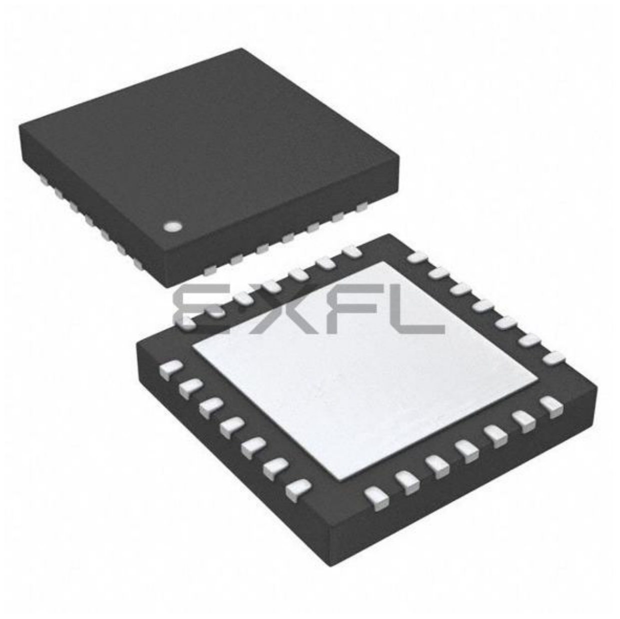
Microchip Technology - PIC24F08KL402-I/ML Datasheet



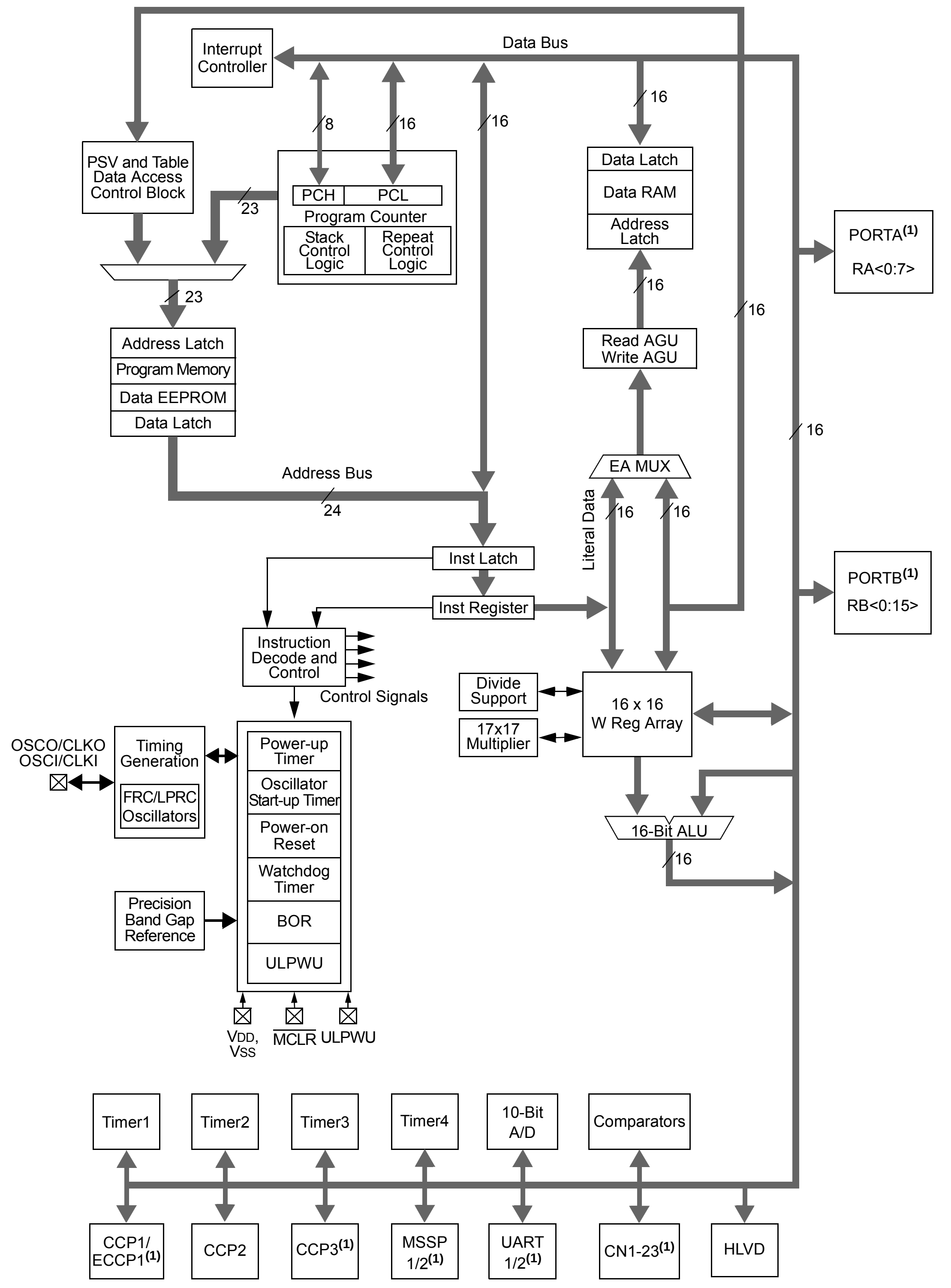
What is "Embedded - Microcontrollers"?

"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	24
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	28-VQFN Exposed Pad
Supplier Device Package	28-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic24f08kl402-i-ml

FIGURE 1-1: PIC24F16KL402 FAMILY GENERAL BLOCK DIAGRAM



Note 1: All pins or features are not implemented on all device pinout configurations. See Table 1-4 and Table 1-5 for I/O port pin descriptions.

4.3.2 DATA ACCESS FROM PROGRAM MEMORY AND DATA EEPROM MEMORY USING TABLE INSTRUCTIONS

The TBLRDL and TBLWTL instructions offer a direct method of reading or writing the lower word of any address within the program memory without going through data space. It also offers a direct method of reading or writing a word of any address within data EEPROM memory. The TBLRDH and TBLWTH instructions are the only method to read or write the upper 8 bits of a program space word as data.

Note:	The TBLRDH and TBLWTH instructions are not used while accessing data EEPROM memory.

The PC is incremented by two for each successive 24-bit program word. This allows program memory addresses to directly map to data space addresses. Program memory can thus be regarded as two, 16-bit word-wide address spaces, residing side by side, each with the same address range. TBLRDL and TBLWTL access the space which contains the least significant data word, and TBLRDH and TBLWTH access the space which contains the upper data byte.

Two table instructions are provided to move byte or word-sized (16-bit) data to and from program space. Both function as either byte or word operations.

1. TBLRDL (Table Read Low): In Word mode, it maps the lower word of the program space location (P<15:0>) to a data address (D<15:0>).

   In Byte mode, either the upper or lower byte of the lower program word is mapped to the lower byte of a data address. The upper byte is selected when the byte select is ‘1’; the lower byte is selected when it is ‘0’.

2. TBLRDH (Table Read High): In Word mode, it maps the entire upper word of a program address (P<23:16>) to a data address. Note that D<15:8>, the ‘phantom’ byte, will always be ‘0’.

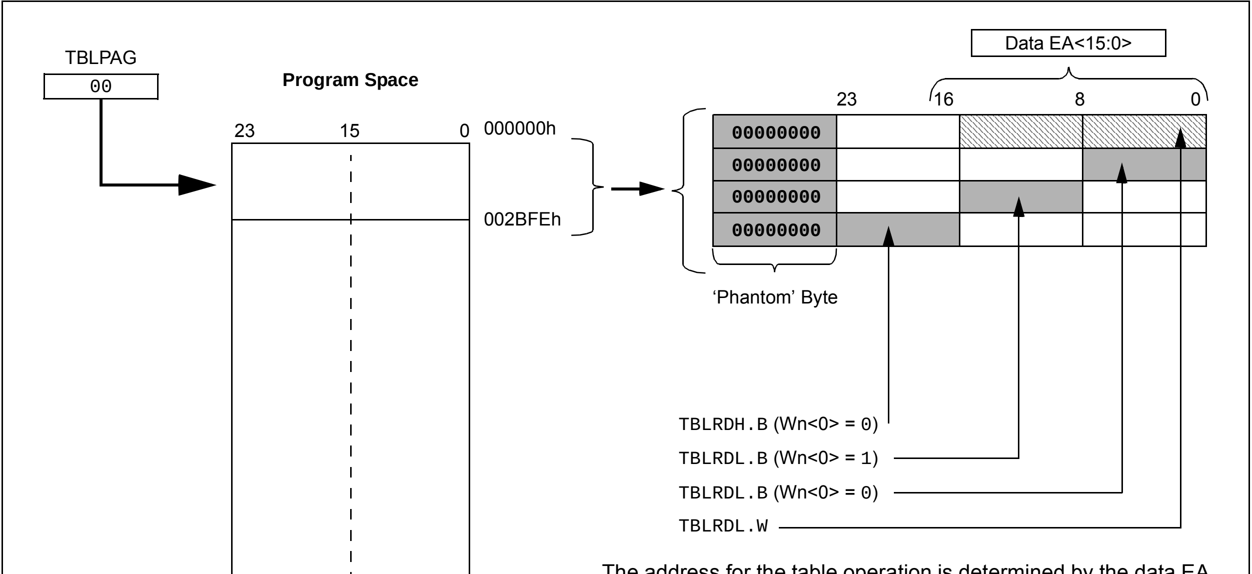
   In Byte mode, it maps the upper or lower byte of the program word to D<7:0> of the data address, as above. Note that the data will always be ‘0’ when the upper ‘phantom’ byte is selected (byte select = 1).

In a similar fashion, two table instructions, TBLWTH and TBLWTL, are used to write individual bytes or words to a program space address. The details of their operation are explained in **Section 5.0 “Flash Program Memory”**.

For all table operations, the area of program memory space to be accessed is determined by the Table Memory Page Address register (TBLPAG). TBLPAG covers the entire program memory space of the device, including user and configuration spaces. When TBLPAG<7> = 0, the table page is located in the user memory space. When TBLPAG<7> = 1, the page is located in configuration space.

Note:	Only Table Read operations will execute in the configuration memory space, and only then, in implemented areas, such as the Device ID. Table write operations are not allowed.

FIGURE 4-6: ACCESSING PROGRAM MEMORY WITH TABLE INSTRUCTIONS



The address for the table operation is determined by the data EA within the page defined by the TBLPAG register. Only read operations are provided; write operations are also valid in the user memory area.

EXAMPLE 5-4: LOADING THE WRITE BUFFERS – ‘C’ LANGUAGE CODE

```
// C example using MPLAB C30

   #define NUM_INSTRUCTION_PER_ROW 64
   int __attribute__ ((space(auto_psv))) progAddr = &progAddr; // Global variable located in Pgm Memory
   unsigned int offset;
   unsigned int i;
   unsigned int progData[2*NUM_INSTRUCTION_PER_ROW];          // Buffer of data to write

   //Set up NVMCON for row programming
   NVMCON = 0x4004;                                           // Initialize NVMCON

   //Set up pointer to the first memory location to be written
   TBLPAG = __builtin_tblpage(&progAddr);                     // Initialize PM Page Boundary SFR
   offset = &progAddr & 0xFFFF;                               // Initialize lower word of address

   //Perform TBLWT instructions to write necessary number of latches
   for(i=0; i < 2*NUM_INSTRUCTION_PER_ROW; i++)
   {
       __builtin_tblwtl(offset, progData[i++]);               // Write to address low word
       __builtin_tblwth(offset, progData[i]);                 // Write to upper byte
       offset = offset + 2;                                   // Increment address
   }
```

EXAMPLE 5-5: INITIATING A PROGRAMMING SEQUENCE – ASSEMBLY LANGUAGE CODE

```
        DISI    #5                        ; Block all interrupts
                                            for next 5 instructions
        MOV     #0x55, W0
        MOV     W0, NVMKEY                ; Write the 55 key
        MOV     #0xAA, W1                 ;
        MOV     W1, NVMKEY                ; Write the AA key
        BSET    NVMCON, #WR               ; Start the erase sequence
        NOP                               ; 2 NOPs required after setting WR
        NOP                               ;
        BTSC    NVMCON, #15               ; Wait for the sequence to be completed
        BRA     $-2                       ;
```

EXAMPLE 5-6: INITIATING A PROGRAMMING SEQUENCE – ‘C’ LANGUAGE CODE

```
// C example using MPLAB C30

asm("DISI #5");               // Block all interrupts for next 5 instructions

__builtin_write_NVM();        // Perform unlock sequence and set WR
```

6.4.1.1 Data EEPROM Bulk Erase

To erase the entire data EEPROM (bulk erase), the address registers do not need to be configured because this operation affects the entire data EEPROM. The following sequence helps in performing a bulk erase:

1. Configure NVMCON to Bulk Erase mode.
2. Clear the NVMIF status bit and enable the NVM interrupt (optional).
3. Write the key sequence to NVMKEY.
4. Set the WR bit to begin the erase cycle.
5. Either poll the WR bit or wait for the NVM interrupt (NVMIF is set).

A typical bulk erase sequence is provided in Example 6-3.

6.4.2 SINGLE-WORD WRITE

To write a single word in the data EEPROM, the following sequence must be followed:

1. Erase one data EEPROM word (as mentioned in **Section 6.4.1 “Erase Data EEPROM”**) if PGMONLY bit (NVMCON<12>) is set to ‘1’.
2. Write the data word into the data EEPROM latch.
3. Program the data word into the EEPROM:
   - Configure the NVMCON register to program one EEPROM word (NVMCON<5:0> = `0001xx`).
   - Clear the NVMIF status bit and enable the NVM interrupt (optional).
   - Write the key sequence to NVMKEY.
   - Set the WR bit to begin the erase cycle.
   - Either poll the WR bit or wait for the NVM interrupt (NVMIF set).
   - To get cleared, wait until NVMIF is set.

A typical single-word write sequence is provided in Example 6-4.

EXAMPLE 6-3: DATA EEPROM BULK ERASE

```
    // Set up NVMCON to bulk erase the data EEPROM
    NVMCON = 0x4050;

    // Disable Interrupts For 5 Instructions
    asm volatile ("disi #5");

    // Issue Unlock Sequence and Start Erase Cycle
    __builtin_write_NVM();
```

EXAMPLE 6-4: SINGLE-WORD WRITE TO DATA EEPROM

```
int __attribute__ ((space(eedata))) eeData = 0x1234;  // Global variable located in EEPROM
   int newData;                                        // New data to write to EEPROM
   unsigned int offset;

   // Set up NVMCON to erase one word of data EEPROM
   NVMCON = 0x4004;

   // Set up a pointer to the EEPROM location to be erased
   TBLPAG = __builtin_tblpage(&eeData);                // Initialize EE Data page pointer
   offset = __builtin_tbloffset(&eeData);              // Initizlize lower word of address
   __builtin_tblwtl(offset, newData);                  // Write EEPROM data to write latch

   asm volatile ("disi #5");                           // Disable Interrupts For 5 Instructions
   __builtin_write_NVM();                              // Issue Unlock Sequence & Start Write Cycle
   while(NVMCONbits.WR=1);                             // Optional: Poll WR bit to wait for
                                                       // write sequence to complete
```

REGISTER 8-1: SR: ALU STATUS REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	DC(1)
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
IPL2(2,3)	IPL1(2,3)	IPL0(2,3)	RA(1)	N(1)	OV(1)	Z(1)	C(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 15-9 **Unimplemented:** Read as ‘0’

bit 7-5 **IPL<2:0>:** CPU Interrupt Priority Level Status bits(2,3)

111 = CPU Interrupt Priority Level is 7 (15); user interrupts disabled
110 = CPU Interrupt Priority Level is 6 (14)
101 = CPU Interrupt Priority Level is 5 (13)
100 = CPU Interrupt Priority Level is 4 (12)
011 = CPU Interrupt Priority Level is 3 (11)
010 = CPU Interrupt Priority Level is 2 (10)
001 = CPU Interrupt Priority Level is 1 (9)
000 = CPU Interrupt Priority Level is 0 (8)

Note 1: See Register 3-1 for the description of these bits, which are not dedicated to interrupt control functions.

2: The IPL bits are concatenated with the IPL3 bit (CORCON<3>) to form the CPU Interrupt Priority Level. The value in parentheses indicates the Interrupt Priority Level if IPL3 = 1.

3: The IPL Status bits are read-only when NSTDIS (INTCON1<15>) = 1.

Note: Bit 8 and bits 4 through 0 are described in **Section 3.0 “CPU”**.

REGISTER 8-6: IFS1: INTERRUPT FLAG STATUS REGISTER 1

R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0	R/W-0	U-0
U2TXIF[(1)]	U2RXIF[(1)]	INT2IF	—	T4IF[(1)]	—	CCP3IF[(1)]	—
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT1IF	CNIF	CMIF	BCL1IF	SSP1IF
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 **U2TXIF:** UART2 Transmitter Interrupt Flag Status bit[(1)]
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 14 **U2RXIF:** UART2 Receiver Interrupt Flag Status bit[(1)]
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 13 **INT2IF:** External Interrupt 2 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 12 **Unimplemented:** Read as ‘0’

bit 11 **T4IF:** Timer4 Interrupt Flag Status bit[(1)]
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 10 **Unimplemented:** Read as ‘0’

bit 9 **CCP3IF:** Capture/Compare/PWM3 Interrupt Flag Status bit[(1)]
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 8-5 **Unimplemented:** Read as ‘0’

bit 4 **INT1IF:** External Interrupt 1 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 3 **CNIF:** Input Change Notification Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 2 **CMIF:** Comparator Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 1 **BCL1IF:** MSSP1 I^{2}C™ Bus Collision Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 0 **SSP1IF:** MSSP1 SPI/I^{2}C Event Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

Note 1: These bits are unimplemented on PIC24FXXKL10X and PIC24FXXKL20X devices.

REGISTER 9-4: REFOCON: REFERENCE OSCILLATOR CONTROL REGISTER

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ROEN	—	ROSSLP	ROSEL	RODIV3	RODIV2	RODIV1	RODIV0
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 **ROEN:** Reference Oscillator Output Enable bit
1 = Reference oscillator is enabled on REFO pin
0 = Reference oscillator is disabled

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **ROSSLP:** Reference Oscillator Output Stop in Sleep bit
1 = Reference oscillator continues to run in Sleep
0 = Reference oscillator is disabled in Sleep

bit 12 **ROSEL:** Reference Oscillator Source Select bit
1 = Primary oscillator is used as the base clock(1)
0 = System clock is used as the base clock; the base clock reflects any clock switching of the device

bit 11-8 **RODIV<3:0>:** Reference Oscillator Divisor Select bits
1111 = Base clock value divided by 32,768
1110 = Base clock value divided by 16,384
1101 = Base clock value divided by 8,192
1100 = Base clock value divided by 4,096
1011 = Base clock value divided by 2,048
1010 = Base clock value divided by 1,024
1001 = Base clock value divided by 512
1000 = Base clock value divided by 256
0111 = Base clock value divided by 128
0110 = Base clock value divided by 64
0101 = Base clock value divided by 32
0100 = Base clock value divided by 16
0011 = Base clock value divided by 8
0010 = Base clock value divided by 4
0001 = Base clock value divided by 2
0000 = Base clock value

bit 7-0 **Unimplemented:** Read as ‘0’

Note 1: The crystal oscillator must be enabled using the FOSC<2:0> bits; the crystal maintains the operation in Sleep mode.

11.0 I/O PORTS

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 I/O Ports, refer to the *"dsPIC33/PIC24 Family Reference Manual"*, **"I/O Ports with Peripheral Pin Select (PPS)"** (DS39711). Note that the PIC24F16KL402 family devices do not support Peripheral Pin Select features.

All of the device pins (except VDD and VSS) are shared between the peripherals and the parallel I/O ports. All I/O input ports feature Schmitt Trigger inputs for improved noise immunity.

11.1 Parallel I/O (PIO) Ports

A parallel I/O port that shares a pin with a peripheral is, in general, subservient to the peripheral. The peripheral's output buffer data and control signals are provided to a pair of multiplexers. The multiplexers select whether the peripheral or the associated port has ownership of the output data and control signals of the I/O pin. Figure 11-1 illustrates how ports are shared with other peripherals and the associated I/O pin to which they are connected.

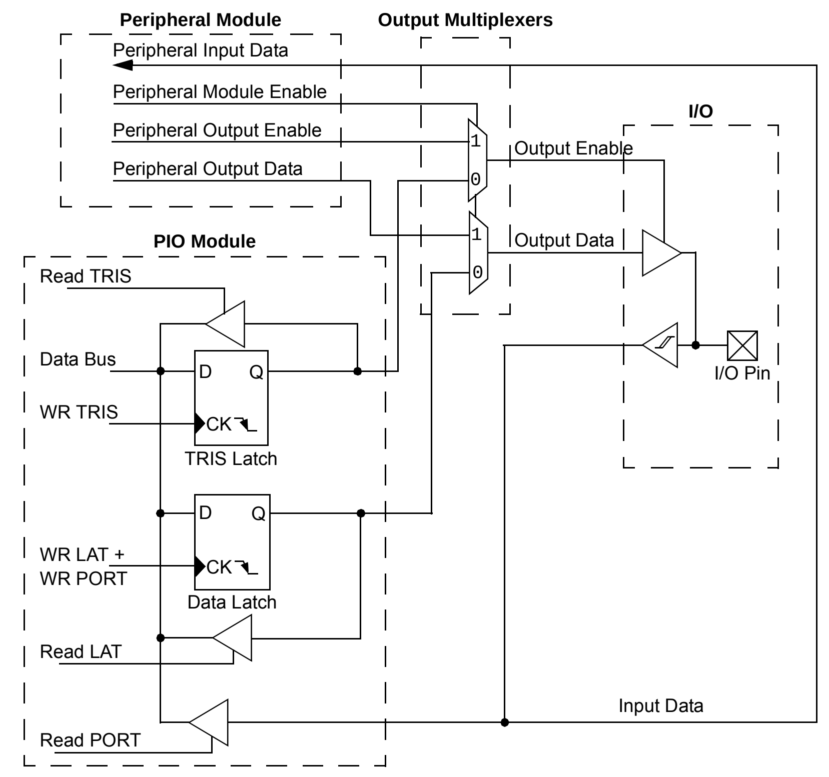
When a peripheral is enabled and the peripheral is actively driving an associated pin, the use of the pin as a general purpose output pin is disabled. The I/O pin may be read, but the output driver for the parallel port bit will be disabled. If a peripheral is enabled, but the peripheral is not actively driving a pin, that pin may be driven by a port.

All port pins have three registers directly associated with their operation as digital I/O. The Data Direction register (TRISx) determines whether the pin is an input or an output. If the data direction bit is a '1', then the pin is an input. All port pins are defined as inputs after a Reset. Reads from the Data Latch register (LATx), read the latch. Writes to the Data Latch, write the latch. Reads from the port (PORTx), read the port pins, while writes to the port pins, write the latch.

Any bit and its associated data and control registers, that are not valid for a particular device, will be disabled. That means the corresponding LATx and TRISx registers, and the port pin will read as zeros.

When a pin is shared with another peripheral or function that is defined as an input only, it is nevertheless, regarded as a dedicated port because there is no other competing source of outputs.

FIGURE 11-1: BLOCK DIAGRAM OF A TYPICAL SHARED I/O PORT STRUCTURE



REGISTER 17-3: SSPxCON1: MSSPx CONTROL REGISTER 1 (SPI MODE)

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	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV(1)	SSPEN(2)	CKP	SSPM3(3)	SSPM2(3)	SSPM1(3)	SSPM0(3)
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 **WCOL:** Write Collision Detect bit
1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPOV:** MSSPx Receive Overflow Indicator bit(1)
SPI Slave mode:
1 = A new byte is received while the SSPxBUF register is still holding the previous data. In case of overflow, the data in SSPxSR is lost. Overflow can only occur in Slave mode. The user must read the SSPxBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).
0 = No overflow

bit 5 **SSPEN:** MSSPx Enable bit(2)
1 = Enables serial port and configures SCKx, SDOx, SDIx and $\overline{\text{SSx}}$ as serial port pins
0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** Clock Polarity Select bit
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level

bit 3-0 **SSPM<3:0>:** MSSPx Mode Select bits(3)
1010 = SPI Master mode, Clock = FOSC/(2 * ([SSPxADD] + 1))(4)
0101 = SPI Slave mode, Clock = SCKx pin; $\overline{\text{SSx}}$ pin control is disabled, $\overline{\text{SSx}}$ can be used as an I/O pin
0100 = SPI Slave mode, Clock = SCKx pin; $\overline{\text{SSx}}$ pin control is enabled
0011 = SPI Master mode, Clock = TMR2 output/2
0010 = SPI Master mode, Clock = FOSC/32
0001 = SPI Master mode, Clock = FOSC/8
0000 = SPI Master mode, Clock = FOSC/2

Note 1: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPxBUF register.
2: When enabled, these pins must be properly configured as input or output.
3: Bit combinations not specifically listed here are either reserved or implemented in I^2C mode only.
4: SSPxADD value of 0 is not supported when the Baud Rate Generator is used in SPI mode.

REGISTER 17-6: SSPxCON3: MSSPx CONTROL REGISTER 3 (SPI MODE)

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

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ACKTIM	PCIE	SCIE	BOEN[(1)]	SDAHT	SBCDE	AHEN	DHEN
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	**ACKTIM:** Acknowledge Time Status bit (I^2C™ mode only) Unused in SPI mode.
bit 6	**PCIE**: Stop Condition Interrupt Enable bit (I^2C mode only) Unused in SPI mode.
bit 5	**SCIE**: Start Condition Interrupt Enable bit (I^2C mode only) Unused in SPI mode.
bit 4	**BOEN:** Buffer Overwrite Enable bit[(1)] <u>In SPI Slave mode:</u> 1 = SSPxBUF updates every time that a new data byte is shifted in, ignoring the BF bit 0 = If a new byte is received with the BF bit of the SSPxSTAT register already set, the SSPOV bit of the SSPxCON1 register is set and the buffer is not updated
bit 3	**SDAHT:** SDAx Hold Time Selection bit (I^2C mode only) Unused in SPI mode.
bit 2	**SBCDE:** Slave Mode Bus Collision Detect Enable bit (I^2C Slave mode only) Unused in SPI mode.
bit 1	**AHEN:** Address Hold Enable bit (I^2C Slave mode only) Unused in SPI mode.
bit 0	**DHEN:** Data Hold Enable bit (Slave mode only) Unused in SPI mode.

Note 1: For daisy-chained SPI operation: Allows the user to ignore all but the last received byte. SSPOV is still set when a new byte is received and BF = 1, but hardware continues to write the most recent byte to SSPxBUF.

REGISTER 18-1: UxMODE: UARTx MODE REGISTER

R/W-0	U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0[(2)]	R/W-0[(2)]
UARTEN	—	USIDL	IREN[(1)]	RTSMD	—	UEN1	UEN0
bit 15							bit 8

R/C-0, HC	R/W-0	R/W-0, HC	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WAKE	LPBACK	ABAUD	RXINV	BRGH	PDSEL1	PDSEL0	STSEL
bit 7							bit 0

Legend:	C = Clearable bit	HC = Hardware Clearable bit	
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 **UARTEN:** UARTx Enable bit

1 = UARTx is enabled; all UARTx pins are controlled by UARTx as defined by UEN<1:0>
0 = UARTx is disabled; all UARTx pins are controlled by port latches, UARTx power consumption is minimal

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **USIDL:** UARTx Stop in Idle Mode bit

1 = Discontinues module operation when device enters Idle mode
0 = Continues module operation in Idle mode

bit 12 **IREN:** IrDA® Encoder and Decoder Enable bit[(1)]

1 = IrDA encoder and decoder are enabled
0 = IrDA encoder and decoder are disabled

bit 11 **RTSMD:** Mode Selection for $\overline{\text{UxRTS}}$ Pin bit

1 = $\overline{\text{UxRTS}}$ pin is in Simplex mode
0 = $\overline{\text{UxRTS}}$ pin is in Flow Control mode

bit 10 **Unimplemented:** Read as ‘0’

bit 9-8 **UEN<1:0>:** UARTx Enable bits[(2)]

11 = UxTX, UxRX and UxBCLK pins are enabled and used; $\overline{\text{UxCTS}}$ pin is controlled by port latches
10 = UxTX, UxRX, $\overline{\text{UxCTS}}$ and $\overline{\text{UxRTS}}$ pins are enabled and used
01 = UxTX, UxRX and $\overline{\text{UxRTS}}$ pins are enabled and used; $\overline{\text{UxCTS}}$ pin is controlled by port latches
00 = UxTX and UxRX pins are enabled and used; $\overline{\text{UxCTS}}$ and $\overline{\text{UxRTS}}$/UxBCLK pins are controlled by port latches

bit 7 **WAKE:** Wake-up on Start Bit Detect During Sleep Mode Enable bit

1 = UARTx will continue to sample the UxRX pin; interrupt is generated on the falling edge, bit is cleared in hardware on the following rising edge
0 = No wake-up is enabled

bit 6 **LPBACK:** UARTx Loopback Mode Select bit

1 = Enables Loopback mode
0 = Loopback mode is disabled

bit 5 **ABAUD:** Auto-Baud Enable bit

1 = Enables baud rate measurement on the next character – requires reception of a Sync field (55h); cleared in hardware upon completion
0 = Baud rate measurement is disabled or completed

bit 4 **RXINV:** Receive Polarity Inversion bit

1 = UxRX Idle state is ‘0’
0 = UxRX Idle state is ‘1’

Note 1: This feature is is only available for the 16x BRG mode (BRGH = 0).
2: Bit availability depends on pin availability.

REGISTER 18-2: UxSTA: UARTx STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	U-0	R/W-0, HC	R/W-0	R-0, HSC	R-1, HSC
UTXISEL1	UTXINV	UTXISEL0	—	UTXBRK	UTXEN	UTXBF	TRMT
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R-1, HSC	R-0, HSC	R-0, HSC	R/C-0, HS	R-0, HSC
URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA
bit 7							bit 0

Legend:	HC = Hardware Clearable bit		
HS = Hardware Settable bit	C = Clearable bit	HSC = Hardware Settable/Clearable bit	
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 15,13 **UTXISEL<1:0>:** UARTx Transmission Interrupt Mode Selection bits

11 = Reserved; do not use
10 = Interrupt when a character is transferred to the Transmit Shift Register (TSR), and as a result, the transmit buffer becomes empty
01 = Interrupt when the last character is shifted out of the Transmit Shift Register; all transmit operations are completed
00 = Interrupt when a character is transferred to the Transmit Shift Register (this implies there is at least one character open in the transmit buffer)

bit 14 **UTXINV:** IrDA® Encoder Transmit Polarity Inversion bit

If IREN = 0:
1 = UxTX Idle ‘0’
0 = UxTX Idle ‘1’

If IREN = 1:
1 = UxTX Idle ‘1’
0 = UxTX Idle ‘0’

bit 12 **Unimplemented:** Read as ‘0’

bit 11 **UTXBRK:** UARTx Transmit Break bit

1 = Sends Sync Break on next transmission – Start bit, followed by twelve ‘0’ bits; followed by Stop bit; cleared by hardware upon completion
0 = Sync Break transmission is disabled or completed

bit 10 **UTXEN:** UARTx Transmit Enable bit

1 = Transmit is enabled; UxTX pin is controlled by UARTx
0 = Transmit is disabled; any pending transmission is aborted and the buffer is reset. UxTX pin is controlled by the PORT register.

bit 9 **UTXBF:** UARTx Transmit Buffer Full Status bit (read-only)

1 = Transmit buffer is full
0 = Transmit buffer is not full, at least one more character can be written

bit 8 **TRMT:** Transmit Shift Register Empty bit (read-only)

1 = Transmit Shift Register is empty and the transmit buffer is empty (the last transmission has completed)
0 = Transmit Shift Register is not empty; a transmission is in progress or queued

bit 7-6 **URXISEL<1:0>:** UARTx Receive Interrupt Mode Selection bits

11 = Interrupt is set on the RSR transfer, making the receive buffer full (i.e., has 2 data characters)
10 = Reserved
01 = Reserved
00 = Interrupt is set when any character is received and transferred from the RSR to the receive buffer; receive buffer has one or more characters

REGISTER 19-2: AD1CON2: A/D CONTROL REGISTER 2

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0	U-0
VCFG2	VCFG1	VCFG0	OFFCAL[(1)]	—	CSCNA	—	—
bit 15							bit 8

R-x	U-0	R/W-0	R/W-0	R/W-0	R/W-0	r-0	R/W-0
r	—	SMPI3	SMPI2	SMPI1	SMPI0	r	ALTS
bit 7							bit 0

Legend:	r = Reserved bit		
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 15-13 **VCFG<2:0>:** Voltage Reference Configuration bits

VCFG<2:0>	VR+	VR-
000	AVDD	AVSS
001	External VREF+ pin	AVSS
010	AVDD	External VREF- pin
011	External VREF+ pin	External VREF- pin
1xx	AVDD	AVSS

bit 12 **OFFCAL:** Offset Calibration bit[(1)]
1 = Conversions to get the offset calibration value
0 = Conversions to get the actual input value

bit 11 **Unimplemented:** Read as ‘0’

bit 10 **CSCNA:** Scan Input Selections for MUX A Input Multiplexer bit
1 = Scans inputs
0 = Does not scan inputs

bit 9-8 **Unimplemented:** Read as ‘0’

bit 7 **Reserved:** Ignore this value

bit 6 **Unimplemented:** Read as ‘0’

bit 5-2 **SMPI<3:0>:** Sample/Convert Sequences Per Interrupt Selection bits
1111 =
•
• = Reserved, do not use (may cause conversion data loss)
•
0010 =
0001 = Interrupts at the completion of conversion for each 2nd sample/convert sequence
0000 = Interrupts at the completion of conversion for each sample/convert sequence

bit 1 **Reserved:** Always maintain as ‘0’

bit 0 **ALTS:** Alternate Input Sample Mode Select bit
1 = Uses MUX A input multiplexer settings for the first sample, then alternates between MUX B and MUX A input multiplexer settings for all subsequent samples
0 = Always uses MUX A input multiplexer settings

Note 1: When the OFFCAL bit is set, inputs are disconnected and tied to AVSS. This sets the inputs of the A/D to zero. Then, the user can perform a conversion. Use of the Calibration mode is not affected by AD1PCFG contents nor channel input selection. Any analog input switches are disconnected from the A/D Converter in this mode. The conversion result is stored by the user software and used to compensate subsequent conversions. This can be done by adding the two’s complement of the result obtained with the OFFCAL bit set to all normal A/D conversions.

20.0 COMPARATOR MODULE

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 module, refer to the *"dsPIC33/PIC24 Family Reference Manual"*, **"Dual Comparator Module"** (DS39710).

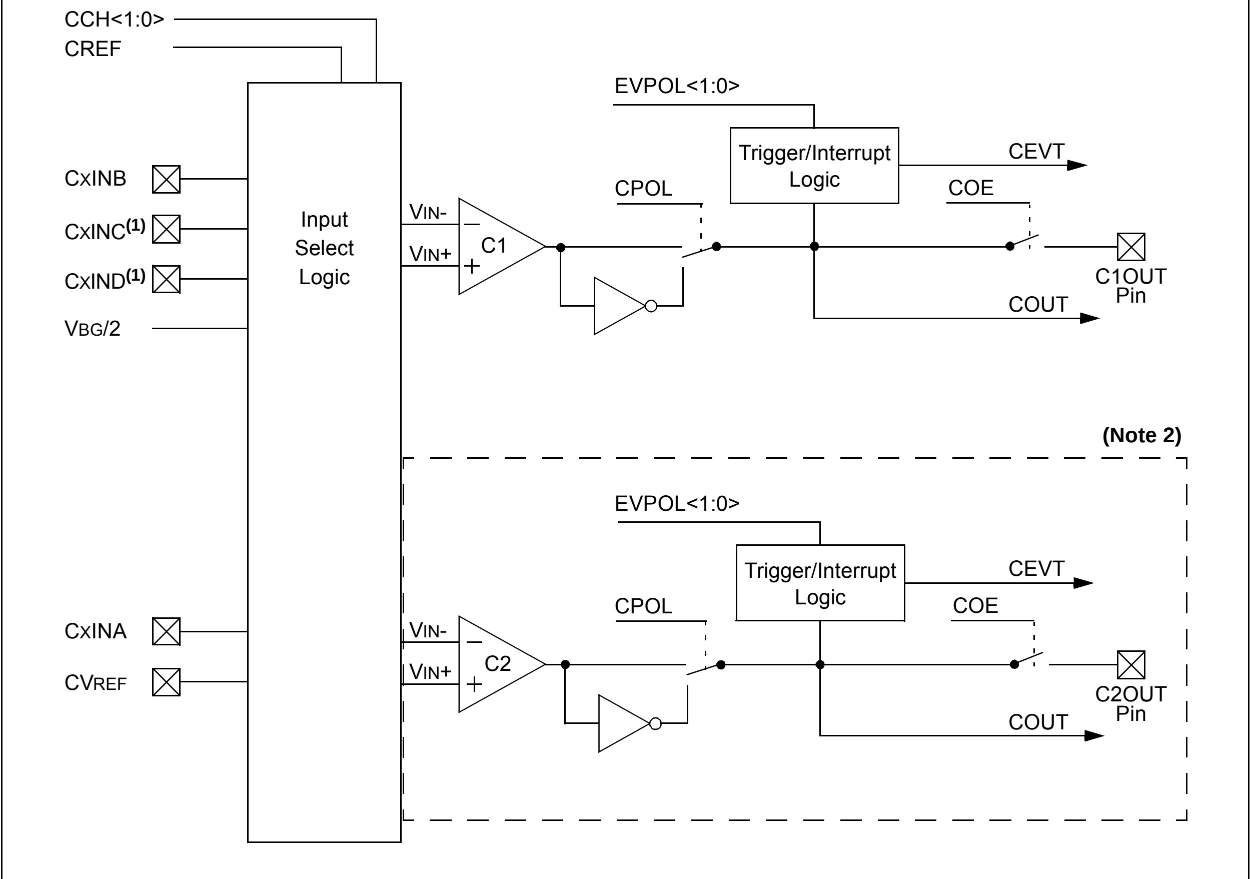
Depending on the particular device, the comparator module provides one or two analog comparators. The inputs to the comparator can be configured to use any one of up to four external analog inputs, as well as a voltage reference input from either the internal band gap reference, divided by 2 (VBG/2), or the comparator voltage reference generator.

The comparator outputs may be directly connected to the CxOUT pins. When the respective COE equals '1', the I/O pad logic makes the unsynchronized output of the comparator available on the pin.

A simplified block diagram of the module is displayed in Figure 20-1. Diagrams of the possible individual comparator configurations are displayed in Figure 20-2.

Each comparator has its own control register, CMxCON (Register 20-1), for enabling and configuring its operation. The output and event status of all three comparators is provided in the CMSTAT register (Register 20-2).

FIGURE 20-1: COMPARATOR MODULE BLOCK DIAGRAM



Note 1: These inputs are unavailable on 14-pin (PIC24FXXKL100/200) devices.
2: Comparator 2 is unimplemented on PIC24FXXKL10X/20X devices.

25.0 INSTRUCTION SET SUMMARY

Note:	This chapter is a brief summary of the PIC24F Instruction Set Architecture (ISA) and is not intended to be a comprehensive reference source.

The PIC24F instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from previous PIC MCU instruction sets. Most instructions are a single program memory word. Only three instructions require two program memory locations.

Each single-word instruction is a 24-bit word divided into an 8-bit opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction. The instruction set is highly orthogonal and is grouped into four basic categories:

- Word or byte-oriented operations
- Bit-oriented operations
- Literal operations
- Control operations

Table 25-1 lists the general symbols used in describing the instructions. The PIC24F instruction set summary in Table 25-2 lists all the instructions, along with the status flags affected by each instruction.

Most word or byte-oriented W register instructions (including barrel shift instructions) have three operands:

- The first source operand, which is typically a register 'Wb' without any address modifier
- The second source operand, which is typically a register 'Ws' with or without an address modifier
- The destination of the result, which is typically a register 'Wd' with or without an address modifier

However, word or byte-oriented file register instructions have two operands:

- The file register specified by the value, 'f'
- The destination, which could either be the file register, 'f', or the W0 register, which is denoted as 'WREG'

Most bit-oriented instructions (including simple rotate/shift instructions) have two operands:

- The W register (with or without an address modifier) or file register (specified by the value of 'Ws' or 'f')
- The bit in the W register or file register (specified by a literal value or indirectly by the contents of register, 'Wb')

The literal instructions that involve data movement may use some of the following operands:

- A literal value to be loaded into a W register or file register (specified by the value of 'k')
- The W register or file register where the literal value is to be loaded (specified by 'Wb' or 'f')

However, literal instructions that involve arithmetic or logical operations use some of the following operands:

- The first source operand, which is a register 'Wb' without any address modifier
- The second source operand, which is a literal value
- The destination of the result (only if not the same as the first source operand), which is typically a register 'Wd' with or without an address modifier

The control instructions may use some of the following operands:

- A program memory address
- The mode of the Table Read and Table Write instructions

All instructions are a single word, except for certain double-word instructions, which were made double-word instructions so that all of the required information is available in these 48 bits. In the second word, the 8 MSbs are '`0`'s. If this second word is executed as an instruction (by itself), it will execute as a `NOP`.

Most single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the Program Counter (PC) is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a `NOP`. Notable exceptions are the `BRA` (unconditional/computed branch), indirect `CALL/GOTO`, all Table Reads and Table Writes, and `RETURN/RETFIE` instructions, which are single-word instructions but take two or three cycles.

Certain instructions that involve skipping over the subsequent instruction require either two or three cycles if the skip is performed, depending on whether the instruction being skipped is a single-word or two-word instruction. Moreover, double-word moves require two cycles. The double-word instructions execute in two instruction cycles.

TABLE 25-2: INSTRUCTION SET OVERVIEW (CONTINUED)

Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles	Status Flags Affected
BTSS	BTSS f,#bit4	Bit Test f, Skip if Set	1	1 (2 or 3)	None
	BTSS Ws,#bit4	Bit Test Ws, Skip if Set	1	1 (2 or 3)	None
BTST	BTST f,#bit4	Bit Test f	1	1	Z
	BTST.C Ws,#bit4	Bit Test Ws to C	1	1	C
	BTST.Z Ws,#bit4	Bit Test Ws to Z	1	1	Z
	BTST.C Ws,Wb	Bit Test Ws<Wb> to C	1	1	C
	BTST.Z Ws,Wb	Bit Test Ws<Wb> to Z	1	1	Z
BTSTS	BTSTS f,#bit4	Bit Test then Set f	1	1	Z
	BTSTS.C Ws,#bit4	Bit Test Ws to C, then Set	1	1	C
	BTSTS.Z Ws,#bit4	Bit Test Ws to Z, then Set	1	1	Z
CALL	CALL lit23	Call Subroutine	2	2	None
	CALL Wn	Call Indirect Subroutine	1	2	None
CLR	CLR f	f = 0x0000	1	1	None
	CLR WREG	WREG = 0x0000	1	1	None
	CLR Ws	Ws = 0x0000	1	1	None
CLRWDT	CLRWDT	Clear Watchdog Timer	1	1	WDTO, Sleep
COM	COM f	f = $\overline{f}$	1	1	N, Z
	COM f,WREG	WREG = $\overline{f}$	1	1	N, Z
	COM Ws,Wd	Wd = $\overline{Ws}$	1	1	N, Z
CP	CP f	Compare f with WREG	1	1	C, DC, N, OV, Z
	CP Wb,#lit5	Compare Wb with lit5	1	1	C, DC, N, OV, Z
	CP Wb,Ws	Compare Wb with Ws (Wb – Ws)	1	1	C, DC, N, OV, Z
CP0	CP0 f	Compare f with 0x0000	1	1	C, DC, N, OV, Z
	CP0 Ws	Compare Ws with 0x0000	1	1	C, DC, N, OV, Z
CPB	CPB f	Compare f with WREG, with Borrow	1	1	C, DC, N, OV, Z
	CPB Wb,#lit5	Compare Wb with lit5, with Borrow	1	1	C, DC, N, OV, Z
	CPB Wb,Ws	Compare Wb with Ws, with Borrow (Wb – Ws – $\overline{C}$)	1	1	C, DC, N, OV, Z
CPSEQ	CPSEQ Wb,Wn	Compare Wb with Wn, Skip if =	1	1 (2 or 3)	None
CPSGT	CPSGT Wb,Wn	Compare Wb with Wn, Skip if >	1	1 (2 or 3)	None
CPSLT	CPSLT Wb,Wn	Compare Wb with Wn, Skip if <	1	1 (2 or 3)	None
CPSNE	CPSNE Wb,Wn	Compare Wb with Wn, Skip if ≠	1	1 (2 or 3)	None
DAW	DAW.B Wn	Wn = Decimal Adjust Wn	1	1	C
DEC	DEC f	f = f –1	1	1	C, DC, N, OV, Z
	DEC f,WREG	WREG = f –1	1	1	C, DC, N, OV, Z
	DEC Ws,Wd	Wd = Ws – 1	1	1	C, DC, N, OV, Z
DEC2	DEC2 f	f = f – 2	1	1	C, DC, N, OV, Z
	DEC2 f,WREG	WREG = f – 2	1	1	C, DC, N, OV, Z
	DEC2 Ws,Wd	Wd = Ws – 2	1	1	C, DC, N, OV, Z
DISI	DISI #lit14	Disable Interrupts for k Instruction Cycles	1	1	None
DIV	DIV.SW Wm,Wn	Signed 16/16-bit Integer Divide	1	18	N, Z, C, OV
	DIV.SD Wm,Wn	Signed 32/16-bit Integer Divide	1	18	N, Z, C, OV
	DIV.UW Wm,Wn	Unsigned 16/16-bit Integer Divide	1	18	N, Z, C, OV
	DIV.UD Wm,Wn	Unsigned 32/16-bit Integer Divide	1	18	N, Z, C, OV
EXCH	EXCH Wns,Wnd	Swap Wns with Wnd	1	1	None
FF1L	FF1L Ws,Wnd	Find First One from Left (MSb) Side	1	1	C
FF1R	FF1R Ws,Wnd	Find First One from Right (LSb) Side	1	1	C

NOTES:

TABLE 26-6: DC CHARACTERISTICS: OPERATING CURRENT (IDD)(2)

DC CHARACTERISTICS			Standard Operating Conditions: 1.8V to 3.6V Operating temperature -40°C ≤ TA ≤ +85°C for Industrial -40°C ≤ TA ≤ +125°C for Extended			
Parameter No.	**Typical(1)**	**Max**	**Units**	**Conditions**		
IDD Current						
DC20	0.154	0.350	mA	1.8V	+85V°C	0.5 MIPS, FOSC = 1 MHz
	0.301	0.630		3.3V		
	—	.500	mA	1.8V	+125°C	
	—	.800		3.3V		
DC22	0.300	—	mA	1.8V	+85°C	1 MIPS, FOSC = 2 MHz
	0.585	—		3.3V		
DC24	7.76	12.0	mA	3.3V	+85°C	16 MIPS, FOSC = 32 MHz
	—	18.0		3.3V	+125°C	
DC26	1.44	—	mA	1.8V	+85°C	FRC (4 MIPS), FOSC = 8 MHz
	2.71	—		3.3V		
DC30	4.00	28.0	μA	1.8V	+85°C	LPRC (15.5 KIPS), FOSC = 31 kHz
	9.00	55.0		3.3V		
	—	45.0	μA	1.8V	+125°C	
	—	90.0		3.3V		

Note 1: Data in the Typical column is at 3.3V, +25°C, unless otherwise stated.

2: IDD is measured with all peripherals disabled. All I/Os are configured as outputs and set low; PMDx bits are set to '1' and WDT, etc., are all disabled.

TABLE 26-7: DC CHARACTERISTICS: IDLE CURRENT (IIDLE)(2)

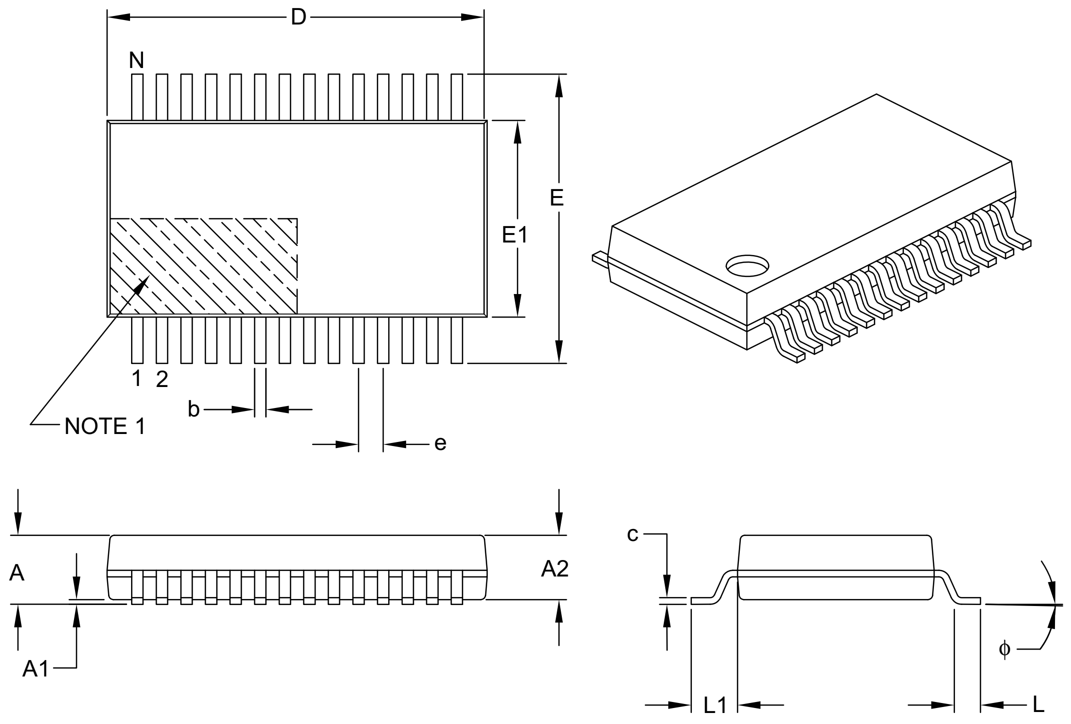
DC CHARACTERISTICS			Standard Operating Conditions: 1.8V to 3.6V Operating temperature -40°C ≤ TA ≤ +85°C for Industrial -40°C ≤ TA ≤ +125°C for Extended			
Parameter No.	**Typical(1)**	**Max**	**Units**	**Conditions**		
Idle Current (IIDLE)						
DC40	0.035	0.080	mA	1.8V	+85°C	0.5 MIPS, FOSC = 1 MHz
	0.077	0.150		3.3V		
	—	0.160	mA	1.8V	+125°C	
	—	0.300		3.3V		
DC42	0.076	—	mA	1.8V	+85°C	1 MIPS, FOSC = 2 MHz
	0.146	—		3.3V		
DC44	2.52	3.20	mA	3.3V	+85°C	16 MIPS, FOSC = 32 MHz
	—	5.00	mA	3.3V	+125°C	
DC46	0.45	—	mA	1.8V	+85°C	FRC (4 MIPS), FOSC = 8 MHz
	0.76	—	mA	3.3V		
DC50	0.87	18.0	μA	1.8V	+85°C	LPRC (15.5 KIPS), FOSC = 31 kHz
	1.55	40.0	μA	3.3V		
	—	27.0	μA	1.8V	+125°C	
	—	50.0	μA	3.3V		

Note 1: Data in the Typical column is at 3.3V, +25°C, unless otherwise stated.

2: IIDLE is measured with all I/Os configured as outputs and set low; PMDx bits are set to '1' and WDT, etc., are all disabled.

28-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

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	28		
Pitch	e	0.65 BSC		
Overall Height	A	–	–	2.00
Molded Package Thickness	A2	1.65	1.75	1.85
Standoff	A1	0.05	–	–
Overall Width	E	7.40	7.80	8.20
Molded Package Width	E1	5.00	5.30	5.60
Overall Length	D	9.90	10.20	10.50
Foot Length	L	0.55	0.75	0.95
Footprint	L1	1.25 REF		
Lead Thickness	c	0.09	–	0.25
Foot Angle	ϕ	0°	4°	8°
Lead Width	b	0.22	–	0.38

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 0.20 mm 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 C04-073B