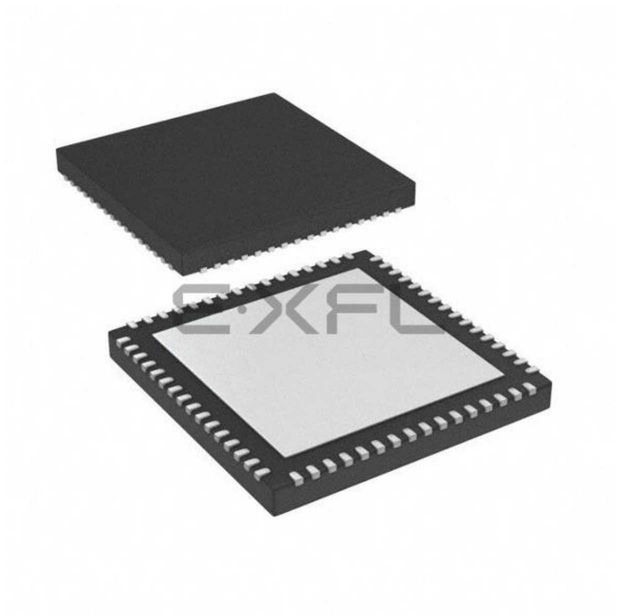


What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	16-Bit
Speed	32MHz
Connectivity	I²C, IrDA, LINbus, PMP, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, HLVD, POR, PWM, WDT
Number of I/O	53
Program Memory Size	512KB (170K x 24)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	32K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-VFQFN Exposed Pad
Supplier Device Package	64-QFN (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic24fj512ga606-i-mr

4.2.2 DATA MEMORY ORGANIZATION AND ALIGNMENT

To maintain backward compatibility with PIC® MCUs and improve Data Space memory usage efficiency, the PIC24F instruction set supports both word and byte operations. As a consequence of byte accessibility, all EA calculations are internally scaled to step through word-aligned memory. For example, the core recognizes that Post-Modified Register Indirect Addressing mode, [Ws++], will result in a value of Ws + 1 for byte operations and Ws + 2 for word operations.

Data byte reads will read the complete word, which contains the byte, using the LSB of any EA to determine which byte to select. The selected byte is placed onto the LSB of the data path. That is, data memory and registers are organized as two parallel, byte-wide entities with shared (word) address decode, but separate write lines. Data byte writes only write to the corresponding side of the array or register which matches the byte address.

All word accesses must be aligned to an even address. Misaligned word data fetches are not supported, so care must be taken when mixing byte and word operations or translating from 8-bit MCU code. If a misaligned read or write is attempted, an address error trap will be generated. If the error occurred on a read, the instruction underway is completed; if it occurred on a write, the instruction will be executed but the write will not occur. In either case, a trap is then executed, allowing the system and/or user to examine the machine state prior to execution of the address Fault.

All byte loads into any W register are loaded into the LSB. The Most Significant Byte (MSB) is not modified.

A Sign-Extend (`SE`) instruction is provided to allow users to translate 8-bit signed data to 16-bit signed values. Alternatively, for 16-bit unsigned data, users can clear the MSB of any W register by executing a Zero-Extend (`ZE`) instruction on the appropriate address.

Although most instructions are capable of operating on word or byte data sizes, it should be noted that some instructions operate only on words.

4.2.3 NEAR DATA SPACE

The 8-Kbyte area between 0000h and 1FFFh is referred to as the Near Data Space. Locations in this space are directly addressable via a 13-bit absolute address field within all memory direct instructions. The remainder of the Data Space is addressable indirectly. Additionally, the whole Data Space is addressable using `MOV` instructions, which support Memory Direct Addressing with a 16-bit address field.

4.2.4 SPECIAL FUNCTION REGISTER (SFR) SPACE

The first 2 Kbytes of the Near Data Space, from 0000h to 07FFh, are primarily occupied with Special Function Registers (SFRs). These are used by the PIC24F core and peripheral modules for controlling the operation of the device.

SFRs are distributed among the modules that they control and are generally grouped together by module. Much of the SFR space contains unused addresses; these are read as '0'. A diagram of the SFR space, showing where the SFRs are actually implemented, is shown in Table 4-3. Each implemented area indicates a 32-byte region where at least one address is implemented as an SFR. A complete list of implemented SFRs, including their addresses, is shown in Tables 4-3 through 4-11.

TABLE 4-3: IMPLEMENTED REGIONS OF SFR DATA SPACE

	SFR Space Address															
	xx00	**xx10**	**xx20**	**xx30**	**xx40**	**xx50**	**xx60**	**xx70**	**xx80**	**xx90**	**xxA0**	**xxB0**	**xxC0**	**xxD0**	**xxE0**	**xxF0**
000h	Core															
100h	OSC	Reset[(1)]	EPMP			CRC	REFO	PMD		Timers			CTM	RTCC		
200h	Capture			Compare			MCCP								Comp	ANCFG
300h	SCCP									UART						SPI
400h	SPI					—	CLC			I^2C				DMA		
500h	DMA		—	—	—	USB						—	—	—	—	—
600h	—	—	—	—	—	I/O										—
700h	—	A/D					—	—	—	PPS						

Legend: — = No implemented SFRs in this block

Note 1: Includes HLVD control.

TABLE 4-8: SFR MAP: 0400h BLOCK (CONTINUED)

File Name	Address	All Resets	File Name	Address	All Resets
DMA (CONTINUED)			**DMA (CONTINUED)**		
DMASRC0	04D0	0000	DMACNT2	04E8	0001
DMADST0	04D2	0000	DMACH3	04EA	0000
DMACNT0	04D4	0001	DMAINT3	04EC	0000
DMACH1	04D6	0000	DMASRC3	04EE	0000
DMAINT1	04D8	0000	DMADST3	04F0	0000
DMASRC1	04DA	0000	DMACNT3	04F2	0001
DMADST1	04DC	0000	DMACH4	04F4	0000
DMACNT1	04DE	0001	DMAINT4	04F6	0000
DMACH2	04E0	0000	DMASRC4	04F8	0000
DMAINT2	04E2	0000	DMADST4	04FA	0000
DMASRC2	04E4	0000	DMACNT4	04FC	0001
DMADST2	04E6	0000	DMACH5	04FE	0000

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

4.2.5.2 Data Write into EDS

In order to write data to EDS, such as in EDS reads, an Address Pointer is set up by loading the required EDS page number into the DSWPAG register and assigning the offset address to one of the W registers. Once the above assignment is done, then the EDS window is enabled by setting bit 15 of the Working register, assigned with the offset address and the accessed location can be written.

Figure 4-2 illustrates how the EDS address is generated for write operations.

When the MSbs of EA are '1', the lower 9 bits of DSWPAG are concatenated to the lower 15 bits of EA to form a 24-bit EDS address for write operations. Example 4-2 shows how to write a byte, word and double word to EDS.

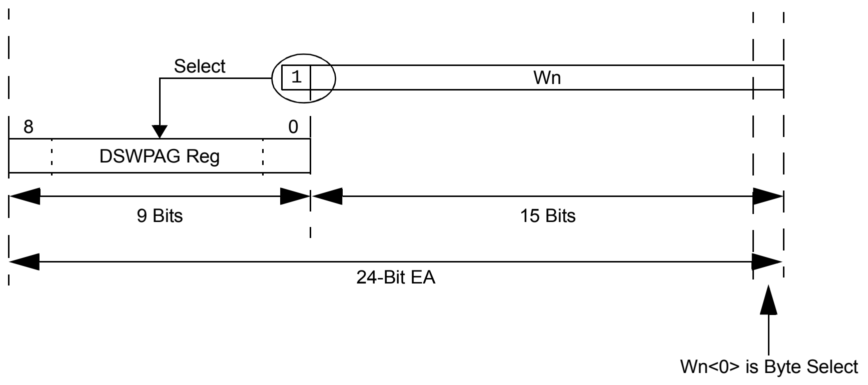
The Data Space Page registers (DSRPAG/DSWPAG) do not update automatically while crossing a page boundary when the rollover happens from 0xFFFF to 0x8000. While developing code in assembly, care must be taken to update the Data Space Page registers when an Address Pointer crosses the page boundary. The 'C' compiler keeps track of the addressing, and increments or decrements the Page registers accordingly, while accessing contiguous data memory locations.

Note 1: All write operations to EDS are executed in a single cycle.

2: Use of Read/Modify/Write operation on any EDS location under a `REPEAT` instruction is not supported. For example, `BCLR`, `BSW`, `BTG`, `RLC f`, `RLNC f`, `RRC f`, `RRNC f`, `ADD f`, `SUB f`, `SUBR f`, `AND f`, `IOR f`, `XOR f`, `ASR f`, `ASL f`.

3: Use the DSRPAG register while performing Read/Modify/Write operations.

FIGURE 4-6: EDS ADDRESS GENERATION FOR WRITE OPERATIONS



EXAMPLE 4-2: EDS WRITE CODE IN ASSEMBLY

```
; Set the EDS page where the data to be written
    mov     #0x0002, w0
    mov     w0, DSWPAG      ;page 2 is selected for write
    mov     #0x0800, w1     ;select the location (0x800) to be written
    bset    w1, #15         ;set the MSB of the base address, enable EDS mode

;Write a byte to the selected location
    mov     #0x00A5, w2
    mov     #0x003C, w3
    mov.b   w2, [w1++]      ;write Low byte
    mov.b   w3, [w1++]      ;write High byte

;Write a word to the selected location
    mov     #0x1234, w2     ;
    mov     w2, [w1]        ;

;Write a Double - word to the selected location
    mov     #0x1122, w2
    mov     #0x4455, w3
    mov.d   w2, [w1]        ;2 EDS writes
```

REGISTER 5-1: DMACON: DMA ENGINE CONTROL REGISTER

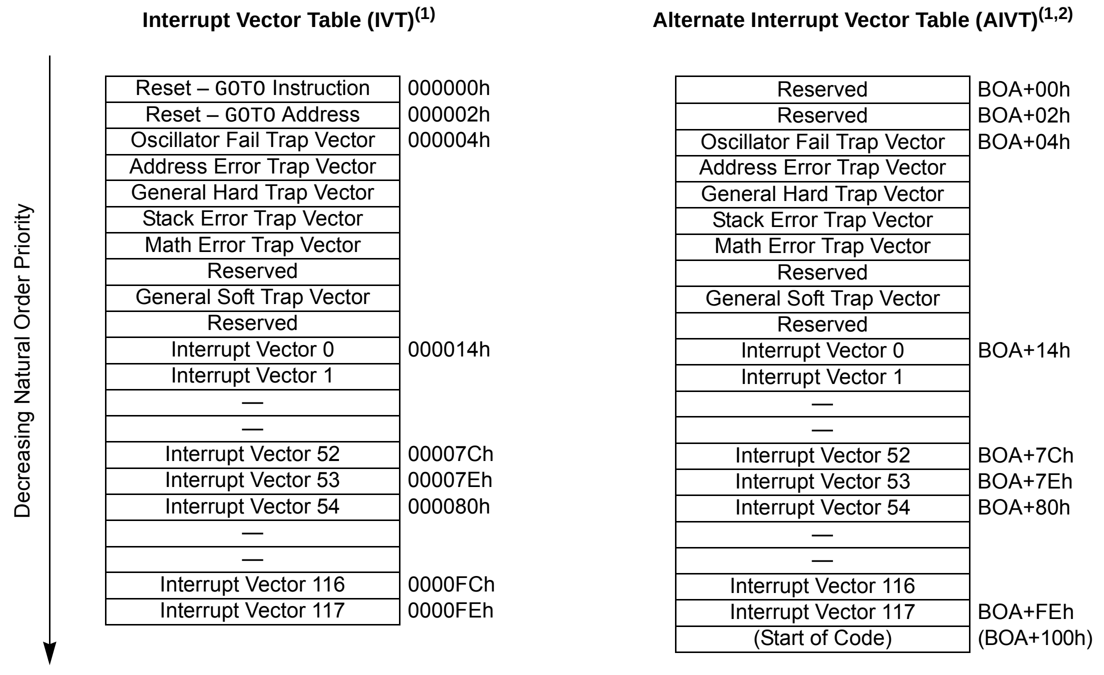
R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
DMAEN	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	PRSSEL
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	**DMAEN:** DMA Module Enable bit `1` = Enables module `0` = Disables module and terminates all active DMA operation(s)
bit 14-1	**Unimplemented:** Read as ‘`0`’
bit 0	**PRSSEL:** Channel Priority Scheme Selection bit `1` = Round-robin scheme `0` = Fixed priority scheme

FIGURE 8-1: PIC24FJ1024GA610/GB610 FAMILY INTERRUPT VECTOR TABLES



Interrupt Vector Table (IVT)[1]

Vector	Address
Reset – GOTO Instruction	000000h
Reset – GOTO Address	000002h
Oscillator Fail Trap Vector	000004h
Address Error Trap Vector	
General Hard Trap Vector	
Stack Error Trap Vector	
Math Error Trap Vector	
Reserved	
General Soft Trap Vector	
Reserved	
Interrupt Vector 0	000014h
Interrupt Vector 1	
—	
—	
Interrupt Vector 52	00007Ch
Interrupt Vector 53	00007Eh
Interrupt Vector 54	000080h
—	
—	
Interrupt Vector 116	0000FCh
Interrupt Vector 117	0000FEh

Alternate Interrupt Vector Table (AIVT)[1,2]

Vector	Address
Reserved	BOA+00h
Reserved	BOA+02h
Oscillator Fail Trap Vector	BOA+04h
Address Error Trap Vector	
General Hard Trap Vector	
Stack Error Trap Vector	
Math Error Trap Vector	
Reserved	
General Soft Trap Vector	
Reserved	
Interrupt Vector 0	BOA+14h
Interrupt Vector 1	
—	
—	
Interrupt Vector 52	BOA+7Ch
Interrupt Vector 53	BOA+7Eh
Interrupt Vector 54	BOA+80h
—	
—	
Interrupt Vector 116	
Interrupt Vector 117	BOA+FEh
(Start of Code)	(BOA+100h)

Legend: BOA: Base Offset Address for AIVT, which is the starting address of the last page of the Boot Segment. All addresses are in hexadecimal.

Note 1: See Table 8-2 for the interrupt vector list.

2: AIVT is only available when a Boot Segment is implemented.

TABLE 8-1: TRAP VECTOR DETAILS

Vector Number	IVT Address	AIVT Address	Trap Source
0	000004h	BOA+04h	Oscillator Failure
1	000006h	BOA+06h	Address Error
2	000008h	BOA+08h	General Hardware Error
3	00000Ah	BOA+0Ah	Stack Error
4	00000Ch	BOA+0Ch	Math Error
5	00000Eh	BOA+0Eh	Reserved
6	000010h	BOA+10h	General Software Error
7	000012h	BOA+12h	Reserved

Legend: BOA = Base Offset Address for the AIVT segment, which is the starting address of the last page of the Boot Segment.

The BOA depends on the size of the Boot Segment defined by $\overline{\text{BSLIM}}$<12:0>:
[($\overline{\text{BSLIM}}$<12:0> – 1) x 0x800]

REGISTER 8-6: INTTREG: INTERRUPT CONTROL AND STATUS REGISTER

R-0	U-0	R/W-0	U-0	R-0	R-0	R-0	R-0
CPUIRQ	—	VHOLD	—	ILR3	ILR2	ILR1	ILR0
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
VECNUM7	VECNUM6	VECNUM5	VECNUM4	VECNUM3	VECNUM2	VECNUM1	VECNUM0
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 **CPUIRQ:** Interrupt Request from Interrupt Controller CPU bit
1 = An interrupt request has occurred but has not yet been Acknowledged by the CPU; this happens when the CPU priority is higher than the interrupt priority
0 = No interrupt request is unacknowledged

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **VHOLD:** Vector Number Capture Configuration bit
1 = The VECNUMx bits contain the value of the highest priority pending interrupt
0 = The VECNUMx bits contain the value of the last Acknowledged interrupt (i.e., the last interrupt that has occurred with higher priority than the CPU, even if other interrupts are pending)

bit 12 **Unimplemented:** Read as ‘0’

bit 11-8 **ILR<3:0>:** New CPU Interrupt Priority Level bits
1111 = CPU Interrupt Priority Level is 15
•
•
•
0001 = CPU Interrupt Priority Level is 1
0000 = CPU Interrupt Priority Level is 0

bit 7-0 **VECNUM<7:0>:** Vector Number of Pending Interrupt bits
11111111 = 255, Reserved; do not use
•
•
•
00001001 = 9, IC1 – Input Capture 1
00001000 = 8, INT0 – External Interrupt 0
00000111 = 7, Reserved; do not use
00000110 = 6, Generic soft error trap
00000101 = 5, Reserved; do not use
00000100 = 4, Math error trap
00000011 = 3, Stack error trap
00000010 = 2, Generic hard trap
00000001 = 1, Address error trap
00000000 = 0, Oscillator fail trap

REGISTER 9-1: OSCCON: OSCILLATOR CONTROL REGISTER

U-0	R-x[(1)]	R-x[(1)]	R-x[(1)]	U-0	R/W-x[(1)]	R/W-x[(1)]	R/W-x[(1)]
—	COSC2	COSC1	COSC0	—	NOSC2	NOSC1	NOSC0
bit 15							bit 8

R/W-0	R/W-0	R-0[(3)]	U-0	R/CO-0	R/W-0	R/W-0	R/W-0
CLKLOCK	IOLOCK[(2)]	LOCK	—	CF	POSCEN	SOSCEN	OSWEN
bit 7							bit 0

Legend:	CO = Clearable Only 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 **Unimplemented:** Read as ‘0’

bit 14-12 **COSC<2:0>:** Current Oscillator Selection bits[(1)]
111 = Oscillator with Frequency Divider (OSCFDIV)
110 = Digitally Controlled Oscillator (DCO)
101 = Low-Power RC Oscillator (LPRC)
100 = Secondary Oscillator (SOSC)
011 = Primary Oscillator with PLL module (XTPLL, ECPLL)
010 = Primary Oscillator (XT, HS, EC)
001 = Fast RC Oscillator with PLL module (FRCPLL)
000 = Fast RC Oscillator (FRC)

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **NOSC<2:0>:** New Oscillator Selection bits[(1)]
111 = Oscillator with Frequency Divider (OSCFDIV)
110 = Digitally Controlled Oscillator (DCO)
101 = Low-Power RC Oscillator (LPRC)
100 = Secondary Oscillator (SOSC)
011 = Primary Oscillator with PLL module (XTPLL, ECPLL)
010 = Primary Oscillator (XT, HS, EC)
001 = Fast RC Oscillator with PLL module (FRCPLL)
000 = Fast RC Oscillator (FRC)

bit 7 **CLKLOCK:** Clock Selection Lock Enable bit
If FSCM is Enabled (FCKSM<1:0> = 00):
1 = Clock and PLL selections are locked
0 = Clock and PLL selections are not locked and may be modified by setting the OSWEN bit
If FSCM is Disabled (FCKSM<1:0> = 1x):
Clock and PLL selections are never locked and may be modified by setting the OSWEN bit.

bit 6 **IOLOCK:** I/O Lock Enable bit[(2)]
1 = I/O lock is active
0 = I/O lock is not active

bit 5 **LOCK:** PLL Lock Status bit[(3)]
1 = PLL module is in lock or PLL module start-up timer is satisfied
0 = PLL module is out of lock, PLL start-up timer is running or PLL is disabled

bit 4 **Unimplemented:** Read as ‘0’

Note 1: Reset values for these bits are determined by the FNOSCx Configuration bits.
2: The state of the IOLOCK bit can only be changed once an unlocking sequence has been executed. In addition, if the IOL1WAY Configuration bit is ‘1’, once the IOLOCK bit is set, it cannot be cleared.
3: This bit also resets to ‘0’ during any valid clock switch or whenever a non-PLL Clock mode is selected.

REGISTER 9-6: OSCDIV: OSCILLATOR DIVISOR REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	DIV<14:8>						
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
DIV<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **Unimplemented:** Read as ‘0’

bit 14-0 **DIV<14:0>:** Reference Clock Divider bits

Specifies the 1/2 period of the reference clock in the source clocks
(ex: Period of ref_clk_output = [Reference Source * 2] * DIV<14:0>).
111111111111111 = Oscillator frequency divided by 65,534 (32,767 * 2)
111111111111110 = Oscillator frequency divided by 65,532 (32,766 * 2)
•
•
•
000000000000011 = Oscillator frequency divided by 6 (3 * 2)
000000000000010 = Oscillator frequency divided by 4 (2 * 2)
000000000000001 = Oscillator frequency divided by 2 (1 * 2) (default)
000000000000000 = Oscillator frequency is unchanged (no divider)

9.4 Clock Switching Operation

With few limitations, applications are free to switch between any of the five clock sources (POSC, SOSC, FRC, DCO and LPRC) under software control and at any time. To limit the possible side effects that could result from this flexibility, PIC24F devices have a safeguard lock built into the switching process.

Note:	The Primary Oscillator mode has three different submodes (XT, HS and EC), which are determined by the POSCMD<1:0> Configuration bits. While an application can switch to and from Primary Oscillator mode in software, it cannot switch between the different primary submodes without reprogramming the device.

9.4.1 ENABLING CLOCK SWITCHING

To enable clock switching, the FCKSM<1> Configuration bit in FOSC must be programmed to '0'. (Refer to **Section 30.1 "Configuration Bits"** for further details.) If the FCKSM<1> Configuration bit is unprogrammed ('1'), the clock switching function and Fail-Safe Clock Monitor function are disabled; this is the default setting.

The NOSC<2:0> control bits (OSCCON<10:8>) do not control the clock selection when clock switching is disabled. However, the COSC<2:0> bits (OSCCON<14:12>) will reflect the clock source selected by the FNOSC<2:0> Configuration bits.

The OSWEN control bit (OSCCON<0>) has no effect when clock switching is disabled; it is held at '0' at all times.

9.4.2 OSCILLATOR SWITCHING SEQUENCE

At a minimum, performing a clock switch requires this basic sequence:

1. If desired, read the COSC<2:0> bits (OSCCON<14:12>) to determine the current oscillator source.
2. Perform the unlock sequence to allow a write to the OSCCON register high byte.
3. Write the appropriate value to the NOSC<2:0> bits (OSCCON<10:8>) for the new oscillator source.
4. Perform the unlock sequence to allow a write to the OSCCON register low byte.
5. Set the OSWEN bit to initiate the oscillator switch.

Once the basic sequence is completed, the system clock hardware responds automatically as follows:

1. The clock switching hardware compares the COSC<2:0> bits with the new value of the NOSC<2:0> bits. If they are the same, then the clock switch is a redundant operation. In this case, the OSWEN bit is cleared automatically and the clock switch is aborted.
2. If a valid clock switch has been initiated, the LOCK (OSCCON<5>) and CF (OSCCON<3>) bits are cleared.
3. The new oscillator is turned on by the hardware if it is not currently running. If a crystal oscillator must be turned on, the hardware will wait until the OST expires. If the new source is using the PLL, then the hardware waits until a PLL lock is detected (LOCK = 1).
4. The hardware waits for 10 clock cycles from the new clock source and then performs the clock switch.
5. The hardware clears the OSWEN bit to indicate a successful clock transition. In addition, the NOSC<2:0> bits values are transferred to the COSC<2:0> bits.
6. The old clock source is turned off at this time, with the exception of LPRC (if WDT or FSCM is enabled) or SOSC (if SOSCEN remains set).

Note 1:	The processor will continue to execute code throughout the clock switching sequence. Timing-sensitive code should not be executed during this time.
2:	Direct clock switches between any Primary Oscillator mode with PLL and FRCPLL mode are not permitted. This applies to clock switches in either direction. In these instances, the application must switch to FRC mode as a transitional clock source between the two PLL modes.

A recommended code sequence for a clock switch includes the following:

1. Disable interrupts during the OSCCON register unlock and write sequence.
2. Execute the unlock sequence for the OSCCON high byte by writing 78h and 9Ah to OSCCON<15:8> in two back-to-back instructions.
3. Write the new oscillator source to the NOSCx bits in the instruction immediately following the unlock sequence.
4. Execute the unlock sequence for the OSCCON low byte by writing 46h and 57h to OSCCON<7:0> in two back-to-back instructions.
5. Set the OSWEN bit in the instruction immediately following the unlock sequence.
6. Continue to execute code that is not clock-sensitive (optional).
7. Invoke an appropriate amount of software delay (cycle counting) to allow the selected oscillator and/or PLL to start and stabilize.
8. Check to see if OSWEN is '0'. If it is, the switch was successful. If OSWEN is still set, then check the LOCK bit to determine the cause of the failure.

The core sequence for unlocking the OSCCON register and initiating a clock switch is shown in Example 9-1.

EXAMPLE 9-1: BASIC CODE SEQUENCE FOR CLOCK SWITCHING

```
;Place the new oscillator selection in W0
;OSCCONH (high byte) Unlock Sequence
MOV        #OSCCONH, w1
MOV        #0x78, w2
MOV        #0x9A, w3
MOV.b      w2, [w1]
MOV.b      w3, [w1]
;Set new oscillator selection
MOV.b      WREG, OSCCONH
;OSCCONL (low byte) unlock sequence
MOV        #OSCCONL, w1
MOV        #0x46, w2
MOV        #0x57, w3
MOV.b      w2, [w1]
MOV.b      w3, [w1]
;Start oscillator switch operation
BSET       OSCCON, #0
```

9.5 FRC Active Clock Tuning

PIC24FJ1024GA610/GB610 family devices include an automatic mechanism to calibrate the FRC during run time. This system uses active clock tuning from a source of known accuracy to maintain the FRC within a very narrow margin of its nominal 8 MHz frequency. This allows for a frequency accuracy that is well within the requirements of the *"USB 2.0 Specification"* regarding full-speed USB devices.

Note: The self-tune feature maintains sufficient accuracy for operation in USB Device mode. For applications that function as a USB host, a high-accuracy clock source (±0.05%) is still required.

The self-tune system is controlled by the bits in the upper half of the OSCTUN register. Setting the STEN bit (OSCTUN<15>) enables the self-tuning feature, allowing the hardware to calibrate to a source selected by the STSRC bit (OSCTUN<12>). When STSRC = `1`, the system uses the Start-of-Frame (SOF) packets from an external USB host for its source. When STSRC = `0`, the system uses the crystal-controlled SOSC for its calibration source. Regardless of the source, the system uses the TUN<5:0> bits (OSCTUN<5:0>) to change the FRC Oscillator's frequency. Frequency monitoring and adjustment is dynamic, occurring continuously during run time. While the system is active, the TUNx bits cannot be written to by software.

Note: To use the USB as a reference clock tuning source (STSRC = `1`), the microcontroller must be configured for USB device operation and connected to a non-suspended USB host or hub port.

If the SOSC is to be used as the reference clock tuning source (STSRC = `0`), the SOSC must also be enabled for clock tuning to occur.

The self-tune system can generate a hardware interrupt, FSTIF. The interrupt can result from a drift of the FRC from the reference, by greater than 0.2% in either direction, or whenever the frequency deviation is beyond the ability of the TUN<5:0> bits to correct (i.e., greater than 1.5%). The STLOCK and STOR status bits (OSCTUN<11,9>) are used to indicate these conditions.

The STLPOL and STORPOL bits (OSCTUN<10,8>) configure the FSTIF interrupt to occur in the presence or the absence of the conditions. It is the user's responsibility to monitor both the STLOCK and STOR bits to determine the exact cause of the interrupt.

Note: The STLPOL and STORPOL bits should be ignored when the self-tune system is disabled (STEN = `0`).

11.3 Interrupt-on-Change (IOC)

The Interrupt-on-Change function of the I/O ports allows the PIC24FJ1024GA610/GB610 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.

Interrupt-on-Change functionality is enabled on a pin by setting the IOCPx and/or IOCNx register bit for that pin. For example, PORTC has register names, IOCPC and IOCNC, for these functions. Setting a value of '1' in the IOCPx register enables interrupts for low-to-high transitions, while setting a value of '1' in the IOCNx register enables interrupts for high-to-low transitions. Setting a value of '1' in both register bits will enable interrupts for either case (e.g., a pulse on the pin will generate two interrupts). In order for any IOC to be detected, the global IOC Interrupt Enable bit (IEC1<3>) must be set, the IOCON bit (PADCON<15>) set and the associated IFSx flag cleared.

When an interrupt request is generated for a pin, the corresponding status flag (IOCFx register bit) will be set, indicating that a Change-of-State occurred on that pin. The IOCFx register bit will remain set until cleared by writing a zero to it. When any IOCFx flag bit in a given port is set, the corresponding IOCPxF bit in the IOCSTAT register will be set. This flag indicates that a change was detected on one of the bits on the given port. The IOCPxF flag will be cleared when all IOCFx<15:0> bits are cleared.

Multiple individual status flags can be cleared by writing a zero to one or more bits using a Read-Modify-Write operation. If another edge is detected on a pin whose status bit is being cleared during the Read-Modify-Write sequence, the associated change flag will still be set at the end of the Read-Modify-Write sequence.

The user should use the instruction sequence (or equivalent) shown in Example 11-1 to clear the Interrupt-on-Change Status registers.

At the end of this sequence, the W0 register will contain a zero for each bit for which the port pin had a change detected. In this way, any indication of a pin changing will not be lost.

Due to the asynchronous and real-time nature of the Interrupt-on-Change, the value read on the port pins may not indicate the state of the port when the change was detected, as a second change can occur during the interval between clearing the flag and reading the port. It is up to the user code to handle this case if it is a possibility in their application. To keep this interval to a minimum, it is recommended that any code modifying the IOCFx registers be run either in the interrupt handler or with interrupts disabled.

Each Interrupt-on-Change (IOC) pin has both a weak pull-up and a weak pull-down connected to it. The pull-ups act as a current source connected to the pin, while the pull-downs act as a current sink connected to the pin. These eliminate the need for external resistors when push button or keypad devices are connected.

The pull-ups and pull-downs are separately enabled using the IOCPUx registers (for pull-ups) and the IOCPDx registers (for pull-downs). Each IOC pin has individual control bits for its pull-up and pull-down. Setting a control bit enables the weak pull-up or pull-down for the corresponding pin.

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

EXAMPLE 11-1: IOC STATUS READ/CLEAR IN ASSEMBLY

```
MOV   0xFFFF, W0   ; Initial mask value 0xFFFF -> W0
XOR   IOCFx, W0    ; W0 has '1' for each bit set in IOCFx
AND   IOCFx        ; IOCFx & W0 ->IOCFx
```

EXAMPLE 11-2: PORT READ/WRITE IN ASSEMBLY

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

EXAMPLE 11-3: PORT READ/WRITE IN 'C'

```
TRISB = 0xFF00;              // Configure PORTB<15:8> as inputs and PORTB<7:0> as outputs
Nop();                       // Delay 1 cycle
If (PORTBbits.RB13){ };      // Test if RB13 is a '1'
```

REGISTER 11-18: RPINR6: PERIPHERAL PIN SELECT INPUT REGISTER 6

U-0	U-0	r-1	r-1	r-1	r-1	r-1	r-1
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	r-1	r-1	r-1	r-1	r-1	r-1
—	—	—	—	—	—	—	—
bit 7							bit 0

Legend:	r = Reserved bit		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-14	**Unimplemented:** Read as ‘0’
bit 13-8	**Reserved**: Maintain as ‘1’
bit 7-6	**Unimplemented:** Read as ‘0’
bit 5-0	**Reserved**: Maintain as ‘1’

REGISTER 11-19: RPINR7: PERIPHERAL PIN SELECT INPUT REGISTER 7

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	IC2R5	IC2R4	IC2R3	IC2R2	IC2R1	IC2R0
bit 15							bit 8

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	IC1R5	IC1R4	IC1R3	IC1R2	IC1R1	IC1R0
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	**IC2R<5:0>:** Assign Input Capture 2 (IC2) to Corresponding RPn or RPIn Pin bits
bit 7-6	**Unimplemented:** Read as ‘0’
bit 5-0	**IC1R<5:0>:** Assign Input Capture 1 (IC1) to Corresponding RPn or RPIn Pin bits

REGISTER 11-32: RPINR25: PERIPHERAL PIN SELECT INPUT REGISTER 25

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	CLCINBR5	CLCINBR4	CLCINBR3	CLCINBR2	CLCINBR1	CLCINBR0
bit 15							bit 8

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	CLCINAR5	CLCINAR4	CLCINAR3	CLCINAR2	CLCINAR1	CLCINAR0
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	**CLCINBR<5:0>:** Assign CLC Input B to Corresponding RPn or RPIn Pin bits
bit 7-6	**Unimplemented:** Read as ‘0’
bit 5-0	**CLCINAR<5:0>:** Assign CLC Input A to Corresponding RPn or RPIn Pin bits

REGISTER 11-33: RPINR27: PERIPHERAL PIN SELECT INPUT REGISTER 27

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	U4CTSR5	U4CTSR4	U4CTSR3	U4CTSR2	U4CTSR1	U4CTSR0
bit 15							bit 8

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	U4RXR5	U4RXR4	U4RXR3	U4RXR2	U4RXR1	U4RXR0
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	**U4CTSR<5:0>:** Assign UART4 Clear-to-Send Input ($\overline{\text{U4CTS}}$) to Corresponding RPn or RPIn Pin bits
bit 7-6	**Unimplemented:** Read as ‘0’
bit 5-0	**U4RXR<5:0>:** Assign UART4 Receive Input (U4RX) to Corresponding RPn or RPIn Pin bits

REGISTER 16-7: CCPxSTATL: CCPx STATUS REGISTER LOW

U-0	U-0	U-0	U-0	U-0	W-0	U-0	U-0
—	—	—	—	—	ICGARM	—	—
bit 15							bit 8

R-0	W1-0	W1-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
CCPTRIG	TRSET	TRCLR	ASEVT	SCEVT	ICDIS	ICOV	ICBNE
bit 7							bit 0

Legend:	C = Clearable bit	W = Writable bit	
R = Readable bit	W1 = Write ‘1’ Only bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-11	**Unimplemented:** Read as ‘0’
bit 10	**ICGARM:** Input Capture Gate Arm bit A write of ‘1’ to this location will arm the Input Capture x module for a one-shot gating event when ICGSM<1:0> = 01 or 10; read as ‘0’.
bit 9-8	**Unimplemented:** Read as ‘0’
bit 7	**CCPTRIG:** CCPx Trigger Status bit 1 = Timer has been triggered and is running 0 = Timer has not been triggered and is held in Reset
bit 6	**TRSET:** CCPx Trigger Set Request bit Write ‘1’ to this location to trigger the timer when TRIGEN = 1 (location always reads as ‘0’).
bit 5	**TRCLR:** CCPx Trigger Clear Request bit Write ‘1’ to this location to cancel the timer Trigger when TRIGEN = 1 (location always reads as ‘0’).
bit 4	**ASEVT:** CCPx Auto-Shutdown Event Status/Control bit 1 = A shutdown event is in progress; CCPx outputs are in the shutdown state 0 = CCPx outputs operate normally
bit 3	**SCEVT:** Single Edge Compare Event Status bit 1 = A single edge compare event has occurred 0 = A single edge compare event has not occurred
bit 2	**ICDIS:** Input Capture x Disable bit 1 = Event on Input Capture x pin (ICMx) does not generate a capture event 0 = Event on Input Capture x pin will generate a capture event
bit 1	**ICOV:** Input Capture x Buffer Overflow Status bit 1 = The Input Capture x FIFO buffer has overflowed 0 = The Input Capture x FIFO buffer has not overflowed
bit 0	**ICBNE:** Input Capture x Buffer Status bit 1 = Input Capture x buffer has data available 0 = Input Capture x buffer is empty

REGISTER 20-4: U1OTGCON: USB ON-THE-GO CONTROL REGISTER

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

R/W-0	R/W-0	R/W-0	R/W-0	r-0	R/W-0	r-0	R/W-0
DPPULUP	DMPULUP	DPPULDWN(1)	DMPULDWN(1)	—	OTGEN(1)	—	VBUSDIS(1)
bit 7							bit 0

Legend:	r = Reserved bit		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-8 **Unimplemented:** Read as ‘0’

bit 7 **DPPULUP:** D+ Pull-up Enable bit
1 = D+ data line pull-up resistor is enabled
0 = D+ data line pull-up resistor is disabled

bit 6 **DMPULUP:** D- Pull-up Enable bit
1 = D- data line pull-up resistor is enabled
0 = D- data line pull-up resistor is disabled

bit 5 **DPPULDWN:** D+ Pull-Down Enable bit(1)
1 = D+ data line pull-down resistor is enabled
0 = D+ data line pull-down resistor is disabled

bit 4 **DMPULDWN:** D- Pull-Down Enable bit(1)
1 = D- data line pull-down resistor is enabled
0 = D- data line pull-down resistor is disabled

bit 3 **Reserved:** Maintain as ‘0’

bit 2 **OTGEN:** OTG Features Enable bit(1)
1 = USB OTG is enabled; all D+/D- pull-up and pull-down bits are enabled
0 = USB OTG is disabled; D+/D- pull-up and pull-down bits are controlled in hardware by the settings of the HOSTEN and USBEN (U1CON<3,0>) bits

bit 1 **Reserved:** Maintain as ‘0’

bit 0 **VBUSDIS:** VBUS Discharge Enable bit(1)
1 = VBUS line is discharged through a resistor
0 = VBUS line is not discharged

Note 1: These bits are only used in Host mode; do not use in Device mode.

22.0 REAL-TIME CLOCK AND CALENDAR WITH TIMESTAMP

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 Real-Time Clock and Calendar, refer to the *"dsPIC33/PIC24 Family Reference Manual"*, **"RTCC with Timestamp"** (DS70005193), which is available from the Microchip web site (www.microchip.com). The information in this data sheet supersedes the information in the FRM.

The RTCC provides the user with a Real-Time Clock and Calendar (RTCC) function that can be calibrated.

Key features of the RTCC module are:

- Selectable Clock Source
- Provides Hours, Minutes and Seconds Using 24-Hour Format
- Visibility of One Half Second Period
- Provides Calendar – Weekday, Date, Month and Year
- Alarm-Configurable for Half a Second, 1 Second, 10 Seconds, 1 Minute, 10 Minutes, 1 Hour, 1 Day, 1 Week, 1 Month or 1 Year
- Alarm Repeat with Decrementing Counter
- Alarm with Indefinite Repeat Chime
- Year 2000 to 2099 Leap Year Correction
- BCD Format for Smaller Software Overhead
- Optimized for Long-Term Battery Operation
- User Calibration of the 32.768 kHz Clock Crystal/ 32K INTRC Frequency with Periodic Auto-Adjust
- Fractional Second Synchronization
- Calibration to within ±2.64 Seconds Error per Month
- Calibrates up to 260 ppm of Crystal Error
- Ability to Periodically Wake-up External Devices without CPU Intervention (external power control)
- Power Control Output for External Circuit Control
- Calibration takes Effect Every 15 Seconds
- Timestamp Capture Register for Time and Date
- Programmable Prescaler and Clock Divider Circuit Allows Operation with Any Clock Source up to 32 MHz, Including 32.768 kHz Crystal, 50/60 Hz Powerline Clock, External Real-Time Clock (RTC) or 31.25 kHz LPRC Clock

22.1 RTCC Source Clock

The RTCC clock divider block converts the incoming oscillator source into accurate 1/2 and 1 second clocks for the RTCC. The clock divider is optimized to work with three different oscillator sources:

- 32.768 kHz Crystal Oscillator
- 31 kHz Low-Power RC Oscillator (LPRC)
- External 50 Hz or 60 Hz Powerline Frequency

An asynchronous prescaler, PS<1:0> (RTCCON2L<5:4>), is provided that allows the RTCC to work with higher speed clock sources, such as the system clock. Divide ratios of 1:16, 1:64 or 1:256 may be selected, allowing sources up to 32 MHz to clock the RTCC.

22.1.1 COARSE FREQUENCY DIVISION

The clock divider block has a 16-bit counter used to divide the input clock frequency. The divide ratio is set by the DIV<15:0> register bits (RTCCON2H<15:0>). The DIV<15:0> bits should be programmed with a value to produce a nominal 1/2 second clock divider count period.

22.1.2 FINE FREQUENCY DIVISION

The fine frequency division is set using the FDIV<4:0> (RTCCON2L<15:11>) bits. Increasing the FDIVx value will lengthen the overall clock divider period.

If FDIV<4:0> = `00000`, the fine frequency division circuit is effectively disabled. Otherwise, it will optionally remove a clock pulse from the input of the clock divider every 1/2 second. This functionality will allow the user to remove up to 31 pulses over a fixed period of 16 seconds, depending on the value of FDIVx.

The value for DIV<15:0> is calculated as shown in Equation 22-1. The fractional remainder of the DIV<15:0> calculation result can be used to calculate the value for FDIV<4:0>.

EQUATION 22-1: RTCC CLOCK DIVIDER OUTPUT FREQUENCY

$$F_{OUT} = \frac{F_{IN}}{2 \bullet (PS\text{<1:0>}\ Prescaler) \bullet (DIV\text{<15:0>} + 1) + \left(\frac{FDIV\text{<4:0>}}{32}\right)}$$

The DIV<15:0> value is the integer part of this calculation:

$$DIV\text{<15:0>} = \frac{F_{IN}}{2 \bullet (PS\text{<1:0>}\ Prescaler)} - 1$$

The FDIV<4:0> value is the fractional part of the DIV<15:0> calculation multiplied by 32.

REGISTER 30-10: FICD CONFIGURATION REGISTER

U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1
—	—	—	—	—	—	—	—
bit 23							bit 16

R/PO-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1
BTSWP	—	—	—	—	—	—	—
bit 15							bit 8

r-1	U-1	R/PO-1	U-1	U-1	U-1	R/PO-1	R/PO-1
—	—	JTAGEN	—	—	—	ICS1	ICS0
bit 7							bit 0

Legend:	PO = Program Once bit	r = Reserved bit	
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘1’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 23-16	**Unimplemented:** Read as ‘1’
bit 15	**BTSWP:** BOOTSWP Instruction Enable bit 1 = BOOTSWP instruction is disabled 0 = BOOTSWP instruction is enabled
bit 14-8	**Unimplemented:** Read as ‘1’
bit 7	**Reserved:** Maintain as ‘1’
bit 6	**Unimplemented:** Read as ‘1’
bit 5	**JTAGEN:** JTAG Port Enable bit 1 = JTAG port is enabled 0 = JTAG port is disabled
bit 4-2	**Unimplemented:** Read as ‘1’
bit 1-0	**ICS<1:0>:** ICD Communication Channel Select bits 11 = Communicates on PGEC1/PGED1 10 = Communicates on PGEC2/PGED2 01 = Communicates on PGEC3/PGED3 00 = Reserved; do not use

TABLE 30-2: DEVICE ID REGISTERS

Address	Name	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FF0000h	DEVID	FAMID<7:0>								DEV<7:0>							
FF0002h	DEVREV	—												REV<3:0>			

TABLE 30-3: DEVICE ID BIT FIELD DESCRIPTIONS

Bit Field	Register	Description
FAMID<7:0>	DEVID	Encodes the family ID of the device.
DEV<7:0>	DEVID	Encodes the individual ID of the device.
REV<3:0>	DEVREV	Encodes the sequential (numerical) revision identifier of the device.

TABLE 30-4: PIC24FJ1024GA610/GB610 FAMILY DEVICE IDs

Device	DEVID
PIC24FJ128GA606	6000h
PIC24FJ256GA606	6008h
PIC24FJ512GA606	6010h
PIC24FJ1024GA606	6018h
PIC24FJ128GA610	6001h
PIC24FJ256GA610	6009h
PIC24FJ512GA610	6011h
PIC24FJ1024GA610	6019h
PIC24FJ128GB606	6004h
PIC24FJ256GB606	600Ch
PIC24FJ512GB606	6014h
PIC24FJ1024GB606	601Ch
PIC24FJ128GB610	6005h
PIC24FJ256GB610	600Dh
PIC24FJ512GB610	6015h
PIC24FJ1024GB610	601Dh

30.2 Unique Device Identifier (UDID)

All PIC24FJ1024GA610/GB610 family devices are individually encoded during final manufacturing with a Unique Device Identifier, or UDID. The UDID cannot be erased by a bulk erase command or any other user-accessible means. This feature allows for manufacturing traceability of Microchip Technology devices in applications where this is a requirement. It may also be used by the application manufacturer for any number of things that may require unique identification, such as:

- Tracking the device
- Unique serial number
- Unique security key

The UDID comprises five 24-bit program words. When taken together, these fields form a unique 120-bit identifier.

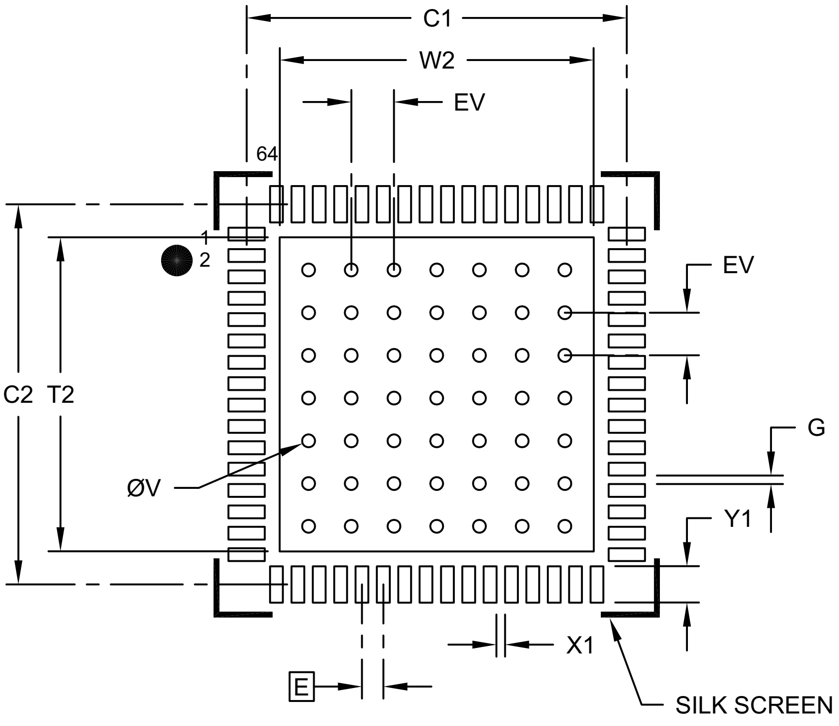
The UDID is stored in five read-only locations, located between 801600h and 801608h in the device configuration space. Table 30-5 lists the addresses of the identifier words.

TABLE 30-5: UDID ADDRESSES

UDID	Address	Description
UDID1	801600	UDID Word 1
UDID2	801602	UDID Word 2
UDID3	801604	UDID Word 3
UDID4	801606	UDID Word 4
UDID5	801608	UDID Word 5

64-Lead Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [QFN] With 0.40 mm Contact Length and 7.70x7.70mm Exposed Pad

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	0.50 BSC		
Optional Center Pad Width	W2			7.50
Optional Center Pad Length	T2			7.50
Contact Pad Spacing	C1		8.90	
Contact Pad Spacing	C2		8.90	
Contact Pad Width (X20)	X1			0.30
Contact Pad Length (X20)	Y1			0.90
Contact Pad to Center Pad (X20)	G	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing No. C04-2213B

NOTES: