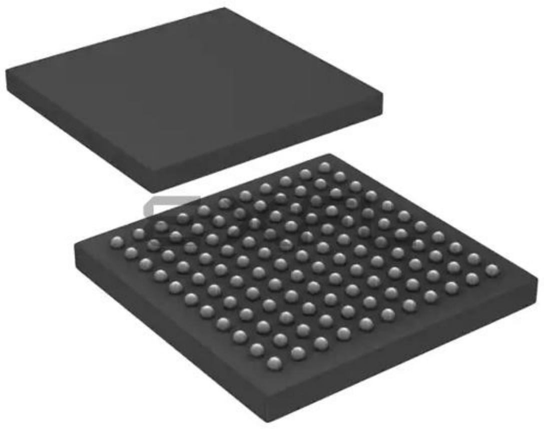
E·XFL

Microchip Technology - PIC24FJ64GA412T-I/BG Datasheet



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	16-Bit
Speed	32MHz
Connectivity	I²C, IrDA, LINbus, PMP/PSP, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, LCD, LVD, POR, PWM, WDT
Number of I/O	102
Program Memory Size	64KB (22K x 24)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 24x10/12b; D/A 1x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	121-TFBGA
Supplier Device Package	121-TFBGA (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic24fj64ga412t-i-bg

Email: info@E-XFL.COM Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

TABLE 4-5: SFR BLOCK 000h

Register	Address	All Resets	Register	Address	All Resets	Register	Address	All Resets
Core			**Interrupt Controller**			IPC7	0B6	0100010001000100
WREG0	000	0000000000000000	INTCON1	080	0000000000000000	IPC8	0B8	0100010001000100
WREG1	002	0000000000000000	INTCON2	082	1000000000000000	IPC9	0BA	0100010001000100
WREG2	004	0000000000000000	INTCON4	086	0000000000000000	IPC10	0BC	0100010001000100
WREG3	006	0000000000000000	IFS0	088	0000000000000000	IPC11	0BE	0100010001000100
WREG4	008	0000000000000000	IFS1	08A	0000000000000000	IPC12	0C0	0100010001000100
WREG5	00A	0000000000000000	IFS2	08C	0000000000000000	IPC13	0C2	0100010001000000
WREG6	00C	0000000000000000	IFS3	08E	0000000000000000	IPC14	0C4	0100010001000100
WREG7	00E	0000000000000000	IFS4	090	0000000000000000	IPC15	0C6	0100010001000100
WREG8	010	0000000000000000	IFS5	092	0000000000000000	IPC16	0C8	0100010001000100
WREG9	012	0000000000000000	IFS6	094	0000000000000000	IPC17	0CA	0100010000000000
WREG10	014	0000000000000000	IFS7	096	0000000000000000	IPC18	0CC	0000000001000100
WREG11	016	0000000000000000	IEC0	098	0000000000000000	IPC19	0CE	0000010001000000
WREG12	018	0000000000000000	IEC1	09A	0000000000000000	IPC20	0D0	0100010001000000
WREG13	01A	0000000000000000	IEC2	09C	0000000000000000	IPC21	0D2	0100010001000100
WREG14	01C	0000000000000000	IEC3	09E	0000000000000000	IPC22	0D4	0100010001000100
WREG15	01E	0000000000000000	IEC4	0A0	0000000000000000	IPC23	0D6	0100010001000100
SPLIM	020	xxxxxxxxxxxxxxx0	IEC5	0A2	0000000000000000	IPC24	0D8	0100010001000100
PCL	02E	0000000000000000	IEC6	0A4	0000000000000000	IPC25	0DA	0000010001000100
PCH	030	0000000000000000	IEC7	0A6	0000000000000000	IPC26	0DC	0000010000000000
DSRPAG	032	0000000000000000	IPC0	0A8	0100010001000100	IPC27	0DE	0100010001000000
DSWPAG	034	0000000000000000	IPC1	0AA	0100010001000100	IPC28	0E0	0100010001000100
RCOUNT	036	xxxxxxxxxxxxxxxx	IPC2	0AC	0100010001000100	IPC29	0E2	0000000001000100
SR	042	0000000000000000	IPC3	0AE	0100010001000100	INTTREG	0E4	0000000000000000
CORCON	044	0000000000000100	IPC4	0B0	0100010001000100			
DISICNT	052	00xxxxxxxxxxxxxx	IPC5	0B2	0100010000000100			
TBLPAG	054	0000000000000000	IPC6	0B4	0100010001000100			

Legend: x = unknown or indeterminate value. Reset and address values are in hexadecimal.

6.2 RTSP Operation

The PIC24F Flash program memory array is organized into rows of 64 instructions or 192 bytes. RTSP allows the user to erase blocks of eight rows (512 instructions) at a time and to program one row at a time. It is also possible to program two words.

The 8-row erase blocks and single row write blocks are edge-aligned, from the beginning of program memory on boundaries of 1536 bytes and 192 bytes, respectively.

When data is written to program memory using `TBLWT` instructions, the data is not written directly to memory. Instead, data written using Table Writes is stored in holding latches until the programming sequence is executed.

Any number of `TBLWT` instructions can be executed and a write will be successfully performed. However, 64 `TBLWT` instructions are required to write the full row of memory.

To ensure that no data is corrupted during a write, any unused address should be programmed with FFFFFFh. This is because the holding latches reset to an unknown state, so if the addresses are left in the Reset state, they may overwrite the locations on rows which were not rewritten.

The basic sequence for RTSP programming is:

- Set up a Table Pointer to point to the programming latches
- Perform a series of `TBLWT` instructions to load the buffers
- Set the NVM Address registers to point to the destination

Programming is performed by setting the control bits in the NVMCON register.

Data can be loaded in any order and the holding registers can be written to multiple times before performing a write operation. Subsequent writes, however, will wipe out any previous writes.

Note: Writing to a location multiple times without erasing is *not* recommended.

All of the Table Write operations are single-word writes (2 instruction cycles), because only the buffers are written. A programming cycle is required for programming each row.

6.3 JTAG Operation

The PIC24F family supports JTAG boundary scan. Boundary scan can improve the manufacturing process by verifying pin to PCB connectivity.

6.4 Enhanced In-Circuit Serial Programming

Enhanced In-Circuit Serial Programming uses an on-board bootloader, known as the Program Executive (PE), to manage the programming process. Using an SPI data frame format, the Program Executive can erase, program and verify program memory. For more information on Enhanced ICSP, see the device programming specification.

6.5 Programming Operations

A complete programming sequence is necessary for programming or erasing the internal Flash in RTSP mode. During a programming or erase operation, the processor stalls (waits) until the operation is finished. Setting the WR bit (NVMCON<15>) starts the operation and the WR bit is automatically cleared when the operation is finished. In Dual Partition modes, programming or erasing the Inactive Partition does not stall the processor; the code in the Active Partition continues to execute during the programming operation.

For more information on programming the device, please refer to the *“dsPIC33/PIC24 Family Reference Manual”*, **“Dual Partition Flash Program Memory”** (DS70005156).

6.6 Control Registers

There are four SFRs used to read and write the program Flash memory:

- NVMCON
- NVMKEY
- NVMADRL
- NVMADRH

The NVMCON register (Register 6-1) controls which blocks are to be erased, which memory type is to be programmed and when the programming cycle starts.

NVMKEY is a write-only register that is used for write protection. To start a programming or erase sequence, the user must consecutively write 55h and AAh to the NVMKEY register. For more information, refer to **Section 6.5 “Programming Operations”**.

The NVMADRL and NVMADRH registers contain the lower word and upper byte of the destination address of the NVM write or erase operation. Some operations (e.g., chip erase, Inactive Partition erase) operate on fixed locations and do not require an address value.

© 2015-2016 Microchip Technology Inc.

REGISTER 8-7: IFS1: INTERRUPT FLAG STATUS REGISTER 1 (CONTINUED)

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 8-10: IFS4: INTERRUPT FLAG STATUS REGISTER 4

U-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
—	DAC1IF	CTMUIF	—	—	—	CCP7IF	HLVDIF
bit 15							bit 8

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
MI2C3IF	SI2C3IF	—	—	CRCIF	U2ERIF	U1ERIF	CCP2IF
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	**DAC1IF:** DAC Converter Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 13	**CTMUIF:** CTMU Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 12-10	**Unimplemented:** Read as ‘0’
bit 9	**CCP7IF:** SCCP7 Capture/Compare Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 8	**HLVDIF:** High/Low-Voltage Detect Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 7	**MI2C3IF:** Master I2C3 Event Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 6	**SI2C3IF:** Slave I2C3 Event Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 5-4	**Unimplemented:** Read as ‘0’
bit 3	**CRCIF:** CRC Generator Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 2	**U2ERIF:** UART2 Error Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 1	**U1ERIF:** UART1 Error Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 0	**CCP2IF:** SCCP2 Capture/Compare Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred

© 2015-2016 Microchip Technology Inc.

REGISTER 8-23: 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	R/W-1	R/W-0	R/W-0
—	IC2IP2	IC2IP1	IC2IP0	—	DMA0IP2	DMA0IP1	DMA0IP0
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 **Unimplemented:** Read as ‘0’

bit 2-0 **DMA0IP<2:0>:** DMA Channel 0 Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

REGISTER 8-45: IPC23: INTERRUPT PRIORITY CONTROL REGISTER 23

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	CCP4IP2	CCP4IP1	CCP4IP0	—	CCP3IP2	CCP3IP1	CCP3IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	SPI4TXIP2	SPI4TXIP1	SPI4TXIP0	—	SPI4IP2	SPI4IP1	SPI4IP0
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 **CCP4IP<2:0>:** SCCP4 Capture/Compare 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 **CCP3IP<2:0>:** SCCP3 Capture/Compare 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 **SPI4TXIP<2:0>:** SPI4 Transmit Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **SPI4IP<2:0>:** SPI4 General Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

11.5.5 CONSIDERATIONS FOR PERIPHERAL PIN SELECTION

The ability to control Peripheral Pin Selection 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 V_{SS}, and all Peripheral Pin Select outputs are disconnected.

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/RPIn 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 a 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 a digital I/O when used with a Peripheral Pin Select.

Example 11-4 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 11-4: CONFIGURING UART1 INPUT AND OUTPUT FUNCTIONS

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

// Configure Input Functions (Table 11-11)
    // Assign U1RX To Pin RP0
    RPINR18bits.U1RXR = 0;

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

// Configure Output Functions (Table 11-12)
    // Assign U1TX To Pin RP2
    RPOR1bits.RP2R = 3;

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

// Lock Registers
asm volatile   ("MOV   #OSCCON, w1   \n"
               "MOV   #0x46, w2     \n"
               "MOV   #0x57, w3     \n"
               "MOV.b  w2, [w1]     \n"
               "MOV.b  w3, [w1]     \n"
               "BSET   OSCCON, #6");

// or use the XC16 built-in macro:
// __builtin_write_OSCCONL(OSCCON | 0x40);
```

REGISTER 11-5: RPINR2: PERIPHERAL PIN SELECT INPUT REGISTER 2

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	OCTRIG2R5	OCTRIG2R4	OCTRIG2R3	OCTRIG2R2	OCTRIG2R1	OCTRIG2R0
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
—	—	INT4R5	INT4R4	INT4R3	INT4R2	INT4R1	INT4R0
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	**OCTRIG2R<5:0>:** Assign Output Compare Trigger 2 (OCTRIG2) to Corresponding RPn or RPIn Pin bits
bit 7-6	**Unimplemented:** Read as ‘0’
bit 5-0	**INT4R<5:0>:** Assign External Interrupt 4 (INT4) to Corresponding RPn or RPIn Pin bits

REGISTER 11-6: RPINR3: PERIPHERAL PIN SELECT INPUT REGISTER 3

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	T3CKR5	T3CKR4	T3CKR3	T3CKR2	T3CKR1	T3CKR0
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
—	—	T2CKR5	T2CKR4	T2CKR3	T2CKR2	T2CKR1	T2CKR0
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	**T3CKR<5:0>:** Assign Timer3 Clock Input (T3CK) to Corresponding RPn or RPIn Pin bits
bit 7-6	**Unimplemented:** Read as ‘0’
bit 5-0	**T2CKR<5:0>:** Assign Timer2 Clock Input (T2CK) to Corresponding RPn or RPIn Pin bits

REGISTER 11-27: RPOR4: PERIPHERAL PIN SELECT OUTPUT REGISTER 4

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP9R5	RP9R4	RP9R3	RP9R2	RP9R1	RP9R0
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
—	—	RP8R5	RP8R4	RP8R3	RP8R2	RP8R1	RP8R0
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	**RP9R<5:0>:** RP9 Output Pin Mapping bits Peripheral Output Number n is assigned to pin, RP9 (see Table 11-12 for peripheral function numbers).
bit 7-6	**Unimplemented:** Read as ‘0’
bit 5-0	**RP8R<5:0>:** RP8 Output Pin Mapping bits Peripheral Output Number n is assigned to pin, RP8 (see Table 11-12 for peripheral function numbers).

REGISTER 11-28: RPOR5: PERIPHERAL PIN SELECT OUTPUT REGISTER 5

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP11R5	RP11R4	RP11R3	RP11R2	RP11R1	RP11R0
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
—	—	RP10R5	RP10R4	RP10R3	RP10R2	RP10R1	RP10R0
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	**RP11R<5:0>:** RP11 Output Pin Mapping bits Peripheral Output Number n is assigned to pin, RP11 (see Table 11-12 for peripheral function numbers).
bit 7-6	**Unimplemented:** Read as ‘0’
bit 5-0	**RP10R<5:0>:** RP10 Output Pin Mapping bits Peripheral Output Number n is assigned to pin, RP10 (see Table 11-12 for peripheral function numbers).

REGISTER 11-33: RPOR10: PERIPHERAL PIN SELECT OUTPUT REGISTER 10

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP21R5	RP21R4	RP21R3	RP21R2	RP21R1	RP21R0
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
—	—	RP20R5	RP20R4	RP20R3	RP20R2	RP20R1	RP20R0
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	**RP21R<5:0>:** RP21 Output Pin Mapping bits Peripheral Output Number n is assigned to pin, RP21 (see Table 11-12 for peripheral function numbers).
bit 7-6	**Unimplemented:** Read as ‘0’
bit 5-0	**RP20R<5:0>:** RP20 Output Pin Mapping bits Peripheral Output Number n is assigned to pin, RP20 (see Table 11-12 for peripheral function numbers).

REGISTER 11-34: RPOR11: PERIPHERAL PIN SELECT OUTPUT REGISTER 11

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP23R5	RP23R4	RP23R3	RP23R2	RP23R1	RP23R0
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
—	—	RP22R5	RP22R4	RP22R3	RP22R2	RP22R1	RP22R0
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	**RP23R<5:0>:** RP23 Output Pin Mapping bits Peripheral Output Number n is assigned to pin, RP23 (see Table 11-12 for peripheral function numbers).
bit 7-6	**Unimplemented:** Read as ‘0’
bit 5-0	**RP22R<5:0>:** RP22 Output Pin Mapping bits Peripheral Output Number n is assigned to pin, RP22 (see Table 11-12 for peripheral function numbers).

FIGURE 14-3: DUAL 16-BIT TIMER MODE

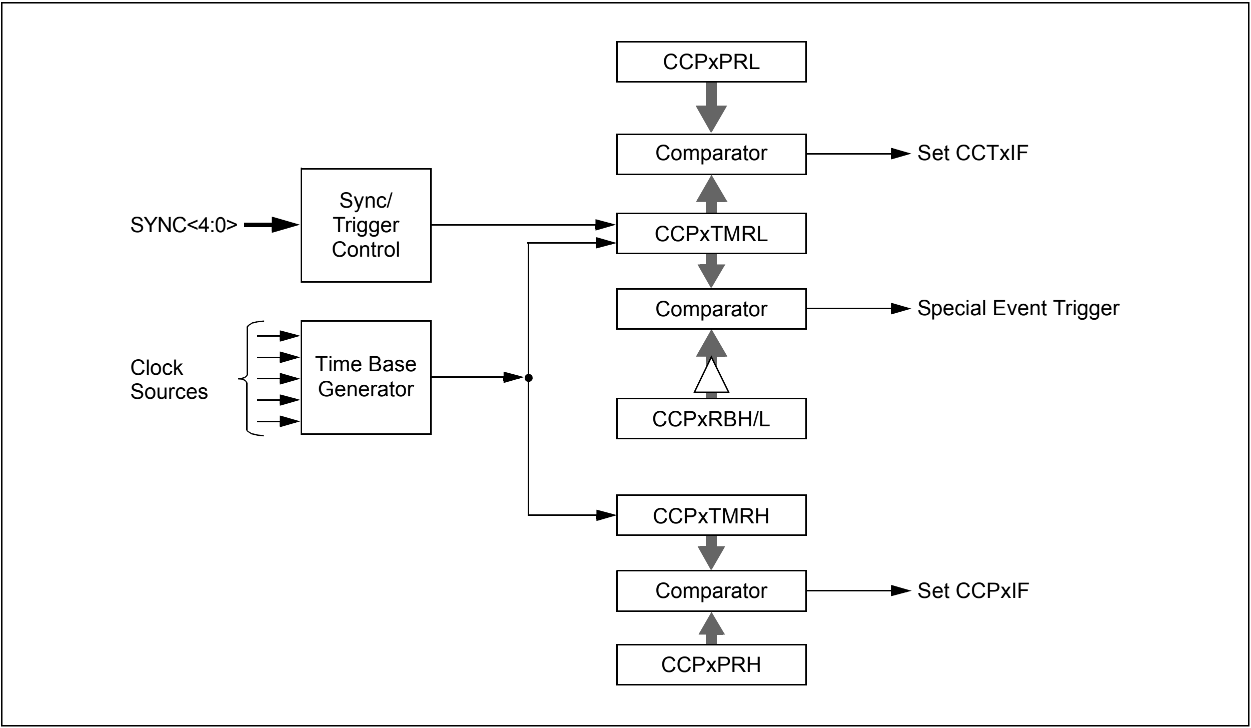
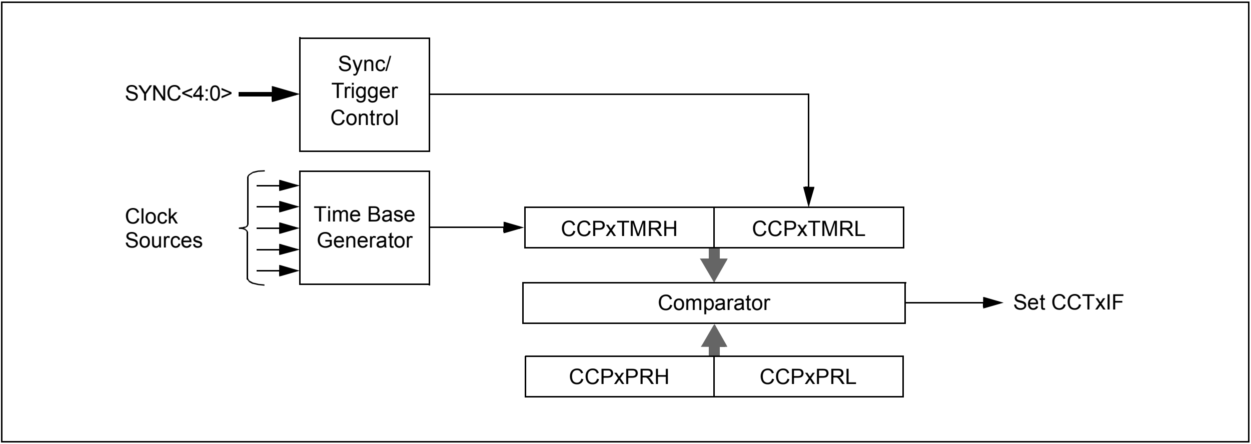


FIGURE 14-4: 32-BIT TIMER MODE



17.0 SERIAL PERIPHERAL INTERFACE (SPI)

Note:	This data sheet summarizes the features of the PIC24FJ256GA412/GB412 family of devices. It is not intended to be a comprehensive reference source. To complement the information in this data sheet, refer to the *"dsPIC33/PIC24 Family Reference Manual"*, **"Serial Peripheral Interface (SPI) with Audio Codec Support"** (DS70005136), which is available from the Microchip web site (www.microchip.com). The information in this data sheet supersedes the information in the FRM.

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 EEPROMs, shift registers, display drivers, A/D Converters, etc. The SPI module is compatible with the Motorola® SPI and SIOP interfaces. All devices in the PIC24FJ256GA412/GB412 family include three SPI modules.

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 a FIFO buffer. The FIFO level depends on the configured mode.

Variable length data can be transmitted and received, from 2 to 32-bits.

Note:	Do not perform Read-Modify-Write operations (such as bit-oriented instructions) on the SPIxBUF 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 module also supports Audio modes. Four different Audio modes are available.

- I^2S mode
- Left Justified
- Right Justified
- PCM/DSP

In each of these modes, the serial clock is free-running and audio data is always transferred.

If an audio protocol data transfer takes place between two devices, then usually one device is the master and the other is the slave. However, audio data can be transferred between two slaves. Because the audio protocols require free-running clocks, the master can be a third party controller. In either case, the master generates two free-running clocks: SCKx and LRC (Left, Right Channel Clock/$\overline{\text{SSx}}$/FSYNC).

The SPI serial interface consists of four pins:

- SDIx: Serial Data Input
- SDOx: Serial Data Output
- SCKx: Shift Clock Input or Output
- $\overline{\text{SSx}}$: 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, $\overline{\text{SSx}}$ is not used. In the 2-pin mode, both SDOx and $\overline{\text{SSx}}$ are not used.

The SPI module has the ability to generate three interrupts reflecting the events that occur during the data communication. The following types of interrupts can be generated:

1. Receive interrupts are signalled by SPIxRXIF. This event occurs when:
   - RX watermark interrupt
   - SPIROV = 1
   - SPIRBF = 1
   - SPIRBE = 1

   provided the respective mask bits are enabled in SPIxIMSKL/H.
2. Transmit interrupts are signalled by SPIxTXIF. This event occurs when:
   - TX watermark interrupt
   - SPITUR = 1
   - SPITBF = 1
   - SPITBE = 1

   provided the respective mask bits are enabled in SPIxIMSKL/H.
3. General interrupts are signalled by SPIxIF. This event occurs when
   - FRMERR = 1
   - SPIBUSY = 1
   - SRMT = 1

   provided the respective mask bits are enabled in SPIxIMSKL/H.

Block diagrams of the module in Standard and Enhanced modes are shown in Figure 17-1 and Figure 17-2.

Note:	In this section, the SPI modules are referred to together as SPIx, or separately as SPI1, SPI2 or SPI3. Special Function Registers will follow a similar notation. For example, SPIxCON1 and SPIxCON2 refer to the control registers for any of the three SPI modules.

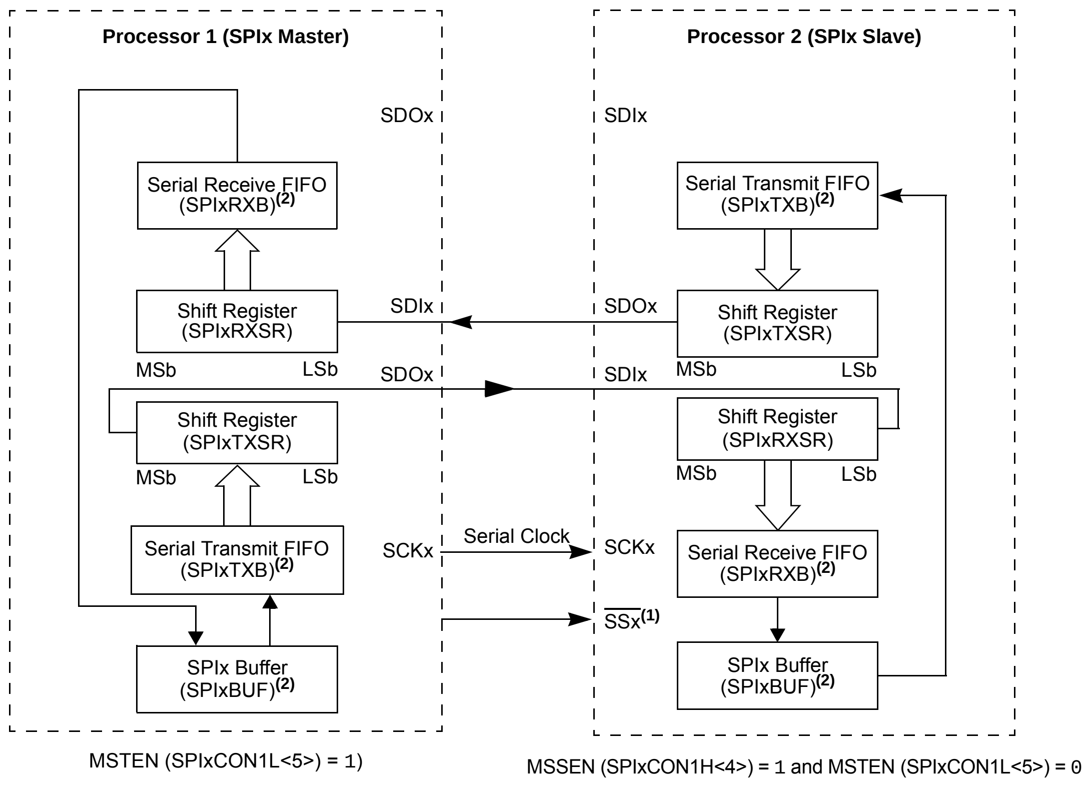
REGISTER 17-4: SPIxSTATL: SPIx STATUS REGISTER LOW (CONTINUED)

bit 3	**SPITBE:** SPIx Transmit Buffer Empty Status bit 1 = SPIxTXB is empty 0 = SPIxTXB is not empty <u>Standard Buffer Mode:</u> Automatically set in hardware when SPIx transfers data from SPIxTXB to SPIxTXSR. Automatically cleared in hardware when SPIxBUF is written, loading SPIxTXB. <u>Enhanced Buffer Mode:</u> Indicates TXELM<5:0> = 000000.
bit 2	**Unimplemented:** Read as '0'
bit 1	**SPITBF:** SPIx Transmit Buffer Full Status bit 1 = SPIxTXB is full 0 = SPIxTXB not full <u>Standard Buffer Mode:</u> Automatically set in hardware when SPIxBUF is written, loading SPIxTXB. Automatically cleared in hardware when SPIx transfers data from SPIxTXB to SPIxTXSR. <u>Enhanced Buffer Mode:</u> Indicates TXELM<5:0> = 111111.
bit 0	**SPIRBF:** SPIx Receive Buffer Full Status bit 1 = SPIxRXB is full 0 = SPIxRXB is not full <u>Standard Buffer Mode:</u> Automatically set in hardware when SPIx transfers data from SPIxRXSR to SPIxRXB. Automatically cleared in hardware when SPIxBUF is read from, reading SPIxRXB. <u>Enhanced Buffer Mode:</u> Indicates RXELM<5:0> = 111111.

Note 1: SPITUR is cleared when SPIEN = 0. When IGNTUR = 1, SPITUR provides dynamic status of the Transmit Underrun condition, but does not stop RX/TX operation and does not need to be cleared by software.

© 2015-2016 Microchip Technology Inc.

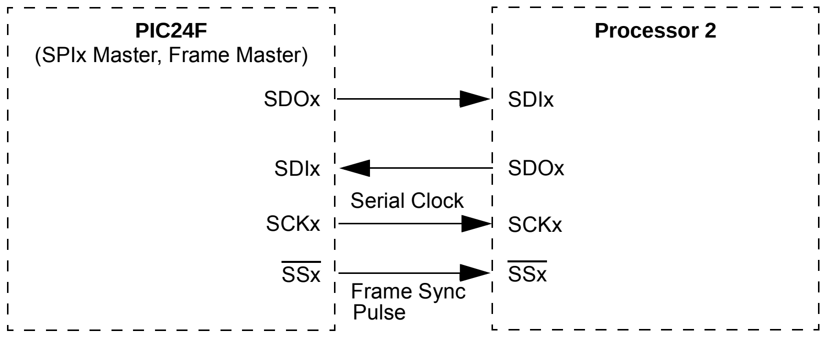
FIGURE 17-4: SPIx MASTER/SLAVE CONNECTION (ENHANCED BUFFER MODES)


Processor 1 (SPIx Master)
Processor 2 (SPIx Slave)
SDOx
SDIx
Serial Receive FIFO (SPIxRXB)(2)
Serial Transmit FIFO (SPIxTXB)(2)
Shift Register (SPIxRXSR)
Shift Register (SPIxTXSR)
SDIx
SDOx
MSb
LSb
SDOx
SDIx
Shift Register (SPIxTXSR)
Shift Register (SPIxRXSR)
MSb
LSb
Serial Transmit FIFO (SPIxTXB)(2)
Serial Clock
SCKx
SCKx
Serial Receive FIFO (SPIxRXB)(2)
SSx(1)
SPIx Buffer (SPIxBUF)(2)
SPIx Buffer (SPIxBUF)(2)
MSTEN (SPIxCON1L<5>) = 1)
MSSEN (SPIxCON1H<4>) = 1 and MSTEN (SPIxCON1L<5>) = 0

Note 1: Using the SSx pin in Slave mode of operation is optional.

2: User must write transmit data to read the received data from SPIxBUF. The SPIxTXB and SPIxRXB registers are memory-mapped to SPIxBUF.

FIGURE 17-5: SPIx MASTER, FRAME MASTER CONNECTION DIAGRAM



19.1 UARTx Baud Rate Generator (BRG)

The UARTx module includes a dedicated, 16-bit Baud Rate Generator. The UxBRG register controls the period of a free-running, 16-bit timer. Equation 19-1 shows the formula for computation of the baud rate when BRGH = 0.

EQUATION 19-1: UARTx BAUD RATE WITH BRGH = 0[1,2]

$$\text{Baud Rate} = \frac{F_{CY}}{16 \bullet (\text{UxBRG} + 1)}$$

$$\text{UxBRG} = \frac{F_{CY}}{16 \bullet \text{Baud Rate}} - 1$$

Note 1: FCY denotes the instruction cycle clock frequency (FOSC/2).

2: Based on FCY = FOSC/2; Doze mode and PLL are disabled.

Example 19-1 shows the calculation of the baud rate error for the following conditions:

- FCY = 4 MHz
- Desired Baud Rate = 9600

The maximum baud rate (BRGH = 0) possible is FCY/16 (for UxBRG = 0) and the minimum baud rate possible is FCY/(16 * 65536).

Equation 19-2 shows the formula for computation of the baud rate when BRGH = 1.

EQUATION 19-2: UARTx BAUD RATE WITH BRGH = 1[1,2]

$$\text{Baud Rate} = \frac{F_{CY}}{4 \bullet (\text{UxBRG} + 1)}$$

$$\text{UxBRG} = \frac{F_{CY}}{4 \bullet \text{Baud Rate}} - 1$$

Note 1: FCY denotes the instruction cycle clock frequency.

2: Based on FCY = FOSC/2; Doze mode and PLL are disabled.

The maximum baud rate (BRGH = 1) possible is FCY/4 (for UxBRG = 0) and the minimum baud rate possible is FCY/(4 * 65536).

Writing a new value to the UxBRG register causes the BRG timer to be reset (cleared). This ensures the BRG does not wait for a timer overflow before generating the new baud rate.

EXAMPLE 19-1: BAUD RATE ERROR CALCULATION (BRGH = 0)[1]

Desired Baud Rate = FCY/(16 (UxBRG + 1))

Solving for UxBRG Value:

UxBRG = ((FCY/Desired Baud Rate)/16) – 1
UxBRG = ((4000000/9600)/16) – 1
UxBRG = 25

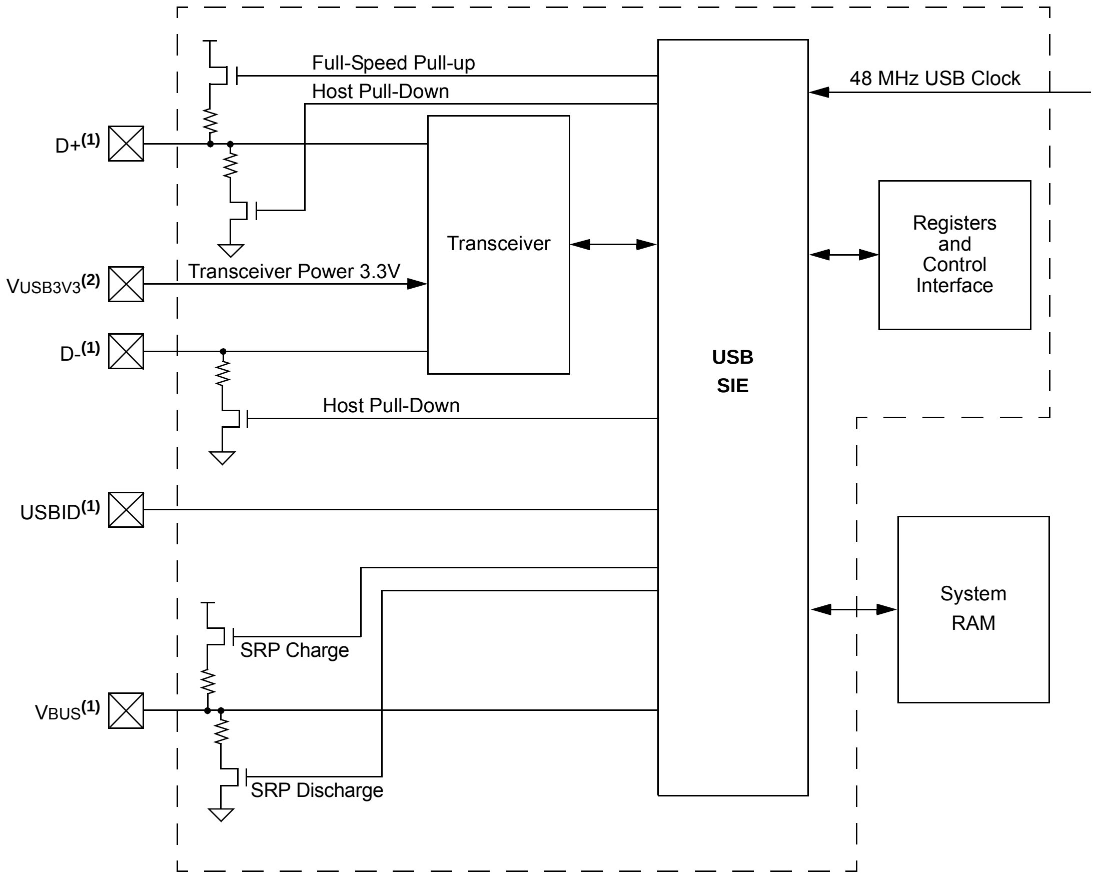
Calculated Baud Rate = 4000000/(16 (25 + 1))
= 9615

Error = (Calculated Baud Rate – Desired Baud Rate)
Desired Baud Rate
= (9615 – 9600)/9600
= 0.16%

Note 1: Based on FCY = FOSC/2; Doze mode and PLL are disabled.

© 2015-2016 Microchip Technology Inc.

FIGURE 20-1: USB OTG MODULE BLOCK DIAGRAM



Note 1: Pins are multiplexed with digital I/O and other device features.

2: Connecting VBUS3V3 to VDD is highly recommended, as floating this input can cause increased IPD currents. The pin should be tied to VDD when the USB functions are not used.

© 2015-2016 Microchip Technology Inc.

20.4.2 RECEIVING AN IN TOKEN IN DEVICE MODE

1. Attach to a USB host and enumerate as described in Chapter 9 of the *“USB 2.0 Specification”*.
2. Create a data buffer and populate it with the data to send to the host.
3. In the appropriate (even or odd) TX BD for the desired endpoint:
   a) Set up the status register (BDnSTAT) with the correct data toggle (DATA0/1) value and the byte count of the data buffer.
   b) Set up the address register (BDnADR) with the starting address of the data buffer.
   c) Set the UOWN bit of the status register to ‘1’.
4. When the USB module receives an IN token, it automatically transmits the data in the buffer. Upon completion, the module updates the status register (BDnSTAT) and sets the Token Complete Interrupt Flag, TRNIF (U1IR<3>).

20.4.3 RECEIVING AN OUT TOKEN IN DEVICE MODE

1. Attach to a USB host and enumerate as described in Chapter 9 of the *“USB 2.0 Specification”*.
2. Create a data buffer with the amount of data you are expecting from the host.
3. In the appropriate (even or odd) TX BD for the desired endpoint:
   a) Set up the status register (BDnSTAT) with the correct data toggle (DATA0/1) value and the byte count of the data buffer.
   b) Set up the address register (BDnADR) with the starting address of the data buffer.
   c) Set the UOWN bit of the status register to ‘1’.
4. When the USB module receives an OUT token, it automatically receives the data sent by the host to the buffer. Upon completion, the module updates the status register (BDnSTAT) and sets the Token Complete Interrupt Flag, TRNIF (U1IR<3>).

20.5 Host Mode Operation

The following sections describe how to perform common Host mode tasks. In Host mode, USB transfers are invoked explicitly by the host software. The host software is responsible for the Acknowledge portion of the transfer. Also, all transfers are performed using the USB Endpoint 0 Control register (U1EP0) and Buffer Descriptors.

20.5.1 ENABLE HOST MODE AND DISCOVER A CONNECTED DEVICE

1. Enable Host mode by setting the HOSTEN bit (U1CON<3>). This causes the Host mode control bits in other USB OTG registers to become available.
2. Enable the D+ and D- pull-down resistors by setting the DPPULDWN and DMPULDWN bits (U1OTGCON<5:4>). Disable the D+ and D- pull-up resistors by clearing the DPPULUP and DMPULUP bits (U1OTGCON<7:6>).
3. At this point, SOF generation begins with the SOF counter loaded with 12,000. Eliminate noise on the USB by clearing the SOFEN bit (U1CON<0>) to disable Start-of-Frame (SOF) packet generation.
4. Enable the device attached interrupt by setting the ATTACHIE bit (U1IE<6>).
5. Wait for the device attached interrupt (U1IR<6> = 1). This is signaled by the USB device changing the state of D+ or D- from ‘0’ to ‘1’ (SE0 to J-state). After it occurs, wait 100 ms for the device power to stabilize.
6. Check the state of the JSTATE and SE0 bits in U1CON. If the JSTATE bit (U1CON<7>) is ‘0’, the connecting device is low speed. If the connecting device is low speed, set the LSPDEN and LSPD bits (U1ADDR<7> and U1EP0<7>) to enable low-speed operation.
7. Reset the USB device by setting the USBRST bit (U1CON<4>) for at least 50 ms, sending Reset signaling on the bus. After 50 ms, terminate the Reset by clearing USBRST.
8. In order to keep the connected device from going into suspend, enable the SOF packet generation by setting the SOFEN bit.
9. Wait 10 ms for the device to recover from Reset.
10. Perform enumeration as described by Chapter 9 of the *“USB 2.0 Specification”*.

NOTES:

© 2015-2016 Microchip Technology Inc.

TABLE 36-11: INTERNAL VOLTAGE REGULATOR SPECIFICATIONS

Operating Conditions: -40°C < TA < +85°C (unless otherwise stated)							
Param No.	**Symbol**	**Characteristics**	**Min**	**Typ**	**Max**	**Units**	**Comments**
DVR10	VBG	Internal Band Gap Reference	—	1.2	—	V	
DVR11	TBG	Band Gap Reference Start-up Time	—	1	—	ms	
DVR20	VRGOUT	Regulator Output Voltage	—	1.8	—	V	VDD > 2.0V
DVR21	CEFC	External Filter Capacitor Value	4.7	10	—	μF	Series Resistance < 3Ω recommended; < 5Ω required.
DVR	TVREG	Start-up Time	—	10	—	μs	PMSLP = 1 with any POR or BOR
DVR30	VLVR	Low-Voltage Regulator Output Voltage	—	1.2	—	V	RETEN = 1, $\overline{\text{LPCFG}}$ = 0

TABLE 36-12: VBAT OPERATING VOLTAGE SPECIFICATIONS

Param No.	**Symbol**	**Characteristic**	**Min**	**Typ**	**Max**	**Units**	**Comments**
DVB01	VBT	Operating Voltage	1.6	—	3.6	V	Battery connected to the VBAT pin, VBTBOR = 0
DVB02			VBATBOR	—	3.6	V	Battery connected to the VBAT pin, VBTBOR = 1
DVB10	VBTADC	VBAT A/D Monitoring Voltage Specification[(1)]	1.6	—	3.6	V	A/D is monitoring the VBAT pin using the internal A/D channel

Note 1: Measuring the A/D value using the A/D is represented by the equation:
Measured Voltage = ((VBAT/2)/VDD) * 4096) for 12-bit A/D.

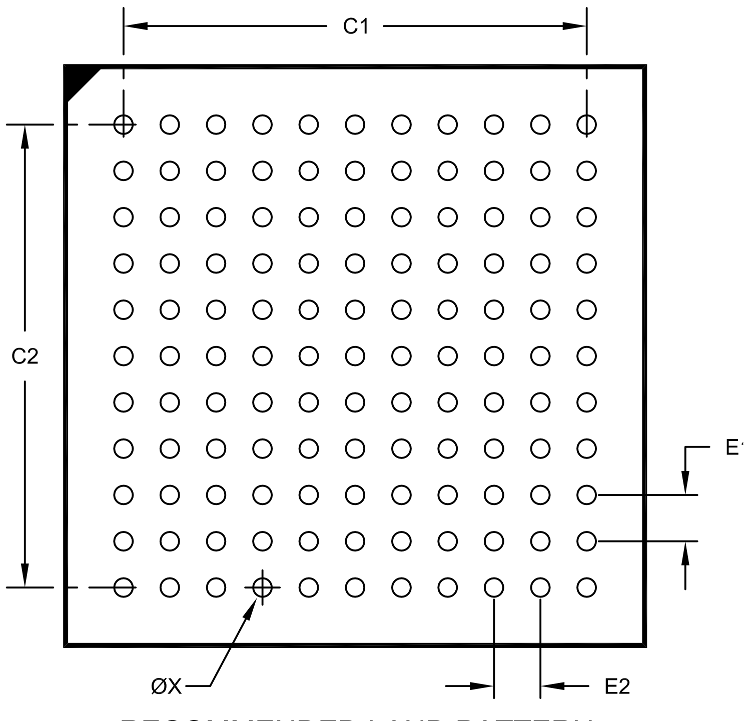
TABLE 36-13: CTMU CURRENT SOURCE SPECIFICATIONS

DC CHARACTERISTICS			**Standard Operating Conditions: 2.0V to 3.6V (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for Industrial					
Param No.	**Sym**	**Characteristic**	**Min**	**Typ[(1)]**	**Max**	**Units**	**Comments**	**Conditions**
DCT10	IOUT1	CTMU Current Source, Base Range	—	550	—	nA	CTMUCON1L<1:0> = 00	2.5V < VDD < VDDMAX
DCT11	IOUT2	CTMU Current Source, 10x Range	—	5.5	—	μA	CTMUCON1L<1:0> = 01	
DCT12	IOUT3	CTMU Current Source, 100x Range	—	55	—	μA	CTMUCON1L<1:0> = 10	
DCT13	IOUT4	CTMU Current Source, 1000x Range	—	550	—	μA	CTMUCON1L<1:0> = 11[(2)]	
DCT21	VΔ	Temperature Diode Voltage Change per Degree Celsius	—	-3	—	mV/°C		

Note 1: Nominal value at center point of current trim range (CTMUCON1L<7:2> = 000000).
2: Do not use this current range with temperature sensing diode.

121-Lead Plastic Thin Profile Ball Grid Array (BG) - 10x10x1.10 mm Body [TFBGA--Formerly XBGA]

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	E1	0.80 BSC		
Contact Pitch	E2	0.80 BSC		
Contact Pad Spacing	C1		8.00	
Contact Pad Spacing	C2		8.00	
Contact Pad Diameter (X121)	X			0.32

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2148 Rev D