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, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	17
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Through Hole
Package / Case	20-DIP (0.300", 7.62mm)
Supplier Device Package	20-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f14k22-e-p

PIC18(L)F1XK22 Family Types

Device	Data Sheet Index	Program Memory		Data Memory		Pins	I/O[1]	10-bit A/D Channels	Comparators	Timers 8-bit/16-bit	ECCP	MSSP	EUSART	SR Latch
		Bytes	Words	SRAM (bytes)	Data EEPROM (bytes)									
PIC18(L)F13K22	(1)	8K	4K	256	256	20	18	12-ch	2	1 / 3	1	1	1	Yes
PIC18(L)F14K22	(1)	16K	8K	512	256	20	18	12-ch	2	1 / 3	1	1	1	Yes

Note 1: One pin is input-only.

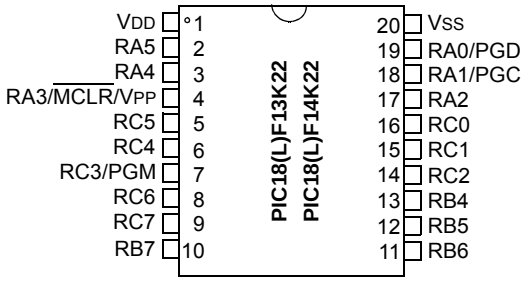
Data Sheet Index: (Unshaded devices are described in this document)

1. DS40001365 PIC18(L)F1XK22 20-Pin Flash Microcontrollers with XLP Technology

Note:	For other small form-factor package availability and marking information, please visit **http://www.microchip.com/packaging** or contact your local sales office.

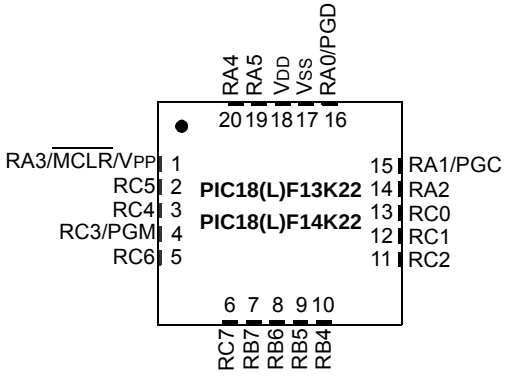
Pin Diagrams

FIGURE 1: 20-PIN PDIP, SSOP, SOIC



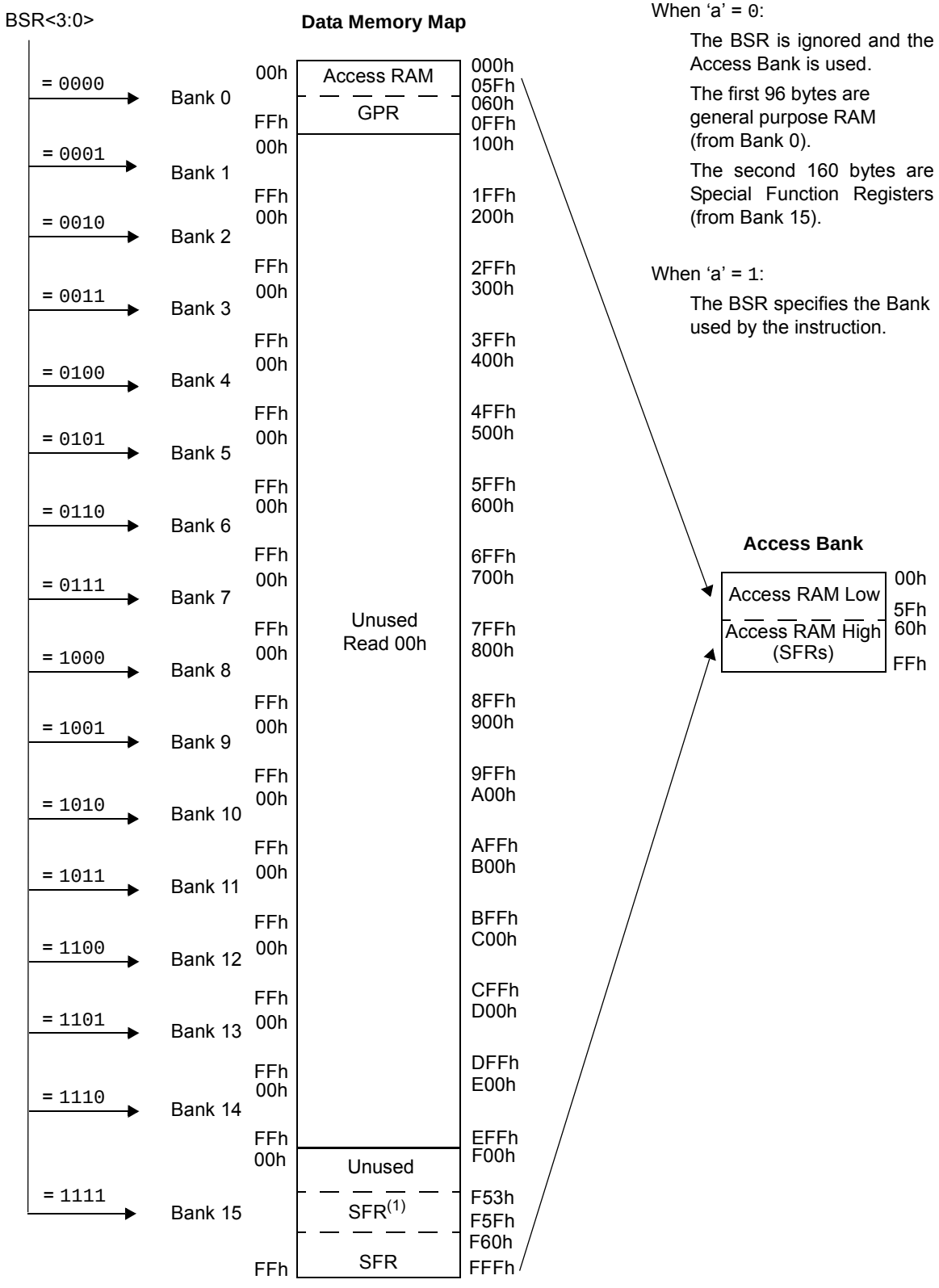
Note: See Table 1 for location of all peripheral functions.

FIGURE 2: 20-PIN QFN (4x4)



Note: See Table 1 for location of all peripheral functions.

FIGURE 3-5: DATA MEMORY MAP FOR PIC18(L)F13K22 DEVICES



Note 1: SFRs occupying F53h to F5Fh address space are not in the virtual bank.

3.3.2 ACCESS BANK

While the use of the BSR with an embedded 8-bit address allows users to address the entire range of data memory, it also means that the user must always ensure that the correct bank is selected. Otherwise, data may be read from or written to the wrong location. This can be disastrous if a GPR is the intended target of an operation, but an SFR is written to instead. Verifying and/or changing the BSR for each read or write to data memory can become very inefficient.

To streamline access for the most commonly used data memory locations, the data memory is configured with an Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 96 bytes of memory (00h-5Fh) in Bank 0 and the last 160 bytes of memory (60h-FFh) in Block 15. The lower half is known as the “Access RAM” and is composed of GPRs. This upper half is also where the device’s SFRs are mapped. These two areas are mapped contiguously in the Access Bank and can be addressed in a linear fashion by an 8-bit address (Figure 3-5 and Figure 3-6).

The Access Bank is used by core PIC18 instructions that include the Access RAM bit (the ‘a’ parameter in the instruction). When ‘a’ is equal to ‘1’, the instruction uses the BSR and the 8-bit address included in the opcode for the data memory address. When ‘a’ is ‘0’, however, the instruction is forced to use the Access Bank address map; the current value of the BSR is ignored entirely.

Using this “forced” addressing allows the instruction to operate on a data address in a single cycle, without updating the BSR first. For 8-bit addresses of 60h and above, this means that users can evaluate and operate on SFRs more efficiently. The Access RAM below 60h is a good place for data values that the user might need to access rapidly, such as immediate computational results or common program variables. Access RAM also allows for faster and more code efficient context saving and switching of variables.

The mapping of the Access Bank is slightly different when the extended instruction set is enabled (XINST Configuration bit = 1). This is discussed in more detail in **Section 3.5.3 “Mapping the Access Bank in Indexed Literal Offset Mode”**.

3.3.3 GENERAL PURPOSE REGISTER FILE

PIC18 devices may have banked memory in the GPR area. This is data RAM, which is available for use by all instructions. GPRs start at the bottom of Bank 0 (address 000h) and grow upwards towards the bottom of the SFR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.

3.3.4 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (FFFh) and extend downward to occupy the top portion of Bank 15 (F60h to FFFh). A list of these registers is given in Table 3-1 and Table 3-2.

The SFRs can be classified into two sets: those associated with the “core” device functionality (ALU, Resets and interrupts) and those related to the peripheral functions. The Reset and Interrupt registers are described in their respective chapters, while the ALU’s STATUS register is described later in this section. Registers related to the operation of a peripheral feature are described in the chapter for that peripheral.

The SFRs are typically distributed among the peripherals whose functions they control. Unused SFR locations are unimplemented and read as ‘0’s.

TABLE 3-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F1XK22 DEVICES

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FD7h	TMR0H	FAFh	SPBRG	F87h	—(2)	F5Fh	—(2)
FFEh	TOSH	FD6h	TMR0L	FAEh	RCREG	F86h	—(2)	F5Eh	—(2)
FFDh	TOSL	FD5h	T0CON	FADh	TXREG	F85h	—(2)	F5Dh	—(2)
FFCh	STKPTR	FD4h	—(2)	FACh	TXSTA	F84h	—(2)	F5Ch	—(2)
FFBh	PCLATU	FD3h	OSCCON	FABh	RCSTA	F83h	—(2)	F5Bh	—(2)
FFAh	PCLATH	FD2h	OSCCON2	FAAh	—(2)	F82h	PORTC	F5Ah	—(2)
FF9h	PCL	FD1h	WDTCON	FA9h	EEADR	F81h	PORTB	F59h	—(2)
FF8h	TBLPTRU	FD0h	RCON	FA8h	EEDATA	F80h	PORTA	F58h	—(2)
FF7h	TBLPTRH	FCFh	TMR1H	FA7h	EECON2(1)	F7Fh	ANSELH	F57h	—(2)
FF6h	TBLPTRL	FCEh	TMR1L	FA6h	EECON1	F7Eh	ANSEL	F56h	—(2)
FF5h	TABLAT	FCDh	T1CON	FA5h	—(2)	F7Dh	—(2)	F55h	—(2)
FF4h	PRODH	FCCh	TMR2	FA4h	—(2)	F7Ch	—(2)	F54h	—(2)
FF3h	PRODL	FCBh	PR2	FA3h	—(2)	F7Bh	—(2)	F53h	—(2)
FF2h	INTCON	FCAh	T2CON	FA2h	IPR2	F7Ah	IOCB		
FF1h	INTCON2	FC9h	SSPBUF	FA1h	PIR2	F79h	IOCA		
FF0h	INTCON3	FC8h	SSPADD	FA0h	PIE2	F78h	WPUB		
FEFh	INDF0(1)	FC7h	SSPSTAT	F9Fh	IPR1	F77h	WPUA		
FEEh	POSTINC0(1)	FC6h	SSPCON1	F9Eh	PIR1	F76h	SLRCON		
FEDh	POSTDEC0(1)	FC5h	SSPCON2	F9Dh	PIE1	F75h	—(2)		
FECh	PREINC0(1)	FC4h	ADRESH	F9Ch	—(2)	F74h	—(2)		
FEBh	PLUSW0(1)	FC3h	ADRESL	F9Bh	OSCTUNE	F73h	—(2)		
FEAh	FSR0H	FC2h	ADCON0	F9Ah	—(2)	F72h	—(2)		
FE9h	FSR0L	FC1h	ADCON1	F99h	—(2)	F71h	—(2)		
FE8h	WREG	FC0h	ADCON2	F98h	—(2)	F70h	—(2)		
FE7h	INDF1(1)	FBFh	CCPR1H	F97h	—(2)	F6Fh	SSPMASK		
FE6h	POSTINC1(1)	FBEh	CCPR1L	F96h	—(2)	F6Eh	—(2)		
FE5h	POSTDEC1(1)	FBDh	CCP1CON	F95h	—(2)	F6Dh	CM1CON0		
FE4h	PREINC1(1)	FBCh	VREFCON2	F94h	TRISC	F6Ch	CM2CON1		
FE3h	PLUSW1(1)	FBBh	VREFCON1	F93h	TRISB	F6Bh	CM2CON0		
FE2h	FSR1H	FBAh	VREFCON0	F92h	TRISA	F6Ah	—(2)		
FE1h	FSR1L	FB9h	PSTRCON	F91h	—(2)	F69h	SRCON1		
FE0h	BSR	FB8h	BAUDCON	F90h	—(2)	F68h	SRCON0		
FDFh	INDF2(1)	FB7h	PWM1CON	F8Fh	—(2)	F67h	—(2)		
FDEh	POSTINC2(1)	FB6h	ECCP1AS	F8Eh	—(2)	F66h	—(2)		
FDDh	POSTDEC2(1)	FB5h	—(2)	F8Dh	—(2)	F65h	—(2)		
FDCh	PREINC2(1)	FB4h	—(2)	F8Ch	—(2)	F64h	—(2)		
FDBh	PLUSW2(1)	FB3h	TMR3H	F8Bh	LATC	F63h	—(2)		
FDAh	FSR2H	FB2h	TMR3L	F8Ah	LATB	F62h	—(2)		
FD9h	FSR2L	FB1h	T3CON	F89h	LATA	F61h	—(2)		
FD8h	STATUS	FB0h	SPBRGH	F88h	—(2)	F60h	—(2)		

Legend: = Unimplemented data memory locations, read as '0',

Note 1: This is not a physical register.

2: Unimplemented registers are read as '0'.

TABLE 8-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	247
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	244
INTCON2	$\overline{\text{RABPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RABIP	244
IOCB	IOCB7	IOCB6	IOCB5	IOCB4					247
LATB	LATB7	LATB6	LATB5	LATB4	—	—	—	—	247
PORTB	RB7	RB6	RB5	RB4	—	—	—	—	247
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	246
SLRCON	—	—	—	—	—	SLRC	SLRB	SLRA	247
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	245
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	247
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	246
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	—	—	—	—	247

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

TABLE 8-5: PORTC I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RC0/AN4/C2IN+	RC0	0	O	DIG	LATC<0> data output.
		1	I	ST	PORTC<0> data input.
	AN4	1	I	ANA	A/D input channel 4.
	C2IN+	1	I	ANA	Comparators C2 noninverting input.
RC1/AN5/ C12IN1-	RC1	0	O	DIG	LATC<1> data output.
		1	I	ST	PORTC<1> data input.
	AN5	1	I	ANA	A/D input channel 5.
	C12IN1-	1	I	ANA	Comparators C1 and C2 inverting input, channel 1.
RC2/AN6/ C12IN2-/P1D	RC2	0	O	DIG	LATC<2> data output.
		1	I	ST	PORTC<2> data input.
	AN6	1	I	ANA	A/D input channel 6.
	C12IN2-	1	I	ANA	Comparators C1 and C2 inverting input, channel 2.
	P1D	0	O	DIG	ECCP1 Enhanced PWM output, channel D.
RC3/AN7/ C12IN3-/P1C/ PGM	RC3	0	O	DIG	LATC<3> data output.
		1	I	ST	PORTC<3> data input.
	AN7	1	I	ANA	A/D input channel 7.
	C12IN3-	1	I	ANA	Comparators C1 and C2 inverting input, channel 3.
	P1C	0	O	DIG	ECCP1 Enhanced PWM output, channel C.
	PGM	x	I	ST	Single-Supply Programming mode entry (ICSP™). Enabled by LVP Configuration bit; all other pin functions disabled.
RC4/C2OUT/P1B/ SRNQ	RC4	0	O	DIG	LATC<4> data output.
		1	I	ST	PORTC<4> data input.
	C2OUT	0	O	DIG	Comparator 2 output.
	P1B	0	O	DIG	ECCP1 Enhanced PWM output, channel B.
	SRNQ	0	O	DIG	SR Latch inverted output
RC5/CCP1/P1A	RC5	0	O	DIG	LATC<5> data output.
		1	I	ST	PORTC<5> data input.
	CCP1	0	O	DIG	ECCP1 compare or PWM output.
		1	I	ST	ECCP1 capture input.
	P1A	0	0	DIG	ECCP1 Enhanced PWM output, channel A.
RC6/AN8/$\overline{SS}$	RC6	0	O	DIG	LATC<6> data output.
		1	I	ST	PORTC<6> data input.
	AN8	1	I	ANA	A/D input channel 8.
	$\overline{SS}$	1	I	TTL	Slave select input for SSP (MSSP module)
RC7/AN9/SDO	RC7	0	O	DIG	LATC<7> data output.
		1	I	ST	PORTC<7> data input.
	AN9	1	I	ANA	A/D input channel 9.
	SDO	0	O	DIG	SPI data output (MSSP module).

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

10.0 TIMER1 MODULE

The Timer1 timer/counter module incorporates the following features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR1H and TMR1L)
- Selectable internal or external clock source and Timer1 oscillator options
- Interrupt-on-overflow
- Reset on CCP Special Event Trigger
- Device clock status flag (T1RUN)

A simplified block diagram of the Timer1 module is shown in Figure 10-1. A block diagram of the module's operation in Read/Write mode is shown in Figure 10-2.

Timer1 can also be used to provide Real-Time Clock (RTC) functionality to applications with only a minimal addition of external components and code overhead.

Timer1 is controlled through the T1CON Control register (Register 10-1). It also contains the Timer1 Oscillator Enable bit (T1OSCEN). Timer1 can be enabled or disabled by setting or clearing control bit, TMR1ON of the T1CON register.

REGISTER 10-1: T1CON: TIMER1 CONTROL REGISTER

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
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 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register read/write of TImer1 in one 16-bit operation
0 = Enables register read/write of Timer1 in two 8-bit operations

bit 6 **T1RUN:** Timer1 System Clock Status bit
1 = Main system clock is derived from Timer1 oscillator
0 = Main system clock is derived from another source

bit 5-4 **T1CKPS<1:0>:** Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable bit
1 = Timer1 oscillator is enabled
0 = Timer1 oscillator is shut off
The oscillator inverter and feedback resistor are turned off to eliminate power drain.

bit 2 **T1SYNC:** Timer1 External Clock Input Synchronization Select bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from the T13CKI pin (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

14.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

14.1 Master SSP (MSSP) Module Overview

The Master Synchronous Serial Port (MSSP) 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 MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I^2C)
  - Full Master mode
  - Slave mode (with general address call)

The I^2C interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode

14.2 SPI Mode

The SPI mode allows eight bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

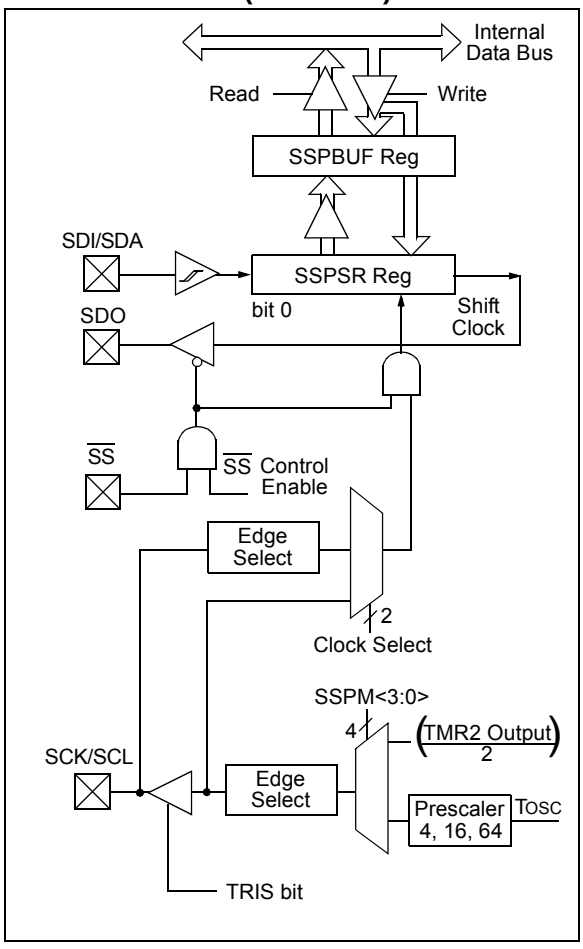
- Serial Data Out – SDO
- Serial Data In – SDI
- Serial Clock – SCK

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select – $\overline{SS}$

Figure 14-1 shows the block diagram of the MSSP module when operating in SPI mode.

FIGURE 14-1: MSSP BLOCK DIAGRAM (SPI MODE)



14.3.2 OPERATION

The MSSP module functions are enabled by setting SSPEN bit of the SSPCON1 register.

The SSPCON1 register allows control of the I^2C operation. Four mode selection bits of the SSPCON1 register allow one of the following I^2C modes to be selected:

- I^2C Master mode, clock = (FOSC/(4*(SSPADD + 1))
- I^2C Slave mode (7-bit address)
- I^2C Slave mode (10-bit address)
- I^2C Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- I^2C Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- I^2C Firmware Controlled Master mode, slave is Idle

Selection of any I^2C mode with the SSPEN bit set, forces the SCL and SDA pins to be open-drain, provided these pins are programmed to inputs by setting the appropriate TRIS bits

Note:	To ensure proper operation of the module, pull-up resistors must be provided externally to the SCL and SDA pins.

14.3.3 SLAVE MODE

In Slave mode, the SCL and SDA pins must be configured as inputs. The MSSP module will override the input state with the output data when required (slave-transmitter).

The I^2C Slave mode hardware will always generate an interrupt on an address match. Through the mode select bits, the user can also choose to interrupt on Start and Stop bits

When an address is matched, or the data transfer after an address match is received, the hardware automatically will generate the Acknowledge ($\overline{ACK}$) pulse and load the SSPBUF register with the received value currently in the SSPSR register.

Any combination of the following conditions will cause the MSSP module not to give this $\overline{ACK}$ pulse:

- The Buffer Full bit, BF bit of the SSPSTAT register, is set before the transfer is received.
- The overflow bit, SSPOV bit of the SSPCON1 register, is set before the transfer is received.

In this case, the SSPSR register value is not loaded into the SSPBUF, but bit SSPIF of the PIR1 register is set. The BF bit is cleared by reading the SSPBUF register, while bit SSPOV is cleared through software.

The SCL clock input must have a minimum high and low for proper operation. The high and low times of the I^2C specification, as well as the requirement of the MSSP module, are shown in **Section 26.0 "Electrical Specifications"**.

14.3.3.1 Addressing

Once the MSSP module has been enabled, it waits for a Start condition to occur. Following the Start condition, the eight bits are shifted into the SSPSR register. All incoming bits are sampled with the rising edge of the clock (SCL) line. The value of register SSPSR<7:1> is compared to the value of the SSPADD register. The address is compared on the falling edge of the eighth clock (SCL) pulse. If the addresses match and the BF and SSPOV bits are clear, the following events occur:

1. The SSPSR register value is loaded into the SSPBUF register.
2. The Buffer Full bit, BF, is set.
3. An $\overline{ACK}$ pulse is generated.
4. MSSP Interrupt Flag bit, SSPIF of the PIR1 register, is set (interrupt is generated, if enabled) on the falling edge of the ninth SCL pulse.

In 10-bit Address mode, two address bytes need to be received by the slave. The five Most Significant bits (MSbs) of the first address byte specify if this is a 10-bit address. Bit R/$\overline{W}$ of the SSPSTAT register must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '11110 A9 A8 0', where 'A9' and 'A8' are the two MSbs of the address. The sequence of events for 10-bit address is as follows, with steps 7 through 9 for the slave-transmitter:

1. Receive first (high) byte of address (bits SSPIF, BF and UA of the SSPSTAT register are set).
2. Read the SSPBUF register (clears bit BF) and clear flag bit, SSPIF.
3. Update the SSPADD register with second (low) byte of address (clears bit UA and releases the SCL line).
4. Receive second (low) byte of address (bits SSPIF, BF and UA are set). If the address matches then the SCL is held until the next step. Otherwise the SCL line is not held.
5. Read the SSPBUF register (clears bit BF) and clear flag bit, SSPIF.
6. Update the SSPADD register with the first (high) byte of address. (This will clear bit UA and release a held SCL line.)
7. Receive Repeated Start condition.
8. Receive first (high) byte of address with R/$\overline{W}$ bit set (bits SSPIF, BF, R/$\overline{W}$ are set).
9. Read the SSPBUF register (clears bit BF) and clear flag bit, SSPIF.
10. Load SSPBUF with byte the slave is to transmit, sets the BF bit.
11. Set the CKP bit to release SCL.

15.3.4 BREAK CHARACTER SEQUENCE

The EUSART module has the capability of sending the special Break character sequences that are required by the LIN bus standard. A Break character consists of a Start bit, followed by 12 '0' bits and a Stop bit.

To send a Break character, set the SENDB and TXEN bits of the TXSTA register. The Break character transmission is then initiated by a write to the TXREG. The value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN specification).

The TRMT bit of the TXSTA register indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 15-9 for the timing of the Break character sequence.

15.3.4.1 Break and Sync Transmit Sequence

The following sequence will start a message frame header made up of a Break, followed by an auto-baud Sync byte. This sequence is typical of a LIN bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to enable the Break sequence.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware and the Sync character is then transmitted.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

15.3.5 RECEIVING A BREAK CHARACTER

The Enhanced EUSART module can receive a Break character in two ways.

The first method to detect a Break character uses the FERR bit of the RCSTA register and the Received data as indicated by RCREG. The Baud Rate Generator is assumed to have been initialized to the expected baud rate.

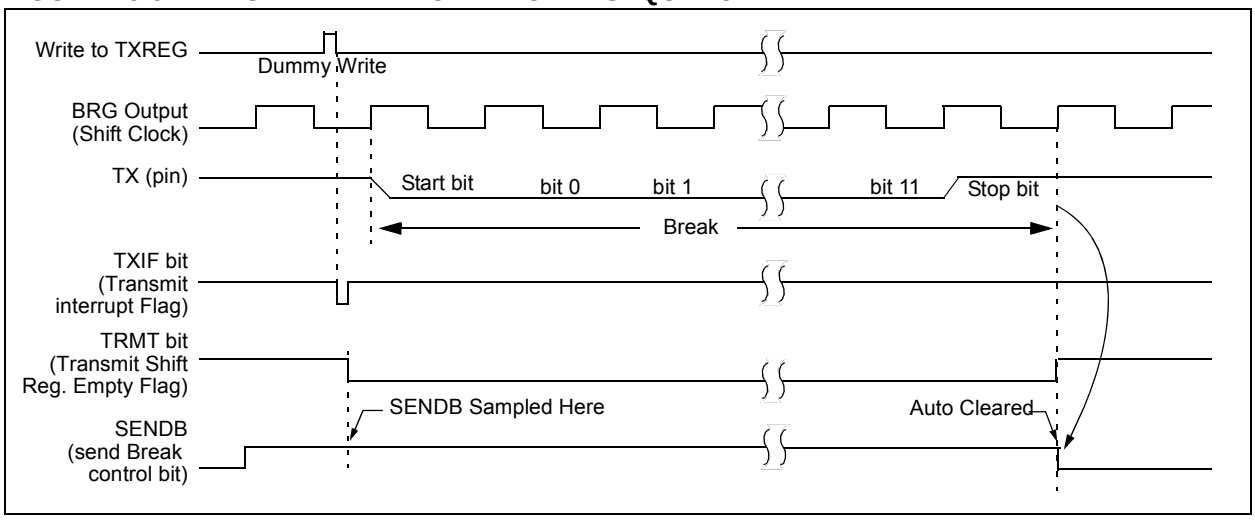
A Break character has been received when;

- RCIF bit is set
- FERR bit is set
- RCREG = 00h

The second method uses the Auto-Wake-up feature described in **Section 15.3.3 "Auto-Wake-up on Break"**. By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RCIF interrupt, and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Detect feature. For both methods, the user can set the ABDEN bit of the BAUDCON register before placing the EUSART in Sleep mode.

FIGURE 15-9: SEND BREAK CHARACTER SEQUENCE



15.4.2.3 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical (**Section 15.4.1.6 "Synchronous Master Reception"**), with the following exceptions:

- Sleep
- CREN bit is always set, therefore the receiver is never Idle
- SREN bit, which is a "don't care" in Slave mode

A character may be received while in Sleep mode by setting the CREN bit prior to entering Sleep. Once the word is received, the RSR register will transfer the data to the RCREG register. If the RCIE enable bit is set, the interrupt generated will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will branch to the interrupt vector.

15.4.2.4 Synchronous Slave Reception Set-up

1. Set the SYNC and SPEN bits and clear the CSRC bit. Set the TRIS bits corresponding to the RX/DT and TX/CK I/O pins.
2. If using interrupts, ensure that the GIE and PEIE bits of the INTCON register are set and set the RCIE bit.
3. If 9-bit reception is desired, set the RX9 bit.
4. Set the CREN bit to enable reception.
5. The RCIF bit will be set when reception is complete. An interrupt will be generated if the RCIE bit was set.
6. If 9-bit mode is enabled, retrieve the Most Significant bit from the RX9D bit of the RCSTA register.
7. Retrieve the eight Least Significant bits from the receive FIFO by reading the RCREG register.
8. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCSTA register or by clearing the SPEN bit which resets the EUSART.

TABLE 15-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	247
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	245
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	248
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	248
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	248
RCREG	EUSART Receive Register								247
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	247
SPBRG	EUSART Baud Rate Generator Register, Low Byte								247
SPBRGH	EUSART Baud Rate Generator Register, High Byte								247
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	247

Legend: — = unimplemented, read as '0'. Shaded cells are not used for synchronous slave reception.

23.2.1 CONTROL REGISTER

Register 23-14 shows the WDTCON register. This is a readable and writable register which contains a control bit that allows software to override the WDT enable Configuration bit, but only if the Configuration bit has disabled the WDT.

REGISTER 23-14: WDTCON: WATCHDOG TIMER CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	SWDTEN(1)
bit 7							bit 0

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

bit 7-1 **Unimplemented:** Read as '0'

bit 0 **SWDTEN:** Software Enable or Disable the Watchdog Timer bit(1)
1 = WDT is turned on
0 = WDT is turned off (Reset value)

Note 1: This bit has no effect if the Configuration bit, WDTEN, is enabled.

TABLE 23-2: SUMMARY OF WATCHDOG TIMER REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN	253
RCON	IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	246
WDTCON	—	—	—	—	—	—	—	SWDTEN	246

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the Watchdog Timer.

23.3 Program Verification and Code Protection

The overall structure of the code protection on the PIC18 Flash devices differs significantly from other PIC microcontroller devices.

The user program memory is divided into five blocks. One of these is a boot block of 0.5K or 2K bytes, depending on the device. The remainder of the memory is divided into individual blocks on binary boundaries.

Each of the five blocks has three code protection bits associated with them. They are:

- Code-Protect bit (CPn)
- Write-Protect bit (WRTn)
- External Block Table Read bit (EBTRn)

Figure 23-2 shows the program memory organization for 8, 16 and 32-Kbyte devices and the specific code protection bit associated with each block. The actual locations of the bits are summarized in Table 23-3.

TABLE 24-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
LITERAL OPERATIONS									
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	Move literal (12-bit) 2nd word	2	1110	1110	00ff	kkkk	None	
		to FSR(f) 1st word		1111	0000	kkkk	kkkk		
MOVLB	k	Move literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS									
TBLRD*		Table Read	2	0000	0000	0000	1000	None	
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None	
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None	
TBLRD+*		Table Read with pre-increment		0000	0000	0000	1011	None	
TBLWT*		Table Write	2	0000	0000	0000	1100	None	
TBLWT*+		Table Write with post-increment		0000	0000	0000	1101	None	
TBLWT*-		Table Write with post-decrement		0000	0000	0000	1110	None	
TBLWT+*		Table Write with pre-increment		0000	0000	0000	1111	None	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.

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

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

INCFSZ **Increment f, skip if 0**

Syntax: INCFSZ f {,d {,a}}

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

Operation: (f) + 1 → dest,
skip if result = 0

Status Affected: None

Encoding:

0011	11da	ffff	ffff

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it 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 (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 24.2.3 "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	Write to destination

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    INCFSZ   CNT, 1, 0
NZERO   :
ZERO    :
```

Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT + 1
If CNT = 0;
PC = Address (ZERO)
If CNT ≠ 0;
PC = Address (NZERO)

INFSNZ **Increment f, skip if not 0**

Syntax: INFSNZ f {,d {,a}}

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

Operation: (f) + 1 → dest,
skip if result ≠ 0

Status Affected: None

Encoding:

0100	10da	ffff	ffff

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it 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 (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 24.2.3 "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	Write to destination

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    INFSNZ  REG, 1, 0
ZERO
NZERO
```

Before Instruction
PC = Address (HERE)
After Instruction
REG = REG + 1
If REG ≠ 0;
PC = Address (NZERO)
If REG = 0;
PC = Address (ZERO)

IORLW **Inclusive OR literal with W**

Syntax: IORLW k

Operands: $0 \le k \le 255$

Operation: (W) .OR. k → W

Status Affected: N, Z

Encoding:

0000	1001	kkkk	kkkk

Description: The contents of W are ORed 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: IORLW 35h

Before Instruction
W = 9Ah
After Instruction
W = BFh

IORWF **Inclusive OR W with f**

Syntax: IORWF f {,d {,a}}

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

Operation: (W) .OR. (f) → dest

Status Affected: N, Z

Encoding:

0001	00da	ffff	ffff

Description: Inclusive OR W with register 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: IORWF RESULT, 0, 1

Before Instruction
RESULT = 13h
W = 91h
After Instruction
RESULT = 13h
W = 93h

MOVLW **Move literal to W**

Syntax: MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow W$

Status Affected: None

Encoding:

0000	1110	kkkk	kkkk

Description: The 8-bit literal ‘k’ is loaded into W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: MOVLW 5Ah

After Instruction

W = 5Ah

MOVWF **Move W to f**

Syntax: MOVWF f {,a}

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

Operation: $(W) \rightarrow f$

Status Affected: None

Encoding:

0110	111a	ffff	ffff

Description: Move data from W to register ‘f’. Location ‘f’ can be anywhere in the 256-byte bank.
If ‘a’ is ‘0’, the Access Bank is selected.
If ‘a’ is ‘1’, the BSR is used to select the GPR bank (default).
If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write register ‘f’

Example: MOVWF REG, 0

Before Instruction

W = 4Fh
REG = FFh

After Instruction

W = 4Fh
REG = 4Fh

RCALL **Relative Call**

Syntax: RCALL n

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

Operation: $(PC) + 2 \rightarrow TOS$,
$(PC) + 2 + 2n \rightarrow PC$

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 $\overline{MCLR}$ Reset.

Status Affected: All

Encoding:

0000	0000	1111	1111

Description: This instruction provides a way to execute a $\overline{MCLR}$ Reset by software.

Words: 1

Cycles: 1

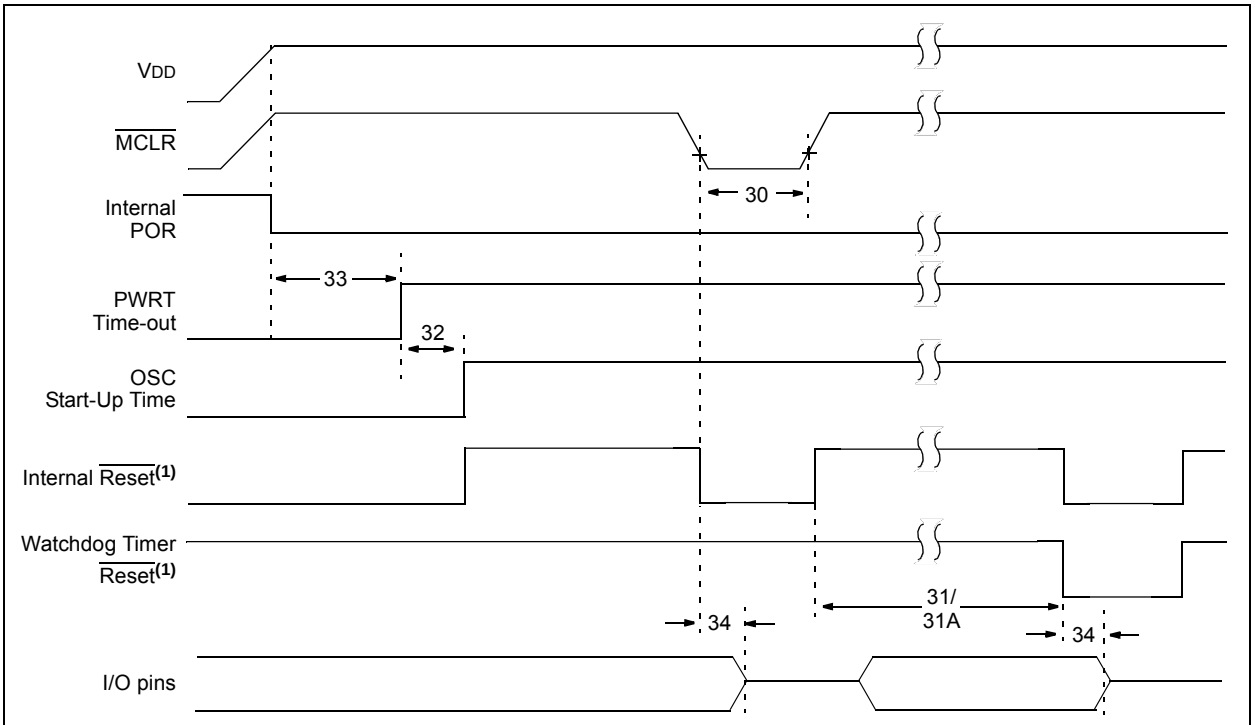
Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start Reset	No operation	No operation

Example: RESET

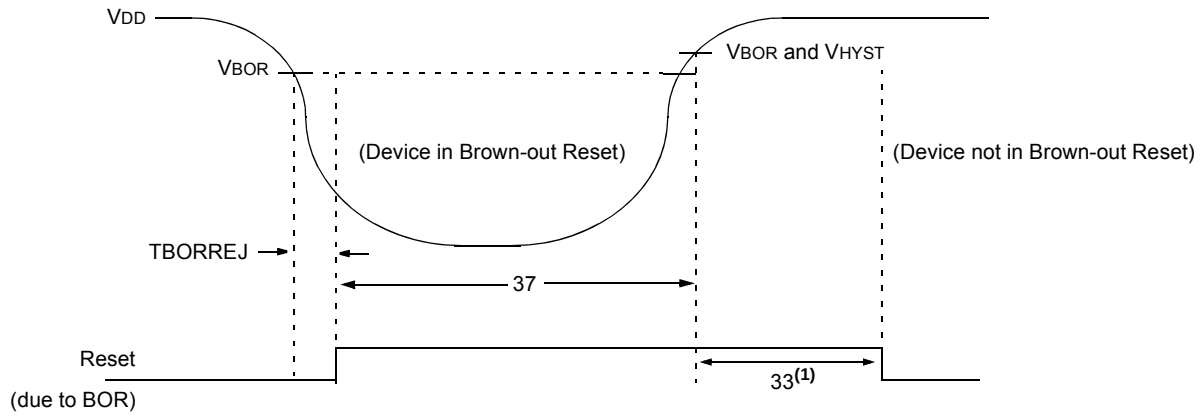
After Instruction
Registers = Reset Value
Flags* = Reset Value

FIGURE 26-10: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING



Note 1: Asserted low.

FIGURE 26-11: BROWN-OUT RESET TIMING AND CHARACTERISTICS



Note 1: 64 ms delay only if $\overline{\text{PWRTE}}$ bit in the Configuration Word register is programmed to '0'. 2 ms delay if $\overline{\text{PWRTE}}$ = 0.

FIGURE 26-12: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS

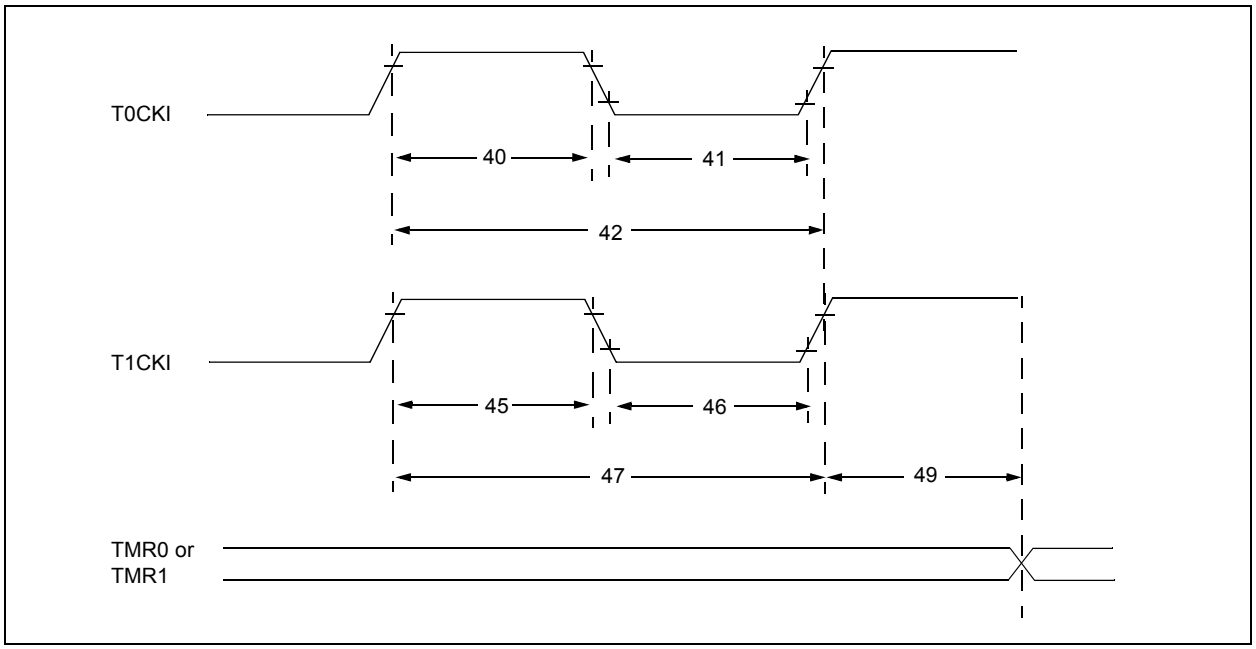


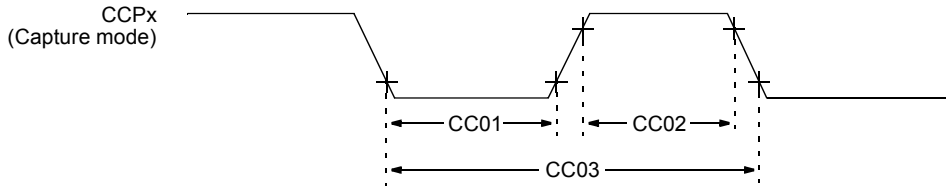
TABLE 26-17: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)								
Param. No.	**Sym.**	**Characteristic**		**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
40*	TT0H	T0CKI High-Pulse Width	No Prescaler	0.5 TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
41*	TT0L	T0CKI Low-Pulse Width	No Prescaler	0.5 TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
42*	TT0P	T0CKI Period		Greater of: 20 or TCY + 40 / N	—	—	ns	N = prescale value (2, 4, ..., 256)
45*	TT1H	T1CKI High Time	Synchronous, No Prescaler	0.5 TCY + 20	—	—	ns	
			Synchronous, with Prescaler	15	—	—	ns	
			Asynchronous	30	—	—	ns	
46*	TT1L	T1CKI Low Time	Synchronous, No Prescaler	0.5 TCY + 20	—	—	ns	
			Synchronous, with Prescaler	15	—	—	ns	
			Asynchronous	30	—	—	ns	
47*	TT1P	T1CKI Input Period	Synchronous	Greater of: 30 or TCY + 40 / N	—	—	ns	N = prescale value (1, 2, 4, 8)
			Asynchronous	60	—	—	ns	
48	FT1	Timer1 Oscillator Input Frequency Range (oscillator enabled by setting bit T1OSCEN)		32.4	32.768	33.1	kHz	
49*	TCKEZTMR1	Delay from External Clock Edge to Timer Increment		2 TOSC	—	7 TOSC	—	Timers in Sync mode

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 26-13: CAPTURE/COMPARE/PWM TIMINGS (CCP)



Note: Refer to Figure 26-7 for load conditions.

TABLE 26-18: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP)

Standard Operating Conditions (unless otherwise stated)								
Param. No.	**Sym.**	**Characteristic**		**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
CC01*	TccL	CCPx Input Low Time	No Prescaler	0.5TCY + 20	—	—	ns	
			With Prescaler	20	—	—	ns	
CC02*	TccH	CCPx Input High Time	No Prescaler	0.5TCY + 20	—	—	ns	
			With Prescaler	20	—	—	ns	
CC03*	TccP	CCPx Input Period		$\frac{3T_{CY} + 40}{N}$	—	—	ns	N = prescale value (1, 4 or 16)

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

TABLE 26-19: PIC18(L)F1XK22 A/D CONVERTER (ADC) CHARACTERISTICS

Standard Operating Conditions (unless otherwise stated) Operating temperature: Tested at 25°C							
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
AD01	NR	Resolution	—	—	10	bit	
AD02	EIL	Integral Error	—	—	±2	LSb	VREF = 3.0V
AD03	EDL	Differential Error	—	—	±1.5	LSb	No missing codes VREF = 3.0V
AD04	EOFF	Offset Error	—	—	±3	LSb	VREF = 3.0V
AD05	EGN	Gain Error	—	—	±3	LSb	VREF = 3.0V
AD06	VREF	Change in Reference Voltage = VREF+ - VREF-(2), (3)	1.8	—	VDD	V	1.8 ≤ VREF+ ≤ VDD + 0.3V VSS - 0.3V ≤ VREF- ≤ VREF+ - 1.8V
AD07	VAIN	Full-Scale Range	VSS	—	VREF	V	
AD08	ZAIN	Recommended Impedance of Analog Voltage Source	—	—	10	kΩ	Can go higher if external 0.01 µF capacitor is present on input pin.

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Total Absolute Error includes integral, differential, offset and gain errors.

2: ADC VREF is from external VREF, VDD pin or FVR, whichever is selected as reference input.

3: FVR voltage selected must be 2.048V or 4.096V.