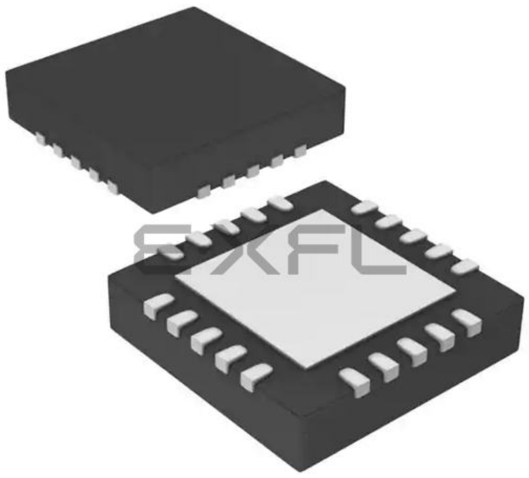


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	48MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	17
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	20-VFQFN Exposed Pad
Supplier Device Package	20-QFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f13k22-e-ml

2.6 Internal Oscillator

The internal oscillator module contains two independent oscillators which are:

- LFINTOSC: Low-Frequency Internal Oscillator
- HFINTOSC: High-Frequency Internal Oscillator

When operating with either oscillator, OSC1 will be an I/O and OSC2 will be either an I/O or CLKOUT. The CLKOUT function is selected by the FOSC bits of the CONFIG1H Configuration register. When OSC2 is configured as CLKOUT, the frequency at the pin is the frequency of the Internal Oscillator divided by 4.

2.6.1 LFINTOSC

The Low-Frequency Internal Oscillator (LFINTOSC) is a 31 kHz internal clock source. The LFINTOSC oscillator is the clock source for:

- Power-up Timer
- Watchdog Timer
- Fail-Safe Clock Monitor

The LFINTOSC is enabled when any of the following conditions are true:

- Power-up Timer is enabled (PWRTEN = 0)
- Watchdog Timer is enabled (WDTEN = 1)
- Watchdog Timer is enabled by software (WDTEN = 0 and SWDTEN = 1)
- Fail-Safe Clock Monitor is enabled (FCMEM = 1)
- SCS1 = 1 and IRCF<2:0> = 000 and INTSRC = 0
- FOSC<3:0> selects the internal oscillator as the primary clock and IRCF<2:0> = 000 and INTSRC = 0
- IESO = 1 (Two-Speed Start-up) and IRCF<2:0> = 000 and INTSRC = 0

2.6.2 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a precision oscillator that is factory-calibrated to operate at 16 MHz. The output of the HFINTOSC connects to a postscaler and a multiplexer (see Figure 2-1). One of eight frequencies can be selected using the IRCF<2:0> bits of the OSCCON register. The following frequencies are available from the HFINTOSC:

- 16 MHZ
- 8 MHZ
- 4 MHZ
- 2 MHZ
- 1 MHZ (Default after Reset)
- 500 kHz
- 250 kHz
- 31 kHz

The HFIOFS bit of the OSCCON register indicates whether the HFINTOSC is stable.

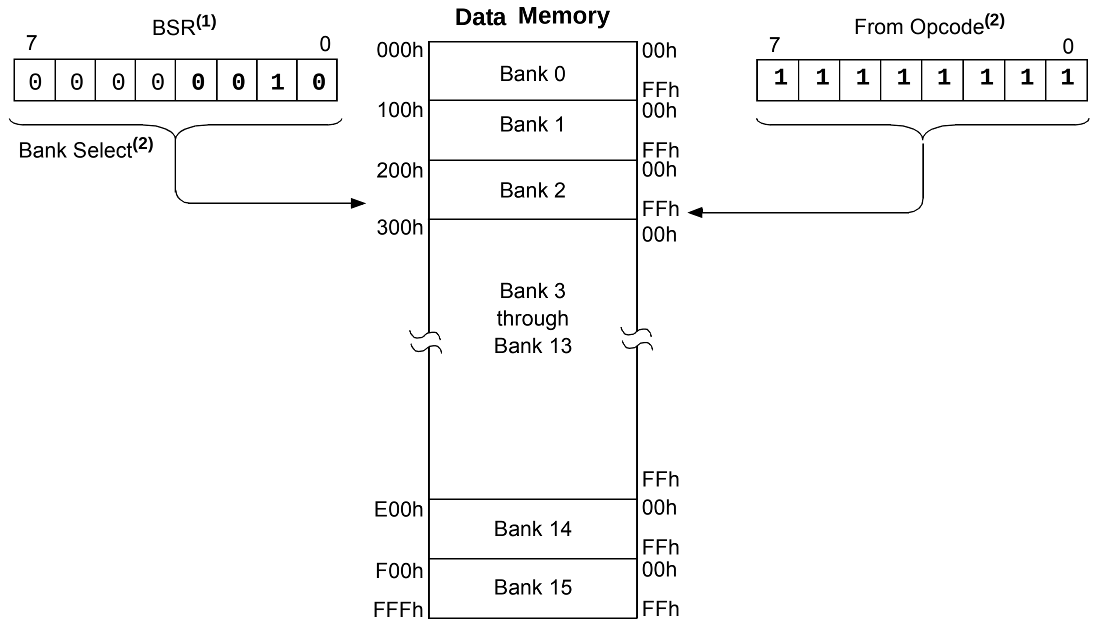
Note 1: Selecting 31 kHz from the HFINTOSC oscillator requires IRCF<2:0> = 000 and the INTSRC bit of the OSCTUNE register to be set. If the INTSRC bit is clear, the system clock will come from the LFINTOSC.

2: Additional adjustments to the frequency of the HFINTOSC can made via the OSCTUNE registers. See Register 2-3 for more details.

The HFINTOSC is enabled if any of the following conditions are true:

- SCS1 = 1 and IRCF<2:0> $\neq$ 000
- SCS1 = 1 and IRCF<2:0> = 000 and INTSRC = 1
- FOSC<3:0> selects the internal oscillator as the primary clock and
  - IRCF<2:0> $\neq$ 000 or
  - IRCF<2:0> = 000 and INTSRC = 1
- IESO = 1 (Two-Speed Start-up) and
  - IRCF<2:0> $\neq$ 000 or
  - IRCF<2:0> = 000 and INTSRC = 1
- FCMEM = 1 (Fail-Safe Clock Monitoring) and
  - IRCF<2:0> $\neq$ 000 or
  - IRCF<2:0> = 000 and INTSRC = 1

FIGURE 3-7: USE OF THE BANK SELECT REGISTER (DIRECT ADDRESSING)



Note 1: The Access RAM bit of the instruction can be used to force an override of the selected bank (BSR<3:0>) to the registers of the Access Bank.

2: The `MOVFF` instruction embeds the entire 12-bit address in the instruction.

4.0 FLASH PROGRAM MEMORY

The Flash program memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed one byte at a time. A write to program memory is executed on blocks of 16 or 8 bytes at a time depending on the specific device (See Table 4-1). Program memory is erased in blocks of 64 bytes at a time. The difference between the write and erase block sizes requires from 4 to 8 block writes to restore the contents of a single block erase. A Bulk Erase operation can not be issued from user code.

TABLE 4-1: WRITE/ERASE BLOCK SIZES

Device	Write Block Size (bytes)	Erase Block Size (bytes)
PIC18(L)F13K22	8	64
PIC18(L)F14K22	16	64

Writing or erasing program memory will cease instruction fetches until the operation is complete. The program memory cannot be accessed during the write or erase, therefore, code cannot execute. An internal programming timer terminates program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a NOP.

4.1 Table Reads and Table Writes

In order to read and write program memory, there are two operations that allow the processor to move bytes between the program memory space and the data RAM:

- Table Read (TBLRD)
- Table Write (TBLWT)

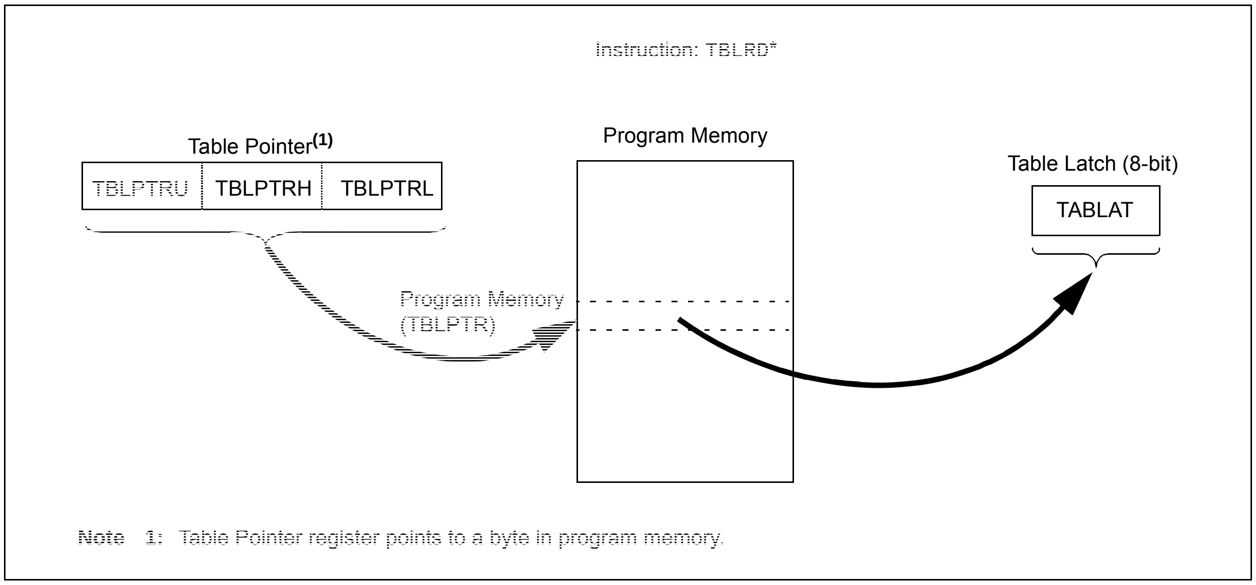
The program memory space is 16-bit wide, while the data RAM space is 8-bit wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

The table read operation retrieves one byte of data directly from program memory and places it into the TABLAT register. Figure 4-1 shows the operation of a table read.

The table write operation stores one byte of data from the TABLAT register into a write block holding register. The procedure to write the contents of the holding registers into program memory is detailed in **Section 4.5 “Writing to Flash Program Memory”**. Figure 4-2 shows the operation of a table write with program memory and data RAM.

Table operations work with byte entities. Tables containing data, rather than program instructions, are not required to be word-aligned. Therefore, a table can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need to be word-aligned.

FIGURE 4-1: TABLE READ OPERATION



4.4 Erasing Flash Program Memory

The minimum erase block is 32 words or 64 bytes. Only through the use of an external programmer, or through ICSP control, can larger blocks of program memory be bulk erased. Word erase in the Flash array is not supported.

When initiating an erase sequence from the Microcontroller itself, a block of 64 bytes of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:6> point to the block being erased. The TBLPTR<5:0> bits are ignored.

The EECON1 register commands the erase operation. The EEPGD bit must be set to point to the Flash program memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation.

The write initiate sequence for EECON2, shown as steps 4 through 6 in **Section 4.4.1 "Flash Program Memory Erase Sequence"**, is used to guard against accidental writes. This is sometimes referred to as a long write.

A long write is necessary for erasing the internal Flash. Instruction execution is halted during the long write cycle. The long write is terminated by the internal programming timer.

4.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

The sequence of events for erasing a block of internal program memory is:

1. Load Table Pointer register with address of block being erased.
2. Set the EECON1 register for the erase operation:
   - set EEPGD bit to point to program memory;
   - clear the CFGS bit to access program memory;
   - set WREN bit to enable writes;
   - set FREE bit to enable the erase.
3. Disable interrupts.
4. Write 55h to EECON2.
5. Write 0AAh to EECON2.
6. Set the WR bit. This will begin the block erase cycle.
7. The CPU will stall for duration of the erase (about 2 ms using internal timer).
8. Re-enable interrupts.

EXAMPLE 4-2: ERASING A FLASH PROGRAM MEMORY BLOCK

```
                 MOVLW   CODE_ADDR_UPPER          ; load TBLPTR with the base
                 MOVWF   TBLPTRU                  ; address of the memory block
                 MOVLW   CODE_ADDR_HIGH
                 MOVWF   TBLPTRH
                 MOVLW   CODE_ADDR_LOW
                 MOVWF   TBLPTRL
        ERASE_BLOCK
                 BSF     EECON1, EEPGD            ; point to Flash program memory
                 BCF     EECON1, CFGS             ; access Flash program memory
                 BSF     EECON1, WREN             ; enable write to memory
                 BSF     EECON1, FREE             ; enable block Erase operation
                 BCF     INTCON, GIE              ; disable interrupts
    Required     MOVLW   55h
    Sequence     MOVWF   EECON2                   ; write 55h
                 MOVLW   0AAh
                 MOVWF   EECON2                   ; write 0AAh
                 BSF     EECON1, WR               ; start erase (CPU stall)
                 BSF     INTCON, GIE              ; re-enable interrupts
```

4.5 Writing to Flash Program Memory

The programming block size is 8 or 16 bytes, depending on the device (See Table 4-1). Word or byte programming is not supported.

Table writes are used internally to load the holding registers needed to program the Flash memory. There are only as many holding registers as there are bytes in a write block (See Table 4-1).

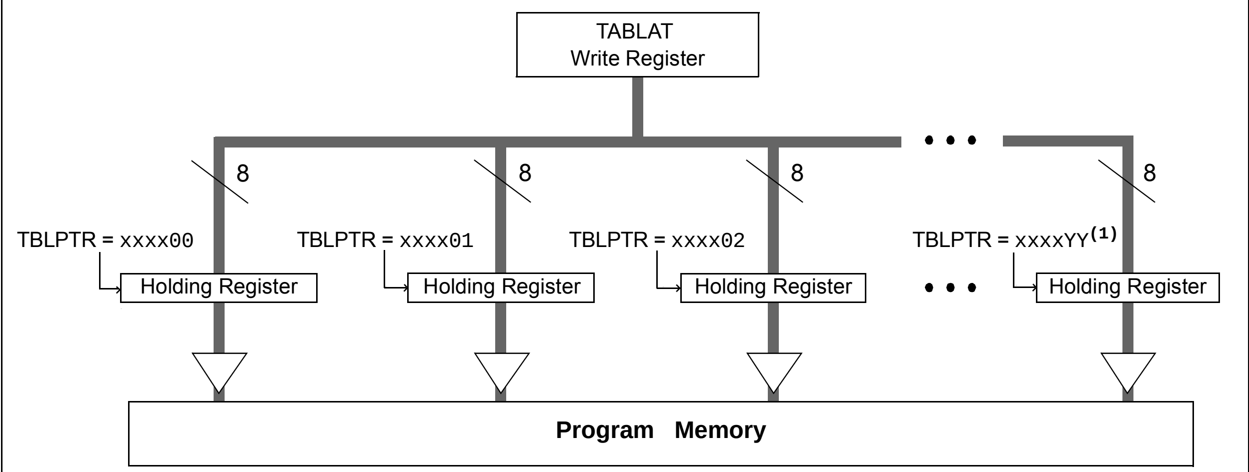
Since the Table Latch (TABLAT) is only a single byte, the `TBLWT` instruction may need to be executed 8, or 16 times, depending on the device, for each programming operation. All of the table write operations will essentially be short writes because only the holding registers are written. After all the holding registers have been written, the programming operation of that block of memory is started by configuring the EECON1 register for a program memory write and performing the long write sequence.

The long write is necessary for programming the internal Flash. Instruction execution is halted during a long write cycle. The long write will be terminated by the internal programming timer.

The EEPROM on-chip timer controls the write time. The write/erase voltages are generated by an on-chip charge pump, rated to operate over the voltage range of the device.

Note: The default value of the holding registers on device Resets and after write operations is FFh. A write of FFh to a holding register does not modify that byte. This means that individual bytes of program memory may be modified, provided that the change does not attempt to change any bit from a '0' to a '1'. When modifying individual bytes, it is not necessary to load all holding registers before executing a long write operation.

FIGURE 4-5: TABLE WRITES TO FLASH PROGRAM MEMORY



Note 1: YY = x7 or xF for 8 or 16 byte write blocks, respectively.

4.5.1 FLASH PROGRAM MEMORY WRITE SEQUENCE

The sequence of events for programming an internal program memory location should be:

1. Read 64 bytes into RAM.
2. Update data values in RAM as necessary.
3. Load Table Pointer register with address being erased.
4. Execute the block erase procedure.
5. Load Table Pointer register with address of first byte being written.
6. Write the 8 or 16 byte block into the holding registers with auto-increment.
7. Set the EECON1 register for the write operation:
   - set EEPGD bit to point to program memory;
   - clear the CFGS bit to access program memory;
   - set WREN to enable byte writes.
8. Disable interrupts.
9. Write 55h to EECON2.
10. Write 0AAh to EECON2.
11. Set the WR bit. This will begin the write cycle.
12. The CPU will stall for duration of the write (about 2 ms using internal timer).
13. Re-enable interrupts.
14. Repeat steps 6 to 13 for each block until all 64 bytes are written.
15. Verify the memory (table read).

This procedure will require about 6 ms to update each write block of memory. An example of the required code is given in Example 4-3.

Note: Before setting the WR bit, the Table Pointer address needs to be within the intended address range of the bytes in the holding registers.

Example 6-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 6-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES<3:0>).

EQUATION 6-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

$$\begin{aligned}
\text{RES3:RES0} &= \text{ARG1H:ARG1L} \bullet \text{ARG2H:ARG2L} \\
&= (\text{ARG1H} \bullet \text{ARG2H} \bullet 2^{16}) + \\
&\quad (\text{ARG1H} \bullet \text{ARG2L} \bullet 2^{8}) + \\
&\quad (\text{ARG1L} \bullet \text{ARG2H} \bullet 2^{8}) + \\
&\quad (\text{ARG1L} \bullet \text{ARG2L})
\end{aligned}$$

EXAMPLE 6-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE

```
    MOVF    ARG1L, W
    MULWF   ARG2L          ; ARG1L * ARG2L->
                           ; PRODH:PRODL
    MOVFF   PRODH, RES1    ;
    MOVFF   PRODL, RES0    ;
;
    MOVF    ARG1H, W
    MULWF   ARG2H          ; ARG1H * ARG2H->
                           ; PRODH:PRODL
    MOVFF   PRODH, RES3    ;
    MOVFF   PRODL, RES2    ;
;
    MOVF    ARG1L, W
    MULWF   ARG2H          ; ARG1L * ARG2H->
                           ; PRODH:PRODL
    MOVF    PRODL, W       ;
    ADDWF   RES1, F        ; Add cross
    MOVF    PRODH, W       ; products
    ADDWFC  RES2, F        ;
    CLRF    WREG           ;
    ADDWFC  RES3, F        ;
;
    MOVF    ARG1H, W       ;
    MULWF   ARG2L          ; ARG1H * ARG2L->
                           ; PRODH:PRODL
    MOVF    PRODL, W       ;
    ADDWF   RES1, F        ; Add cross
    MOVF    PRODH, W       ; products
    ADDWFC  RES2, F        ;
    CLRF    WREG           ;
    ADDWFC  RES3, F        ;
```

Example 6-4 shows the sequence to do a 16 x 16 signed multiply. Equation 6-2 shows the algorithm used. The 32-bit result is stored in four registers (RES<3:0>). To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

EQUATION 6-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

$$\begin{aligned}
\text{RES3:RES0} &= \text{ARG1H:ARG1L} \bullet \text{ARG2H:ARG2L} \\
&= (\text{ARG1H} \bullet \text{ARG2H} \bullet 2^{16}) + \\
&\quad (\text{ARG1H} \bullet \text{ARG2L} \bullet 2^{8}) + \\
&\quad (\text{ARG1L} \bullet \text{ARG2H} \bullet 2^{8}) + \\
&\quad (\text{ARG1L} \bullet \text{ARG2L}) + \\
&\quad (-1 \bullet \text{ARG2H<7>} \bullet \text{ARG1H:ARG1L} \bullet 2^{16}) + \\
&\quad (-1 \bullet \text{ARG1H<7>} \bullet \text{ARG2H:ARG2L} \bullet 2^{16})
\end{aligned}$$

EXAMPLE 6-4: 16 x 16 SIGNED MULTIPLY ROUTINE

```
    MOVF    ARG1L, W
    MULWF   ARG2L          ; ARG1L * ARG2L ->
                           ; PRODH:PRODL
    MOVFF   PRODH, RES1    ;
    MOVFF   PRODL, RES0    ;
;
    MOVF    ARG1H, W
    MULWF   ARG2H          ; ARG1H * ARG2H ->
                           ; PRODH:PRODL
    MOVFF   PRODH, RES3    ;
    MOVFF   PRODL, RES2    ;
;
    MOVF    ARG1L, W
    MULWF   ARG2H          ; ARG1L * ARG2H ->
                           ; PRODH:PRODL
    MOVF    PRODL, W       ;
    ADDWF   RES1, F        ; Add cross
    MOVF    PRODH, W       ; products
    ADDWFC  RES2, F        ;
    CLRF    WREG           ;
    ADDWFC  RES3, F        ;
;
    MOVF    ARG1H, W       ;
    MULWF   ARG2L          ; ARG1H * ARG2L ->
                           ; PRODH:PRODL
    MOVF    PRODL, W       ;
    ADDWF   RES1, F        ; Add cross
    MOVF    PRODH, W       ; products
    ADDWFC  RES2, F        ;
    CLRF    WREG           ;
    ADDWFC  RES3, F        ;
;
    BTFSS   ARG2H, 7       ; ARG2H:ARG2L neg?
    BRA     SIGN_ARG1      ; no, check ARG1
    MOVF    ARG1L, W       ;
    SUBWF   RES2           ;
    MOVF    ARG1H, W       ;
    SUBWFB  RES3
;
SIGN_ARG1
    BTFSS   ARG1H, 7       ; ARG1H:ARG1L neg?
    BRA     CONT_CODE      ; no, done
    MOVF    ARG2L, W       ;
    SUBWF   RES2           ;
    MOVF    ARG2H, W       ;
    SUBWFB  RES3
;
CONT_CODE
    :
```

7.6 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Enable registers (PIE1 and PIE2). When IPEN = `0`, the PEIE bit must be set to enable any of these peripheral interrupts.

REGISTER 7-6: PIE1: PERIPHERAL INTERRUPT ENABLE (FLAG) REGISTER 1

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7	**Unimplemented:** Read as ‘`0`’
bit 6	**ADIE:** A/D Converter Interrupt Enable bit `1` = Enables the A/D interrupt `0` = Disables the A/D interrupt
bit 5	**RCIE:** EUSART Receive Interrupt Enable bit `1` = Enables the EUSART receive interrupt `0` = Disables the EUSART receive interrupt
bit 4	**TXIE:** EUSART Transmit Interrupt Enable bit `1` = Enables the EUSART transmit interrupt `0` = Disables the EUSART transmit interrupt
bit 3	**SSPIE:** Master Synchronous Serial Port Interrupt Enable bit `1` = Enables the MSSP interrupt `0` = Disables the MSSP interrupt
bit 2	**CCP1IE:** CCP1 Interrupt Enable bit `1` = Enables the CCP1 interrupt `0` = Disables the CCP1 interrupt
bit 1	**TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit `1` = Enables the TMR2 to PR2 match interrupt `0` = Disables the TMR2 to PR2 match interrupt
bit 0	**TMR1IE:** TMR1 Overflow Interrupt Enable bit `1` = Enables the TMR1 overflow interrupt `0` = Disables the TMR1 overflow interrupt

TABLE 8-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	247
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	247
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	246
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0	246
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	244
INTCON2	$\overline{\text{RABPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RABIP	244
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	244
LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	247
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	247
PSTRCON	—	—	—	STRSYNC	STRD	STRC	STRB	STRA	246
VREFCON1	D1EN	D1LPS	DAC1OE	---	D1PSS1	D1PSS0	---	D1NSS	246
SLRCON	—	—	—	—	—	SLRC	SLRB	SLRA	247
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	245
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	247
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	245
T3CON	RD16	—	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	246

14.3.4.5 Clock Synchronization and the CKP bit

When the CKP bit is cleared, the SCL output is forced to ‘0’. However, clearing the CKP bit will not assert the SCL output low until the SCL output is already sampled low. Therefore, the CKP bit will not assert the SCL line until an external I^{2}C master device has already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the I^{2}C bus have deasserted SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 14-12).

FIGURE 14-12: CLOCK SYNCHRONIZATION TIMING

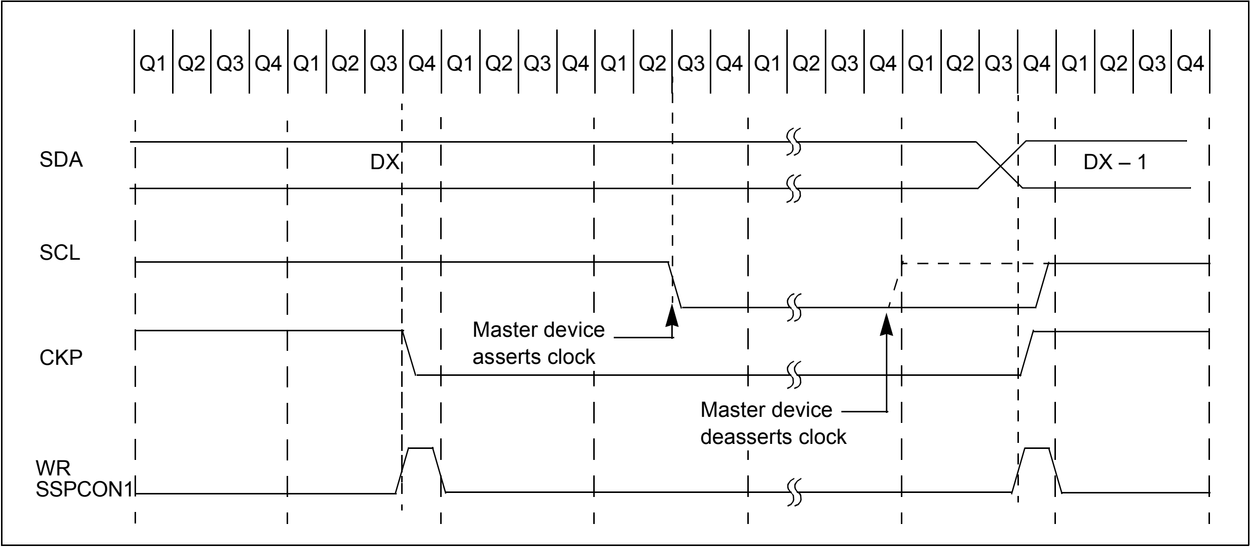
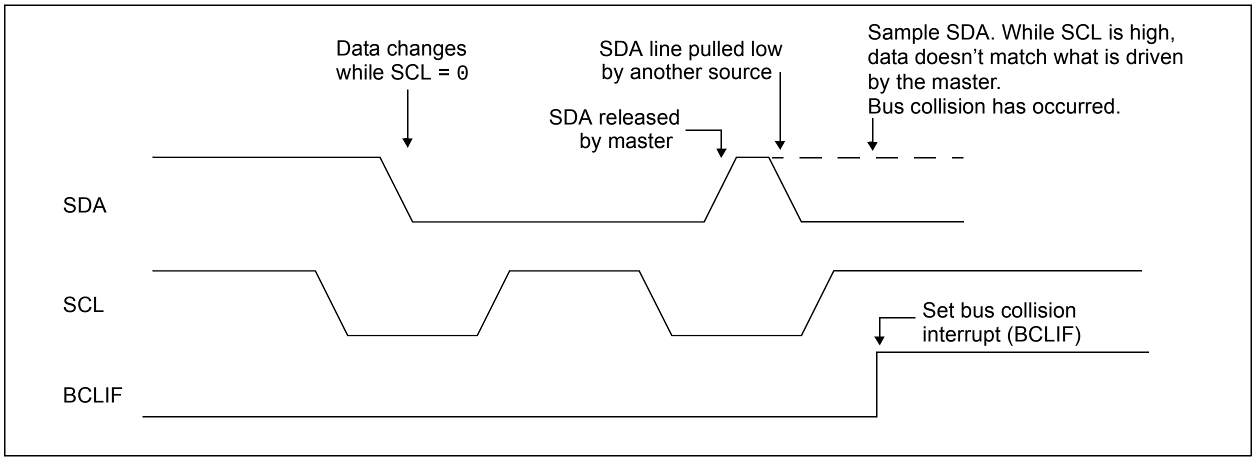

Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
SDA
DX
DX – 1
SCL
CKP
Master device asserts clock
Master device deasserts clock
WR SSPCON1

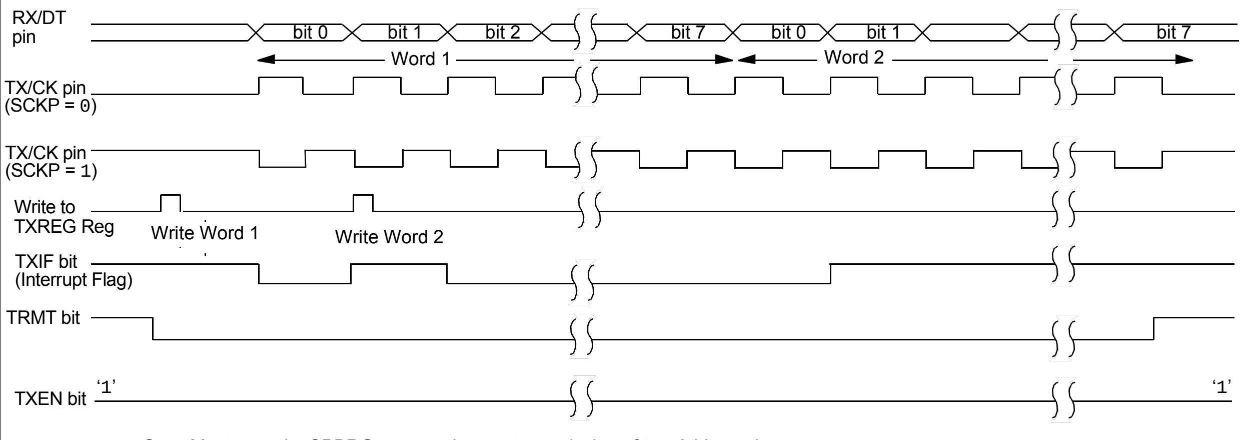
FIGURE 14-25: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE


Data changes while SCL = 0
SDA line pulled low by another source
Sample SDA. While SCL is high, data doesn't match what is driven by the master. Bus collision has occurred.
SDA released by master
SDA
SCL
Set bus collision interrupt (BCLIF)
BCLIF

15.4.1.5 Synchronous Master Transmission Set-up

1. Initialize the SPBRGH, SPBRG register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 15.3 "EUSART Baud Rate Generator (BRG)"**).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC. Set the TRIS bits corresponding to the RX/DT and TX/CK I/O pins.
3. Disable Receive mode by clearing bits SREN and CREN.
4. Enable Transmit mode by setting the TXEN bit.
5. If 9-bit transmission is desired, set the TX9 bit.
6. If interrupts are desired, set the TXIE, GIE and PEIE interrupt enable bits.
7. If 9-bit transmission is selected, the ninth bit should be loaded in the TX9D bit.
8. Start transmission by loading data to the TXREG register.

FIGURE 15-10: SYNCHRONOUS TRANSMISSION



Note: Sync Master mode, SPBRG = 0, continuous transmission of two 8-bit words.

FIGURE 15-11: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)

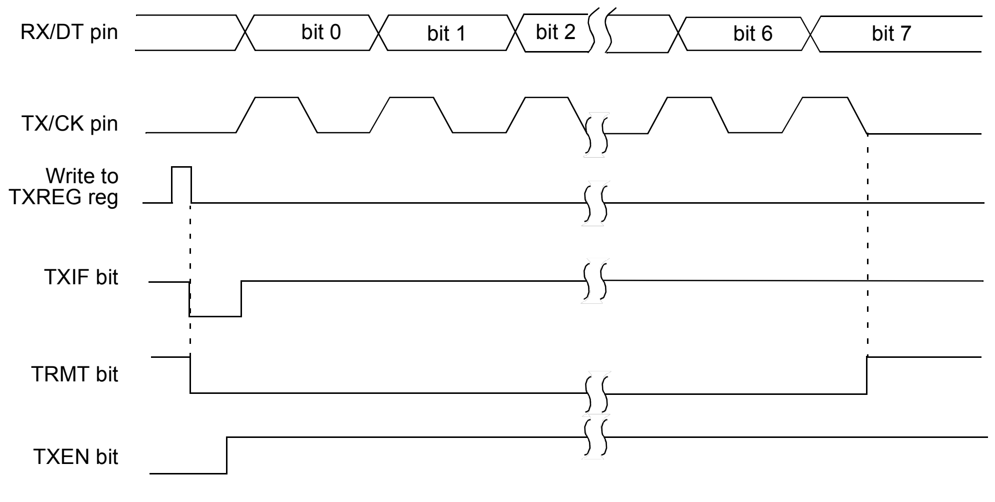


TABLE 15-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	247
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	245
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	248
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	248
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	248
RCREG	EUSART Receive Register								247
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	247
SPBRG	EUSART Baud Rate Generator Register, Low Byte								247
SPBRGH	EUSART Baud Rate Generator Register, High Byte								247
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	247

Legend: — = unimplemented, read as '0'. Shaded cells are not used for synchronous master reception.

17.0 COMPARATOR MODULE

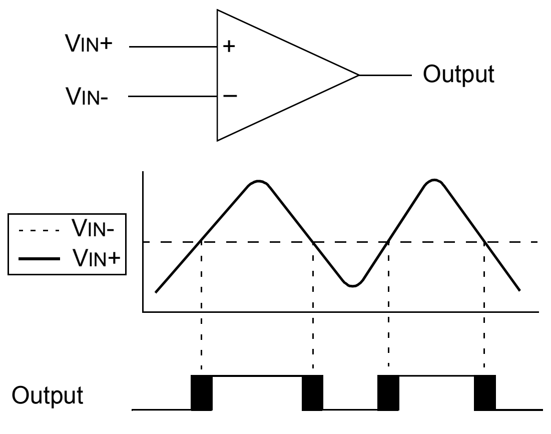
Comparators are used to interface analog circuits to a digital circuit by comparing two analog voltages and providing a digital indication of their relative magnitudes. The comparators are very useful mixed signal building blocks because they provide analog functionality independent of the program execution. The Analog Comparator module includes the following features:

- Independent comparator control
- Programmable input selection
- Comparator output is available internally/externally
- Programmable output polarity
- Interrupt-on-Change
- Wake-up from Sleep
- Programmable Speed/Power optimization
- PWM shutdown
- Programmable and fixed voltage reference

17.1 Comparator Overview

A single comparator is shown in Figure 17-1 along with the relationship between the analog input levels and the digital output. When the analog voltage at VIN+ is less than the analog voltage at VIN-, the output of the comparator is a digital low level. When the analog voltage at VIN+ is greater than the analog voltage at VIN-, the output of the comparator is a digital high level.

FIGURE 17-1: SINGLE COMPARATOR



Note: The black areas of the output of the comparator represents the uncertainty due to input offsets and response time.

17.5 Operation During Sleep

The comparator, if enabled before entering Sleep mode, remains active during Sleep. The additional current consumed by the comparator is shown separately in **Section 26.0 "Electrical Specifications"**. If the comparator is not used to wake the device, power consumption can be minimized while in Sleep mode by turning off the comparator. Each comparator is turned off by clearing the CxON bit of the CMxCON0 register.

A change to the comparator output can wake-up the device from Sleep. To enable the comparator to wake the device from Sleep, the CxIE bit of the PIE2 register and the PEIE bit of the INTCON register must be set. The instruction following the `SLEEP` instruction always executes following a wake from Sleep. If the GIE bit of the INTCON register is also set, the device will then execute the Interrupt Service Routine.

17.6 Effects of a Reset

A device Reset forces the CMxCON0 and CM2CON1 registers to their Reset states. This forces both comparators and the voltage references to their Off states.

20.3 Register Definitions: FVR Control

REGISTER 20-1: VREFCON0: FIXED VOLTAGE REFERENCE CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-1	U-0	U-0	U-0	U-0
FVR1EN	FVR1ST	FVR1S<1:0>		—	—	—	—
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7 **FVR1EN:** Fixed Voltage Reference Enable bit
0 = Fixed Voltage Reference is disabled
1 = Fixed Voltage Reference is enabled

bit 6 **FVR1ST:** Fixed Voltage Reference Ready Flag bit
0 = Fixed Voltage Reference output is not ready or not enabled
1 = Fixed Voltage Reference output is ready for use

bit 5-4 **FVR1S<1:0>:** Fixed Voltage Reference Selection bits
00 = Fixed Voltage Reference Peripheral output is off
01 = Fixed Voltage Reference Peripheral output is 1x (1.024V)
10 = Fixed Voltage Reference Peripheral output is 2x (2.048V)(1)
11 = Fixed Voltage Reference Peripheral output is 4x (4.096V)(1)

bit 3-2 **Reserved:** Read as ‘0’. Maintain these bits clear.

bit 1-0 **Unimplemented:** Read as ‘0’.

Note 1: Fixed Voltage Reference output cannot exceed VDD.

TABLE 20-1: SUMMARY OF REGISTERS ASSOCIATED WITH FIXED VOLTAGE REFERENCE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
VREFCON0	FVR1EN	FVR1ST	FVR1S<1:0>		—	—	—	—	232

Legend: — = unimplemented locations, read as ‘0’. Shaded bits are not used by the FVR module.

POP — Pop Top of Return Stack

Syntax:	POP
Operands:	None
Operation:	(TOS) $\rightarrow$ bit bucket
Status Affected:	None
Encoding:	`0000` `0000` `0000` `0110`
Description:	The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	POP TOS value	No operation

Example:
```
POP
GOTO      NEW
```

Before Instruction

TOS	=	0031A2h
Stack (1 level down)	=	014332h

After Instruction

TOS	=	014332h
PC	=	NEW

PUSH — Push Top of Return Stack

Syntax:	PUSH
Operands:	None
Operation:	(PC + 2) $\rightarrow$ TOS
Status Affected:	None
Encoding:	`0000` `0000` `0000` `0101`
Description:	The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	PUSH PC + 2 onto return stack	No operation	No operation

Example:
```
PUSH
```

Before Instruction

TOS	=	345Ah
PC	=	0124h

After Instruction

PC	=	0126h
TOS	=	0126h
Stack (1 level down)	=	345Ah

RETURN — Return from Subroutine

Syntax:	RETURN {s}
Operands:	s ∈ [0,1]
Operation:	(TOS) → PC, if s = 1 (WS) → W, (STATUSS) → Status, (BSRS) → BSR, PCLATU, PCLATH are unchanged
Status Affected:	None
Encoding:	`0000` `0000` `0001` `001s`
Description:	Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If 's'= `1`, the contents of the shadow registers, WS, STATUSS and BSRS, are loaded into their corresponding registers, W, Status and BSR. If 's' = `0`, no update of these registers occurs (default).
Words:	1
Cycles:	2

Q Cycle Activity:

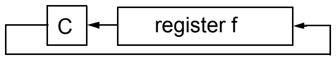
Q1	Q2	Q3	Q4
Decode	No operation	Process Data	POP PC from stack
No operation	No operation	No operation	No operation

<u>Example</u>: `RETURN`

After Instruction:
PC = TOS

RLCF — Rotate Left f through Carry

Syntax:	RLCF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f<n>) → dest<n + 1>, (f<7>) → C, (C) → dest<0>
Status Affected:	C, N, Z
Encoding:	`0011` `01da` `ffff` `ffff`
Description:	The contents of register 'f' are rotated one bit to the left through the CARRY flag. If 'd' is '`0`', the result is placed in W. If 'd' is '`1`', the result is stored back in register 'f' (default). If 'a' is '`0`', the Access Bank is selected. If 'a' is '`1`', the BSR is used to select the GPR bank (default). If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



Words:	1
Cycles:	1

Q Cycle Activity:

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

<u>Example</u>: `RLCF REG, 0, 0`

```
Before Instruction
    REG   =   1110 0110
    C     =   0
After Instruction
    REG   =   1110 0110
    W     =   1100 1100
    C     =   1
```

TABLE 26-4: PRIMARY RUN SUPPLY CURRENT

PIC18LF1XK22		Standard Operating Conditions (unless otherwise stated)					
PIC18F1XK22		**Standard Operating Conditions (unless otherwise stated)**					
Param. No.	**Device Characteristics**	**Typ.**	**Max.**	**Units**	**Conditions**		
D014	**Supply Current (IDD)**[1, 2, 4, 5]	0.20	0.32	mA	-40°C to +125°C	VDD = 1.8V	FOSC = 1 MHz (**PRI_RUN**, EC Med Osc)
D014A		0.27	0.39	mA	-40°C to +125°C	VDD = 3.0V	
D014		.20	.32	mA	-40°C to +125°C	VDD = 2.3V	FOSC = 1 MHz (**PRI_RUN**, EC Med Osc)
D014A		.27	.39	mA	-40°C to +125°C	VDD = 3.0V	
D014B		.30	.42	mA	-40°C to +125°C	VDD = 5.0V	
D015		1.7	2.6	mA	-40°C to +125°C	VDD = 1.8V	FOSC = 16 MHz (**PRI_RUN**, EC High Osc)
D015A		3.0	4.2	mA	-40°C to +125°C	VDD = 3.0V	
D015		2.4	3.2	mA	-40°C to +125°C	VDD = 2.3V	FOSC = 16 MHz (**PRI_RUN**, EC High Osc)
D015A		3.0	4.2	mA	-40°C to +125°C	VDD = 3.0V	
D015B		3.3	4.4	mA	-40°C to +125°C	VDD = 5.0V	
D016		11.5	14.0	mA	-40°C to +125°C	VDD = 3.0V	FOSC = 64 MHz (**PRI_RUN**, EC High Osc)
D016		11.9	14.4	mA	-40°C to +125°C	VDD = 2.3V	FOSC = 64 MHz (**PRI_RUN**, EC High Osc)
D016A		12.1	14.6	mA	-40°C to +125°C	VDD = 5.0V	
D017		2.1	2.9	mA	-40°C to +125°C	VDD = 1.8V	FOSC = 4 MHz 16 MHz Internal (**PRI_RUN HS+PLL**)
D017A		3.1	4.0	mA	-40°C to +125°C	VDD = 3.0V	
D017		2.1	2.9	mA	-40°C to +125°C	VDD = 2.3V	FOSC = 4 MHz 16 MHz Internal (**PRI_RUN HS+PLL**)
D017A		3.1	4.0	mA	-40°C to +125°C	VDD = 3.0V	
D017B		3.3	4.5	mA	-40°C to +125°C	VDD = 5.0V	
D018		10	15	mA	-40°C to +125°C	VDD = 3.0V	FOSC = 16 MHz 64 MHz Internal (**PRI_RUN HS+PLL**)
D018		12.4	15.4	mA	-40°C to +125°C	VDD = 3.0V	FOSC = 16 MHz 64 MHz Internal (**PRI_RUN HS+PLL**)
D018A		12.6	15.6	mA	-40°C to +125°C	VDD = 5.0V	

* These parameters are characterized but not tested.

Note 1: The test conditions for all IDD measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD; $\overline{\text{MCLR}}$ = VDD; WDT disabled.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.

3: For RC oscillator configurations, current through REXT is not included. The current through the resistor can be extended by the formula IR = VDD/2REXT (mA) with REXT in kΩ.

4: FVR and BOR are disabled.

5: When a single temperature range is provided for a parameter, the specification applies to both industrial and extended temperature devices.

FIGURE 27-19: MEMLOW TYPICAL PRI_RUN I$_{DD}$ (EC)

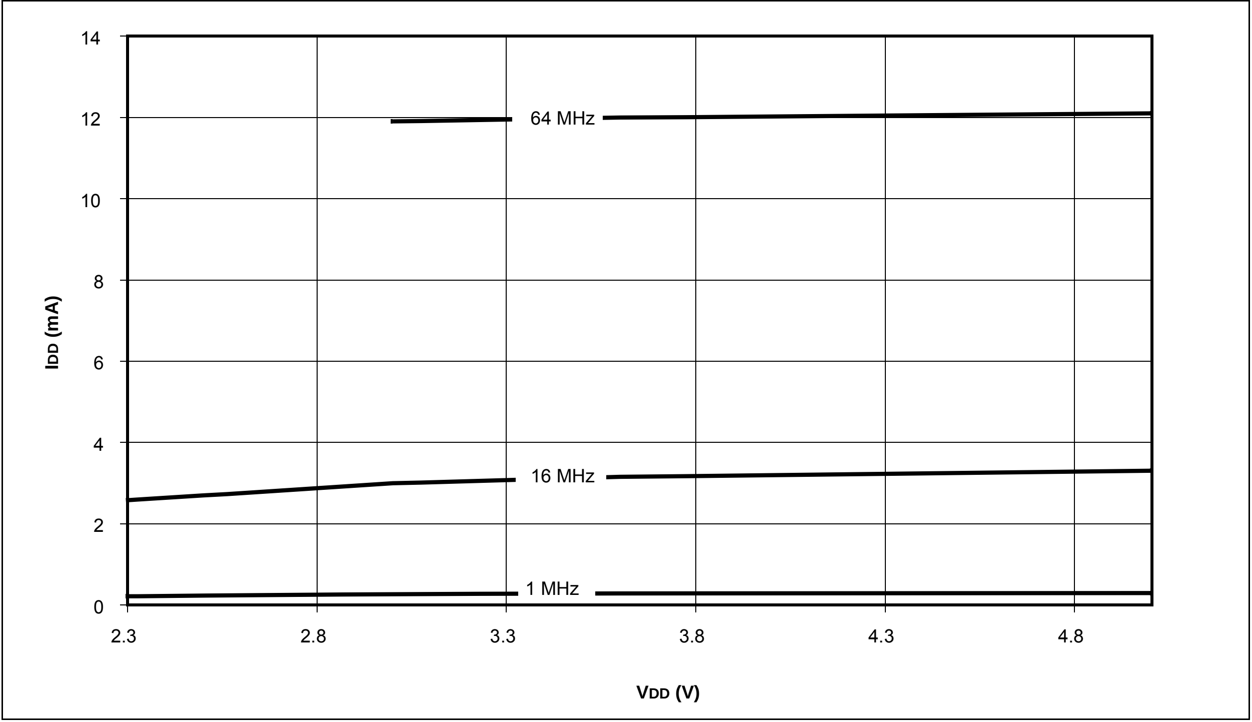
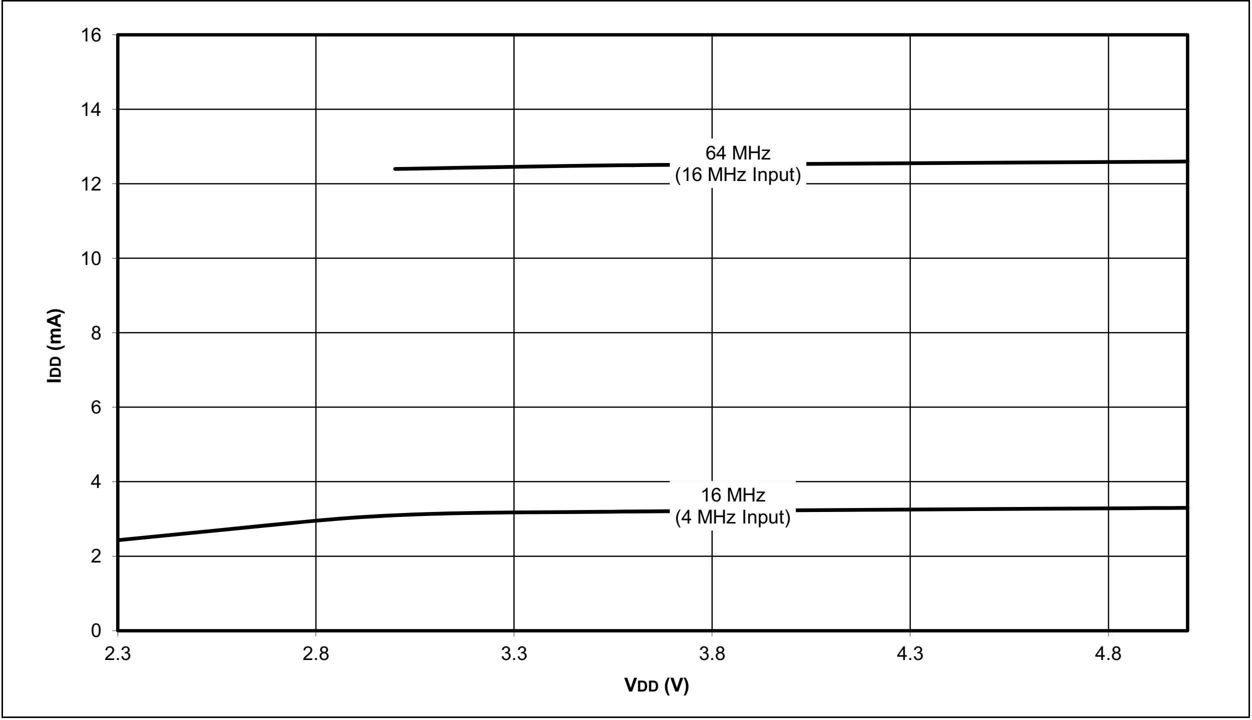
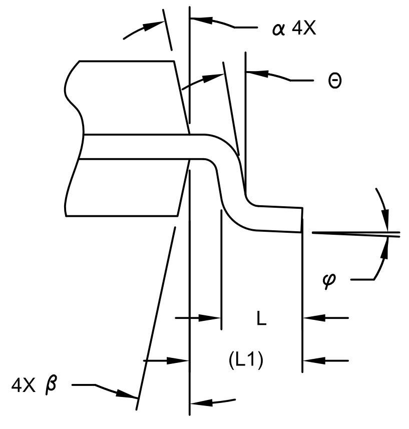


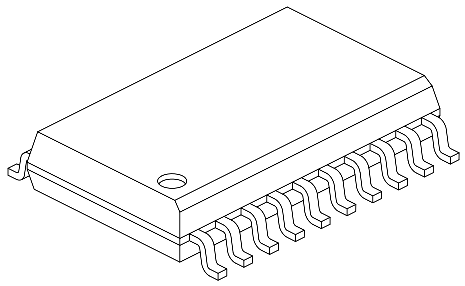
FIGURE 27-20: MEMLOW TYPICAL PRI_RUN I$_{DD}$ (HS + PLL)



20-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

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





VIEW C

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N	20		
Pitch	e	1.27 BSC		
Overall Height	A	-	-	2.65
Molded Package Thickness	A2	2.05	-	-
Standoff §	A1	0.10	-	0.30
Overall Width	E	10.30 BSC		
Molded Package Width	E1	7.50 BSC		
Overall Length	D	12.80 BSC		
Chamfer (Optional)	h	0.25	-	0.75
Foot Length	L	0.40	-	1.27
Footprint	L1	1.40 REF		
Lead Angle	Θ	0°	-	-
Foot Angle	φ	0°	-	8°
Lead Thickness	c	0.20	-	0.33
Lead Width	b	0.31	-	0.51
Mold Draft Angle Top	α	5°	-	15°
Mold Draft Angle Bottom	β	5°	-	15°

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing No. C04-094C Sheet 2 of 2