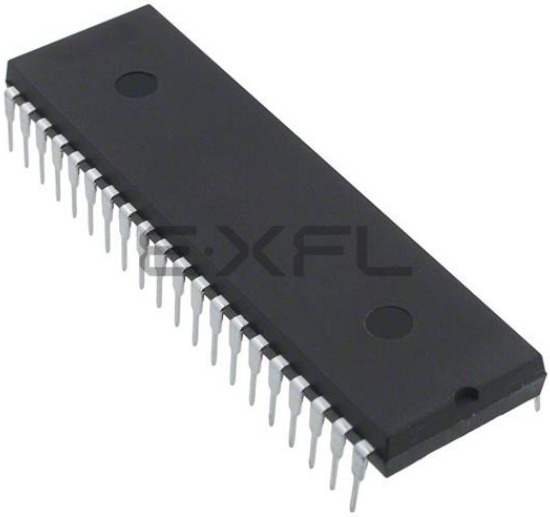


Welcome to E-XFL.COM

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	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 35x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	40-DIP (0.600", 15.24mm)
Supplier Device Package	40-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f45k40-i-p

TABLE 4-3: SUMMARY OF REGISTERS ASSOCIATED WITH CLOCK SOURCES

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
OSCCON1	—	NOSC<2:0>			NDIV<3:0>				36
OSCCON2	—	COSC<2:0>			CDIV<3:0>				37
OSCCON3	CSWHOLD	SOSCPWR	—	ORDY	NOSCR	—	—	—	38
OSCSTAT	EXTOR	HFOR	MFOR	LFOR	SOR	ADOR	—	PLLR	39
OSCTUNE	—	—	HFTUN<5:0>						41
OSCFRQ	—	—	—	—	HFFRQ<3:0>				40
OSCEN	EXTOEN	HFOEN	MFOEN	LFOEN	SOSCEN	ADOEN	—	—	42
PIE1	OSCFIE	CSWIE	—	—	—	—	ADTIE	ADIE	180
PIR1	OSCFIF	CSWIF	—	—	—	—	ADTIF	ADIF	172
IPR1	OSCFIP	CSWIP	—	—	—	—	ADTIP	ADIP	188

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

TABLE 4-4: SUMMARY OF CONFIGURATION WORD WITH CLOCK SOURCES

Name	Bits	Bit -/7	Bit -/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0	Register on Page
CONFIG1	13:8	—	—	FCMEN	—	CSWEN	—	—	$\overline{\text{CLKOUTEN}}$	23
	7:0	—	RSTOSC<2:0>			—	FEXTOSC<2:0>			

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

8.12 Start-up Sequence

Upon the release of a POR or BOR, the following must occur before the device will begin executing:

1. Power-up Timer runs to completion (if enabled).
2. Oscillator start-up timer runs to completion (if required for selected oscillator source).
3. $\overline{\text{MCLR}}$ must be released (if enabled).

The total time out will vary based on oscillator configuration and Power-up Timer configuration. See **Section 4.0 "Oscillator Module (with Fail-Safe Clock Monitor)"** for more information.

The Power-up Timer and oscillator start-up timer run independently of $\overline{\text{MCLR}}$ Reset. If $\overline{\text{MCLR}}$ is kept low long enough, the Power-up Timer and oscillator Start-up Timer will expire. Upon bringing $\overline{\text{MCLR}}$ high, the device will begin execution after 10 FOSC cycles (see Figure 8-4). This is useful for testing purposes or to synchronize more than one device operating in parallel.

FIGURE 8-4: RESET START-UP SEQUENCE

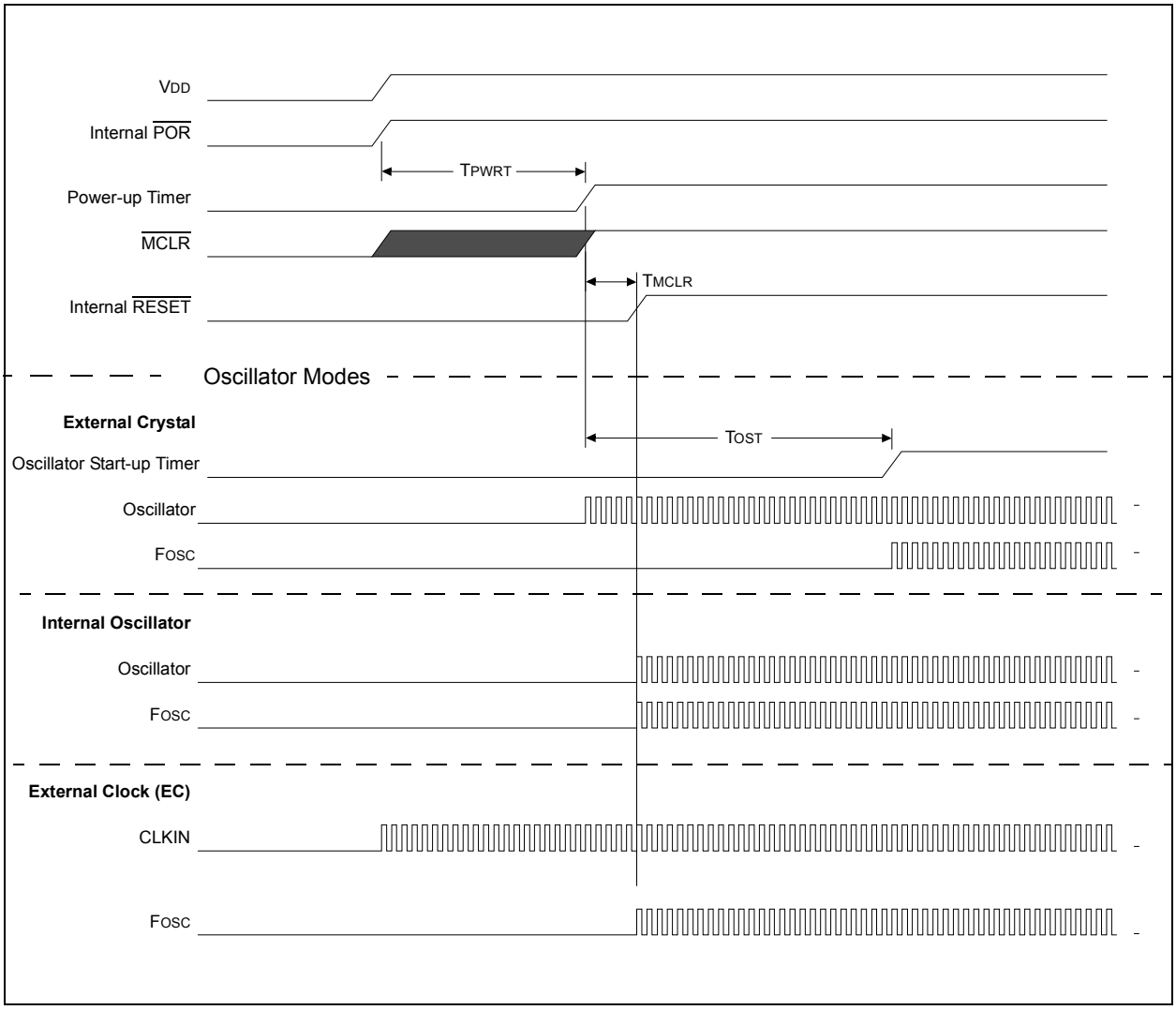


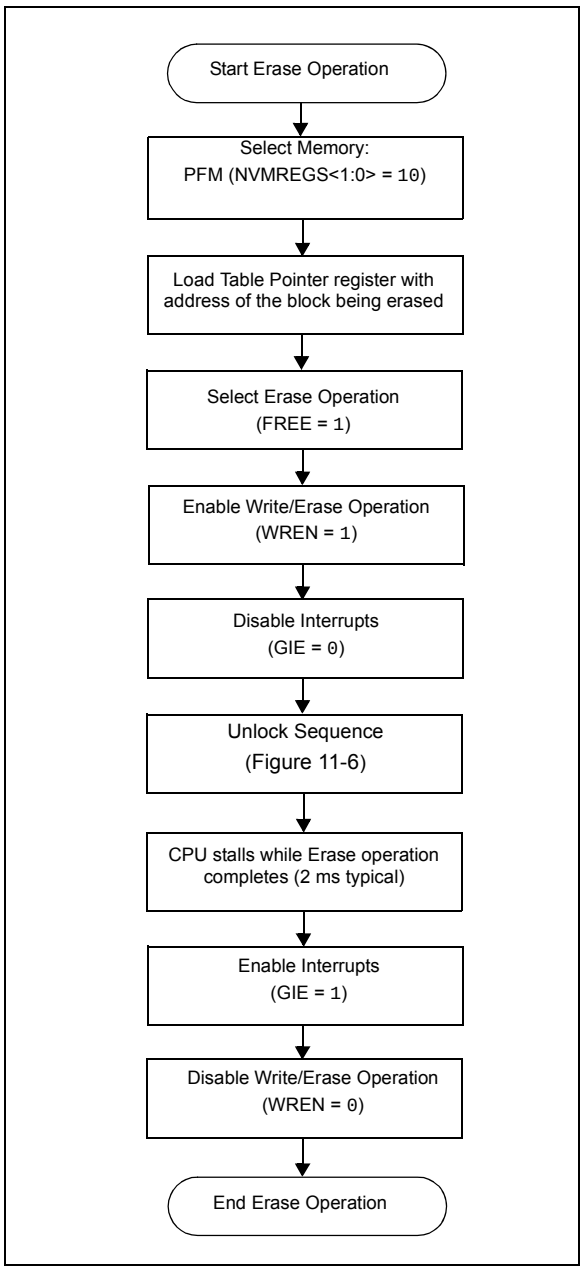
TABLE 10-4: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/45/46K40 DEVICES

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
F5Fh	ADPCH	F31h	FVRCON	F03h	RD4PPS(1)	ED5h	WDTPSH	EA7h	T3CKIPPS
F5Eh	ADPRE	F30h	HLVDCON1	F02h	RD3PPS(1)	ED4h	WDTPSL	EA6h	T1GPPS
F5Dh	ADCAP	F2Fh	HLVDCON0	F01h	RD2PPS(1)	ED3h	WDTCON1	EA5h	T1CKIPPS
F5Ch	ADACQ	F2Eh	ANSELE(2)	F00h	RD1PPS(2)	ED2h	WDTCON0	EA4h	T0CKIPPS
F5Bh	ADCON3	F2Dh	WPUE	EFFh	RD0PPS(2)	ED1h	PIR7	EA3h	INT2PPS
F5Ah	ADCON2	F2Ch	ODCONE(2)	EFEh	RC7PPS	ED0h	PIR6	EA2h	INT1PPS
F59h	ADCON1	F2Bh	SLRCONE(2)	EFDh	RC6PPS	ECFh	PIR5	EA1h	INT0PPS
F58h	ADREF	F2Ah	INLVLE	EFCh	RC5PPS	ECEh	PIR4	EA0h	PPSLOCK
F57h	ADCLK	F29h	IOCEP	EFBh	RC4PPS	ECDh	PIR3	E9Fh	BAUD2CON
F56h	ADACT	F28h	IOCEN	EFAh	RC3PPS	ECCh	PIR2	E9Eh	TX2STA
F55h	MDCARH	F27h	IOCEF	EF9h	RC2PPS	ECBh	PIR1	E9Dh	RC2STA
F54h	MDCARL	F26h	ANSELD(2)	EF8h	RC1PPS	ECAh	PIR0	E9Ch	SP2BRGH
F53h	MDSRC	F25h	WPUD(2)	EF7h	RC0PPS	EC9h	PIE7	E9Bh	SP2BRGL
F52h	MDCON1	F24h	ODCOND(2)	EF6h	RB7PPS	EC8h	PIE6	E9Ah	TX2REG
F51h	MDCON0	F23h	SLRCOND(2)	EF5h	RB6PPS	EC7h	PIE5	E99h	RC2REG
F50h	SCANDTRIG	F22h	INLVLD(2)	EF4h	RB5PPS	EC6h	PIE4	E98h	SSP2CON3
F4Fh	SCANCON0	F21h	ANSELC	EF3h	RB4PPS	EC5h	PIE3	E97h	SSP2CON2
F4Eh	SCANHADRU	F20h	WPUC	EF2h	RB3PPS	EC4h	PIE2	E96h	SSP2CON1
F4Dh	SCANHADRH	F1Fh	ODCONC	EF1h	RB2PPS	EC3h	PIE1	E95h	SSP2STAT
F4Ch	SCANHADRL	F1Eh	SLRCONC	EF0h	RB1PPS	EC2h	PIE0	E94h	SSP2MSK
F4Bh	SCANLADRU	F1Dh	INLVLC	EEFh	RB0PPS	EC1h	IPR7	E93h	SSP2ADD
F4Ah	SCANLADRH	F1Ch	IOCCP	EEEh	RA7PPS	EC0h	IPR6	E92h	SSP2BUF
F49h	SCANLADRL	F1Bh	IOCCN	EEDh	RA6PPS	EBFh	IPR5	E91h	SSP2SSPPS
F48h	CWG1STR	F1Ah	IOCCF	EECh	RA5PPS	EBEh	IPR4	E90h	SSP2DATPPS
F47h	CWG1AS1	F19h	ANSELB	EEBh	RA4PPS	EBDh	IPR3	E8Fh	SSP2CLKPPS
F46h	CWG1AS0	F18h	WPUB	EEAh	RA3PPS	EBCh	IPR2	E8Eh	TX2PPS
F45h	CWG1CON1	F17h	ODCONB	EE9h	RA2PPS	EBBh	IPR1	E8Dh	RX2PPS
F44h	CWG1CON0	F16h	SLRCONB	EE8h	RA1PPS	EBAh	IPR0		
F43h	CWG1DBF	F15h	INLVLB	EE7h	RA0PPS	EB9h	SSP1SSPPS		
F42h	CWG1DBR	F14h	IOCBP	EE6h	PMD5	EB8h	SSP1DATPPS		
F41h	CWG1ISM	F13h	IOCBN	EE5h	PMD4	EB7h	SSP1CLKPPS		
F40h	CWG1CLKCON	F12h	IOCBF	EE4h	PMD3	EB6h	TX1PPS		
F3Fh	CLKRCLK	F11h	ANSELA	EE3h	PMD2	EB5h	RX1PPS		
F3Eh	CLKRCON	F10h	WPUA	EE2h	PMD1	EB4h	MDSRCPPS		
F3Dh	CMOUT	F0Fh	ODCONA	EE1h	PMD0	EB3h	MDCARHPPS		
F3Ch	CM1PCH	F0Eh	SLRCONA	EE0h	BORCON	EB2h	MDCARLPPS		
F3Bh	CM1NCH	F0Dh	INLVLA	EDFh	VREGCON(1)	EB1h	CWGINPPS		
F3Ah	CM1CON1	F0Ch	IOCAP	EDEh	OSCFRQ	EB0h	CCP2PPS		
F39h	CM1CON0	F0Bh	IOCAN	EDDh	OSCTUNE	EAFh	CCP1PPS		
F38h	CM2PCH	F0Ah	IOCAF	EDCh	OSCEN	EAEh	ADACTPPS		
F37h	CM2NCH	F09h	RE2PPS(2)	EDBh	OSCSTAT	EADh	T6INPPS		
F36h	CM2CON1	F08h	RE1PPS(2)	EDAh	OSCCON3	EACh	T4INPPS		
F35h	CM2CON0	F07h	RE0PPS(2)	ED9h	OSCCON2	EABh	T2INPPS		
F34h	DAC1CON1	F06h	RD7PPS(2)	ED8h	OSCCON1	EAAh	T5GPPS		
F33h	DAC1CON0	F05h	RD6PPS(2)	ED7h	CPUDOZE	EA9h	T5CKIPPS		
F32h	ZCDCON	F04h	RD5PPS(2)	ED6h	WDTTMR	EA8h	T3GPPS		

Note 1: Not available on LF parts

2: Not available on PIC18(L)F26K40 (28-pin variants).

FIGURE 11-7: PFM ROW ERASE FLOWCHART



11.1.6 WRITING TO PROGRAM FLASH MEMORY

The programming write block size is described in Table 11-3. Word or byte programming is not supported. Table writes are used internally to load the holding registers needed to program the Flash memory. There are only as many holding registers as there are bytes in a write block. Refer to Table 11-3 for write latch size.

Since the table latch (TABLAT) is only a single byte, the TBLWT instruction needs to be executed multiple times for each programming operation. The write protection state is ignored for this operation. All of the table write operations will essentially be short writes because only the holding registers are written. NVMIF is not affected while writing to the holding registers.

After all the holding registers have been written, the programming operation of that block of memory is started by configuring the NVMCON1 register for a program memory write and performing the long write sequence.

If the PFM address in the TBLPTR is write-protected or if TBLPTR points to an invalid location, the WR bit is cleared without any effect and the WREER is signaled.

The long write is necessary for programming the internal Flash. CPU operation is suspended during a long write cycle and resumes when the operation is complete. The long write operation completes in one instruction cycle. When complete, WR is cleared in hardware and NVMIF is set and an interrupt will occur if NVMIE is also set. The latched data is reset to all '1s'. WREN is not changed.

The internal programming timer controls the write time. The write/erase voltages are generated by an on-chip charge pump, rated to operate over the voltage range of the device.

Note:	The default value of the holding registers on device Resets and after write operations is FFh. A write of FFh to a holding register does not modify that byte. This means that individual bytes of program memory may be modified, provided that the change does not attempt to change any bit from a '0' to a '1'. When modifying individual bytes, it is not necessary to load all holding registers before executing a long write operation.

REGISTER 14-23: IPR5: PERIPHERAL INTERRUPT PRIORITY REGISTER 5

U-0	U-0	U-0	U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1
—	—	—	—	—	TMR5GIP	TMR3GIP	TMR1GIP
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 **TMR5GIP:** TMR5 Gate Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR3GIP:** TMR3 Gate Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1GIP:** TMR1 Gate Interrupt Priority bit
1 = High priority
0 = Low priority

22.2 Register Definitions: PWM Control

Long bit name prefixes for the PWM peripherals are shown in Table 22-3. Refer to **Section 1.4.2.2 "Long Bit Names"** for more information.

TABLE 22-3:

Peripheral	Bit Name Prefix
PWM3	PWM3
PWM4	PWM4

REGISTER 22-1: PWMxCON: PWM CONTROL REGISTER

R/W-0/0	U-0	R-0/0	R/W-0/0	U-0	U-0	U-0	U-0
EN	—	OUT	POL	—	—	—	—
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 **EN:** PWM Module Enable bit
1 = PWM module is enabled
0 = PWM module is disabled

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

bit 5 **OUT:** PWM Module Output Level When Bit is Read

bit 4 **POL:** PWM Output Polarity Select bit
1 = PWM output is inverted
0 = PWM output is normal

bit 3-0 **Unimplemented:** Read as '0'

REGISTER 22-3: PWMxDCH: PWM DUTY CYCLE HIGH BITS

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
DCH<7:0>							
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 7-0 **DCh<7:0>:** PWM Duty Cycle Most Significant bits
These bits are the MSbs of the PWM duty cycle. The two LSbs are found in PWMxDCL Register.

REGISTER 22-4: PWMxDCL: PWM DUTY CYCLE LOW BITS

R/W-x/u	R/W-x/u	U-0	U-0	U-0	U-0	U-0	U-0
DCL<7:6>		—	—	—	—	—	—
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 7-6 **DC<8:9>:** PWM Duty Cycle Least Significant bits
These bits are the LSbs of the PWM duty cycle. The MSbs are found in PWMxDCH Register.

bit 5-0 **Unimplemented:** Read as ‘0’

26.5 SPI Mode Operation

Transmissions involve two shift registers, eight bits in size, one in the master and one in the slave. With either the master or the slave device, data is always shifted out one bit at a time, with the Most Significant bit (MSb) shifted out first. At the same time, a new Least Significant bit (LSb) is shifted into the same register.

Figure 26-3 shows a typical connection between two processors configured as master and slave devices.

Data is shifted out of both shift registers on the programmed clock edge and latched on the opposite edge of the clock.

The master device transmits information out on its SDO output pin which is connected to, and received by, the slave's SDI input pin. The slave device transmits information out on its SDO output pin, which is connected to, and received by, the master's SDI input pin.

To begin communication, the master device first sends out the clock signal. Both the master and the slave devices should be configured for the same clock polarity.

The master device starts a transmission by sending out the MSb from its shift register. The slave device reads this bit from that same line and saves it into the LSb position of its shift register.

During each SPI clock cycle, a full-duplex data transmission occurs. This means that while the master device is sending out the MSb from its shift register (on its SDO pin) and the slave device is reading this bit and saving it as the LSb of its shift register, that the slave device is also sending out the MSb from its shift register (on its SDO pin) and the master device is reading this bit and saving it as the LSb of its shift register.

After eight bits have been shifted out, the master and slave have exchanged register values.

If there is more data to exchange, the shift registers are loaded with new data and the process repeats itself.

Whether the data is meaningful or not (dummy data), depends on the application software. This leads to three scenarios for data transmission:

- Master sends useful data and slave sends dummy data.
- Master sends useful data and slave sends useful data.
- Master sends dummy data and slave sends useful data.

Transmissions may involve any number of clock cycles. When there is no more data to be transmitted, the master stops sending the clock signal and it deselects the slave.

Every slave device connected to the bus that has not been selected through its slave select line must disregard the clock and transmission signals and must not transmit out any data of its own.

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPxCON1<5:0> and SSPxSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

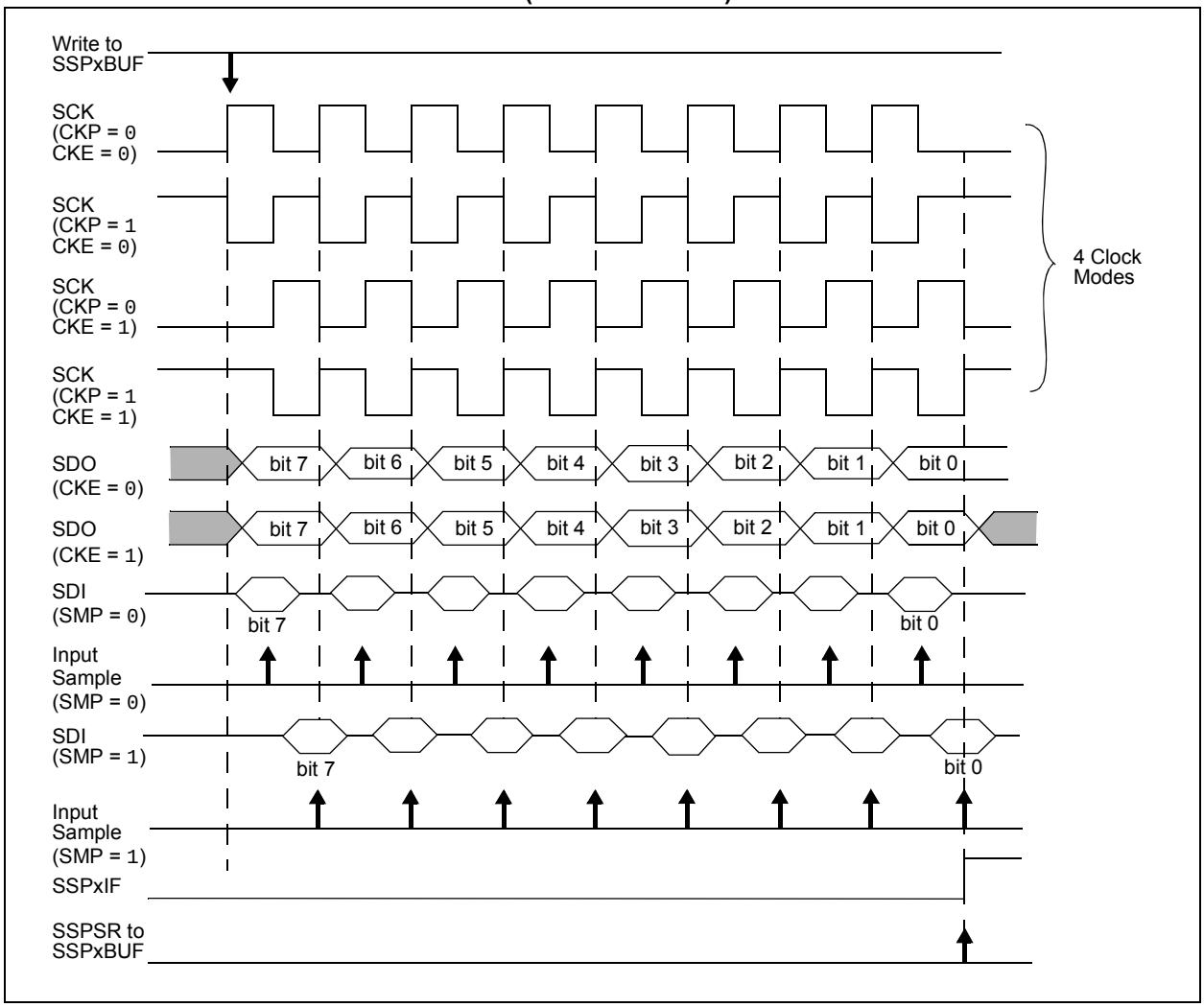
To enable the serial port, SSP Enable bit, SSPEN of the SSPxCON1 register, must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPxCONx registers and then set the SSPEN bit. This configures the SDI, SDO, SCK and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI must have corresponding TRIS bit set
- SDO must have corresponding TRIS bit cleared
- SCK (Master mode) must have corresponding TRIS bit cleared
- SCK (Slave mode) must have corresponding TRIS bit set
- $\overline{SS}$ must have corresponding TRIS bit set

Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

The MSSP consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPxBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPxBUF holds the data that was written to the SSPSR until the received data is ready. Once the eight bits of data have been received, that byte is moved to the SSPxBUF register. Then, the Buffer Full Detect bit, BF of the SSPxSTAT register, and the interrupt flag bit, SSPxIF, are set. This double-buffering of the received data (SSPxBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPxBUF register during transmission/reception of data will be ignored and the write collision detect bit, WCOL of the SSPxCON1 register, will be set. User software must clear the WCOL bit to allow the following write(s) to the SSPxBUF register to complete successfully.

FIGURE 26-4: SPI MODE WAVEFORM (MASTER MODE)



26.5.2 SPI SLAVE MODE

In Slave mode, the data is transmitted and received as external clock pulses appear on SCK. When the last bit is latched, the SSPxIF interrupt flag bit is set.

Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCK pin. The Idle state is determined by the CKP bit of the SSPxCON1 register.

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. The shift register is clocked from the SCK pin input and when a byte is received, the device will generate an interrupt. If enabled, the device will wake-up from Sleep.

26.5.3 DAISY-CHAIN CONFIGURATION

The SPI bus can sometimes be connected in a daisy-chain configuration. The first slave output is connected to the second slave input, the second slave output is connected to the third slave input, and so on. The final slave output is connected to the master input. Each slave sends out, during a second group of clock pulses, an exact copy of what was received during the first group of clock pulses. The whole chain acts as one large communication shift register. The daisy-chain feature only requires a single Slave Select line from the master device.

Figure 26-5 shows the block diagram of a typical daisy-chain connection when operating in SPI mode.

In a daisy-chain configuration, only the most recent byte on the bus is required by the slave. Setting the BOEN bit of the SSPxCON3 register will enable writes to the SSPxBUF register, even if the previous byte has not been read. This allows the software to ignore data that may not apply to it.

REGISTER 27-3: BAUDxCON: BAUD RATE CONTROL REGISTER

R-0/0	R-1/1	U-0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0
ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN
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 **ABDOVF:** Auto-Baud Detect Overflow bit
Asynchronous mode:
1 = Auto-baud timer overflowed
0 = Auto-baud timer did not overflow
Synchronous mode:
Don't care

bit 6 **RCIDL**: Receive Idle Flag bit
Asynchronous mode:
1 = Receiver is Idle
0 = Start bit has been received and the receiver is receiving
Synchronous mode:
Don't care

bit 5 **Unimplemented:** Read as '0'

bit 4 **SCKP**: Synchronous Clock Polarity Select bit
Asynchronous mode:
1 = Idle state for transmit (TX) is a low level (transmit data inverted)
0 = Idle state for transmit (TX) is a high level (transmit data is non-inverted)
Synchronous mode:
1 = Data is clocked on rising edge of the clock
0 = Data is clocked on falling edge of the clock

bit 3 **BRG16:** 16-bit Baud Rate Generator bit
1 = 16-bit Baud Rate Generator is used
0 = 8-bit Baud Rate Generator is used

bit 2 **Unimplemented:** Read as '0'

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = Receiver is waiting for a falling edge. No character will be received, byte RCxIF will be set. WUE will automatically clear after RCxIF is set.
0 = Receiver is operating normally
Synchronous mode:
Don't care

bit 0 **ABDEN**: Auto-Baud Detect Enable bit
Asynchronous mode:
1 = Auto-Baud Detect mode is enabled (clears when auto-baud is complete)
0 = Auto-Baud Detect mode is disabled
Synchronous mode:
Don't care

27.2.2.8 Asynchronous Reception Setup:

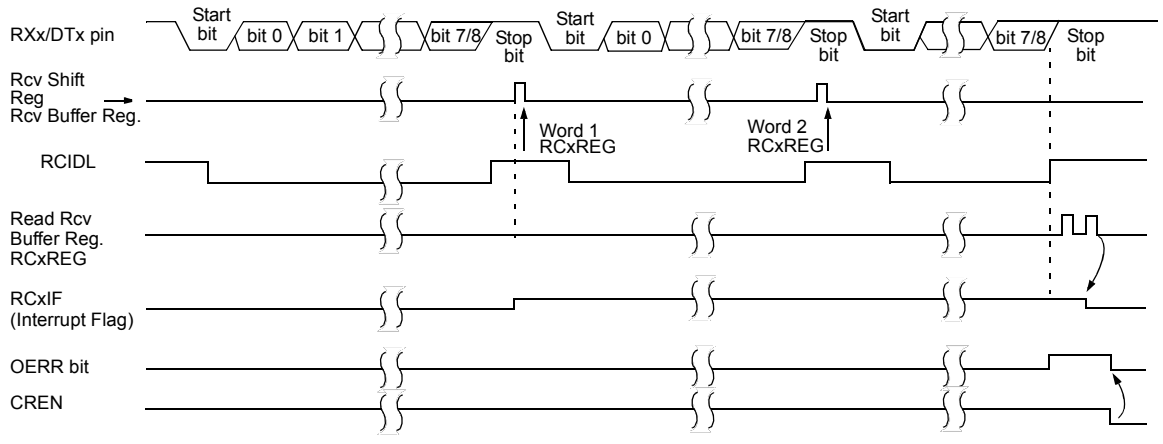
1. Initialize the SPxBRGH:SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 27.4 "EUSART Baud Rate Generator (BRG)"**).
2. Clear the ANSEL bit for the RXx pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCxIE bit of the PIE3 register and the GIE and PEIE bits of the INTCON register.
5. If 9-bit reception is desired, set the RX9 bit.
6. Enable reception by setting the CREN bit.
7. The RCxIF interrupt flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCxIE interrupt enable bit was also set.
8. Read the RCxSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
9. Get the received eight Least Significant data bits from the receive buffer by reading the RCxREG register.
10. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

27.2.2.9 9-Bit Address Detection Mode Setup

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPxBRGH:SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 27.4 "EUSART Baud Rate Generator (BRG)"**).
2. Clear the ANSEL bit for the RXx pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCxIE bit of the PIE3 register and the GIE and PEIE bits of the INTCON register.
5. Enable 9-bit reception by setting the RX9 bit.
6. Enable address detection by setting the ADDEN bit.
7. Enable reception by setting the CREN bit.
8. The RCxIF interrupt flag bit will be set when a character with the ninth bit set is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCxIE interrupt enable bit was also set.
9. Read the RCxSTA register to get the error flags. The ninth data bit will always be set.
10. Get the received eight Least Significant data bits from the receive buffer by reading the RCxREG register. Software determines if this is the device's address.
11. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.
12. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and generate interrupts.

FIGURE 27-5: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RXx input. The RCxREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

27.3 Clock Accuracy with Asynchronous Operation

The factory calibrates the internal oscillator block output (INTOSC). However, the INTOSC frequency may drift as V_{DD} or temperature changes, and this directly affects the asynchronous baud rate. Two methods may be used to adjust the baud rate clock, but both require a reference clock source of some kind.

The first (preferred) method uses the OSCTUNE register to adjust the INTOSC output. Adjusting the value in the OSCTUNE register allows for fine resolution changes to the system clock source. See **Section 4.3.2.3 "Internal Oscillator Frequency Adjustment"** for more information.

The other method adjusts the value in the Baud Rate Generator. This can be done automatically with the Auto-Baud Detect feature (see **Section 27.4.1 "Auto-Baud Detect"**). There may not be fine enough resolution when adjusting the Baud Rate Generator to compensate for a gradual change in the peripheral clock frequency.

31.3 ADC Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 31-4. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), refer to Figure 31-4. **The maximum recommended impedance for analog sources is 10 kΩ.** As the source impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (or changed), an ADC acquisition must be completed before the conversion can be started. To calculate the minimum acquisition time, Equation 31-1 may be used. This equation assumes that 1/2 LSb error is used (1,024 steps for the ADC). The 1/2 LSb error is the maximum error allowed for the ADC to meet its specified resolution.

EQUATION 31-1: ACQUISITION TIME EXAMPLE

Assumptions: $Temperature = 50°C$ *and external impedance of* $10k\Omega$ *5.0V* V_{DD}

$$T_{ACQ} = \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient}$$
$$= T_{AMP} + T_C + T_{COFF}$$
$$= 2\mu s + T_C + [(\text{Temperature} - 25°C)(0.05\mu s/°C)]$$

The value for T_C *can be approximated with the following equations:*

$$V_{APPLIED}\left(1 - \frac{1}{(2^{n+1}) - 1}\right) = V_{CHOLD} \quad ;[1]\ V_{CHOLD}\ \text{charged to within 1/2 lsb}$$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{CHOLD} \quad ;[2]\ V_{CHOLD}\ \text{charge response to}\ V_{APPLIED}$$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{APPLIED}\left(1 - \frac{1}{(2^{n+1}) - 1}\right) \quad ;\text{combining [1] and [2]}$$

Note: Where n = number of bits of the ADC.

Solving for T_C*:*

$$T_C = -C_{HOLD}(R_{IC} + R_{SS} + R_S)\ ln(1/2047)$$
$$= -10pF(1k\Omega + 7k\Omega + 10k\Omega)\ ln(0.0004885)$$
$$= 1.37\mu s$$

Therefore:

$$T_{ACQ} = 2\mu s + 892ns + [(50°C - 25°C)(0.05\mu s/°C)]$$
$$= 4.62\mu s$$

Note 1: The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

2: The charge holding capacitor (CHOLD) is not discharged after each conversion.

3: The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

REGISTER 31-2: ADCON1: ADC CONTROL REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	R/W-0/0
ADPPOL	ADIPEN	ADGPOL	—	—	—	—	ADDSEN
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 **ADDPOL:** Precharge Polarity bit
If ADPRE>0x00:

ADPPOL	**Action During 1st Precharge Stage**	
	External (selected analog I/O pin)	**Internal (AD sampling capacitor)**
1	Shorted to AV_{DD}	C_{HOLD} shorted to V_{SS}
0	Shorted to V_{SS}	C_{HOLD} shorted to AV_{DD}

Otherwise:
The bit is ignored

bit 6 **ADIPEN:** A/D Inverted Precharge Enable bit
If ADDSEN = 1
1 = The precharge and guard signals in the second conversion cycle are the opposite polarity of the first cycle
0 = Both Conversion cycles use the precharge and guards specified by ADPPOL and ADGPOL
Otherwise:
The bit is ignored

bit 5 **ADGPOL:** Guard Ring Polarity Selection bit
1 = ADC guard Ring outputs start as digital high during Precharge stage
0 = ADC guard Ring outputs start as digital low during Precharge stage

bit 4-1 **Unimplemented:** Read as '0'

bit 0 **ADDSEN:** Double-sample enable bit
1 = Two conversions are performed on each trigger. Data from the first conversion appears in ADPREV
0 = One conversion is performed for each trigger

BCF **Bit Clear f**

Syntax: BCF f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: $0 \rightarrow f<b>$

Status Affected: None

Encoding:

1001	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is cleared.
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 $f \le 95$ (5Fh). See **Section 35.2.3 "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: `BCF FLAG_REG, 7, 0`

Before Instruction
FLAG_REG = C7h
After Instruction
FLAG_REG = 47h

BN **Branch if Negative**

Syntax: BN n

Operands: $-128 \le n \le 127$

Operation: if NEGATIVE bit is '1'
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0110	nnnn	nnnn

Description: If the NEGATIVE bit is '1', then the program will branch.
The 2's complement number '2n' is added 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 then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

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

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example:
```
HERE    BN   Jump
```
Before Instruction
PC = address (HERE)
After Instruction
If NEGATIVE = 1;
PC = address (Jump)
If NEGATIVE = 0;
PC = address (HERE + 2)

RCALL **Relative Call**

Syntax: RCALL n

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

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 $\overline{\text{MCLR}}$ Reset.

Status Affected: All

Encoding:

0000	0000	1111	1111

Description: This instruction provides a way to execute a $\overline{\text{MCLR}}$ 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 35.2.3 "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

35.2.5 SPECIAL CONSIDERATIONS WITH MICROCHIP MPLAB® IDE TOOLS

The latest versions of Microchip's software tools have been designed to fully support the extended instruction set of the PIC18(L)F2x/4xK40 family of devices. This includes the MPLAB C18 C compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is '0', disabling the extended instruction set and Indexed Literal Offset Addressing mode. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option, or dialog box within the environment, that allows the user to configure the language tool and its settings for the project
- A command line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.

36.11 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

36.12 Third-Party Development Tools

Microchip also offers a great collection of tools from third-party vendors. These tools are carefully selected to offer good value and unique functionality.

- Device Programmers and Gang Programmers from companies, such as SoftLog and CCS
- Software Tools from companies, such as Gimpel and Trace Systems
- Protocol Analyzers from companies, such as Saleae and Total Phase
- Demonstration Boards from companies, such as MikroElektronika, Digilent® and Olimex
- Embedded Ethernet Solutions from companies, such as EZ Web Lynx, WIZnet and IPLogika®

TABLE 37-14: ANALOG-TO-DIGITAL CONVERTER (ADC) CONVERSION TIMING SPECIFICATIONS

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
AD20	TAD	ADC Clock Period	1	—	9	µs	Using FOSC as the ADC clock source ADOCS = 0
AD21			—	2	—	µs	Using FRC as the ADC clock source ADOCS = 1
AD22	TCNV	Conversion Time(1)	—	11 + 3TCY	—	TAD	Set of GO/$\overline{\text{DONE}}$ bit to Clear of GO/$\overline{\text{DONE}}$ bit
AD23	TACQ	Acquisition Time	—	2	—	µs	
AD24	THCD	Sample and Hold Capacitor Disconnect Time	—	—	—	µs	FOSC-based clock source FRC-based clock source

* 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: Does not apply for the ADCRC oscillator.

FIGURE 37-10: ADC CONVERSION TIMING (ADC CLOCK FOSC-BASED)