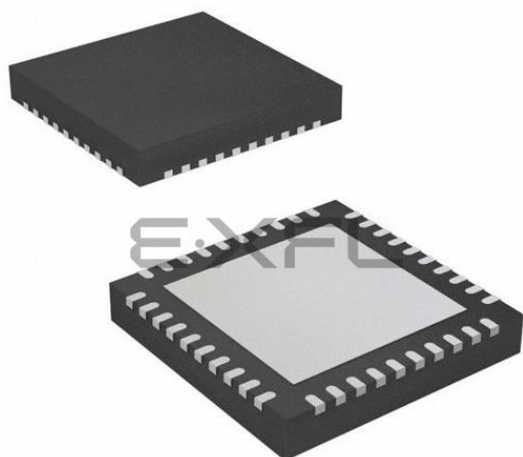




Welcome to [E-XFL.COM](https://www.e-xfl.com)

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	64MHz
Connectivity	I ² C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	35
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 30x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	40-UQFN Exposed Pad
Supplier Device Package	40-UQFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf44k22t-i-mv

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

OSC Mode	OSC1 Pin	OSC2 Pin
RC, INTOSC with CLKOUT	Floating, external resistor should pull high	At logic low (clock/4 output)
RC with IO	Floating, external resistor should pull high	Configured as PORTA, bit 6
INTOSC with IO	Configured as PORTA, bit 7	Configured as PORTA, bit 6
EC with IO	Floating, pulled by external clock	Configured as PORTA, bit 6
EC with CLKOUT	Floating, pulled by external clock	At logic low (clock/4 output)
LP, XT, HS	Feedback inverter disabled at quiescent voltage level	Feedback inverter disabled at quiescent voltage level

Note: See Table 4-2 in **Section 4.0 “Reset”** for time-outs due to Sleep and $\overline{\text{MCLR}}$ Reset.

2.11 Clock Switching

The system clock source can be switched between external and internal clock sources via software using the System Clock Select (SCS<1:0>) bits of the OSCCON register.

PIC18(L)F2X/4XK22 devices contain circuitry to prevent clock “glitches” when switching between clock sources. A short pause in the device clock occurs during the clock switch. The length of this pause is the sum of two cycles of the old clock source and three to four cycles of the new clock source. This formula assumes that the new clock source is stable.

Clock transitions are discussed in greater detail in **Section 3.1.2 “Entering Power-Managed Modes”**.

2.11.1 SYSTEM CLOCK SELECT (SCS<1:0>) BITS

The System Clock Select (SCS<1:0>) bits of the OSCCON register select the system clock source that is used for the CPU and peripherals.

- When SCS<1:0> = 00, the system clock source is determined by configuration of the FOSC<3:0> bits in the CONFIG1H Configuration register.
- When SCS<1:0> = 10, the system clock source is chosen by the internal oscillator frequency selected by the INTSRC bit of the OSCTUNE register, the MFIOSEL bit of the OSCCON2 register and the IRCF<2:0> bits of the OSCCON register.
- When SCS<1:0> = 01, the system clock source is the 32.768 kHz secondary oscillator shared with Timer1, Timer3 and Timer5.

After a Reset, the SCS<1:0> bits of the OSCCON register are always cleared.

Note: Any automatic clock switch, which may occur from Two-Speed Start-up or Fail-Safe Clock Monitor, does not update the SCS<1:0> bits of the OSCCON register. The user can monitor the SOSCRUN, MFIOFS and LFIOFS bits of the OSCCON2 register, and the HFIOFS and OSTS bits of the OSCCON register to determine the current system clock source.

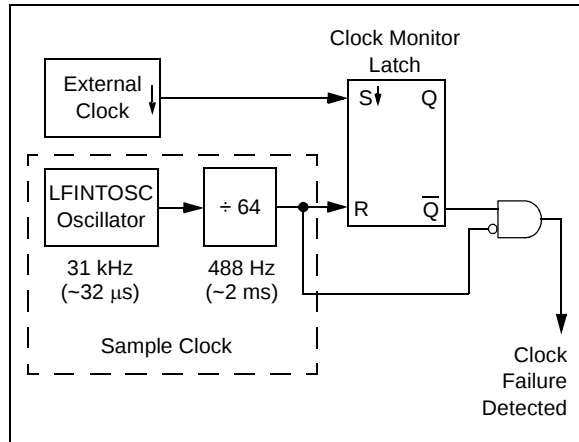
2.11.2 OSCILLATOR START-UP TIME-OUT STATUS (OSTS) BIT

The Oscillator Start-up Time-out Status (OSTS) bit of the OSCCON register indicates whether the system clock is running from the external clock source, as defined by the FOSC<3:0> bits in the CONFIG1H Configuration register, or from the internal clock source. In particular, when the primary oscillator is the source of the primary clock, OSTS indicates that the Oscillator Start-up Timer (OST) has timed out for LP, XT or HS modes.

2.13 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the device to continue operating should the external oscillator fail. The FSCM can detect oscillator failure any time after the Oscillator Start-up Timer (OST) has expired. The FSCM is enabled by setting the FCMEN bit in the CONFIG1H Configuration register. The FSCM is applicable to all external oscillator modes (LP, XT, HS, EC, RC and RCIO).

FIGURE 2-10: FSCM BLOCK DIAGRAM



2.13.1 FAIL-SAFE DETECTION

The FSCM module detects a failed oscillator by comparing the external oscillator to the FSCM sample clock. The sample clock is generated by dividing the LFINTOSC by 64 (see Figure 2-10). Inside the fail detector block is a latch. The external clock sets the latch on each falling edge of the external clock. The sample clock clears the latch on each rising edge of the sample clock. A failure is detected when an entire half-cycle of the sample clock elapses before the primary clock goes low.

2.13.2 FAIL-SAFE OPERATION

When the external clock fails, the FSCM switches the device clock to an internal clock source and sets the bit flag OSCFIF of the PIR2 register. The OSCFIF flag will generate an interrupt if the OSCFIE bit of the PIE2 register is also set. The device firmware can then take steps to mitigate the problems that may arise from a failed clock. The system clock will continue to be sourced from the internal clock source until the device firmware successfully restarts the external oscillator and switches back to external operation. An automatic transition back to the failed clock source will not occur.

The internal clock source chosen by the FSCM is determined by the IRCF<2:0> bits of the OSCCON register. This allows the internal oscillator to be configured before a failure occurs.

2.13.3 FAIL-SAFE CONDITION CLEARING

The Fail-Safe condition is cleared by either one of the following:

- Any Reset
- By toggling the SCS1 bit of the OSCCON register

Both of these conditions restart the OST. While the OST is running, the device continues to operate from the INTOSC selected in OSCCON. When the OST times out, the Fail-Safe condition is cleared and the device automatically switches over to the external clock source. The Fail-Safe condition need not be cleared before the OSCFIF flag is cleared.

2.13.4 RESET OR WAKE-UP FROM SLEEP

The FSCM is designed to detect an oscillator failure after the Oscillator Start-up Timer (OST) has expired. The OST is used after waking up from Sleep and after any type of Reset. The OST is not used with the EC or RC Clock modes so that the FSCM will be active as soon as the Reset or wake-up has completed.

Note: Due to the wide range of oscillator start-up times, the Fail-Safe circuit is not active during oscillator start-up (i.e., after exiting Reset or Sleep). After an appropriate amount of time, the user should check the OSTS bit of the OSCCON register to verify the oscillator start-up and that the system clock switchover has successfully completed.

Note: When the device is configured for Fail-Safe clock monitoring in either HS, XT, or LS Oscillator modes then the IESO configuration bit should also be set so that the clock will automatically switch from the internal clock to the external oscillator when the OST times out.

4.5 Brown-out Reset (BOR)

PIC18(L)F2X/4XK22 devices implement a BOR circuit that provides the user with a number of configuration and power-saving options. The BOR is controlled by the BORV<1:0> and BOREN<1:0> bits of the CONFIG2L Configuration register. There are a total of four BOR configurations which are summarized in Table 4-1.

The BOR threshold is set by the BORV<1:0> bits. If BOR is enabled (any values of BOREN<1:0>, except '00'), any drop of VDD below VBOR for greater than TBOR will reset the device. A Reset may or may not occur if VDD falls below VBOR for less than TBOR. The chip will remain in Brown-out Reset until VDD rises above VBOR.

If the Power-up Timer is enabled, it will be invoked after VDD rises above VBOR; it then will keep the chip in Reset for an additional time delay, TPWRT. If VDD drops below VBOR while the Power-up Timer is running, the chip will go back into a Brown-out Reset and the Power-up Timer will be initialized. Once VDD rises above VBOR, the Power-up Timer will execute the additional time delay.

BOR and the Power-on Timer (PWRT) are independently configured. Enabling BOR Reset does not automatically enable the PWRT.

The BOR circuit has an output that feeds into the POR circuit and rearms the POR within the operating range of the BOR. This early rearming of the POR ensures that the device will remain in Reset in the event that VDD falls below the operating range of the BOR circuitry.

4.5.1 DETECTING BOR

When BOR is enabled, the $\overline{\text{BOR}}$ bit always resets to '0' on any BOR or POR event. This makes it difficult to determine if a BOR event has occurred just by reading the state of $\overline{\text{BOR}}$ alone. A more reliable method is to simultaneously check the state of both $\overline{\text{POR}}$ and $\overline{\text{BOR}}$. This assumes that the POR and $\overline{\text{BOR}}$ bits are reset to '1' by software immediately after any POR event. If BOR is '0' while POR is '1', it can be reliably assumed that a BOR event has occurred.

4.5.2 SOFTWARE ENABLED BOR

When BOREN<1:0> = 01, the BOR can be enabled or disabled by the user in software. This is done with the SBOREN control bit of the RCON register. Setting SBOREN enables the BOR to function as previously described. Clearing SBOREN disables the BOR entirely. The SBOREN bit operates only in this mode; otherwise it is read as '0'.

Placing the BOR under software control gives the user the additional flexibility of tailoring the application to the environment without having to reprogram the device to change BOR configuration. It also allows the user to tailor device power consumption in software by eliminating the incremental current that the BOR consumes. While the BOR current is typically very small, it may have some impact in low-power applications.

Note: Even when BOR is under software control, the BOR Reset voltage level is still set by the BORV<1:0> Configuration bits. It cannot be changed by software.

4.5.3 DISABLING BOR IN SLEEP MODE

When BOREN<1:0> = 10, the BOR remains under hardware control and operates as previously described. Whenever the device enters Sleep mode, however, the BOR is automatically disabled. When the device returns to any other operating mode, BOR is automatically re-enabled.

This mode allows for applications to recover from brown-out situations, while actively executing code, when the device requires BOR protection the most. At the same time, it saves additional power in Sleep mode by eliminating the small incremental BOR current.

4.5.4 MINIMUM BOR ENABLE TIME

Enabling the BOR also enables the Fixed Voltage Reference (FVR) when no other peripheral requiring the FVR is active. The BOR becomes active only after the FVR stabilizes. Therefore, to ensure BOR protection, the FVR settling time must be considered when enabling the BOR in software or when the BOR is automatically enabled after waking from Sleep. If the BOR is disabled, in software or by reentering Sleep before the FVR stabilizes, the BOR circuit will not sense a BOR condition. The FVRST bit of the VREFCON0 register can be used to determine FVR stability.

6.5 Erasing Flash Program Memory

The minimum erase block is 32 words or 64 bytes. Only through the use of an external programmer, or through ICSP™ control, can larger blocks of program memory be bulk erased. Word erase in the Flash array is not supported.

When initiating an erase sequence from the microcontroller itself, a block of 64 bytes of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:6> point to the block being erased. The TBLPTR<5:0> bits are ignored.

The EECON1 register commands the erase operation. The EEPGD bit must be set to point to the Flash program memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation.

The write initiate sequence for EECON2, shown as steps 4 through 6 in **Section 6.5.1 “Flash Program Memory Erase Sequence”**, is used to guard against accidental writes. This is sometimes referred to as a long write.

A long write is necessary for erasing the internal Flash. Instruction execution is halted during the long write cycle. The long write is terminated by the internal programming timer.

6.5.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

The sequence of events for erasing a block of internal program memory is:

1. Load Table Pointer register with address of block being erased.
2. Set the EECON1 register for the erase operation:
 - set EEPGD bit to point to program memory;
 - clear the CFGS bit to access program memory;
 - set WREN bit to enable writes;
 - set FREE bit to enable the erase.
3. Disable interrupts.
4. Write 55h to EECON2.
5. Write 0AAh to EECON2.
6. Set the WR bit. This will begin the block erase cycle.
7. The CPU will stall for duration of the erase (about 2 ms using internal timer).
8. Re-enable interrupts.

EXAMPLE 6-2: ERASING A FLASH PROGRAM MEMORY 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	
ERASE_BLOCK	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 block Erase operation
	BCF	INTCON, GIE	; disable interrupts
Required Sequence	MOVLW	55h	
	MOVWF	EECON2	; write 55h
	MOVLW	0AAh	
	MOVWF	EECON2	; write 0AAh
	BSF	EECON1, WR	; start erase (CPU stall)
	BSF	INTCON, GIE	; re-enable interrupts

PIC18(L)F2X/4XK22

REGISTER 9-12: PIE4: PERIPHERAL INTERRUPT ENABLE (FLAG) REGISTER 4

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	CCP5IE	CCP4IE	CCP3IE
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 7-3 **Unimplemented:** Read as '0'

bit 2 **CCP5IE:** CCP5 Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 1 **CCP4IE:** CCP4 Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 0 **CCP3IE:** CCP3 Interrupt Enable bit

1 = Enabled

0 = Disabled

REGISTER 9-13: PIE5: PERIPHERAL INTERRUPT ENABLE (FLAG) REGISTER 5

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	TMR6IE	TMR5IE	TMR4IE
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 7-3 **Unimplemented:** Read as '0'

bit 2 **TMR6IE:** TMR6 to PR6 Match Interrupt Enable bit

1 = Enables the TMR6 to PR6 match interrupt

0 = Disables the TMR6 to PR6 match interrupt

bit 1 **TMR5IE:** TMR5 Overflow Interrupt Enable bit

1 = Enables the TMR5 overflow interrupt

0 = Disables the TMR5 overflow interrupt

bit 0 **TMR4IE:** TMR4 to PR4 Match Interrupt Enable bit

1 = Enables the TMR4 to PR4 match interrupt

0 = Disables the TMR4 to PR4 match interrupt

10.6 PORTE Registers

Depending on the particular PIC18(L)F2X/4XK22 device selected, PORTE is implemented in two different ways.

10.6.1 PORTE ON 40/44-PIN DEVICES

For PIC18(L)F2X/4XK22 devices, PORTE is a 4-bit wide port. Three pins (RE0/P3A/CCP3/AN5, RE1/P3B/AN6 and RE2/CCP5/AN7) are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers. When selected as an analog input, these pins will read as '0's.

The corresponding data direction register is TRISE. Setting a TRISE bit (= 1) will make the corresponding PORTE pin an input (i.e., disable the output driver). Clearing a TRISE bit (= 0) will make the corresponding PORTE pin an output (i.e., enable the output driver and put the contents of the output latch on the selected pin).

TRISE controls the direction of the REx pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

The Data Latch register (LATE) is also memory mapped. Read-modify-write operations on the LATE register read and write the latched output value for PORTE.

Note: On a Power-on Reset, RE<2:0> are configured as analog inputs.

The fourth pin of PORTE (MCLR/VPP/RE3) is an input only pin. Its operation is controlled by the MCLRE Configuration bit. When selected as a port pin (MCLRE = 0), it functions as a digital input only pin; as such, it does not have TRIS or LAT bits associated with its operation. Otherwise, it functions as the device's Master Clear input. In either configuration, RE3 also functions as the programming voltage input during programming.

Note: On a Power-on Reset, RE3 is enabled as a digital input only if Master Clear functionality is disabled.

EXAMPLE 10-5: INITIALIZING PORTE

```
CLRF    PORTE    ; Initialize PORTE by
                  ; clearing output
                  ; data latches
CLRF    LATE      ; Alternate method
                  ; to clear output
                  ; data latches
CLRF    ANSELE    ; Configure analog pins
                  ; for digital only
MOVLW   05h       ; Value used to
                  ; initialize data
                  ; direction
MOVWF   TRISE     ; Set RE<0> as input
                  ; RE<1> as output
                  ; RE<2> as input
```

10.6.2 PORTE ON 28-PIN DEVICES

For PIC18F2XK22 devices, PORTE is only available when Master Clear functionality is disabled (MCLR = 0). In these cases, PORTE is a single bit, input only port comprised of RE3 only. The pin operates as previously described.

10.6.3 RE3 WEAK PULL-UP

The port RE3 pin has an individually controlled weak internal pull-up. When set, the WPUE3 (TRISE<7>) bit enables the RE3 pin pull-up. The RBPU bit of the INTCON2 register controls pull-ups on both PORTB and PORTE. When RBPU = 0, the weak pull-ups become active on all pins which have the WPUE3 or WPUBx bits set. When set, the RBPU bit disables all weak pull-ups. The pull-ups are disabled on a Power-on Reset. When the RE3 port pin is configured as MCLR, (CONFIG3H<7>, MCLRE=1 and CONFIG4L<2>, LVP=0), or configured for Low Voltage Programming, (MCLRE=x and LVP=1), the pull-up is always enabled and the WPUE3 bit has no effect.

10.6.4 PORTE OUTPUT PRIORITY

Each PORTE pin is multiplexed with other functions. The pins, their combined functions and their output priorities are briefly described here. For additional information, refer to the appropriate section in this data sheet.

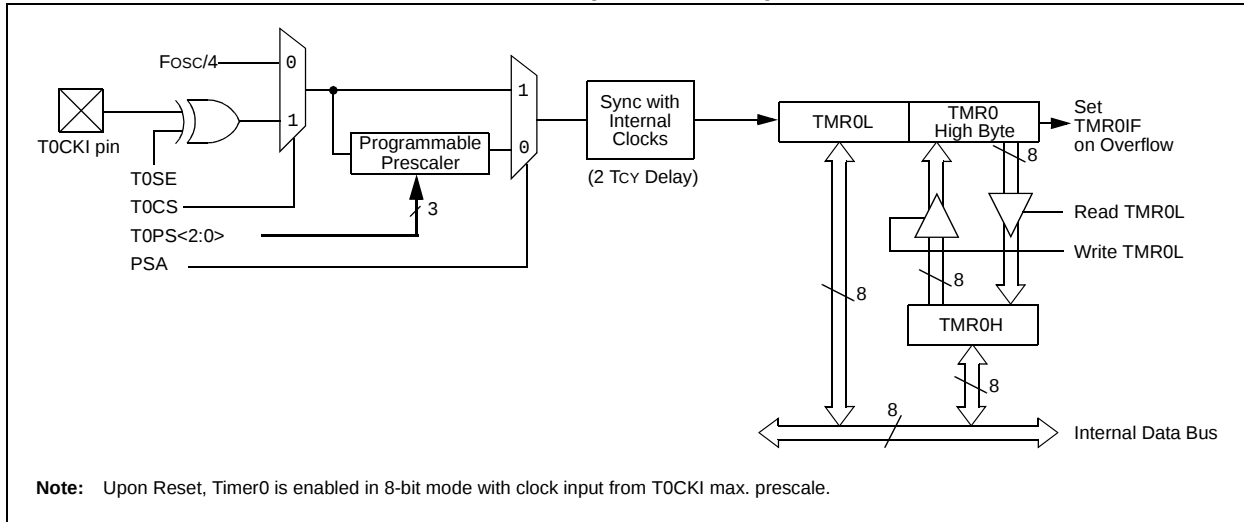
When multiple outputs are enabled, the actual pin control goes to the peripheral with the higher priority. Table 10-4 lists the PORTE pin functions from the highest to the lowest priority.

Analog input functions, such as ADC, comparator and SR latch inputs, are not shown in the priority lists.

These inputs are active when the I/O pin is set for Analog mode using the ANSELx registers. Digital output functions may control the pin when it is in Analog mode with the priority shown below.

PIC18(L)F2X/4XK22

FIGURE 11-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)



11.4 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not directly readable or writable; its value is set by the PSA and T0PS<2:0> bits of the T0CON register which determine the prescaler assignment and prescale ratio.

Clearing the PSA bit assigns the prescaler to the Timer0 module. When the prescaler is assigned, prescale values from 1:2 through 1:256 in integer power-of-2 increments are selectable.

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

Note: Writing to TMR0 when the prescaler is assigned to Timer0 will clear the prescaler count but will not change the prescaler assignment.

11.4.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control and can be changed “on-the-fly” during program execution.

11.5 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or from FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF flag bit. The interrupt can be masked by clearing the TMR0IE bit of the INTCON register. Before re-enabling the interrupt, the TMR0IF bit must be cleared by software in the Interrupt Service Routine.

Since Timer0 is shut down in Sleep mode, the TMR0 interrupt cannot awaken the processor from Sleep.

TABLE 11-1: REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	109
INTCON2	RBPŪ	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	110
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS<2:0>			154
TMR0H	Timer0 Register, High Byte								—
TMR0L	Timer0 Register, Low Byte								—
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	151

Legend: — = unimplemented locations, read as ‘0’. Shaded bits are not used by Timer0.

PIC18(L)F2X/4XK22

15.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP1 AND MSSP2) MODULE

15.1 Master SSPx (MSSPx) Module Overview

The Master Synchronous Serial Port (MSSPx) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be Serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSPx module can operate in one of two modes:

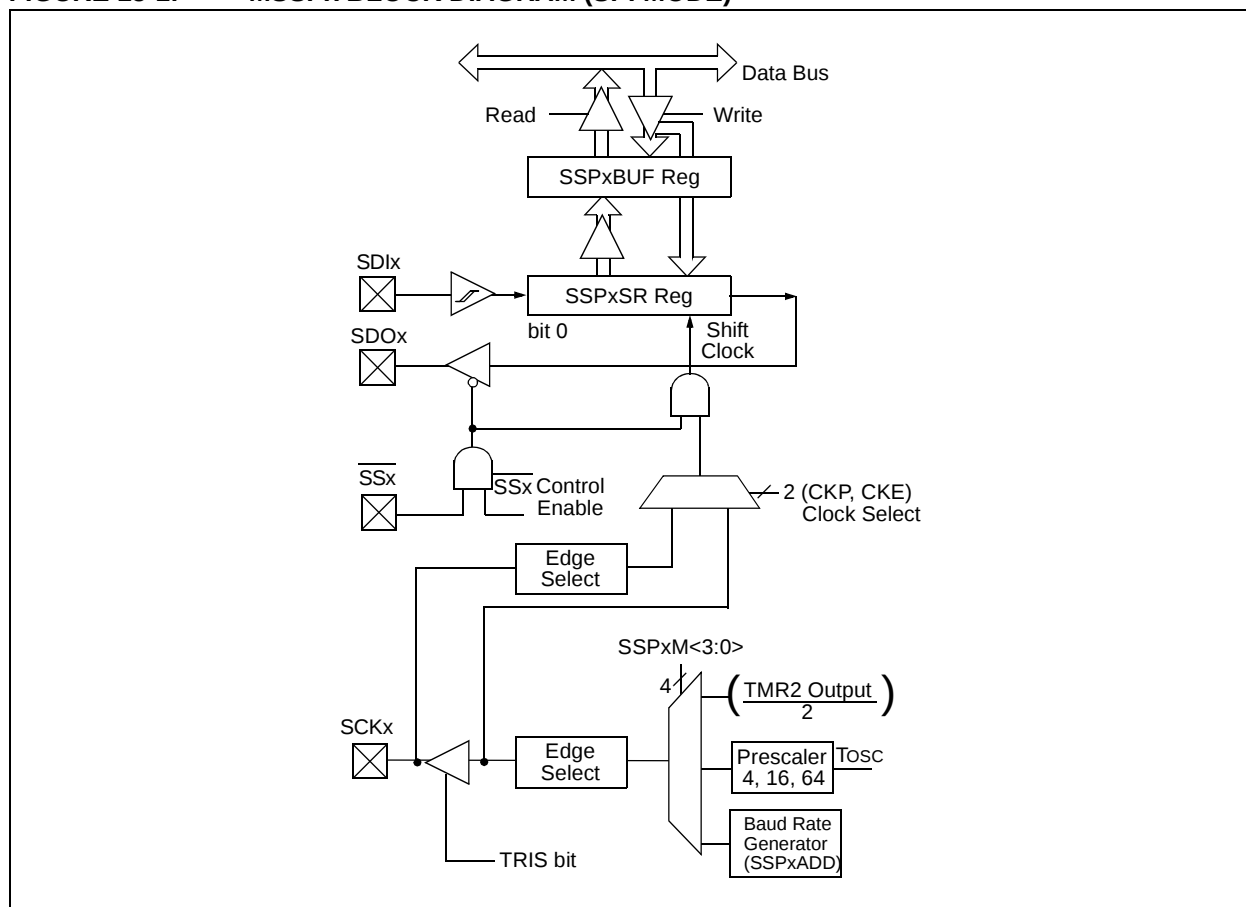
- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I²C)

The SPI interface supports the following modes and features:

- Master mode
- Slave mode
- Clock Parity
- Slave Select Synchronization (Slave mode only)
- Daisy chain connection of slave devices

Figure 15-1 is a block diagram of the SPI interface module.

FIGURE 15-1: MSSPx BLOCK DIAGRAM (SPI MODE)



15.3 I²C Mode Overview

The Inter-Integrated Circuit Bus (I²C) is a multi-master serial data communication bus. Devices communicate in a master/slave environment where the master devices initiate the communication. A slave device is controlled through addressing.

The I²C bus specifies two signal connections:

- Serial Clock (SCLx)
- Serial Data (SDAx)

Figure 15-2 shows the block diagram of the MSSPx module when operating in I²C mode.

Both the SCLx and SDAx connections are bidirectional open-drain lines, each requiring pull-up resistors for the supply voltage. Pulling the line to ground is considered a logical zero and letting the line float is considered a logical one.

Figure 15-11 shows a typical connection between two processors configured as master and slave devices.

The I²C bus can operate with one or more master devices and one or more slave devices.

There are four potential modes of operation for a given device:

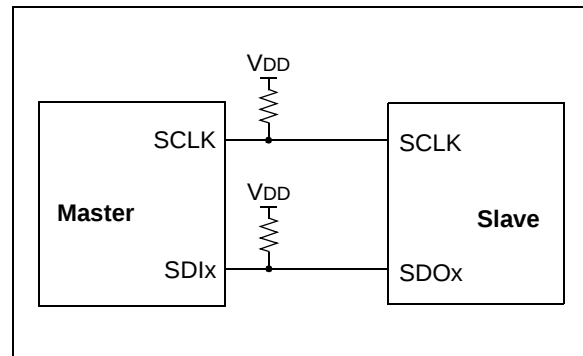
- Master Transmit mode
(master is transmitting data to a slave)
- Master Receive mode
(master is receiving data from a slave)
- Slave Transmit mode
(slave is transmitting data to a master)
- Slave Receive mode
(slave is receiving data from the master)

To begin communication, a master device starts out in Master Transmit mode. The master device sends out a Start bit followed by the address byte of the slave it intends to communicate with. This is followed by a single Read/Write bit, which determines whether the master intends to transmit to or receive data from the slave device.

If the requested slave exists on the bus, it will respond with an Acknowledge bit, otherwise known as an ACK. The master then continues in either Transmit mode or Receive mode and the slave continues in the complement, either in Receive mode or Transmit mode, respectively.

A Start bit is indicated by a high-to-low transition of the SDAx line while the SCLx line is held high. Address and data bytes are sent out, Most Significant bit (MSb) first. The Read/Write bit is sent out as a logical one when the master intends to read data from the slave, and is sent out as a logical zero when it intends to write data to the slave.

FIGURE 15-11: I²C MASTER/SLAVE CONNECTION



The Acknowledge bit ($\overline{\text{ACK}}$) is an active-low signal, which holds the SDAx line low to indicate to the transmitter that the slave device has received the transmitted data and is ready to receive more.

The transition of data bits is always performed while the SCLx line is held low. Transitions that occur while the SCLx line is held high are used to indicate Start and Stop bits.

If the master intends to write to the slave, then it repeatedly sends out a byte of data, with the slave responding after each byte with an $\overline{\text{ACK}}$ bit. In this example, the master device is in Master Transmit mode and the slave is in Slave Receive mode.

If the master intends to read from the slave, then it repeatedly receives a byte of data from the slave, and responds after each byte with an $\overline{\text{ACK}}$ bit. In this example, the master device is in Master Receive mode and the slave is Slave Transmit mode.

On the last byte of data communicated, the master device may end the transmission by sending a Stop bit. If the master device is in Receive mode, it sends the Stop bit in place of the last ACK bit. A Stop bit is indicated by a low-to-high transition of the SDAx line while the SCLx line is held high.

In some cases, the master may want to maintain control of the bus and re-initiate another transmission. If so, the master device may send another Start bit in place of the Stop bit or last ACK bit when it is in receive mode.

The I²C bus specifies three message protocols;

- Single message where a master writes data to a slave.
- Single message where a master reads data from a slave.
- Combined message where a master initiates a minimum of two writes, or two reads, or a combination of writes and reads, to one or more slaves.

15.6.8 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN, of the SSPxCON2 register. When this bit is set, the SCLx pin is pulled low and the contents of the Acknowledge data bit are presented on the SDAx pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCLx pin is deasserted (pulled high). When the SCLx pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG. The SCLx pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSPx module then goes into Idle mode (Figure 15-30).

15.6.8.1 WCOL Status Flag

If the user writes the SSPxBUF when an Acknowledge sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write does not occur).

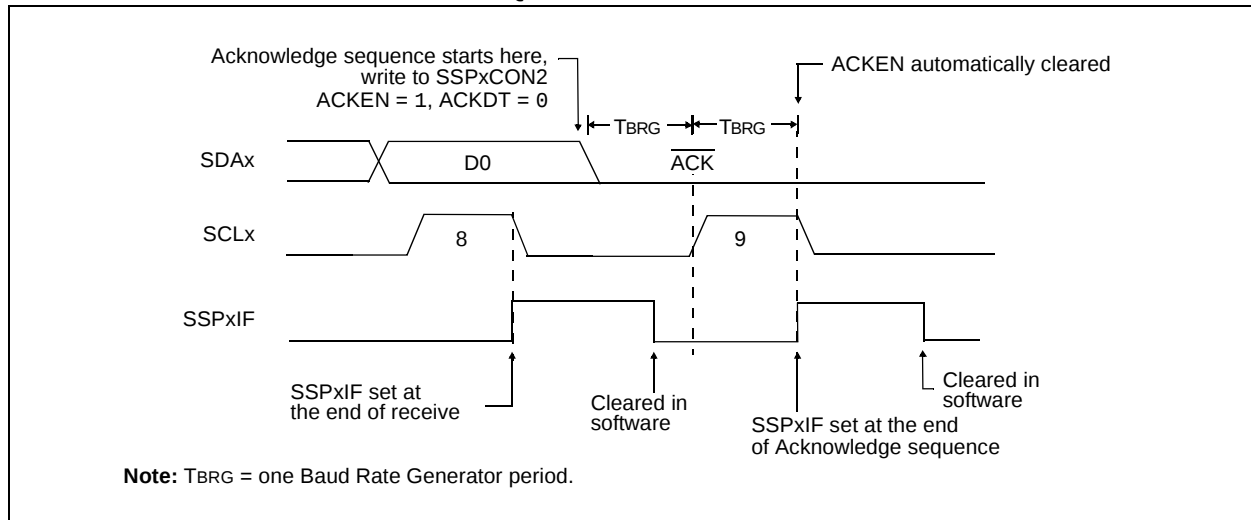
15.6.9 STOP CONDITION TIMING

A Stop bit is asserted on the SDAx pin at the end of a receive/transmit by setting the Stop Sequence Enable bit, PEN, of the SSPxCON2 register. At the end of a receive/transmit, the SCLx line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDAx line low. When the SDAx line is sampled low, the Baud Rate Generator is reloaded and counts down to '0'. When the Baud Rate Generator times out, the SCLx pin will be brought high and one TBRG (Baud Rate Generator rollover count) later, the SDAx pin will be deasserted. When the SDAx pin is sampled high while SCLx is high, the P bit of the SSPxSTAT register is set. A TBRG later, the PEN bit is cleared and the SSPxIF bit is set (Figure 15-31).

15.6.9.1 WCOL Status Flag

If the user writes the SSPxBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

FIGURE 15-30: ACKNOWLEDGE SEQUENCE WAVEFORM



15.6.13 MULTI-MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDAx pin, arbitration takes place when the master outputs a '1' on SDAx, by letting SDAx float high and another master asserts a '0'. When the SCLx pin floats high, data should be stable. If the expected data on SDAx is a '1' and the data sampled on the SDAx pin is '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLxIF, and reset the I²C port to its Idle state (Figure 15-32).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDAx and SCLx lines are deasserted and the SSPxBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I²C bus is free, the user can resume communication by asserting a Start condition.

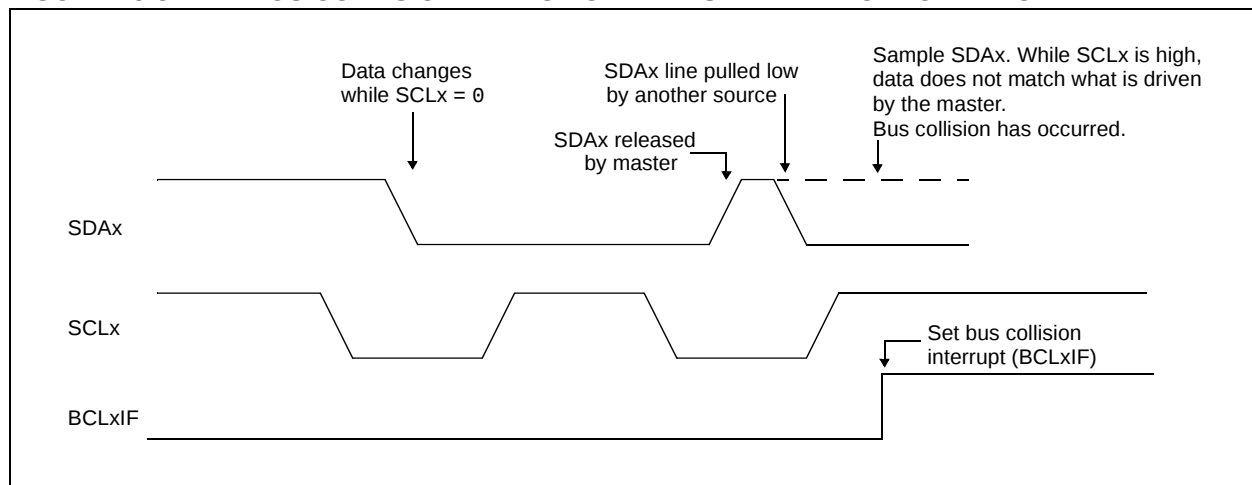
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDAx and SCLx lines are deasserted and the respective control bits in the SSPxCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I²C bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDAx and SCLx pins. If a Stop condition occurs, the SSPxIF bit will be set.

A write to the SSPxBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the I²C bus can be taken when the P bit is set in the SSPxSTAT register, or the bus is Idle and the S and P bits are cleared.

FIGURE 15-32: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE



PIC18(L)F2X/4XK22

16.4 EUSART Baud Rate Generator (BRG)

The Baud Rate Generator (BRG) is an 8-bit or 16-bit timer that is dedicated to the support of both the asynchronous and synchronous EUSART operation. By default, the BRG operates in 8-bit mode. Setting the BRG16 bit of the BAUDCONx register selects 16-bit mode.

The SPBRGHx:SPBRGx register pair determines the period of the free running baud rate timer. In Asynchronous mode the multiplier of the baud rate period is determined by both the BRGH bit of the TXSTAx register and the BRG16 bit of the BAUDCONx register. In Synchronous mode, the BRGH bit is ignored.

Table 16-3 contains the formulas for determining the baud rate. Example 16-1 provides a sample calculation for determining the baud rate and baud rate error.

Typical baud rates and error values for various Asynchronous modes have been computed for your convenience and are shown in Table 16-5. It may be advantageous to use the high baud rate (BRGH = 1), or the 16-bit BRG (BRG16 = 1) to reduce the baud rate error. The 16-bit BRG mode is used to achieve slow baud rates for fast oscillator frequencies.

Writing a new value to the SPBRGHx, SPBRGx register pair causes the BRG timer to be reset (or cleared). This ensures that the BRG does not wait for a timer overflow before outputting the new baud rate.

If the system clock is changed during an active receive operation, a receive error or data loss may result. To avoid this problem, check the status of the RCIDL bit to make sure that the receive operation is Idle before changing the system clock.

EXAMPLE 16-1: CALCULATING BAUD RATE ERROR

For a device with Fosc of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

$$\text{Desired Baud Rate} = \frac{F_{OSC}}{64(SPBRGHx:SPBRGx) + 1}$$

Solving for SPBRGHx:SPBRGx:

$$X = \frac{\frac{F_{OSC}}{\text{Desired Baud Rate}}}{64} - 1$$

$$= \frac{\frac{16000000}{9600}}{64} - 1$$

$$= [25.042] = 25$$

$$\text{Calculated Baud Rate} = \frac{16000000}{64(25 + 1)}$$

$$= 9615$$

$$\text{Error} = \frac{\text{Calc. Baud Rate} - \text{Desired Baud Rate}}{\text{Desired Baud Rate}}$$

$$= \frac{(9615 - 9600)}{9600} = 0.16\%$$

TABLE 16-3: BAUD RATE FORMULAS

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	$F_{OSC}/[64 (n+1)]$
0	0	1	8-bit/Asynchronous	$F_{OSC}/[16 (n+1)]$
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	$F_{OSC}/[4 (n+1)]$
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

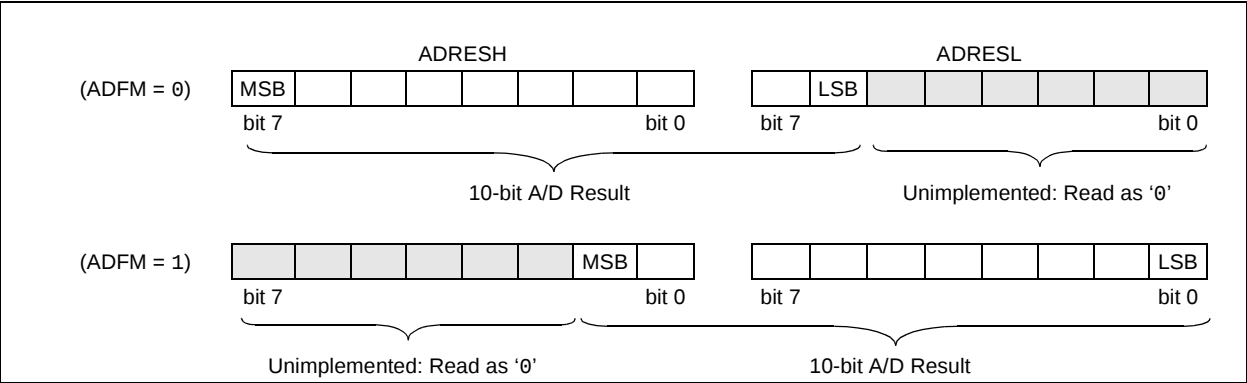
Legend: x = Don't care, n = value of SPBRGHx, SPBRGx register pair.

17.1.7 RESULT FORMATTING

The 10-bit A/D conversion result can be supplied in two formats, left justified or right justified. The ADFM bit of the ADCON2 register controls the output format.

Figure 17-2 shows the two output formats.

FIGURE 17-2: 10-BIT A/D CONVERSION RESULT FORMAT



PIC18(L)F2X/4XK22

17.2 ADC Operation

17.2.1 STARTING A CONVERSION

To enable the ADC module, the ADON bit of the ADCON0 register must be set to a '1'. Setting the GO/DONE bit of the ADCON0 register to a '1' will, depending on the ACQT bits of the ADCON2 register, either immediately start the Analog-to-Digital conversion or start an acquisition delay followed by the Analog-to-Digital conversion.

Figure 17-3 shows the operation of the A/D converter after the GO bit has been set and the ACQT<2:0> bits are cleared. A conversion is started after the following instruction to allow entry into SLEEP mode before the conversion begins.

Figure 17-4 shows the operation of the A/D converter after the GO bit has been set and the ACQT<2:0> bits are set to '010' which selects a 4 TAD acquisition time before the conversion starts.

Note: The GO/DONE bit should not be set in the same instruction that turns on the ADC. Refer to **Section 17.2.10 "A/D Conversion Procedure"**.

FIGURE 17-3: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)

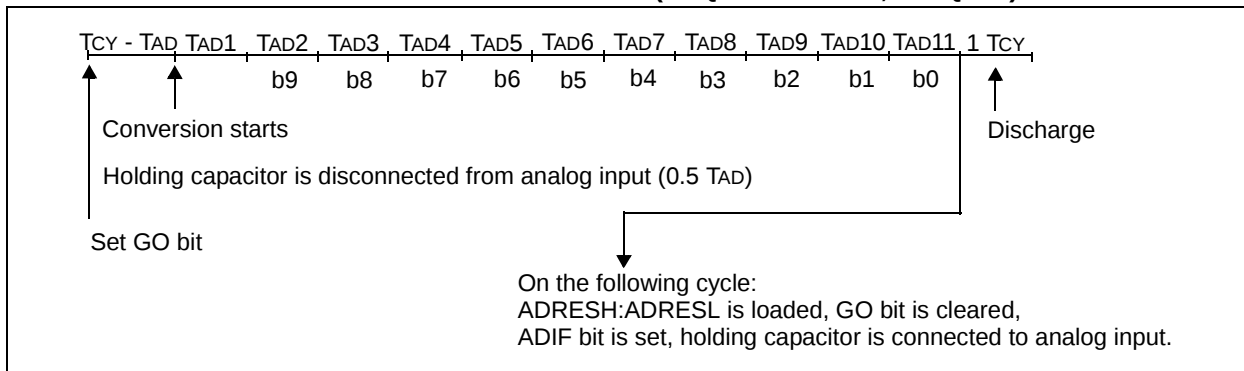
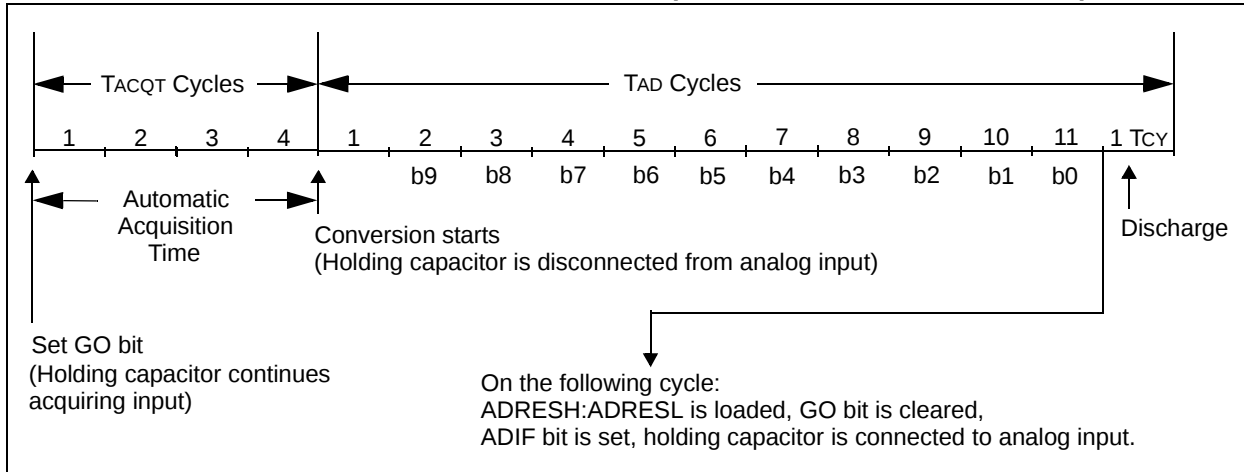


FIGURE 17-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)



PIC18(L)F2X/4XK22

FIGURE 24-4: EXTERNAL BLOCK TABLE READ (EBTRn) DISALLOWED

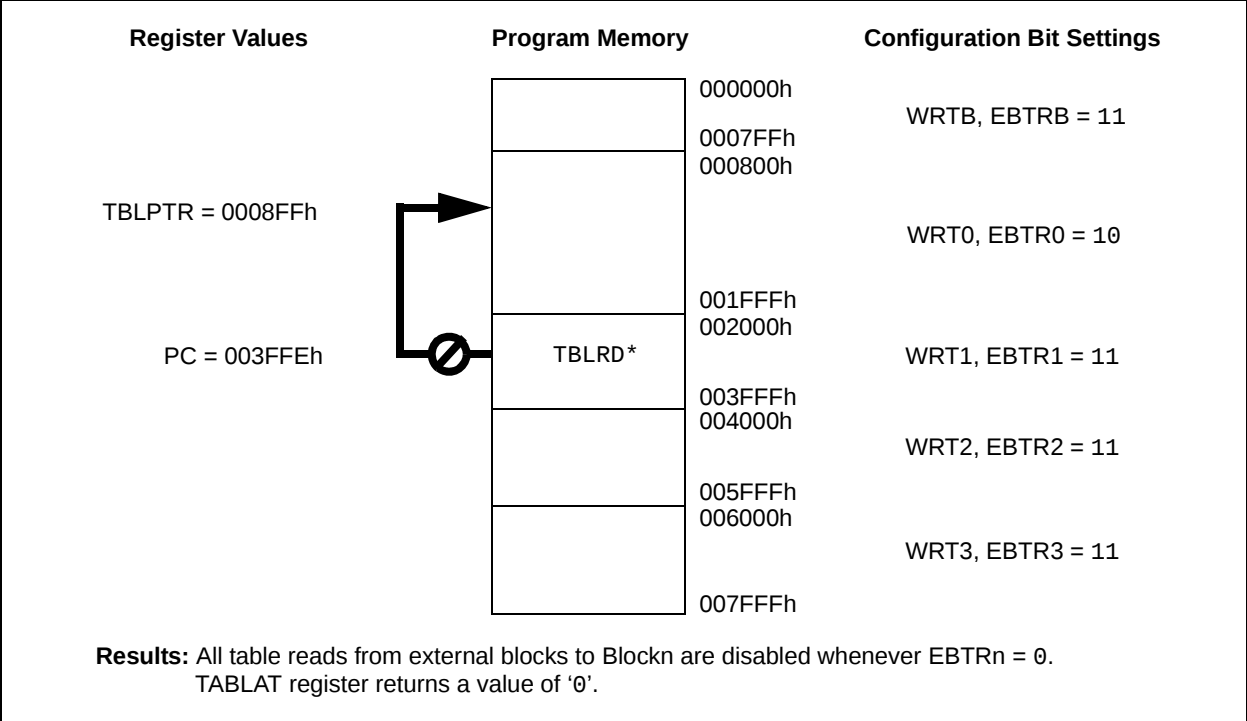
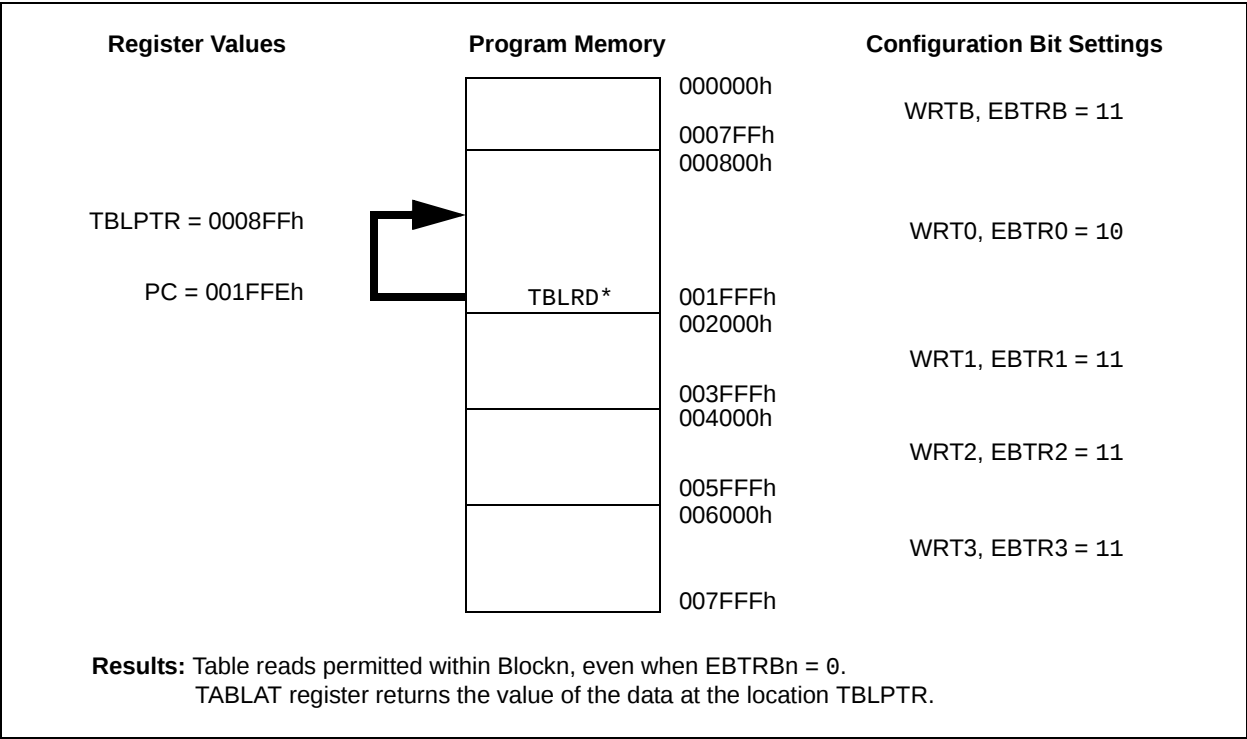


FIGURE 24-5: EXTERNAL BLOCK TABLE READ (EBTRn) ALLOWED



MOVFF Move f to f

Syntax: MOVFF f_s, f_d

Operands: $0 \leq f_s \leq 4095$
 $0 \leq f_d \leq 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:	1100	ffff	ffff	ffff f_s
1st word (source)	1111	ffff	ffff	ffff f_d
2nd word (destin.)				

Description: The contents of source register ' f_s ' are moved to destination register ' f_d '. Location of source ' f_s ' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination ' f_d ' can also be anywhere from 000h to FFFh. Either source or destination can be W (a useful special situation). MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port). The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ' f ' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register ' f ' (dest)

Example: MOVFF REG1, REG2

Before Instruction

REG1 = 33h
 REG2 = 11h

After Instruction

REG1 = 33h
 REG2 = 33h

MOVLB Move literal to low nibble in BSR

Syntax: MOVLW k

Operands: $0 \leq k \leq 255$

Operation: $k \rightarrow \text{BSR}$

Status Affected: None

Encoding:	0000	0001	kkkk	kkkk
-----------	------	------	------	------

Description: The 8-bit literal 'k' is loaded into the Bank Select Register (BSR). The value of BSR<7:4> always remains '0', regardless of the value of $k_7:k_4$.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: MOVLB 5

Before Instruction

BSR Register = 02h

After Instruction

BSR Register = 05h

POP		Pop Top of Return Stack								
Syntax:	POP									
Operands:	None									
Operation:	(TOS) → bit bucket									
Status Affected:	None									
Encoding:	<table border="1"><tr><td>0000</td><td>0000</td><td>0000</td><td>0110</td></tr></table>						0000	0000	0000	0110
0000	0000	0000	0110							
Description:	The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.									
Words:	1									
Cycles:	1									
Q Cycle Activity:										
	Q1	Q2	Q3	Q4						
	Decode	No operation	POP TOS value	No operation						

Example:

POP	NEW
GOTO	

Before Instruction

TOS	=	0031A2h
Stack (1 level down)	=	014332h

After Instruction

TOS	=	014332h
PC	=	NEW

PUSH		Push Top of Return Stack								
Syntax:	PUSH									
Operands:	None									
Operation:	(PC + 2) → TOS									
Status Affected:	None									
Encoding:	<table border="1"><tr><td>0000</td><td>0000</td><td>0000</td><td>0101</td></tr></table>						0000	0000	0000	0101
0000	0000	0000	0101							
Description:	The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.									
Words:	1									
Cycles:	1									
Q Cycle Activity:										
	Q1	Q2	Q3	Q4						
	Decode	PUSH PC + 2 onto return stack	No operation	No operation						

Example:

PUSH

Before Instruction

TOS	=	345Ah
PC	=	0124h

After Instruction

PC	=	0126h
TOS	=	0126h
Stack (1 level down)	=	345Ah

25.2.2 EXTENDED INSTRUCTION SET

ADDFSR Add Literal to FSR

Syntax:	ADDFSR f, k			
Operands:	$0 \leq k \leq 63$ $f \in [0, 1, 2]$			
Operation:	$FSR(f) + k \rightarrow FSR(f)$			
Status Affected:	None			
Encoding:	1110	1000	ffkk	kkkk
Description:	The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'.			
Words:	1			
Cycles:	1			
Q Cycle Activity:				
	Q1	Q2	Q3	Q4
	Decode	Read literal 'k'	Process Data	Write to FSR

Example: ADDFSR 2, 23h

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 0422h

ADDULNK Add Literal to FSR2 and Return

Syntax:	ADDULNK k			
Operands:	$0 \leq k \leq 63$			
Operation:	$FSR2 + k \rightarrow FSR2,$ $(TOS) \rightarrow PC$			
Status Affected:	None			
Encoding:	1110	1000	11kk	kkkk
Description:	The 6-bit literal 'k' is added to the contents of FSR2. A RETURN is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a NOP is performed during the second cycle. This may be thought of as a special case of the ADDFSR instruction, where $f = 3$ (binary '11'); it operates only on FSR2.			
Words:	1			
Cycles:	2			

Q Cycle Activity:

	Q1	Q2	Q3	Q4
	Decode	Read literal 'k'	Process Data	Write to FSR
	No Operation	No Operation	No Operation	No Operation

Example: ADDULNK 23h

Before Instruction
FSR2 = 03FFh
PC = 0100h
After Instruction
FSR2 = 0422h
PC = (TOS)

Note: All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction syntax then becomes: {label} instruction argument(s).

PIC18(L)F2X/4XK22

27.7 DC Characteristics: Secondary Oscillator Supply Current, PIC18(L)F2X/4XK22

PIC18LF2X/4XK22		Standard Operating Conditions (unless otherwise stated)					
		Operating temperature $-40^{\circ}\text{C} \leq T_{\text{A}} \leq +125^{\circ}\text{C}$					
PIC18F2X/4XK22		Standard Operating Conditions (unless otherwise stated)					
		Operating temperature $-40^{\circ}\text{C} \leq T_{\text{A}} \leq +125^{\circ}\text{C}$					
Param No.	Device Characteristics	Typ	Max	Units	Conditions		
D135		0.9	18	μA	-40°C	$V_{\text{DD}} = 1.8\text{V}$	Fosc = 32 kHz (SEC_IDLE mode, SOSC source)
		1.0	18	μA	$+25^{\circ}\text{C}$		
		1.1	—	μA	$+60^{\circ}\text{C}$		
		1.3	20	μA	$+85^{\circ}\text{C}$		
		2.3	22	μA	$+125^{\circ}\text{C}$		
D136		1.3	20	μA	-40°C	$V_{\text{DD}} = 3.0\text{V}$	
		1.4	20	μA	$+25^{\circ}\text{C}$		
		1.5	—	μA	$+60^{\circ}\text{C}$		
		1.8	22	μA	$+85^{\circ}\text{C}$		
		2.9	25	μA	$+125^{\circ}\text{C}$		
D137		12	30	μA	-40°C	$V_{\text{DD}} = 2.3\text{V}$	Fosc = 32 kHz (SEC_IDLE mode, SOSC source)
		13	30	μA	$+25^{\circ}\text{C}$		
		14	30	μA	$+85^{\circ}\text{C}$		
		16	45	μA	$+125^{\circ}\text{C}$		
D138		13	35	μA	-40°C	$V_{\text{DD}} = 3.0\text{V}$	
		14	35	μA	$+25^{\circ}\text{C}$		
		16	35	μA	$+85^{\circ}\text{C}$		
		18	50	μA	$+125^{\circ}\text{C}$		
D139		14	40	μA	-40°C	$V_{\text{DD}} = 5.0\text{V}$	
		15	40	μA	$+25^{\circ}\text{C}$		
		16	40	μA	$+85^{\circ}\text{C}$		
		18	60	μA	$+125^{\circ}\text{C}$		

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 I_{DD} measurements in active operation mode are:

All I/O pins set as outputs driven to V_{SS};

MCLR = V_{DD};

SOSCI / SOSCO = complementary external square wave, from rail-to-rail.

PIC18(L)F2X/4XK22

FIGURE 28-28: PIC18F2X/4XK22 TYPICAL I_{DD} : RC_RUN HF-INTOSC

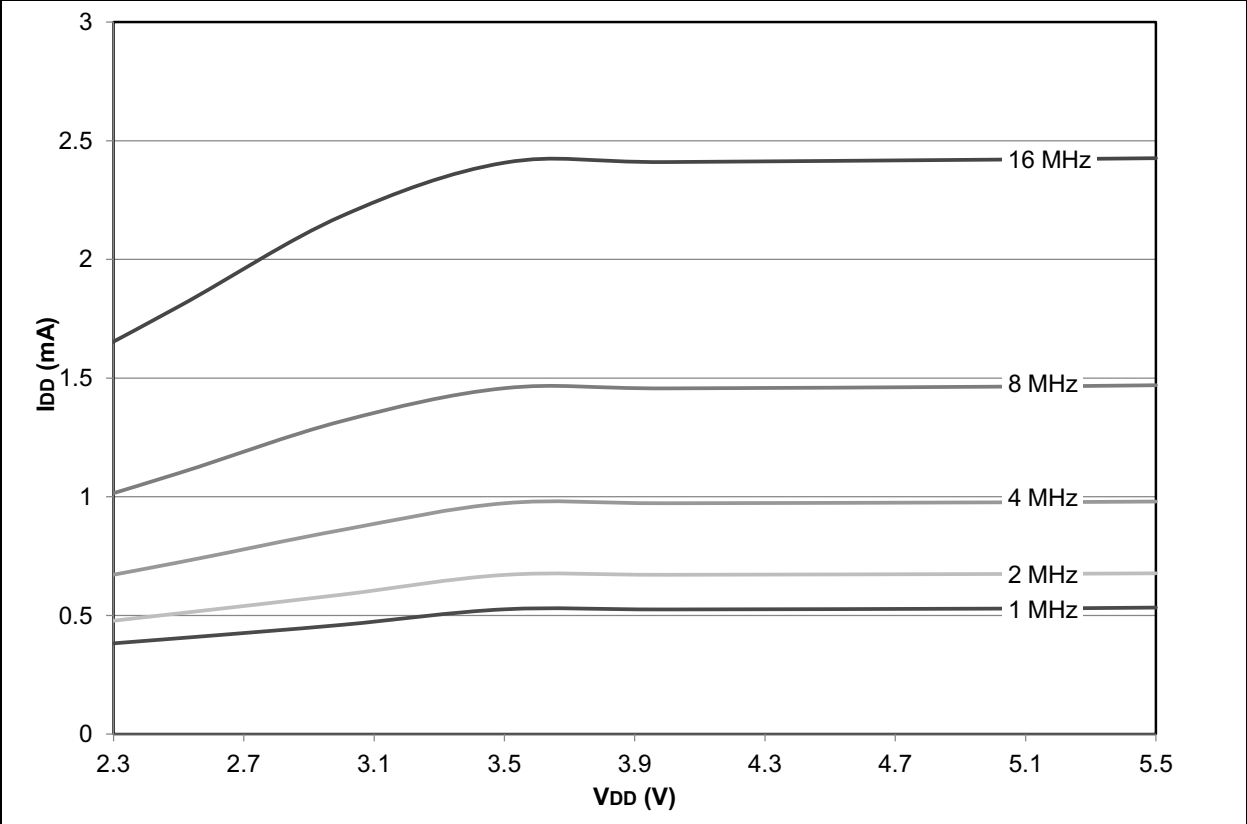


FIGURE 28-29: PIC18F2X/4XK22 MAXIMUM I_{DD} : RC_RUN HF-INTOSC

