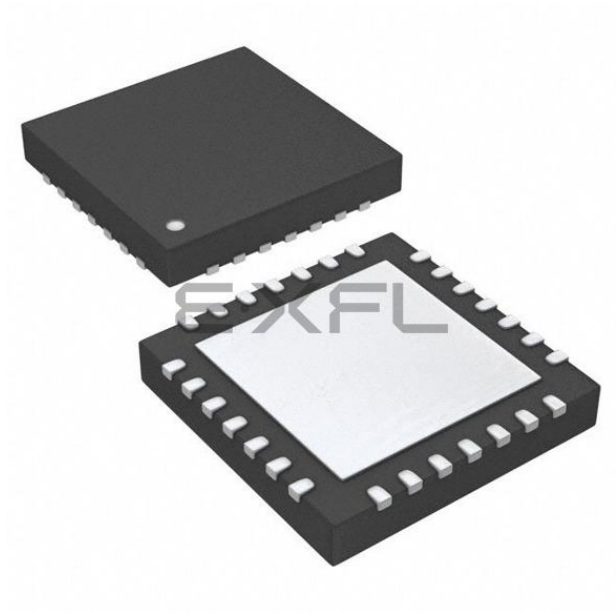


**Welcome to E-XFL.COM**

## 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 | 8-Bit |
| Speed | 32MHz |
| Connectivity | I²C, LINbus, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, LCD, POR, PWM, WDT |
| Number of I/O | 25 |
| Program Memory Size | 7KB (4K x 14) |
| Program Memory Type | FLASH |
| EEPROM Size | 256 x 8 |
| RAM Size | 256 x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 3.6V |
| Data Converters | A/D 11x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 28-UFQFN Exposed Pad |
| Supplier Device Package | 28-UQFN (4x4) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic16lf1933t-i-mv |

**TABLE 3-9: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)**

| Address | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bank 7** | | | | | | | | | | | |
| 380h(2) | INDF0 | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | xxxx xxxx |
| 381h(2) | INDF1 | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | xxxx xxxx |
| 382h(2) | PCL | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 383h(2) | STATUS | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 384h(2) | FSR0L | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 385h(2) | FSR0H | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 386h(2) | FSR1L | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 387h(2) | FSR1H | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 388h(2) | BSR | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 389h(2) | WREG | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 38Ah(1, 2) | PCLATH | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 38Bh(2) | INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |
| 38Ch | — | Unimplemented | | | | | | | | — | — |
| 38Dh | — | Unimplemented | | | | | | | | — | — |
| 38Eh | — | Unimplemented | | | | | | | | — | — |
| 38Fh | — | Unimplemented | | | | | | | | — | — |
| 390h | — | Unimplemented | | | | | | | | — | — |
| 391h | — | Unimplemented | | | | | | | | — | — |
| 392h | — | Unimplemented | | | | | | | | — | — |
| 393h | — | Unimplemented | | | | | | | | — | — |
| 394h | IOCBP | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | IOCBP3 | IOCBP2 | IOCBP1 | IOCBP0 | 0000 0000 | 0000 0000 |
| 395h | IOCBN | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | IOCBN3 | IOCBN2 | IOCBN1 | IOCBN0 | 0000 0000 | 0000 0000 |
| 396h | IOCBF | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | IOCBF3 | IOCBF2 | IOCBF1 | IOCBF0 | 0000 0000 | 0000 0000 |
| 397h | — | Unimplemented | | | | | | | | — | — |
| 398h | — | Unimplemented | | | | | | | | — | — |
| 399h | — | Unimplemented | | | | | | | | — | — |
| 39Ah | — | Unimplemented | | | | | | | | — | — |
| 39Bh | — | Unimplemented | | | | | | | | — | — |
| 39Ch | — | Unimplemented | | | | | | | | — | — |
| 39Dh | — | Unimplemented | | | | | | | | — | — |
| 39Eh | — | Unimplemented | | | | | | | | — | — |
| 39Fh | — | Unimplemented | | | | | | | | — | — |

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

**Note 1:** The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<14:8>, whose contents are transferred to the upper byte of the program counter.

**2:** These registers can be addressed from any bank.

**3:** Unimplemented, read as '1'.

### 7.6.1 PIE1 REGISTER

**REGISTER 7-2: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| TMR1GIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7 **TMR1GIE:** Timer1 Gate Interrupt Enable bit
1 = Enables the Timer1 Gate Acquisition interrupt
0 = Disables the Timer1 Gate Acquisition interrupt

bit 6 **ADIE:** A/D Converter (ADC) Interrupt Enable bit
1 = Enables the ADC interrupt
0 = Disables the ADC interrupt

bit 5 **RCIE:** USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE:** USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE:** Synchronous Serial Port (MSSP) Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

bit 2 **CCP1IE:** CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the Timer2 to PR2 match interrupt
0 = Disables the Timer2 to PR2 match interrupt

bit 0 **TMR1IE:** Timer1 Overflow Interrupt Enable bit
1 = Enables the Timer1 overflow interrupt
0 = Disables the Timer1 overflow interrupt

**Note:** Bit PEIE of the INTCON register must be set to enable any peripheral interrupt.

**TABLE 7-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 82 |
| OPTION_REG | $\overline{\text{WPUEN}}$ | INTEDG | TMROCS | TMROSE | PSA | PS<2:0> | | | 173 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 83 |
| PIE2 | OSFIE | C2IE | C1IE | EEIE | BCLIE | LCDIE | — | CCP2IE | 84 |
| PIE3 | — | CCP5IE | CCP4IE | CCP3IE | TMR6IE | — | TMR4IE | — | 85 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 86 |
| PIR2 | OSFIF | C2IF | C1IF | EEIF | BCLIF | LCDIF | — | CCP2IF | 87 |
| PIR3 | — | CCP5IF | CCP4IF | CCP3IF | TMR6IF | — | TMR4IF | — | 88 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by Interrupts.

## 15.2 ADC Operation

### 15.2.1 STARTING A CONVERSION

To enable the ADC module, the ADON bit of the ADCON0 register must be set to a '1'. Setting the GO/$\overline{\text{DONE}}$ bit of the ADCON0 register to a '1' will start the Analog-to-Digital conversion.

| | |
|---|---|
| **Note:** | The GO/$\overline{\text{DONE}}$ bit should not be set in the same instruction that turns on the ADC. Refer to **Section 15.2.6 "A/D Conversion Procedure"**. |

### 15.2.2 COMPLETION OF A CONVERSION

When the conversion is complete, the ADC module will:

- Clear the GO/$\overline{\text{DONE}}$ bit
- Set the ADIF Interrupt Flag bit
- Update the ADRESH and ADRESL registers with new conversion result

### 15.2.3 TERMINATING A CONVERSION

If a conversion must be terminated before completion, the GO/$\overline{\text{DONE}}$ bit can be cleared in software. The ADRESH and ADRESL registers will be updated with the partially complete Analog-to-Digital conversion sample. Incomplete bits will match the last bit converted.

| | |
|---|---|
| **Note:** | A device Reset forces all registers to their Reset state. Thus, the ADC module is turned off and any pending conversion is terminated. |

### 15.2.4 ADC OPERATION DURING SLEEP

The ADC module can operate during Sleep. This requires the ADC clock source to be set to the FRC option. When the FRC clock source is selected, the ADC waits one additional instruction before starting the conversion. This allows the SLEEP instruction to be executed, which can reduce system noise during the conversion. If the ADC interrupt is enabled, the device will wake-up from Sleep when the conversion completes. If the ADC interrupt is disabled, the ADC module is turned off after the conversion completes, although the ADON bit remains set.

When the ADC clock source is something other than FRC, a SLEEP instruction causes the present conversion to be aborted and the ADC module is turned off, although the ADON bit remains set.

### 15.2.5 SPECIAL EVENT TRIGGER

The Special Event Trigger of the CCPx/ECCPX module allows periodic ADC measurements without software intervention. When this trigger occurs, the GO/$\overline{\text{DONE}}$ bit is set by hardware and the Timer1 counter resets to zero.

**TABLE 15-2: SPECIAL EVENT TRIGGER**

| Device | CCPx/ECCPx |
|---|---|
| PIC16(L)F1933 | CCP5 |

Using the Special Event Trigger does not assure proper ADC timing. It is the user's responsibility to ensure that the ADC timing requirements are met.

Refer to **Section 23.0 "Capture/Compare/PWM Modules"** for more information.

### 21.6.2 TIMER1 GATE SOURCE SELECTION

The Timer1 gate source can be selected from one of four different sources. Source selection is controlled by the T1GSS bits of the T1GCON register. The polarity for each available source is also selectable. Polarity selection is controlled by the T1GPOL bit of the T1GCON register.

**TABLE 21-4: TIMER1 GATE SOURCES**

| T1GSS | Timer1 Gate Source |
|---|---|
| 00 | Timer1 Gate Pin |
| 01 | Overflow of Timer0 (TMR0 increments from FFh to 00h) |
| 10 | Comparator 1 Output sync_C1OUT (optionally Timer1 synchronized output) |
| 11 | Comparator 2 Output sync_C2OUT (optionally Timer1 synchronized output) |

#### 21.6.2.1 T1G Pin Gate Operation

The T1G pin is one source for Timer1 gate control. It can be used to supply an external source to the Timer1 gate circuitry.

#### 21.6.2.2 Timer0 Overflow Gate Operation

When Timer0 increments from FFh to 00h, a low-to-high pulse will automatically be generated and internally supplied to the Timer1 gate circuitry.

#### 21.6.2.3 Comparator C1 Gate Operation

The output resulting from a Comparator 1 operation can be selected as a source for Timer1 gate control. The Comparator 1 output (sync_C1OUT) can be synchronized to the Timer1 clock or left asynchronous. For more information see **Section 18.4.1 "Comparator Output Synchronization"**.

#### 21.6.2.4 Comparator C2 Gate Operation

The output resulting from a Comparator 2 operation can be selected as a source for Timer1 gate control. The Comparator 2 output (sync_C2OUT) can be synchronized to the Timer1 clock or left asynchronous. For more information see **Section 18.4.1 "Comparator Output Synchronization"**.

### 21.6.3 TIMER1 GATE TOGGLE MODE

When Timer1 Gate Toggle mode is enabled, it is possible to measure the full-cycle length of a Timer1 gate signal, as opposed to the duration of a single level pulse.

The Timer1 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See Figure 21-4 for timing details.

Timer1 Gate Toggle mode is enabled by setting the T1GTM bit of the T1GCON register. When the T1GTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

| | |
|---|---|
| **Note:** | Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation. |

### 21.6.4 TIMER1 GATE SINGLE-PULSE MODE

When Timer1 Gate Single-Pulse mode is enabled, it is possible to capture a single-pulse gate event. Timer1 Gate Single-Pulse mode is first enabled by setting the T1GSPM bit in the T1GCON register. Next, the T1GGO/$\overline{\text{DONE}}$ bit in the T1GCON register must be set. The Timer1 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the T1GGO/$\overline{\text{DONE}}$ bit will automatically be cleared. No other gate events will be allowed to increment Timer1 until the T1GGO/$\overline{\text{DONE}}$ bit is once again set in software. See Figure 21-5 for timing details.

If the Single-Pulse Gate mode is disabled by clearing the T1GSPM bit in the T1GCON register, the T1GGO/$\overline{\text{DONE}}$ bit should also be cleared.

Enabling the Toggle mode and the Single-Pulse mode simultaneously will permit both sections to work together. This allows the cycle times on the Timer1 gate source to be measured. See Figure 21-6 for timing details.

### 21.6.5 TIMER1 GATE VALUE STATUS

When Timer1 Gate Value Status is utilized, it is possible to read the most current level of the gate control value. The value is stored in the T1GVAL bit in the T1GCON register. The T1GVAL bit is valid even when the Timer1 gate is not enabled (TMR1GE bit is cleared).

### 21.6.6 TIMER1 GATE EVENT INTERRUPT

When Timer1 Gate Event Interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of T1GVAL occurs, the TMR1GIF flag bit in the PIR1 register will be set. If the TMR1GIE bit in the PIE1 register is set, then an interrupt will be recognized.

The TMR1GIF flag bit operates even when the Timer1 gate is not enabled (TMR1GE bit is cleared).

## 23.3 PWM Overview

Pulse-Width Modulation (PWM) is a scheme that provides power to a load by switching quickly between fully on and fully off states. The PWM signal resembles a square wave where the high portion of the signal is considered the on state and the low portion of the signal is considered the off state. The high portion, also known as the pulse width, can vary in time and is defined in steps. A larger number of steps applied, which lengthens the pulse width, also supplies more power to the load. Lowering the number of steps applied, which shortens the pulse width, supplies less power. The PWM period is defined as the duration of one complete cycle or the total amount of on and off time combined.

PWM resolution defines the maximum number of steps that can be present in a single PWM period. A higher resolution allows for more precise control of the pulse width time and in turn the power that is applied to the load.

The term duty cycle describes the proportion of the on time to the off time and is expressed in percentages, where 0% is fully off and 100% is fully on. A lower duty cycle corresponds to less power applied and a higher duty cycle corresponds to more power applied.

Figure 23-3 shows a typical waveform of the PWM signal.

### 23.3.1 STANDARD PWM OPERATION

The standard PWM function described in this section is available and identical for CCP modules ECCP1, ECCP2, ECCP3, CCP4 and CCP5.

The standard PWM mode generates a Pulse-Width Modulation (PWM) signal on the CCPx pin with up to 10 bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

- PRx registers
- TxCON registers
- CCPRxL registers
- CCPxCON registers

Figure 23-4 shows a simplified block diagram of PWM operation.

| | |
|---|---|
| **Note 1:** | The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin. |
| **2:** | Clearing the CCPxCON register will relinquish control of the CCPx pin. |

**FIGURE 23-3: CCP PWM OUTPUT SIGNAL**

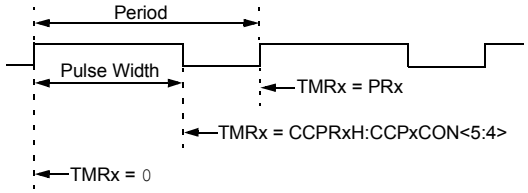
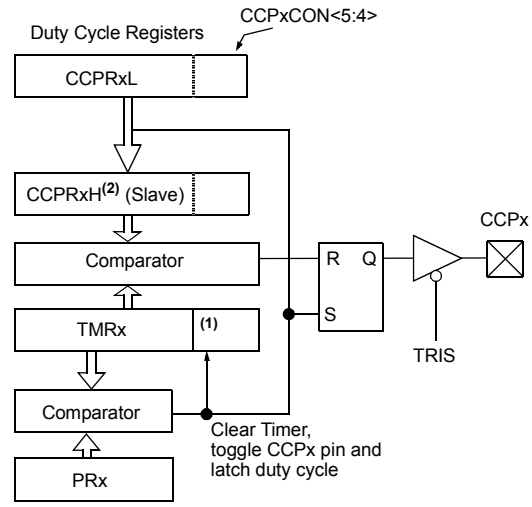



**FIGURE 23-4: SIMPLIFIED PWM BLOCK DIAGRAM**




**Note 1:** The 8-bit timer TMRx register is concatenated with the 2-bit internal system clock (FOSC), or 2 bits of the prescaler, to create the 10-bit time base.

**2:** In PWM mode, CCPRxH is a read-only register.

### 23.4.4 AUTO-RESTART MODE

The Enhanced PWM can be configured to automatically restart the PWM signal once the auto-shutdown condition has been removed. Auto-restart is enabled by setting the PxRSEN bit in the PWMxCON register.

If auto-restart is enabled, the CCPxASE bit will remain set as long as the auto-shutdown condition is active. When the auto-shutdown condition is removed, the CCPxASE bit will be cleared via hardware and normal operation will resume.

**FIGURE 23-15: PWM AUTO-SHUTDOWN WITH AUTO-RESTART (PXRSEN = 1)**

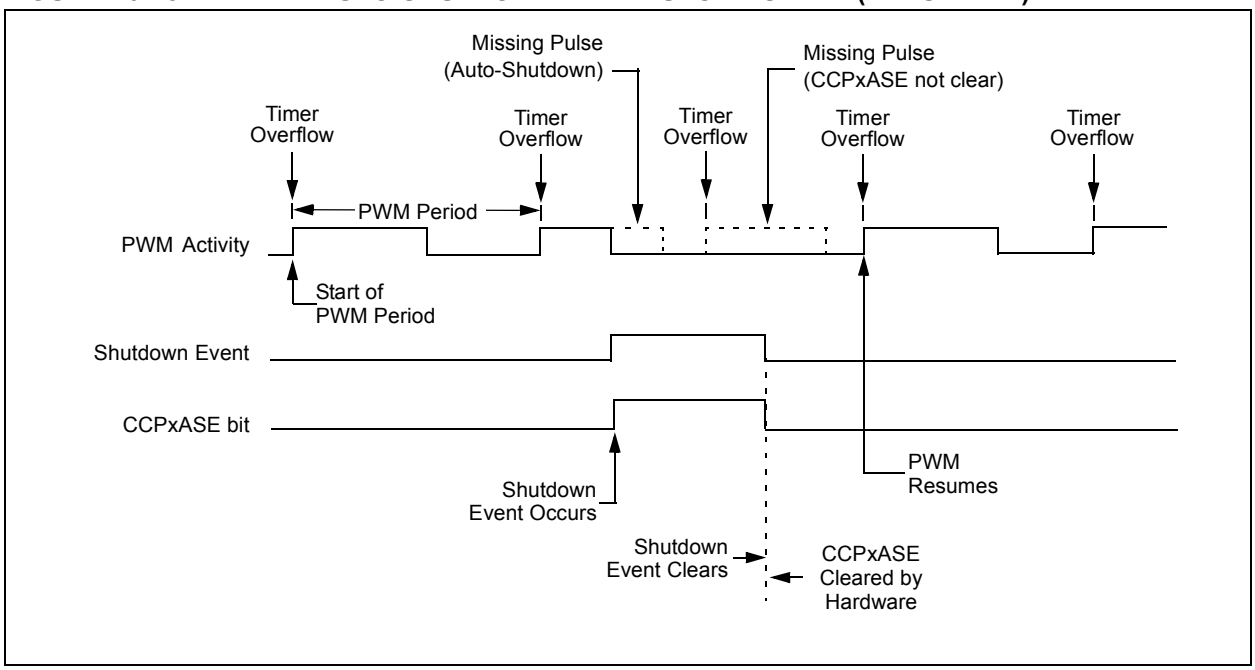


**TABLE 23-10: SUMMARY OF REGISTERS ASSOCIATED WITH ENHANCED PWM**

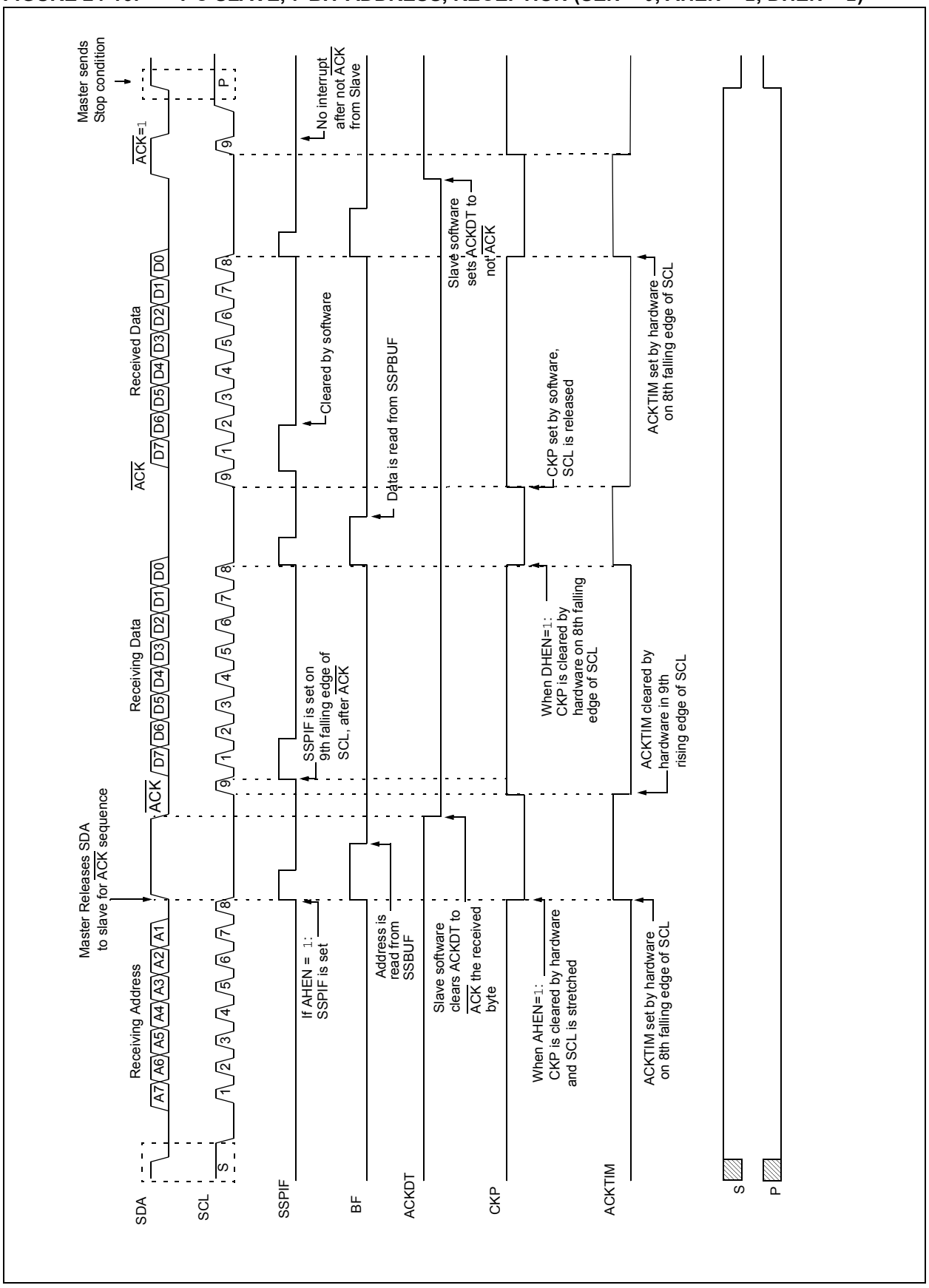
| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| CCPxCON | PxM<1:0>(1) | | DCxB<1:0> | | CCPxM<3:0> | | | | 214 |
| CCPxAS | CCPxASE | CCPxAS<2:0> | | | PSSxAC<1:0> | | PSSxBD<1:0> | | 216 |
| CCPTMRS0 | C4TSEL<1:0> | | C3TSEL<1:0> | | C2TSEL<1:0> | | C1TSEL<1:0> | | 215 |
| CCPTMRS1 | — | — | — | — | — | — | C5TSEL<1:0> | | 215 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 82 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 83 |
| PIE2 | OSFIE | C2IE | C1IE | EEIE | BCLIE | LCDIE | — | CCP2IE | 84 |
| PIE3 | — | CCP5IE | CCP4IE | CCP3IE | TMR6IE | — | TMR4IE | — | 85 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 86 |
| PIR2 | OSFIF | C2IF | C1IF | EEIF | BCLIF | LCDIF | — | CCP2IF | 87 |
| PIR3 | — | CCP5IF | CCP4IF | CCP3IF | TMR6IF | — | TMR4IF | — | 88 |
| PRx | Timer2/4/6 Period Register | | | | | | | | 187* |
| PSTRxCON | — | — | — | STRxSYNC | STRxD | STRxC | STRxB | STRxA | 218 |
| PWMxCON | PxRSEN | PxDC<6:0> | | | | | | | 217 |
| TxCON | — | TxOUTPS<3:0> | | | | TMRxON | TxCKPS<:0>1 | | 189 |
| TMRx | Timer2/4/6 Module Register | | | | | | | | 187 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 116 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 121 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 125 |

**Legend:** — = Unimplemented location, read as '0'. Shaded cells are not used by the PWM.

**Note 1:** Applies to ECCP modules only.

* Page provides register information.

**FIGURE 24-16: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 1)**



**REGISTER 24-4: SSPCON3: SSP CONTROL REGISTER 3**

| R-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7 **ACKTIM:** Acknowledge Time Status bit ($I^2C$ mode only)**(3)**
1 = Indicates the $I^2C$ bus is in an Acknowledge sequence, set on $8^{TH}$ falling edge of SCL clock
0 = Not an Acknowledge sequence, cleared on $9^{TH}$ rising edge of SCL clock

bit 6 **PCIE**: Stop Condition Interrupt Enable bit ($I^2C$ mode only)
1 = Enable interrupt on detection of Stop condition
0 = Stop detection interrupts are disabled**(2)**

bit 5 **SCIE**: Start Condition Interrupt Enable bit ($I^2C$ mode only)
1 = Enable interrupt on detection of Start or Restart conditions
0 = Start detection interrupts are disabled**(2)**

bit 4 **BOEN:** Buffer Overwrite Enable bit
In SPI Slave mode:**(1)**
1 = SSPBUF updates every time that a new data byte is shifted in ignoring the BF bit
0 = If new byte is received with BF bit of the SSPSTAT register already set, SSPOV bit of the SSPCON1 register is set, and the buffer is not updated
In $I^2C$ Master and SPI Master mode:
This bit is ignored.
In $I^2C$ Slave mode:
1 = SSPBUF is updated and $\overline{ACK}$ is generated for a received address/data byte, ignoring the state of the SSPOV bit only if the BF bit = 0.
0 = SSPBUF is only updated when SSPOV is clear

bit 3 **SDAHT:** SDA Hold Time Selection bit ($I^2C$ mode only)
1 = Minimum of 300 ns hold time on SDA after the falling edge of SCL
0 = Minimum of 100 ns hold time on SDA after the falling edge of SCL

bit 2 **SBCDE:** Slave Mode Bus Collision Detect Enable bit ($I^2C$ Slave mode only)
If on the rising edge of SCL, SDA is sampled low when the module is outputting a high state, the BCLIF bit of the PIR2 register is set, and bus goes idle
1 = Enable slave bus collision interrupts
0 = Slave bus collision interrupts are disabled

bit 1 **AHEN:** Address Hold Enable bit ($I^2C$ Slave mode only)
1 = Following the 8th falling edge of SCL for a matching received address byte; CKP bit of the SSPCON1 register will be cleared and the SCL will be held low.
0 = Address holding is disabled

bit 0 **DHEN:** Data Hold Enable bit ($I^2C$ Slave mode only)
1 = Following the 8th falling edge of SCL for a received data byte; slave hardware clears the CKP bit of the SSPCON1 register and SCL is held low.
0 = Data holding is disabled

**Note 1:** For daisy-chained SPI operation; allows the user to ignore all but the last received byte. SSPOV is still set when a new byte is received and BF = 1, but hardware continues to write the most recent byte to SSPBUF.
**2:** This bit has no effect in Slave modes that Start and Stop condition detection is explicitly listed as enabled.
**3:** The ACKTIM Status bit is only active when the AHEN bit or DHEN bit is set.

**NOTES:**

## 25.5 EUSART Synchronous Mode

Synchronous serial communications are typically used in systems with a single master and one or more slaves. The master device contains the necessary circuitry for baud rate generation and supplies the clock for all devices in the system. Slave devices can take advantage of the master clock by eliminating the internal clock generation circuitry.

There are two signal lines in Synchronous mode: a bidirectional data line and a clock line. Slaves use the external clock supplied by the master to shift the serial data into and out of their respective receive and transmit shift registers. Since the data line is bidirectional, synchronous operation is half-duplex only. Half-duplex refers to the fact that master and slave devices can receive and transmit data, but not both simultaneously. The EUSART can operate as either a master or slave device.

Start and Stop bits are not used in synchronous transmissions.

### 25.5.1 SYNCHRONOUS MASTER MODE

The following bits are used to configure the EUSART for synchronous master operation:

- SYNC = 1
- CSRC = 1
- SREN = 0 (for transmit); SREN = 1 (for receive)
- CREN = 0 (for transmit); CREN = 1 (for receive)
- SPEN = 1

Setting the SYNC bit of the TXSTA register configures the device for synchronous operation. Setting the CSRC bit of the TXSTA register configures the device as a master. Clearing the SREN and CREN bits of the RCSTA register ensures that the device is in the Transmit mode, otherwise the device will be configured to receive. Setting the SPEN bit of the RCSTA register enables the EUSART.

#### 25.5.1.1 Master Clock

Synchronous data transfers use a separate clock line, which is synchronous with the data. A device configured as a master transmits the clock on the TX/CK line. The TX/CK pin output driver is automatically enabled when the EUSART is configured for synchronous transmit or receive operation. Serial data bits change on the leading edge to ensure they are valid at the trailing edge of each clock. One clock cycle is generated for each data bit. Only as many clock cycles are generated as there are data bits.

#### 25.5.1.2 Clock Polarity

A clock polarity option is provided for Microwire compatibility. Clock polarity is selected with the SCKP bit of the BAUDCON register. Setting the SCKP bit sets the clock Idle state as high. When the SCKP bit is set, the data changes on the falling edge of each clock. Clearing the SCKP bit sets the Idle state as low. When the SCKP bit is cleared, the data changes on the rising edge of each clock.

#### 25.5.1.3 Synchronous Master Transmission

Data is transferred out of the device on the RX/DT pin. The RX/DT and TX/CK pin output drivers are automatically enabled when the EUSART is configured for synchronous master transmit operation.

A transmission is initiated by writing a character to the TXREG register. If the TSR still contains all or part of a previous character the new character data is held in the TXREG until the last bit of the previous character has been transmitted. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXREG is immediately transferred to the TSR. The transmission of the character commences immediately following the transfer of the data to the TSR from the TXREG.

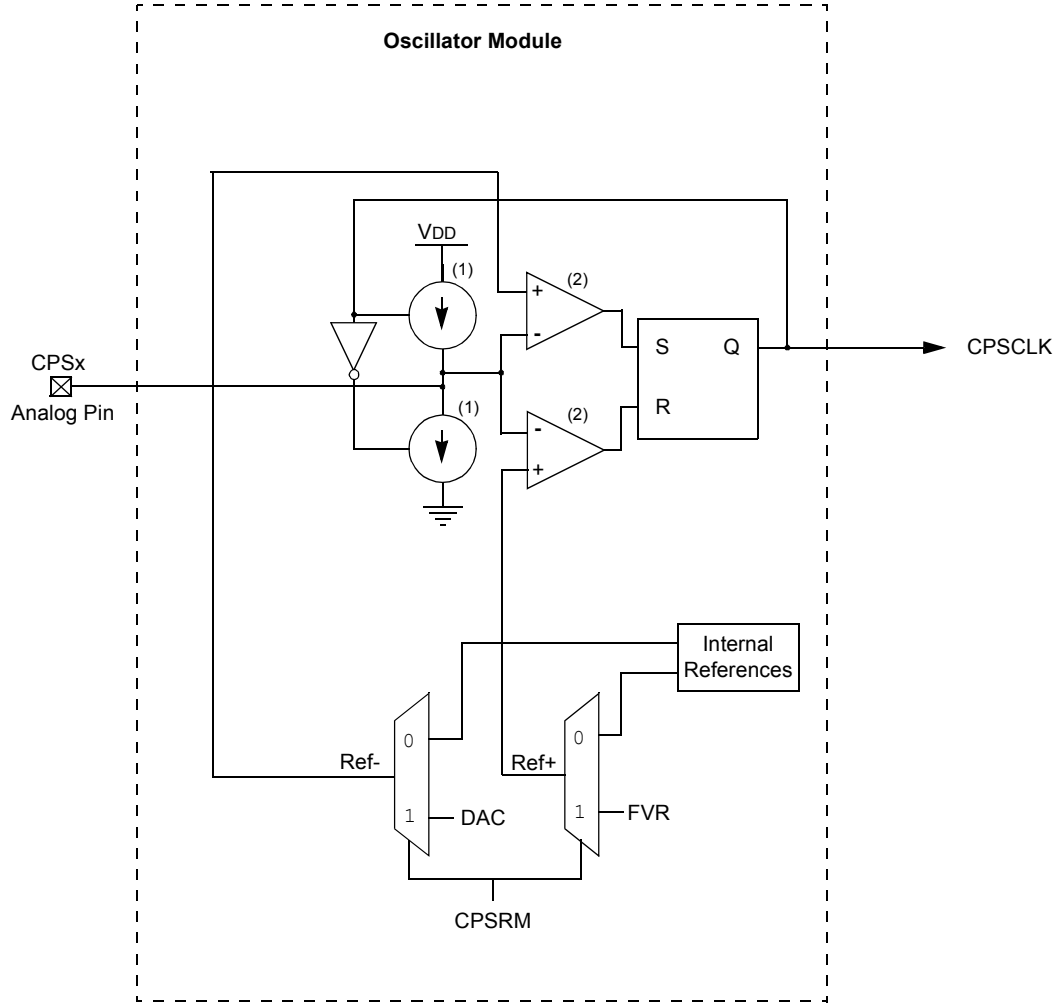
Each data bit changes on the leading edge of the master clock and remains valid until the subsequent leading clock edge.

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

#### 25.5.1.4 Synchronous Master Transmission Set-up:

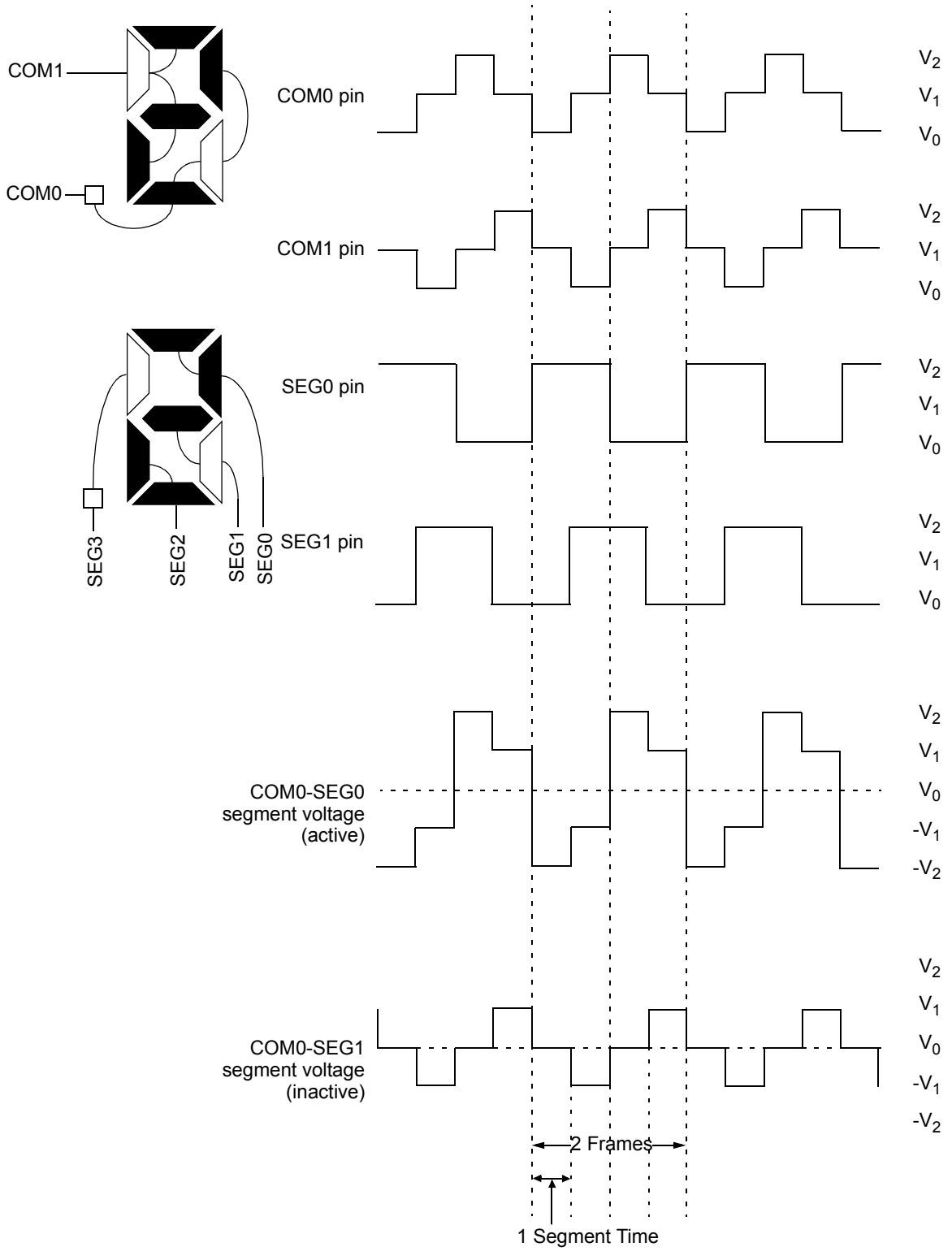
1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 25.4 "EUSART Baud Rate Generator (BRG)"**).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Disable Receive mode by clearing bits SREN and CREN.
4. Enable Transmit mode by setting the TXEN bit.
5. If 9-bit transmission is desired, set the TX9 bit.
6. If interrupts are desired, set the TXIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
7. If 9-bit transmission is selected, the ninth bit should be loaded in the TX9D bit.
8. Start transmission by loading data to the TXREG register.

**FIGURE 26-2: CAPACITIVE SENSING OSCILLATOR BLOCK DIAGRAM**



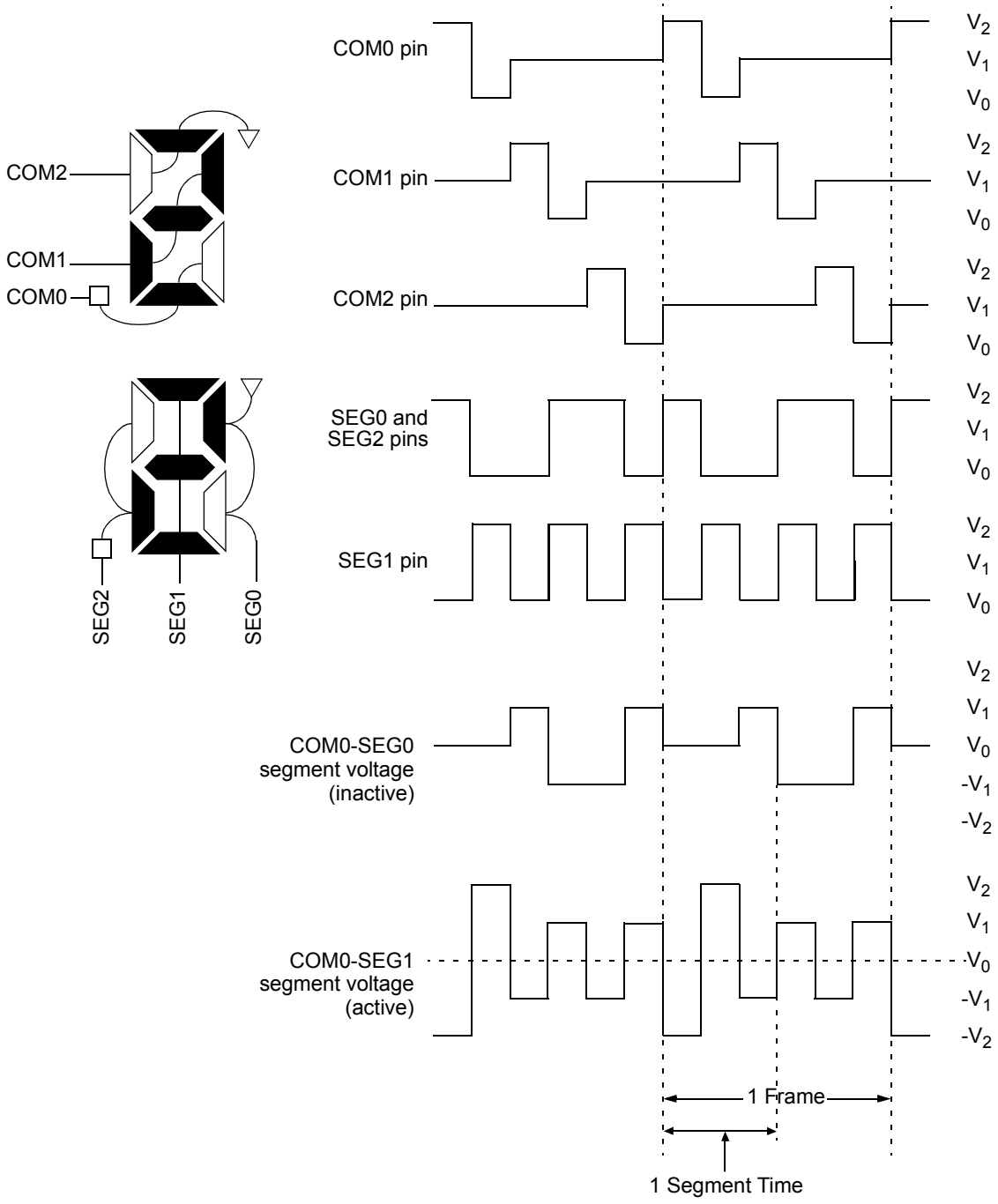

**Note 1:** Module enable, voltage reference mode, and current range selections are not shown.
**2:** Comparators remain active in Noise Detection mode.

**FIGURE 27-10: TYPE-B WAVEFORMS IN 1/2 MUX, 1/2 BIAS DRIVE**




**Note:** 1 Frame = 2 single segment times.

**FIGURE 27-13: TYPE-A WAVEFORMS IN 1/3 MUX, 1/2 BIAS DRIVE**




**Note:** 1 Frame = 2 single segment times.

**TABLE 29-3: PIC16(L)F1933 INSTRUCTION SET**

| Mnemonic, Operands | | Description | Cycles | 14-Bit Opcode MSb | | | LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| **BYTE-ORIENTED FILE REGISTER OPERATIONS** | | | | | | | | | |
| ADDWF | f, d | Add W and f | 1 | 00 | 0111 | dfff | ffff | C, DC, Z | 2 |
| ADDWFC | f, d | Add with Carry W and f | 1 | 11 | 1101 | dfff | ffff | C, DC, Z | 2 |
| ANDWF | f, d | AND W with f | 1 | 00 | 0101 | dfff | ffff | Z | 2 |
| ASRF | f, d | Arithmetic Right Shift | 1 | 11 | 0111 | dfff | ffff | C, Z | 2 |
| LSLF | f, d | Logical Left Shift | 1 | 11 | 0101 | dfff | ffff | C, Z | 2 |
| LSRF | f, d | Logical Right Shift | 1 | 11 | 0110 | dfff | ffff | C, Z | 2 |
| CLRF | f | Clear f | 1 | 00 | 0001 | lfff | ffff | Z | 2 |
| CLRW | – | Clear W | 1 | 00 | 0001 | 0000 | 00xx | Z | |
| COMF | f, d | Complement f | 1 | 00 | 1001 | dfff | ffff | Z | 2 |
| DECF | f, d | Decrement f | 1 | 00 | 0011 | dfff | ffff | Z | 2 |
| INCF | f, d | Increment f | 1 | 00 | 1010 | dfff | ffff | Z | 2 |
| IORWF | f, d | Inclusive OR W with f | 1 | 00 | 0100 | dfff | ffff | Z | 2 |
| MOVF | f, d | Move f | 1 | 00 | 1000 | dfff | ffff | Z | 2 |
| MOVWF | f | Move W to f | 1 | 00 | 0000 | 1fff | ffff | | 2 |
| RLF | f, d | Rotate Left f through Carry | 1 | 00 | 1101 | dfff | ffff | C | 2 |
| RRF | f, d | Rotate Right f through Carry | 1 | 00 | 1100 | dfff | ffff | C | 2 |
| SUBWF | f, d | Subtract W from f | 1 | 00 | 0010 | dfff | ffff | C, DC, Z | 2 |
| SUBWFB | f, d | Subtract with Borrow W from f | 1 | 11 | 1011 | dfff | ffff | C, DC, Z | 2 |
| SWAPF | f, d | Swap nibbles in f | 1 | 00 | 1110 | dfff | ffff | | 2 |
| XORWF | f, d | Exclusive OR W with f | 1 | 00 | 0110 | dfff | ffff | Z | 2 |
| **BYTE ORIENTED SKIP OPERATIONS** | | | | | | | | | |
| DECFSZ | f, d | Decrement f, Skip if 0 | 1(2) | 00 | 1011 | dfff | ffff | | 1, 2 |
| INCFSZ | f, d | Increment f, Skip if 0 | 1(2) | 00 | 1111 | dfff | ffff | | 1, 2 |
| **BIT-ORIENTED FILE REGISTER OPERATIONS** | | | | | | | | | |
| BCF | f, b | Bit Clear f | 1 | 01 | 00bb | bfff | ffff | | 2 |
| BSF | f, b | Bit Set f | 1 | 01 | 01bb | bfff | ffff | | 2 |
| **BIT-ORIENTED SKIP OPERATIONS** | | | | | | | | | |
| BTFSC | f, b | Bit Test f, Skip if Clear | 1 (2) | 01 | 10bb | bfff | ffff | | 1, 2 |
| BTFSS | f, b | Bit Test f, Skip if Set | 1 (2) | 01 | 11bb | bfff | ffff | | 1, 2 |
| **LITERAL OPERATIONS** | | | | | | | | | |
| ADDLW | k | Add literal and W | 1 | 11 | 1110 | kkkk | kkkk | C, DC, Z | |
| ANDLW | k | AND literal with W | 1 | 11 | 1001 | kkkk | kkkk | Z | |
| IORLW | k | Inclusive OR literal with W | 1 | 11 | 1000 | kkkk | kkkk | Z | |
| MOVLB | k | Move literal to BSR | 1 | 00 | 0000 | 001k | kkkk | | |
| MOVLP | k | Move literal to PCLATH | 1 | 11 | 0001 | 1kkk | kkkk | | |
| MOVLW | k | Move literal to W | 1 | 11 | 0000 | kkkk | kkkk | | |
| SUBLW | k | Subtract W from literal | 1 | 11 | 1100 | kkkk | kkkk | C, DC, Z | |
| XORLW | k | Exclusive OR literal with W | 1 | 11 | 1010 | kkkk | kkkk | Z | |

**Note 1:** If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

**2:** If this instruction addresses an INDF register and the MSb of the corresponding FSR is set, this instruction will require one additional instruction cycle.

**TABLE 29-3: PIC16(L)F1933 INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | | Description | Cycles | 14-Bit Opcode MSb | LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|---|
| **CONTROL OPERATIONS** | | | | | | | |
| BRA | k | Relative Branch | 2 | 11 001k kkkk kkkk | | | |
| BRW | – | Relative Branch with W | 2 | 00 0000 0000 1011 | | | |
| CALL | k | Call Subroutine | 2 | 10 0kkk kkkk kkkk | | | |
| CALLW | – | Call Subroutine with W | 2 | 00 0000 0000 1010 | | | |
| GOTO | k | Go to address | 2 | 10 1kkk kkkk kkkk | | | |
| RETFIE | k | Return from interrupt | 2 | 00 0000 0000 1001 | | | |
| RETLW | k | Return with literal in W | 2 | 11 0100 kkkk kkkk | | | |
| RETURN | – | Return from Subroutine | 2 | 00 0000 0000 1000 | | | |
| **INHERENT OPERATIONS** | | | | | | | |
| CLRWDT | – | Clear Watchdog Timer | 1 | 00 0000 0110 0100 | | $\overline{TO}$, $\overline{PD}$ | |
| NOP | – | No Operation | 1 | 00 0000 0000 0000 | | | |
| OPTION | – | Load OPTION_REG register with W | 1 | 00 0000 0110 0010 | | | |
| RESET | – | Software device Reset | 1 | 00 0000 0000 0001 | | | |
| SLEEP | – | Go into Standby mode | 1 | 00 0000 0110 0011 | | $\overline{TO}$, $\overline{PD}$ | |
| TRIS | f | Load TRIS register with W | 1 | 00 0000 0110 0fff | | | |
| **C-COMPILER OPTIMIZED** | | | | | | | |
| ADDFSR | n, k | Add Literal k to FSRn | 1 | 11 0001 0nkk kkkk | | | |
| MOVIW | n mm | Move Indirect FSRn to W with pre/post inc/dec modifier, mm | 1 | 00 0000 0001 0nmm | | Z | **2, 3** |
| | k[n] | Move INDFn to W, Indexed Indirect. | 1 | 11 1111 0nkk kkkk | | Z | **2** |
| MOVWI | n mm | Move W to Indirect FSRn with pre/post inc/dec modifier, mm | 1 | 00 0000 0001 1nmm | | | **2, 3** |
| | k[n] | Move W to INDFn, Indexed Indirect. | 1 | 11 1111 1nkk kkkk | | | **2** |

**Note 1:** If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

**2:** If this instruction addresses an INDF register and the MSb of the corresponding FSR is set, this instruction will require one additional instruction cycle.

**3:** See Table in the MOVIW and MOVWI instruction descriptions.

**MOVWI** **Move W to INDFn**

Syntax: [ *label* ] MOVWI ++FSRn
[ *label* ] MOVWI --FSRn
[ *label* ] MOVWI FSRn++
[ *label* ] MOVWI FSRn--
[ *label* ] MOVWI k[FSRn]

Operands: n ∈ [0,1]
mm ∈ [00,01, 10, 11]
-32 ≤ k ≤ 31

Operation: W → INDFn
Effective address is determined by
- FSR + 1 (preincrement)
- FSR - 1 (predecrement)
- FSR + k (relative offset)

After the Move, the FSR value will be either:
- FSR + 1 (all increments)
- FSR - 1 (all decrements)

Unchanged

Status Affected: None

| Mode | Syntax | mm |
|---|---|---|
| Preincrement | ++FSRn | 00 |
| Predecrement | --FSRn | 01 |
| Postincrement | FSRn++ | 10 |
| Postdecrement | FSRn-- | 11 |

Description: This instruction is used to move data between W and one of the indirect registers (INDFn). Before/after this move, the pointer (FSRn) is updated by pre/post incrementing/decrementing it.

Note: The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the FSRn.

FSRn is limited to the range 0000h - FFFFh. Incrementing/decrementing it beyond these bounds will cause it to wrap-around.

The increment/decrement operation on FSRn WILL NOT affect any Status bits.

**NOP** **No Operation**

Syntax: [ *label* ] NOP

Operands: None

Operation: No operation

Status Affected: None

Description: No operation.

Words: 1

Cycles: 1

Example:
```
NOP
```

**OPTION** **Load OPTION_REG Register with W**

Syntax: [ *label* ] OPTION

Operands: None

Operation: (W) → OPTION_REG

Status Affected: None

Description: Move data from W register to OPTION_REG register.

Words: 1

Cycles: 1

Example:
```
OPTION
```
Before Instruction
OPTION_REG = 0xFF
W = 0x4F
After Instruction
OPTION_REG = 0x4F
W = 0x4F

**RESET** **Software Reset**

Syntax: [ *label* ] RESET

Operands: None

Operation: Execute a device Reset. Resets the $\overline{RI}$ flag of the PCON register.

Status Affected: None

Description: This instruction provides a way to execute a hardware Reset by software.

**FIGURE 31-21: IDD TYPICAL, HFINTOSC MODE, PIC16LF1933 ONLY**

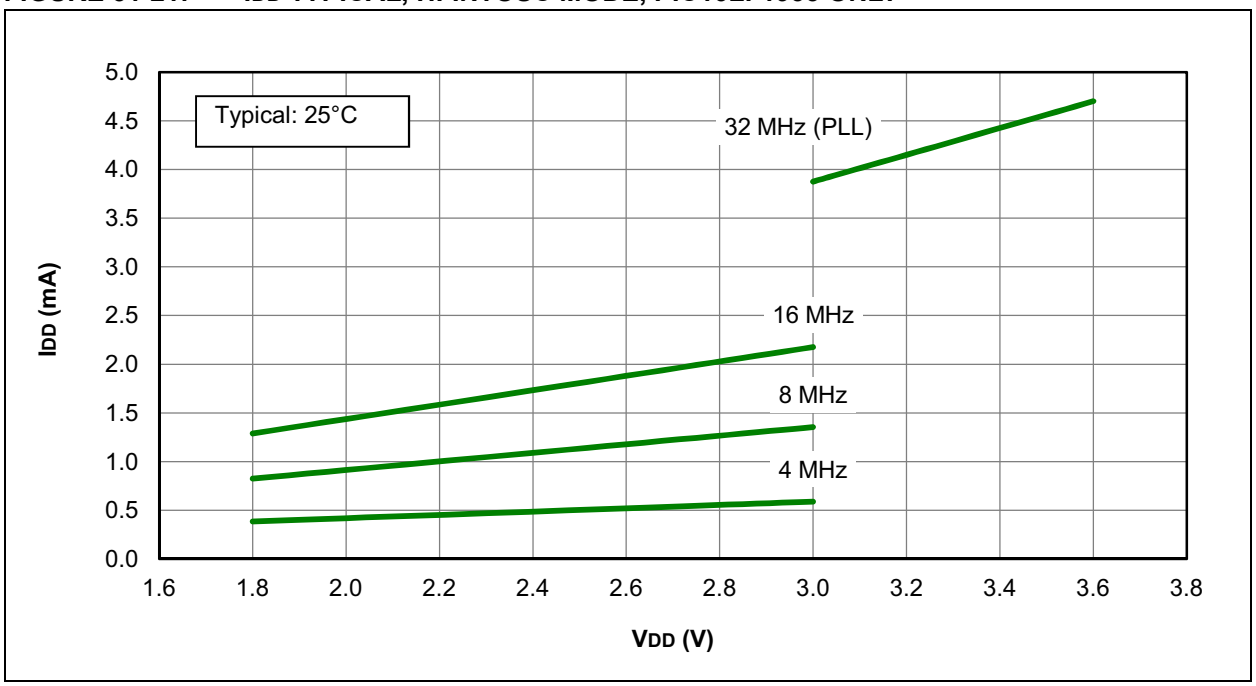
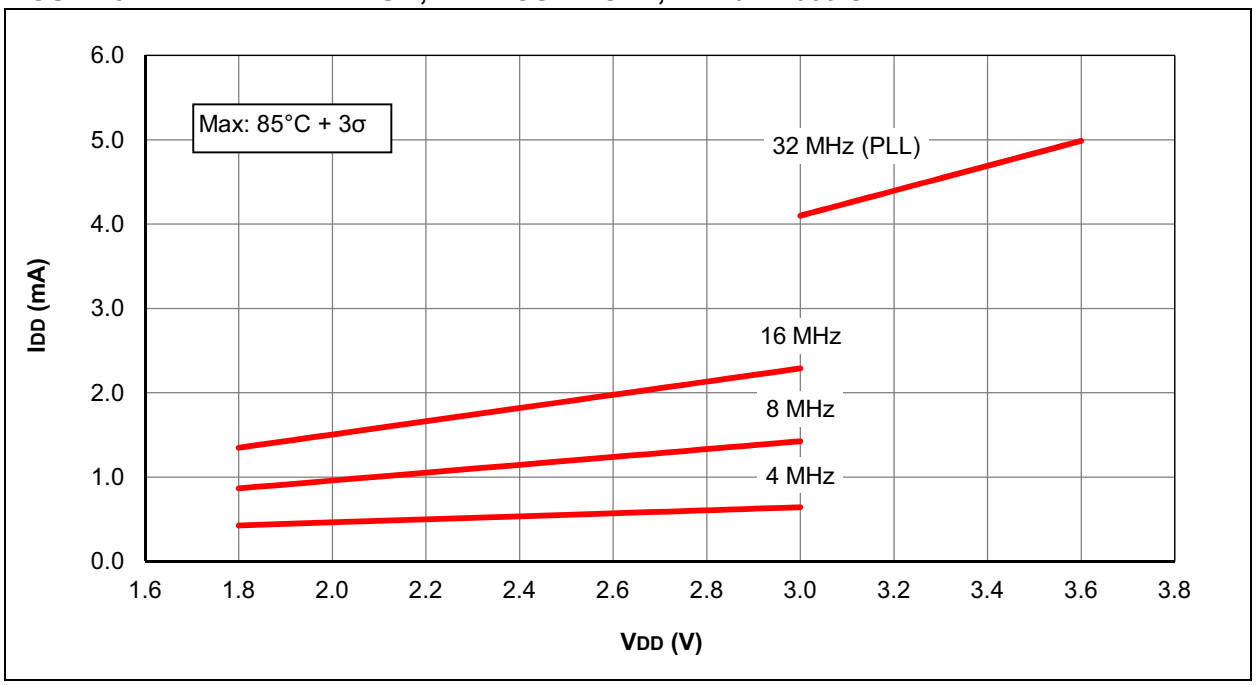



**FIGURE 31-22: IDD MAXIMUM, HFINTOSC MODE, PIC16LF1933 ONLY**



## 32.2 MPLAB C Compilers for Various Device Families

The MPLAB C Compiler code development systems are complete ANSI C compilers for Microchip's PIC18, PIC24 and PIC32 families of microcontrollers and the dsPIC30 and dsPIC33 families of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

## 32.3 HI-TECH C for Various Device Families

The HI-TECH C Compiler code development systems are complete ANSI C compilers for Microchip's PIC family of microcontrollers and the dsPIC family of digital signal controllers. These compilers provide powerful integration capabilities, omniscient code generation and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

The compilers include a macro assembler, linker, preprocessor, and one-step driver, and can run on multiple platforms.

## 32.4 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multi-purpose source files
- Directives that allow complete control over the assembly process

## 32.5 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler and the MPLAB C18 C Compiler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

## 32.6 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC devices. MPLAB C Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility