transfer of the data to the TSR (if the TSR is empty). If the TSR was empty and TXREG was written before writing the “new” TX9D, the “present” value of TX9D is loaded.

Steps to follow when setting up a Synchronous Master Transmission:

1. Initialize the SPBRG register for the appropriate baud rate (see Baud Rate Generator Section for details).
2. Enable the synchronous master serial port by setting the SYNC, SPEN, and CSRC bits.
3. Ensure that the CREN and SREN bits are clear (these bits override transmission when set).
4. If interrupts are desired, then set the TXIE bit (the GLINTD bit must be clear and the PEIE bit must be set).
5. If 9-bit transmission is desired, then set the TX9 bit.
6. If 9-bit transmission is selected, the ninth bit should be loaded in TX9D.
7. Start transmission by loading data to the TXREG register.
8. Enable the transmission by setting TXEN.

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

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

When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. Buffer full bit, BF (SSPSTAT<0>), indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, bit BF is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally the MSSP interrupt is used to determine when the transmission/reception has completed. The SSPBUF must be read and/or written. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur. Example 15-1 shows the loading of the SSPBUF (SSPSR) for data transmission.

EXAMPLE 15-1: LOADING THE SSPBUF (SSPSR) REGISTER

```
     MOVLB 6                ; Bank 6
LOOP BTFSS SSPSTAT, BF      ; Has data been
                            ;   received
                            ;   (transmit
                            ;   complete)?
     GOTO  LOOP             ; No
     MOVPF SSPBUF, RXDATA   ; Save in user RAM
     MOVFP TXDATA, SSPBUF   ; New data to xmit
```

The SSPSR is not directly readable, or writable and can only be accessed by addressing the SSPBUF register. Additionally, the MSSP status register (SSPSTAT) indicates the various status conditions.

15.1.2 ENABLING SPI I/O

To enable the serial port, MSSP Enable bit, SSPEN (SSPCON1<5>), must be set. To reset or reconfigure SPI mode, clear bit SSPEN, re-initialize the SSPCON registers and then set bit SSPEN. This configures the SDI, SDO, SCK and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the DDR register) appropriately programmed. That is:

- SDI is automatically controlled by the SPI module
- SDO must have DDRB<7> cleared
- SCK (Master mode) must have DDRB<6> cleared
- SCK (Slave mode) must have DDRB<6> set
- $\overline{SS}$ must have PORTA<2> set

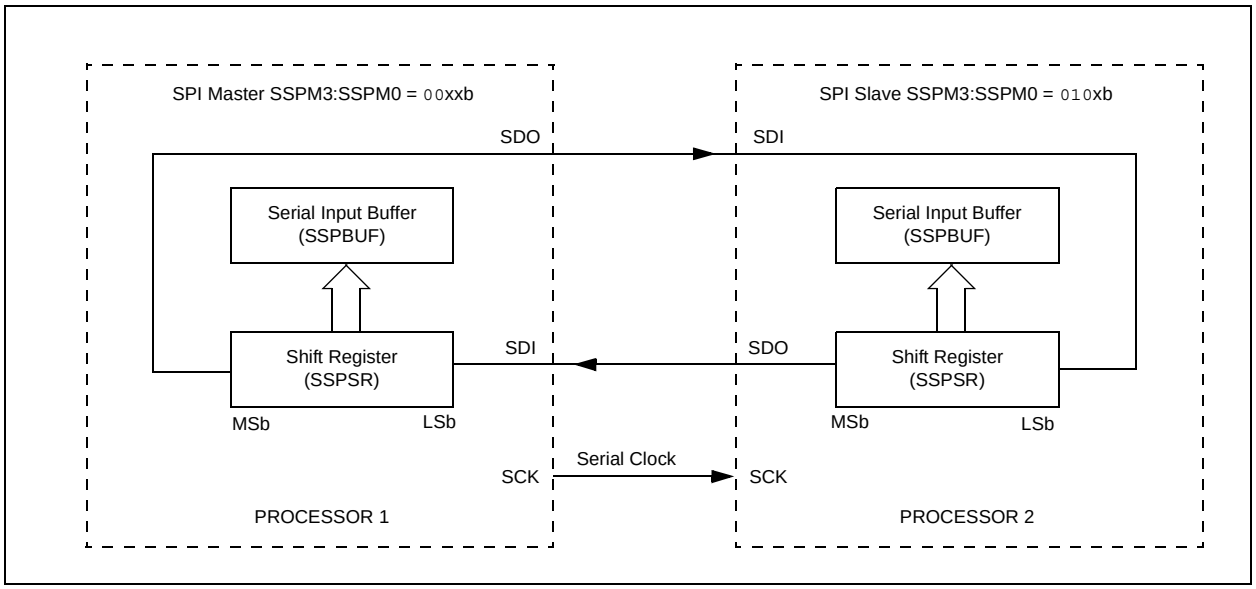
Any serial port function that is not desired may be overridden by programming the corresponding data direction (DDR) register to the opposite value.

15.1.3 TYPICAL CONNECTION

Figure 15-5 shows a typical connection between two microcontrollers. The master controller (Processor 1) initiates the data transfer by sending the SCK signal. Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock. Both processors should be programmed to same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends data — Slave sends dummy data
- Master sends data — Slave sends data
- Master sends dummy data — Slave sends data

FIGURE 15-5: SPI MASTER/SLAVE CONNECTION



© 1998-2013 Microchip Technology Inc.

15.2.15 CLOCK ARBITRATION

Clock arbitration occurs when the master, during any receive, transmit, or Repeated Start/Stop condition, deasserts the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the baud rate generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the baud rate generator is reloaded with the contents of SSPADD<6:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count, in the event that the clock is held low by an external device (Figure 15-33).

15.2.16 SLEEP OPERATION

While in SLEEP mode, the I^2C 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).

15.2.17 EFFECTS OF A RESET

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

FIGURE 15-33: CLOCK ARBITRATION TIMING IN MASTER TRANSMIT MODE

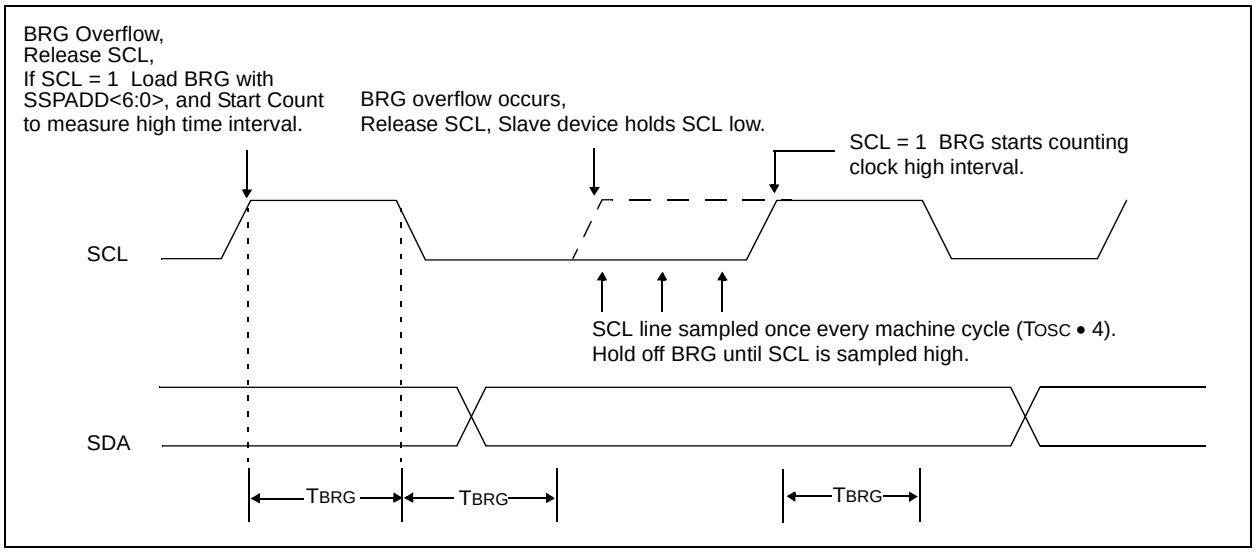


FIGURE 15-36: BUS COLLISION DURING START CONDITION (SCL = 0)

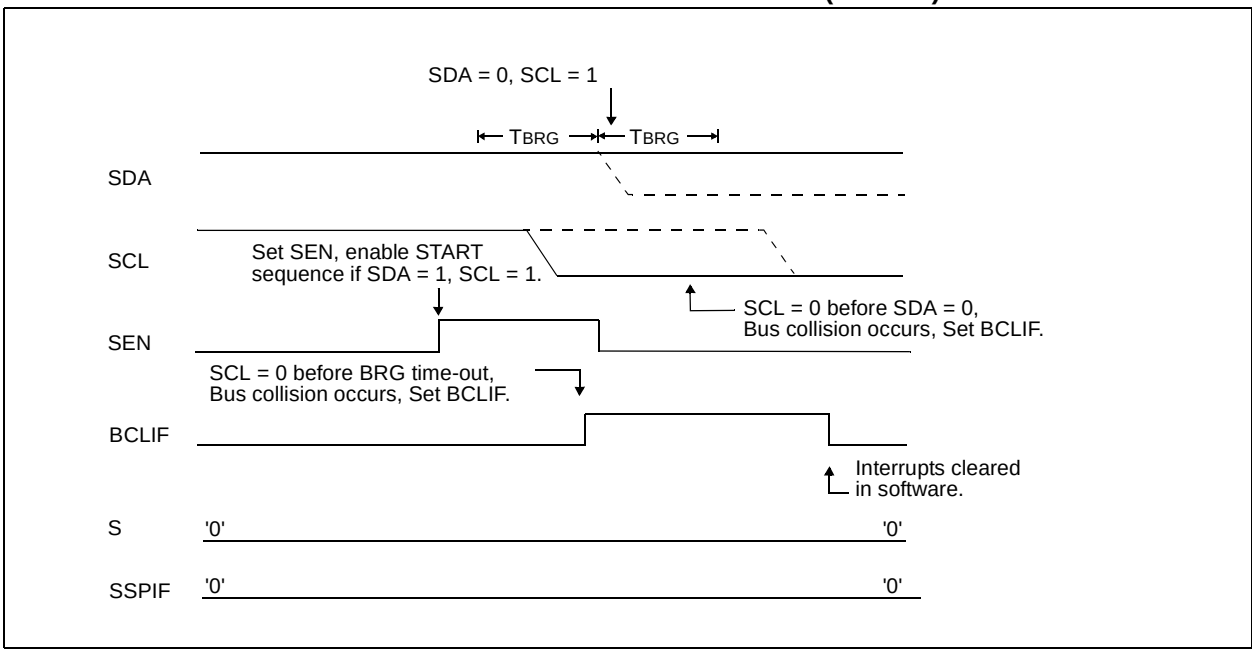
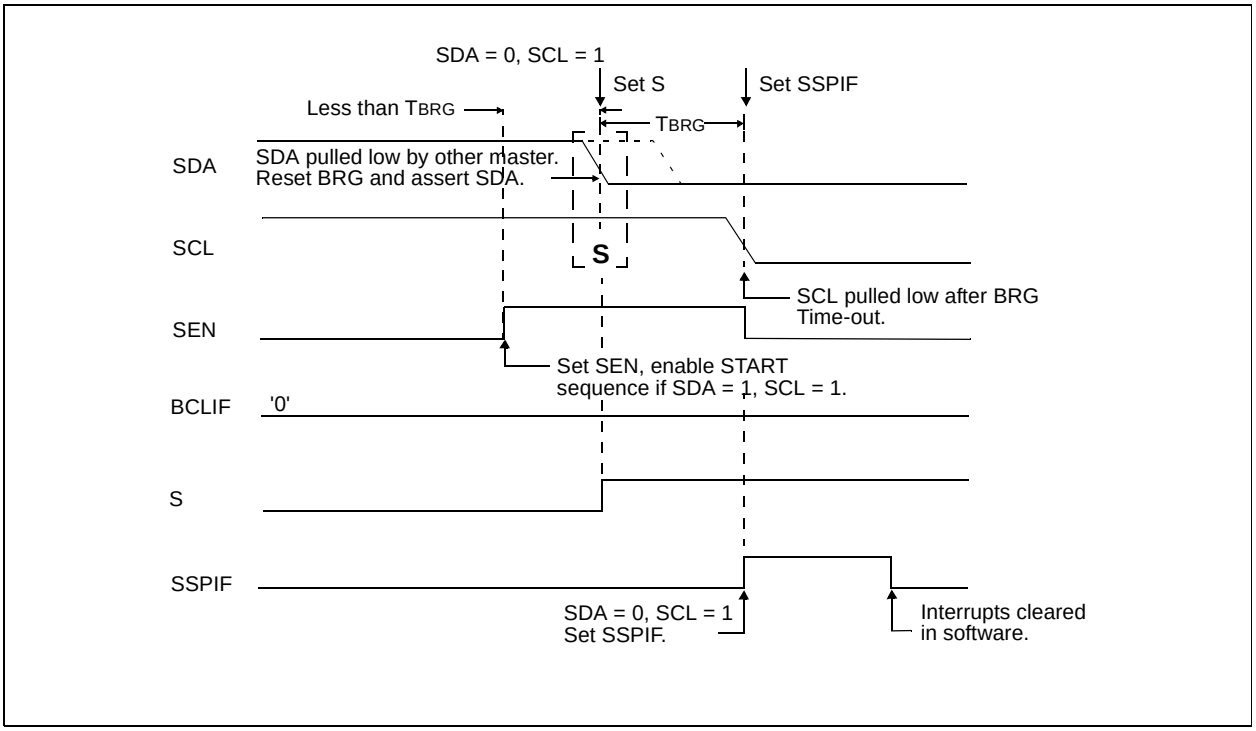


FIGURE 15-37: BRG RESET DUE TO SDA COLLISION DURING START CONDITION



© 1998-2013 Microchip Technology Inc.

15.2.18.3 Bus Collision During a STOP Condition

Bus collision occurs during a STOP condition if:

a) After the SDA pin has been de-asserted and allowed to float high, SDA is sampled low after the BRG has timed out.

b) After the SCL pin is de-asserted, SCL is sampled low before SDA goes high.

The STOP condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the baud rate generator is loaded with SSPADD<6:0> and counts down to ‘0’. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0'. If the SCL pin is sampled low before SDA is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 15-40).

FIGURE 15-40: BUS COLLISION DURING A STOP CONDITION (CASE 1)

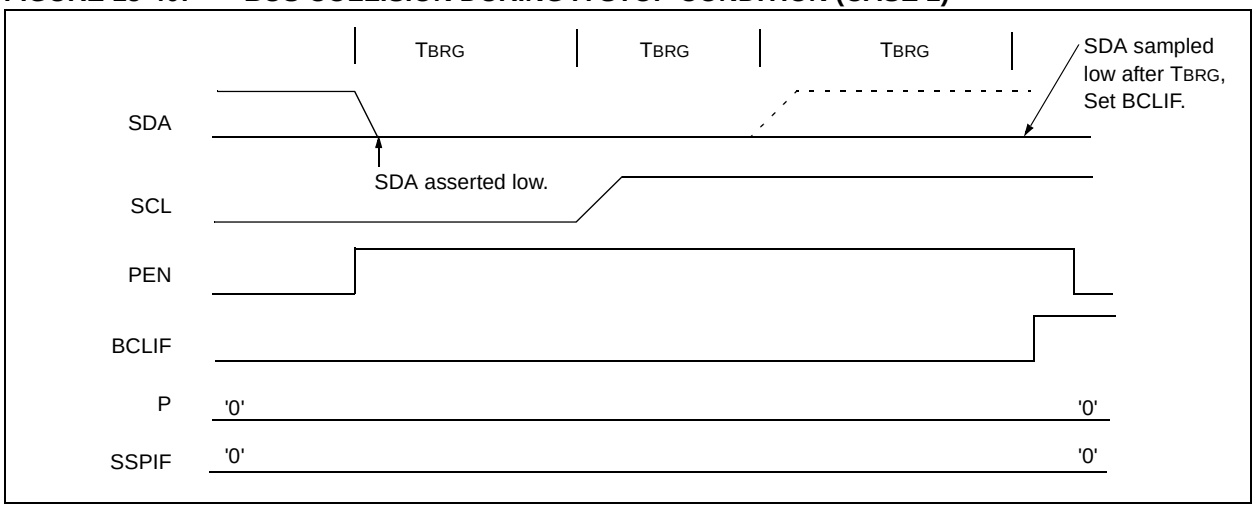
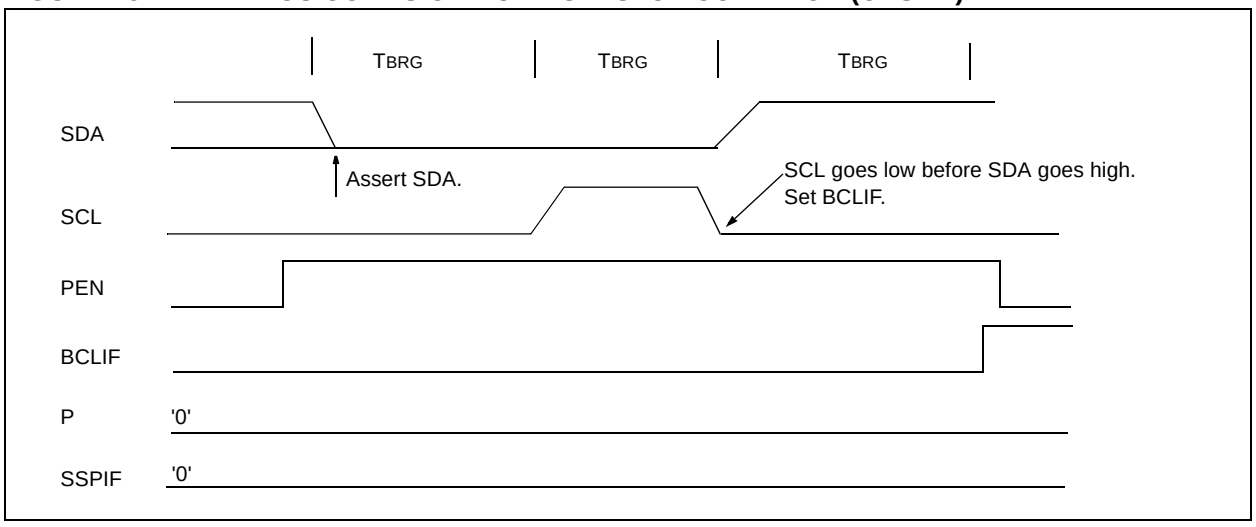


FIGURE 15-41: BUS COLLISION DURING A STOP CONDITION (CASE 2)



© 1998-2013 Microchip Technology Inc.

16.4.1 A/D RESULT REGISTERS

The ADRESH:ADRESL register pair is the location where the 10-bit A/D result is loaded at the completion of the A/D conversion. This register pair is 16-bits wide. The A/D module gives the flexibility to left or right justify the 10-bit result in the 16-bit result register. The A/D Format Select bit (ADFM) controls this justification. Figure 16-6 shows the operation of the A/D result justification. The extra bits are loaded with '0's'. When an A/D result will not overwrite these locations (A/D disable), these registers may be used as two general purpose 8-bit registers.

16.5 A/D Operation During SLEEP

The A/D module can operate during SLEEP mode. This requires that the A/D clock source be set to RC (ADCS1:ADCS0 = 11). When the RC clock source is selected, the A/D module waits one instruction cycle before starting the conversion. This allows the SLEEP instruction to be executed, which eliminates all digital switching noise from the conversion. When the conversion is completed, the GO/DONE bit will be cleared, and the result loaded into the ADRES register. If the A/D interrupt is enabled, the device will wake-up from SLEEP. If the A/D interrupt is not enabled, the A/D module will then be turned off, although the ADON bit will remain set.

When the A/D clock source is another clock option (not RC), a SLEEP instruction will cause the present conversion to be aborted and the A/D module to be turned off, though the ADON bit will remain set.

Turning off the A/D places the A/D module in its lowest current consumption state.

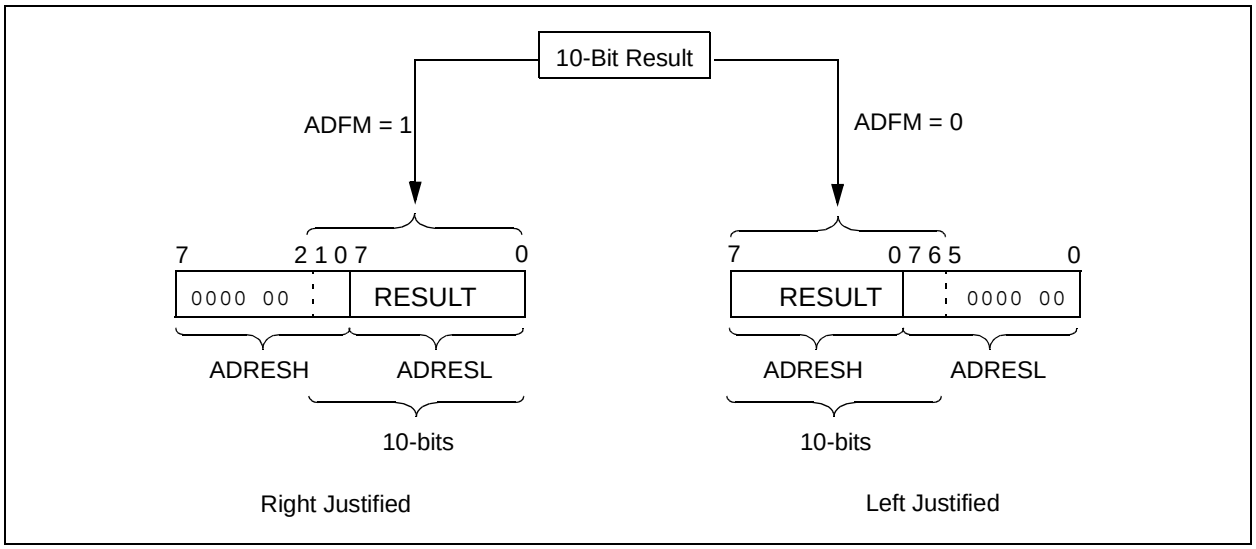
Note:	For the A/D module to operate in SLEEP, the A/D clock source must be set to RC (ADCS1:ADCS0 = 11). To allow the conversion to occur during SLEEP, ensure the SLEEP instruction immediately follows the instruction that sets the GO/DONE bit.

16.6 Effects of a RESET

A device RESET forces all registers to their RESET state. This forces the A/D module to be turned off, and any conversion is aborted.

The value that is in the ADRESH:ADRESL registers is not modified for a Power-on Reset. The ADRESH:ADRESL registers will contain unknown data after a Power-on Reset.

FIGURE 16-6: A/D RESULT JUSTIFICATION



© 1998-2013 Microchip Technology Inc.

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

© 1998-2013 Microchip Technology Inc.

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

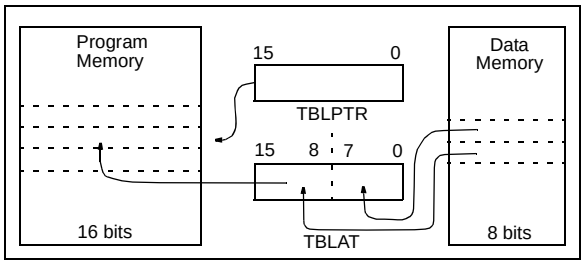
Example 2: `TABLWT 0, 0, REG`

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

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

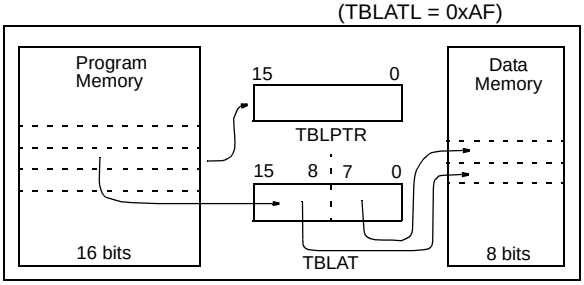
RAM	=	0xAF	
TBLAT	=	0x00AF	(TBLATH = 0x00) (TBLATL = 0xAF)

Before Instruction

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

After Instruction

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



NOTES:

© 1998-2013 Microchip Technology Inc.

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.

© 1998-2013 Microchip Technology Inc.

FIGURE 21-9: TYPICAL IDD vs. FOSC OVER VDD (XT MODE)

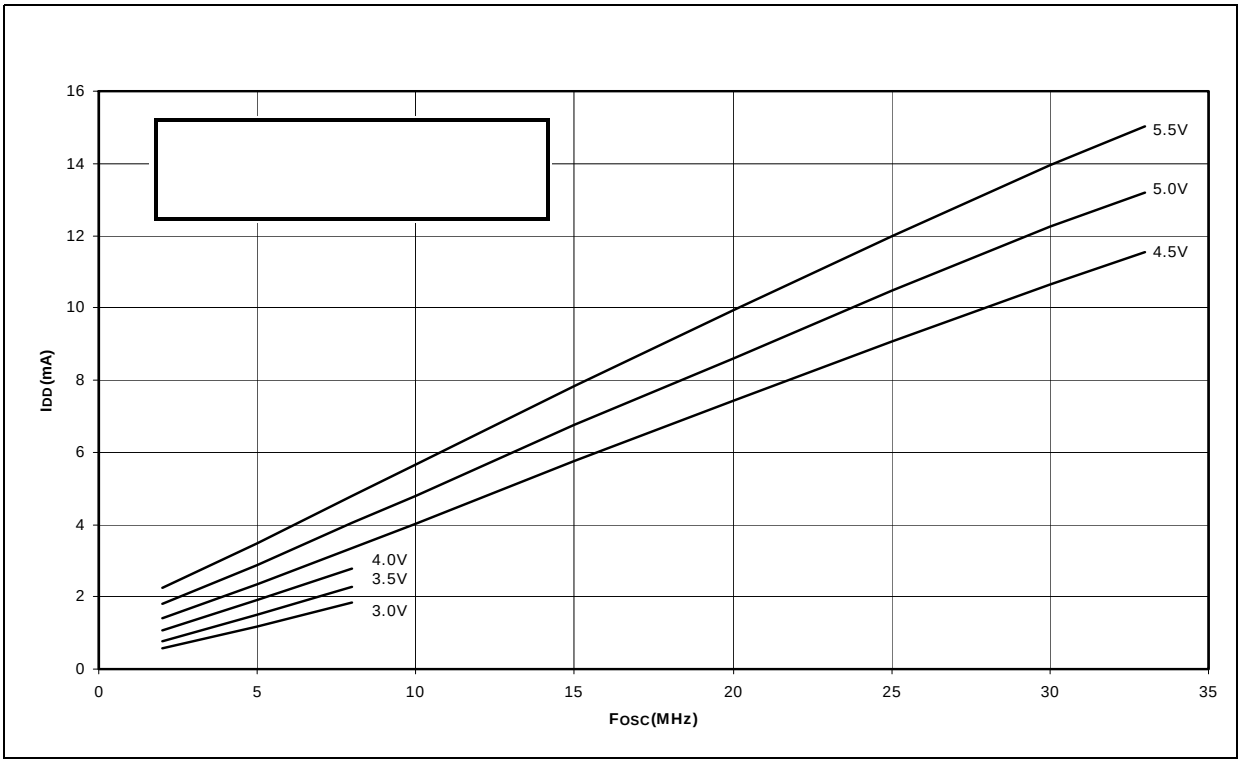
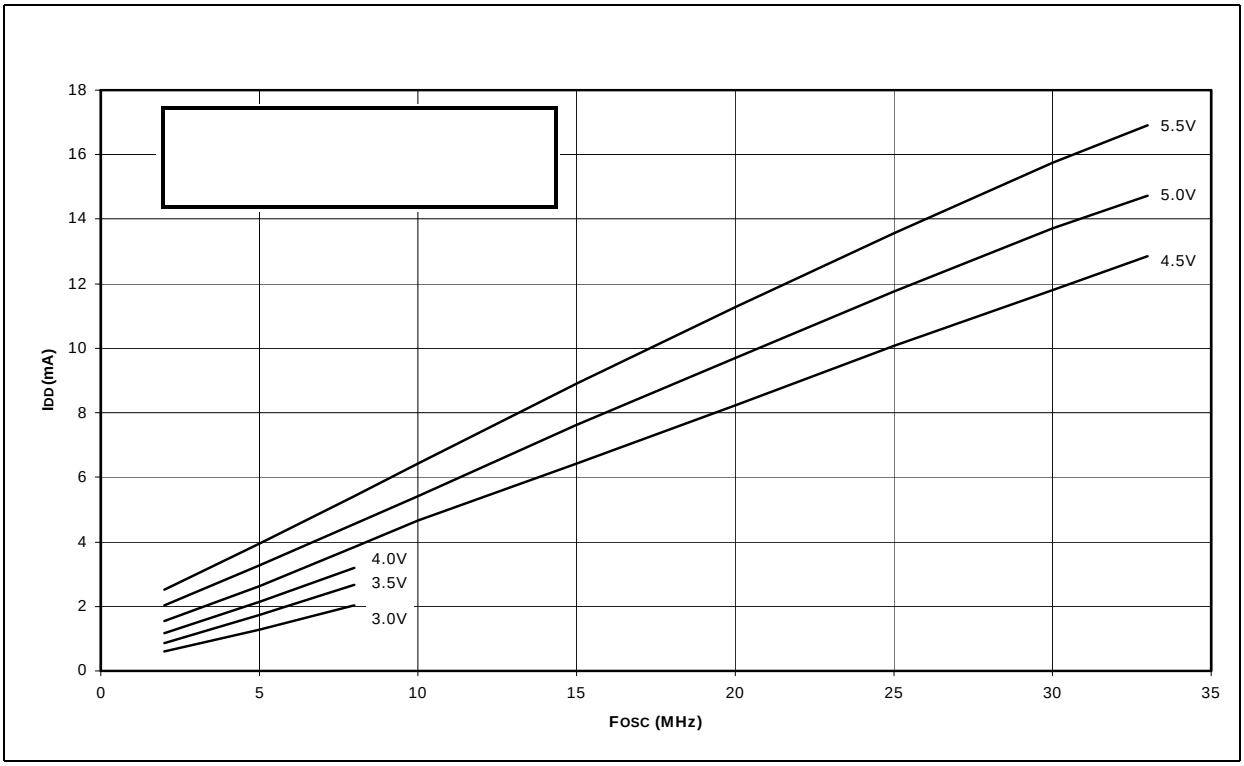
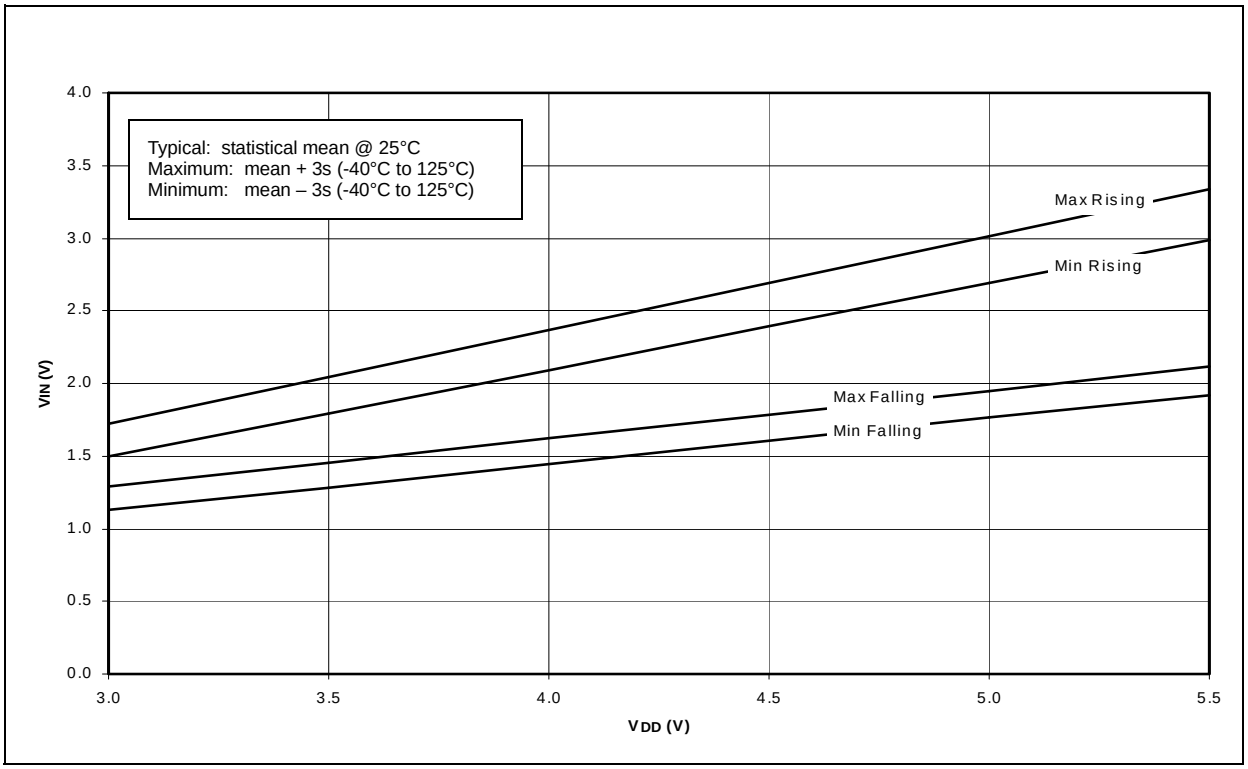


FIGURE 21-10: MAXIMUM IDD vs. FOSC OVER VDD (XT MODE)



© 1998-2013 Microchip Technology Inc.

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

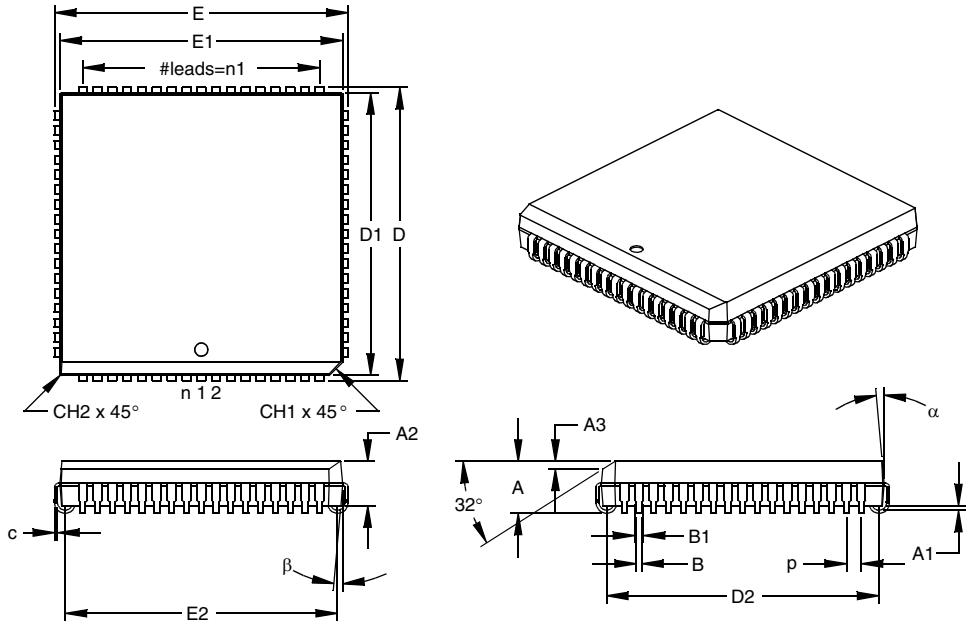


NOTES:

© 1998-2013 Microchip Technology Inc.

84-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		68			68	
Pitch	p		.050			1.27	
Pins per Side	n1		17			17	
Overall Height	A	.165	.173	.180	4.19	4.39	4.57
Molded Package Thickness	A2	.145	.153	.160	3.68	3.87	4.06
Standoff §	A1	.020	.028	.035	0.51	0.71	0.89
Side 1 Chamfer Height	A3	.024	.029	.034	0.61	0.74	0.86
Corner Chamfer 1	CH1	.040	.045	.050	1.02	1.14	1.27
Corner Chamfer (others)	CH2	.000	.005	.010	0.00	0.13	0.25
Overall Width	E	.985	.990	.995	25.02	25.15	25.27
Overall Length	D	.985	.990	.995	25.02	25.15	25.27
Molded Package Width	E1	.950	.954	.958	24.13	24.23	24.33
Molded Package Length	D1	.950	.954	.958	24.13	24.23	24.33
Footprint Width	E2	.890	.920	.930	22.61	23.37	23.62
Footprint Length	D2	.890	.920	.930	22.61	23.37	23.62
Lead Thickness	c	.008	.011	.013	0.20	0.27	0.33
Upper Lead Width	B1	.026	.029	.032	0.66	0.74	0.81
Lower Lead Width	B	.013	.020	.021	0.33	0.51	0.53
Mold Draft Angle Top	α	0	5	10	0	5	10
Mold Draft Angle Bottom	β	0	5	10	0	5	10

* Controlling Parameter
§ Significant Characteristic
Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MO-047
Drawing No. C04-093

© 1998-2013 Microchip Technology Inc.