Microchip Technology - PIC24F08KA101-E/P Datasheet

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, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, 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	1.5K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 9x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Through Hole
Package / Case	20-DIP (0.300", 7.62mm)
Supplier Device Package	20-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic24f08ka101-e-p

The internal oscillator block also provides a stable reference source for the Fail-Safe Clock Monitor (FSCM). This option constantly monitors the main clock source against a reference signal provided by the internal oscillator and enables the controller to switch to the internal oscillator, allowing for continued low-speed operation or a safe application shutdown.

1.1.4 EASY MIGRATION

Regardless of the memory size, all the devices share the same rich set of peripherals, allowing for a smooth migration path as applications grow and evolve.

The consistent pinout scheme used throughout the entire family also helps in migrating to the next larger device. This is true when moving between devices with the same pin count, or even jumping from 20-pin to 28-pin devices.

The PIC24F family is pin compatible with devices in the dsPIC33 family, and shares some compatibility with the pinout schema for PIC18 and dsPIC30. This extends the ability of applications to grow from the relatively simple, to the powerful and complex.

1.2 Other Special Features

Communications: The PIC24F16KA102 family incorporates a range of serial communication peripherals to handle a range of application requirements. There is an I^{2}C module that supports both the Master and Slave modes of operation. It also comprises UARTs with built-in IrDA^{ﬁ} encoders/decoders and an SPI module.

Real-Time Clock/Calendar: This module implements a full-featured clock and calendar with alarm functions in hardware, freeing up timer resources and program memory space for use of the core application.

10-Bit A/D Converter: This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period, and faster sampling speed. The 16-deep result buffer can be used either in Sleep to reduce power, or in Active mode to improve throughput.

Charge Time Measurement Unit (CTMU) Interface: The PIC24F16KA102 family includes the new CTMU interface module, which can be used for capacitive touch sensing, proximity sensing and also for precision time measurement and pulse generation.

1.3 Details on Individual Family Members

Devices in the PIC24F16KA102 family are available in 20-pin and 28-pin packages. The general block diagram for all devices is displayed in Figure 1-1.

The devices are different from each other in two ways:

1. Flash program memory (8 Kbytes for PIC24F08KA devices, 16 Kbytes for PIC24F16KA devices).
2. Available I/O pins and ports (18 pins on two ports for 20-pin devices and 24 pins on two ports for 28-pin devices).
3. Alternate SCLx and SDAx pins are available only in 28-pin devices and not in 20-pin devices.

All other features for devices in this family are identical; these are summarized in Table 1-1.

A list of the pin features available on the PIC24F16KA102 family devices, sorted by function, is provided in Table 1-2.

Note:	Table 1-1 provides the pin location of individual peripheral features and not how they are multiplexed on the same pin. This information is provided in the pinout diagrams on pages 4, 5 and 6 of the data sheet. Multiplexed features are sorted by the priority given to a feature, with the highest priority peripheral being listed first.

4.0 MEMORY ORGANIZATION

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

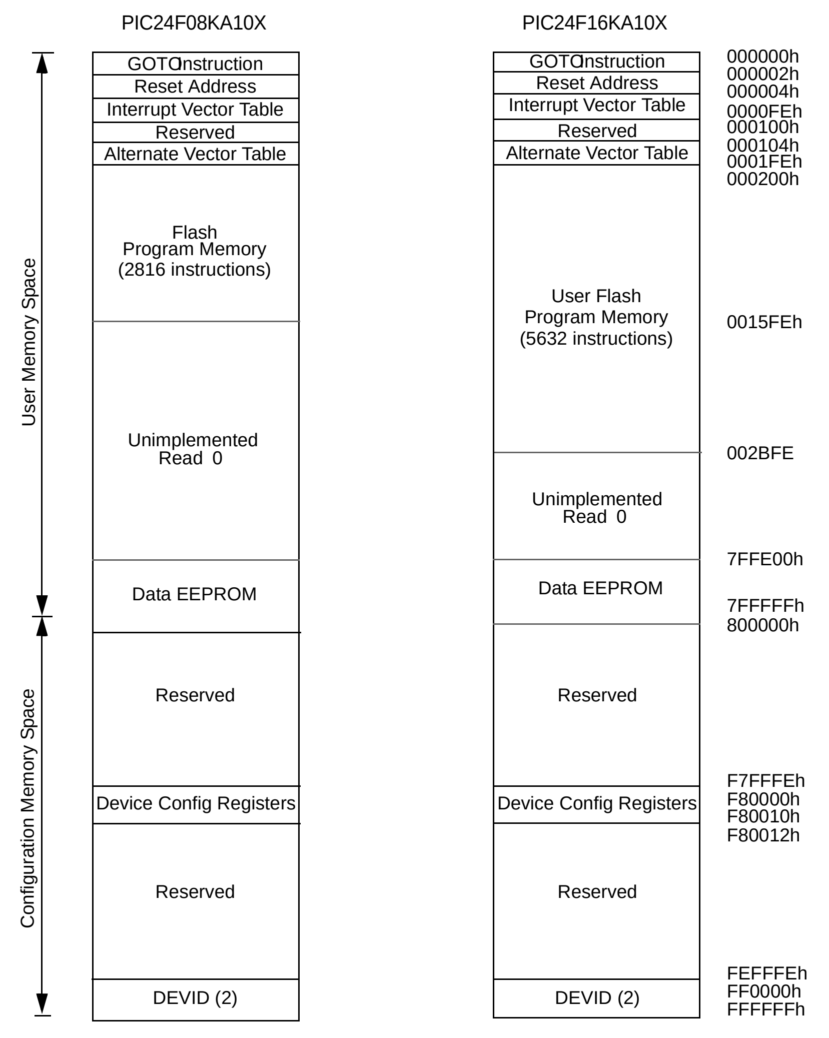
4.1 Program Address Space

The program address memory space of the PIC24F devices 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” .

The 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/TBLWToperations, which use TBLPAG<7> to permit access to the Configuration bits and Device ID sections of the configuration memory space.

Memory maps for the PIC24F16KA102 family of devices are displayed in Figure 4-1.

FIGURE 4-1: PROGRAM SPACE MEMORY MA P FOR PIC24F16KA102 FAMILY DEVICES



Note: Memory areas are not displayed to scale.

TABLE 4-3: CPU CORE REGISTERS MAP

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
WREG0	0000	Working Register 0																0000
WREG1	0002	Working Register 1																0000
WREG2	0004	Working Register 2																0000
WREG3	0006	Working Register 3																0000
WREG4	0008	Working Register 4																0000
WREG5	000A	Working Register 5																0000
WREG6	000C	Working Register 6																0000
WREG7	000E	Working Register 7																0000
WREG8	0010	Working Register 8																0000
WREG9	0012	Working Register 9																0000
WREG10	0014	Working Register 10																0000
WREG11	0016	Working Register 11																0000
WREG12	0018	Working Register 12																0000
WREG13	001A	Working Register 13																0000
WREG14	001C	Working Register 14																0000
WREG15	001E	Working Register 15																0800
SPLIM	0020	Stack Pointer Limit Value Register																xxxx
PCL	002E	Program Counter Low Byte Register																0000
PCH	0030	▨	▨	▨	▨	▨	▨	▨	▨	Program Counter Register High Byte								0000
TBLPAG	0032	▨	▨	▨	▨	▨	▨	▨	▨	Table Memory Page Address Register								0000
PSVPAG	0034	▨	▨	▨	▨	▨	▨	▨	▨	Program Space Visibility Page Address Register								0000
RCOUNT	0036	REPEAT Loop Counter Register																xxxx
SR	0042	▨	▨	▨	▨	▨	▨	▨	DC	IPL2	IPL1	IPL0	RA	N	OV	Z	C	0000
CORCON	0044	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	IPL3	PSV	▨	▨	0000
DISICNT	0052	▨	▨	Disable Interrupts Counter Register														xxxx

Legend: = unimplemented, read as 0 . Reset values are shown in hexadecimal.

TABLE 4-15: A/D REGISTER MAP

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
ADC1BUF0	0300	A/D Data Buffer 0																xxxx
ADC1BUF1	0302	A/D Data Buffer 1																xxxx
ADC1BUF2	0304	A/D Data Buffer 2																xxxx
ADC1BUF3	0306	A/D Data Buffer 3																xxxx
ADC1BUF4	0308	A/D Data Buffer 4																xxxx
ADC1BUF5	030A	A/D Data Buffer 5																xxxx
ADC1BUF6	030C	A/D Data Buffer 6																xxxx
ADC1BUF7	030E	A/D Data Buffer 7																xxxx
ADC1BUF8	0310	A/D Data Buffer 8																xxxx
ADC1BUF9	0312	A/D Data Buffer 9																xxxx
ADC1BUFA	0314	A/D Data Buffer 10																xxxx
ADC1BUFB	0316	A/D Data Buffer 11																xxxx
ADC1BUFC	0318	A/D Data Buffer 12																xxxx
ADC1BUFD	031A	A/D Data Buffer 13																xxxx
ADC1BUFE	031C	A/D Data Buffer 14																xxxx
ADC1BUFF	031E	A/D Data Buffer 15																xxxx
AD1CON1	0320	ADON	▨	ADSIDL	▨	▨	▨	FORM1	FORM0	SSRC2	SSRC1	SSRC0	▨	▨	ASAM	SAMP	DONE	0000
AD1CON2	0322	VCFG2	VCFG1	VCFG0	OFFCAL	▨	CSCNA	▨	▨	BUFS	▨	SMPI3	SMPI2	SMPI1	SMPI0	BUFM	ALTS	0000
AD1CON3	0324	ADRC	▨	▨	SAMC4	SAMC3	SAMC2	SAMC1	SAMC0	▨	▨	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0	0000
AD1CHS	0328	CH0NB	▨	▨	▨	CH0SB3	CH0SB2	CH0SB1	CH0SB0	CH0NA	▨	▨	CH0SA4	CH0SA3	CH0SA2	CH0SA1	CH0SA0	0000
AD1PCFG	032C	▨	▨	▨	PCFG12	PCFG11	PCFG10	▨	▨	▨	▨	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000
AD1CSSL	0330	▨	▨	▨	CSSL12	CSSL11	CSSL10	▨	▨	▨	▨	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0	0000

Legend: = unimplemented, read as 0 . Reset values are shown in hexadecimal.

TABLE 4-16: CTMU REGISTER MAP

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
CTMUCON	033C	CTMUEN	▨	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	CTTRIG	EDG2POL	EDG2SEL1	EDG2SEL0	EDG1POL	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT	0000
CTMUICON	033E	ITRIM5	ITRIM4	ITRIM3	ITRIM2	ITRIM1	ITRIM0	IRNG1	IRNG0	▨	▨	▨	▨	▨	▨	▨	▨	0000

Legend: = unimplemented, read as 0 . Reset values are shown in hexadecimal.

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 2 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 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 .

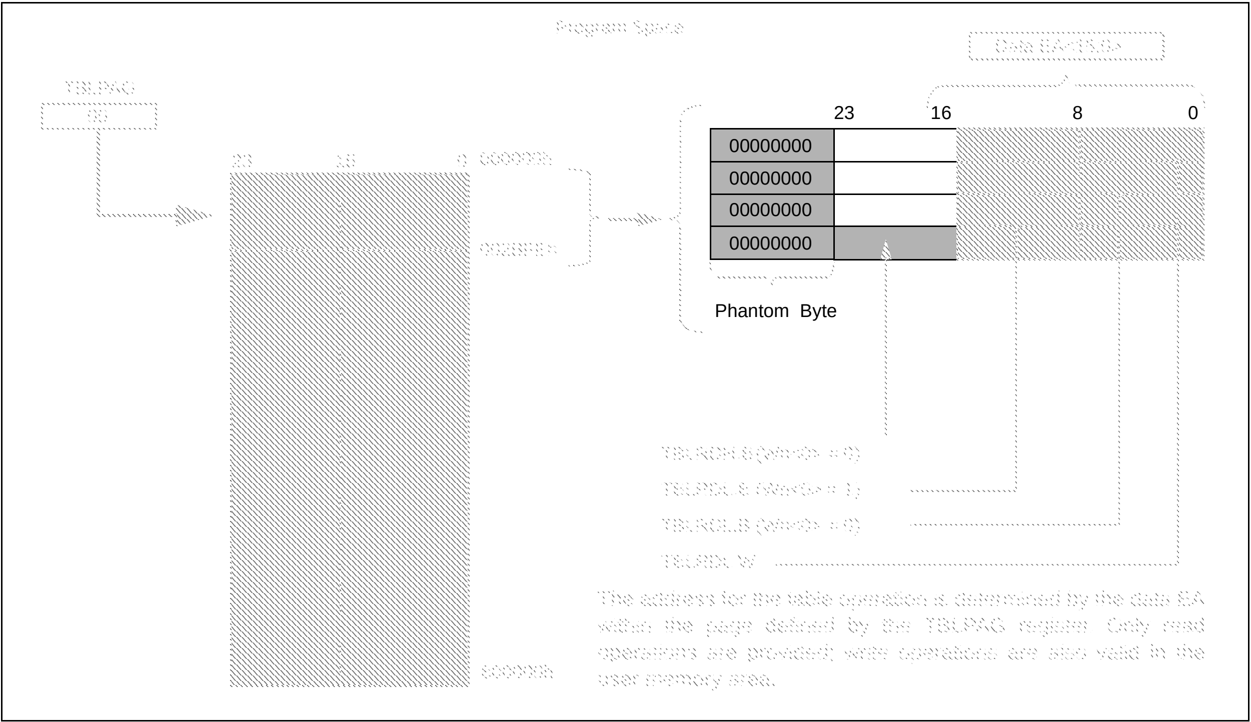
   In Byte mode, it maps the upper or lower byte of the program word to D<7:0> of the data address, as above. Note that the data will always be 0 when the upper phantom byte is selected (byte select = 1).

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


Program Space
Data EA<15:0>
TBLPAG
00
23
15
0
000000h
800000h
23
16
8
0
00000000
00000000
00000000
00000000
Phantom Byte
TBLRDH.B (Wn<0> = 0)
TBLRDL.B (Wn<0> = 1)
TBLRDL.B (Wn<0> = 0)
TBLRDL.W
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.

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 bulk erase:

1. Configure NVMCON to Bulk Erase mode.
2. Clear NVMIF status bit and enable NVM interrupt (optional).
3. Write the key sequence to NVMKEY.
4. Set the WR bit to begin 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 the 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 NVMIF status bit and enable NVM interrupt (optional).
   - Write the key sequence to NVMKEY.
   - Set the WR bit to begin erase cycle.
   - Either poll the WR bit or wait for the NVM interrupt (NVMIF is 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;    // Variable located in EEPROM,declared as a
                                                                  global variable.
   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
```

REGISTER 8-3: INTCON1: INTERRUPT CONTROL REGISTER 1

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
NSTDIS							
bit 15							bit 8

U-0	U-0	U-0	R/W-0, HS	R/W-0, HS	R/W-0, HS	R/W-0, HS	U-0
			MATHERR	ADDRERR	STKERR	OSCFAIL	
bit 7							bit 0

Legend:	HS = Hardware Settable 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	NSTDIS: Interrupt Nesting Disable bit 1 = Interrupt nesting is disabled 0 = Interrupt nesting is enabled
bit 14-5	Unimplemented: Read as 0
bit 4	MATHERR: Arithmetic Error Trap Status bit 1 = Overflow trap has occurred 0 = Overflow trap has not occurred
bit 3	ADDRERR: Address Error Trap Status bit 1 = Address error trap has occurred 0 = Address error trap has not occurred
bit 2	STKERR: Stack Error Trap Status bit 1 = Stack error trap has occurred 0 = Stack error trap has not occurred
bit 1	OSCFAIL: Oscillator Failure Trap Status bit 1 = Oscillator failure trap has occurred 0 = Oscillator failure trap has not occurred
bit 0	Unimplemented: Read as 0

9.0 OSCILLATOR CONFIGURATION

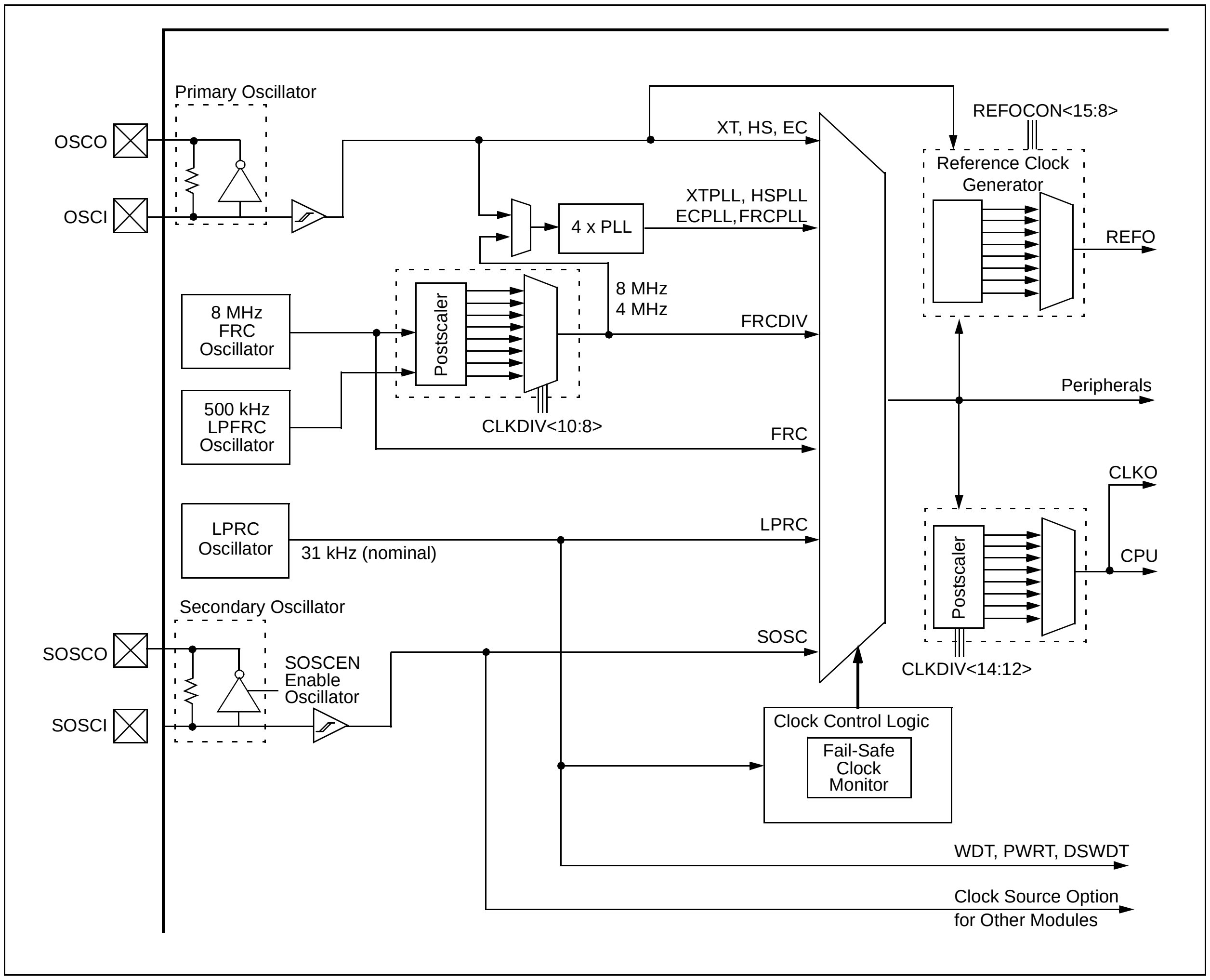
Note:	This data sheet summarizes the features of this group of PIC24F devices. It is not intended to be a comprehensive reference source. For more information on Oscillator Configuration, refer to the *“PIC24F Family Reference Manual”*, Section 38. “Oscillator with 500 kHz Low-Power FRC” (DS39726).

The oscillator system for the PIC24F16KA102 family of devices has the following features:

- A total of five external and internal oscillator options as clock sources, providing 11 different clock modes.
- On-chip 4x Phase Locked Loop (PLL) to boost internal operating frequency on select internal and external oscillator sources.
- Software-controllable switching between various clock sources.
- Software-controllable postscaler for selective clocking of CPU for system power savings.
- System frequency range declaration bits for EC mode. When using an external clock source, the current consumption is reduced by setting the declaration bits to the expected frequency range.
- A Fail-Safe Clock Monitor (FSCM) that detects clock failure and permits safe application recovery or shutdown.

Figure 9-1 provides a simplified diagram of the oscillator system.

FIGURE 9-1: PIC24F16KA102 FAMILY CLOCK DIAGRAM



9.1 CPU Clocking Scheme

The system clock source can be provided by one of four sources:

- Primary Oscillator (POSC) on the OSCI and OSCO pins
- Secondary Oscillator (SOSC) on the SOSCI and SOSCO pins

  The PIC24F16KA102 family devices consist of two types of secondary oscillator:
  - High-Power Secondary Oscillator
  - Low-Power Secondary Oscillator

  These can be selected by using the SOSCSEL (FOSC<5>) bit.
- Fast Internal RC (FRC) Oscillator
  - 8 MHz FRC Oscillator
  - 500 kHz Lower Power FRC Oscillator
- Low-Power Internal RC (LPRC) Oscillator

The primary oscillator and 8 MHz FRC sources have the option of using the internal 4x PLL. The frequency of the FRC clock source can optionally be reduced by the programmable clock divider. The selected clock source generates the processor and peripheral clock sources.

The processor clock source is divided by two to produce the internal instruction cycle clock, FCY. In this document, the instruction cycle clock is also denoted by FOSC/2. The internal instruction cycle clock, FOSC/2, can be provided on the OSCO I/O pin for some operating modes of the primary oscillator.

9.2 Initial Configuration on POR

The oscillator source (and operating mode) that is used at a device Power-on Reset (POR) event is selected using Configuration bit settings. The Oscillator Configuration bit settings are located in the Configuration registers in the program memory (refer to Section 26.1 “Configuration Bits” for further details). The Primary Oscillator Configuration bits, POSCMD<1:0> (FOSC<1:0>), and the Initial Oscillator Select Configuration bits, FNOSC<2:0> (FOSCSEL<2:0>), select the oscillator source that is used at a POR. The FRC Primary Oscillator with Postscaler (FRCDIV) is the default (unprogrammed) selection. The secondary oscillator, or one of the internal oscillators, may be chosen by programming these bit locations. The EC mode frequency range Configuration bits, POSCFREQ<1:0> (FOSC<4:3>), optimize power consumption when running in EC mode. The default configuration is frequency range is greater than 8 MHz .

The Configuration bits allow users to choose between the various clock modes, shown in Table 9-1.

9.2.1 CLOCK SWITCHING MODE CONFIGURATION BITS

The FCKSM Configuration bits (FOSC<7:6>) are used jointly to configure device clock switching and the FSCM. Clock switching is enabled only when FCKSM1 is programmed (0). The FSCM is enabled only when FCKSM<1:0> are both programmed (00).

TABLE 9-1: CONFIGURATION BIT VA LUES FOR CLOCK SELECTION

Oscillator Mode	Oscillator Source	POSCMD<1:0>	FNOSC<2:0>	Note
8 MHz FRC Oscillator with Postscaler (FRCDIV)	Internal	11	111	1, 2
500 MHz FRC Oscillator with Postscaler (LPFRCDIV)	Internal	11	110	1
Low-Power RC Oscillator (LPRC)	Internal	11	101	1
Secondary (Timer1) Oscillator (SOSC)	Secondary	00	100	1
Primary Oscillator (HS) with PLL Module (HSPLL)	Primary	10	011	
Primary Oscillator (EC) with PLL Module (ECPLL)	Primary	00	011	
Primary Oscillator (HS)	Primary	10	010	
Primary Oscillator (XT)	Primary	01	010	
Primary Oscillator (EC)	Primary	00	010	
8 MHz FRC Oscillator with PLL Module (FRCPLL)	Internal	11	001	1
8 MHz FRC Oscillator (FRC)	Internal	11	000	1

Note 1: OSCO pin function is determined by the OSCIOFNC Configuration bit.

2: This is the default oscillator mode for an unprogrammed (erased) device.

10.3 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 while 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.

10.4 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 the PMD bits are used. Most peripheral modules have an enable bit; exceptions include capture, compare and RTCC.

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, enhancing power savings for extremely critical power applications.

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 *"PIC24F Family Reference Manual"*, Section 12. "I/O Ports with Peripheral Pin Select (PPS)" (DS39711). Note that the PIC24F16KA102 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. The logic also prevents loop through , in which a port s digital output can drive the input of a peripheral that shares the same pin. Figure 11-1 displays 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.

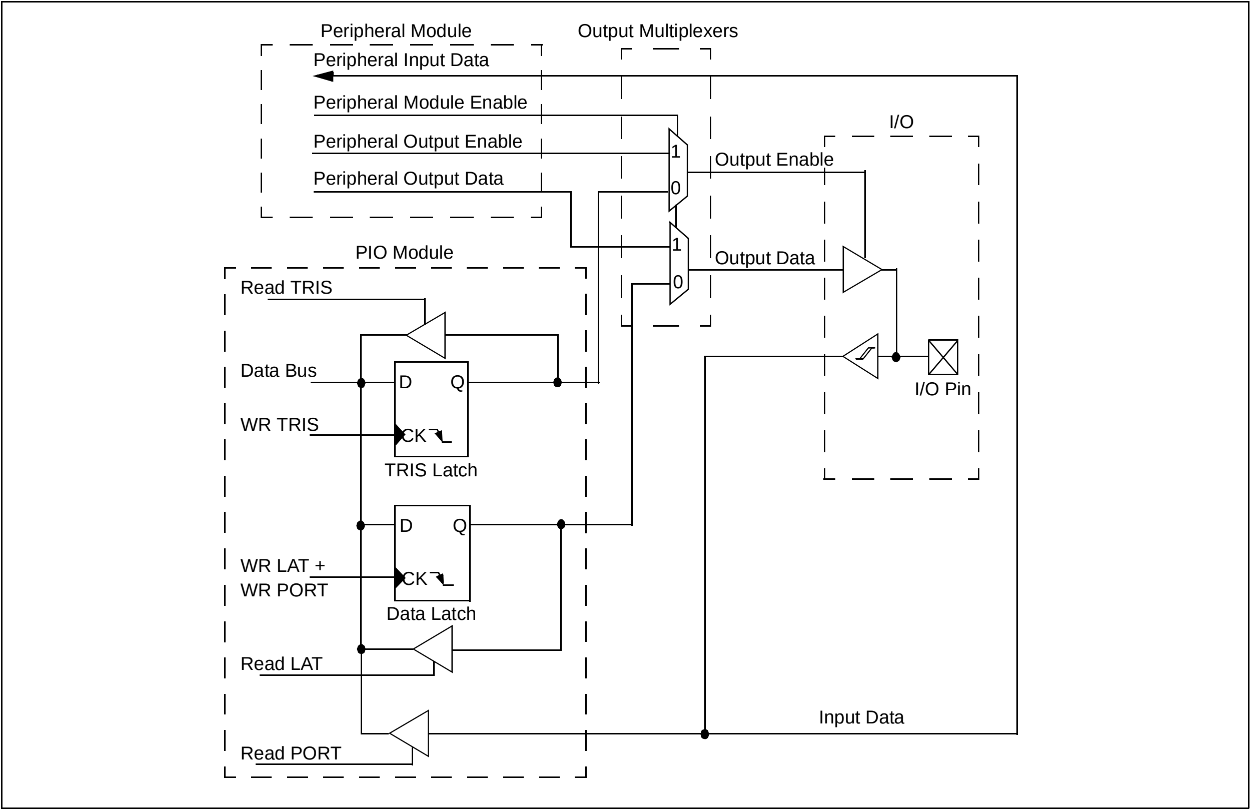
All port pins have three registers directly associated with their operation as digital I/O. The Data Direction register (TRISx) determines whether the pin is an input or an output. If the data direction bit is a 1 , then the pin is an input. All port pins are defined as inputs after a Reset. Reads from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

Note: The I/O pins retain their state during Deep Sleep. They will retain this state at wake-up until the software restore bit (RELEASE) is cleared.

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



REGISTER 17-1: I2C1CON: I2C1 CONTROL REGISTER

R/W-0	U-0	R/W-0	R/W-1 HC	R/W-0	R/W-0	R/W-0	R/W-0
I2CEN		I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEN
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0, HC	R/W-0, HC	R/W-0, HC	R/W-0, HC	R/W-0, HC
GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

Legend:	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 I2CEN: I2C1 Enable bit
1 = Enables the I2C1 module and configures the SDA1 and SCL1 pins as serial port pins
0 = Disables the I2C1 module; all I^{2}C pins are controlled by port functions

bit 14 Unimplemented: Read as 0

bit 13 I2CSIDL: Stop in Idle Mode bit
1 = Discontinues module operation when device enters an Idle mode
0 = Continues module operation in Idle mode

bit 12 SCLREL: SCL1 Release Control bit (when operating as I^{2}C slave)
1 = Releases SCL1 clock
0 = Holds SCL1 clock low (clock stretch)

If STREN = 1:
Bit is R/$\overline{W}$ (i.e., software may write 0 to initiate stretch and write 1 to release clock). Hardware is clear at the beginning of slave transmission; hardware is clear at the end of slave reception.

If STREN = 0:
Bit is R/S (i.e., software may only write 1 to release clock). Hardware is clear at the beginning of slave transmission.

bit 11 IPMIEN: Intelligent Peripheral Management Interface (IPMI) Enable bit
1 = IPMI Support mode is enabled; all addresses are Acknowledged
0 = IPMI Support mode is disabled

bit 10 A10M: 10-Bit Slave Addressing bit
1 = I2C1ADD is a 10-bit slave address
0 = I2C1ADD is a 7-bit slave address

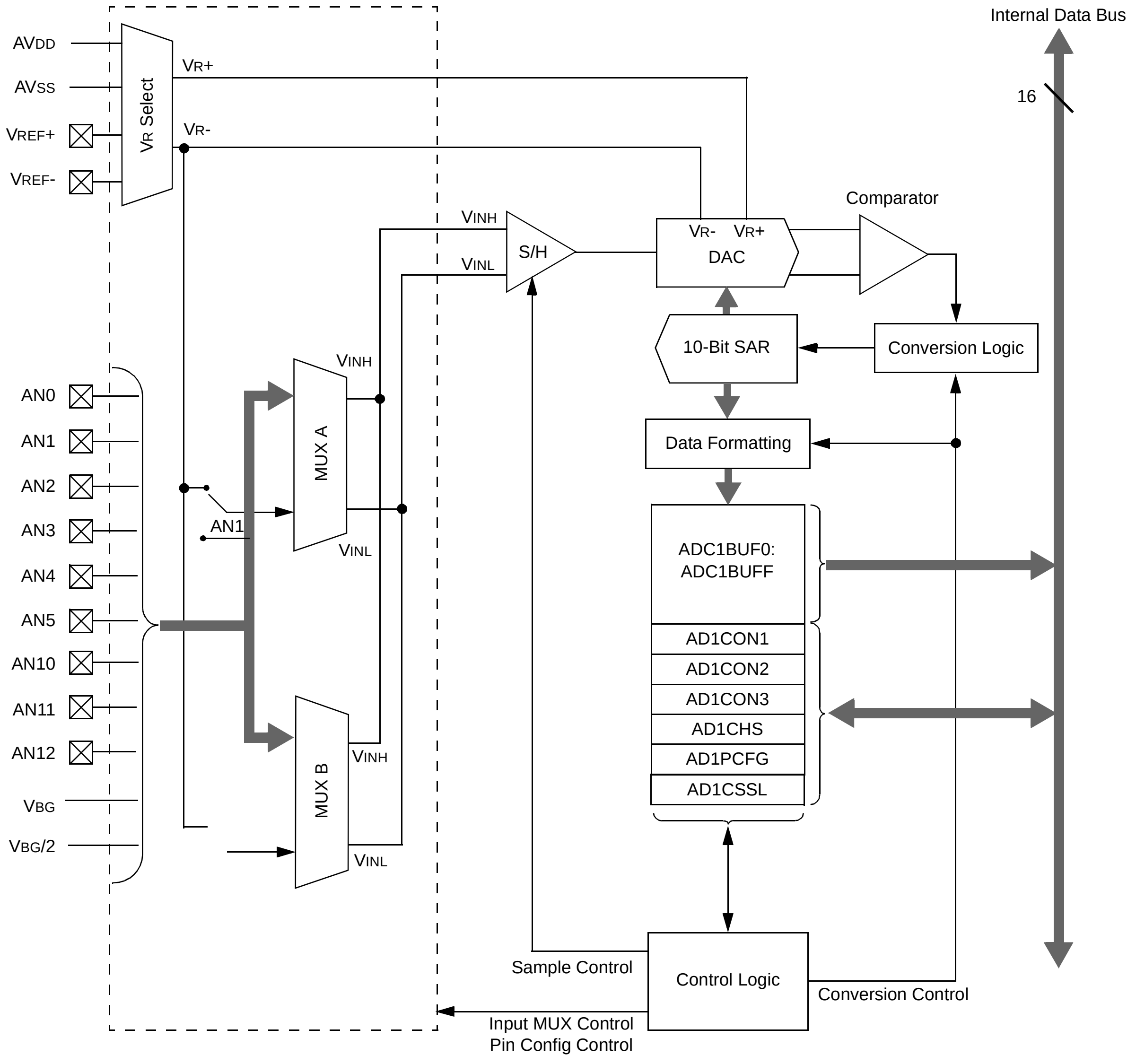
bit 9 DISSLW: Disable Slew Rate Control bit
1 = Slew rate control is disabled
0 = Slew rate control is enabled

bit 8 SMEN: SMBus Input Levels bit
1 = Enables I/O pin thresholds compliant with the SMBus specification
0 = Disables the SMBus input thresholds

bit 7 GCEN: General Call Enable bit (when operating as I^{2}C slave)
1 = Enables interrupt when a general call address is received in the I2C1RSR (module is enabled for reception)
0 = General call address is disabled

bit 6 STREN: SCL1 Clock Stretch Enable bit (when operating as I^{2}C slave)
Used in conjunction with the SCLREL bit.
1 = Enables software or receive clock stretching
0 = Disables software or receive clock stretching

FIGURE 22-1: 10-BIT HIGH-SPEED A/D CONVERTER BLOCK DIAGRAM



REGISTER 22-6: AD1CSSL: A/D INPUT SCAN SELECT REGISTER (LOW)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	U-0	U-0
—	—	—	CSSL12	CSSL11	CSSL10	—	—
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
—	—	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0
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 15-13	Unimplemented: Read as 0
bit 12-10	CSSL<12:10>: A/D Input Pin Scan Selection bits 1 = Corresponding analog channel is selected for input scan 0 = Analog channel omitted from input scan
bit 9-6	Unimplemented: Read as 0
bit 5-0	CSSL<5:0>: A/D Input Pin Scan Selection bits 1 = Corresponding analog channel is selected for input scan 0 = Analog channel omitted from input scan

REGISTER 26-6: FPOR: RESET CONFIGURATION REGISTER

R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	U-0	R/P-1	R/P-1
MCLRE(2)	BORV1(3)	BORV0(3)	I2C1SEL(1)	PWRTEN	—	BOREN1	BOREN0
bit 7							bit 0

Legend:			
R = Readable bit	P = Programmable 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	MCLRE: $\overline{\text{MCLR}}$ Pin Enable bit(2) 1 = $\overline{\text{MCLR}}$ pin is enabled; RA5 input pin is disabled 0 = RA5 input pin is enabled; $\overline{\text{MCLR}}$ is disabled
bit 6-5	BORV<1:0>: Brown-out Reset Enable bits(3) 11 = Brown-out Reset is set to the lowest voltage 10 = Brown-out Reset 01 = Brown-out Reset is set to the highest voltage 00 = Low-Power Brown-out Reset occurs around 2.0V
bit 4	I2C1SEL: Alternate I2C1 Pin Mapping bit(1) 0 = Alternate location for SCL1/SDA1 pins 1 = Default location for SCL1/SDA1 pins
bit 3	PWRTEN: Power-up Timer Enable bit 0 = PWRT is disabled 1 = PWRT is enabled
bit 2	Unimplemented: Read as 0
bit 1-0	BOREN<1:0>: Brown-out Reset Enable bits 11 = Brown-out Reset is enabled in hardware; SBOREN bit is disabled 10 = Brown-out Reset is enabled only while device is active and disabled in Sleep; SBOREN bit is disabled 01 = Brown-out Reset is controlled with the SBOREN bit setting 00 = Brown-out Reset is disabled in hardware; SBOREN bit is disabled

Note 1: Applies only to 28-pin devices.
2: The MCLRE fuse can only be changed when using the VPP-Based ICSP mode entry. This prevents a user from accidentally locking out the device from the low-voltage test entry.
3: Refer to Section 29.0, Electrical Characteristics for the BOR voltages.

27.0 DEVELOPMENT SUPPORT

The PIC$^{\text{fi}}$ microcontrollers and dsPIC$^{\text{fi}}$ digital signal controllers are supported with a full range of software and hardware development tools:

Integrated Development Environment
- MPLAB$^{\text{fi}}$ IDE Software

Compilers/Assemblers/Linkers
- MPLAB C Compiler for Various Device Families
- HI-TECH C$^{\text{fi}}$ for Various Device Families
- MPASM™ Assembler
- MPLINK™ Object Linker/ MPLIB™ Object Librarian
- MPLAB Assembler/Linker/Librarian for Various Device Families

Simulators
- MPLAB SIM Software Simulator

Emulators
- MPLAB REAL ICE In-Circuit Emulator

In-Circuit Debuggers
- MPLAB ICD 3
- PICkit 3 Debug Express

Device Programmers
- PICkit 2 Programmer
- MPLAB PM3 Device Programmer

Low-Cost Demonstration/Development Boards, Evaluation Kits, and Starter Kits

27.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16/32-bit microcontroller market. The MPLAB IDE is a Windows$^{\text{fi}}$ operating system-based application that contains:

A single graphical interface to all debugging tools
- Simulator
- Programmer (sold separately)
- In-Circuit Emulator (sold separately)
- In-Circuit Debugger (sold separately)

A full-featured editor with color-coded context

A multiple project manager

Customizable data windows with direct edit of contents

High-level source code debugging

Mouse over variable inspection

Drag and drop variables from source to watch windows

Extensive on-line help

Integration of select third party tools, such as IAR C Compilers

The MPLAB IDE allows you to:

Edit your source files (either C or assembly)

One-touch compile or assemble, and download to emulator and simulator tools (automatically updates all project information)

Debug using:
- Source files (C or assembly)
- Mixed C and assembly
- Machine code

MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost-effective simulators, through low-cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increased flexibility and power.

TABLE 28-2: 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	N, Z
	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	N, Z
	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