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	Surface Mount
Package / Case	20-VQFN Exposed Pad
Supplier Device Package	20-VQFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic24f08ka101-e-mq

TABLE 4-12: PORTA 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[1]	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
TRISA	02C0	—	—	—	—	—	—	—	—	TRISA7[4]	TRISA6	—	TRISA4	TRISA3[5,6]	TRISA2[5]	TRISA1	TRISA0	00DF
PORTA	02C2	—	—	—	—	—	—	—	—	RA7[4]	RA6	RA5	RA4[3]	RA3[5,6]	RA2[5]	RA1[2]	RA0[2]	xxxx
LATA	02C4	—	—	—	—	—	—	—	—	LATA7[4]	LATA6	—	LATA4	LATA3[5,6]	LATA2[5]	LATA1	LATA0	xxxx
ODCA	02C6	—	—	—	—	—	—	—	—	ODA7[4]	ODA6	—	ODA4	ODA3[5,6]	ODA2[5]	ODA1	ODA0	0000

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

Note 1: This bit is available only when MCLRE = 0.

2: A read of RA1 and RA0 results in '0' when debug is active on the PGC2/PGD2 pin.

3: A read of RA4 results in '0' when debug is active on the PGC3/PGD3 pin.

4: These bits are not implemented in 20-pin devices.

5: These bits are available only when the primary oscillator is disabled (POSCMD<1:0> = 00); otherwise read as '0'.

6: These bits are available only when the primary oscillator is disabled or EC mode is selected (POSCMD<1:0> = 00 or 11) and CLKO is disabled (OSCIOFNC = 0); otherwise read as '0'.

TABLE 4-13: PORTB 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
TRISB	02C8	TRISB15	TRISB14	TRISB13	TRISB12	TRISB11[3]	TRISB10[3]	TRISB9	TRISB8	TRISB7	TRISB6[3]	TRISB5[3]	TRISB4	TRISB3[3]	TRISB2	TRISB1	TRISB0	FFFF
PORTB	02CA	RB15	RB14	RB13	RB12	RB11[3]	RB10[3]	RB9	RB8	RB7	RB6[3]	RB5[3]	RB4[2]	RB3[3]	RB2	RB1[1]	RB0[1]	xxxx
LATB	02CC	LATB15	LATB14	LATB13	LATB12	LATB11[3]	LATB10[3]	LATB9	LATB8	LATB7	LATB6[3]	LATB5[3]	LATB4	LATB3[3]	LATB2	LATB1	LATB0	xxxx
ODCB	02CE	ODB15	ODB14	ODB13	ODB12	ODB11	ODB10	ODB9	ODB8	ODB7	ODB6	ODB5	ODB4	ODB3	ODB2	ODB1	ODB0	0000

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

Note 1: A read of RB1 and RB0 results in '0' when debug is active on the PGEC1/PGED1 pins.

2: A read of RB4 results in '0' when debug is active on the PGEC3/PGED3 pins.

3: PORTB bits, 11, 10, 6, 5 and 3, are not implemented in 20-pin devices.

TABLE 4-14: PAD CONFIGURATION 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
PADCFG1	02FC	—	—	—	—	—	—	—	—	—	—	—	SMBUSDEL	OC1TRIS	RTSECSEL1	RTSECSEL0	—	0000

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

6.3 NVM Address Register

As with Flash program memory, the NVM Address Registers, NVMADRU and NVMADR, form the 24-bit Effective Address (EA) of the selected row or word for data EEPROM operations. The NVMADRU register is used to hold the upper 8 bits of the EA, while the NVMADR register is used to hold the lower 16 bits of the EA. These registers are not mapped into the Special Function Register (SFR) space; instead, they directly capture the EA<23:0> of the last table write instruction that has been executed and selects the data EEPROM row to erase. Figure 6-1 depicts the program memory EA that is formed for programming and erase operations.

Like program memory operations, the Least Significant bit (LSb) of NVMADR is restricted to even addresses. This is because any given address in the data EEPROM space consists of only the lower word of the program memory width; the upper word, including the uppermost "phantom byte", are unavailable. This means that the LSb of a data EEPROM address will always be '0'.

Similarly, the Most Significant bit (MSb) of NVMADRU is always '0', since all addresses lie in the user program space.

FIGURE 6-1: DATA EEPROM ADDRESSING WITH TBLPAG AND NVM ADDRESS REGISTERS
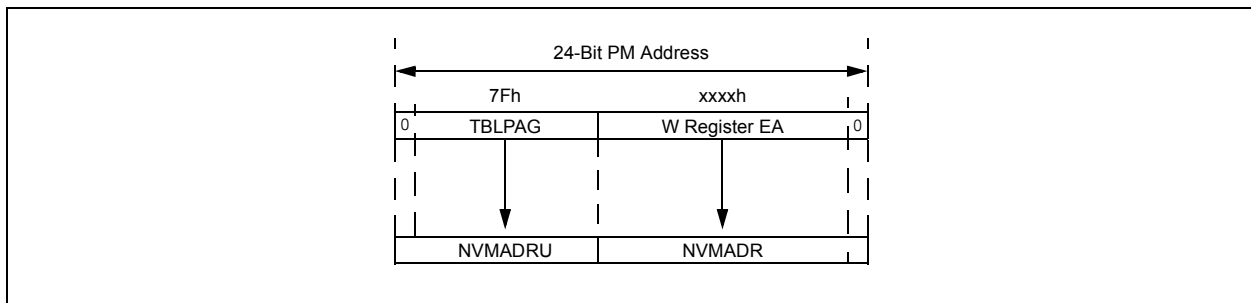
6.4 Data EEPROM Operations

The EEPROM block is accessed using table read and write operations, similar to those used for program memory. The TBLWTH and TBLRDH instructions are not required for data EEPROM operations, since the memory is only 16 bits wide (data on the lower address is valid only). The following programming operations can be performed on the data EEPROM:

• Erase one, four or eight words
• Bulk erase the entire data EEPROM
• Write one word
• Read one word

> **Note 1:** Unexpected results will be obtained should the user attempt to read the EEPROM while a programming or erase operation is underway.
>
> **2:** The C30 C compiler includes library procedures to automatically perform the table read and table write operations, manage the Table Pointer and write buffers, and unlock and initiate memory write sequences. This eliminates the need to create assembler macros or time critical routines in C for each application.

The library procedures are used in the code examples detailed in the following sections. General descriptions of each process are provided for users who are not using the C30 compiler libraries.

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
```

7.1 Clock Source Selection at Reset

If clock switching is enabled, the system clock source at device Reset is chosen, as shown in Table 7-2. If clock switching is disabled, the system clock source is always selected according to the oscillator Configuration bits. Refer to **Section 9.0 "Oscillator Configuration"** for further details.

TABLE 7-2: OSCILLATOR SELECTION vs. TYPE OF RESET (CLOCK SWITCHING ENABLED)

Reset Type	Clock Source Determinant
POR	FNOSC Configuration bits (FNOSC<10:8>)
BOR	
$\overline{\text{MCLR}}$	COSC Control bits (OSCCON<14:12>)
WDTO	
SWR	

7.2 Device Reset Times

The Reset times for various types of device Reset are summarized in Table 7-3. Note that the system Reset signal, $\overline{\text{SYSRST}}$, is released after the POR and PWRT delay times expire.

The time at which the device actually begins to execute code will also depend on the system oscillator delays, which include the Oscillator Start-up Timer (OST) and the PLL lock time. The OST and PLL lock times occur in parallel with the applicable $\overline{\text{SYSRST}}$ delay times.

The FSCM delay determines the time at which the FSCM begins to monitor the system clock source after the $\overline{\text{SYSRST}}$ signal is released.

TABLE 7-3: RESET DELAY TIMES FOR VARIOUS DEVICE RESETS

Reset Type	Clock Source	$\overline{\text{SYSRST}}$ Delay	System Clock Delay	Notes
POR[6]	EC	$T_{POR} + T_{PWRT}$	—	1, 2
	FRC, FRCDIV	$T_{POR} + T_{PWRT}$	T_{FRC}	1, 2, 3
	LPRC	$T_{POR} + T_{PWRT}$	T_{LPRC}	1, 2, 3
	ECPLL	$T_{POR} + T_{PWRT}$	T_{LOCK}	1, 2, 4
	FRCPLL	$T_{POR} + T_{PWRT}$	$T_{FRC} + T_{LOCK}$	1, 2, 3, 4
	XT, HS, SOSC	$T_{POR} + T_{PWRT}$	T_{OST}	1, 2, 5
	XTPLL, HSPLL	$T_{POR} + T_{PWRT}$	$T_{OST} + T_{LOCK}$	1, 2, 4, 5
BOR	EC	T_{PWRT}	—	2
	FRC, FRCDIV	T_{PWRT}	T_{FRC}	2, 3
	LPRC	T_{PWRT}	T_{LPRC}	2, 3
	ECPLL	T_{PWRT}	T_{LOCK}	2, 4
	FRCPLL	T_{PWRT}	$T_{FRC} + T_{LOCK}$	2, 3, 4
	XT, HS, SOSC	T_{PWRT}	T_{OST}	2, 5
	XTPLL, HSPLL	T_{PWRT}	$T_{FRC} + T_{LOCK}$	2, 3, 4
All Others	Any Clock	—	—	**None**

Note 1: T_{POR} = Power-on Reset (POR) delay.

2: T_{PWRT} = 64 ms nominal if the Power-up Timer (PWRT) is enabled; otherwise, it is zero.

3: T_{FRC} and T_{LPRC} = RC oscillator start-up times.

4: T_{LOCK} = PLL lock time.

5: T_{OST} = Oscillator Start-up Timer (OST). A 10-bit counter waits 1024 oscillator periods before releasing the oscillator clock to the system.

6: If Two-Speed Start-up is enabled, regardless of the primary oscillator selected, the device starts with FRC, and in such cases, FRC start-up time is valid.

Note: For detailed operating frequency and timing specifications, see **Section 29.0 "Electrical Characteristics"**.

10.2.4.2   Exiting Deep Sleep Mode

Deep Sleep mode exits on any one of the following events:

- POR event on V$_{DD}$ supply. If there is no DSBOR circuit to re-arm the V$_{DD}$ supply POR circuit, the external V$_{DD}$ supply must be lowered to the natural arming voltage of the POR circuit.
- DSWDT time-out. When the DSWDT timer times out, the device exits Deep Sleep.
- RTCC alarm (if RTCEN = 1).
- Assertion ('0') of the $\overline{MCLR}$ pin.
- Assertion of the INT0 pin (if the interrupt was enabled before Deep Sleep mode was entered). The polarity configuration is used to determine the assertion level ('0' or '1') of the pin that will cause an exit from Deep Sleep mode. Exiting from Deep Sleep mode requires a change on the INT0 pin while in Deep Sleep mode.

> **Note:** Any interrupt pending when entering Deep Sleep mode is cleared,

Exiting Deep Sleep mode generally does not retain the state of the device and is equivalent to a Power-on Reset (POR) of the device. Exceptions to this include the RTCC (if present), which remains operational through the wake-up, the DSGPRx registers and the DSWDT bit.

Wake-up events that occur from the time Deep Sleep exits until the time the POR sequence completes are ignored and are not be captured in the DSWAKE register.

The sequence for exiting Deep Sleep mode is:

1. After a wake-up event, the device exits Deep Sleep and performs a POR. The DSEN bit is cleared automatically. Code execution resumes at the Reset vector.
2. To determine if the device exited Deep Sleep, read the Deep Sleep bit, DPSLP (RCON<10>). This bit will be set if there was an exit from Deep Sleep mode; if the bit is set, clear it.
3. Determine the wake-up source by reading the DSWAKE register.
4. Determine if a DSBOR event occurred during Deep Sleep mode by reading the DSBOR bit (DSCON<1>).
5. If application context data has been saved, read it back from the DSGPR0 and DSGPR1 registers.
6. Clear the RELEASE bit (DSCON<0>).

10.2.4.3   Saving Context Data with the DSGPR0/DSGPR1 Registers

As exiting Deep Sleep mode causes a POR, most Special Function Registers reset to their default POR values. In addition, because V$_{DDCORE}$ power is not supplied in Deep Sleep mode, information in data RAM may be lost when exiting this mode.

Applications which require critical data to be saved prior to Deep Sleep may use the Deep Sleep General Purpose registers, DSGPR0 and DSGPR1, or data EEPROM (if available). Unlike other SFRs, the contents of these registers are preserved while the device is in Deep Sleep mode. After exiting Deep Sleep, software can restore the data by reading the registers and clearing the RELEASE bit (DSCON<0>).

10.2.4.4   I/O Pins During Deep Sleep

During Deep Sleep, the general purpose I/O pins retain their previous states and the Secondary Oscillator (SOSC) will remain running, if enabled. Pins that are configured as inputs (TRISx bit set), prior to entry into Deep Sleep, remain high-impedance during Deep Sleep. Pins that are configured as outputs (TRISx bit clear), prior to entry into Deep Sleep, remain as output pins during Deep Sleep. While in this mode, they continue to drive the output level determined by their corresponding LATx bit at the time of entry into Deep Sleep.

Once the device wakes back up, all I/O pins continue to maintain their previous states, even after the device has finished the POR sequence and is executing application code again. Pins configured as inputs during Deep Sleep remain high-impedance and pins configured as outputs continue to drive their previous value. After waking up, the TRIS and LAT registers, and the SOSCEN bit (OSCCON<1>) are reset. If firmware modifies any of these bits or registers, the I/O will not immediately go to the newly configured states. Once the firmware clears the RELEASE bit (DSCON<0>), the I/O pins are "released". This causes the I/O pins to take the states configured by their respective TRIS and LAT bit values.

This means that keeping the SOSC running after waking up requires the SOSCEN bit to be set before clearing RELEASE.

If the Deep Sleep BOR (DSBOR) is enabled, and a DSBOR or a true POR event occurs during Deep Sleep, the I/O pins will be immediately released, similar to clearing the RELEASE bit. All previous state information will be lost, including the general purpose DSGPR0 and DSGPR1 contents.

If a $\overline{MCLR}$ Reset event occurs during Deep Sleep, the DSGPRx, DSCON and DSWAKE registers will remain valid, and the RELEASE bit will remain set. The state of the SOSC will also be retained. The I/O pins, however, will be reset to their $\overline{MCLR}$ Reset state. Since RELEASE is still set, changes to the SOSCEN bit (OSCCON<1>) cannot take effect until the RELEASE bit is cleared.

In all other Deep Sleep wake-up cases, application firmware must clear the RELEASE bit in order to reconfigure the I/O pins.

11.1.1 OPEN-DRAIN CONFIGURATION

In addition to the PORT, LAT and TRIS registers for data control, each port pin can also be individually configured for either digital or open-drain output. This is controlled by the Open-Drain Control register, ODCx, associated with each port. Setting any of the bits configures the corresponding pin to act as an open-drain output.

The maximum open-drain voltage allowed is the same as the maximum V$_{IH}$ specification.

11.2 Configuring Analog Port Pins

The use of the AD1PCFG and TRIS register controls the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bit set (input). If the TRIS bit is cleared (output), the digital output level (V$_{OH}$ or V$_{OL}$) will be converted.

When reading the PORT register, all pins configured as analog input channels will read as cleared (a low level). Analog levels on any pin that is defined as a digital input (including the ANx pins) may cause the input buffer to consume current that exceeds the device specifications.

11.2.1 I/O PORT WRITE/READ TIMING

One instruction cycle is required between a port direction change or port write operation and a read operation of the same port. Typically, this instruction would be a NOP.

11.3 Input Change Notification

The input change notification function of the I/O ports allows the PIC24F16KA102 family of devices to generate interrupt requests to the processor in response to a Change-of-State (COS) on selected input pins. This feature is capable of detecting input Change-of-States even in Sleep mode, when the clocks are disabled. Depending on the device pin count, there are up to 23 external signals (CN0 through CN22) that may be selected (enabled) for generating an interrupt request on a Change-of-State.

There are six control registers associated with the CN module. The CNEN1 and CNEN2 registers contain the interrupt enable control bits for each of the CN input pins. Setting any of these bits enables a CN interrupt for the corresponding pins.

Each CN pin also has a weak pull-up/pull-down connected to it. The pull-ups act as a current source that is connected to the pin and the pull-downs act as a current sink to eliminate the need for external resistors when push button or keypad devices are connected.

On any pin, only the pull-up resistor or the pull-down resistor should be enabled, but not both of them. If the push button or the keypad is connected to V$_{DD}$, enable the pull-down, or if they are connected to V$_{SS}$, enable the pull-up resistors. The pull-ups are enabled separately using the CNPU1 and CNPU2 registers, which contain the control bits for each of the CN pins.

Setting any of the control bits enables the weak pull-ups for the corresponding pins. The pull-downs are enabled separately using the CNPD1 and CNPD2 registers, which contain the control bits for each of the CN pins. Setting any of the control bits enables the weak pull-downs for the corresponding pins.

When the internal pull-up is selected, the pin uses V$_{DD}$ as the pull-up source voltage. When the internal pull-down is selected, the pins are pulled down to V$_{SS}$ by an internal resistor. Make sure that there is no external pull-up source/pull-down sink when the internal pull-ups/pull-downs are enabled.

> **Note:** Pull-ups and pull-downs on change notification pins should always be disabled whenever the port pin is configured as a digital output.

EXAMPLE 11-1: PORT WRITE/READ EXAMPLE

```
MOV    0xFF00, W0;          //Configure PORTB<15:8> as inputs and PORTB<7:0> as outputs
MOV    W0, TRISBB;
NOP;                        //Delay 1 cycle
BTSS   PORTB, #13;          //Next Instruction

Equivalent 'C' Code
TRISB = 0xFF00;             //Configure PORTB<15:8> as inputs and PORTB<7:0> as outputs
NOP();                      //Delay 1 cycle
if(PORTBbits.RB13 == 1)     // execute following code if PORTB pin 13 is set.
{
}
```

16.0 SERIAL PERIPHERAL INTERFACE (SPI)

> **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 Serial Peripheral Interface, refer to the *"PIC24F Family Reference Manual"*, **Section 23. "Serial Peripheral Interface (SPI)"** (DS39699).

The Serial Peripheral Interface (SPI) module is a synchronous serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial data EEPROMs, shift registers, display drivers, A/D Converters, etc. The SPI module is compatible with the SPI and SIOP interfaces from Motorola®.

The module supports operation in two buffer modes. In Standard mode, data is shifted through a single serial buffer. In Enhanced Buffer mode, data is shifted through an 8-level FIFO buffer.

> **Note:** Do not perform read-modify-write operations (such as bit-oriented instructions) on the SPI1BUF register in either Standard or Enhanced Buffer mode.

The module also supports a basic framed SPI protocol while operating in either Master or Slave mode. A total of four framed SPI configurations are supported.

The SPI serial interface consists of four pins:
- SDI1: Serial Data Input
- SDO1: Serial Data Output
- SCK1: Shift Clock Input or Output
- $\overline{SS1}$: Active-Low Slave Select or Frame Synchronization I/O Pulse

The SPI module can be configured to operate using 2, 3 or 4 pins. In the 3-pin mode, $\overline{SS1}$ is not used. In the 2-pin mode, both SDO1 and $\overline{SS1}$ are not used.

Block diagrams of the module in Standard and Enhanced Buffer modes are displayed in Figure 16-1 and Figure 16-2.

The devices of the PIC24F16KA102 family offer one SPI module on a device.

> **Note:** In this section, the SPI module is referred to as SPI1, or separately as SPI1. Special Function Registers (SFRs) will follow a similar notation. For example, SPI1CON1 or SPI1CON2 refers to the control register for the SPI1 module.

To set up the SPI module for the Standard Master mode of operation:

1. If using interrupts:
   a) Clear the respective SPI1IF bit in the IFS0 register.
   b) Set the respective SPI1IE bit in the IEC0 register.
   c) Write the respective SPI1IPx bits in the IPC2 register to set the interrupt priority.
2. Write the desired settings to the SPI1CON1 and SPI1CON2 registers with the MSTEN bit (SPI1CON1<5>) = 1.
3. Clear the SPIROV bit (SPI1STAT<6>).
4. Enable SPI operation by setting the SPIEN bit (SPI1STAT<15>).
5. Write the data to be transmitted to the SPI1BUF register. Transmission (and reception) will start as soon as data is written to the SPI1BUF register.

To set up the SPI module for the Standard Slave mode of operation:

1. Clear the SPI1BUF register.
2. If using interrupts:
   a) Clear the respective SPI1IF bit in the IFS0 register.
   b) Set the respective SPI1IE bit in the IEC0 register.
   c) Write the respective SPI1IP bits in the IPC2 register to set the interrupt priority.
3. Write the desired settings to the SPI1CON1 and SPI1CON2 registers with the MSTEN bit (SPI1CON1<5>) = 0.
4. Clear the SMP bit.
5. If the CKE bit is set, then the SSEN bit (SPI1CON1<7>) must be set to enable the $\overline{SS1}$ pin.
6. Clear the SPIROV bit (SPI1STAT<6>).
7. Enable SPI operation by setting the SPIEN bit (SPI1STAT<15>).

REGISTER 17-1: I2C1CON: I2C1 CONTROL REGISTER (CONTINUED)

bit 5 **ACKDT:** Acknowledge Data bit (when operating as I^{2}C master; applicable during master receive)
Value that will be transmitted when the software initiates an Acknowledge sequence.
1 = Sends NACK during Acknowledge
0 = Sends ACK during Acknowledge

bit 4 **ACKEN:** Acknowledge Sequence Enable bit
(when operating as I^{2}C master; applicable during master receive)
1 = Initiates Acknowledge sequence on SDA1 and SCL1 pins and transmits ACKDT data bit; hardware is clear at the end of the master Acknowledge sequence
0 = Acknowledge sequence is not in progress

bit 3 **RCEN:** Receive Enable bit (when operating as I^{2}C master)
1 = Enables Receive mode for I^{2}C; hardware is clear at the end of eighth bit of master receive data byte
0 = Receive sequence not in progress

bit 2 **PEN:** Stop Condition Enable bit (when operating as I^{2}C master)
1 = Initiates Stop condition on SDA1 and SCL1 pins; hardware is clear at end of master Stop sequence
0 = Stop condition is not in progress

bit 1 **RSEN:** Repeated Start Condition Enable bit (when operating as I^{2}C master)
1 = Initiates Repeated Start condition on SDA1 and SCL1 pins; hardware is clear at end of master Repeated Start sequence
0 = Repeated Start condition is not in progress

bit 0 **SEN:** Start Condition Enable bit (when operating as I^{2}C master)
1 = Initiates Start condition on SDA1 and SCL1 pins; hardware is clear at end of master Start sequence
0 = Start condition is not in progress

21.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

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 High/Low-Voltage Detect, refer to the *"PIC24F Family Reference Manual"*, **Section 36. "High-Level Integration with Programmable High/Low-Voltage Detect (HLVD)"** (DS39725).

The High/Low-Voltage Detect module (HLVD) is a programmable circuit that allows the user to specify both the device voltage trip point and the direction of change.

An interrupt flag is set if the device experiences an excursion past the trip point in the direction of change. If the interrupt is enabled, the program execution will branch to the interrupt vector address and the software can then respond to the interrupt.

The HLVD Control register (see Register 21-1) completely controls the operation of the HLVD module. This allows the circuitry to be "turned off" by the user under software control, which minimizes the current consumption for the device.

FIGURE 21-1: HIGH/LOW-VOLTAGE DETECT (HLVD) MODULE BLOCK DIAGRAM
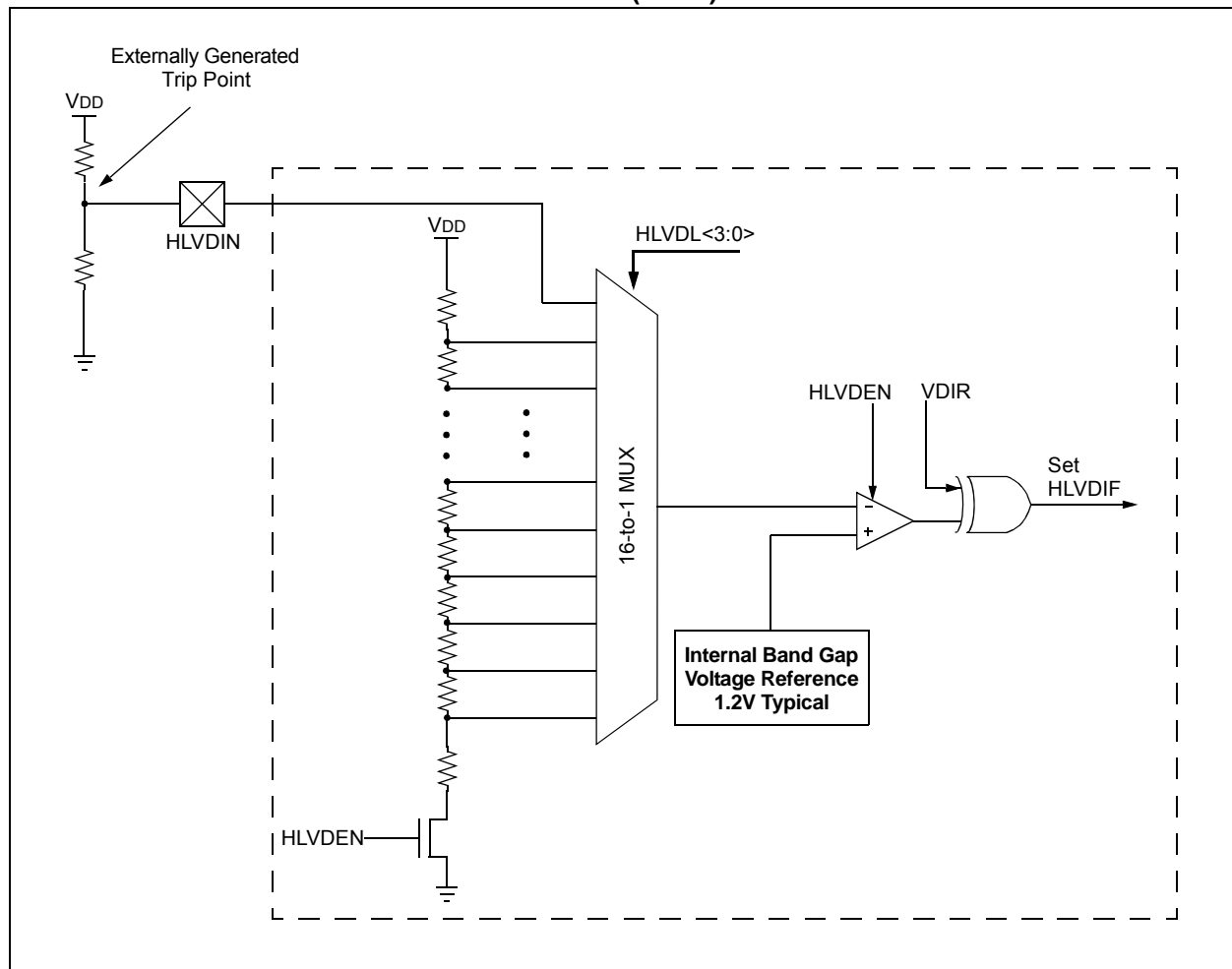
25.2 Measuring Time

Time measurements on the pulse width can be similarly performed using the A/D module's internal capacitor (C$_{AD}$) and a precision resistor for current calibration. Figure 25-2 displays the external connections used for time measurements, and how the CTMU and A/D modules are related in this application. This example also shows both edge events coming from the external CTED pins, but other configurations using internal edge sources are possible.

25.3 Pulse Generation and Delay

The CTMU module can also generate an output pulse with edges that are not synchronous with the device's system clock. More specifically, it can generate a pulse with a programmable delay from an edge event input to the module.

When the module is configured for pulse generation delay by setting the TGEN bit (CTMUCON<12>), the internal current source is connected to the B input of Comparator 2. A capacitor (C$_{DELAY}$) is connected to the Comparator 2 pin, C2INB, and the comparator voltage reference, CV$_{REF}$, is connected to C2INA. CV$_{REF}$ is then configured for a specific trip point. The module begins to charge C$_{DELAY}$ when an edge event is detected. When C$_{DELAY}$ charges above the CV$_{REF}$ trip point, a pulse is output on CTPLS. The length of the pulse delay is determined by the value of C$_{DELAY}$ and the CV$_{REF}$ trip point.

Figure 25-3 shows the external connections for pulse generation, as well as the relationship of the different analog modules required. While CTED1 is shown as the input pulse source, other options are available. A detailed discussion on pulse generation with the CTMU module is provided in the "*PIC24F Family Reference Manual*".

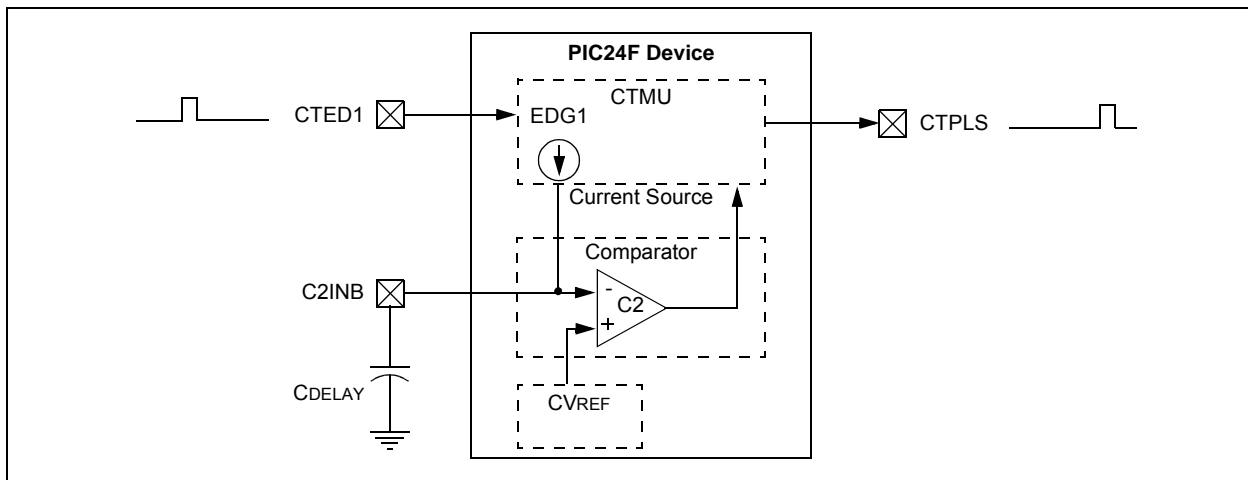**FIGURE 25-2:** TYPICAL CONNECTIONS AND INTERNAL CONFIGURATION FOR TIME MEASUREMENT
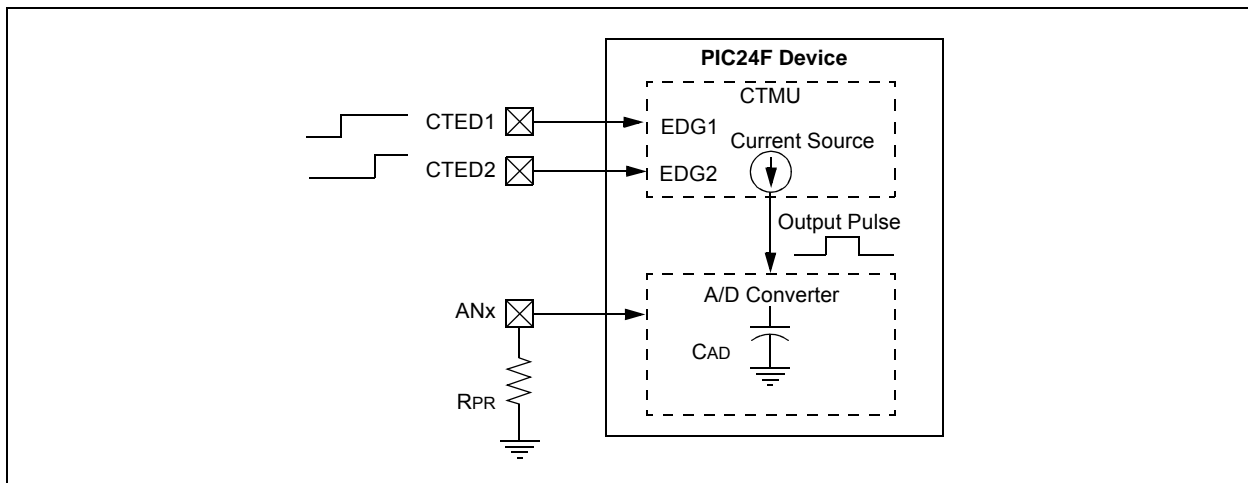
FIGURE 25-3: TYPICAL CONNECTIONS AND INTERNAL CONFIGURATION FOR PULSE DELAY GENERATION

REGISTER 26-4: FOSC: OSCILLATOR CONFIGURATION REGISTER

R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
FCKSM1	FCKSM0	SOSCSEL	POSCFREQ1	POSCFREQ0	OSCIOFNC	POSCMD1	POSCMD0
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 7-6 **FCKSM<1:0>:** Clock Switching and Monitor Selection Configuration bits

    1x = Clock switching is disabled, Fail-Safe Clock Monitor is disabled
    01 = Clock switching is enabled, Fail-Safe Clock Monitor is disabled
    00 = Clock switching is enabled, Fail-Safe Clock Monitor is enabled

bit 5 **SOSCSEL:** Secondary Oscillator Select bit

    1 = Secondary oscillator is configured for high-power operation
    0 = Secondary oscillator is configured for low-power operation

bit 4-3 **POSCFREQ<1:0>:** Primary Oscillator Frequency Range Configuration bits

    11 = Primary oscillator/external clock input frequency is greater than 8 MHz
    10 = Primary oscillator/external clock input frequency is between 100 kHz and 8 MHz
    01 = Primary oscillator/external clock input frequency is less than 100 kHz
    00 = Reserved; do not use

bit 2 **OSCIOFNC:** CLKO Enable Configuration bit

    1 = CLKO output signal active on the OSCO pin; primary oscillator must be disabled or configured for the External Clock mode (EC) for the CLKO to be active (POSCMD<1:0> = 11 or 00)
    0 = CLKO output is disabled

bit 1-0 **POSCMD<1:0>:** Primary Oscillator Configuration bits

    11 = Primary Oscillator mode is disabled
    10 = HS Oscillator mode is selected
    01 = XT Oscillator mode is selected
    00 = External Clock mode is selected

27.2 MPLAB C Compilers for Various Device Families

The MPLAB C Compiler code development systems are complete ANSI C compilers for Microchip's PIC18, PIC24 and PIC32 families of microcontrollers and the dsPIC30 and dsPIC33 families of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

27.3 HI-TECH C for Various Device Families

The HI-TECH C Compiler code development systems are complete ANSI C compilers for Microchip's PIC family of microcontrollers and the dsPIC family of digital signal controllers. These compilers provide powerful integration capabilities, omniscient code generation and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

The compilers include a macro assembler, linker, pre-processor, and one-step driver, and can run on multiple platforms.

27.4 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code and COFF files for debugging.

The MPASM Assembler features include:

• Integration into MPLAB IDE projects
• User-defined macros to streamline assembly code
• Conditional assembly for multi-purpose source files
• Directives that allow complete control over the assembly process

27.5 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler and the MPLAB C18 C Compiler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

• Efficient linking of single libraries instead of many smaller files
• Enhanced code maintainability by grouping related modules together
• Flexible creation of libraries with easy module listing, replacement, deletion and extraction

27.6 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC devices. MPLAB C Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

• Support for the entire device instruction set
• Support for fixed-point and floating-point data
• Command line interface
• Rich directive set
• Flexible macro language
• MPLAB IDE compatibility

27.7   MPLAB SIM Software Simulator

The MPLAB 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 SIM Software Simulator fully supports symbolic debugging using the MPLAB C 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.

27.8   MPLAB REAL ICE In-Circuit Emulator System

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 PIC® Flash MCUs and dsPIC® Flash DSCs with the easy-to-use, powerful graphical user interface of the MPLAB Integrated Development Environment (IDE), included with each kit.

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 (RJ11) 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 IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

27.9   MPLAB ICD 3 In-Circuit Debugger System

MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost effective high-speed hardware debugger/programmer for Microchip Flash Digital Signal Controller (DSC) and microcontroller (MCU) devices. It debugs and programs PIC® Flash microcontrollers and dsPIC® DSCs with the powerful, yet easy-to-use graphical user interface of MPLAB Integrated Development Environment (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.

27.10   PICkit 3 In-Circuit Debugger/ Programmer and PICkit 3 Debug Express

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 Integrated Development Environment (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 an 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™.

The PICkit 3 Debug Express include the PICkit 3, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

28.0 INSTRUCTION SET SUMMARY

> **Note:** This chapter is a brief summary of the PIC24F instruction set architecture 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 28-1 lists the general symbols used in describing the instructions. The PIC24F instruction set summary in Table 28-2 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 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 29-11: DC CHARACTERISTICS: PROGRAM MEMORY

DC CHARACTERISTICS			**Standard Operating Conditions: 1.8V to 3.6V (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for Industrial -40°C ≤ TA ≤ +125°C for Extended				
Param No.	Sym	Characteristic	Min	Typ[1]	Max	Units	Conditions
		Program Flash Memory					
D130	E_P	Cell Endurance	10,000[2]	—	—	E/W	
D131	V_{PR}	V_{DD} for Read	V_{MIN}	—	3.6	V	V_{MIN} = Minimum operating voltage
D133A	T_{IW}	Self-Timed Write Cycle Time	—	2	—	ms	
D134	T_{RETD}	Characteristic Retention	40	—	—	Year	Provided no other specifications are violated
D135	I_{DDP}	Supply Current During Programming	—	10	—	mA	

Note 1: Data in "Typ" column is at 3.3V, 25°C unless otherwise stated.
   2: Self-write and block erase.

TABLE 29-12: DC CHARACTERISTICS: DATA EEPROM MEMORY

DC CHARACTERISTICS			**Standard Operating Conditions: 1.8V to 3.6V (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for Industrial -40°C ≤ TA ≤ +125°C for Extended				
Param No.	Sym	Characteristic	Min	Typ[1]	Max	Units	Conditions
		Data EEPROM Memory					
D140	E_{PD}	Cell Endurance	100,000	—	—	E/W	
D141	V_{PRD}	V_{DD} for Read	V_{MIN}	—	3.6	V	V_{MIN} = Minimum operating voltage
D143A	T_{IWD}	Self-Timed Write Cycle Time	—	4	—	ms	
D143B	T_{REF}	Number of Total Write/Erase Cycles Before Refresh	—	10M	—	E/W	
D144	T_{RETDD}	Characteristic Retention	40	—	—	Year	Provided no other specifications are violated
D145	I_{DDPD}	Supply Current During Programming	—	7	—	mA	

Note 1: Data in "Typ" column is at 3.3V, 25°C unless otherwise stated.

TABLE 29-22: AC SPECIFICATIONS

Symbol	Characteristics	Min	Typ	Max	Units
T_{LW}	BCLKx High Time	20	$T_{CY}/2$	—	ns
T_{HW}	BCLKx Low Time	20	$(T_{CY} * BRGx) + T_{CY}/2$	—	ns
T_{BLD}	BCLKx Falling Edge Delay from UxTX	-50	—	50	ns
T_{BHD}	BCLKx Rising Edge Delay from UxTX	$T_{CY}/2 - 50$	—	$T_{CY}/2 + 50$	ns
T_{WAK}	Min. Low on UxRX Line to Cause Wake-up	—	1	—	μs
T_{CTS}	Min. Low on $\overline{UxCTS}$ Line to Start Transmission	T_{CY}	—	—	ns
T_{SETUP}	Start bit Falling Edge to System Clock Rising Edge Setup Time	3	—	—	ns
$T_{STDELAY}$	Maximum Delay in the Detection of the Start bit Falling Edge	—	—	$T_{CY} + T_{SETUP}$	ns

TABLE 29-23: A/D CONVERSION TIMING REQUIREMENTS[1]

A/D CHARACTERISTICS				Standard Operating Conditions: 1.8V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for Industrial -40°C ≤ TA ≤ +125°C for Extended				
Param No.	Symbol	Characteristic	Min.	Typ	Max.	Units	Conditions	
Clock Parameters								
AD50	T_{AD}	A/D Clock Period	75	—	—	ns	T_{CY} = 75 ns, AD1CON3 is in the default state	
AD51	T_{RC}	A/D Internal RC Oscillator Period	—	250	—	ns		
Conversion Rate								
AD55	T_{CONV}	Conversion Time	—	12	—	T_{AD}		
AD56	F_{CNV}	Throughput Rate	—	—	500	ksps	AVDD ≥ 2.7V	
AD57	T_{SAMP}	Sample Time	—	1	—	T_{AD}		
AD58	T_{ACQ}	Acquisition Time	750	—	—	ns	**(Note 2)**	
AD59	T_{SWC}	Switching Time from Convert to Sample	—	—	**(Note 3)**			
AD60	T_{DIS}	Discharge Time	0.5	—	—	T_{AD}		
Clock Parameters								
AD61	T_{PSS}	Sample Start Delay from Setting Sample bit (SAMP)	2	—	3	T_{AD}		

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.

FIGURE 29-16: **PWM MODULE TIMING REQUIREMENTS**
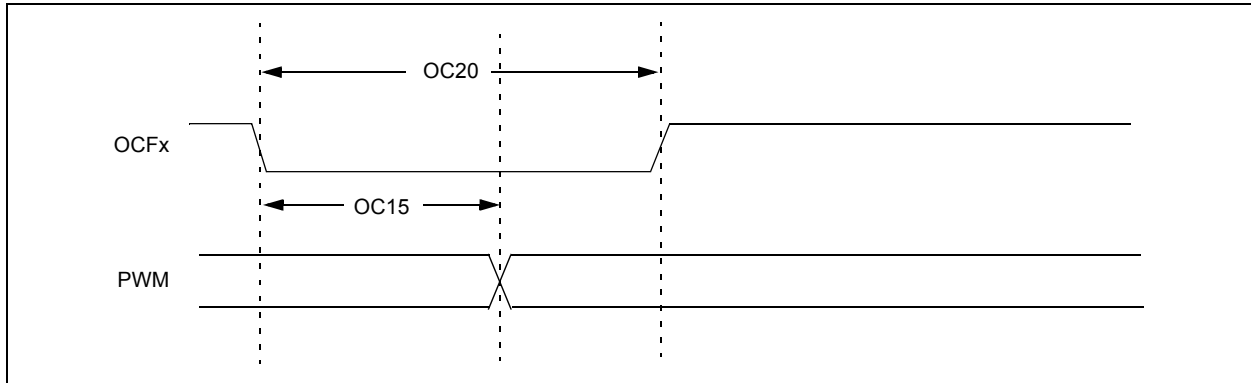
TABLE 29-35: **PWM TIMING REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
OC15	T$_{FD}$	Fault Input to PWM I/O Change	—	—	25	ns	V$_{DD}$ = 3.0V, -40°C to +125°C
OC20	T$_{FH}$	Fault Input Pulse Width	50	—	—	ns	V$_{DD}$ = 3.0V, -40°C to +125°C

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

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
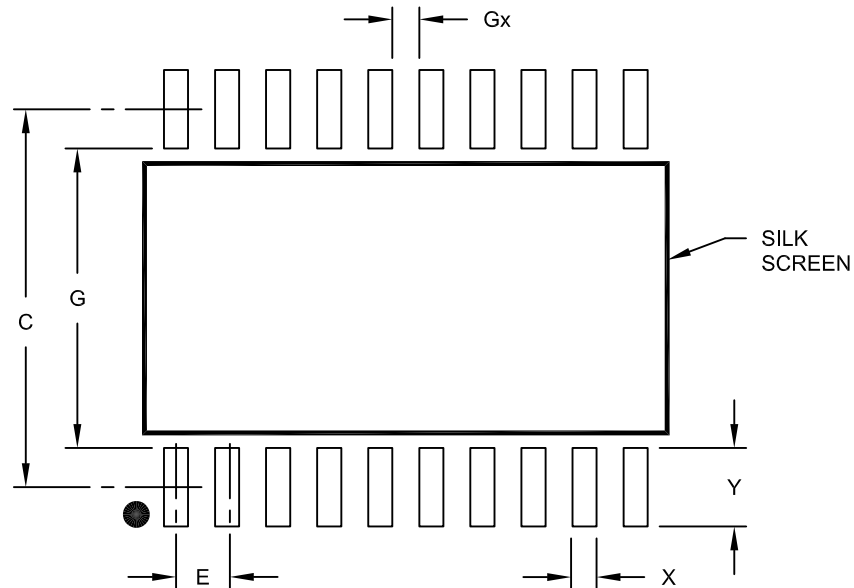
RECOMMENDED LAND PATTERN

	Units	MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E	1.27 BSC		
Contact Pad Spacing	C		9.40	
Contact Pad Width (X20)	X			0.60
Contact Pad Length (X20)	Y			1.95
Distance Between Pads	Gx	0.67		
Distance Between Pads	G	7.45		

Notes:
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2094A

NOTES:

PIC24F16KA102 FAMILY