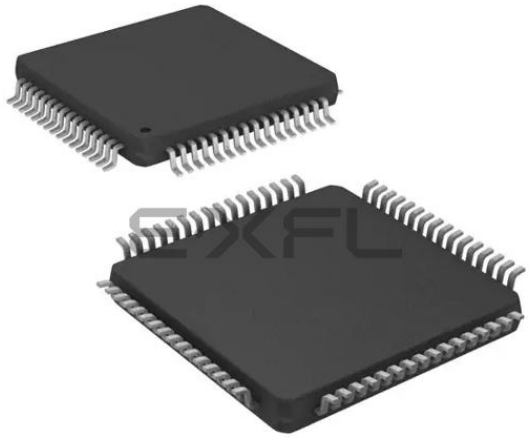


Welcome to E-XFL.COM

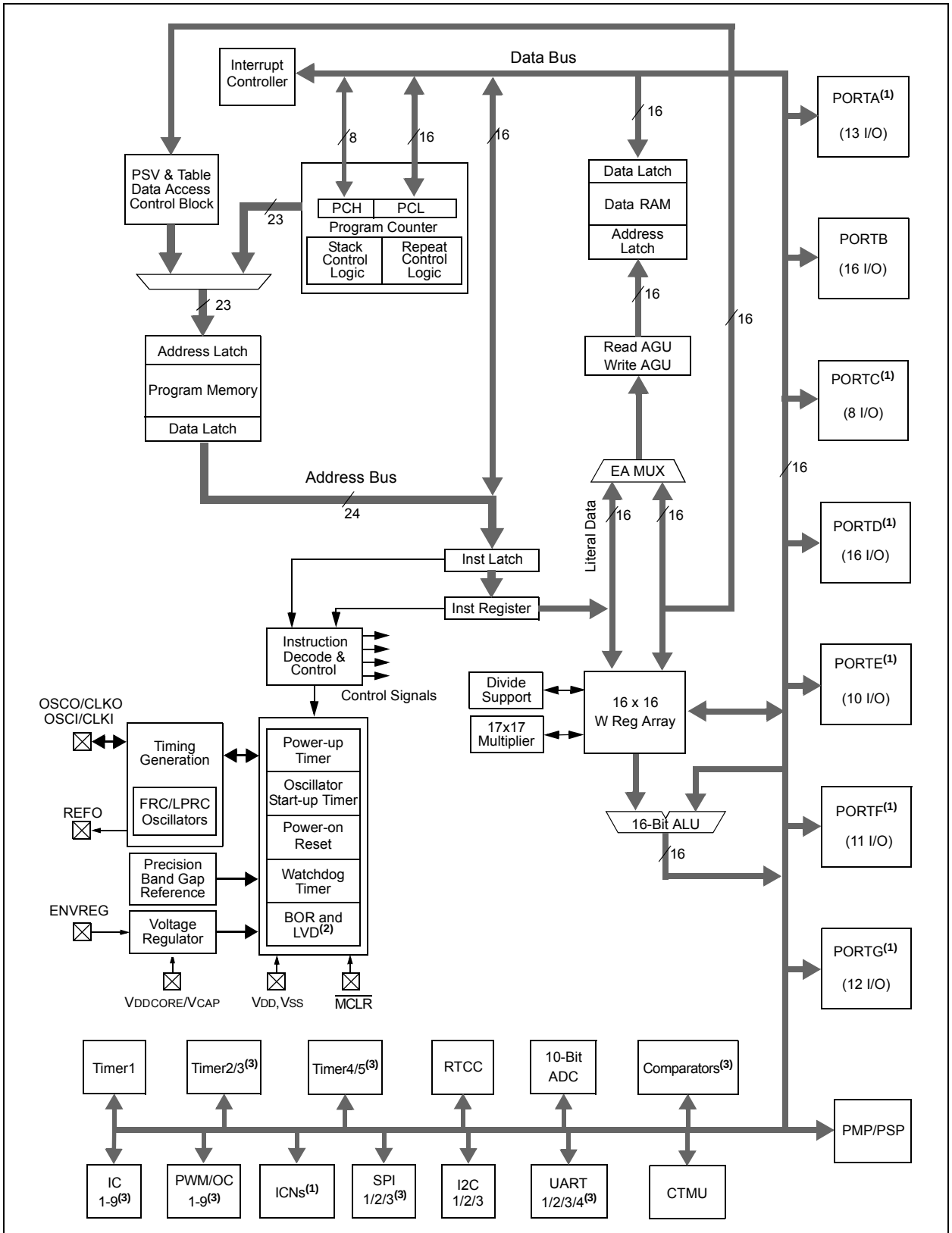
What is "Embedded - Microcontrollers"?

"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, PMP, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	53
Program Memory Size	128KB (43K x 24)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	16K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic24fj128ga106-e-pt

FIGURE 1-1: PIC24FJ256GA110 FAMILY GENERAL BLOCK DIAGRAM



Note 1: Not all I/O pins or features are implemented on all device pinout configurations. See Table 1-4 for specific implementations by pin count.
2: BOR functionality is provided when the on-board voltage regulator is enabled.
3: These peripheral I/Os are only accessible through remappable pins.

2.4 Voltage Regulator Pins (ENVREG/DISVREG and VCAP/VDDCORE)

Note: This section applies only to PIC24F J devices with an on-chip voltage regulator.

The on-chip voltage regulator enable/disable pin (ENVREG or DISVREG, depending on the device family) must always be connected directly to either a supply voltage or to ground. The particular connection is determined by whether or not the regulator is to be used:

- For ENVREG, tie to VDD to enable the regulator, or to ground to disable the regulator
- For DISVREG, tie to ground to enable the regulator or to VDD to disable the regulator

Refer to **Section 25.2 "On-Chip Voltage Regulator"** for details on connecting and using the on-chip regulator.

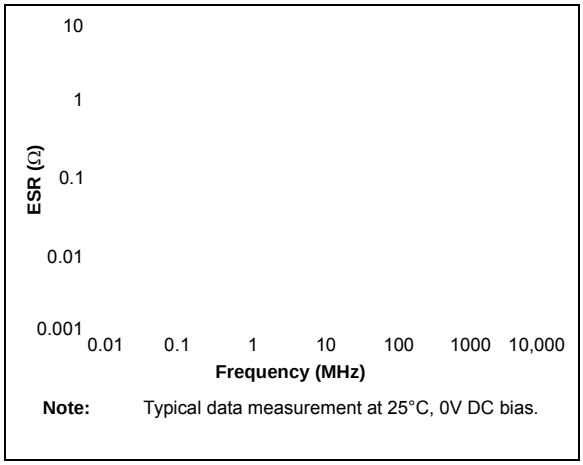
When the regulator is enabled, a low-ESR (< 5Ω) capacitor is required on the VCAP/VDDCORE pin to stabilize the voltage regulator output voltage. The VCAP/VDDCORE pin must not be connected to VDD and must use a capacitor of 10 µF connected to ground. The type can be ceramic or tantalum. Suitable examples of capacitors are shown in Table 2-1. Capacitors with equivalent specification can be used.

Designers may use Figure 2-3 to evaluate ESR equivalence of candidate devices.

The placement of this capacitor should be close to VCAP/VDDCORE. It is recommended that the trace length not exceed 0.25 inch (6 mm). Refer to **Section 28.0 "Electrical Characteristics"** for additional information.

When the regulator is disabled, the VCAP/VDDCORE pin must be tied to a voltage supply at the VDDCORE level. Refer to **Section 28.0 "Electrical Characteristics"** for information on VDD and VDDCORE.

FIGURE 2-3: FREQUENCY vs. ESR PERFORMANCE FOR SUGGESTED VCAP



Note: Typical data measurement at 25°C, 0V DC bias.

TABLE 2-1: SUITABLE CAPACITOR EQUIVALENTS

Make	Part #	Nominal Capacitance	Base Tolerance	Rated Voltage	Temp. Range
TDK	C3216X7R1C106K	10 µF	±10%	16V	-55 to 125ºC
TDK	C3216X5R1C106K	10 µF	±10%	16V	-55 to 85ºC
Panasonic	ECJ-3YX1C106K	10 µF	±10%	16V	-55 to 125ºC
Panasonic	ECJ-4YB1C106K	10 µF	±10%	16V	-55 to 85ºC
Murata	GRM32DR71C106KA01L	10 µF	±10%	16V	-55 to 125ºC
Murata	GRM31CR61C106KC31L	10 µF	±10%	16V	-55 to 85ºC

REGISTER 3-2: CORCON: CPU CONTROL REGISTER

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

U-0	U-0	U-0	U-0	R/C-0	R/W-0	U-0	U-0
—	—	—	—	IPL3(1)	PSV	—	—
bit 7							bit 0

Legend:	C = 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 15-4 **Unimplemented:** Read as '0'

bit 3 **IPL3:** CPU Interrupt Priority Level Status bit(1)
1 = CPU interrupt priority level is greater than 7
0 = CPU interrupt priority level is 7 or less

bit 2 **PSV:** Program Space Visibility in Data Space Enable bit
1 = Program space visible in data space
0 = Program space not visible in data space

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

Note 1: User interrupts are disabled when IPL3 = 1.

TABLE 4-27: SYSTEM 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
RCON	0740	TRAPR	IOPUWR	—	—	—	—	CM	PMSLP	EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR	**Note 1**
OSCCON	0742	—	COSC2	COSC1	COSC0	—	NOSC2	NOSC1	NOSC0	CLKLOCK	IOLOCK	LOCK	—	CF	POSCEN	SOSCEN	OSWEN	**Note 2**
CLKDIV	0744	ROI	DOZE2	DOZE1	DOZE0	DOZEN	RCDIV2	RCDIV1	RCDIV0	—	—	—	—	—	—	—	—	0100
OSCTUN	0748	—	—	—	—	—	—	—	—	—	—	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0	0000
REFOCON	074E	ROEN	—	ROSSLP	ROSEL	RODIV3	RODIV2	RODIV1	RODIV0	—	—	—	—	—	—	—	—	0000

Legend: — = unimplemented, read as '0'. Reset values are shown in hexadecimal.
Note 1: The Reset value of the RCON register is dependent on the type of Reset event. See **Section 6.0 "Resets"** for more information.
2: The Reset value of the OSCCON register is dependent on both the type of Reset event and the device configuration. See **Section 8.0 "Oscillator Configuration"** for more information.

TABLE 4-28: NVM 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
NVMCON	0760	WR	WREN	WRERR	—	—	—	—	—	—	ERASE	—	—	NVMOP3	NVMOP2	NVMOP1	NVMOP0	0000(1)
NVMKEY	0766	—	—	—	—	—	—	—	—	NVMKEY<7:0>								0000

Legend: — = unimplemented, read as '0'. Reset values are shown in hexadecimal.
Note 1: Reset value shown is for POR only. Value on other Reset states is dependent on the state of memory write or erase operations at the time of Reset.

TABLE 4-29: PMD 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
PMD1	0770	T5MD	T4MD	T3MD	T2MD	T1MD	—	—	—	I2C1MD	U2MD	U1MD	SPI2MD	SPI1MD	—	—	ADC1MD	0000
PMD2	0772	IC8MD	IC7MD	IC6MD	IC5MD	IC4MD	IC3MD	IC2MD	IC1MD	OC8MD	OC7MD	OC6MD	OC5MD	OC4MD	OC3MD	OC2MD	OC1MD	0000
PMD3	0774	—	—	—	—	—	CMPMD	RTCCMD	PMPMD	CRCMD	—	—	—	U3MD	I2C3MD	I2C2MD	—	0000
PMD4	0776	—	—	—	—	—	—	—	—	—	—	U4MD	—	REFOMD	CTMUMD	LVDMD	—	0000
PMD5	0778	—	—	—	—	—	—	—	IC9MD	—	—	—	—	—	—	—	OC9MD	0000
PMD6	077A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	SPI3MD	0000

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

4.2.5 SOFTWARE STACK

In addition to its use as a working register, the W15 register in PIC24F devices is also used as a Software Stack Pointer. The pointer always points to the first available free word and grows from lower to higher addresses. It predecrements for stack pops and post-increments for stack pushes, as shown in Figure 4-4. Note that for a PC push during any CALL instruction, the MSB of the PC is zero-extended before the push, ensuring that the MSB is always clear.

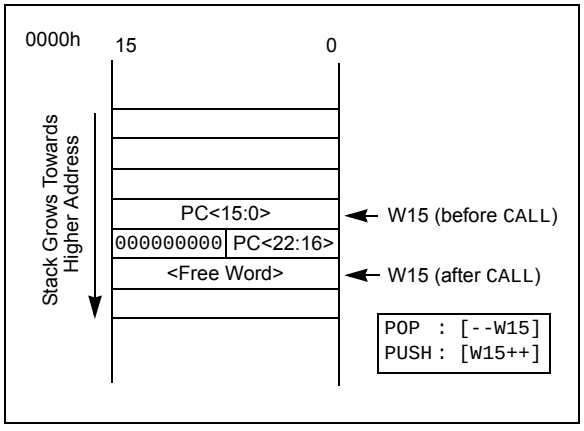
Note:	A PC push during exception processing will concatenate the SRL register to the MSB of the PC prior to the push.

The Stack Pointer Limit Value (SPLIM) register, associated with the Stack Pointer, sets an upper address boundary for the stack. SPLIM is uninitialized at Reset. As is the case for the Stack Pointer, SPLIM<0> is forced to '0' because all stack operations must be word-aligned. Whenever an EA is generated using W15 as a source or destination pointer, the resulting address is compared with the value in SPLIM. If the contents of the Stack Pointer (W15) and the SPLIM register are equal, and a push operation is performed, a stack error trap will not occur. The stack error trap will occur on a subsequent push operation. Thus, for example, if it is desirable to cause a stack error trap when the stack grows beyond address 2000h in RAM, initialize the SPLIM with the value, 1FFEh.

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

A write to the SPLIM register should not be immediately followed by an indirect read operation using W15.

FIGURE 4-4: CALL STACK FRAME



4.3 Interfacing Program and Data Memory Spaces

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

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

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

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

4.3.1 ADDRESSING PROGRAM SPACE

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

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

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

Table 4-30 and Figure 4-5 show how the program EA is created for table operations and remapping accesses from the data EA. Here, P<23:0> refers to a program space word, whereas D<15:0> refers to a data space word.

REGISTER 7-6: IFS1: INTERRUPT FLAG STATUS REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	—
bit 15							bit 8

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IC8IF	IC7IF	—	INT1IF	CNIF	CMIF	MI2C1IF	SI2C1IF
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 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 14 **U2RXIF:** UART2 Receiver Interrupt Flag Status bit
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 **T5IF:** Timer5 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

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

bit 10 **OC4IF:** Output Compare Channel 4 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 9 **OC3IF:** Output Compare Channel 3 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 8 **Unimplemented:** Read as '0'

bit 7 **IC8IF:** Input Capture Channel 8 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 6 **IC7IF:** Input Capture Channel 7 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 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 **MI2C1IF:** Master I2C1 Event Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 0 **SI2C1IF:** Slave I2C1 Event Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

REGISTER 7-14: IEC3: INTERRUPT ENABLE CONTROL REGISTER 3

U-0	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0
—	RTCIE	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	U-0
—	INT4IE(1)	INT3IE(1)	—	—	MI2C2IE	SI2C2IE	—
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 **Unimplemented:** Read as '0'

bit 14 **RTCIE:** Real-Time Clock/Calendar Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 13-7 **Unimplemented:** Read as '0'

bit 6 **INT4IE:** External Interrupt 4 Enable bit(1)
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 5 **INT3IE:** External Interrupt 3 Enable bit(1)
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 4-3 **Unimplemented:** Read as '0'

bit 2 **MI2C2IE:** Master I2C2 Event Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 1 **SI2C2IE:** Slave I2C2 Event Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

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

Note 1: If an external interrupt is enabled, the interrupt input must also be configured to an available RPn or RPIn pin. See **Section 10.4 "Peripheral Pin Select"** for more information.

REGISTER 7-18: IPC1: INTERRUPT PRIORITY CONTROL REGISTER 1

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T2IP2	T2IP1	T2IP0	—	OC2IP2	OC2IP1	OC2IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	IC2IP2	IC2IP1	IC2IP0	—	—	—	—
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 **Unimplemented:** Read as ‘0’

bit 14-12 **T2IP<2:0>:** Timer2 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **OC2IP<2:0>:** Output Compare Channel 2 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **IC2IP<2:0>:** Input Capture Channel 2 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 3-0 **Unimplemented:** Read as ‘0’

REGISTER 7-26: IPC9: INTERRUPT PRIORITY CONTROL REGISTER 9

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	IC5IP2	IC5IP1	IC5IP0	—	IC4IP2	IC4IP1	IC4IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	IC3IP2	IC3IP1	IC3IP0	—	—	—	—
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 **Unimplemented:** Read as '0'

bit 14-12 **IC5IP<2:0>:** Input Capture Channel 5 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 11 **Unimplemented:** Read as '0'

bit 10-8 **IC4IP<2:0>:** Input Capture Channel 4 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

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

bit 6-4 **IC3IP<2:0>:** Input Capture Channel 3 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 3-0 **Unimplemented:** Read as '0'

7.4 Interrupt Setup Procedures

7.4.1 INITIALIZATION

To configure an interrupt source:

1. Set the NSTDIS control bit (INTCON1<15>) if nested interrupts are not desired.
2. Select the user-assigned priority level for the interrupt source by writing the control bits in the appropriate IPCx register. The priority level will depend on the specific application and type of interrupt source. If multiple priority levels are not desired, the IPCx register control bits for all enabled interrupt sources may be programmed to the same non-zero value.

Note:	At a device Reset, the IPCx registers are initialized, such that all user interrupt sources are assigned to priority level 4.

3. Clear the interrupt status flag bit associated with the peripheral in the associated IFSx register.
4. Enable the interrupt source by setting the interrupt enable control bit associated with the source in the appropriate IECx register.

7.4.2 INTERRUPT SERVICE ROUTINE

The method that is used to declare an ISR and initialize the IVT with the correct vector address will depend on the programming language (i.e., 'C' or assembler) and the language development toolsuite that is used to develop the application. In general, the user must clear the interrupt flag in the appropriate IFSx register for the source of the interrupt that the ISR handles. Otherwise, the ISR will be re-entered immediately after exiting the routine. If the ISR is coded in assembly language, it must be terminated using a `RETFIE` instruction to unstack the saved PC value, SRL value and old CPU priority level.

7.4.3 TRAP SERVICE ROUTINE

A Trap Service Routine (TSR) is coded like an ISR, except that the appropriate trap status flag in the INTCON1 register must be cleared to avoid re-entry into the TSR.

7.4.4 INTERRUPT DISABLE

All user interrupts can be disabled using the following procedure:

1. Push the current SR value onto the software stack using the `PUSH` instruction.
2. Force the CPU to priority level 7 by inclusive ORing the value E0h with SRL.

To enable user interrupts, the `POP` instruction may be used to restore the previous SR value.

Note that only user interrupts with a priority level of 7 or less can be disabled. Trap sources (level 8-15) cannot be disabled.

The `DISI` instruction provides a convenient way to disable interrupts of priority levels 1-6 for a fixed period of time. Level 7 interrupt sources are not disabled by the `DISI` instruction.

10.4.5 CONSIDERATIONS FOR PERIPHERAL PIN SELECTION

The ability to control Peripheral Pin Select options introduces several considerations into application design that could be overlooked. This is particularly true for several common peripherals that are available only as remappable peripherals.

The main consideration is that the Peripheral Pin Selects are not available on default pins in the device's default (Reset) state. Since all RPINRx registers reset to '111111' and all RPORx registers reset to '000000', all Peripheral Pin Select inputs are tied to Vss and all Peripheral Pin Select outputs are disconnected.

Note:	In tying Peripheral Pin Select inputs to RP63, RP63 does not have to exist on a device for the registers to be reset to it.

This situation requires the user to initialize the device with the proper peripheral configuration before any other application code is executed. Since the IOLOCK bit resets in the unlocked state, it is not necessary to execute the unlock sequence after the device has come out of Reset. For application safety, however, it is best to set IOLOCK and lock the configuration after writing to the control registers.

Because the unlock sequence is timing critical, it must be executed as an assembly language routine in the same manner as changes to the oscillator configuration. If the bulk of the application is written in C or another high-level language, the unlock sequence should be performed by writing in-line assembly.

Choosing the configuration requires the review of all Peripheral Pin Selects and their pin assignments, especially those that will not be used in the application. In all cases, unused pin-selectable peripherals should be disabled completely. Unused peripherals should have their inputs assigned to an unused RPn pin function. I/O pins with unused RPn functions should be configured with the null peripheral output.

The assignment of a peripheral to a particular pin does not automatically perform any other configuration of the pin's I/O circuitry. In theory, this means adding a pin-selectable output to a pin may mean inadvertently driving an existing peripheral input when the output is driven. Users must be familiar with the behavior of other fixed peripherals that share a remappable pin and know when to enable or disable them. To be safe, fixed digital peripherals that share the same pin should be disabled when not in use.

Along these lines, configuring a remappable pin for a specific peripheral does not automatically turn that feature on. The peripheral must be specifically configured for operation and enabled, as if it were tied to a fixed pin. Where this happens in the application code (immediately following device Reset and peripheral configuration or inside the main application routine) depends on the peripheral and its use in the application.

A final consideration is that Peripheral Pin Select functions neither override analog inputs, nor reconfigure pins with analog functions for digital I/O. If a pin is configured as an analog input on device Reset, it must be explicitly reconfigured as digital I/O when used with a Peripheral Pin Select.

Example 10-2 shows a configuration for bidirectional communication with flow control using UART1. The following input and output functions are used:

- Input Functions: U1RX, $\overline{\text{U1CTS}}$
- Output Functions: U1TX, $\overline{\text{U1RTS}}$

EXAMPLE 10-2: CONFIGURING UART1 INPUT AND OUTPUT FUNCTIONS

```
// Unlock Registers
__builtin_write_OSCCONL(OSCCON & 0xBF);

// Configure Input Functions (Table 9-1))
    // Assign U1RX To Pin RP0
    RPINR18bits.U1RXR = 0;

    // Assign U1CTS To Pin RP1
    RPINR18bits.U1CTSR = 1;

// Configure Output Functions (Table 9-2)
    // Assign U1TX To Pin RP2
    RPOR1bits.RP2R = 3;

    // Assign U1RTS To Pin RP3
    RPOR1bits.RP3R = 4;

// Lock Registers
__builtin_write_OSCCONL(OSCCON | 0x40);
```

REGISTER 10-28: RPOR6: PERIPHERAL PIN SELECT OUTPUT REGISTER 6

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP13R5	RP13R4	RP13R3	RP13R2	RP13R1	RP13R0
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
—	—	RP12R5	RP12R4	RP12R3	RP12R2	RP12R1	RP12R0
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 15-14 **Unimplemented:** Read as '0'

bit 13-8 **RP13R<5:0>:** RP13 Output Pin Mapping bits
Peripheral output number n is assigned to pin, RP13 (see Table 10-3 for peripheral function numbers).

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **RP12R<5:0>:** RP12 Output Pin Mapping bits
Peripheral output number n is assigned to pin, RP12 (see Table 10-3 for peripheral function numbers).

REGISTER 10-29: RPOR7: PERIPHERAL PIN SELECT OUTPUT REGISTER 7

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP15R5(1)	RP15R4(1)	RP15R3(1)	RP15R2(1)	RP15R1(1)	RP15R0(1)
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
—	—	RP14R5	RP14R4	RP14R3	RP14R2	RP14R1	RP14R0
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 15-14 **Unimplemented:** Read as '0'

bit 13-8 **RP15R<5:0>:** RP15 Output Pin Mapping bits(1)
Peripheral output number n is assigned to pin, RP15 (see Table 10-3 for peripheral function numbers).

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **RP14R<5:0>:** RP14 Output Pin Mapping bits
Peripheral output number n is assigned to pin, RP14 (see Table 10-3 for peripheral function numbers).

Note 1: Unimplemented in 64-pin devices; read as '0'.

NOTES:

14.0 OUTPUT COMPARE WITH DEDICATED TIMER

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, refer to the *"PIC24F Family Reference Manual"*, **Section 35. "Output Compare with Dedicated Timer"** (DS39723)

Devices in the PIC24FJ256GA110 family all feature 9 independent enhanced output compare modules. Each of these modules offers a wide range of configuration and operating options for generating pulse trains on internal device events, and can produce Pulse-Width Modulated (PWM) waveforms for driving power applications.

Key features of the enhanced output compare module include:

- Hardware-configurable for 32-bit operation in all modes by cascading two adjacent modules
- Synchronous and Trigger modes of output compare operation, with up to 30 user-selectable trigger/sync sources available
- Two separate Period registers (a main register, OCxR, and a secondary register, OCxRS) for greater flexibility in generating pulses of varying widths
- Configurable for single pulse or continuous pulse generation on an output event, or continuous PWM waveform generation
- Up to 6 clock sources available for each module, driving a separate internal 16-bit counter

14.1 General Operating Modes

14.1.1 SYNCHRONOUS AND TRIGGER MODES

By default, the enhanced output compare module operates in a free-running mode. The internal, 16-bit counter, OCxTMR, counts up continuously, wrapping around from FFFFh to 0000h on each overflow, with its period synchronized to the selected external clock source. Compare or PWM events are generated each time a match between the internal counter and one of the Period registers occurs.

In Synchronous mode, the module begins performing its compare or PWM operation as soon as its selected clock source is enabled. Whenever an event occurs on the selected sync source, the module's internal counter is reset. In Trigger mode, the module waits for a sync event from another internal module to occur before allowing the counter to run.

Free-running mode is selected by default, or any time that the SYNCSEL bits (OCxCON2<4:0>) are set to '00000'. Synchronous or Trigger modes are selected any time the SYNCSEL bits are set to any value except '00000'. The OCTRIG bit (OCxCON2<7>) selects either Synchronous or Trigger mode; setting the bit selects Trigger mode operation. In both modes, the SYNCSEL bits determine the sync/trigger source.

14.1.2 CASCADED (32-BIT) MODE

By default, each module operates independently with its own set of 16-Bit Timer and Duty Cycle registers. To increase resolution, adjacent even and odd modules can be configured to function as a single 32-bit module. (For example, modules 1 and 2 are paired, as are modules 3 and 4, and so on.) The odd-numbered module (OCx) provides the Least Significant 16 bits of the 32-bit register pairs, and the even module (OCy) provides the Most Significant 16 bits. Wraparounds of the OCx registers cause an increment of their corresponding OCy registers.

Cascaded operation is configured in hardware by setting the OC32 bits (OCxCON2<8>) for both modules.

REGISTER 16-1: I2CxCON: I2Cx 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:** I2Cx Enable bit
1 = Enables the I2Cx module and configures the SDAx and SCLx pins as serial port pins
0 = Disables I2Cx module. All I^2C 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:** SCLx Release Control bit (when operating as I^2C Slave)
1 = Releases SCLx clock
0 = Holds SCLx 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 clear at beginning of slave transmission. Hardware clear at end of slave reception.
If STREN = 0:
Bit is R/S (i.e., software may only write '1' to release clock). Hardware clear at beginning of slave transmission.

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

bit 10 **A10M:** 10-Bit Slave Addressing bit
1 = I2CxADD is a 10-bit slave address
0 = I2CxADD is a 7-bit slave address

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

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

bit 7 **GCEN:** General Call Enable bit (when operating as I^2C slave)
1 = Enables interrupt when a general call address is received in the I2CxRSR (module is enabled for reception)
0 = General call address disabled

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

19.1 RTCC Module Registers

The RTCC module registers are organized into three categories:

- RTCC Control Registers
- RTCC Value Registers
- Alarm Value Registers

19.1.1 REGISTER MAPPING

To limit the register interface, the RTCC Timer and Alarm Time registers are accessed through corresponding register pointers. The RTCC Value register window (RTCVALH and RTCVALL) uses the RTCPTR bits (RCFGCAL<9:8>) to select the desired Timer register pair (see Table 19-1).

By writing to the RTCVALH byte, the RTCC Pointer value, RTCPTR<1:0> bits, decrement by one until they reach '00'. Once they reach '00', the MINUTES and SECONDS value will be accessible through RTCVALH and RTCVALL until the pointer value is manually changed.

TABLE 19-1: RTCVAL REGISTER MAPPING

RTCPTR <1:0>	RTCC Value Register Window	
	RTCVAL<15:8>	RTCVAL<7:0>
00	MINUTES	SECONDS
01	WEEKDAY	HOURS
10	MONTH	DAY
11	—	YEAR

The Alarm Value register window (ALRMVALH and ALRMVALL) uses the ALRMPTR bits (ALCFGRPT<9:8>) to select the desired Alarm register pair (see Table 19-2).

By writing to the ALRMVALH byte, the Alarm Pointer value, ALRMPTR<1:0> bits, decrement by one until they reach '00'. Once they reach '00', the ALRMMIN and ALRMSEC value will be accessible through ALRMVALH and ALRMVALL until the pointer value is manually changed.

TABLE 19-2: ALRMVAL REGISTER MAPPING

ALRMPTR <1:0>	Alarm Value Register Window	
	ALRMVAL<15:8>	ALRMVAL<7:0>
00	ALRMMIN	ALRMSEC
01	ALRMWD	ALRMHR
10	ALRMMNTH	ALRMDAY
11	—	—

Considering that the 16-bit core does not distinguish between 8-bit and 16-bit read operations, the user must be aware that when reading either the ALRMVALH or ALRMVALL bytes will decrement the ALRMPTR<1:0> value. The same applies to the RTCVALH or RTCVALL bytes with the RTCPTR<1:0> being decremented.

Note: This only applies to read operations and not write operations.

19.1.2 WRITE LOCK

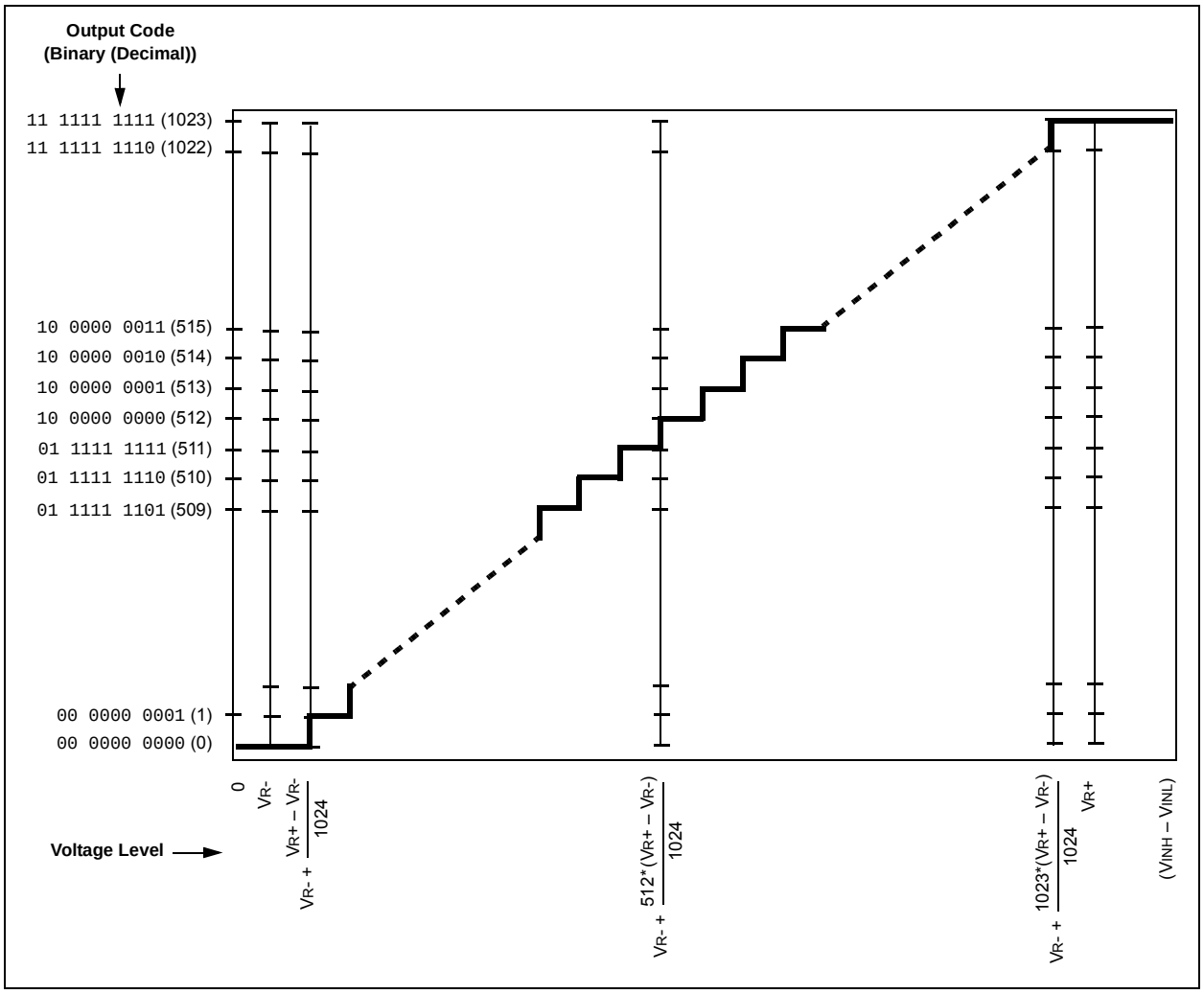
In order to perform a write to any of the RTCC Timer registers, the RTCWREN bit (RCFGCAL<13>) must be set (refer to Example 19-1).

Note: To avoid accidental writes to the timer, it is recommended that the RTCWREN bit (RCFGCAL<13>) is kept clear at any other time. For the RTCWREN bit to be set, there is only 1 instruction cycle time window allowed between the unlock sequence and the setting of RTCWREN; therefore, it is recommended that code follow the procedure in Example 19-1.
For applications written in C, the unlock sequence should be implemented using in-line assembly.

EXAMPLE 19-1: SETTING THE RTCWREN BIT

```
asm volatile("disi #5");
__builtin_write_RTCWEN();
```

FIGURE 21-3: A/D TRANSFER FUNCTION



24.0 CHARGE TIME MEASUREMENT UNIT (CTMU)

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, refer to the *"PIC24F Family Reference Manual"*, **Section 11. "Charge Time Measurement Unit (CTMU)"** (DS39724).

The Charge Time Measurement Unit is a flexible analog module that provides accurate differential time measurement between pulse sources, as well as asynchronous pulse generation. Its key features include:

- Four edge input trigger sources
- Polarity control for each edge source
- Control of edge sequence
- Control of response to edges
- Time measurement resolution of 1 nanosecond
- Accurate current source suitable for capacitive measurement

Together with other on-chip analog modules, the CTMU can be used to precisely measure time, measure capacitance, measure relative changes in capacitance or generate output pulses that are independent of the system clock. The CTMU module is ideal for interfacing with capacitive-based sensors.

The CTMU is controlled through two registers: CTMUCON and CTMUICON. CTMUCON enables the module and controls edge source selection, edge source polarity selection, and edge sequencing. The CTMUICON register controls the selection and trim of the current source.

24.1 Measuring Capacitance

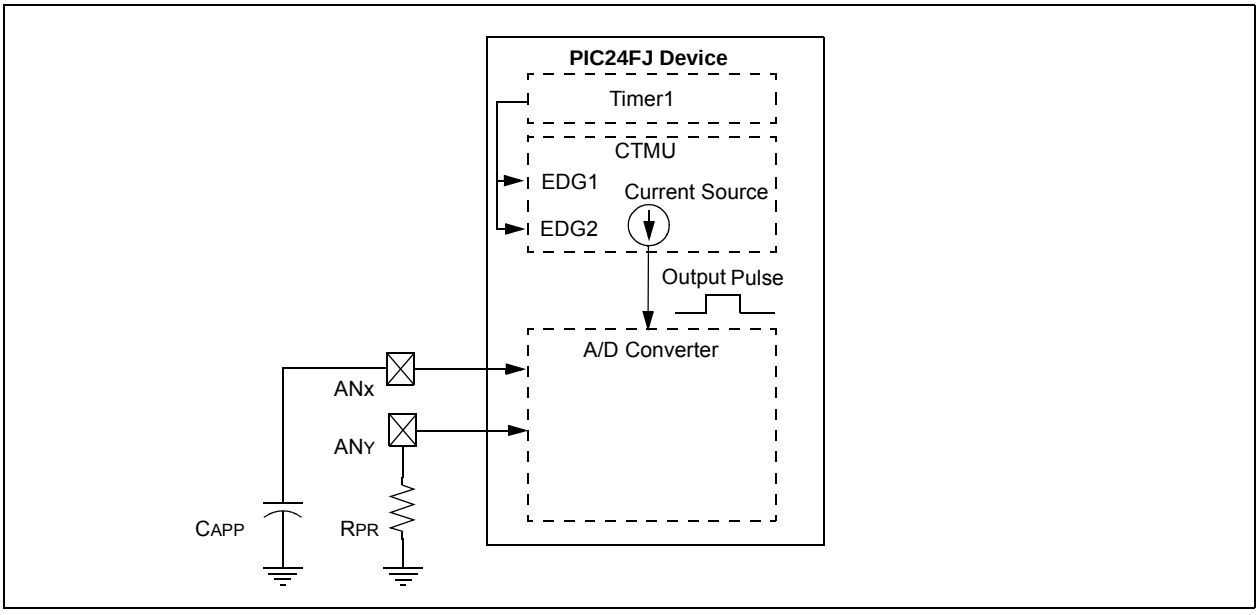
The CTMU module measures capacitance by generating an output pulse, with a width equal to the time, between edge events on two separate input channels. The pulse edge events to both input channels can be selected from four sources: two internal peripheral modules (OC1 and Timer1) and two external pins (CTEDG1 and CTEDG2). This pulse is used with the module's precision current source to calculate capacitance according to the relationship

$$I = C \bullet \frac{dV}{dT}$$

For capacitance measurements, the A/D Converter samples an external capacitor (C_{APP}) on one of its input channels after the CTMU output's pulse. A Precision Resistor (R_{PR}) provides current source calibration on a second A/D channel. After the pulse ends, the converter determines the voltage on the capacitor. The actual calculation of capacitance is performed in software by the application.

Figure 24-1 shows the external connections used for capacitance measurements, and how the CTMU and A/D modules are related in this application. This example also shows the edge events coming from Timer1, but other configurations using external edge sources are possible. A detailed discussion on measuring capacitance and time with the CTMU module is provided in the *"PIC24F Family Reference Manual"*.

FIGURE 24-1: TYPICAL CONNECTIONS AND INTERNAL CONFIGURATION FOR CAPACITANCE MEASUREMENT



25.4.2 CODE SEGMENT PROTECTION

In addition to global General Segment protection, a separate subrange of the program memory space can be individually protected against writes and erases. This area can be used for many purposes where a separate block of erase and write-protected code is needed, such as bootloader applications. Unlike common boot block implementations, the specially protected segment in the PIC24FJ256GA110 family devices can be located by the user anywhere in the program space and configured in a wide range of sizes.

Code segment protection provides an added level of protection to a designated area of program memory by disabling the NVM safety interlock whenever a write or erase address falls within a specified range. It does not override General Segment protection controlled by the GCP or GWRP bits. For example, if GCP and GWRP are enabled, enabling segmented code protection for the bottom half of program memory does not undo General Segment protection for the top half.

The size and type of protection for the segmented code range are configured by the WPFPx, WPEND, WPCFG and WPDIS bits in Flash Configuration Word 3. Code segment protection is enabled by programming the WPDIS bit (= 0). The WPFP bits specify the size of the segment to be protected by specifying the 512-word code page that is the start or end of the protected segment. The specified region is inclusive, therefore, this page will also be protected.

The WPEND bit determines if the protected segment uses the top or bottom of the program space as a boundary. Programming WPEND (= 0) sets the bottom of program memory (000000h) as the lower boundary of the protected segment. Leaving WPEND unprogrammed (= 1) protects the specified page through the last page of implemented program memory, including the Configuration Word locations.

A separate bit, WPCFG, is used to independently protect the last page of program space, including the Flash Configuration Words. If WPEND is set to protect the bottom of program memory, programming WPCFG (= 0) protects the last page. This may be useful in circumstances where write protection is needed for both a code segment in the bottom of memory, as well as the Flash Configuration Words.

The various options for segment code protection are shown in Table 25-2.

25.4.3 CONFIGURATION REGISTER PROTECTION

The Configuration registers are protected against inadvertent or unwanted changes, or reads in two ways. The primary protection method is the same as that of the RP registers – shadow registers contain a complimentary value which is constantly compared with the actual value.

To safeguard against unpredictable events, Configuration bit changes resulting from individual cell level disruptions (such as ESD events) will cause a parity error and trigger a device Reset.

The data for the Configuration registers is derived from the Flash Configuration Words in program memory. When the GCP bit is set, the source data for device configuration is also protected as a consequence. Even if General Segment protection is not enabled, the device configuration can be protected by using the appropriate code segment protection setting.

TABLE 25-2: SEGMENT CODE PROTECTION CONFIGURATION OPTIONS

Segment Configuration Bits			Write/Erase Protection of Code Segment
WPDIS	WPEND	WPCFG	
1	x	x	No additional protection enabled; all program memory protection is configured by GCP and GWRP
0	1	x	Addresses from the first address of code page, defined by WPFP<7:0> through the end of implemented program memory (inclusive), are write/erase protected including Flash Configuration Words
0	0	1	Address, 000000h through the last address of code page, defined by WPFP<7:0> (inclusive), is protected
0	0	0	Address, 000000h through the last address of code page, defined by WPFP<7:0> (inclusive) are write/erase protected and the last page is also write/erase protected.

NOTES: