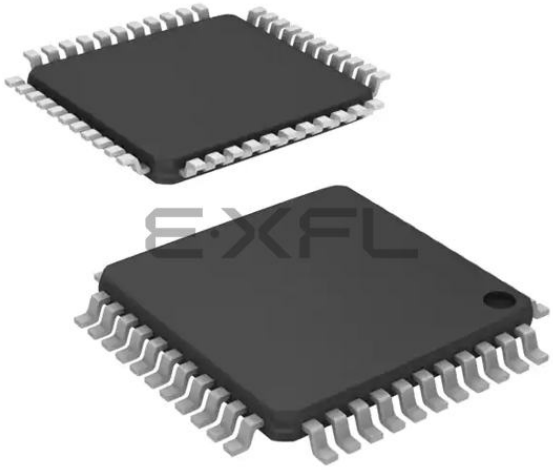


**Welcome to E-XFL.COM**

## 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 | Obsolete |
| Core Processor | PIC |
| Core Size | 8-Bit |
| Speed | 48MHz |
| Connectivity | UART/USART, USB |
| Peripherals | Brown-out Detect/Reset, HLVD, POR, PWM, WDT |
| Number of I/O | 34 |
| Program Memory Size | 16KB (8K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 768 x 8 |
| Voltage - Supply (Vcc/Vdd) | 2V ~ 5.5V |
| Data Converters | A/D 13x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 44-TQFP |
| Supplier Device Package | 44-TQFP (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic18lf4450t-i-pt |

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.



QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

*Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

### 3.2.3 RC_RUN MODE

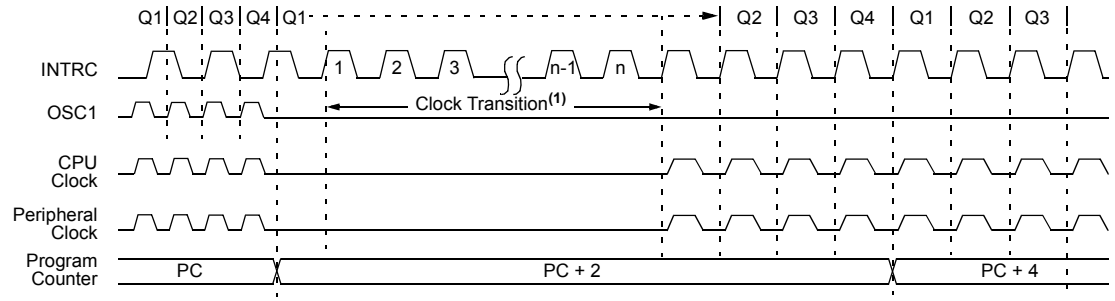
In RC_RUN mode, the CPU and peripherals are clocked from the internal oscillator; the primary clock is shut down. When using the INTRC source, this mode provides the best power conservation of all the Run modes while still executing code. It works well for user applications which are not highly timing sensitive or do not require high-speed clocks at all times.

If the primary clock source is the internal oscillator (INTRC), there are no distinguishable differences between the PRI_RUN and RC_RUN modes during execution. However, a clock switch delay will occur during entry to and exit from RC_RUN mode. Therefore, if the primary clock source is the internal oscillator, the use of RC_RUN mode is not recommended.

This mode is entered by setting SCS1 to '1'. Although it is ignored, it is recommended that SCS0 also be cleared; this is to maintain software compatibility with future devices. When the clock source is switched to the INTRC (see Figure 3-3), the primary oscillator is shut down and the OSTS bit is cleared.

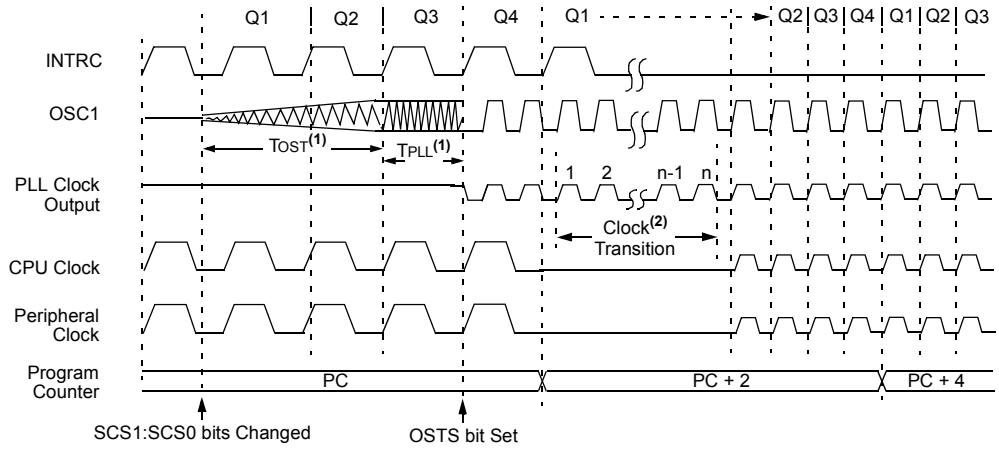
On transitions from RC_RUN mode to PRI_RUN mode, the device continues to be clocked from the INTRC while the primary clock is started. When the primary clock becomes ready, a clock switch to the primary clock occurs (see Figure 3-4). When the clock switch is complete, the OSTS bit is set and the primary clock is providing the device clock. The IDLEN and SCS bits are not affected by the switch. The INTRC source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

**FIGURE 3-3: TRANSITION TIMING TO RC_RUN MODE**




**Note 1:** Clock transition typically occurs within 2-4 TOSC.

**FIGURE 3-4: TRANSITION TIMING FROM RC_RUN MODE TO PRI_RUN MODE**




**Note 1:** TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.

**2:** Clock transition typically occurs within 2-4 TOSC.

### 3.5.4 EXIT WITHOUT AN OSCILLATOR START-UP DELAY

Certain exits from power-managed modes do not invoke the OST at all. There are two cases:

- PRI_IDLE mode, where the primary clock source is not stopped; and
- The primary clock source is not any of the XT or HS modes

In these instances, the primary clock source either does not require an oscillator start-up delay, since it is already running (PRI_IDLE), or normally does not require an oscillator start-up delay (EC and any internal oscillator modes). However, a fixed delay of interval TCSD following the wake event is still required when leaving Sleep and Idle modes to allow the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

**TABLE 3-2: EXIT DELAY ON WAKE-UP BY RESET FROM SLEEP MODE OR ANY IDLE MODE (BY CLOCK SOURCES)**

| Microcontroller Clock Source | | Exit Delay | Clock Ready Status Bit (OSCCON) |
|---|---|---|---|
| **Before Wake-up** | **After Wake-up** | | |
| Primary Device Clock (PRI_IDLE mode) | XT, HS | None | OSTS |
| | XTPLL, HSPLL | | |
| | EC | | |
| | INTRC(1) | | |
| T1OSC or INTRC(1) | XT, HS | TOST(3) | OSTS |
| | XTPLL, HSPLL | TOST + $t_{rc}$(3) | |
| | EC | TCSD(2) | |
| | INTRC(1) | TIOBST(4) | |
| INTRC(1) | XT, HS | TOST(3) | OSTS |
| | XTPLL, HSPLL | TOST + $t_{rc}$(3) | |
| | EC | TCSD(2) | |
| | INTRC(1) | None | |
| None (Sleep mode) | XT, HS | TOST(3) | OSTS |
| | XTPLL, HSPLL | TOST + $t_{rc}$(3) | |
| | EC | TCSD(2) | |
| | INTRC(1) | TIOBST(4) | |

**Note 1:** In this instance, refers specifically to the 31 kHz INTRC clock source.

**2:** TCSD (parameter 38, Table 21-10) is a required delay when waking from Sleep and all Idle modes and runs concurrently with any other required delays (see **Section 3.4 "Idle Modes"**).

**3:** TOST is the Oscillator Start-up Timer period (parameter 32, Table 21-10). $t_{rc}$ is the PLL lock time-out (parameter F12, Table 21-7); it is also designated as TPLL.

**4:** Execution continues during TIOBST (parameter 39, Table 21-10), the INTRC stabilization period.

### 5.2.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSb = 0). To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSb will always read '0' (see **Section 5.1.1 "Program Counter"**).

Figure 5-4 shows an example of how instruction words are stored in the program memory.

The CALL and GOTO instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Figure 5-4 shows how the instruction, GOTO 0006h, is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 19.0 "Instruction Set Summary"** provides further details of the instruction set.

**FIGURE 5-4: INSTRUCTIONS IN PROGRAM MEMORY**

| | | | LSB = 1 | LSB = 0 | Word Address ↓ |
|---|---|---|---|---|---|
| Program Memory Byte Locations → | | | | | 000000h |
| | | | | | 000002h |
| | | | | | 000004h |
| | | | | | 000006h |
| Instruction 1: | MOVLW | 055h | 0Fh | 55h | 000008h |
| Instruction 2: | GOTO | 0006h | EFh | 03h | 00000Ah |
| | | | F0h | 00h | 00000Ch |
| Instruction 3: | MOVFF | 123h, 456h | C1h | 23h | 00000Eh |
| | | | F4h | 56h | 000010h |
| | | | | | 000012h |
| | | | | | 000014h |

### 5.2.4 TWO-WORD INSTRUCTIONS

The standard PIC18 instruction set has four two-word instructions: CALL, MOVFF, GOTO and LSFR. In all cases, the second word of the instructions always has '1111' as its four Most Significant bits; the other 12 bits are literal data, usually a data memory address.

The use of '1111' in the 4 MSbs of an instruction specifies a special form of NOP. If the instruction is executed in proper sequence, immediately after the first word, the data in the second word is accessed and used by the instruction sequence. If the first word is skipped for some reason and the second word is executed by itself, a NOP is executed instead. This is necessary for cases when the two-word instruction is preceded by a conditional instruction that changes the PC. Example 5-4 shows how this works.

| **Note:** | See **Section 5.5 "Program Memory and the Extended Instruction Set"** for information on two-word instruction in the extended instruction set. |
|---|---|

**EXAMPLE 5-4: TWO-WORD INSTRUCTIONS**

| **CASE 1:** | | | |
|---|---|---|---|
| **Object Code** | **Source Code** | | |
| 0110 0110 0000 0000 | TSTFSZ | REG1 | ; is RAM location 0? |
| 1100 0001 0010 0011 | MOVFF | REG1, REG2 | ; No, skip this word |
| 1111 0100 0101 0110 | | | ; Execute this word as a NOP |
| 0010 0100 0000 0000 | ADDWF | REG3 | ; continue code |
| **CASE 2:** | | | |
| **Object Code** | **Source Code** | | |
| 0110 0110 0000 0000 | TSTFSZ | REG1 | ; is RAM location 0? |
| 1100 0001 0010 0011 | MOVFF | REG1, REG2 | ; Yes, execute this word |
| 1111 0100 0101 0110 | | | ; 2nd word of instruction |
| 0010 0100 0000 0000 | ADDWF | REG3 | ; continue code |

**TABLE 5-2: REGISTER FILE SUMMARY (PIC18F2450/4450)**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| TOSU | — | — | — | Top-of-Stack Upper Byte (TOS<20:16>) | | | | | ---0 0000 | 49, 54 |
| TOSH | Top-of-Stack High Byte (TOS<15:8>) | | | | | | | | 0000 0000 | 49, 54 |
| TOSL | Top-of-Stack Low Byte (TOS<7:0>) | | | | | | | | 0000 0000 | 49, 54 |
| STKPTR | STKFUL | STKUNF | — | SP4 | SP3 | SP2 | SP1 | SP0 | 00-0 0000 | 49, 55 |
| PCLATU | — | — | — | Holding Register for PC<20:16> | | | | | ---0 0000 | 49, 54 |
| PCLATH | Holding Register for PC<15:8> | | | | | | | | 0000 0000 | 49, 54 |
| PCL | PC Low Byte (PC<7:0>) | | | | | | | | 0000 0000 | 49, 54 |
| TBLPTRU | — | — | bit 21(1) | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | | --00 0000 | 49, 76 |
| TBLPTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | | 0000 0000 | 49, 76 |
| TBLPTRL | Program Memory Table Pointer Low Byte (TBLPTR<7:0>) | | | | | | | | 0000 0000 | 49, 76 |
| TABLAT | Program Memory Table Latch | | | | | | | | 0000 0000 | 49, 76 |
| PRODH | Product Register High Byte | | | | | | | | xxxx xxxx | 49, 83 |
| PRODL | Product Register Low Byte | | | | | | | | xxxx xxxx | 49, 83 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 49, 87 |
| INTCON2 | $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | RBIP | 1111 -1-1 | 49, 88 |
| INTCON3 | INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF | 11-0 0-00 | 49, 89 |
| INDF0 | Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register) | | | | | | | | N/A | 49, 68 |
| POSTINC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) | | | | | | | | N/A | 49, 69 |
| POSTDEC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register) | | | | | | | | N/A | 49, 69 |
| PREINC0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) | | | | | | | | N/A | 49, 69 |
| PLUSW0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W | | | | | | | | N/A | 49, 69 |
| FSR0H | — | — | — | — | Indirect Data Memory Address Pointer 0 High Byte | | | | ---- 0000 | 49, 68 |
| FSR0L | Indirect Data Memory Address Pointer 0 Low Byte | | | | | | | | xxxx xxxx | 49, 68 |
| WREG | Working Register | | | | | | | | xxxx xxxx | 49, |
| INDF1 | Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register) | | | | | | | | N/A | 49, 68 |
| POSTINC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) | | | | | | | | N/A | 49, 69 |
| POSTDEC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register) | | | | | | | | N/A | 49, 69 |
| PREINC1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) | | | | | | | | N/A | 49, 69 |
| PLUSW1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W | | | | | | | | N/A | 49, 69 |
| FSR1H | — | — | — | — | Indirect Data Memory Address Pointer 1 High Byte | | | | ---- 0000 | 49, 68 |
| FSR1L | Indirect Data Memory Address Pointer 1 Low Byte | | | | | | | | xxxx xxxx | 49, 68 |
| BSR | — | — | — | — | Bank Select Register | | | | ---- 0000 | 49, 59 |
| INDF2 | Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) | | | | | | | | N/A | 50, 68 |
| POSTINC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) | | | | | | | | N/A | 50, 69 |
| POSTDEC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) | | | | | | | | N/A | 50, 69 |
| PREINC2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) | | | | | | | | N/A | 50, 69 |
| PLUSW2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by W | | | | | | | | N/A | 50, 69 |
| FSR2H | — | — | — | — | Indirect Data Memory Address Pointer 2 High Byte | | | | ---- 0000 | 50, 68 |
| FSR2L | Indirect Data Memory Address Pointer 2 Low Byte | | | | | | | | xxxx xxxx | 50, 68 |
| STATUS | — | — | — | N | OV | Z | DC | C | ---x xxxx | 50, 66 |
| TMR0H | Timer0 Register High Byte | | | | | | | | 0000 0000 | 50, 113 |
| TMR0L | Timer0 Register Low Byte | | | | | | | | xxxx xxxx | 50, 113 |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 | 1111 1111 | 50, 111 |

**Legend:** x = unknown, u = unchanged, - = unimplemented, q = value depends on condition. Shaded cells are unimplemented, read as '0'.

**Note 1:** Bit 21 of the TBLPTRU allows access to the device Configuration bits.

**2:** The SBOREN bit is only available when BOREN<1:0> = 01; otherwise, the bit reads as '0'.

**3:** These registers and/or bits are not implemented on 28-pin devices and are read as '0'. Reset values are shown for 40/44-pin devices; individual unimplemented bits should be interpreted as '–'.

**4:** RA6 is configured as a port pin based on various primary oscillator modes. When the port pin is disabled, all of the associated bits read '0'.

**5:** RE3 is only available as a port pin when the MCLRE Configuration bit is clear; otherwise, the bit reads as '0'.

**6:** RC5 and RC4 are only available as port pins when the USB module is disabled (UCON<3> = 0).

**TABLE 5-2: REGISTER FILE SUMMARY (PIC18F2450/4450) (CONTINUED)**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| OSCCON | IDLEN | — | — | — | OSTS | — | SCS1 | SCS0 | 0--- q-00 | 50, 31 |
| HLVDCON | VDIRMAG | — | IRVST | HLVDEN | HLVDL3 | HLVDL2 | HLVDL1 | HLVDL0 | 0-00 0101 | 50, 185 |
| WDTCON | — | — | — | — | — | — | — | SWDTEN | --- ---0 | 50, 204 |
| RCON | IPEN | SBOREN(2) | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 0q-1 11q0 | 50, 42 |
| TMR1H | Timer1 Register High Byte | | | | | | | | xxxx xxxx | 50, 120 |
| TMR1L | Timer1 Register Low Byte | | | | | | | | xxxx xxxx | 50, 120 |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{T1SYNC}$ | TMR1CS | TMR1ON | 0000 0000 | 50, 115 |
| TMR2 | Timer2 Register | | | | | | | | 0000 0000 | 50, 122 |
| PR2 | Timer2 Period Register | | | | | | | | 1111 1111 | 50, 122 |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | -000 0000 | 50, 121 |
| ADRESH | A/D Result Register High Byte | | | | | | | | xxxx xxxx | 50, 184 |
| ADRESL | A/D Result Register Low Byte | | | | | | | | xxxx xxxx | 50, 184 |
| ADCON0 | — | — | CHS3 | CHS2 | CHS1 | CHS0 | GO/$\overline{DONE}$ | ADON | --00 0000 | 50, 175 |
| ADCON1 | — | — | VCFG1 | VCFG0 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | --00 qqqq | 50, 176 |
| ADCON2 | ADFM | — | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 | 0-00 0000 | 50, 177 |
| CCPR1H | Capture/Compare/PWM Register 1 High Byte | | | | | | | | xxxx xxxx | 50, 124 |
| CCPR1L | Capture/Compare/PWM Register 1 Low Byte | | | | | | | | xxxx xxxx | 50, 124 |
| CCP1CON | — | — | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | --00 0000 | 50, 123, |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 01-0 0-00 | 51, 156, |
| SPBRGH | EUSART Baud Rate Generator Register High Byte | | | | | | | | 0000 0000 | 50, 157 |
| SPBRG | EUSART Baud Rate Generator Register Low Byte | | | | | | | | 0000 0000 | 50, 157 |
| RCREG | EUSART Receive Register | | | | | | | | 0000 0000 | 50, 165 |
| TXREG | EUSART Transmit Register | | | | | | | | 0000 0000 | 51, 163 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 51, 154 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 51, 155 |
| EECON2 | Data Memory Control Register 2 (not a physical register) | | | | | | | | 0000 0000 | 51, 74 |
| EECON1 | — | CFGS | — | FREE | WRERR | WREN | WR | — | -x-0 x00- | 51, 75 |
| IPR2 | OSCFIP | — | USBIP | — | — | HLVDIP | — | — | 1-1- -1-- | 51, 95 |
| PIR2 | OSCFIF | — | USBIF | — | — | HLVDIF | — | — | 0-0- -0-- | 51, 91 |
| PIE2 | OSCFIE | — | USBIE | — | — | HLVDIE | — | — | 0-0- -0-- | 51, 93 |
| IPR1 | — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP | -111 -111 | 51, 94 |
| PIR1 | — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF | -000 -000 | 51, 90 |
| PIE1 | — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE | -000 -000 | 51, 92 |
| TRISE(3) | — | — | — | — | — | TRISE2 | TRISE1 | TRISE0 | ---- -111 | 51, 110 |
| TRISD(3) | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 1111 1111 | 51, 108 |
| TRISC | TRISC7 | TRISC6 | — | — | — | TRISC2 | TRISC1 | TRISC0 | 11-- -111 | 51, 106 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 1111 1111 | 51, 103 |
| TRISA | — | TRISA6(4) | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | -111 1111 | 51, 100 |
| LATE(3) | — | — | — | — | — | LATE2 | LATE1 | LATE0 | ---- -xxx | 51, 110 |
| LATD(3) | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 | xxxx xxxx | 51, 108 |
| LATC | LATC7 | LATC6 | — | — | — | LATC2 | LATC1 | LATC0 | xx-- -xxx | 51, 106 |
| LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 | xxxx xxxx | 51, 103 |
| LATA | — | LATA6(4) | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 | -xxx xxxx | 51, 100 |
| PORTE | — | — | — | — | RE3(5) | RE2(3) | RE1(3) | RE0(3) | ---- x000 | 51, 109 |
| PORTD(3) | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | xxxx xxxx | 51, 108 |

**Legend:** x = unknown, u = unchanged, - = unimplemented, q = value depends on condition. Shaded cells are unimplemented, read as '0'.

**Note 1:** Bit 21 of the TBLPTRU allows access to the device Configuration bits.

**2:** The SBOREN bit is only available when BOREN<1:0> = 01; otherwise, the bit reads as '0'.

**3:** These registers and/or bits are not implemented on 28-pin devices and are read as '0'. Reset values are shown for 40/44-pin devices; individual unimplemented bits should be interpreted as '-'.

**4:** RA6 is configured as a port pin based on various primary oscillator modes. When the port pin is disabled, all of the associated bits read '0'.

**5:** RE3 is only available as a port pin when the MCLRE Configuration bit is clear; otherwise, the bit reads as '0'.

**6:** RC5 and RC4 are only available as port pins when the USB module is disabled (UCON<3> = 0).

## 5.5 Program Memory and the Extended Instruction Set

The operation of program memory is unaffected by the use of the extended instruction set.

Enabling the extended instruction set adds eight additional two-word commands to the existing PIC18 instruction set: ADDFSR, ADDULNK, CALLW, MOVSF, MOVSS, PUSHL, SUBFSR and SUBULNK. These instructions are executed as described in **Section 5.2.4 "Two-Word Instructions"**.

## 5.6 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = 1) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different. This is due to the introduction of a new addressing mode for the data memory space. This mode also alters the behavior of Indirect Addressing using FSR2 and its associated operands.

What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect Addressing mode; inherent and literal instructions do not change at all. Indirect Addressing with FSR0 and FSR1 also remains unchanged.

### 5.6.1 INDEXED ADDRESSING WITH LITERAL OFFSET

Enabling the PIC18 extended instruction set changes the behavior of Indirect Addressing using the FSR2 register pair and its associated file operands. Under the proper conditions, instructions that use the Access Bank – that is, most bit-oriented and byte-oriented instructions – can invoke a form of Indexed Addressing using an offset specified in the instruction. This special addressing mode is known as Indexed Addressing with Literal Offset or Indexed Literal Offset mode.

When using the extended instruction set, this addressing mode requires the following:

- The use of the Access Bank is forced ('a' = 0); and
- The file address argument is less than or equal to 5Fh.

Under these conditions, the file address of the instruction is not interpreted as the lower byte of an address (used with the BSR in Direct Addressing), or as an 8-bit address in the Access Bank. Instead, the value is interpreted as an offset value to an Address Pointer specified by FSR2. The offset and the contents of FSR2 are added to obtain the target address of the operation.

### 5.6.2 INSTRUCTIONS AFFECTED BY INDEXED LITERAL OFFSET MODE

Any of the core PIC18 instructions that can use Direct Addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte-oriented and bit-oriented instructions, or almost one-half of the standard PIC18 instruction set. Instructions that only use Inherent or Literal Addressing modes are unaffected.

Additionally, byte-oriented and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit is '1') or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled in shown in Figure 5-8.

Those who desire to use byte-oriented or bit-oriented instructions in the Indexed Literal Offset mode should note the changes to assembler syntax for this mode. This is described in more detail in **Section 19.2.1 "Extended Instruction Syntax"**.

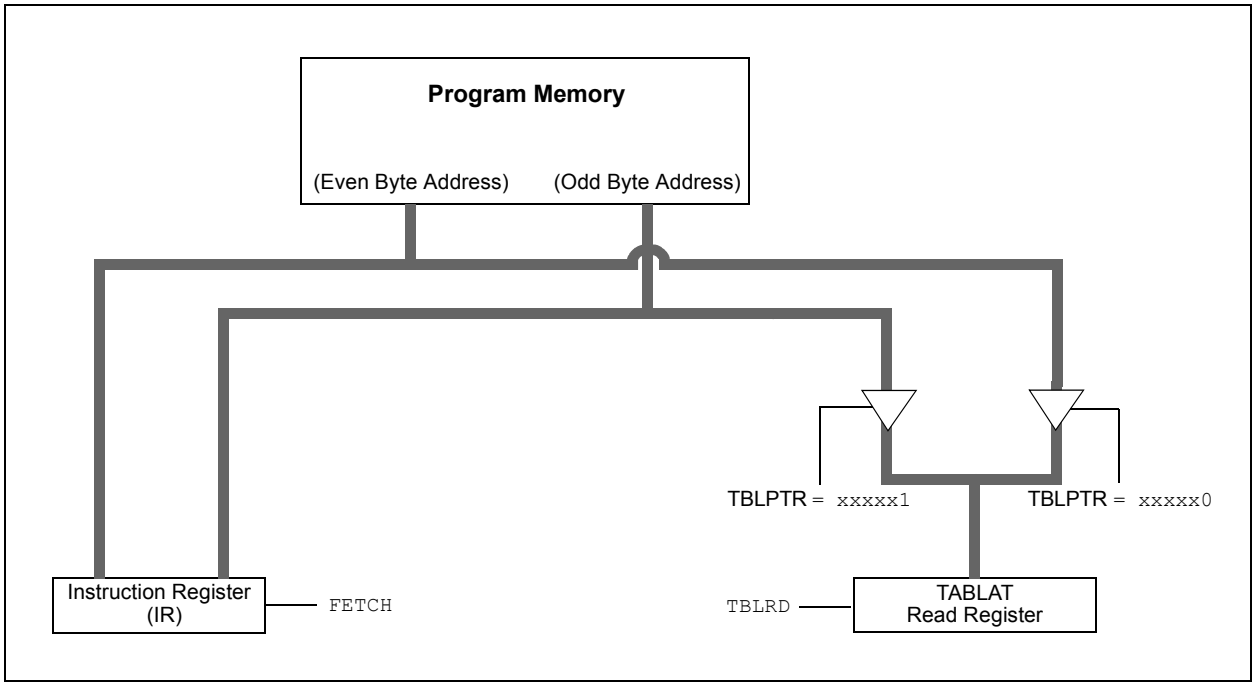
## 6.3 Reading the Flash Program Memory

The TBLRD instruction is used to retrieve data from program memory and places it into data RAM. Table reads from program memory are performed one byte at a time.

TBLPTR points to a byte address in program space. Executing TBLRD places the byte pointed to into TABLAT. In addition, TBLPTR can be modified automatically for the next table read operation.

The internal program memory is typically organized by words. The Least Significant bit of the address selects between the high and low bytes of the word. Figure 6-4 shows the interface between the internal program memory and the TABLAT.

**FIGURE 6-4: READS FROM FLASH PROGRAM MEMORY**




**EXAMPLE 6-1: READING A FLASH PROGRAM MEMORY WORD**

```
            MOVLW     CODE_ADDR_UPPER         ; Load TBLPTR with the base
            MOVWF     TBLPTRU                 ; address of the word
            MOVLW     CODE_ADDR_HIGH
            MOVWF     TBLPTRH
            MOVLW     CODE_ADDR_LOW
            MOVWF     TBLPTRL
READ_WORD
            TBLRD*+                           ; read into TABLAT and increment
            MOVF      TABLAT, W               ; get data
            MOVWF     WORD_EVEN
            TBLRD*+                           ; read into TABLAT and increment
            MOVF      TABLAT, W               ; get data
            MOVF      WORD_ODD
```

**REGISTER 8-2: INTCON2: INTERRUPT CONTROL REGISTER 2**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | U-0 | R/W-1 | U-0 | R/W-1 |
|---|---|---|---|---|---|---|---|
| $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | RBIP |
| 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 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = All PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG0:** External Interrupt 0 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 5 **INTEDG1:** External Interrupt 1 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 4 **INTEDG2:** External Interrupt 2 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

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

bit 2 **TMR0IP:** TMR0 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

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

bit 0 **RBIP:** RB Port Change Interrupt Priority bit
1 = High priority
0 = Low priority

**Note:** Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global interrupt enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

**REGISTER 8-7: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2**

| R/W-0 | U-0 | R/W-0 | U-0 | U-0 | R/W-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| OSCFIE | — | USBIE | — | — | HLVDIE | — | — |
| 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 7 **OSCFIE:** Oscillator Fail Interrupt Enable bit
1 = Enabled
0 = Disabled

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

bit 5 **USBIE:** USB Interrupt Enable bit
1 = Enabled
0 = Disabled

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

bit 2 **HLVDIE:** High/Low-Voltage Detect Interrupt Enable bit
1 = Enabled
0 = Disabled

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

## 14.0 UNIVERSAL SERIAL BUS (USB)

This section describes the details of the USB peripheral. Because of the very specific nature of the module, knowledge of USB is expected. Some high-level USB information is provided in **Section 14.9 "Overview of USB"** only for application design reference. Designers are encouraged to refer to the official specification published by the USB Implementers Forum (USB-IF) for the latest information. USB Specification Revision 2.0 is the most current specification at the time of publication of this document.

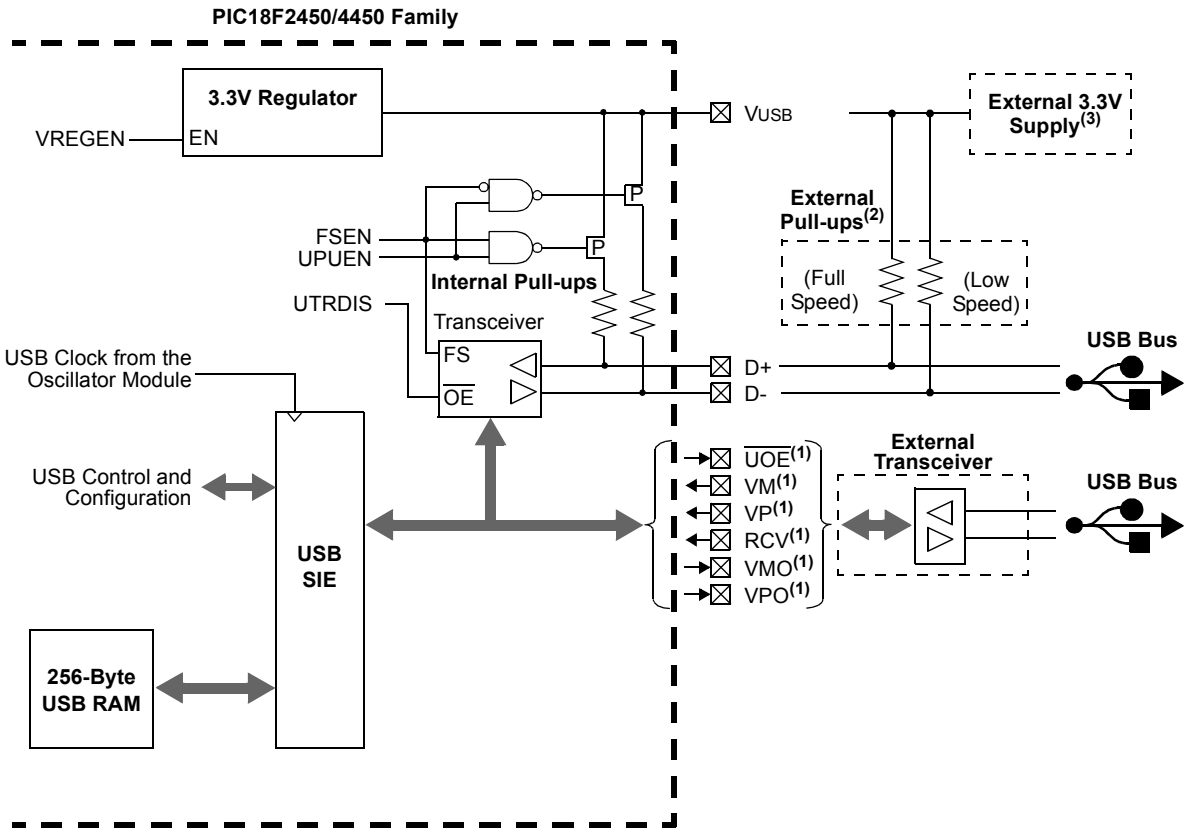
### 14.1 Overview of the USB Peripheral

The PIC18F2450/4450 device family contains a full-speed and low-speed, compatible USB Serial Interface Engine (SIE) that allows fast communication between any USB host and the PIC® microcontroller. The SIE can be interfaced directly to the USB, utilizing the internal transceiver, or it can be connected through an external transceiver. An internal 3.3V regulator is also available to power the internal transceiver in 5V applications.

Some special hardware features have been included to improve performance. Dual port memory in the device's data memory space (USB RAM) has been supplied to share direct memory access between the microcontroller core and the SIE. Buffer descriptors are also provided, allowing users to freely program endpoint memory usage within the USB RAM space.

Figure 14-1 presents a general overview of the USB peripheral and its features.

**FIGURE 14-1: USB PERIPHERAL AND OPTIONS**




**Note 1:** This signal is only available if the internal transceiver is disabled (UTRDIS = 1).

**2:** The pull-ups can be supplied either from the VUSB pin or from an external 3.3V supply.

**3:** Do not enable the internal regulator when using an external 3.3V supply.

**TABLE 14-1: DIFFERENTIAL OUTPUTS TO TRANSCEIVER**

| VPO | VMO | Bus State |
|---|---|---|
| 0 | 0 | Single-Ended Zero |
| 0 | 1 | Differential '0' |
| 1 | 0 | Differential '1' |
| 1 | 1 | Illegal Condition |

**TABLE 14-2: SINGLE-ENDED INPUTS FROM TRANSCEIVER**

| VP | VM | Bus State |
|---|---|---|
| 0 | 0 | Single-Ended Zero |
| 0 | 1 | Low Speed |
| 1 | 0 | High Speed |
| 1 | 1 | Error |

The $\overline{UOE}$ signal toggles the state of the external transceiver. This line is pulled low by the device to enable the transmission of data from the SIE to an external device.

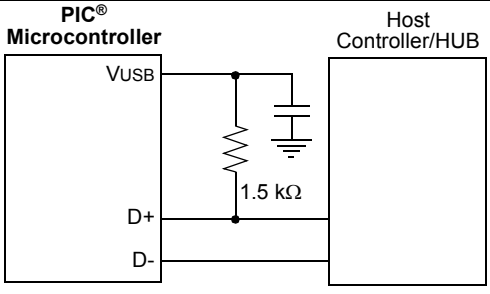
#### 14.2.2.3 Internal Pull-up Resistors

The PIC18F2450/4450 devices have built-in pull-up resistors designed to meet the requirements for low-speed and full-speed USB. The UPUEN bit (UCFG<4>) enables the internal pull-ups. Figure 14-1 shows the pull-ups and their control.

#### 14.2.2.4 Pull-up Resistors

The PIC18F2450/4450 devices require an external pull-up resistor to meet the requirements for low-speed and full-speed USB. Either an external 3.3V supply or the VUSB pin may be used to pull up D+ or D-. The pull-up resistor must be 1.5 kΩ (±5%) as required by the USB specifications. Figure 14-3 shows an example with the VUSB pin.

**FIGURE 14-3: EXTERNAL CIRCUITRY**




**Note:** The above setting shows a typical connection for a full-speed configuration using an on-chip regulator and an external pull-up resistor.

#### 14.2.2.5 Ping-Pong Buffer Configuration

The usage of ping-pong buffers is configured using the PPB1:PPB0 bits. Refer to **Section 14.4.4 "Ping-Pong Buffering"** for a complete explanation of the ping-pong buffers.

#### 14.2.2.6 USB Output Enable Monitor

The USB $\overline{OE}$ monitor provides indication as to whether the SIE is listening to the bus or actively driving the bus. This is enabled by default when using an external transceiver or when UCFG<6> = 1.

The USB $\overline{OE}$ monitoring is useful for initial system debugging, as well as scope triggering during eye pattern generation tests.

#### 14.2.2.7 Eye Pattern Test Enable

An automatic eye pattern test can be generated by the module when the UCFG<7> bit is set. The eye pattern output will be observable based on module settings, meaning that the user is first responsible for configuring the SIE clock settings, pull-up resistor and Transceiver mode. In addition, the module has to be enabled.

Once UTEYE is set, the module emulates a switch from a receive to transmit state and will start transmitting a J-K-J-K bit sequence (K-J-K-J for full speed). The sequence will be repeated indefinitely while the Eye Pattern Test mode is enabled.

Note that this bit should never be set while the module is connected to an actual USB system. This test mode is intended for board verification to aid with USB certification tests. It is intended to show a system developer the noise integrity of the USB signals which can be affected by board traces, impedance mismatches and proximity to other system components. It does not properly test the transition from a receive to a transmit state. Although the eye pattern is not meant to replace the more complex USB certification test, it should aid during first order system debugging.

#### 14.2.2.8 Internal Regulator

The PIC18F2450/4450 devices have a built-in 3.3V regulator to provide power to the internal transceiver and provide a source for the external pull-ups. An external 220 nF (±20%) capacitor is required for stability.

**Note:** The drive from VUSB is sufficient to only drive an external pull-up in addition to the internal transceiver.

The regulator is disabled by default and can be enabled through the VREGEN Configuration bit. When enabled, the voltage is visible on pin VUSB. When the regulator is disabled, a 3.3V source must be provided through the VUSB pin for the internal transceiver. If the internal transceiver is disabled, VUSB is not used.

**Note 1:** Do not enable the internal regulator if an external regulator is connected to VUSB.

**2:** VDD must be greater than or equal to VUSB at all times, even with the regulator disabled.

### 14.2.4 USB ENDPOINT CONTROL

Each of the 16 possible bidirectional endpoints has its own independent control register, UEPn (where 'n' represents the endpoint number). Each register has an identical complement of control bits. The prototype is shown in Register 14-4.

The EPHSHK bit (UEPn<4>) controls handshaking for the endpoint; setting this bit enables USB handshaking. Typically, this bit is always set except when using isochronous endpoints.

The EPCONDIS bit (UEPn<3>) is used to enable or disable USB control operations (SETUP) through the endpoint. Clearing this bit enables SETUP transactions. Note that the corresponding EPINEN and EPOUTEN bits must be set to enable IN and OUT transactions. For Endpoint 0, this bit should always be cleared since the USB specifications identify Endpoint 0 as the default control endpoint.

The EPOUTEN bit (UEPn<2>) is used to enable or disable USB OUT transactions from the host. Setting this bit enables OUT transactions. Similarly, the EPINEN bit (UEPn<1>) enables or disables USB IN transactions from the host.

The EPSTALL bit (UEPn<0>) is used to indicate a STALL condition for the endpoint. If a STALL is issued on a particular endpoint, the EPSTALL bit for that endpoint pair will be set by the SIE. This bit remains set until it is cleared through firmware, or until the SIE is reset.

**REGISTER 14-4: UEPn: USB ENDPOINT n CONTROL REGISTER (UEP0 THROUGH UEP15)**

| U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL(1) |
| 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 7-5 **Unimplemented:** Read as '0'

bit 4 **EPHSHK:** Endpoint Handshake Enable bit
1 = Endpoint handshake enabled
0 = Endpoint handshake disabled (typically used for isochronous endpoints)

bit 3 **EPCONDIS:** Bidirectional Endpoint Control bit
If EPOUTEN = 1 and EPINEN = 1:
1 = Disable Endpoint n from control transfers; only IN and OUT transfers allowed
0 = Enable Endpoint n for control (SETUP) transfers; IN and OUT transfers also allowed

bit 2 **EPOUTEN:** Endpoint Output Enable bit
1 = Endpoint n output enabled
0 = Endpoint n output disabled

bit 1 **EPINEN:** Endpoint Input Enable bit
1 = Endpoint n input enabled
0 = Endpoint n input disabled

bit 0 **EPSTALL:** Endpoint Stall Indicator bit
1 = Endpoint n has issued one or more STALL packets
0 = Endpoint n has not issued any STALL packets

#### 14.4.1.3 BDnSTAT Register (SIE Mode)

When the BD and its buffer are owned by the SIE, most of the bits in BDnSTAT take on a different meaning. The configuration is shown in Register 14-6. Once UOWN is set, any data or control settings previously written there by the user will be overwritten with data from the SIE.

The BDnSTAT register is updated by the SIE with the token Packet Identifier (PID) which is stored in BDnSTAT<5:3>. The transfer count in the corresponding BDnCNT register is updated. Values that overflow the 8-bit register carry over to the two most significant digits of the count, stored in BDnSTAT<1:0>.

### 14.4.2 BD BYTE COUNT

The byte count represents the total number of bytes that will be transmitted during an IN transfer. After an IN transfer, the SIE will return the number of bytes sent to the host.

For an OUT transfer, the byte count represents the maximum number of bytes that can be received and stored in USB RAM. After an OUT transfer, the SIE will return the actual number of bytes received. If the number of bytes received exceeds the corresponding byte count, the data packet will be rejected and a NAK handshake will be generated. When this happens, the byte count will not be updated.

The 10-bit byte count is distributed over two registers. The lower 8 bits of the count reside in the BDnCNT register. The upper two bits reside in BDnSTAT<1:0>. This represents a valid byte range of 0 to 1023.

### 14.4.3 BD ADDRESS VALIDATION

The BD Address register pair contains the starting RAM address location for the corresponding endpoint buffer. For an endpoint starting location to be valid, it must fall in the range of the USB RAM, 400h to 4FFh. No mechanism is available in hardware to validate the BD address.

If the value of the BD address does not point to an address in the USB RAM, or if it points to an address within another endpoint's buffer, data is likely to be lost or overwritten. Similarly, overlapping a receive buffer (OUT endpoint) with a BD location in use can yield unexpected results. When developing USB applications, the user may want to consider the inclusion of software-based address validation in their code.

**REGISTER 14-6: BDnSTAT: BUFFER DESCRIPTOR n STATUS REGISTER (BD0STAT THROUGH BD63STAT), SIE MODE (DATA RETURNED BY THE SIDE TO THE MICROCONTROLLER)**

| R/W-x | U-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| UOWN | — | PID3 | PID2 | PID1 | PID0 | BC9 | BC8 |
| 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 7 **UOWN:** USB Own bit
1 = The SIE owns the BD and its corresponding buffer

bit 6 **Reserved:** Not written by the SIE

bit 5-2 **PID3:PID0:** Packet Identifier bits
The received token PID value of the last transfer (IN, OUT or SETUP transactions only).

bit 1-0 **BC9:BC8:** Byte Count 9 and 8 bits
These bits are updated by the SIE to reflect the actual number of bytes received on an OUT transfer and the actual number of bytes transmitted on an IN transfer.

### 14.5.3 USB ERROR INTERRUPT STATUS REGISTER (UEIR)

The USB Error Interrupt Status register (Register 14-9) contains the flag bits for each of the error sources within the USB peripheral. Each of these sources is controlled by a corresponding interrupt enable bit in the UEIE register. All of the USB error flags are ORed together to generate the USB Error Interrupt Flag (UERRIF) at the top level of the interrupt logic.

Each error bit is set as soon as the error condition is detected. Thus, the interrupt will typically not correspond with the end of a token being processed.

Once an interrupt bit has been set by the SIE, it must be cleared by software by writing a '0'.

**REGISTER 14-9: UEIR: USB ERROR INTERRUPT STATUS REGISTER**

| R/C-0 | U-0 | U-0 | R/C-0 | R/C-0 | R/C-0 | R/C-0 | R/C-0 |
|---|---|---|---|---|---|---|---|
| BTSEF | — | — | BTOEF | DFN8EF | CRC16EF | CRC5EF | PIDEF |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | C = Clearable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **BTSEF:** Bit Stuff Error Flag bit
1 = A bit stuff error has been detected
0 = No bit stuff error

bit 6-5 **Unimplemented:** Read as '0'

bit 4 **BTOEF:** Bus Turnaround Time-out Error Flag bit
1 = Bus turnaround time-out has occurred (more than 16 bit times of Idle from previous EOP elapsed)
0 = No bus turnaround time-out

bit 3 **DFN8EF:** Data Field Size Error Flag bit
1 = The data field was not an integral number of bytes
0 = The data field was an integral number of bytes

bit 2 **CRC16EF:** CRC16 Failure Flag bit
1 = The CRC16 failed
0 = The CRC16 passed

bit 1 **CRC5EF:** CRC5 Host Error Flag bit
1 = The token packet was rejected due to a CRC5 error
0 = The token packet was accepted

bit 0 **PIDEF:** PID Check Failure Flag bit
1 = PID check failed
0 = PID check passed

## 15.1 Baud Rate Generator (BRG)

The BRG is a dedicated, 8-bit or 16-bit generator that supports both the Asynchronous and Synchronous modes of the EUSART. By default, the BRG operates in 8-bit mode. Setting the BRG16 bit (BAUDCON<3>) selects 16-bit mode.

The SPBRGH:SPBRG register pair controls the period of a free-running timer. In Asynchronous mode, bits BRGH (TXSTA<2>) and BRG16 (BAUDCON<3>) also control the baud rate. In Synchronous mode, BRGH is ignored. Table 15-1 shows the formula for computation of the baud rate for different EUSART modes which only apply in Master mode (internally generated clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRGH:SPBRG registers can be calculated using the formulas in Table 15-1. From this, the error in baud rate can be determined. An example calculation is shown in Example 15-1. Typical baud rates and error values for the various Asynchronous modes are shown in Table 15-2. It may be advantageous to use the high baud rate (BRGH = 1) or the 16-bit BRG to reduce the baud rate error, or achieve a slow baud rate for a fast oscillator frequency.

Writing a new value to the SPBRGH:SPBRG registers causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

### 15.1.1 OPERATION IN POWER-MANAGED MODES

The device clock is used to generate the desired baud rate. When one of the power-managed modes is entered, the new clock source may be operating at a different frequency. This may require an adjustment to the value in the SPBRG register pair.

### 15.1.2 SAMPLING

The data on the RX pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

**TABLE 15-1: BAUD RATE FORMULAS**

| Configuration Bits | | | BRG/EUSART Mode | Baud Rate Formula |
|---|---|---|---|---|
| SYNC | BRG16 | BRGH | | |
| 0 | 0 | 0 | 8-Bit/Asynchronous | FOSC/[64 (n + 1)] |
| 0 | 0 | 1 | 8-Bit/Asynchronous | FOSC/[16 (n + 1)] |
| 0 | 1 | 0 | 16-Bit/Asynchronous | |
| 0 | 1 | 1 | 16-Bit/Asynchronous | FOSC/[4 (n + 1)] |
| 1 | 0 | x | 8-Bit/Synchronous | |
| 1 | 1 | x | 16-Bit/Synchronous | |

**Legend:** x = Don't care, n = Value of SPBRGH:SPBRG register pair

### 15.2.4 AUTO-WAKE-UP ON SYNC BREAK CHARACTER

During Sleep mode, all clocks to the EUSART are suspended. Therefore, the Baud Rate Generator is inactive and proper byte reception cannot be performed. The auto-wake-up feature allows the controller to wake-up due to activity on the RX/DT line while the EUSART is operating in Asynchronous mode.

The auto-wake-up feature is enabled by setting the WUE bit (BAUDCON<1>). Once set, the typical receive sequence on RX/DT is disabled and the EUSART remains in an Idle state, monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a high-to-low transition on the RX/DT line. (This coincides with the start of a Sync Break or a Wake-up Signal character for the LIN protocol.)

Following a wake-up event, the module generates an RCIF interrupt. The interrupt is generated synchronously to the Q clocks in normal operating modes (Figure 15-8) and asynchronously if the device is in Sleep mode (Figure 15-9). The interrupt condition is cleared by reading the RCREG register.

The WUE bit is automatically cleared once a low-to-high transition is observed on the RX line following the wake-up event. At this point, the EUSART module is in Idle mode and returns to normal operation. This signals to the user that the Sync Break event is over.

#### 15.2.4.1 Special Considerations Using Auto-Wake-up

Since auto-wake-up functions by sensing rising edge transitions on RX/DT, information with any state changes before the Stop bit may signal a false End-of-Character (EOC) and cause data or framing errors. To work properly, therefore, the initial character in the transmission must be all '0's. This can be 00h (8 bits) for standard RS-232 devices or 000h (12 bits) for LIN bus.

Oscillator start-up time must also be considered, especially in applications using oscillators with longer start-up intervals (i.e., XT or HS mode). The Sync Break (or Wake-up Signal) character must be of sufficient length and be followed by a sufficient interval to allow enough time for the selected oscillator to start and provide proper initialization of the EUSART.

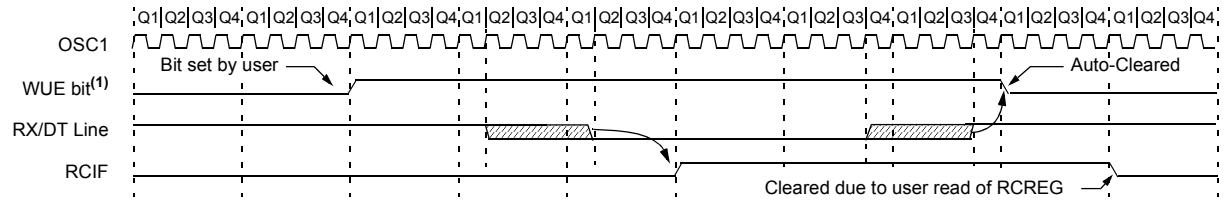
#### 15.2.4.2 Special Considerations Using the WUE Bit

The timing of WUE and RCIF events may cause some confusion when it comes to determining the validity of received data. As noted, setting the WUE bit places the EUSART in an Idle mode. The wake-up event causes a receive interrupt by setting the RCIF bit. The WUE bit is cleared after this when a rising edge is seen on RX/DT. The interrupt condition is then cleared by reading the RCREG register. Ordinarily, the data in RCREG will be dummy data and should be discarded.

The fact that the WUE bit has been cleared (or is still set) and the RCIF flag is set should not be used as an indicator of the integrity of the data in RCREG. Users should consider implementing a parallel method in firmware to verify received data integrity.

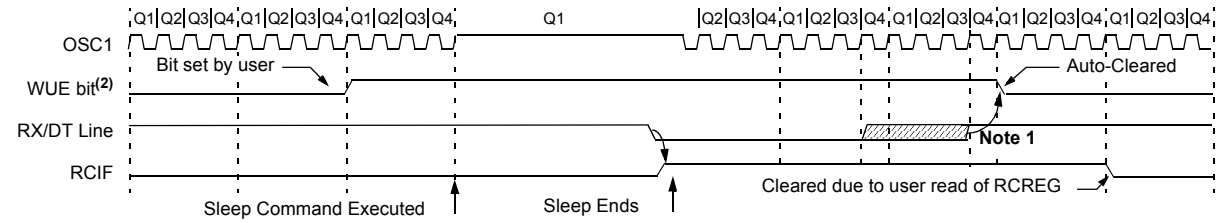
To assure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

**FIGURE 15-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION**




**Note 1:** The EUSART remains in Idle while the WUE bit is set.

**FIGURE 15-9: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP**




**Note 1:** If the wake-up event requires long oscillator warm-up time, the auto-clear of the WUE bit can occur while the *stposc* signal is still active. This sequence should not depend on the presence of Q clocks.

**2:** The EUSART remains in Idle while the WUE bit is set.

## 17.2 HLVD Setup

The following steps are needed to set up the HLVD module:

1. Disable the module by clearing the HLVDEN bit (HLVDCON<4>).
2. Write the value to the HLVDL3:HLVDL0 bits that selects the desired HLVD trip point.
3. Set the VDIRMAG bit to detect high voltage (VDIRMAG = 1) or low voltage (VDIRMAG = 0).
4. Enable the HLVD module by setting the HLVDEN bit.
5. Clear the HLVD Interrupt Flag, HLVDIF (PIR2<2>), which may have been set from a previous interrupt.
6. Enable the HLVD interrupt, if interrupts are desired, by setting the HLVDIE and GIE/GIEH bits (PIE2<2> and INTCON<7>). An interrupt will not be generated until the IRVST bit is set.

## 17.3 Current Consumption

When the module is enabled, the HLVD comparator and voltage divider are enabled and will consume static current. The total current consumption, when enabled, is specified in electrical specification parameter D022 (**Section 270 "DC Characteristics"**).

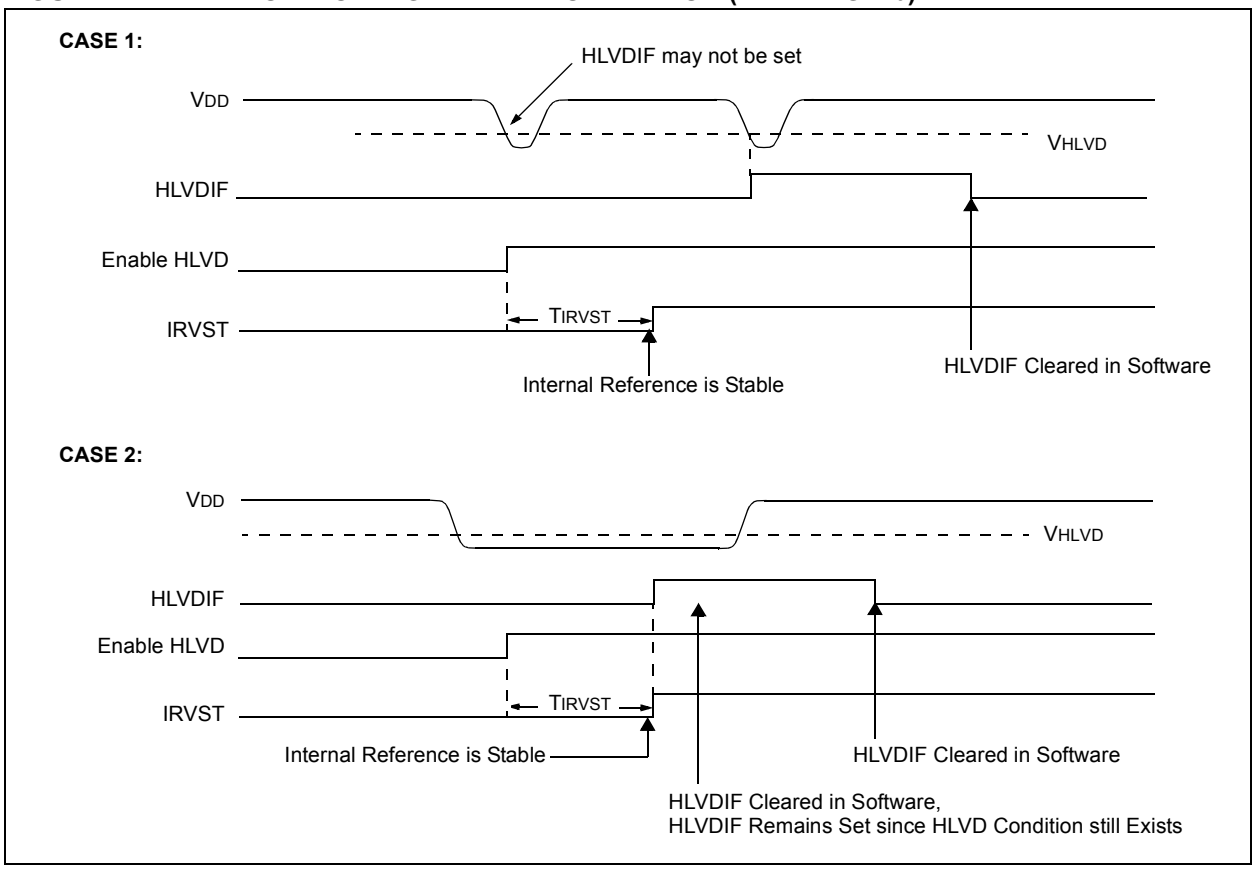
Depending on the application, the HLVD module does not need to be operating constantly. To decrease the current requirements, the HLVD circuitry may only need to be enabled for short periods where the voltage is checked. After doing the check, the HLVD module may be disabled.

## 17.4 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in electrical specification parameter D420 (see Table 21-4 in **Section 21.0 "Electrical Characteristics"**), may be used by other internal circuitry, such as the Programmable Brown-out Reset. If the HLVD or other circuits using the voltage reference are disabled to lower the device's current consumption, the reference voltage circuit will require time to become stable before a low or high-voltage condition can be reliably detected. This start-up time, TIRVST, is an interval that is independent of device clock speed. It is specified in electrical specification parameter 36 (Table 21-10).

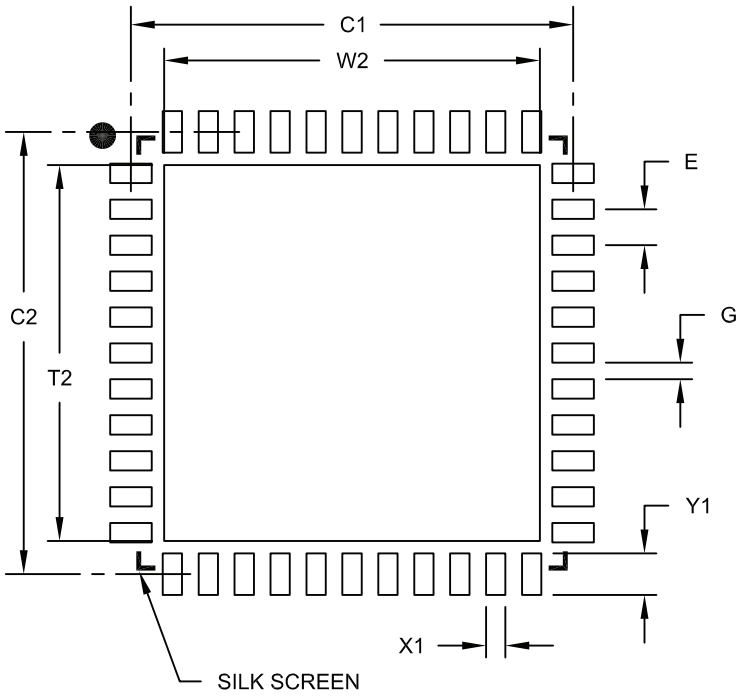
The HLVD interrupt flag is not enabled until TIRVST has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval. Refer to Figure 17-2 or Figure 17-3.

**FIGURE 17-2: LOW-VOLTAGE DETECT OPERATION (VDIRMAG = 0)**



## 44-Lead Plastic Quad Flat, No Lead Package (ML) – 8x8 mm Body [QFN]

**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 | | 0.65 BSC | |
| Optional Center Pad Width | W2 | | | 6.80 |
| Optional Center Pad Length | T2 | | | 6.80 |
| Contact Pad Spacing | C1 | | 8.00 | |
| Contact Pad Spacing | C2 | | 8.00 | |
| Contact Pad Width (X44) | X1 | | | 0.35 |
| Contact Pad Length (X44) | Y1 | | | 0.80 |
| Distance Between Pads | G | 0.25 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

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

Microchip Technology Drawing No. C04-2103A

## T