



Welcome to [E-XFL.COM](https://www.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	8-Bit
Speed	40MHz
Connectivity	I ² C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	23
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1.5K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 5x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf252-i-so

PIC18FXX2

2.7 Effects of SLEEP Mode on the On-Chip Oscillator

When the device executes a `SLEEP` instruction, the on-chip clocks and oscillator are turned off and the device is held at the beginning of an instruction cycle (Q1 state). With the oscillator off, the OSC1 and OSC2 signals will stop oscillating. Since all the transistor

switching currents have been removed, SLEEP mode achieves the lowest current consumption of the device (only leakage currents). Enabling any on-chip feature that will operate during SLEEP will increase the current consumed during SLEEP. The user can wake from SLEEP through external RESET, Watchdog Timer Reset, or through an interrupt.

TABLE 2-3: OSC1 AND OSC2 PIN STATES IN SLEEP MODE

OSC Mode	OSC1 Pin	OSC2 Pin
RC	Floating, external resistor should pull high	At logic low
RCIO	Floating, external resistor should pull high	Configured as PORTA, bit 6
ECIO	Floating	Configured as PORTA, bit 6
EC	Floating	At logic low
LP, XT, and HS	Feedback inverter disabled, at quiescent voltage level	Feedback inverter disabled, at quiescent voltage level

Note: See Table 3-1, in the “Reset” section, for time-outs due to SLEEP and MCLR Reset.

2.8 Power-up Delays

Power up delays are controlled by two timers, so that no external RESET circuitry is required for most applications. The delays ensure that the device is kept in RESET, until the device power supply and clock are stable. For additional information on RESET operation, see Section 3.0.

The first timer is the Power-up Timer (PWRT), which optionally provides a fixed delay of 72 ms (nominal) on power-up only (POR and BOR). The second timer is the Oscillator Start-up Timer (OST), intended to keep the chip in RESET until the crystal oscillator is stable.

With the PLL enabled (HS/PLL Oscillator mode), the time-out sequence following a Power-on Reset is different from other Oscillator modes. The time-out sequence is as follows: First, the PWRT time-out is invoked after a POR time delay has expired. Then, the Oscillator Start-up Timer (OST) is invoked. However, this is still not a sufficient amount of time to allow the PLL to lock at high frequencies. The PWRT timer is used to provide an additional fixed 2 ms (nominal) time-out to allow the PLL ample time to lock to the incoming clock frequency.

PIC18FXX2

FIGURE 3-3: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ TIED TO V_{DD})

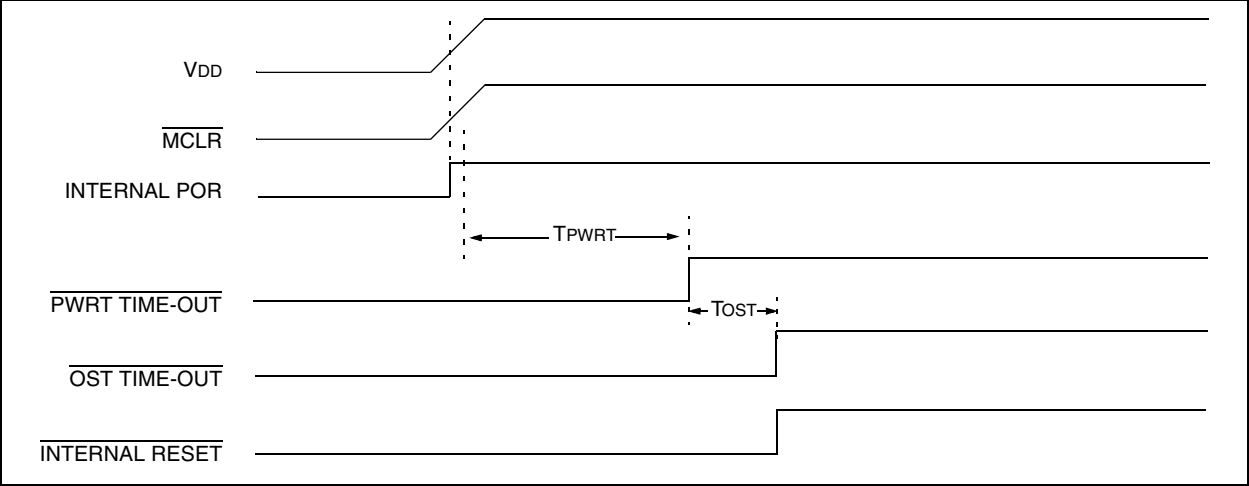


FIGURE 3-4: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO V_{DD}): CASE 1

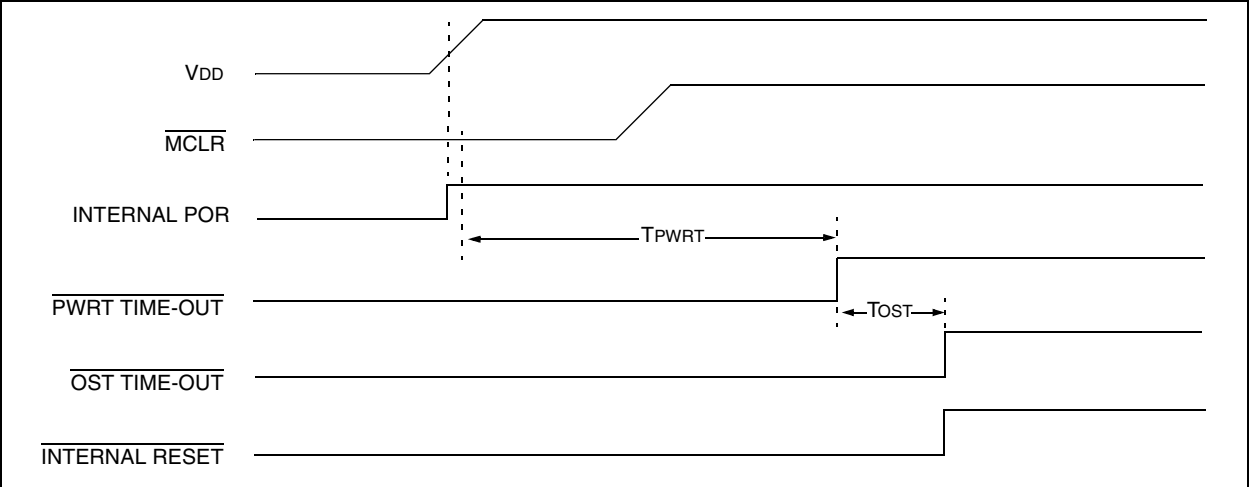
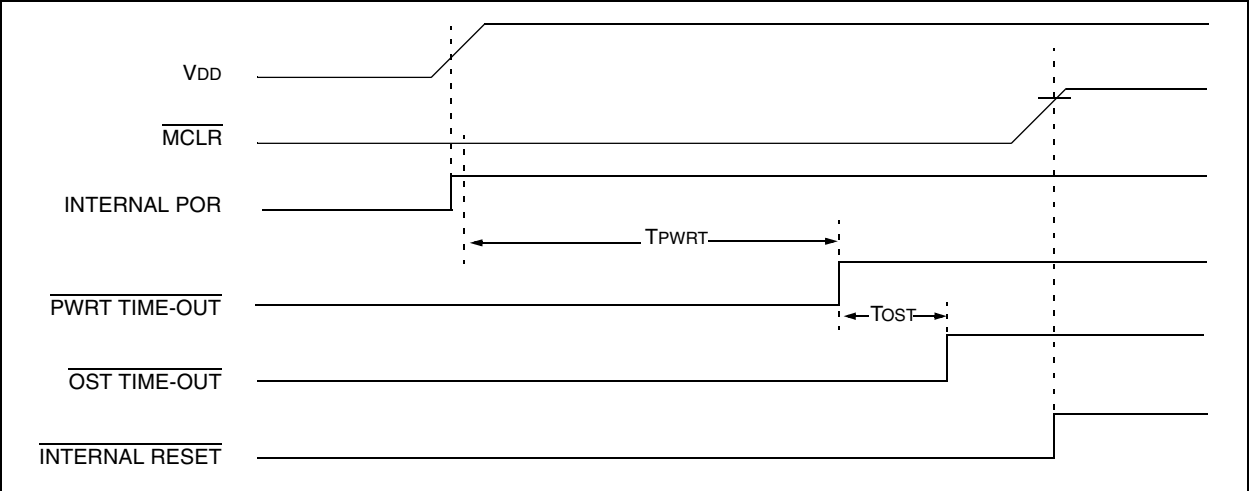


FIGURE 3-5: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO V_{DD}): CASE 2



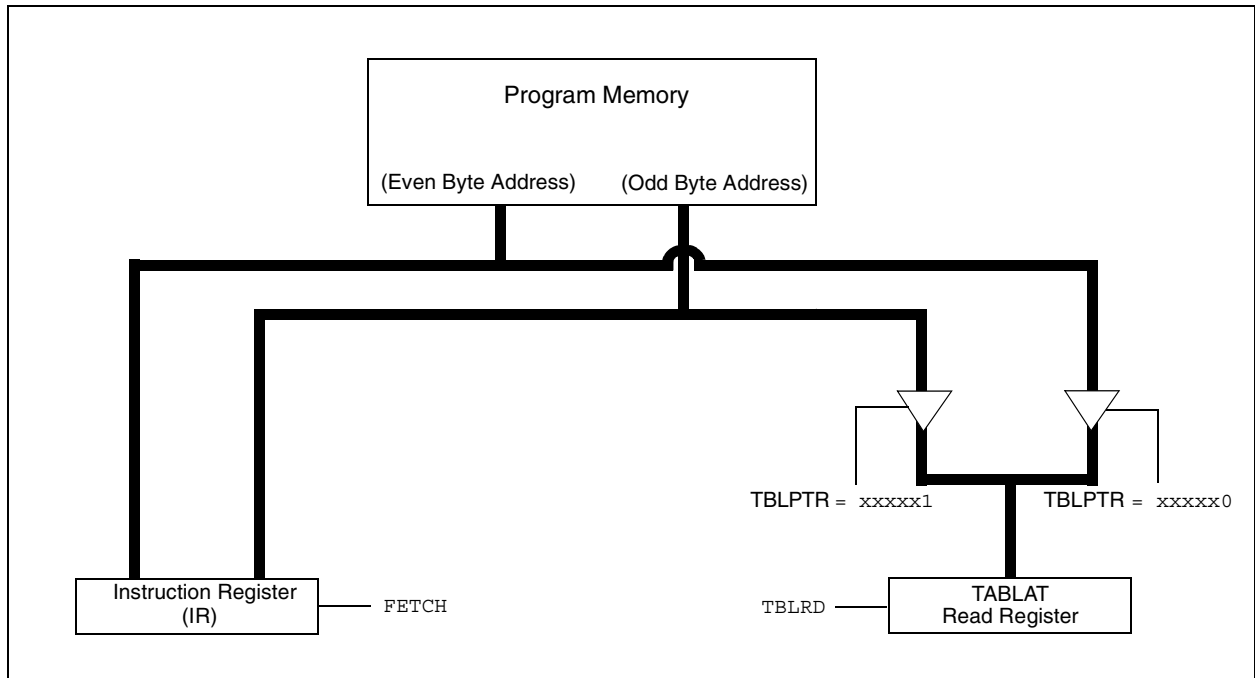
5.3 Reading the FLASH Program Memory

The `TBLRD` instruction is used to retrieve data from program memory and place 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 5-4 shows the interface between the internal program memory and the `TABLAT`.

FIGURE 5-4: READS FROM FLASH PROGRAM MEMORY



EXAMPLE 5-1: READING A FLASH PROGRAM MEMORY WORD

```

MOV LW CODE_ADDR_UPPER      ; Load TBLPTR with the base
MOV WF TBLPTRU              ; address of the word
MOV LW CODE_ADDR_HIGH
MOV WF TBLPTRH
MOV LW CODE_ADDR_LOW
MOV WF TBLPTRL

READ_WORD
TBLRD*+                    ; read into TABLAT and increment
MOV F TABLAT, W            ; get data
MOV WF WORD_EVEN
TBLRD*+                    ; read into TABLAT and increment
MOV F TABLAT, W            ; get data
MOV WF WORD_ODD
    
```

6.0 DATA EEPROM MEMORY

The Data EEPROM is readable and writable during normal operation over the entire V_{DD} range. The data memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers (SFR).

There are four SFRs used to read and write the program and data EEPROM memory. These registers are:

- EECON1
- EECON2
- EEDATA
- EEADR

The EEPROM data memory allows byte read and write. When interfacing to the data memory block, EEDATA holds the 8-bit data for read/write and EEADR holds the address of the EEPROM location being accessed. These devices have 256 bytes of data EEPROM with an address range from 0h to FFh.

The EEPROM data memory is rated for high erase/write cycles. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an on-chip timer. The write time will vary with voltage and temperature, as well as from chip to chip. Please refer to parameter D122 (Electrical Characteristics, Section 22.0) for exact limits.

6.1 EEADR

The address register can address up to a maximum of 256 bytes of data EEPROM.

6.2 EECON1 and EECON2 Registers

EECON1 is the control register for EEPROM memory accesses.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the EEPROM write sequence.

Control bits RD and WR initiate read and write operations, respectively. These bits cannot be cleared, only set, in software. They are cleared in hardware at the completion of the read or write operation. The inability to clear the WR bit in software prevents the accidental or premature termination of a write operation.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a MCLR Reset, or a WDT Time-out Reset during normal operation. In these situations, the user can check the WRERR bit and rewrite the location. It is necessary to reload the data and address registers (EEDATA and EEADR), due to the RESET condition forcing the contents of the registers to zero.

Note: Interrupt flag bit, EEIF in the PIR2 register, is set when write is complete. It must be cleared in software.
--

6.3 Reading the Data EEPROM Memory

To read a data memory location, the user must write the address to the EEADR register, clear the EEPGD control bit (EECON1<7>), clear the CFGS control bit

(EECON1<6>), and then set control bit RD (EECON1<0>). The data is available for the very next instruction cycle; therefore, the EEDATA register can be read by the next instruction. EEDATA will hold this value until another read operation, or until it is written to by the user (during a write operation).

EXAMPLE 6-1: DATA EEPROM READ

```
MOVLW DATA_EE_ADDR ;
MOVWF EEADR          ; Data Memory Address to read
BCF    EECON1, EEPGD ; Point to DATA memory
BCF    EECON1, CFGS  ; Access program FLASH or Data EEPROM memory
BSF    EECON1, RD     ; EEPROM Read
MOVF   EEDATA, W      ; W = EEDATA
```

6.4 Writing to the Data EEPROM Memory

To write an EEPROM data location, the address must first be written to the EEADR register and the data written to the EEDATA register. Then the sequence in Example 6-2 must be followed to initiate the write cycle.

The write will not initiate if the above sequence is not exactly followed (write 55h to EECON2, write AAh to EECON2, then set WR bit) for each byte. It is strongly recommended that interrupts be disabled during this code segment.

Additionally, the WREN bit in EECON1 must be set to enable writes. This mechanism prevents accidental writes to data EEPROM due to unexpected code exe-

cution (i.e., runaway programs). The WREN bit should be kept clear at all times, except when updating the EEPROM. The WREN bit is not cleared by hardware.

After a write sequence has been initiated, EECON1, EEADR and EEDATA cannot be modified. The WR bit will be inhibited from being set unless the WREN bit is set. The WREN bit must be set on a previous instruction. Both WR and WREN cannot be set with the same instruction.

At the completion of the write cycle, the WR bit is cleared in hardware and the EEPROM Write Complete Interrupt Flag bit (EEIF) is set. The user may either enable this interrupt, or poll this bit. EEIF must be cleared by software.

EXAMPLE 6-2: DATA EEPROM WRITE

```
MOVLW DATA_EE_ADDR ;
MOVWF EEADR          ; Data Memory Address to read
MOVLW DATA_EE_DATA ;
MOVWF EEDATA         ; Data Memory Value to write
BCF    EECON1, EEPGD ; Point to DATA memory
BCF    EECON1, CFGS  ; Access program FLASH or Data EEPROM memory
BSF    EECON1, WREN   ; Enable writes

Required Sequence
BCF    INTCON, GIE    ; Disable interrupts
MOVLW  55h            ;
MOVWF  EECON2         ; Write 55h
MOVLW  AAh            ;
MOVWF  EECON2         ; Write AAh
BSF    EECON1, WR     ; Set WR bit to begin write
BSF    INTCON, GIE    ; Enable interrupts

.                    ; user code execution
.
.
BCF    EECON1, WREN   ; Disable writes on write complete (EEIF set)
```

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

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE

bit 7

bit 0

- bit 7-5 **Unimplemented:** Read as '0'
- bit 4 **EEIE:** Data EEPROM/FLASH Write Operation Interrupt Enable bit
 1 = Enabled
 0 = Disabled
- bit 3 **BCLIE:** Bus Collision Interrupt Enable bit
 1 = Enabled
 0 = Disabled
- bit 2 **LVDIE:** Low Voltage Detect Interrupt Enable bit
 1 = Enabled
 0 = Disabled
- bit 1 **TMR3IE:** TMR3 Overflow Interrupt Enable bit
 1 = Enables the TMR3 overflow interrupt
 0 = Disables the TMR3 overflow interrupt
- bit 0 **CCP2IE:** CCP2 Interrupt Enable bit
 1 = Enables the CCP2 interrupt
 0 = Disables the CCP2 interrupt

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

PIC18FXX2

NOTES:

10.1 Timer0 Operation

Timer0 can operate as a timer or as a counter.

Timer mode is selected by clearing the T0CS bit. In Timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0L register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0L register.

Counter mode is selected by setting the T0CS bit. In Counter mode, Timer0 will increment, either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit (T0SE). Clearing the T0SE bit selects the rising edge. Restrictions on the external clock input are discussed below.

When an external clock input is used for Timer0, it must meet certain requirements. The requirements ensure the external clock can be synchronized with the internal phase clock (TOSC). Also, there is a delay in the actual incrementing of Timer0 after synchronization.

10.2 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not readable or writable.

The PSA and T0PS2:T0PS0 bits determine the prescaler assignment and prescale ratio.

Clearing bit PSA will assign the prescaler to the Timer0 module. When the prescaler is assigned to the Timer0 module, prescale values of 1:2, 1:4,..., 1:256 are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0L register (e.g., `CLRF TMR0`, `MOVWF TMR0`, `BSF TMR0`, `x....etc.`) will clear the prescaler count.

Note: Writing to TMR0L when the prescaler is assigned to Timer0 will clear the prescaler count, but will not change the prescaler assignment.

10.2.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control, (i.e., it can be changed “on-the-fly” during program execution).

10.3 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF bit. The interrupt can be masked by clearing the TMR0IE bit. The TMR0IE bit must be cleared in software by the Timer0 module Interrupt Service Routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from SLEEP, since the timer is shut-off during SLEEP.

10.4 16-Bit Mode Timer Reads and Writes

TMR0H is not the high byte of the timer/counter in 16-bit mode, but is actually a buffered version of the high byte of Timer0 (refer to Figure 10-2). The high byte of the Timer0 counter/timer is not directly readable nor writable. TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16-bits of Timer0 without having to verify that the read of the high and low byte were valid due to a rollover between successive reads of the high and low byte.

A write to the high byte of Timer0 must also take place through the TMR0H buffer register. Timer0 high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16-bits of Timer0 to be updated at once.

TABLE 10-1: REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
TMR0L	Timer0 Module Low Byte Register								xxxx xxxx	uuuu uuuu
TMR0H	Timer0 Module High Byte Register								0000 0000	0000 0000
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	1111 1111
TRISA	PORTA Data Direction Register								-111 1111	-111 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

PIC18FXX2

13.1 Timer3 Operation

Timer3 can operate in one of these modes:

- As a timer
- As a synchronous counter
- As an asynchronous counter

The Operating mode is determined by the clock select bit, TMR3CS (T3CON<1>).

When TMR3CS = 0, Timer3 increments every instruction cycle. When TMR3CS = 1, Timer3 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

When the Timer1 oscillator is enabled (T1OSCEN is set), the RC1/T1OSI and RC0/T1OSO/T1CKI pins become inputs. That is, the TRISC<1:0> value is ignored, and the pins are read as '0'.

Timer3 also has an internal “RESET input”. This RESET can be generated by the CCP module (Section 14.0).

FIGURE 13-1: TIMER3 BLOCK DIAGRAM

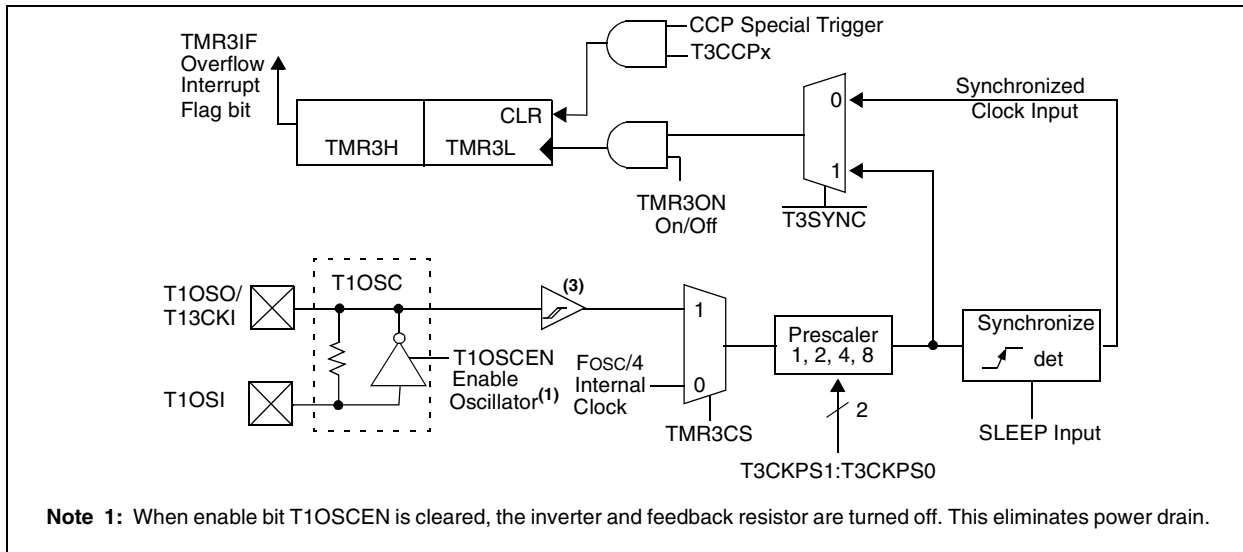
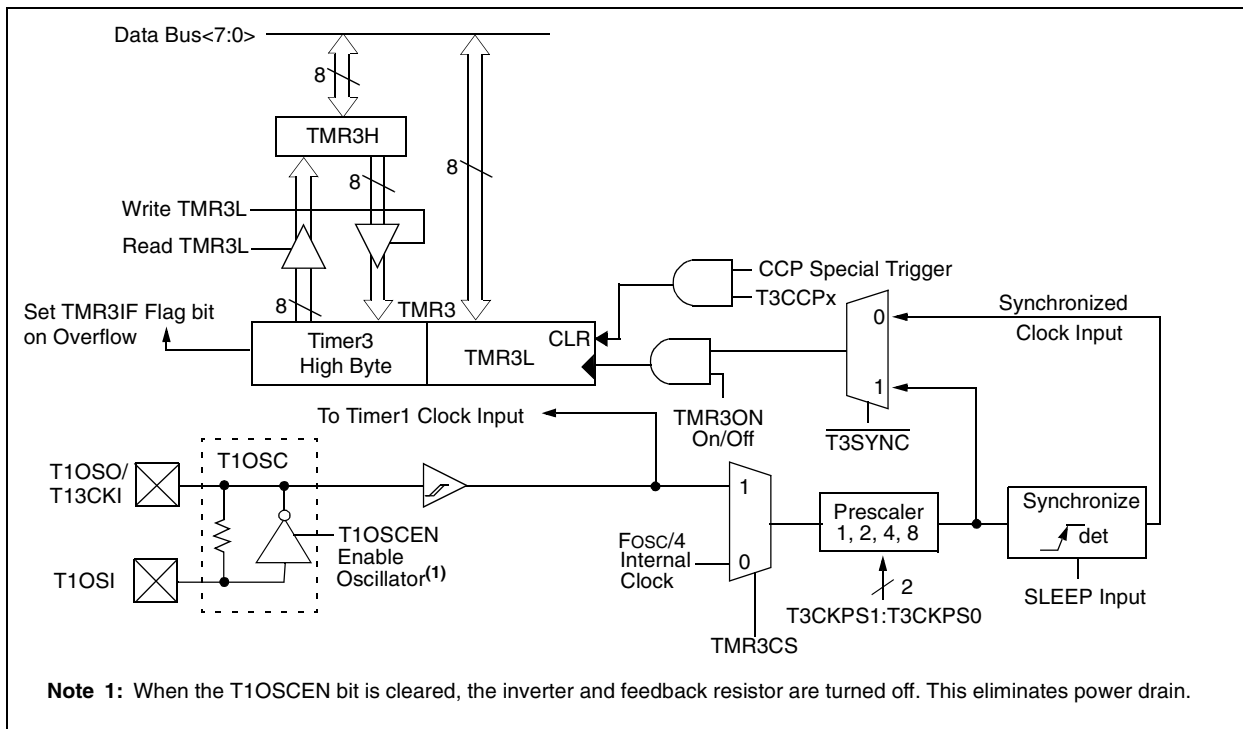


FIGURE 13-2: TIMER3 BLOCK DIAGRAM CONFIGURED IN 16-BIT READ/WRITE MODE



PIC18FXX2

NOTES:

PIC18FXX2

15.3.5 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave (Processor 2, Figure 15-2) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a "Line Activity Monitor" mode.

The clock polarity is selected by appropriately programming the CKP bit (SSPCON1<4>). This then, would give waveforms for SPI communication as shown in

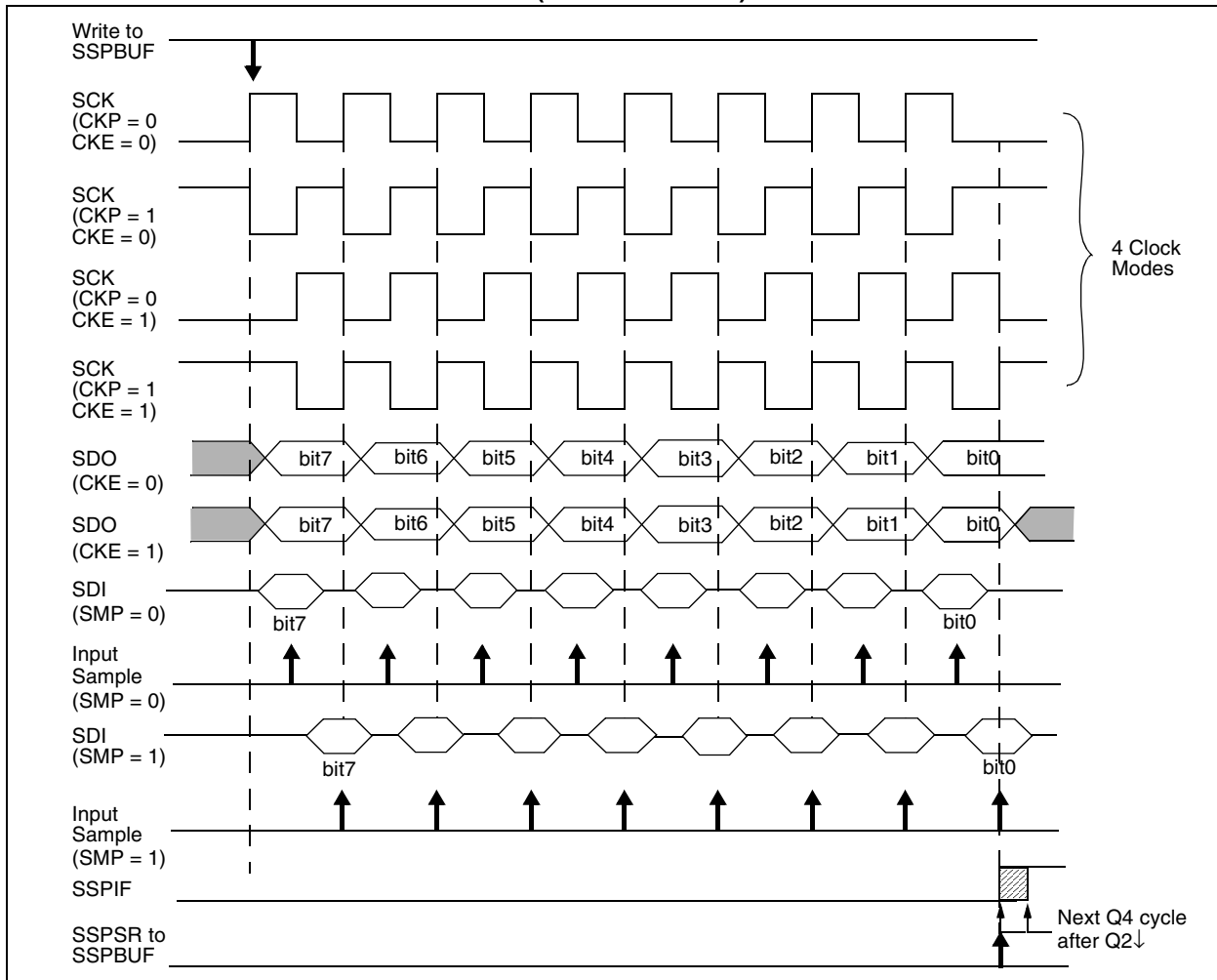
Figure 15-3, Figure 15-5, and Figure 15-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- $F_{osc}/4$ (or T_{CY})
- $F_{osc}/16$ (or $4 \cdot T_{CY}$)
- $F_{osc}/64$ (or $16 \cdot T_{CY}$)
- $\text{Timer2 output}/2$

This allows a maximum data rate (at 40 MHz) of 10.00 Mbps.

Figure 15-3 shows the waveforms for Master mode. When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPBUF is loaded with the received data is shown.

FIGURE 15-3: SPI MODE WAVEFORM (MASTER MODE)



15.4.8 I²C MASTER MODE START CONDITION TIMING

To initiate a START condition, the user sets the START condition enable bit, SEN (SSPCON2<0>). If the SDA and SCL pins are sampled high, the baud rate generator is reloaded with the contents of SSPADD<6:0> and starts its count. If SCL and SDA are both sampled high when the baud rate generator times out (TBRG), the SDA pin is driven low. The action of the SDA being driven low, while SCL is high, is the START condition and causes the S bit (SSPSTAT<3>) to be set. Following this, the baud rate generator is reloaded with the contents of SSPADD<6:0> and resumes its count. When the baud rate generator times out (TBRG), the SEN bit (SSPCON2<0>) will be automatically cleared by hardware, the baud rate generator is suspended, leaving the SDA line held low and the START condition is complete.

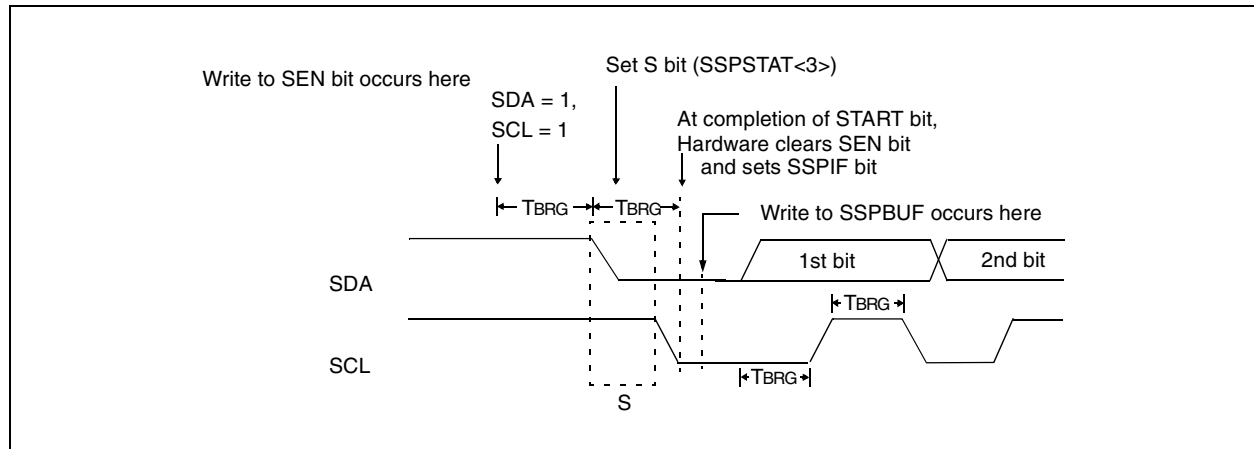
Note: If at the beginning of the START condition, the SDA and SCL pins are already sampled low, or if during the START condition the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLIF is set, the START condition is aborted, and the I²C module is reset into its IDLE state.

15.4.8.1 WCOL Status Flag

If the user writes the SSPBUF when a START sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

Note: Because queueing of events is not allowed, writing to the lower 5 bits of SSPCON2 is disabled until the START condition is complete.

FIGURE 15-19: FIRST START BIT TIMING



PIC18FXX2

16.2 USART Asynchronous Mode

In this mode, the USART uses standard non-return-to-zero (NRZ) format (one START bit, eight or nine data bits and one STOP bit). The most common data format is 8-bits. An on-chip dedicated 8-bit baud rate generator can be used to derive standard baud rate frequencies from the oscillator. The USART transmits and receives the LSb first. The USART's transmitter and receiver are functionally independent, but use the same data format and baud rate. The baud rate generator produces a clock, either x16 or x64 of the bit shift rate, depending on bit BRGH (TXSTA<2>). Parity is not supported by the hardware, but can be implemented in software (and stored as the ninth data bit). Asynchronous mode is stopped during SLEEP.

Asynchronous mode is selected by clearing bit SYNC (TXSTA<4>).

The USART Asynchronous module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver

16.2.1 USART ASYNCHRONOUS TRANSMITTER

The USART transmitter block diagram is shown in Figure 16-1. The heart of the transmitter is the Transmit (serial) Shift Register (TSR). The shift register obtains its data from the read/write transmit buffer, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the STOP bit has been transmitted from the previous load. As soon as the STOP bit is transmitted, the TSR is loaded with new data from the TXREG register (if available). Once the TXREG register transfers the data to the TSR register (occurs in one Tcy), the TXREG register is empty and

flag bit TXIF (PIR1<4>) is set. This interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE1<4>). Flag bit TXIF will be set, regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicated the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. Status bit TRMT is a read-only bit, which is set when the TSR register is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty.

Note 1: The TSR register is not mapped in data memory, so it is not available to the user.

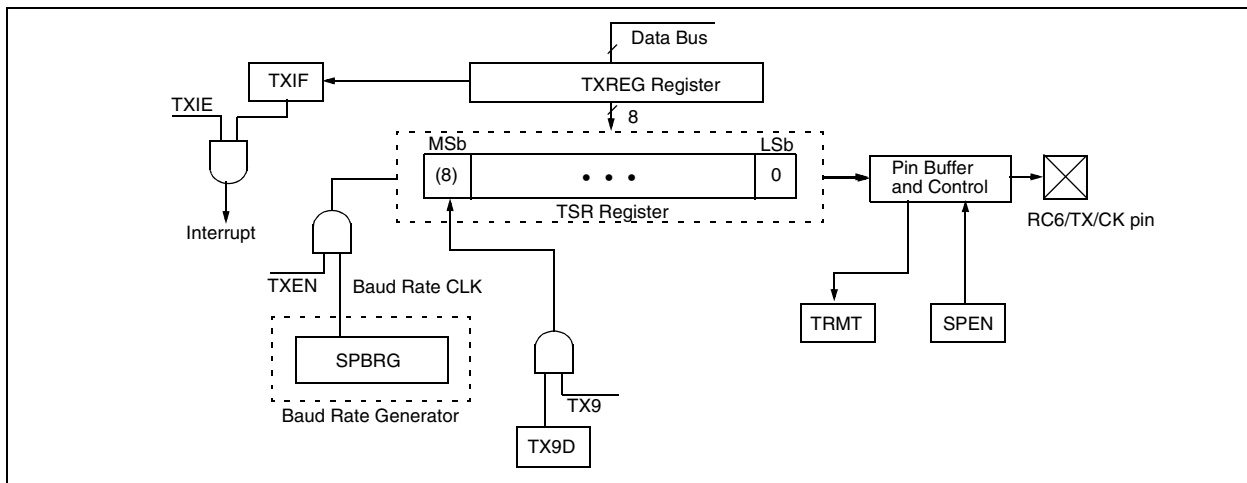
2: Flag bit TXIF is set when enable bit TXEN is set.

To set up an asynchronous transmission:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH (Section 16.1).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set transmit bit TX9. Can be used as address/data bit.
5. Enable the transmission by setting bit TXEN, which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).

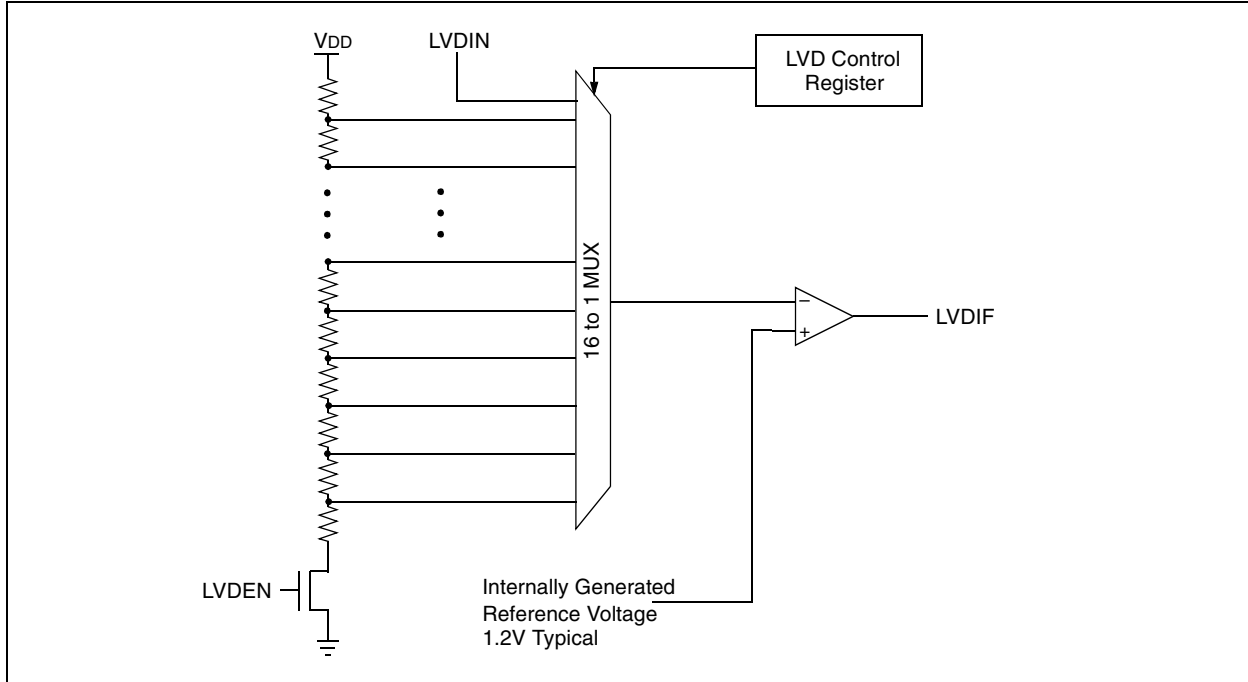
Note: TXIF is not cleared immediately upon loading data into the transmit buffer TXREG. The flag bit becomes valid in the second instruction cycle following the load instruction.

FIGURE 16-1: USART TRANSMIT BLOCK DIAGRAM



PIC18FXX2

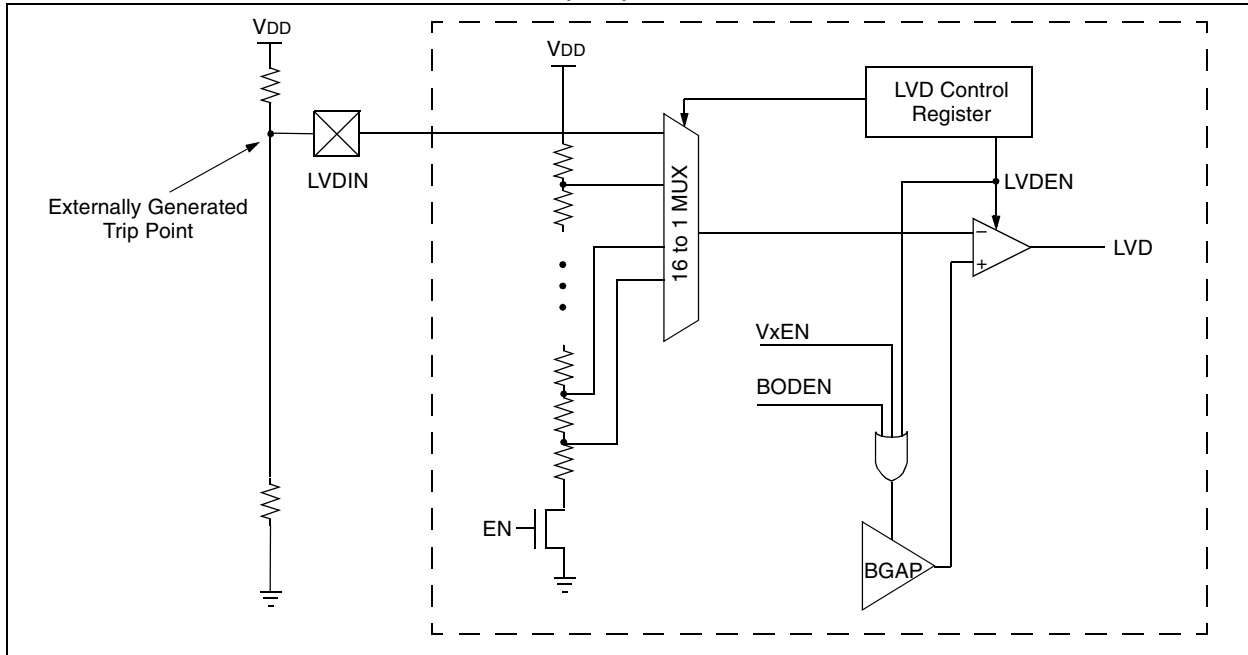
FIGURE 18-2: LOW VOLTAGE DETECT (LVD) BLOCK DIAGRAM



The LVD module has an additional feature that allows the user to supply the trip voltage to the module from an external source. This mode is enabled when bits LV DL3:LV DL0 are set to 1111. In this state, the comparator input is multiplexed from the external input pin,

LVDIN (Figure 18-3). This gives users flexibility, because it allows them to configure the Low Voltage Detect interrupt to occur at any voltage in the valid operating range.

FIGURE 18-3: LOW VOLTAGE DETECT (LVD) WITH EXTERNAL INPUT BLOCK DIAGRAM



18.1 Control Register

The Low Voltage Detect Control register controls the operation of the Low Voltage Detect circuitry.

REGISTER 18-1: LVDCON REGISTER

U-0	U-0	R-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
—	—	IRVST	LV DEN	LV DL3	LV DL2	LV DL1	LV DL0
bit 7		bit 0					

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

bit 5 **IRVST:** Internal Reference Voltage Stable Flag bit
 1 = Indicates that the Low Voltage Detect logic will generate the interrupt flag at the specified voltage range
 0 = Indicates that the Low Voltage Detect logic will not generate the interrupt flag at the specified voltage range and the LVD interrupt should not be enabled

bit 4 **LV DEN:** Low Voltage Detect Power Enable bit
 1 = Enables LVD, powers up LVD circuit
 0 = Disables LVD, powers down LVD circuit

bit 3-0 **LV DL3:LV DL0:** Low Voltage Detection Limit bits
 1111 = External analog input is used (input comes from the LVDIN pin)
 1110 = 4.5V - 4.77V
 1101 = 4.2V - 4.45V
 1100 = 4.0V - 4.24V
 1011 = 3.8V - 4.03V
 1010 = 3.6V - 3.82V
 1001 = 3.5V - 3.71V
 1000 = 3.3V - 3.50V
 0111 = 3.0V - 3.18V
 0110 = 2.8V - 2.97V
 0101 = 2.7V - 2.86V
 0100 = 2.5V - 2.65V
 0011 = 2.4V - 2.54V
 0010 = 2.2V - 2.33V
 0001 = 2.0V - 2.12V
 0000 = Reserved

Note: LV DL3:LV DL0 modes which result in a trip point below the valid operating voltage of the device are not tested.

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

19.3 Power-down Mode (SLEEP)

Power-down mode is entered by executing a `SLEEP` instruction.

If enabled, the Watchdog Timer will be cleared, but keeps running, the $\overline{PD}$ bit (`RCON<3>`) is cleared, the $\overline{TO}$ (`RCON<4>`) bit is set, and the oscillator driver is turned off. The I/O ports maintain the status they had before the `SLEEP` instruction was executed (driving high, low or hi-impedance).

For lowest current consumption in this mode, place all I/O pins at either V_{DD} or V_{SS} , ensure no external circuitry is drawing current from the I/O pin, power-down the A/D and disable external clocks. Pull all I/O pins that are hi-impedance inputs, high or low externally, to avoid switching currents caused by floating inputs. The `T0CKI` input should also be at V_{DD} or V_{SS} for lowest current consumption. The contribution from on-chip pull-ups on `PORTB` should be considered.

The $\overline{MCLR}$ pin must be at a logic high level (V_{IHMC}).

19.3.1 WAKE-UP FROM SLEEP

The device can wake-up from `SLEEP` through one of the following events:

1. External `RESET` input on $\overline{MCLR}$ pin.
2. Watchdog Timer Wake-up (if `WDT` was enabled).
3. Interrupt from `INT` pin, `RB` port change or a Peripheral Interrupt.

The following peripheral interrupts can wake the device from `SLEEP`:

1. PSP read or write.
2. `TMR1` interrupt. Timer1 must be operating as an asynchronous counter.
3. `TMR3` interrupt. Timer3 must be operating as an asynchronous counter.
4. CCP Capture mode interrupt.
5. Special event trigger (Timer1 in Asynchronous mode using an external clock).
6. MSSP (`START/STOP`) bit detect interrupt.
7. MSSP transmit or receive in Slave mode (`SPI/I2C`).
8. USART RX or TX (Synchronous Slave mode).
9. A/D conversion (when A/D clock source is `RC`).
10. EEPROM write operation complete.
11. LVD interrupt.

Other peripherals cannot generate interrupts, since during `SLEEP`, no on-chip clocks are present.

External $\overline{MCLR}$ Reset will cause a device `RESET`. All other events are considered a continuation of program execution and will cause a “wake-up”. The $\overline{TO}$ and $\overline{PD}$ bits in the `RCON` register can be used to determine the cause of the device `RESET`. The $\overline{PD}$ bit, which is set on power-up, is cleared when `SLEEP` is invoked. The $\overline{TO}$ bit is cleared, if a `WDT` time-out occurred (and caused wake-up).

When the `SLEEP` instruction is being executed, the next instruction (`PC + 2`) is pre-fetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be set (enabled). Wake-up is regardless of the state of the `GIE` bit. If the `GIE` bit is clear (disabled), the device continues execution at the instruction after the `SLEEP` instruction. If the `GIE` bit is set (enabled), the device executes the instruction after the `SLEEP` instruction and then branches to the interrupt address. In cases where the execution of the instruction following `SLEEP` is not desirable, the user should have a `NOP` after the `SLEEP` instruction.

19.3.2 WAKE-UP USING INTERRUPTS

When global interrupts are disabled (`GIE` cleared) and any interrupt source has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If an interrupt condition (interrupt flag bit and interrupt enable bits are set) occurs **before** the execution of a `SLEEP` instruction, the `SLEEP` instruction will complete as a `NOP`. Therefore, the `WDT` and `WDT` postscaler will not be cleared, the $\overline{TO}$ bit will not be set and $\overline{PD}$ bits will not be cleared.
- If the interrupt condition occurs **during or after** the execution of a `SLEEP` instruction, the device will immediately wake-up from `SLEEP`. The `SLEEP` instruction will be completely executed before the wake-up. Therefore, the `WDT` and `WDT` postscaler will be cleared, the $\overline{TO}$ bit will be set and the $\overline{PD}$ bit will be cleared.

Even if the flag bits were checked before executing a `SLEEP` instruction, it may be possible for flag bits to become set before the `SLEEP` instruction completes. To determine whether a `SLEEP` instruction executed, test the $\overline{PD}$ bit. If the $\overline{PD}$ bit is set, the `SLEEP` instruction was executed as a `NOP`.

To ensure that the `WDT` is cleared, a `CLRWDT` instruction should be executed before a `SLEEP` instruction.

CLRF		Clear f						
Syntax:	[<i>label</i>] CLRF f [,a]							
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]							
Operation:	000h → f 1 → Z							
Status Affected:	Z							
Encoding:	<table border="1"><tr><td>0110</td><td>101a</td><td>ffff</td><td>ffff</td></tr></table>				0110	101a	ffff	ffff
0110	101a	ffff	ffff					
Description:	Clears the contents of the specified register. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).							
Words:	1							
Cycles:	1							
Q Cycle Activity:								
	Q1	Q2	Q3	Q4				
	Decode	Read register 'f'	Process Data	Write register 'f'				

Example: CLRF FLAG_REG, 1

Before Instruction
FLAG_REG = 0x5A

After Instruction
FLAG_REG = 0x00

CLRWDT		Clear Watchdog Timer							
Syntax:	[<i>label</i>] CLRWDT								
Operands:	None								
Operation:	000h → WDT, 000h → WDT postscaler, 1 → $\overline{TO}$, 1 → $\overline{PD}$								
Status Affected:	$\overline{TO}$, $\overline{PD}$								
Encoding:	<table border="1"><tr><td>0000</td><td>0000</td><td>0000</td><td>0100</td></tr></table>				0000	0000	0000	0100	
0000	0000	0000	0100						
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the <u>postscaler</u> of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	Q1	Q2	Q3	Q4					
	Decode	No operation	Process Data	No operation					

Example: CLRWDT

Before Instruction
WDT Counter = ?

After Instruction
WDT Counter = 0x00
WDT Postscaler = 0
 $\overline{TO}$ = 1
 $\overline{PD}$ = 1

RCALL		Relative Call						
Syntax:	[<i>label</i>] RCALL n							
Operands:	-1024 ≤ n ≤ 1023							
Operation:	(PC) + 2 → TOS, (PC) + 2 + 2n → PC							
Status Affected:	None							
Encoding:	<table><tr><td>1101</td><td>1nnn</td><td>nnnn</td><td>nnnn</td></tr></table>				1101	1nnn	nnnn	nnnn
1101	1nnn	nnnn	nnnn					
Description:	Subroutine call with a jump up to 1K from the current location. First, return address (PC+2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is a two-cycle instruction.							
Words:	1							
Cycles:	2							

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n' Push PC to stack	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: HERE RCALL Jump

Before Instruction

PC = Address (HERE)

After Instruction

PC = Address (Jump)

TOS = Address (HERE+2)

RESET	Reset								
Syntax:	[<i>label</i>] RESET								
Operands:	None								
Operation:	Reset all registers and flags that are affected by a MCLR Reset.								
Status Affected:	All								
Encoding:	<table><tr><td>0000</td><td>0000</td><td>1111</td><td>1111</td></tr></table>	0000	0000	1111	1111				
0000	0000	1111	1111						
Description:	This instruction provides a way to execute a MCLR Reset in software.								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Start reset</td><td>No operation</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Start reset	No operation	No operation
Q1	Q2	Q3	Q4						
Decode	Start reset	No operation	No operation						

Example: RESET

After Instruction

Registers = Reset Value

Flags* = Reset Value

PIC18FXX2

FIGURE 22-20: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

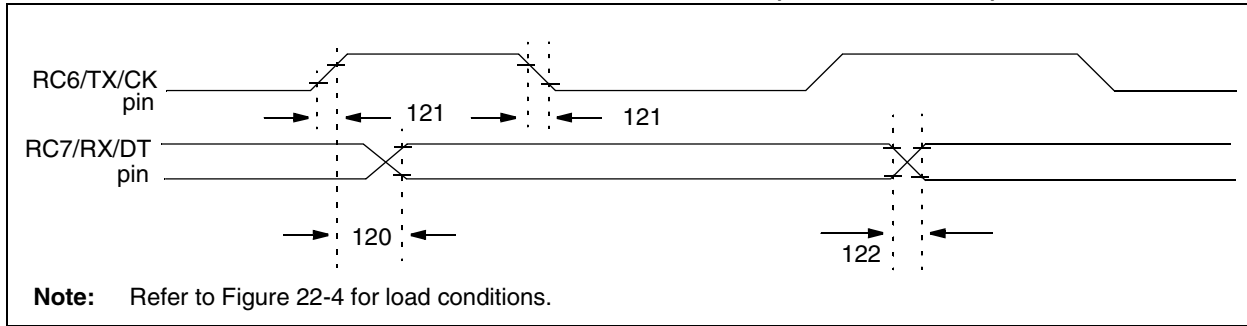


TABLE 22-19: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
120	TckH2dtV	<u>SYNC XMIT (MASTER & SLAVE)</u> Clock high to data out valid				
			PIC18FXXX	—	50	ns
			PIC18LFXXX	—	150	ns VDD = 2V
121	Tckr	Clock out rise time and fall time (Master mode)	PIC18FXXX	—	25	ns
			PIC18LFXXX	—	60	ns VDD = 2V
122	Tdtr	Data out rise time and fall time	PIC18FXXX	—	25	ns
			PIC18LFXXX	—	60	ns VDD = 2V

FIGURE 22-21: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING

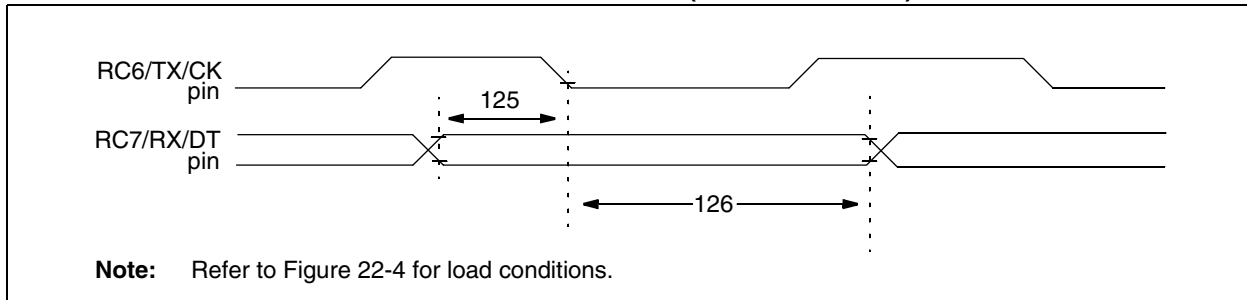


TABLE 22-20: USART SYNCHRONOUS RECEIVE REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
125	TdtV2ckl	<u>SYNC RCV (MASTER & SLAVE)</u> Data hold before CK ↓ (DT hold time)	10	—	ns	
126	TckL2dtl	Data hold after CK ↓ (DT hold time)	PIC18FXXX	15	—	ns
			PIC18LFXXX	20	—	ns VDD = 2V