**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 | 16-Bit |
| Speed | 32MHz |
| Connectivity | I²C, IrDA, LINbus, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, HLVD, POR, PWM, WDT |
| Number of I/O | 18 |
| Program Memory Size | 8KB (2.75K x 24) |
| Program Memory Type | FLASH |
| EEPROM Size | 512 x 8 |
| RAM Size | 1K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 3.6V |
| Data Converters | A/D 12x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 20-SOIC (0.295", 7.50mm Width) |
| Supplier Device Package | 20-SOIC |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic24f08kl401t-i-so |

## 4.0 MEMORY ORGANIZATION

As Harvard architecture devices, the PIC24F microcontrollers feature separate program and data memory space and bussing. This architecture also allows the direct access of program memory from the data space during code execution.

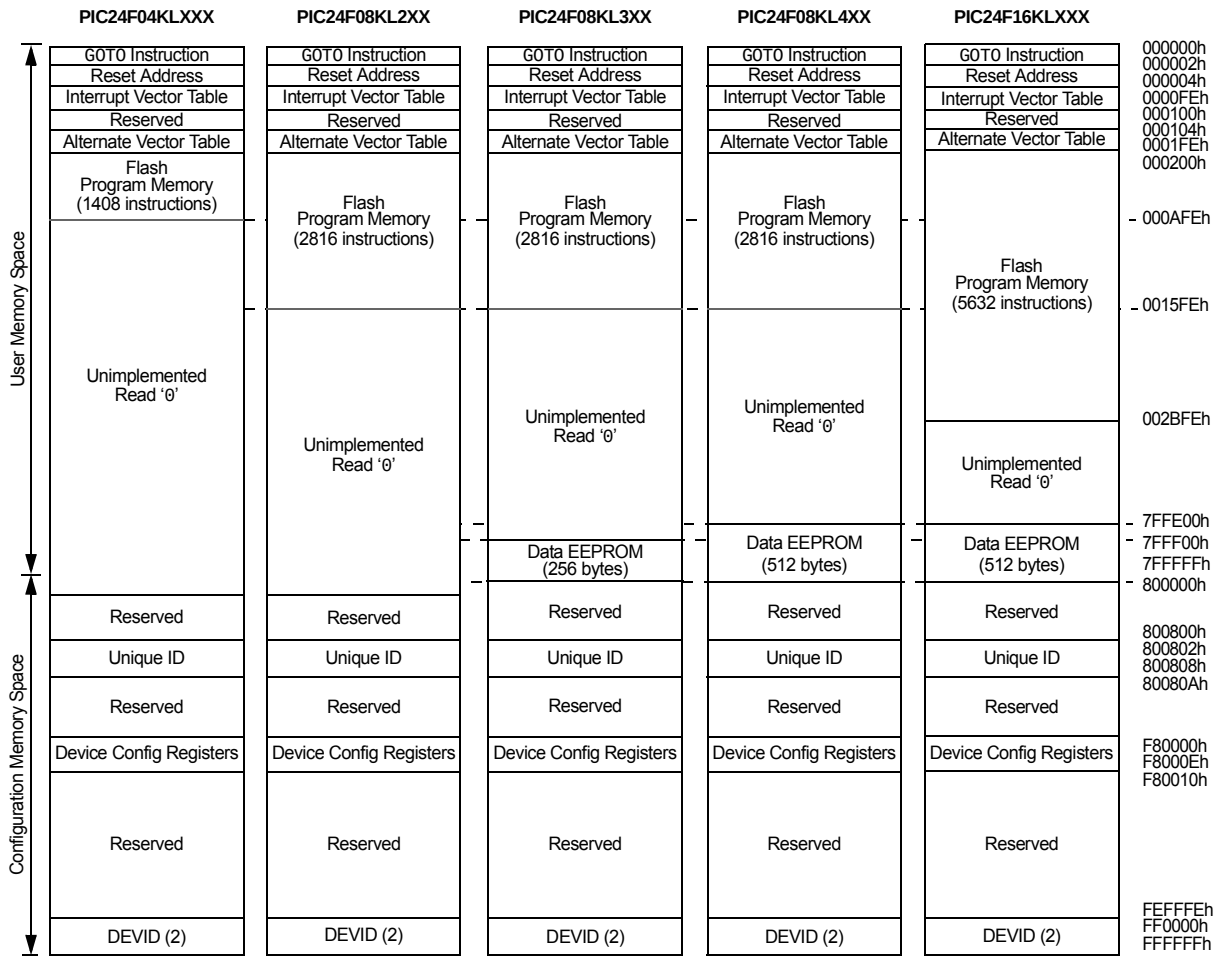
### 4.1 Program Address Space

The program address memory space of the PIC24F16KL402 family is 4M instructions. The space is addressable by a 24-bit value derived from either the 23-bit Program Counter (PC) during program execution, or from a table operation or data space remapping, as described in **Section 4.3 "Interfacing Program and Data Memory Spaces"**.

User access to the program memory space is restricted to the lower half of the address range (000000h to 7FFFFFh). The exception is the use of TBLRD/TBLWT operations, which use TBLPAG<7> to permit access to the Configuration bits and Device ID sections of the configuration memory space.

Memory maps for the PIC24F16KL402 family of devices are shown in Figure 4-1.

**FIGURE 4-1: PROGRAM SPACE MEMORY MAP FOR PIC24F16KL402 FAMILY DEVICES**




**Note:** Memory areas are not displayed to scale.

### 4.1.1 PROGRAM MEMORY ORGANIZATION

The program memory space is organized in word-addressable blocks. Although it is treated as 24 bits wide, it is more appropriate to think of each address of the program memory as a lower and upper word, with the upper byte of the upper word being unimplemented. The lower word always has an even address, while the upper word has an odd address, as shown in Figure 4-2.

Program memory addresses are always word-aligned on the lower word, and addresses are incremented or decremented by two during code execution. This arrangement also provides compatibility with data memory space addressing and makes it possible to access data in the program memory space.

### 4.1.2 HARD MEMORY VECTORS

All PIC24F devices reserve the addresses between 00000h and 000200h for hard-coded program execution vectors. A hardware Reset vector is provided to redirect code execution from the default value of the PC on device Reset to the actual start of code. A GOTO instruction is programmed by the user at 000000h, with the actual address for the start of code at 000002h.

PIC24F devices also have two Interrupt Vector Tables (IVT), located from 000004h to 0000FFh and 000104h to 0001FFh. These vector tables allow each of the many device interrupt sources to be handled by separate ISRs. A more detailed discussion of the Interrupt Vector Tables is provided in **Section 8.1 "Interrupt Vector Table (IVT)"**.

### 4.1.3 DATA EEPROM

In the PIC24F16KL402 family, the data EEPROM is mapped to the top of the user program memory space, starting at address, 7FFE00, and expanding up to address, 7FFFFF.

The data EEPROM is organized as 16-bit wide memory and 256 words deep. This memory is accessed using Table Read and Table Write operations, similar to the user code memory.

### 4.1.4 DEVICE CONFIGURATION WORDS

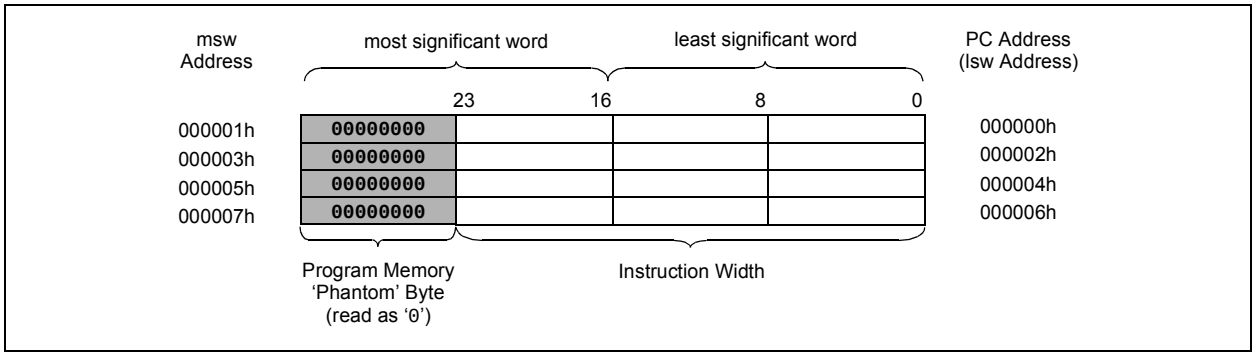
Table 4-1 provides the addresses of the device Configuration Words for the PIC24F16KL402 family. Their location in the memory map is shown in Figure 4-1.

For more information on device Configuration Words, see **Section 23.0 "Special Features"**.

**TABLE 4-1: DEVICE CONFIGURATION WORDS FOR PIC24F16KL402 FAMILY DEVICES**

| Configuration Words | Configuration Word Addresses |
|---|---|
| FBS | F80000 |
| FGS | F80004 |
| FOSCSEL | F80006 |
| FOSC | F80008 |
| FWDT | F8000A |
| FPOR | F8000C |
| FICD | F8000E |

**FIGURE 4-2: PROGRAM MEMORY ORGANIZATION**



### 4.2.5 SOFTWARE STACK

In addition to its use as a Working register, the W15 register in PIC24F devices is also used as a Software Stack Pointer. The pointer always points to the first available free word and grows from lower to higher addresses. It predecrements for stack pops and post-increments for stack pushes, as shown in Figure 4-4.

Note that for a PC push during any CALL instruction, the MSB of the PC is zero-extended before the push, ensuring that the MSB is always clear.

| **Note:** | A PC push during exception processing will concatenate the SRL register to the MSB of the PC prior to the push. |
|---|---|

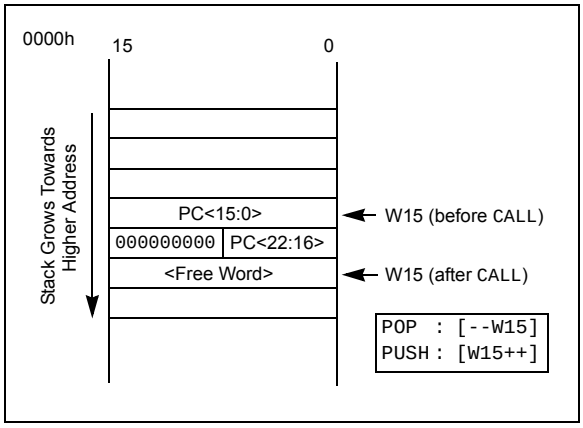
The Stack Pointer Limit Value (SPLIM) register, associated with the Stack Pointer, sets an upper address boundary for the stack. SPLIM is uninitialized at Reset. As is the case for the Stack Pointer, SPLIM<0> is forced to '0' as all stack operations must be word-aligned. Whenever an EA is generated, using W15 as a source or destination pointer, the resulting address is compared with the value in SPLIM. If the contents of the Stack Pointer (W15) and the SPLIM register are equal, and a push operation is performed, a stack error trap will not occur. The stack error trap will occur on a subsequent push operation.

Thus, for example, if it is desirable to cause a stack error trap when the stack grows beyond address, 0DF6, in RAM, initialize the SPLIM with the value, 0DF4.

Similarly, a Stack Pointer underflow (stack error) trap is generated when the Stack Pointer address is found to be less than 0800h. This prevents the stack from interfering with the Special Function Register (SFR) space.

| **Note:** | A write to the SPLIM register should not be immediately followed by an indirect read operation using W15. |
|---|---|

**FIGURE 4-4: CALL STACK FRAME**




## 4.3 Interfacing Program and Data Memory Spaces

The PIC24F architecture uses a 24-bit wide program space and 16-bit wide data space. The architecture is also a modified Harvard scheme, meaning that data can also be present in the program space. To use this data successfully, it must be accessed in a way that preserves the alignment of information in both spaces.

Apart from the normal execution, the PIC24F architecture provides two methods by which the program space can be accessed during operation:

- Using table instructions to access individual bytes or words anywhere in the program space
- Remapping a portion of the program space into the data space, PSV

Table instructions allow an application to read or write small areas of the program memory. This makes the method ideal for accessing data tables that need to be updated from time to time. It also allows access to all bytes of the program word. The remapping method allows an application to access a large block of data on a read-only basis, which is ideal for look-ups from a large table of static data. It can only access the least significant word (lsw) of the program word.

### 4.3.1 ADDRESSING PROGRAM SPACE

Since the address ranges for the data and program spaces are 16 and 24 bits, respectively, a method is needed to create a 23-bit or 24-bit program address from 16-bit data registers. The solution depends on the interface method to be used.

For table operations, the 8-bit Table Memory Page Address register (TBLPAG) is used to define a 32K word region within the program space. This is concatenated with a 16-bit EA to arrive at a full 24-bit program space address. In this format, the Most Significant bit (MSb) of TBLPAG is used to determine if the operation occurs in the user memory (TBLPAG<7> = 0) or the configuration memory (TBLPAG<7> = 1).

For remapping operations, the 8-bit Program Space Visibility Page Address register (PSVPAG) is used to define a 16K word page in the program space. When the MSb of the EA is '1', PSVPAG is concatenated with the lower 15 bits of the EA to form a 23-bit program space address. Unlike the table operations, this limits remapping operations strictly to the user memory area.

Table 4-20 and Figure 4-5 show how the program EA is created for table operations and remapping accesses from the data EA. Here, P<23:0> bits refer to a program space word, whereas the D<15:0> bits refer to a data space word.

## 6.0 DATA EEPROM MEMORY

**Note:** This data sheet summarizes the features of this group of PIC24F devices. It is not intended to be a comprehensive reference source. For more information on Data EEPROM, refer to the *"dsPIC33/PIC24 Family Reference Manual"*, **"Data EEPROM"** (DS39720).

The data EEPROM memory is a Nonvolatile Memory (NVM), separate from the program and volatile data RAM. Data EEPROM memory is based on the same Flash technology as program memory, and is optimized for both long retention and a higher number of erase/write cycles.

The data EEPROM is mapped to the top of the user program memory space, with the top address at program memory address, 7FFFFFh. For PIC24FXXKL4XX devices, the size of the data EEPROM is 256 words (7FFE00h to 7FFFFFh). For PIC24FXXKL3XX devices, the size of the data EEPROM is 128 words (7FFF00h to 7FFFFFh). The data EEPROM is not implemented in PIC24F08KL20X or PIC24F04KL10X devices.

The data EEPROM is organized as 16-bit wide memory. Each word is directly addressable, and is readable and writable during normal operation over the entire VDD range.

Unlike the Flash program memory, normal program execution is not stopped during a data EEPROM program or erase operation.

The data EEPROM programming operations are controlled using the three NVM Control registers:

- NVMCON: Nonvolatile Memory Control Register
- NVMKEY: Nonvolatile Memory Key Register
- NVMADR: Nonvolatile Memory Address Register

### 6.1 NVMCON Register

The NVMCON register (Register 6-1) is also the primary control register for data EEPROM program/erase operations. The upper byte contains the control bits used to start the program or erase cycle, and the flag bit to indicate if the operation was successfully performed. The lower byte of NVMCOM configures the type of NVM operation that will be performed.

### 6.2 NVMKEY Register

The NVMKEY is a write-only register that is used to prevent accidental writes or erasures of data EEPROM locations.

To start any programming or erase sequence, the following instructions must be executed first, in the exact order provided:

1. Write 55h to NVMKEY.
2. Write AAh to NVMKEY.

After this sequence, a write will be allowed to the NVMCON register for one instruction cycle. In most cases, the user will simply need to set the WR bit in the NVMCON register to start the program or erase cycle. Interrupts should be disabled during the unlock sequence.

The MPLAB® C30 C compiler provides a defined library procedure (`builtin_write_NVM`) to perform the unlock sequence. Example 6-1 illustrates how the unlock sequence can be performed with in-line assembly.

**EXAMPLE 6-1: DATA EEPROM UNLOCK SEQUENCE**

```
//Disable Interrupts For 5 instructions
asm volatile("disi #5");
//Issue Unlock Sequence
asm volatile ("mov #0x55, W0      \n"
              "mov W0, NVMKEY     \n"
              "mov #0xAA, W1      \n"
              "mov W1, NVMKEY     \n");
// Perform Write/Erase operations
asm volatile ("bset NVMCON, #WR   \n"
              "nop                \n"
              "nop                \n");
```

## 8.3 Interrupt Control and Status Registers

Depending on the particular device, the PIC24F16KL402 family of devices implements up to 28 registers for the interrupt controller:

- INTCON1
- INTCON2
- IFS0 through IFS5
- IEC0 through IEC5
- IPC0 through IPC7, ICP9, IPC12, ICP16, ICP18 and IPC20
- INTTREG

Global interrupt control functions are controlled from INTCON1 and INTCON2. INTCON1 contains the Interrupt Nesting Disable (NSTDIS) bit, as well as the control and status flags for the processor trap sources. The INTCON2 register controls the external interrupt request signal behavior and the use of the AIV table.

The IFSx registers maintain all of the interrupt request flags. Each source of interrupt has a status bit, which is set by the respective peripherals or external signal, and is cleared via software.

The IECx registers maintain all of the interrupt enable bits. These control bits are used to individually enable interrupts from the peripherals or external signals.

The IPCx registers are used to set the Interrupt Priority Level for each source of interrupt. Each user interrupt source can be assigned to one of eight priority levels.

The INTTREG register contains the associated interrupt vector number and the new CPU Interrupt Priority Level, which are latched into the Vector Number (VECNUM<6:0>) and the Interrupt Level (ILR<3:0>) bit fields in the INTTREG register. The new Interrupt Priority Level is the priority of the pending interrupt.

The interrupt sources are assigned to the IFSx, IECx and IPCx registers in the same sequence listed in Table 8-2. For example, the INT0 (External Interrupt 0) is depicted as having a vector number and a natural order priority of 0. The INT0IF status bit is found in IFS0<0>, the INT0IE enable bit in IEC0<0> and the INT0IP<2:0> priority bits are in the first position of IPC0 (IPC0<2:0>).

Although they are not specifically part of the interrupt control hardware, two of the CPU control registers contain bits that control interrupt functionality. The ALU STATUS Register (SR) contains the IPL<2:0> bits (SR<7:5>). These indicate the current CPU Interrupt Priority Level. The user may change the current CPU priority level by writing to the IPL bits.

The CORCON register contains the IPL3 bit, which together with the IPL<2:0> bits, also indicates the current CPU priority level. IPL3 is a read-only bit so that the trap events cannot be masked by the user's software.

All interrupt registers are described in Register 8-3 through Register 8-30, in the following sections.

**REGISTER 8-5: IFS0: INTERRUPT FLAG STATUS REGISTER 0**

| R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| NVMIF | — | AD1IF | U1TXIF | U1RXIF | — | — | T3IF |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| T2IF | CCP2IF | — | — | T1IF | CCP1IF | — | INT0IF |
| 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 15 **NVMIF:** NVM Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 14 **Unimplemented:** Read as '0'

bit 13 **AD1IF:** A/D Conversion Complete Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 12 **U1TXIF:** UART1 Transmitter Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 11 **U1RXIF:** UART1 Receiver Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 10-9 **Unimplemented:** Read as '0'

bit 8 **T3IF:** Timer3 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 7 **T2IF:** Timer2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 6 **CCP2IF:** Capture/Compare/PWM2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 5-4 **Unimplemented:** Read as '0'

bit 3 **T1IF:** Timer1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 2 **CCP1IF:** Capture/Compare/PWM1 Interrupt Flag Status bit (ECCP1 on PIC24FXXKL40X devices)
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 1 **Unimplemented:** Read as '0'

bit 0 **INT0IF:** External Interrupt 0 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

**REGISTER 8-28: IPC18: INTERRUPT PRIORITY CONTROL REGISTER 18**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-1 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | HLVDIP2 | HLVDIP1 | HLVDIP0 |
| 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 15-3 **Unimplemented:** Read as ‘0’

bit 2-0 **HLVDIP<2:0>:** High/Low-Voltage Detect Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

**REGISTER 8-29: IPC20: INTERRUPT PRIORITY CONTROL REGISTER 20**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-1 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | ULPWUIP2 | ULPWUIP1 | ULPWUIP0 |
| 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 15-3 **Unimplemented:** Read as ‘0’

bit 6-4 **ULPWUIP<2:0>:** Ultra Low-Power Wake-up Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

## 10.4 Doze Mode

Generally, changing clock speed and invoking one of the power-saving modes are the preferred strategies for reducing power consumption. There may be circumstances, however, where this is not practical. For example, it may be necessary for an application to maintain uninterrupted, synchronous communication, even while it is doing nothing else. Reducing system clock speed may introduce communication errors, while using a power-saving mode may stop communications completely.

Doze mode is a simple and effective alternative method to reduce power consumption while the device is still executing code. In this mode, the system clock continues to operate from the same source and at the same speed. Peripheral modules continue to be clocked at the same speed, while the CPU clock speed is reduced. Synchronization between the two clock domains is maintained, allowing the peripherals to access the SFRs while the CPU executes code at a slower rate.

Doze mode is enabled by setting the DOZEN bit (CLKDIV<11>). The ratio between peripheral and core clock speed is determined by the DOZE<2:0> bits (CLKDIV<14:12>). There are eight possible configurations, from 1:1 to 1:128, with 1:1 being the default.

It is also possible to use Doze mode to selectively reduce power consumption in event driven applications. This allows clock-sensitive functions, such as synchronous communications, to continue without interruption. Meanwhile, the CPU Idles, waiting for something to invoke an interrupt routine. Enabling the automatic return to full-speed CPU operation on interrupts is enabled by setting the ROI bit (CLKDIV<15>). By default, interrupt events have no effect on Doze mode operation.

## 10.5 Selective Peripheral Module Control

Idle and Doze modes allow users to substantially reduce power consumption by slowing or stopping the CPU clock. Even so, peripheral modules still remain clocked and thus, consume power. There may be cases where the application needs what these modes do not provide: the allocation of power resources to CPU processing, with minimal power consumption from the peripherals.

PIC24F devices address this requirement by allowing peripheral modules to be selectively disabled, reducing or eliminating their power consumption. This can be done with two control bits:

- The Peripheral Enable bit, generically named, “XXXEN”, located in the module’s main control SFR.
- The Peripheral Module Disable (PMD) bit, generically named, “XXXMD”, located in one of the PMD Control registers.

Both bits have similar functions in enabling or disabling its associated module. Setting the PMD bit for a module disables all clock sources to that module, reducing its power consumption to an absolute minimum. In this state, the control and status registers associated with the peripheral will also be disabled, so writes to those registers will have no effect, and read values will be invalid. Many peripheral modules have a corresponding PMD bit.

In contrast, disabling a module by clearing its XXXEN bit, disables its functionality, but leaves its registers available to be read and written to. Power consumption is reduced, but not by as much as when the PMD bits are used.

To achieve more selective power savings, peripheral modules can also be selectively disabled when the device enters Idle mode. This is done through the control bit of the generic name format, “XXXIDL”. By default, all modules that can operate during Idle mode will do so. Using the disable on Idle feature disables the module while in Idle mode, allowing further reduction of power consumption during Idle mode. This enhances power savings for extremely critical power applications.

**REGISTER 16-1: CCPxCON: CCPx CONTROL REGISTER (STANDARD CCP MODULES)**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | DCxB1 | DCxB0 | CCPxM3(1) | CCPxM2(1) | CCPxM1(1) | CCPxM0(1) |
| 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 15-6 **Unimplemented:** Read as '0'

bit 5-4 **DCxB<1:0>**: PWM Duty Cycle Bit 1 and Bit 0 for CCPx Module bits

Capture and Compare modes:
Unused.

PWM mode:
These bits are the two Least Significant bits (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight Most Significant bits (DCxB<9:2>) of the duty cycle are found in CCPRxL.

bit 3-0 **CCPxM<3:0>**: CCPx Module Mode Select bits(1)

1111 = Reserved
1110 = Reserved
1101 = Reserved
1100 = PWM mode
1011 = Compare mode: Special Event Trigger; resets timer on CCPx match (CCPxIF bit is set)
1010 = Compare mode: Generates software interrupt on compare match (CCPxIF bit is set, CCPx pin reflects I/O state)
1001 = Compare mode: Initializes CCPx pin high; on compare match, forces CCPx pin low (CCPxIF bit is set)
1000 = Compare mode: Initializes CCPx pin low; on compare match, forces CCPx pin high (CCPxIF bit is set)
0111 = Capture mode: Every 16th rising edge
0110 = Capture mode: Every 4th rising edge
0101 = Capture mode: Every rising edge
0100 = Capture mode: Every falling edge
0011 = Reserved
0010 = Compare mode: Toggles output on match (CCPxIF bit is set)
0001 = Reserved
0000 = Capture/Compare/PWM is disabled (resets CCPx module)

**Note 1:** CCPxM<3:0> = 1011 will only reset the timer and not start the A/D conversion on a CCPx match.

**REGISTER 16-5: PSTR1CON: ECCP1 PULSE STEERING CONTROL REGISTER(1)**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-1 |
|---|---|---|---|---|---|---|---|
| CMPL1 | CMPL0 | — | STRSYNC | STRD | STRC | STRB | STRA |
| 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 15-8 **Unimplemented:** Read as '0'

bit 7-6 **CMPL<1:0>:** Complementary Mode Output Assignment Steering bits
- 00 = Complementary output assignment is disabled; the STR<D:A> bits are used to determine Steering mode
- 01 = P1A and P1B are selected as the complementary output pair
- 10 = P1A and P1C are selected as the complementary output pair
- 11 = P1A and P1D are selected as the complementary output pair

bit 5 **Unimplemented:** Read as '0'

bit 4 **STRSYNC:** Steering Sync bit
- 1 = Output steering update occurs on the next PWM period
- 0 = Output steering update occurs at the beginning of the instruction cycle boundary

bit 3 **STRD:** Steering Enable D bit
- 1 = P1D pin has the PWM waveform with polarity control from CCP1M<1:0>
- 0 = P1D pin is assigned to port pin

bit 2 **STRC:** Steering Enable C bit
- 1 = P1C pin has the PWM waveform with polarity control from CCP1M<1:0>
- 0 = P1C pin is assigned to port pin

bit 1 **STRB:** Steering Enable B bit
- 1 = P1B pin has the PWM waveform with polarity control from CCP1M<1:0>
- 0 = P1B pin is assigned to port pin

bit 0 **STRA:** Steering Enable A bit
- 1 = P1A pin has the PWM waveform with polarity control from CCP1M<1:0>
- 0 = P1A pin is assigned to port pin

**Note 1:** This register is only implemented on PIC24FXXKL40X/30X devices. In addition, PWM Steering mode is available only when CCP1M<3:2> = 11 and PM<1:0> = 00.

**REGISTER 17-10: PADCFG1: PAD CONFIGURATION CONTROL REGISTER**

| U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | SDO2DIS(1) | SCK2DIS(1) | SDO1DIS | SCK1DIS |
| bit 15 | | | | | | | bit 8 |

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| 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 15-12 **Unimplemented:** Read as '0'

bit 11 **SDO2DIS:** MSSP2 SDO2 Pin Disable bit(1)
- 1 = The SPI output data (SDO2) of MSSP2 to the pin is disabled
- 0 = The SPI output data (SDO2) of MSSP2 is output to the pin

bit 10 **SCK2DIS:** MSSP2 SCK2 Pin Disable bit(1)
- 1 = The SPI clock (SCK2) of MSSP2 to the pin is disabled
- 0 = The SPI clock (SCK2) of MSSP2 is output to the pin

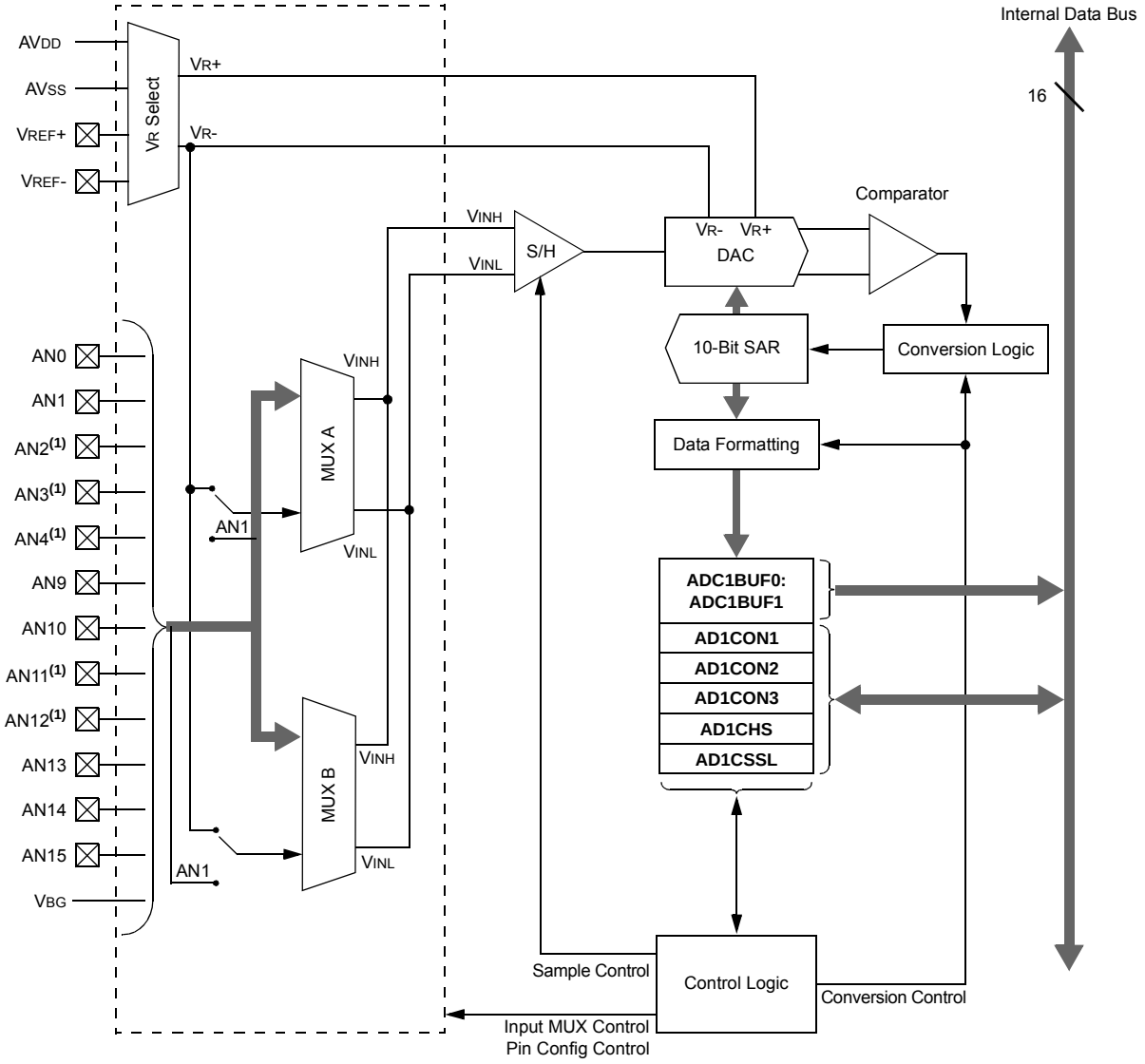
bit 9 **SDO1DIS:** MSSP1 SDO1 Pin Disable bit
- 1 = The SPI output data (SDO1) of MSSP1 to the pin is disabled
- 0 = The SPI output data (SDO1) of MSSP1 is output to the pin

bit 8 **SCK1DIS:** MSSP1 SCK1 Pin Disable bit
- 1 = The SPI clock (SCK1) of MSSP1 to the pin is disabled
- 0 = The SPI clock (SCK1) of MSSP1 is output to the pin

bit 7-0 **Unimplemented:** Read as '0'

**Note 1:** These bits are implemented only on PIC24FXXKL40X/30X devices.

**FIGURE 19-1: 10-BIT HIGH-SPEED A/D CONVERTER BLOCK DIAGRAM**




**Note 1:** Unimplemented in 14-pin devices.

## 21.0 COMPARATOR VOLTAGE REFERENCE

**Note:** This data sheet summarizes the features of this group of PIC24F devices. It is not intended to be a comprehensive reference source. For more information on the Comparator Voltage Reference, refer to the *"dsPIC33/PIC24 Family Reference Manual"*, **"Comparator Voltage Reference Module"** (DS39709).

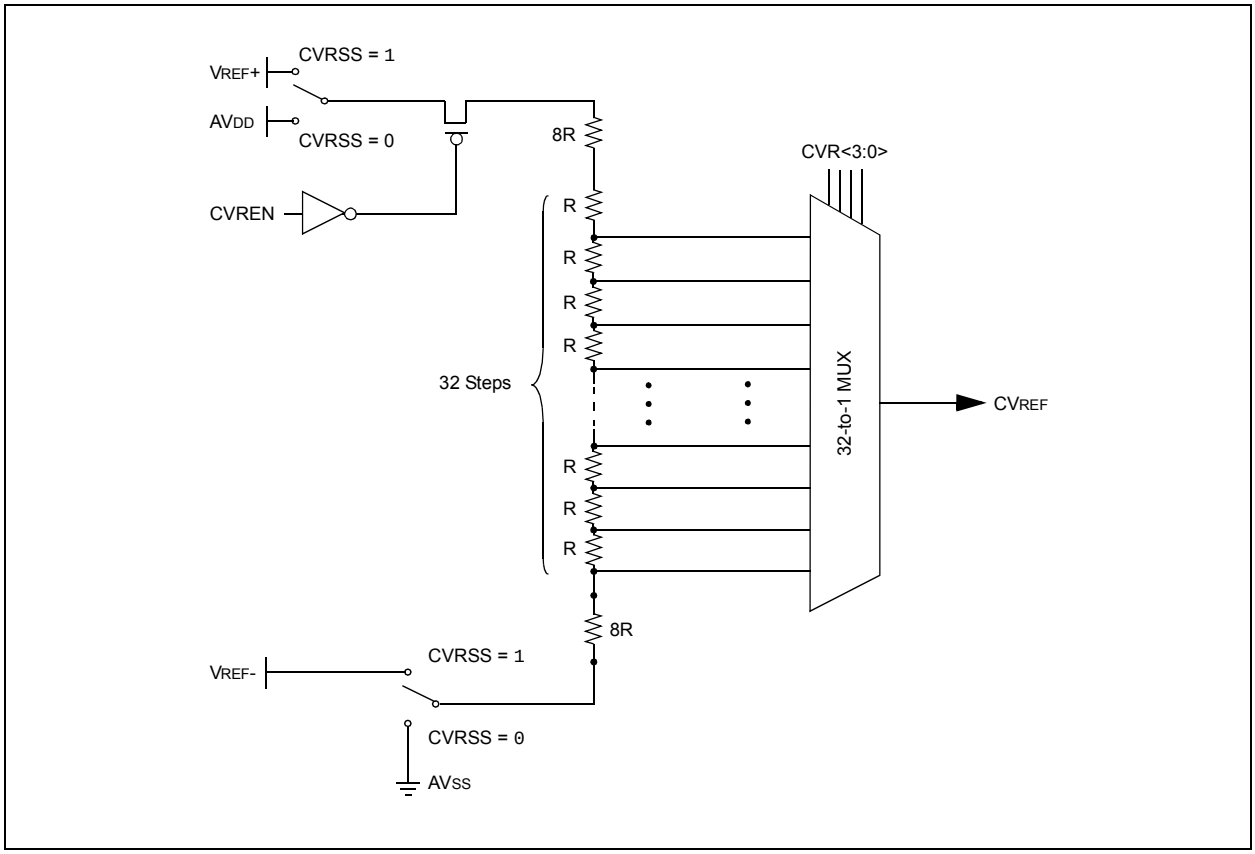
### 21.1 Configuring the Comparator Voltage Reference

The comparator voltage reference module is controlled through the CVRCON register (Register 21-1). The comparator voltage reference provides a range of output voltages, with 32 distinct levels.

The comparator voltage reference supply voltage can come from either VDD and VSS, or the external VREF+ and VREF-. The voltage source is selected by the CVRSS bit (CVRCON<5>).

The settling time of the comparator voltage reference must be considered when changing the CVREF output.

**FIGURE 21-1: COMPARATOR VOLTAGE REFERENCE BLOCK DIAGRAM**



## 24.6 MPLAB X SIM Software Simulator

The MPLAB X SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

## 24.7 MPLAB REAL ICE In-Circuit Emulator System

The MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ-11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 24.8 MPLAB ICD 3 In-Circuit Debugger System

The MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB IDE.

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

## 24.9 PICkit 3 In-Circuit Debugger/ Programmer

The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB IDE. The MPLAB PICkit 3 is connected to the design engineer's PC using a full-speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the Reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™).

## 24.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at $V_{DDMIN}$ and $V_{DDMAX}$ for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.

**TABLE 25-2: INSTRUCTION SET OVERVIEW**

| Assembly Mnemonic | Assembly Syntax | | Description | # of Words | # of Cycles | Status Flags Affected |
|---|---|---|---|---|---|---|
| ADD | ADD | f | f = f + WREG | 1 | 1 | C, DC, N, OV, Z |
| | ADD | f,WREG | WREG = f + WREG | 1 | 1 | C, DC, N, OV, Z |
| | ADD | #lit10,Wn | Wd = lit10 + Wd | 1 | 1 | C, DC, N, OV, Z |
| | ADD | Wb,Ws,Wd | Wd = Wb + Ws | 1 | 1 | C, DC, N, OV, Z |
| | ADD | Wb,#lit5,Wd | Wd = Wb + lit5 | 1 | 1 | C, DC, N, OV, Z |
| ADDC | ADDC | f | f = f + WREG + (C) | 1 | 1 | C, DC, N, OV, Z |
| | ADDC | f,WREG | WREG = f + WREG + (C) | 1 | 1 | C, DC, N, OV, Z |
| | ADDC | #lit10,Wn | Wd = lit10 + Wd + (C) | 1 | 1 | C, DC, N, OV, Z |
| | ADDC | Wb,Ws,Wd | Wd = Wb + Ws + (C) | 1 | 1 | C, DC, N, OV, Z |
| | ADDC | Wb,#lit5,Wd | Wd = Wb + lit5 + (C) | 1 | 1 | C, DC, N, OV, Z |
| AND | AND | f | f = f .AND. WREG | 1 | 1 | N, Z |
| | AND | f,WREG | WREG = f .AND. WREG | 1 | 1 | N, Z |
| | AND | #lit10,Wn | Wd = lit10 .AND. Wd | 1 | 1 | N, Z |
| | AND | Wb,Ws,Wd | Wd = Wb .AND. Ws | 1 | 1 | N, Z |
| | AND | Wb,#lit5,Wd | Wd = Wb .AND. lit5 | 1 | 1 | N, Z |
| ASR | ASR | f | f = Arithmetic Right Shift f | 1 | 1 | C, N, OV, Z |
| | ASR | f,WREG | WREG = Arithmetic Right Shift f | 1 | 1 | C, N, OV, Z |
| | ASR | Ws,Wd | Wd = Arithmetic Right Shift Ws | 1 | 1 | C, N, OV, Z |
| | ASR | Wb,Wns,Wnd | Wnd = Arithmetic Right Shift Wb by Wns | 1 | 1 | N, Z |
| | ASR | Wb,#lit5,Wnd | Wnd = Arithmetic Right Shift Wb by lit5 | 1 | 1 | N, Z |
| BCLR | BCLR | f,#bit4 | Bit Clear f | 1 | 1 | None |
| | BCLR | Ws,#bit4 | Bit Clear Ws | 1 | 1 | None |
| BRA | BRA | C,Expr | Branch if Carry | 1 | 1 (2) | None |
| | BRA | GE,Expr | Branch if Greater than or Equal | 1 | 1 (2) | None |
| | BRA | GEU,Expr | Branch if Unsigned Greater than or Equal | 1 | 1 (2) | None |
| | BRA | GT,Expr | Branch if Greater than | 1 | 1 (2) | None |
| | BRA | GTU,Expr | Branch if Unsigned Greater than | 1 | 1 (2) | None |
| | BRA | LE,Expr | Branch if Less than or Equal | 1 | 1 (2) | None |
| | BRA | LEU,Expr | Branch if Unsigned Less than or Equal | 1 | 1 (2) | None |
| | BRA | LT,Expr | Branch if Less than | 1 | 1 (2) | None |
| | BRA | LTU,Expr | Branch if Unsigned Less than | 1 | 1 (2) | None |
| | BRA | N,Expr | Branch if Negative | 1 | 1 (2) | None |
| | BRA | NC,Expr | Branch if Not Carry | 1 | 1 (2) | None |
| | BRA | NN,Expr | Branch if Not Negative | 1 | 1 (2) | None |
| | BRA | NOV,Expr | Branch if Not Overflow | 1 | 1 (2) | None |
| | BRA | NZ,Expr | Branch if Not Zero | 1 | 1 (2) | None |
| | BRA | OV,Expr | Branch if Overflow | 1 | 1 (2) | None |
| | BRA | Expr | Branch Unconditionally | 1 | 2 | None |
| | BRA | Z,Expr | Branch if Zero | 1 | 1 (2) | None |
| | BRA | Wn | Computed Branch | 1 | 2 | None |
| BSET | BSET | f,#bit4 | Bit Set f | 1 | 1 | None |
| | BSET | Ws,#bit4 | Bit Set Ws | 1 | 1 | None |
| BSW | BSW.C | Ws,Wb | Write C bit to Ws<Wb> | 1 | 1 | None |
| | BSW.Z | Ws,Wb | Write Z bit to Ws<Wb> | 1 | 1 | None |
| BTG | BTG | f,#bit4 | Bit Toggle f | 1 | 1 | None |
| | BTG | Ws,#bit4 | Bit Toggle Ws | 1 | 1 | None |
| BTSC | BTSC | f,#bit4 | Bit Test f, Skip if Clear | 1 | 1 (2 or 3) | None |
| | BTSC | Ws,#bit4 | Bit Test Ws, Skip if Clear | 1 | 1 (2 or 3) | None |

**TABLE 25-2: INSTRUCTION SET OVERVIEW (CONTINUED)**

| Assembly Mnemonic | Assembly Syntax | | Description | # of Words | # of Cycles | Status Flags Affected |
|---|---|---|---|---|---|---|
| GOTO | GOTO | Expr | Go to Address | 2 | 2 | None |
| | GOTO | Wn | Go to Indirect | 1 | 2 | None |
| INC | INC | f | f = f + 1 | 1 | 1 | C, DC, N, OV, Z |
| | INC | f,WREG | WREG = f + 1 | 1 | 1 | C, DC, N, OV, Z |
| | INC | Ws,Wd | Wd = Ws + 1 | 1 | 1 | C, DC, N, OV, Z |
| INC2 | INC2 | f | f = f + 2 | 1 | 1 | C, DC, N, OV, Z |
| | INC2 | f,WREG | WREG = f + 2 | 1 | 1 | C, DC, N, OV, Z |
| | INC2 | Ws,Wd | Wd = Ws + 2 | 1 | 1 | C, DC, N, OV, Z |
| IOR | IOR | f | f = f .IOR. WREG | 1 | 1 | N, Z |
| | IOR | f,WREG | WREG = f .IOR. WREG | 1 | 1 | N, Z |
| | IOR | #lit10,Wn | Wd = lit10 .IOR. Wd | 1 | 1 | N, Z |
| | IOR | Wb,Ws,Wd | Wd = Wb .IOR. Ws | 1 | 1 | N, Z |
| | IOR | Wb,#lit5,Wd | Wd = Wb .IOR. lit5 | 1 | 1 | N, Z |
| LNK | LNK | #lit14 | Link Frame Pointer | 1 | 1 | None |
| LSR | LSR | f | f = Logical Right Shift f | 1 | 1 | C, N, OV, Z |
| | LSR | f,WREG | WREG = Logical Right Shift f | 1 | 1 | C, N, OV, Z |
| | LSR | Ws,Wd | Wd = Logical Right Shift Ws | 1 | 1 | C, N, OV, Z |
| | LSR | Wb,Wns,Wnd | Wnd = Logical Right Shift Wb by Wns | 1 | 1 | N, Z |
| | LSR | Wb,#lit5,Wnd | Wnd = Logical Right Shift Wb by lit5 | 1 | 1 | N, Z |
| MOV | MOV | f,Wn | Move f to Wn | 1 | 1 | None |
| | MOV | [Wns+Slit10],Wnd | Move [Wns+Slit10] to Wnd | 1 | 1 | None |
| | MOV | f | Move f to f | 1 | 1 | N, Z |
| | MOV | f,WREG | Move f to WREG | 1 | 1 | None |
| | MOV | #lit16,Wn | Move 16-bit Literal to Wn | 1 | 1 | None |
| | MOV.b | #lit8,Wn | Move 8-bit Literal to Wn | 1 | 1 | None |
| | MOV | Wn,f | Move Wn to f | 1 | 1 | None |
| | MOV | Wns,[Wns+Slit10] | Move Wns to [Wns+Slit10] | 1 | 1 | None |
| | MOV | Wso,Wdo | Move Ws to Wd | 1 | 1 | None |
| | MOV | WREG,f | Move WREG to f | 1 | 1 | None |
| | MOV.D | Wns,Wd | Move Double from W(ns):W(ns+1) to Wd | 1 | 2 | None |
| | MOV.D | Ws,Wnd | Move Double from Ws to W(nd+1):W(nd) | 1 | 2 | None |
| MUL | MUL.SS | Wb,Ws,Wnd | {Wnd+1, Wnd} = Signed(Wb) * Signed(Ws) | 1 | 1 | None |
| | MUL.SU | Wb,Ws,Wnd | {Wnd+1, Wnd} = Signed(Wb) * Unsigned(Ws) | 1 | 1 | None |
| | MUL.US | Wb,Ws,Wnd | {Wnd+1, Wnd} = Unsigned(Wb) * Signed(Ws) | 1 | 1 | None |
| | MUL.UU | Wb,Ws,Wnd | {Wnd+1, Wnd} = Unsigned(Wb) * Unsigned(Ws) | 1 | 1 | None |
| | MUL.SU | Wb,#lit5,Wnd | {Wnd+1, Wnd} = Signed(Wb) * Unsigned(lit5) | 1 | 1 | None |
| | MUL.UU | Wb,#lit5,Wnd | {Wnd+1, Wnd} = Unsigned(Wb) * Unsigned(lit5) | 1 | 1 | None |
| | MUL | f | W3:W2 = f * WREG | 1 | 1 | None |
| NEG | NEG | f | f = $\overline{f}$ + 1 | 1 | 1 | C, DC, N, OV, Z |
| | NEG | f,WREG | WREG = $\overline{f}$ + 1 | 1 | 1 | C, DC, N, OV, Z |
| | NEG | Ws,Wd | Wd = $\overline{Ws}$ + 1 | 1 | 1 | C, DC, N, OV, Z |
| NOP | NOP | | No Operation | 1 | 1 | None |
| | NOPR | | No Operation | 1 | 1 | None |
| POP | POP | f | Pop f from Top-of-Stack (TOS) | 1 | 1 | None |
| | POP | Wdo | Pop from Top-of-Stack (TOS) to Wdo | 1 | 1 | None |
| | POP.D | Wnd | Pop from Top-of-Stack (TOS) to W(nd):W(nd+1) | 1 | 2 | None |
| | POP.S | | Pop Shadow Registers | 1 | 1 | All |
| PUSH | PUSH | f | Push f to Top-of-Stack (TOS) | 1 | 1 | None |
| | PUSH | Wso | Push Wso to Top-of-Stack (TOS) | 1 | 1 | None |
| | PUSH.D | Wns | Push W(ns):W(ns+1) to Top-of-Stack (TOS) | 1 | 2 | None |
| | PUSH.S | | Push Shadow Registers | 1 | 1 | None |

**TABLE 26-36: A/D CONVERSION TIMING REQUIREMENTS(1)**

| AC CHARACTERISTICS | | | Standard Operating Conditions: 1.8V to 3.6V (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for Industrial | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic** | **Min.** | **Typ** | **Max.** | **Units** | **Conditions** |
| **Clock Parameters** | | | | | | | |
| AD50 | TAD | A/D Clock Period | 75 | — | — | ns | TCY = 75 ns, AD1CON3 is in default state |
| AD51 | TRC | A/D Internal RC Oscillator Period | — | 250 | — | ns | |
| **Conversion Rate** | | | | | | | |
| AD55 | TCONV | Conversion Time | — | 12 | — | TAD | |
| AD56 | FCNV | Throughput Rate | — | — | 500 | ksps | AVDD ≥ 2.7V |
| AD57 | TSAMP | Sample Time | — | 1 | — | TAD | |
| AD58 | TACQ | Acquisition Time | 750 | — | — | ns | **(Note 2)** |
| AD59 | TSWC | Switching Time from Convert to Sample | — | — | **(Note 3)** | — | |
| AD60 | TDIS | Discharge Time | 0.5 | — | — | TAD | |
| **Clock Parameters** | | | | | | | |
| AD61 | TPSS | Sample Start Delay from Setting Sample bit (SAMP) | 2 | — | 3 | TAD | |

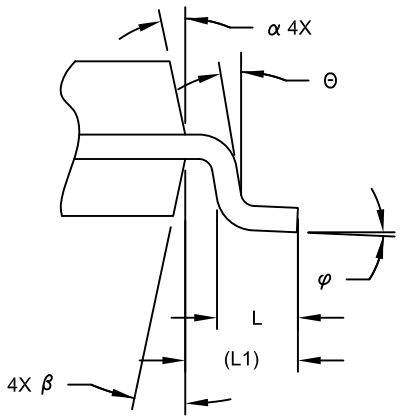
**Note 1:** Because the sample caps will eventually lose charge, clock rates below 10 kHz can affect linearity performance, especially at elevated temperatures.

**2:** The time for the holding capacitor to acquire the “New” input voltage when the voltage changes full scale after the conversion (VDD to VSS or VSS to VDD).

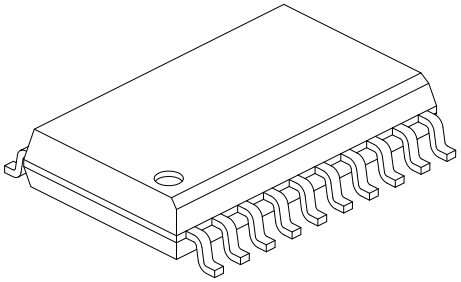
**3:** On the following cycle of the device clock.

## 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

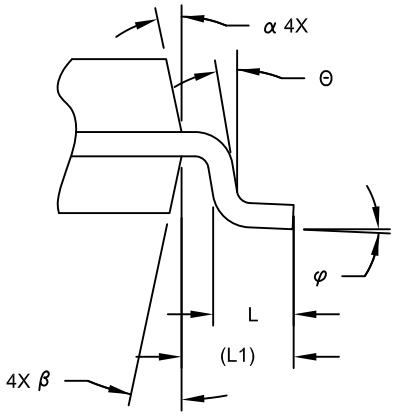
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

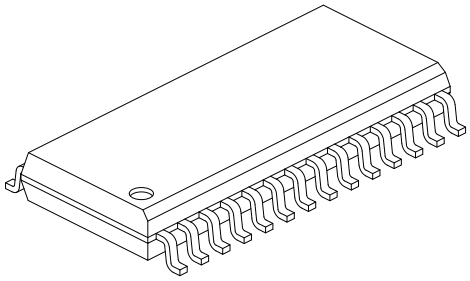
Microchip Technology Drawing No. C04-094C Sheet 2 of 2

## 28-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 | | 28 | |
| 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 | | 17.90 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 | 0.18 | - | 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

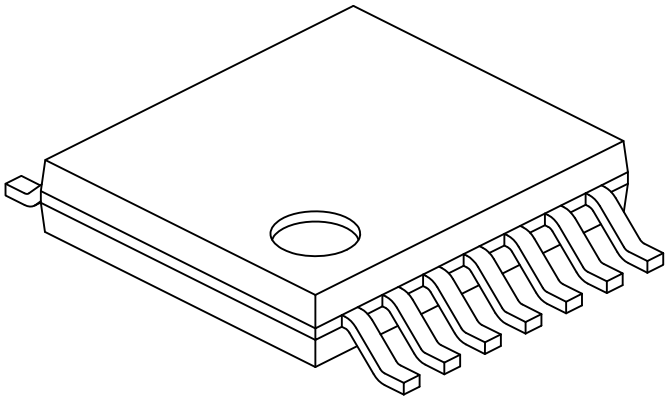
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing C04-052C Sheet 2 of 2

### 14-Lead Plastic Thin Shrink Small Outline (ST) - 4.4 mm Body [TSSOP]

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



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 14 | |
| Pitch | e | | 0.65 BSC | |
| Overall Height | A | - | - | 1.20 |
| Molded Package Thickness | A2 | 0.80 | 1.00 | 1.05 |
| Standoff | A1 | 0.05 | - | 0.15 |
| Overall Width | E | | 6.40 BSC | |
| Molded Package Width | E1 | 4.30 | 4.40 | 4.50 |
| Molded Package Length | D | 4.90 | 5.00 | 5.10 |
| Foot Length | L | 0.45 | 0.60 | 0.75 |
| Footprint | (L1) | | 1.00 REF | |
| Foot Angle | φ | 0° | - | 8° |
| Lead Thickness | c | 0.09 | - | 0.20 |
| Lead Width | b | 0.19 | - | 0.30 |

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.15mm per side.
3. 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.

Microchip Technology Drawing No. C04-087C Sheet 2 of 2