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	48MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	24
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 19x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f26k22-e-ss

2.7.1 LFINTOSC

The Low-Frequency Internal Oscillator (LFINTOSC) is a 31.25 kHz internal clock source. The LFINTOSC is not tunable, but is designed to be stable across temperature and voltage. See **Section 27.0 "Electrical Specifications"** for the LFINTOSC accuracy specifications.

The output of the LFINTOSC can be a clock source to the primary clock or the INTOSC clock (see Figure 2-1). The LFINTOSC is also the clock source for the Power-up Timer (PWRT), Watchdog Timer (WDT) and Fail-Safe Clock Monitor (FSCM).

2.7.2 FREQUENCY SELECT BITS (IRCF)

The HFINTOSC (16 MHz) and MFINTOSC (500 MHz) outputs connect to a divide circuit that provides frequencies of 16 MHz to 31.25 kHz. These divide circuit frequencies, along with the 31.25 kHz LFINTOSC output, are multiplexed to provide a single INTOSC clock output (see Figure 2-1). The IRCF<2:0> bits of the OSCCON register, the MFIOSEL bit of the OSCCON2 register and the INTSRC bit of the OSCTUNE register, select the output frequency of the internal oscillators. One of eight frequencies can be selected via software:

- 16 MHz
- 8 MHz
- 4 MHz
- 2 MHz
- 1 MHz (default after Reset)
- 500 kHz (MFINTOSC or HFINTOSC)
- 250 kHz (MFINTOSC or HFINTOSC)
- 31 kHz (LFINTOSC, MFINTOSC or HFINTOSC)

2.7.3 INTOSC FREQUENCY DRIFT

The factory calibrates the internal oscillator block outputs (HFINTOSC/MFINTOSC) for 16 MHz/500 kHz. However, this frequency may drift as V_{DD} or temperature changes. It is possible to adjust the HFINTOSC/MFINTOSC frequency by modifying the value of the TUN<5:0> bits in the OSCTUNE register. This has no effect on the LFINTOSC clock source frequency.

Tuning the HFINTOSC/MFINTOSC source requires knowing when to make the adjustment, in which direction it should be made and, in some cases, how large a change is needed. Three possible compensation techniques are discussed in the following sections. However, other techniques may be used.

2.7.3.1 Compensating with the EUSART

An adjustment may be required when the EUSART begins to generate framing errors or receives data with errors while in Asynchronous mode. Framing errors indicate that the device clock frequency is too high; to adjust for this, decrement the value in OSCTUNE to reduce the clock frequency. On the other hand, errors in data may suggest that the clock speed is too low; to compensate, increment OSCTUNE to increase the clock frequency.

2.7.3.2 Compensating with the Timers

This technique compares device clock speed to some reference clock. Two timers may be used; one timer is clocked by the peripheral clock, while the other is clocked by a fixed reference source, such as the Timer1 oscillator.

Both timers are cleared, but the timer clocked by the reference generates interrupts. When an interrupt occurs, the internally clocked timer is read and both timers are cleared. If the internally clocked timer value is greater than expected, then the internal oscillator block is running too fast. To adjust for this, decrement the OSCTUNE register.

2.7.3.3 Compensating with the CCP Module in Capture Mode

A CCP module can use free running Timer1, Timer3 or Timer5 clocked by the internal oscillator block and an external event with a known period (i.e., AC power frequency). The time of the first event is captured in the CCPRxH:CCPRxL registers and is recorded for use later. When the second event causes a capture, the time of the first event is subtracted from the time of the second event. Since the period of the external event is known, the time difference between events can be calculated.

If the measured time is much greater than the calculated time, the internal oscillator block is running too fast; to compensate, decrement the OSCTUNE register. If the measured time is much less than the calculated time, the internal oscillator block is running too slow; to compensate, increment the OSCTUNE register.

4.7 Reset State of Registers

Some registers are unaffected by a Reset. Their status is unknown on POR and unchanged by all other Resets. All other registers are forced to a “Reset state” depending on the type of Reset that occurred.

Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits from the RCON register, $\overline{RI}$, $\overline{TO}$, $\overline{PD}$, $\overline{POR}$ and $\overline{BOR}$, are set or cleared differently in different Reset situations, as indicated in Table 4-3. These bits are used by software to determine the nature of the Reset.

Table 5-2 describes the Reset states for all of the Special Function Registers. The table identifies differences between Power-On Reset (POR)/Brown-Out Reset (BOR) and all other Resets, (i.e., Master Clear, WDT Resets, STKFUL, STKUNF, etc.). Additionally, the table identifies register bits that are changed when the device receives a wake-up from WDT or other interrupts.

TABLE 4-3: STATUS BITS, THEIR SIGNIFICANCE AND THE INITIALIZATION CONDITION FOR RCON REGISTER

Condition	Program Counter	RCON Register						STKPTR Register	
		SBOREN	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	STKFUL	STKUNF
Power-on Reset	0000h	1	1	1	1	0	0	0	0
RESET Instruction	0000h	u(2)	0	u	u	u	u	u	u
Brown-out Reset	0000h	u(2)	1	1	1	u	0	u	u
$\overline{MCLR}$ during Power-Managed Run Modes	0000h	u(2)	u	1	u	u	u	u	u
$\overline{MCLR}$ during Power-Managed Idle Modes and Sleep Mode	0000h	u(2)	u	1	0	u	u	u	u
WDT Time-out during Full Power or Power-Managed Run Mode	0000h	u(2)	u	0	u	u	u	u	u
$\overline{MCLR}$ during Full Power Execution	0000h	u(2)	u	u	u	u	u	u	u
Stack Full Reset (STVREN = 1)	0000h	u(2)	u	u	u	u	u	1	u
Stack Underflow Reset (STVREN = 1)	0000h	u(2)	u	u	u	u	u	u	1
Stack Underflow Error (not an actual Reset, STVREN = 0)	0000h	u(2)	u	u	u	u	u	u	1
WDT Time-out during Power-Managed Idle or Sleep Modes	PC + 2	u(2)	u	0	0	u	u	u	u
Interrupt Exit from Power-Managed Modes	PC + 2(1)	u(2)	u	u	0	u	u	u	u

Legend: u = unchanged

Note 1: When the wake-up is due to an interrupt and the GIEH or GIEL bits are set, the PC is loaded with the interrupt vector (008h or 0018h).

2: Reset state is ‘1’ for SBOREN and unchanged for all other Resets when software BOR is enabled (BOREN<1:0> Configuration bits = 01). Otherwise, the Reset state is ‘0’.

TABLE 4-4: REGISTERS ASSOCIATED WITH RESETS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
RCON	IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	56
STKPTR	STKFUL	STKUNF	—	STKPTR<4:0>					67

Legend: — = unimplemented locations, read as ‘0’. Shaded bits are not used for Resets.

5.1.2.3 PUSH and POP Instructions

Since the Top-of-Stack is readable and writable, the ability to push values onto the stack and pull values off the stack without disturbing normal program execution is a desirable feature. The PIC18 instruction set includes two instructions, PUSH and POP, that permit the TOS to be manipulated under software control. TOSU, TOSH and TOSL can be modified to place data or a return address on the stack.

The PUSH instruction places the current PC value onto the stack. This increments the Stack Pointer and loads the current PC value onto the stack.

The POP instruction discards the current TOS by decrementing the Stack Pointer. The previous value pushed onto the stack then becomes the TOS value.

5.2 Register Definitions: Stack Pointer

REGISTER 5-1: STKPTR: STACK POINTER REGISTER

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKFUL(1)	STKUNF(1)	—	STKPTR<4:0>				
bit 7							bit 0

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

bit 7 **STKFUL:** Stack Full Flag bit(1)
1 = Stack became full or overflowed
0 = Stack has not become full or overflowed

bit 6 **STKUNF:** Stack Underflow Flag bit(1)
1 = Stack Underflow occurred
0 = Stack Underflow did not occur

bit 5 **Unimplemented:** Read as ‘0’

bit 4-0 **STKPTR<4:0>:** Stack Pointer Location bits

Note 1: Bit 7 and bit 6 are cleared by user software or by a POR.

5.2.0.1 Stack Full and Underflow Resets

Device Resets on Stack Overflow and Stack Underflow conditions are enabled by setting the STVREN bit in Configuration Register 4L. When STVREN is set, a full or underflow will set the appropriate STKFUL or STKUNF bit and then cause a device Reset. When STVREN is cleared, a full or underflow condition will set the appropriate STKFUL or STKUNF bit but not cause a device Reset. The STKFUL or STKUNF bits are cleared by the user software or a Power-on Reset.

5.2.1 FAST REGISTER STACK

A fast register stack is provided for the Status, WREG and BSR registers, to provide a “fast return” option for interrupts. The stack for each register is only one level deep and is neither readable nor writable. It is loaded with the current value of the corresponding register when the processor vectors for an interrupt. All interrupt sources will push values into the stack registers. The values in the registers are then loaded back into their associated registers if the RETFIE,FAST instruction is used to return from the interrupt.

If both low and high priority interrupts are enabled, the stack registers cannot be used reliably to return from low priority interrupts. If a high priority interrupt occurs while servicing a low priority interrupt, the stack register values stored by the low priority interrupt will be overwritten. In these cases, users must save the key registers by software during a low priority interrupt.

If interrupt priority is not used, all interrupts may use the fast register stack for returns from interrupt. If no interrupts are used, the fast register stack can be used to restore the Status, WREG and BSR registers at the end of a subroutine call. To use the fast register stack for a subroutine call, a CALL label, FAST instruction must be executed to save the Status, WREG and BSR registers to the fast register stack. A RETURN, FAST instruction is then executed to restore these registers from the fast register stack.

Example 5-1 shows a source code example that uses the fast register stack during a subroutine call and return.

5.3 PIC18 Instruction Cycle

5.3.1 CLOCKING SCHEME

The microcontroller clock input, whether from an internal or external source, is internally divided by four to generate four non-overlapping quadrature clocks (Q1, Q2, Q3 and Q4). Internally, the program counter is incremented on every Q1; the instruction is fetched from the program memory and latched into the instruction register during Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 5-3.

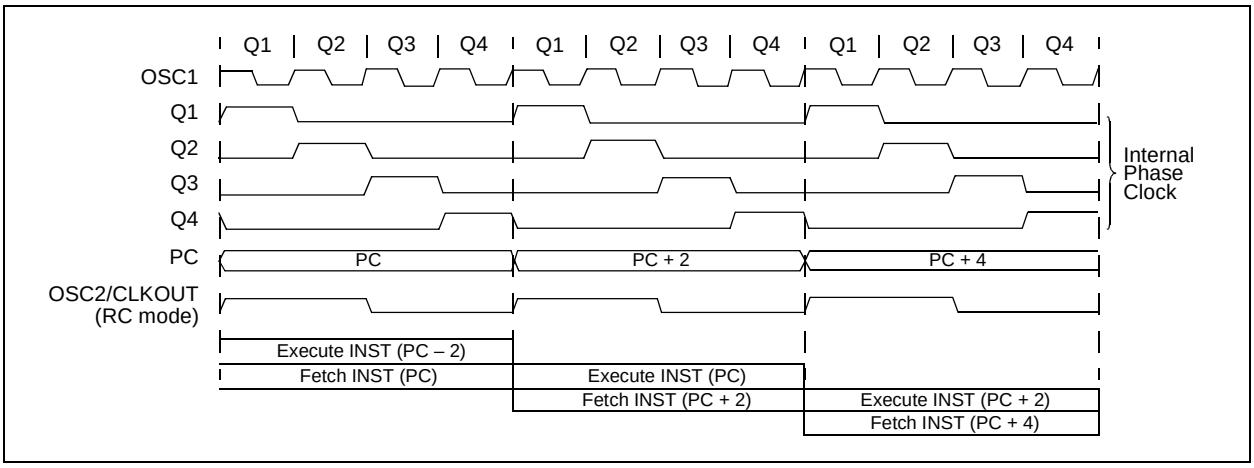
5.3.2 INSTRUCTION FLOW/PIPELINING

An “Instruction Cycle” consists of four Q cycles: Q1 through Q4. The instruction fetch and execute are pipelined in such a manner that a fetch takes one instruction cycle, while the decode and execute take another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., `GOTO`), then two cycles are required to complete the instruction (Example 5-3).

A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the Instruction Register (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3 and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

FIGURE 5-3: CLOCK/INSTRUCTION CYCLE



EXAMPLE 5-3: INSTRUCTION PIPELINE FLOW

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOVLW 55h	Fetch 1	Execute 1				
2. MOVWF PORTB		Fetch 2	Execute 2			
3. BRA SUB_1			Fetch 3	Execute 3		
4. BSF PORTA, BIT3 (Forced NOP)				Fetch 4	Flush (NOP)	
5. Instruction @ address SUB_1					Fetch SUB_1	Execute SUB_1

Note: All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is “flushed” from the pipeline while the new instruction is being fetched and then executed.

EXAMPLE 6-3: WRITING TO FLASH PROGRAM MEMORY

```
                    MOVLW     D'64'                 ; number of bytes in erase block
                    MOVWF     COUNTER
                    MOVLW     BUFFER_ADDR_HIGH      ; point to buffer
                    MOVWF     FSR0H
                    MOVLW     BUFFER_ADDR_LOW
                    MOVWF     FSR0L
                    MOVLW     CODE_ADDR_UPPER       ; Load TBLPTR with the base
                    MOVWF     TBLPTRU               ; address of the memory block
                    MOVLW     CODE_ADDR_HIGH
                    MOVWF     TBLPTRH
                    MOVLW     CODE_ADDR_LOW
                    MOVWF     TBLPTRL
READ_BLOCK
                    TBLRD*+                         ; read into TABLAT, and inc
                    MOVF      TABLAT, W             ; get data
                    MOVWF     POSTINC0              ; store data
                    DECFSZ    COUNTER               ; done?
                    BRA       READ_BLOCK            ; repeat
MODIFY_WORD
                    MOVLW     BUFFER_ADDR_HIGH      ; point to buffer
                    MOVWF     FSR0H
                    MOVLW     BUFFER_ADDR_LOW
                    MOVWF     FSR0L
                    MOVLW     NEW_DATA_LOW          ; update buffer word
                    MOVWF     POSTINC0
                    MOVLW     NEW_DATA_HIGH
                    MOVWF     INDF0
ERASE_BLOCK
                    MOVLW     CODE_ADDR_UPPER       ; load TBLPTR with the base
                    MOVWF     TBLPTRU               ; address of the memory block
                    MOVLW     CODE_ADDR_HIGH
                    MOVWF     TBLPTRH
                    MOVLW     CODE_ADDR_LOW
                    MOVWF     TBLPTRL
                    BSF       EECON1, EEPGD         ; point to Flash program memory
                    BCF       EECON1, CFGS          ; access Flash program memory
                    BSF       EECON1, WREN          ; enable write to memory
                    BSF       EECON1, FREE          ; enable Erase operation
                    BCF       INTCON, GIE           ; disable interrupts
                    MOVLW     55h
Required            MOVWF     EECON2                ; write 55h
Sequence            MOVLW     0AAh
                    MOVWF     EECON2                ; write 0AAh
                    BSF       EECON1, WR            ; start erase (CPU stall)
                    BSF       INTCON, GIE           ; re-enable interrupts
                    TBLRD*-                         ; dummy read decrement
                    MOVLW     BUFFER_ADDR_HIGH      ; point to buffer
                    MOVWF     FSR0H
                    MOVLW     BUFFER_ADDR_LOW
                    MOVWF     FSR0L
WRITE_BUFFER_BACK
                    MOVLW     BlockSize             ; number of bytes in holding register
                    MOVWF     COUNTER
                    MOVLW     D'64'/BlockSize       ; number of write blocks in 64 bytes
                    MOVWF     COUNTER2
WRITE_BYTE_TO_HREGS
                    MOVF      POSTINC0, W           ; get low byte of buffer data
                    MOVWF     TABLAT                ; present data to table latch
                    TBLWT+*                         ; write data, perform a short write
                                                    ; to internal TBLWT holding register.
```

TABLE 10-8: PORTC I/O SUMMARY

Pin Name	Function	TRIS Setting	ANSEL setting	Pin Type	Buffer Type	Description
RC0/P2B/T3CKI/T3G/ T1CKI/SOSCO	RC0	0	—	O	DIG	LATC<0> data output; not affected by analog input.
		1	—	I	ST	PORTC<0> data input; disabled when analog input enabled.
	P2B(2)	0	—	O	DIG	Enhanced CCP2 PWM output 2.
	T3CKI(1)	1	—	I	ST	Timer3 clock input.
	T3G	1	—	I	ST	Timer3 external clock gate input.
	T1CKI	1	—	I	ST	Timer1 clock input.
	SOSCO	x	—	O	XTAL	Secondary oscillator output.
RC1/P2A/CCP2/SOSCI	RC1	0	—	O	DIG	LATC<1> data output; not affected by analog input.
		1	—	I	ST	PORTC<1> data input; disabled when analog input enabled.
	P2A	0	—	O	DIG	Enhanced CCP2 PWM output 1.
	CCP2(1)	0	—	O	DIG	Compare 2 output/PWM 2 output.
		1	—	I	ST	Capture 2 input.
	SOSCI	x	—	I	XTAL	Secondary oscillator input.
RC2/CTPLS/P1A/ CCP1/T5CKI/AN14	RC2	0	0	O	DIG	LATC<2> data output; not affected by analog input.
		1	0	I	ST	PORTC<2> data input; disabled when analog input enabled.
	CTPLS	0	0	O	DIG	CTMU pulse generator output.
	P1A	0	0	O	DIG	Enhanced CCP1 PWM output 1.
	CCP1	0	0	O	DIG	Compare 1 output/PWM 1 output.
		1	0	I	ST	Capture 1 input.
	T5CKI	1	0	I	ST	Timer5 clock input.
	AN14	1	1	I	AN	Analog input 14.
RC3/SCK1/SCL1/AN15	RC3	0	0	O	DIG	LATC<3> data output; not affected by analog input.
		1	0	I	ST	PORTC<3> data input; disabled when analog input enabled.
	SCK1	0	0	O	DIG	MSSP1 SPI Clock output.
		1	0	I	ST	MSSP1 SPI Clock input.
	SCL1	0	0	O	DIG	MSSP1 I^2C Clock output.
		1	0	I	I^2C	MSSP1 I^2C Clock input.
	AN15	1	1	I	AN	Analog input 15.
RC4/SDI1/SDA1/AN16	RC4	0	0	O	DIG	LATC<4> data output; not affected by analog input.
		1	0	I	ST	PORTC<4> data input; disabled when analog input enabled.
	SDI1	1	0	I	ST	MSSP1 SPI data input.
	SDA1	0	0	O	DIG	MSSP1 I^2C data output.
		1	0	I	I^2C	MSSP1 I^2C data input.
	AN16	1	1	I	AN	Analog input 16.

Legend: AN = Analog input or output; TTL = TTL compatible input; HV = High Voltage; OD = Open Drain; XTAL = Crystal; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I^2C = Schmitt Trigger input with I^2C.

Note 1: Default pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are set.

2: Alternate pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are clear.

3: Function on PORTD and PORTE for PIC18(L)F4XK22 devices.

14.1 Capture Mode

The Capture mode function described in this section is identical for all CCP and ECCP modules available on this device family.

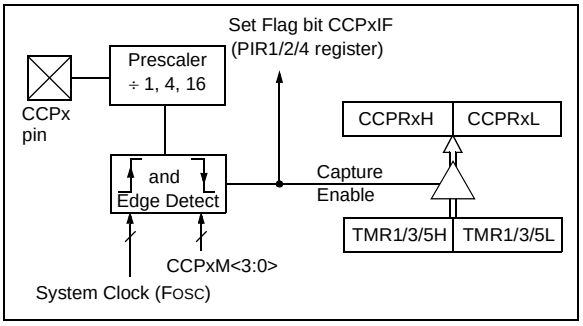
Capture mode makes use of the 16-bit Timer resources, Timer1, Timer3 and Timer5. The timer resources for each CCP capture function are independent and are selected using the CCPTMRS0 and CCPTMRS1 registers. When an event occurs on the CCPx pin, the 16-bit CCPRxH:CCPRxL register pair captures and stores the 16-bit value of the TMRxH:TMRxL register pair, respectively. An event is defined as one of the following and is configured by the CCPxM<3:0> bits of the CCPxCON register:

- Every falling edge
- Every rising edge
- Every 4th rising edge
- Every 16th rising edge

When a capture is made, the corresponding Interrupt Request Flag bit CCPxIF of the PIR1, PIR2 or PIR4 register is set. The interrupt flag must be cleared in software. If another capture occurs before the value in the CCPRxH:CCPRxL register pair is read, the old captured value is overwritten by the new captured value.

Figure 14-1 shows a simplified diagram of the Capture operation.

FIGURE 14-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



14.1.1 CCP PIN CONFIGURATION

In Capture mode, the CCPx pin should be configured as an input by setting the associated TRIS control bit.

Some CCPx outputs are multiplexed on a couple of pins. Table 14-2 shows the CCP output pin multiplexing. Selection of the output pin is determined by the CCPxMX bits in Configuration register 3H (CONFIG3H). Refer to Register 24-4 for more details.

Note:	If the CCPx pin is configured as an output, a write to the port can cause a capture condition.

TABLE 14-2: CCP PIN MULTIPLEXING

CCP OUTPUT	CONFIG 3H Control Bit	Bit Value	PIC18(L)F2XK22 I/O pin	PIC18(L)F4XK22 I/O pin
CCP2	CCP2MX	0	RB3	RB3
		1(*)	RC1	RC1
CCP3	CCP3MX	0(*)	RC6	RE0
		1	RB5	RB5

Legend: * = Default

14.1.2 TIMER1 MODE RESOURCE

The 16-bit Timer resource must be running in Timer mode or Synchronized Counter mode for the CCP module to use the capture feature. In Asynchronous Counter mode, the capture operation may not work.

See **Section 12.0 "Timer1/3/5 Module with Gate Control"** for more information on configuring the 16-bit Timers.

14.1.3 SOFTWARE INTERRUPT MODE

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE interrupt enable bit of the PIE1, PIE2 or PIE4 register clear to avoid false interrupts. Additionally, the user should clear the CCPxIF interrupt flag bit of the PIR1, PIR2 or PIR4 register following any change in Operating mode.

Note:	Clocking the 16-bit Timer resource from the system clock (FOSC) should not be used in Capture mode. In order for Capture mode to recognize the trigger event on the CCPx pin, the Timer resource must be clocked from the instruction clock (FOSC/4) or from an external clock source.

14.4 PWM (Enhanced Mode)

The enhanced PWM function described in this section is available for CCP modules ECCP1, ECCP2 and ECCP3, with any differences between modules noted.

The enhanced PWM mode generates a Pulse-Width Modulation (PWM) signal on up to four different output pins with up to ten bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

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

The ECCP modules have the following additional PWM registers which control Auto-shutdown, Auto-restart, Dead-band Delay and PWM Steering modes:

- ECCPxAS registers
- PSTRxCON registers
- PWMxCON registers

The enhanced PWM module can generate the following five PWM Output modes:

- Single PWM
- Half-Bridge PWM
- Full-Bridge PWM, Forward mode
- Full-Bridge PWM, Reverse mode
- Single PWM with PWM Steering mode

To select an Enhanced PWM Output mode, the PxM<1:0> bits of the CCPxCON register must be configured appropriately.

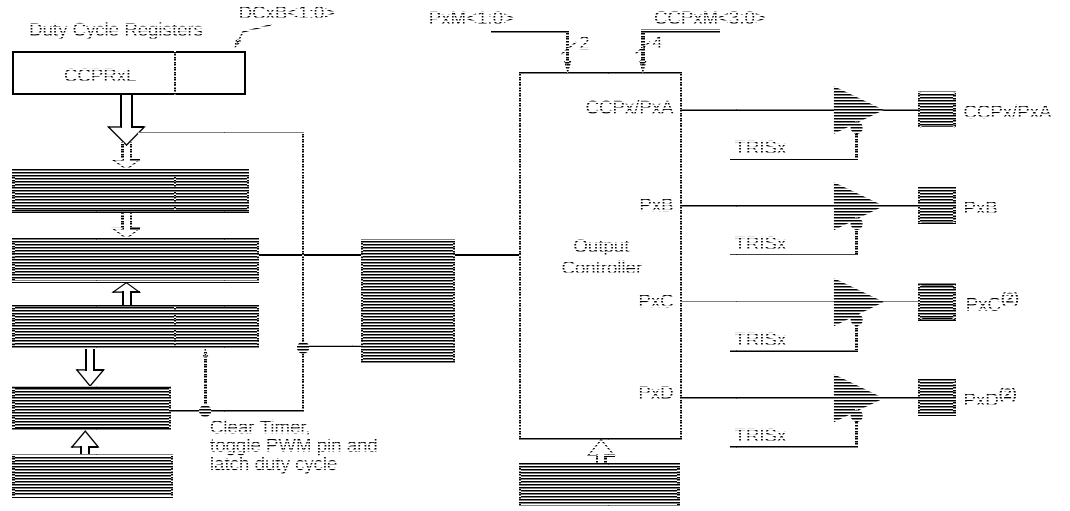
The PWM outputs are multiplexed with I/O pins and are designated PxA, PxB, PxC and PxD. The polarity of the PWM pins is configurable and is selected by setting the CCPxM bits in the CCPxCON register appropriately.

Figure 14-5 shows an example of a simplified block diagram of the Enhanced PWM module.

Table 14-12 shows the pin assignments for various Enhanced PWM modes.

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.
3:	Any pin not used in the enhanced PWM mode is available for alternate pin functions, if applicable.
4:	To prevent the generation of an incomplete waveform when the PWM is first enabled, the ECCP module waits until the start of a new PWM period before generating a PWM signal.

FIGURE 14-5: EXAMPLE SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODE



Note 1: The 8-bit timer TMRx register is concatenated with the 2-bit internal Q clock, or 2 bits of the prescaler to create the 10-bit time base.
2: PxC and PxD are not available on half-bridge ECCP modules.

15.5.4 SLAVE MODE 10-BIT ADDRESS RECEPTION

This section describes a standard sequence of events for the MSSPx module configured as an I^2C slave in 10-bit Addressing mode (Figure 15-20) and is used as a visual reference for this description.

This is a step by step process of what must be done by slave software to accomplish I^2C communication.

1. Bus starts Idle.
2. Master sends Start condition; S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Master sends matching high address with R/$\overline{W}$ bit clear; UA bit of the SSPxSTAT register is set.
4. Slave sends $\overline{ACK}$ and SSPxIF is set.
5. Software clears the SSPxIF bit.
6. Software reads received address from SSPxBUF clearing the BF flag.
7. Slave loads low address into SSPxADD, releasing SCLx.
8. Master sends matching low address byte to the slave; UA bit is set.

Note:	Updates to the SSPxADD register are not allowed until after the $\overline{ACK}$ sequence.

9. Slave sends $\overline{ACK}$ and SSPxIF is set.

Note:	If the low address does not match, SSPxIF and UA are still set so that the slave software can set SSPxADD back to the high address. BF is not set because there is no match. CKP is unaffected.

10. Slave clears SSPxIF.
11. Slave reads the received matching address from SSPxBUF clearing BF.
12. Slave loads high address into SSPxADD.
13. Master clocks a data byte to the slave and clocks out the slaves $\overline{ACK}$ on the 9th SCLx pulse; SSPxIF is set.
14. If SEN bit of SSPxCON2 is set, CKP is cleared by hardware and the clock is stretched.
15. Slave clears SSPxIF.
16. Slave reads the received byte from SSPxBUF clearing BF.
17. If SEN is set the slave sets CKP to release the SCLx.
18. Steps 13-17 repeat for each received byte.
19. Master sends Stop to end the transmission.

15.5.5 10-BIT ADDRESSING WITH ADDRESS OR DATA HOLD

Reception using 10-bit addressing with AHEN or DHEN set is the same as with 7-bit modes. The only difference is the need to update the SSPxADD register using the UA bit. All functionality, specifically when the CKP bit is cleared and SCLx line is held low are the same. Figure 15-21 can be used as a reference of a slave in 10-bit addressing with AHEN set.

Figure 15-22 shows a standard waveform for a slave transmitter in 10-bit Addressing mode.

REGISTER 15-5: SSPxCON3: SSPx CONTROL REGISTER 3 (CONTINUED)

bit 0 **DHEN:** Data Hold Enable bit (I^2C Slave mode only)
1 = Following the 8th falling edge of SCLx for a received data byte; slave hardware clears the CKP bit of the SSPxCON1 register and SCLx 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. SSPxOV is still set when a new byte is received and BF = 1, but hardware continues to write the most recent byte to SSPxBUF.
2: This bit has no effect in Slave modes for which Start and Stop condition detection is explicitly listed as enabled.
3: The ACKTIM Status bit is active only when the AHEN bit or DHEN bit is set.

REGISTER 15-6: SSPxMSK: SSPx MASK REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0
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 7-1 **MSK<7:1>:** Mask bits
1 = The received address bit n is compared to SSPxADD<n> to detect I^2C address match
0 = The received address bit n is not used to detect I^2C address match

bit 0 **MSK<0>:** Mask bit for I^2C Slave mode, 10-bit Address
I^2C Slave mode, 10-bit address (SSPxM<3:0> = 0111 or 1111):
1 = The received address bit 0 is compared to SSPxADD<0> to detect I^2C address match
0 = The received address bit 0 is not used to detect I^2C address match
I^2C Slave mode, 7-bit address, the bit is ignored

REGISTER 16-2: RCSTAx: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-0
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
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 **SPEN:** Serial Port Enable bit

1 = Serial port enabled (configures RXx/DTx and TXx/CKx pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-bit Receive Enable bit

1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit

Asynchronous mode:

Don't care

Synchronous mode – Master:

1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.

Synchronous mode – Slave

Don't care

bit 4 **CREN:** Continuous Receive Enable bit

Asynchronous mode:

1 = Enables receiver
0 = Disables receiver

Synchronous mode:

1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit

Asynchronous mode 9-bit (RX9 = 1):

1 = Enables address detection, enable interrupt and load the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit

Asynchronous mode 8-bit (RX9 = 0):

Don't care

bit 2 **FERR:** Framing Error bit

1 = Framing error (can be updated by reading RCREGx register and receive next valid byte)
0 = No framing error

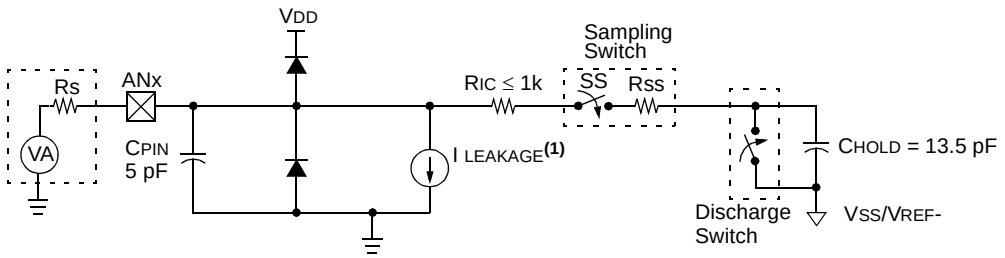
bit 1 **OERR:** Overrun Error bit

1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

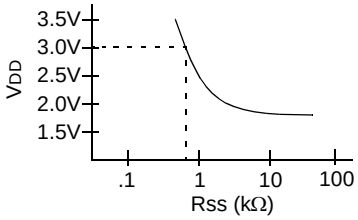
bit 0 **RX9D:** Ninth bit of Received Data

This can be address/data bit or a parity bit and must be calculated by user firmware.

FIGURE 17-5: ANALOG INPUT MODEL



Legend: CPIN = Input Capacitance
I LEAKAGE = Leakage current at the pin due to various junctions
RIC = Interconnect Resistance
SS = Sampling Switch
CHOLD = Sample/Hold Capacitance



Note 1: See **Section 27.0 "Electrical Specifications"**.

FIGURE 17-6: ADC TRANSFER FUNCTION

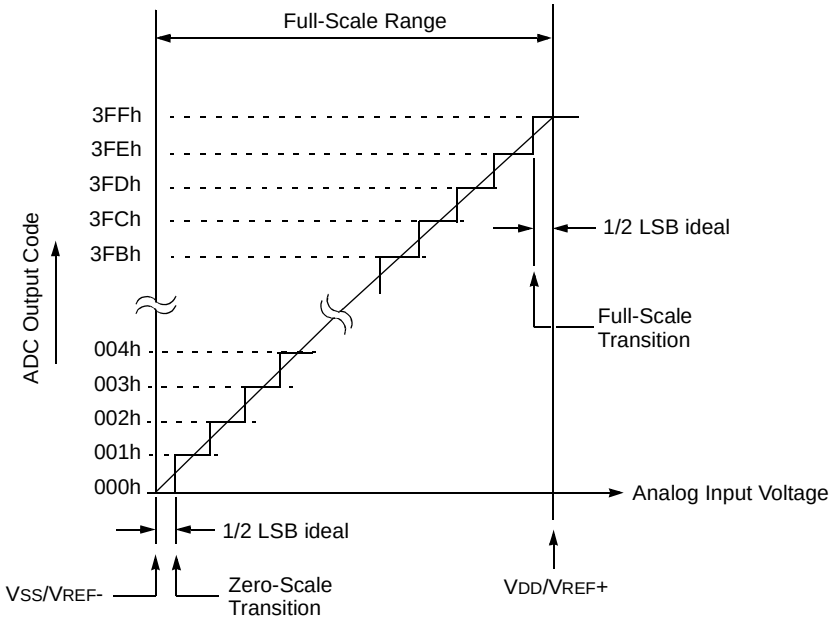


FIGURE 20-1: DIVSRCLK BLOCK DIAGRAM

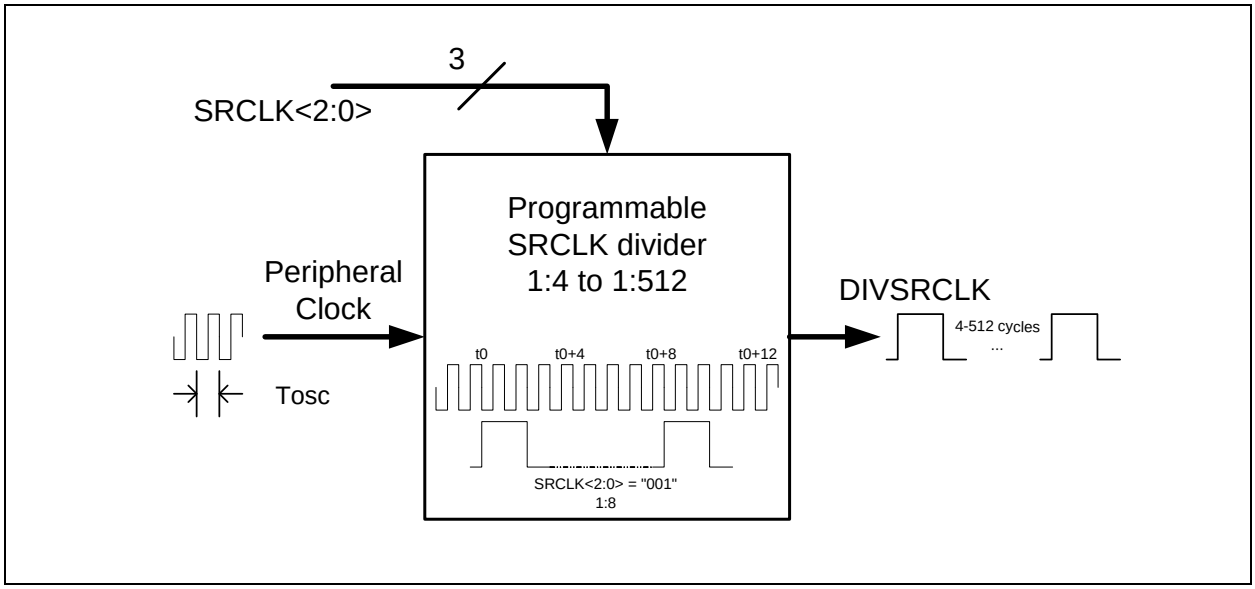
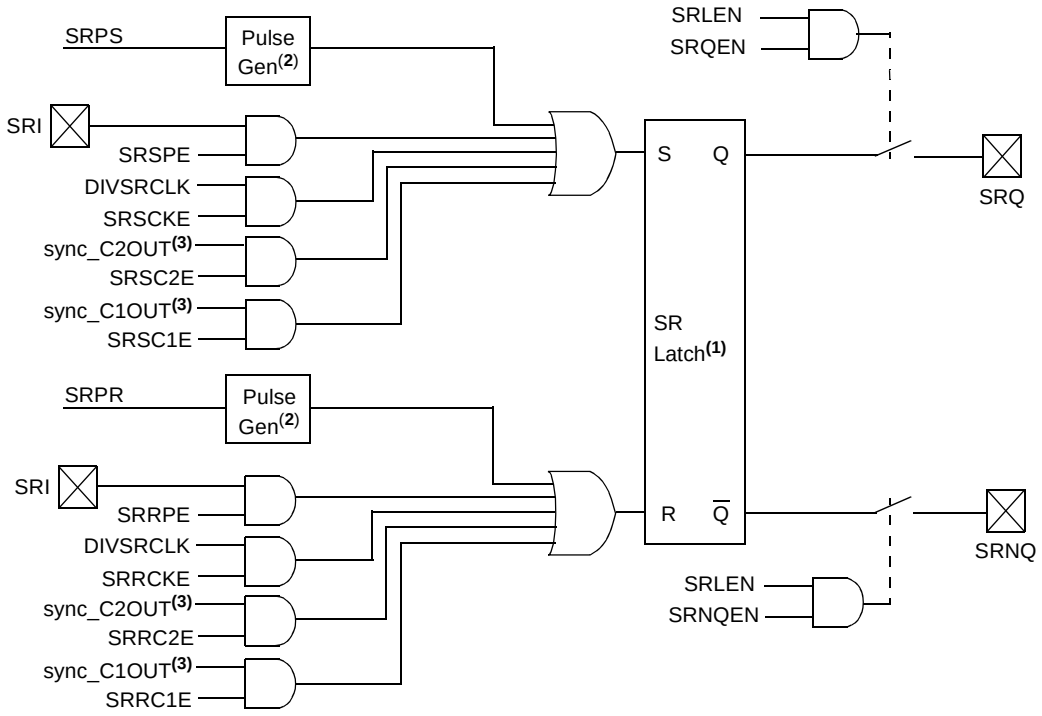


FIGURE 20-2: SR LATCH SIMPLIFIED BLOCK DIAGRAM



Note 1: If R = 1 and S = 1 simultaneously, Q = 0, $\overline{Q}$ = 1
2: Pulse generator causes a pulse width of 2 TOSC clock cycles.
3: Name denotes the connection point at the comparator output.

RCALL **Relative Call**

Syntax: RCALL n

Operands: $-1024 \leq n \leq 1023$

Operation: (PC) + 2 → TOS,
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

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 2-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: RESET

Operands: None

Operation: Reset all registers and flags that are affected by a MCLR Reset.

Status Affected: All

Encoding:

0000	0000	1111	1111

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

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start Reset	No operation	No operation

Example: RESET

After Instruction
Registers = Reset Value
Flags* = Reset Value

TSTFSZ **Test f, skip if 0**

Syntax: TSTFSZ f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: skip if f = 0

Status Affected: None

Encoding:

0110	011a	ffff	ffff

Description: If 'f' = 0, the next instruction fetched during the current instruction execution is discarded and a NOP is executed, making this a 2-cycle instruction.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    TSTFSZ  CNT, 1
NZERO   :
ZERO    :
```
Before Instruction
PC = Address (HERE)
After Instruction
If CNT = 00h,
PC = Address (ZERO)
If CNT ≠ 00h,
PC = Address (NZERO)

XORLW **Exclusive OR literal with W**

Syntax: XORLW k

Operands: $0 \le k \le 255$

Operation: (W) .XOR. k → W

Status Affected: N, Z

Encoding:

0000	1010	kkkk	kkkk

Description: The contents of W are XORed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example:
```
XORLW   0AFh
```
Before Instruction
W = B5h
After Instruction
W = 1Ah

27.6 DC Characteristics: Primary Idle Supply Current, PIC18(L)F2X/4XK22

PIC18LF2X/4XK22		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C					
PIC18F2X/4XK22		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +125°C					
Param No.	**Device Characteristics**	**Typ**	**Max**	**Units**	**Conditions**		
D100	Supply Current (IDD)[1],[2]	0.030	0.050	mA	-40°C to +125°C	VDD = 1.8V	Fosc = 1 MHz (**PRI_IDLE** mode, ECM source)
D101		0.045	0.065	mA	-40°C to +125°C	VDD = 3.0V	
D102		0.06	0.12	mA	-40°C to +125°C	VDD = 2.3V	Fosc = 1 MHz (**PRI_IDLE** mode, ECM source)
D103		0.08	0.15	mA	-40°C to +125°C	VDD = 3.0V	
D104		0.13	0.20	mA	-40°C to +125°C	VDD = 5.0V	
D105		0.45	0.8	mA	-40°C to +125°C	VDD = 1.8V	Fosc = 20 MHz (**PRI_IDLE** mode, ECH source)
D106		0.70	1.0	mA	-40°C to +125°C	VDD = 3.0V	
D107		0.55	0.8	mA	-40°C to +125°C	VDD = 2.3V	Fosc = 20 MHz (**PRI_IDLE** mode, ECH source)
D108		0.75	1.0	mA	-40°C to +125°C	VDD = 3.0V	
D109		0.90	1.2	mA	-40°C to +125°C	VDD = 5.0V	
D110		2.25	3.0	mA	-40°C to +125°C	VDD = 3.0V	Fosc = 64 MHz (**PRI_IDLE** mode, ECH source)
D111		2.25	3.0	mA	-40°C to +125°C	VDD = 3.0V	Fosc = 64 MHz (**PRI_IDLE** mode, ECH source)
D112		2.60	3.5	mA	-40°C to +125°C	VDD = 5.0V	
D113		0.35	0.6	mA	-40°C to +125°C	VDD = 1.8V	Fosc = 4 MHz 16 MHz Internal (**PRI_IDLE** mode, ECM + PLL source)
D114		0.55	0.8	mA	-40°C to +125°C	VDD = 3.0V	
D115		0.45	0.6	mA	-40°C to +125°C	VDD = 2.3V	Fosc = 4 MHz 16 MHz Internal (**PRI_IDLE** mode, ECM + PLL source)
D116		0.60	0.9	mA	-40°C to +125°C	VDD = 3.0V	
D117		0.70	1.0	mA	-40°C to +125°C	VDD = 5.0V	
D118		2.2	3.0	mA	-40°C to +125°C	VDD = 3.0V	Fosc = 16 MHz 64 MHz Internal (**PRI_IDLE** mode, ECH + PLL source)
D119		2.2	3.0	mA	-40°C to +125°C	VDD = 3.0V	Fosc = 16 MHz 64 MHz Internal (**PRI_IDLE** mode, ECH + PLL source)
D120		2.5	3.5	mA	-40°C to +125°C	VDD = 5.0V	

Note 1: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

Test condition: All Peripheral Module Control bits in PMD0, PMD1 and PMD2 set to '1'.

2: The test conditions for all IDD measurements in active operation mode are:

All I/O pins set as outputs driven to Vss;
MCLR = VDD;
OSC1 = external square wave, from rail-to-rail (PRI_RUN and PRI_IDLE only).

27.8 DC Characteristics: Input/Output Characteristics, PIC18(L)F2X/4XK22

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	VIL	**Input Low Voltage**					
		I/O PORT:					
D140		with TTL buffer	—	—	0.8	V	4.5V ≤ VDD ≤ 5.5V
D140A			—	—	0.15 VDD	V	1.8V ≤ VDD ≤ 4.5V
D141		with Schmitt Trigger buffer	—	—	0.2 VDD	V	2.0V ≤ VDD ≤ 5.5V
		with I2C levels	—	—	0.3 VDD	V	
		with SMBus levels	—	—	0.8	V	2.7V ≤ VDD ≤ 5.5V
D142		$\overline{MCLR}$, OSC1 (RC mode)(1)	—	—	0.2 VDD	V	
D142A		OSC1 (HS mode)	—	—	0.3 VDD	V	
	VIH	**Input High Voltage**					
		I/O ports:		—	—		
D147		with TTL buffer	2.0	—	—	V	4.5V ≤ VDD ≤ 5.5V
D147A			0.25 VDD + 0.8	—	—	V	1.8V ≤ VDD ≤ 4.5V
D148		with Schmitt Trigger buffer	0.8 VDD	—	—	V	2.0V ≤ VDD ≤ 5.5V
		with I2C levels	0.7 VDD	—	—	V	
		with SMBus levels	2.1	—	—	V	2.7V ≤ VDD ≤ 5.5V
D149		$\overline{MCLR}$	0.8 VDD	—	—	V	
D150A		OSC1 (HS mode)	0.7 VDD	—	—	V	
D150B		OSC1 (RC mode)(1)	0.9 VDD	—	—	V	
	IIL	**Input Leakage I/O and $\overline{MCLR}$(2),(3)**					VSS ≤ VPIN ≤ VDD, Pin at high-impedance
D155		I/O ports and $\overline{MCLR}$	— — — —	0.1 0.7 4 35	50 100 200 1000	nA nA nA nA	≤ +25°C(4) +60°C +85°C +125°C
	IPU	**Weak Pull-up Current(4)**					
D158	IPURB	PORTB weak pull-up current	25 25	85 130	200 300	µA µA	VDD = 3.3V, VPIN = VSS VDD = 5.0V, VPIN = VSS

Note 1: In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC® device be driven with an external clock while in RC mode.

2: The leakage current on the $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as current sourced by the pin.

4: Parameter is characterized but not tested.

FIGURE 28-17: PIC18LF2X/4XK22 DELTA IPD FVR

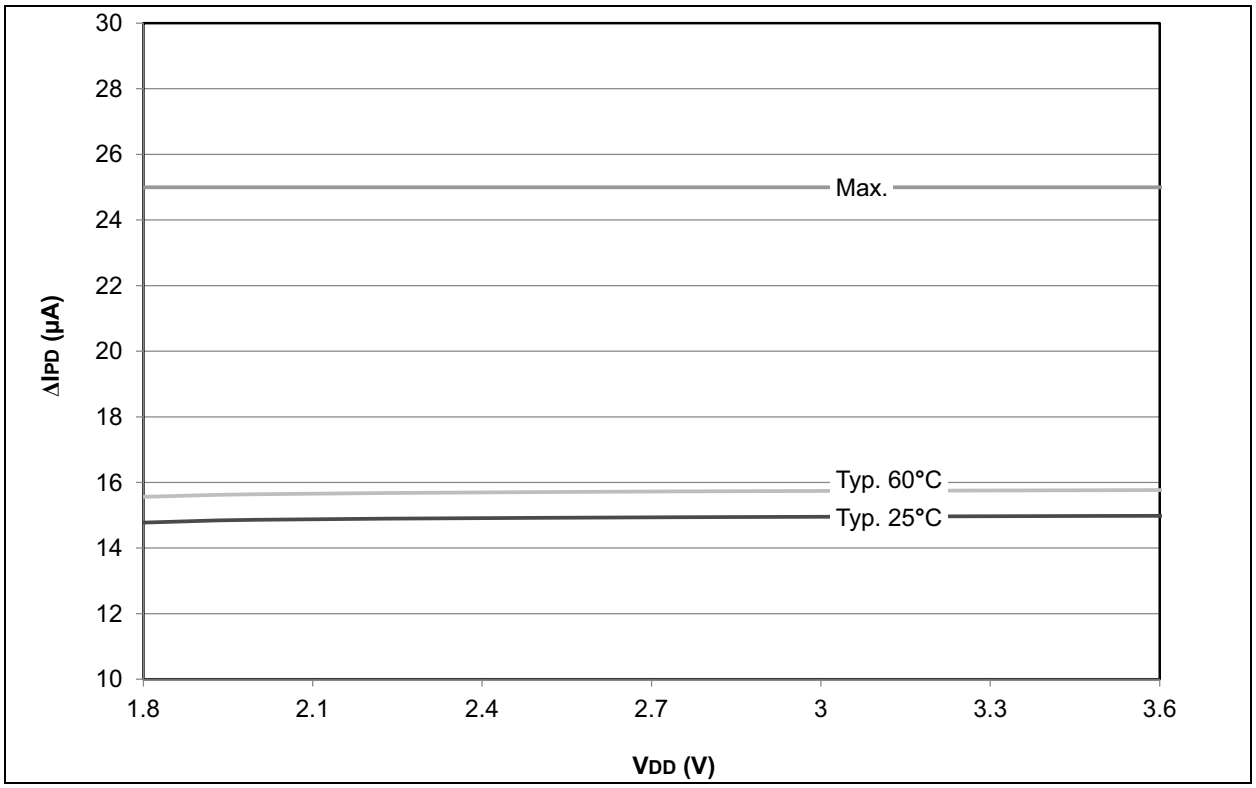
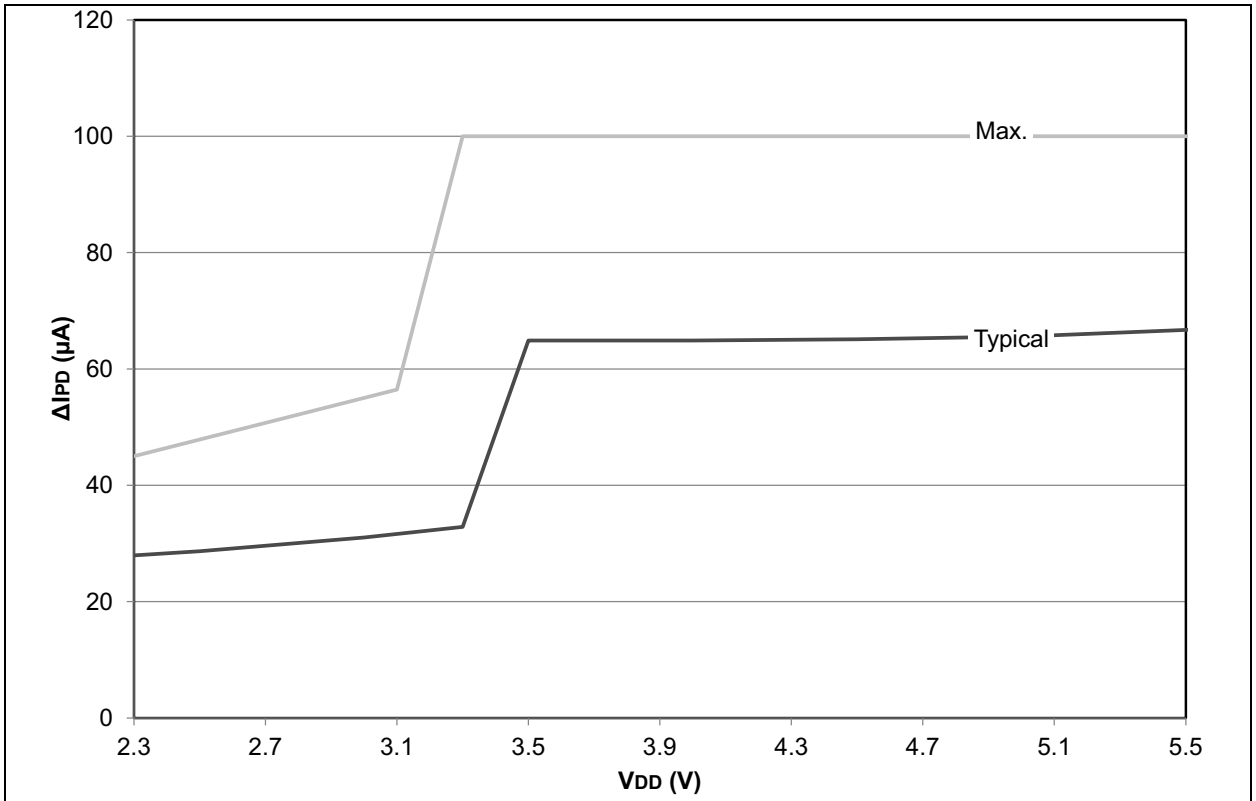


FIGURE 28-18: PIC18F2X/4XK22 DELTA IPD FVR



Note 1: On the PIC18F2X/4XK22, enabling the FVR results in significantly more Sleep current when the part enters Voltage Regulation mode at VDD ~ 3.2V.

FIGURE 28-78: PIC18F2X/4XK22 TYPICAL IDD: SEC_IDLE 32.768 kHz

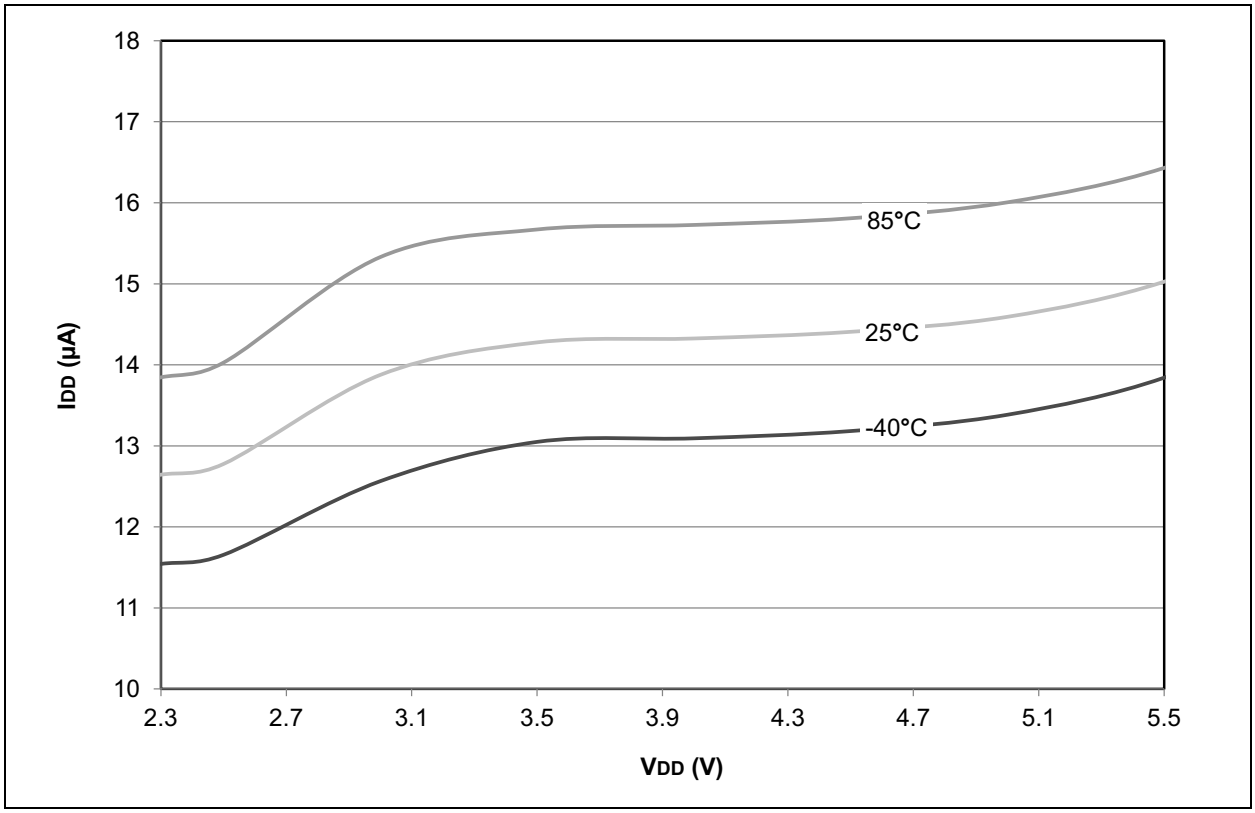
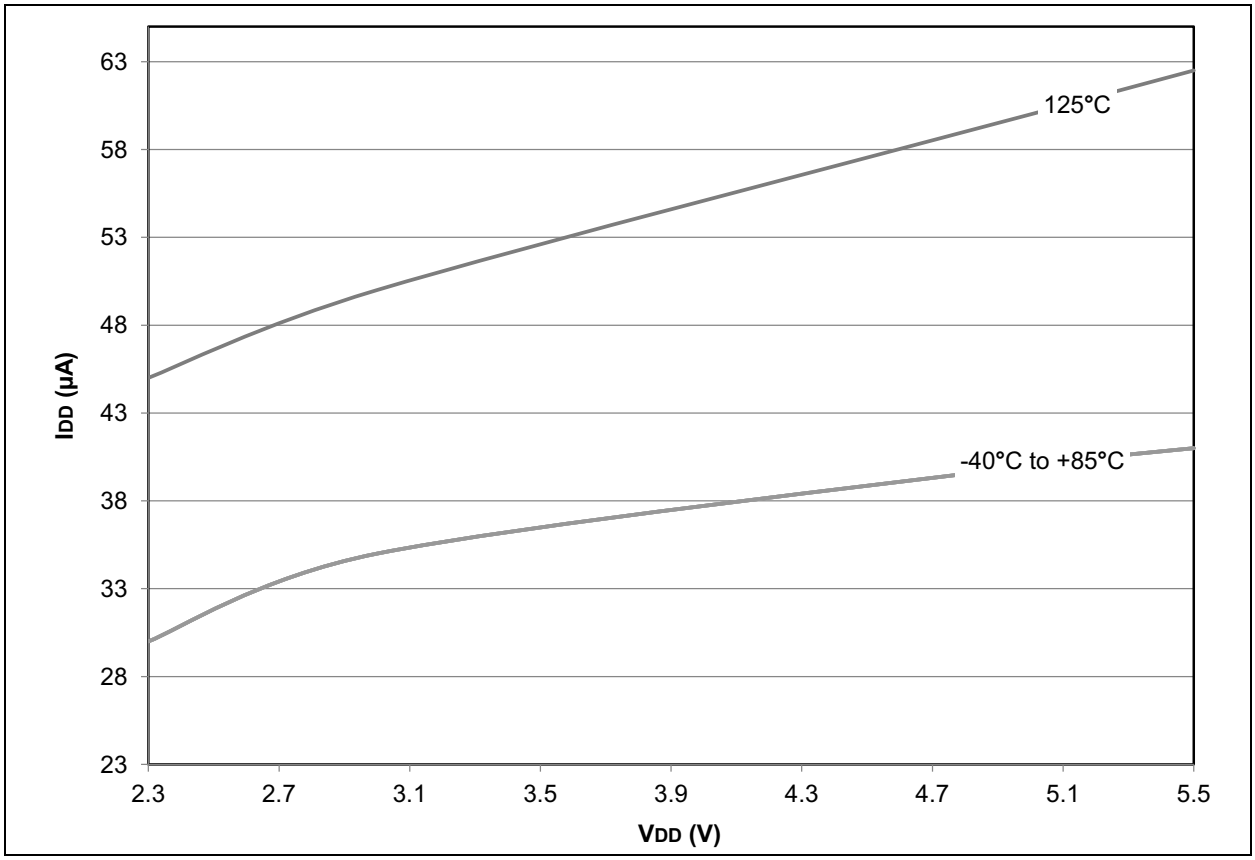
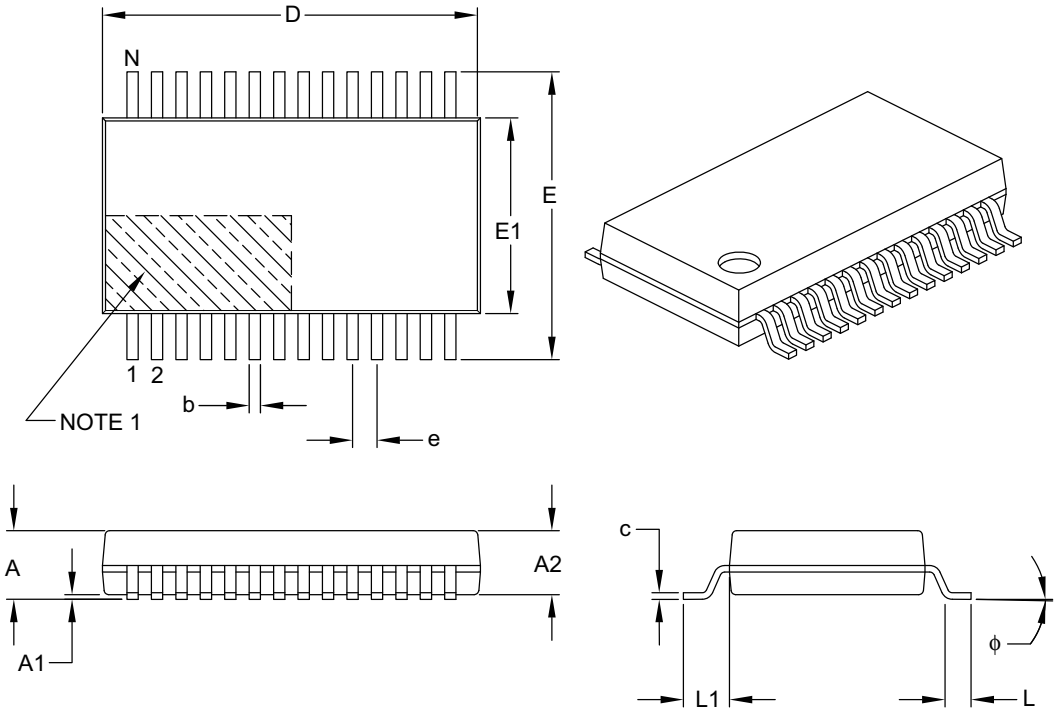


FIGURE 28-79: PIC18F2X/4XK22 MAXIMUM IDD: SEC_IDLE 32.768 kHz



28-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		0.65 BSC	
Overall Height	A	–	–	2.00
Molded Package Thickness	A2	1.65	1.75	1.85
Standoff	A1	0.05	–	–
Overall Width	E	7.40	7.80	8.20
Molded Package Width	E1	5.00	5.30	5.60
Overall Length	D	9.90	10.20	10.50
Foot Length	L	0.55	0.75	0.95
Footprint	L1		1.25 REF	
Lead Thickness	c	0.09	–	0.25
Foot Angle	ϕ	0°	4°	8°
Lead Width	b	0.22	–	0.38

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-073B