



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.

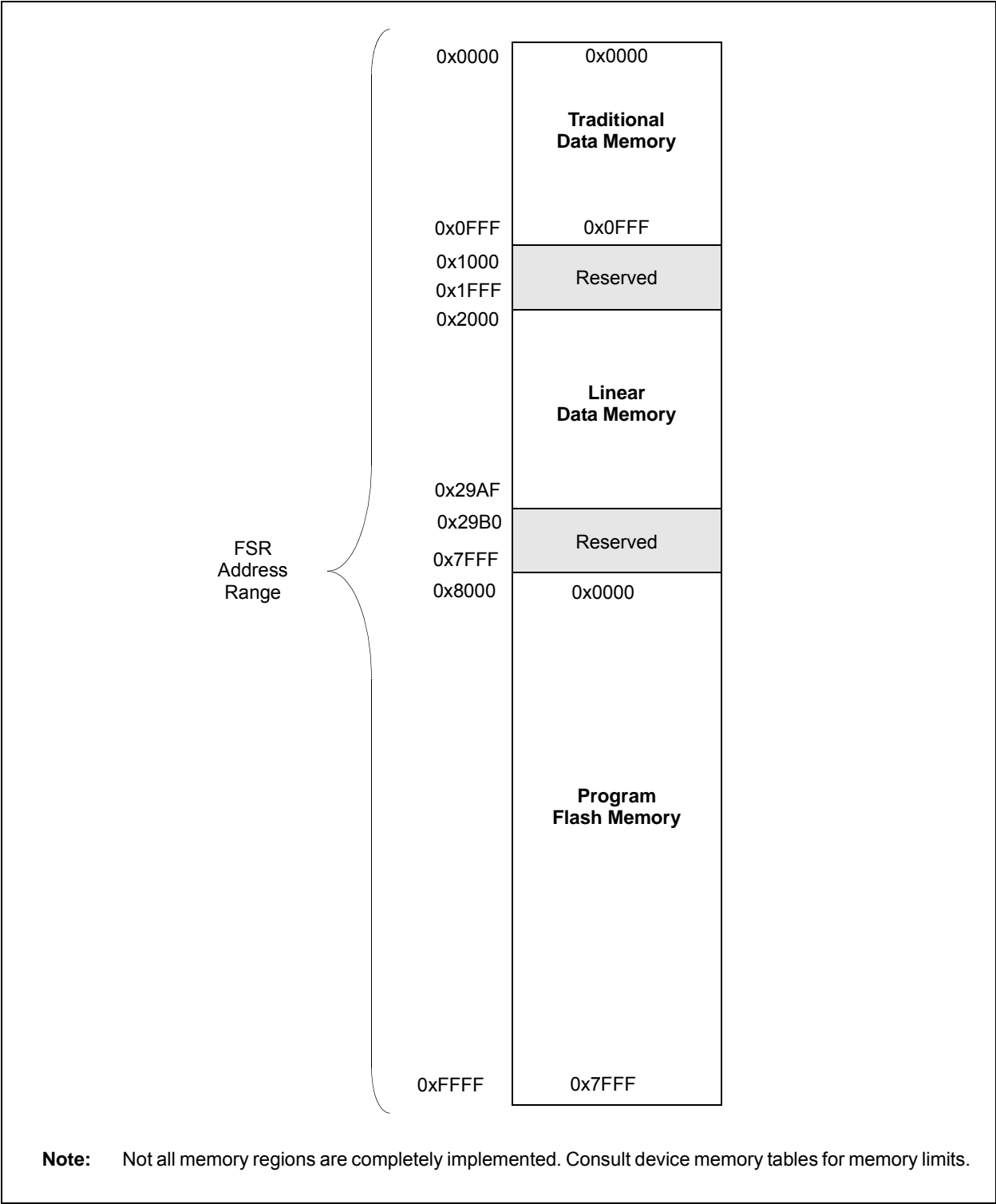
### Applications of "[Embedded - Microcontrollers](#)"

#### Details

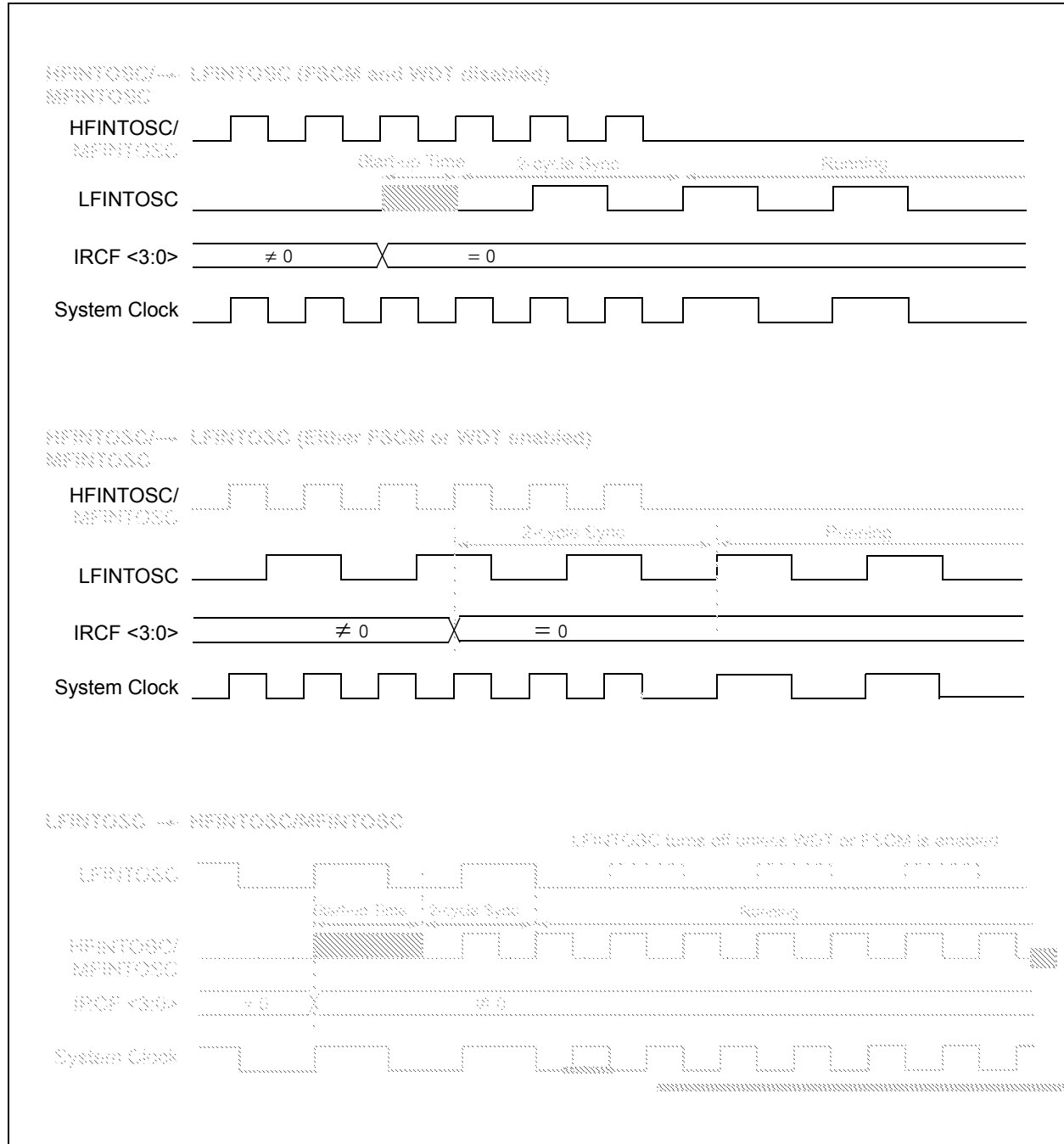
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	32MHz
Connectivity	I <sup>2</sup> C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	25
Program Memory Size	7KB (4K x 14)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 17x10b; D/A 1x5b, 1x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	28-VQFN Exposed Pad
Supplier Device Package	28-QFN (6x6)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/pic16lf1713-e-ml">https://www.e-xfl.com/product-detail/microchip-technology/pic16lf1713-e-ml</a>

# PIC16(L)F1713/6

FIGURE 3-9: INDIRECT ADDRESSING



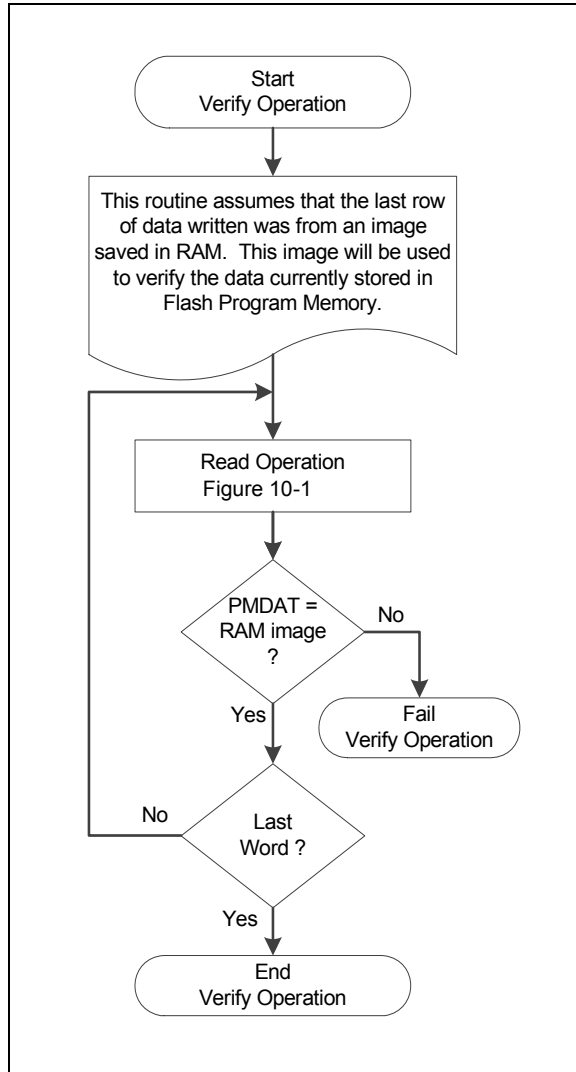
**FIGURE 6-7: INTERNAL OSCILLATOR SWITCH TIMING**



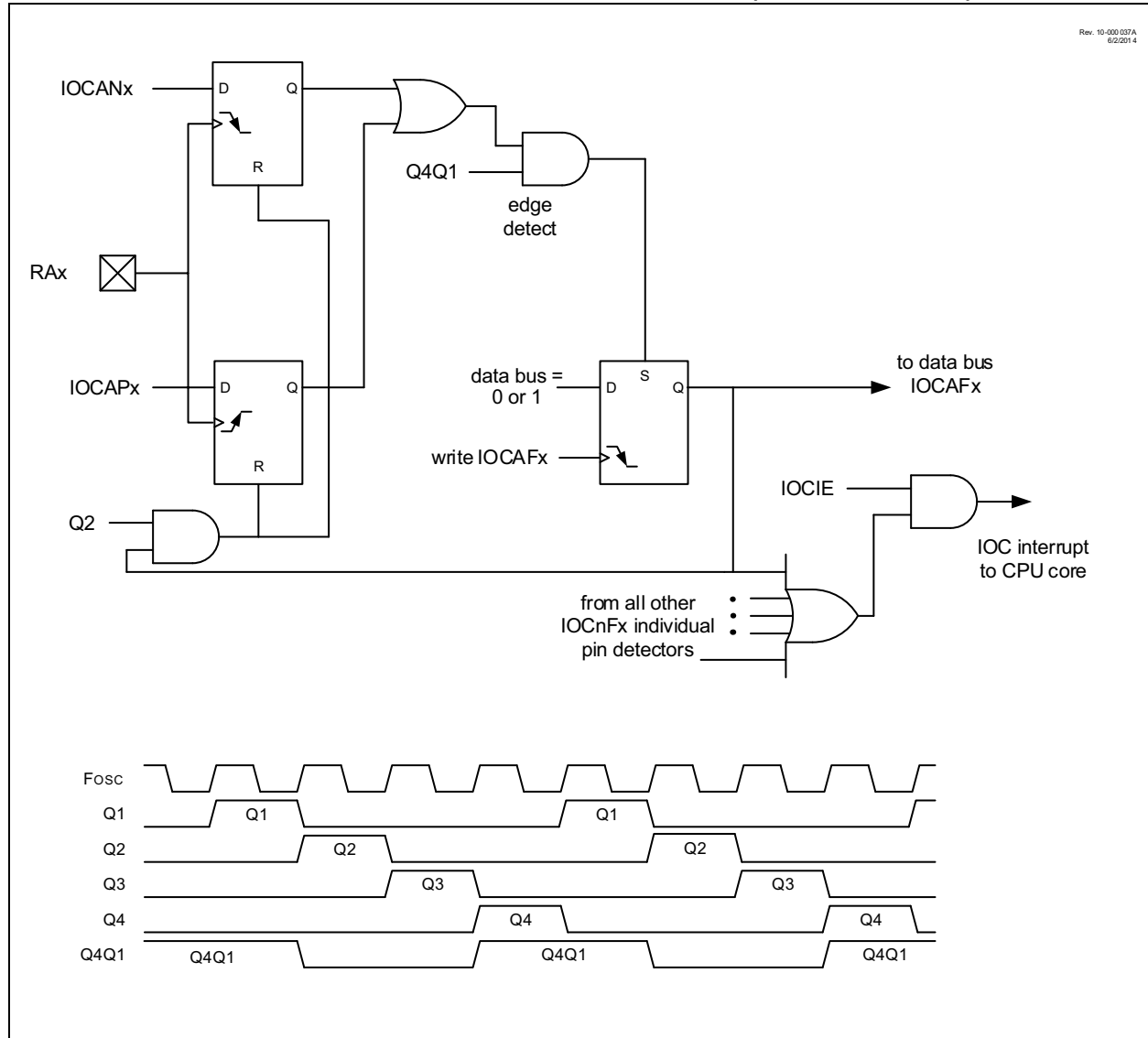
## 10.5 Write Verify

It is considered good programming practice to verify that program memory writes agree with the intended value. Since program memory is stored as a full page then the stored program memory contents are compared with the intended data stored in RAM after the last write is complete.

**FIGURE 10-8: FLASH PROGRAM MEMORY VERIFY FLOWCHART**



**FIGURE 13-1: INTERRUPT-ON-CHANGE BLOCK DIAGRAM (PORTA EXAMPLE)**



## REGISTER 13-7: IOCCP: INTERRUPT-ON-CHANGE PORTC POSITIVE EDGE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
IOCCP7	IOCCP6	IOCCP5	IOCCP4	IOCCP3	IOCCP2	IOCCP1	IOCCP0
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

**IOCCP<7:0>:** Interrupt-on-Change PORTC Positive Edge Enable bits

1 = Interrupt-on-Change enabled on the pin for a positive going edge. IOCCFx bit and IOCIF flag will be set upon detecting an edge.

0 = Interrupt-on-Change disabled for the associated pin.

## REGISTER 13-8: IOCCN: INTERRUPT-ON-CHANGE PORTC NEGATIVE EDGE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
IOCCN7	IOCCN6	IOCCN5	IOCCN4	IOCCN3	IOCCN2	IOCCN1	IOCCN0
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

**IOCCN<7:0>:** Interrupt-on-Change PORTC Negative Edge Enable bits

1 = Interrupt-on-Change enabled on the pin for a negative going edge. IOCCFx bit and IOCIF flag will be set upon detecting an edge.

0 = Interrupt-on-Change disabled for the associated pin.

## REGISTER 18-8: COGxASD1: COG AUTO-SHUTDOWN CONTROL REGISTER 1

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	GxAS3E	GxAS2E	GxAS1E	GxAS0E
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

q = Value depends on condition

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **GxAS3E:** COGx Auto-shutdown Source Enable bit 3

1 = COGx is shutdown when CLC2 output is low

0 = CLC2 output has no effect on shutdown

bit 2 **GxAS2E:** COGx Auto-shutdown Source Enable bit 2

1 = COGx is shutdown when Comparator 2 output is low

0 = Comparator 2 output has no effect on shutdown

bit 1 **GxAS1E:** COGx Auto-shutdown Source Enable bit 1

1 = COGx is shutdown when Comparator 1 output is low

0 = Comparator 1 output has no effect on shutdown

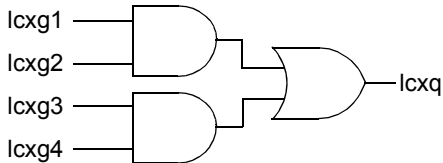
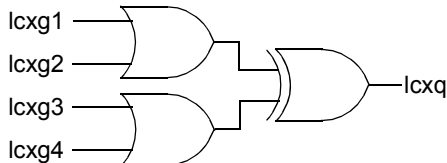
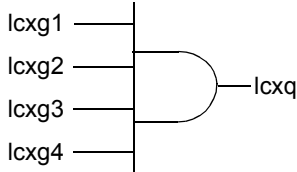
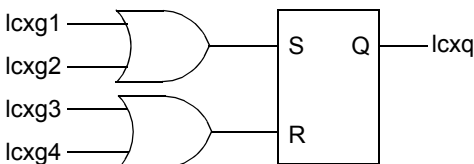
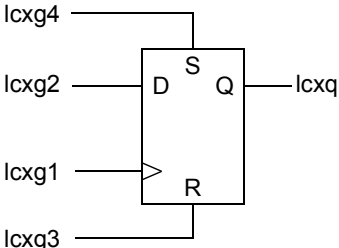
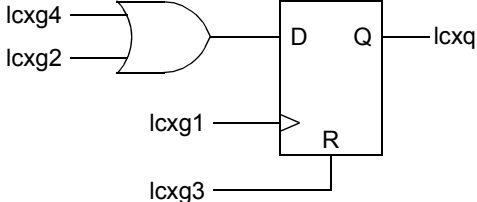
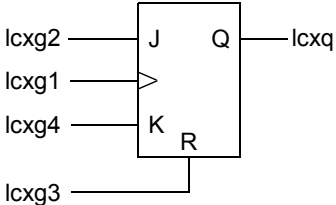
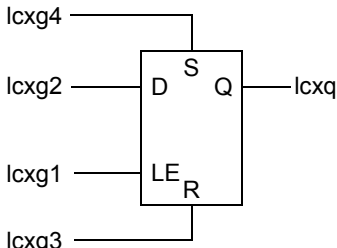
bit 0 **GxAS0E:** COGx Auto-shutdown Source Enable bit 0

1 = COGx is shutdown when Pin selected with COGxPPS control is low

0 = Pin selected with COGxPPS control has no effect on shutdown

# PIC16(L)F1713/6

FIGURE 19-3: PROGRAMMABLE LOGIC FUNCTIONS

<p><b>AND - OR</b></p>  <p><math>LCxMODE&lt;2:0&gt;= 000</math></p>	<p><b>OR - XOR</b></p>  <p><math>LCxMODE&lt;2:0&gt;= 001</math></p>
<p><b>4-Input AND</b></p>  <p><math>LCxMODE&lt;2:0&gt;= 010</math></p>	<p><b>S-R Latch</b></p>  <p><math>LCxMODE&lt;2:0&gt;= 011</math></p>
<p><b>1-Input D Flip-Flop with S and R</b></p>  <p><math>LCxMODE&lt;2:0&gt;= 100</math></p>	<p><b>2-Input D Flip-Flop with R</b></p>  <p><math>LCxMODE&lt;2:0&gt;= 101</math></p>
<p><b>J-K Flip-Flop with R</b></p>  <p><math>LCxMODE&lt;2:0&gt;= 110</math></p>	<p><b>1-Input Transparent Latch with S and R</b></p>  <p><math>LCxMODE&lt;2:0&gt;= 111</math></p>



## REGISTER 20-3: NCOxACCL: NCOx ACCUMULATOR REGISTER – LOW BYTE

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
NCOxACC<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                  **NCOxACC<7:0>**: NCOx Accumulator, Low Byte

## REGISTER 20-4: NCOxACCH: NCOx ACCUMULATOR REGISTER – HIGH BYTE

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
NCOxACC<15:8>							
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                  **NCOxACC<15:8>**: NCOx Accumulator, High Byte

## REGISTER 20-5: NCOxACCU: NCOx ACCUMULATOR REGISTER – UPPER BYTE

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	NCOxACC<19:16>			
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 7-4                  **Unimplemented:** Read as '0'

bit 3-0                  **NCOxACC<19:16>**: NCOx Accumulator, Upper Byte

# PIC16(L)F1713/6

## REGISTER 21-2: ADCON1: ADC CONTROL REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
ADFM	ADCS<2:0>			—	ADNREF	ADPREF<1:0>	
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      **ADFM:** ADC Result Format Select bit  
1 = Right justified. Six Most Significant bits of ADRESH are set to '0' when the conversion result is loaded.  
0 = Left justified. Six Least Significant bits of ADRESL are set to '0' when the conversion result is loaded.
- bit 6-4      **ADCS<2:0>:** ADC Conversion Clock Select bits  
111 = FRC (clock supplied from an internal RC oscillator)  
110 = Fosc/64  
101 = Fosc/16  
100 = Fosc/4  
011 = FRC (clock supplied from an internal RC oscillator)  
010 = Fosc/32  
001 = Fosc/8  
000 = Fosc/2
- bit 3      **Unimplemented:** Read as '0'
- bit 2      **ADNREF:** A/D Negative Voltage Reference Configuration bit  
1 = VREF- is connected to Vref- pin  
0 = VREF- is connected to VSS
- bit 1-0      **ADPREF<1:0>:** ADC Positive Voltage Reference Configuration bits  
11 = VREF+ is connected to internal Fixed Voltage Reference (FVR) module<sup>(1)</sup>  
10 = VREF+ is connected to external VREF+ pin<sup>(1)</sup>  
01 = Reserved  
00 = VREF+ is connected to VDD

**Note 1:** When selecting the VREF+ pin as the source of the positive reference, be aware that a minimum voltage specification exists. See Table 34-16: ADC Conversion Requirements for details.



# PIC16(L)F1713/6

## 29.1 Capture Mode

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

Capture mode makes use of the 16-bit Timer1 resource. When an event occurs on the CCPx pin, the 16-bit CCPRxH:CCPRxL register pair captures and stores the 16-bit value of the TMR1H:TMR1L register pair, respectively. An event is defined as one of the following and is configured by the CCPxM<3:0> bits of the CCPxCON register:

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

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

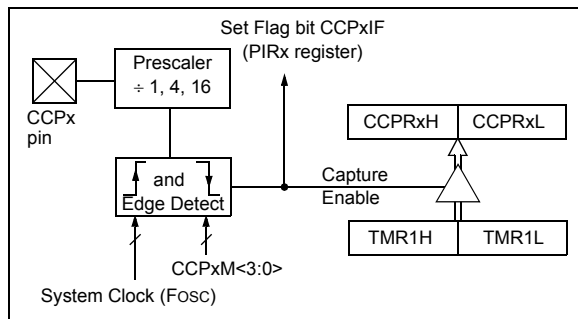
Figure 29-1 shows a simplified diagram of the capture operation.

### 29.1.1 CCP PIN CONFIGURATION

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

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

**FIGURE 29-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**



### 29.1.2 TIMER1 MODE RESOURCE

Timer1 must be running in Timer mode or Synchronized Counter mode for the CCP module to use the capture feature. In Asynchronous Counter mode, the capture operation may not work.

See **Section 26.0 “Timer1 Module with Gate Control”** for more information on configuring Timer1.

### 29.1.3 SOFTWARE INTERRUPT MODE

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

**Note:** Clocking Timer1 from the system clock (Fosc) should not be used in Capture mode. In order for Capture mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock (Fosc/4) or from an external clock source.

### 29.1.4 CCP PRESCALER

There are four prescaler settings specified by the CCPxM<3:0> bits of the CCPxCON register. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. Any Reset will clear the prescaler counter.

Switching from one capture prescaler to another does not clear the prescaler and may generate a false interrupt. To avoid this unexpected operation, turn the module off by clearing the CCPxCON register before changing the prescaler. Example 29-1 demonstrates the code to perform this function.

### EXAMPLE 29-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
BANKSEL CCPxCON    ;Set Bank bits to point
                    ;to CCPxCON
CLRWF  CCPxCON      ;Turn CCP module off
MOVLW  NEW_CAPT_PS  ;Load the W reg with
                    ;the new prescaler
                    ;move value and CCP ON
MOVWF  CCPxCON      ;Load CCPxCON with this
                    ;value
```

## 29.1.5 CAPTURE DURING SLEEP

Capture mode depends upon the Timer1 module for proper operation. There are two options for driving the Timer1 module in Capture mode. It can be driven by the instruction clock ( $F_{osc}/4$ ), or by an external clock source.

When Timer1 is clocked by  $F_{osc}/4$ , Timer1 will not increment during Sleep. When the device wakes from Sleep, Timer1 will continue from its previous state.

Capture mode will operate during Sleep when Timer1 is clocked by an external clock source.

## 29.2 Compare Mode

The Compare mode function described in this section is available and identical for all CCP modules.

Compare mode makes use of the 16-bit Timer1 resource. The 16-bit value of the CCPRxH:CCPRxL register pair is constantly compared against the 16-bit value of the TMR1H:TMR1L register pair. When a match occurs, one of the following events can occur:

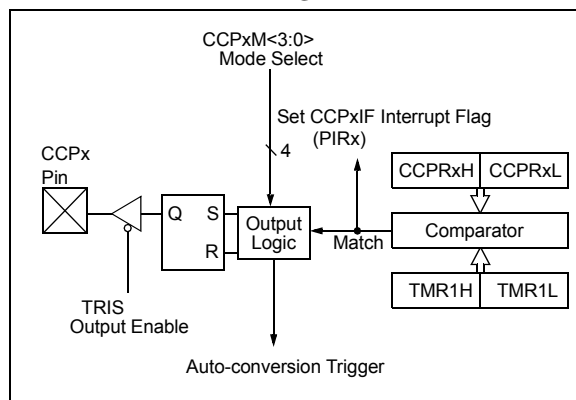
- Toggle the CCPx output
- Set the CCPx output
- Clear the CCPx output
- Generate an Auto-conversion Trigger
- Generate a Software Interrupt

The action on the pin is based on the value of the CCPxM<3:0> control bits of the CCPxCON register. At the same time, the interrupt flag CCPxIF bit is set.

All Compare modes can generate an interrupt.

Figure 29-2 shows a simplified diagram of the compare operation.

**FIGURE 29-2: COMPARE MODE OPERATION BLOCK DIAGRAM**



## 29.2.1 CCPX PIN CONFIGURATION

The user must configure the CCPx pin as an output by clearing the associated TRIS bit.

**Note:** Clearing the CCPxCON register will force the CCPx compare output latch to the default low level. This is not the PORT I/O data latch.

## 29.2.2 TIMER1 MODE RESOURCE

In Compare mode, Timer1 must be running in either Timer mode or Synchronized Counter mode. The compare operation may not work in Asynchronous Counter mode.

See **Section 26.0 “Timer1 Module with Gate Control”** for more information on configuring Timer1.

**Note:** Clocking Timer1 from the system clock ( $F_{osc}$ ) should not be used in Compare mode. In order for Compare mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock ( $F_{osc}/4$ ) or from an external clock source.

## 29.2.3 SOFTWARE INTERRUPT MODE

When Generate Software Interrupt mode is chosen (CCPxM<3:0> = 1010), the CCPx module does not assert control of the CCPx pin (see the CCPxCON register).

## 29.2.4 AUTO-CONVERSION TRIGGER

When Auto-conversion Trigger mode is chosen (CCPxM<3:0> = 1011), the CCPx module does the following:

- Resets Timer1
- Starts an ADC conversion if ADC is enabled

The CCPx module does not assert control of the CCPx pin in this mode.

The Auto-conversion Trigger output of the CCP occurs immediately upon a match between the TMR1H, TMR1L register pair and the CCPRxH, CCPRxL register pair. The TMR1H, TMR1L register pair is not reset until the next rising edge of the Timer1 clock. The Auto-conversion Trigger output starts an ADC conversion (if the ADC module is enabled). This allows the CCPRxH, CCPRxL register pair to effectively provide a 16-bit programmable period register for Timer1.

## 30.2 SPI Mode Overview

The Serial Peripheral Interface (SPI) bus is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is controlled through a Chip Select known as Slave Select.

The SPI bus specifies four signal connections:

- Serial Clock (SCK)
- Serial Data Out (SDO)
- Serial Data In (SDI)
- Slave Select ( $\overline{SS}$ )

Figure 30-1 shows the block diagram of the MSSP module when operating in SPI mode.

The SPI bus operates with a single master device and one or more slave devices. When multiple slave devices are used, an independent Slave Select connection is required from the master device to each slave device.

Figure 30-4 shows a typical connection between a master device and multiple slave devices.

The master selects only one slave at a time. Most slave devices have tri-state outputs so their output signal appears disconnected from the bus when they are not selected.

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 30-5 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.

## 30.5.3.3 7-bit Transmission with Address Hold Enabled

Setting the AHEN bit of the SSPCON3 register enables additional clock stretching and interrupt generation after the eighth falling edge of a received matching address. Once a matching address has been clocked in, CKP is cleared and the SSPIF interrupt is set.

Figure 30-19 displays a standard waveform of a 7-bit address slave transmission with AHEN enabled.

1. Bus starts Idle.
2. Master sends Start condition; the S bit of SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Master sends matching address with  $\overline{R/W}$  bit set. After the eighth falling edge of the SCL line the CKP bit is cleared and SSPIF interrupt is generated.
4. Slave software clears SSPIF.
5. Slave software reads ACKTIM bit of SSPCON3 register, and  $\overline{R/W}$  and  $\overline{D/A}$  of the SSPSTAT register to determine the source of the interrupt.
6. Slave reads the address value from the SSPBUF register clearing the BF bit.
7. Slave software decides from this information if it wishes to ACK or not ACK and sets the ACKDT bit of the SSPCON2 register accordingly.
8. Slave sets the CKP bit releasing SCL.
9. Master clocks in the  $\overline{ACK}$  value from the slave.
10. Slave hardware automatically clears the CKP bit and sets SSPIF after the ACK if the  $\overline{R/W}$  bit is set.
11. Slave software clears SSPIF.
12. Slave loads value to transmit to the master into SSPBUF setting the BF bit.

**Note:** SSPBUF cannot be loaded until after the  $\overline{ACK}$ .

13. Slave sets the CKP bit releasing the clock.
14. Master clocks out the data from the slave and sends an  $\overline{ACK}$  value on the 9th SCL pulse.
15. Slave hardware copies the  $\overline{ACK}$  value into the ACKSTAT bit of the SSPCON2 register.
16. Steps 10-15 are repeated for each byte transmitted to the master from the slave.
17. If the master sends a not  $\overline{ACK}$  the slave releases the bus allowing the master to send a Stop and end the communication.

**Note:** Master must send a not  $\overline{ACK}$  on the last byte to ensure that the slave releases the SCL line to receive a Stop.

## 31.5 EUSART Synchronous Mode

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

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

Start and Stop bits are not used in synchronous transmissions.

### 31.5.1 SYNCHRONOUS MASTER MODE

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

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

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

#### 31.5.1.1 Master Clock

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

#### 31.5.1.2 Clock Polarity

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

#### 31.5.1.3 Synchronous Master Transmission

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

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

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

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

#### 31.5.1.4 Synchronous Master Transmission Setup:

1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 31.4 “EUSART Baud Rate Generator (BRG)”**).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Disable Receive mode by clearing bits SREN and CREN.
4. Enable Transmit mode by setting the TXEN bit.
5. If 9-bit transmission is desired, set the TX9 bit.
6. If interrupts are desired, set the TXIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
7. If 9-bit transmission is selected, the ninth bit should be loaded in the TX9D bit.
8. Start transmission by loading data to the TXREG register.



## 31.5.2 SYNCHRONOUS SLAVE MODE

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

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

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

### 31.5.2.1 EUSART Synchronous Slave Transmit

The operation of the Synchronous Master and Slave modes are identical (see **Section 31.5.1.3 “Synchronous Master Transmission”**), except in the case of the Sleep mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

1. The first character will immediately transfer to the TSR register and transmit.
2. The second word will remain in the TXREG register.
3. The TXIF bit will not be set.
4. After the first character has been shifted out of TSR, the TXREG register will transfer the second character to the TSR and the TXIF bit will now be set.
5. If the PEIE and TXIE bits are set, the interrupt will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will call the Interrupt Service Routine.

### 31.5.2.2 Synchronous Slave Transmission Setup:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the ANSEL bit for the CK pin (if applicable).
3. Clear the CREN and SREN bits.
4. If interrupts are desired, set the TXIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
5. If 9-bit transmission is desired, set the TX9 bit.
6. Enable transmission by setting the TXEN bit.
7. If 9-bit transmission is selected, insert the Most Significant bit into the TX9D bit.
8. Start transmission by writing the Least Significant eight bits to the TXREG register.

# PIC16(L)F1713/6

## 31.5.2.3 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical (**Section 31.5.1.5 “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.

## 31.5.2.4 Synchronous Slave Reception Setup:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the ANSEL bit for both the CK and DT pins (if applicable).
3. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
4. If 9-bit reception is desired, set the RX9 bit.
5. Set the CREN bit to enable reception.
6. The RCIF bit will be set when reception is complete. An interrupt will be generated if the RCIE bit was set.
7. If 9-bit mode is enabled, retrieve the Most Significant bit from the RX9D bit of the RCSTA register.
8. Retrieve the eight Least Significant bits from the receive FIFO by reading the RCREG register.
9. 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 31-10: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION**

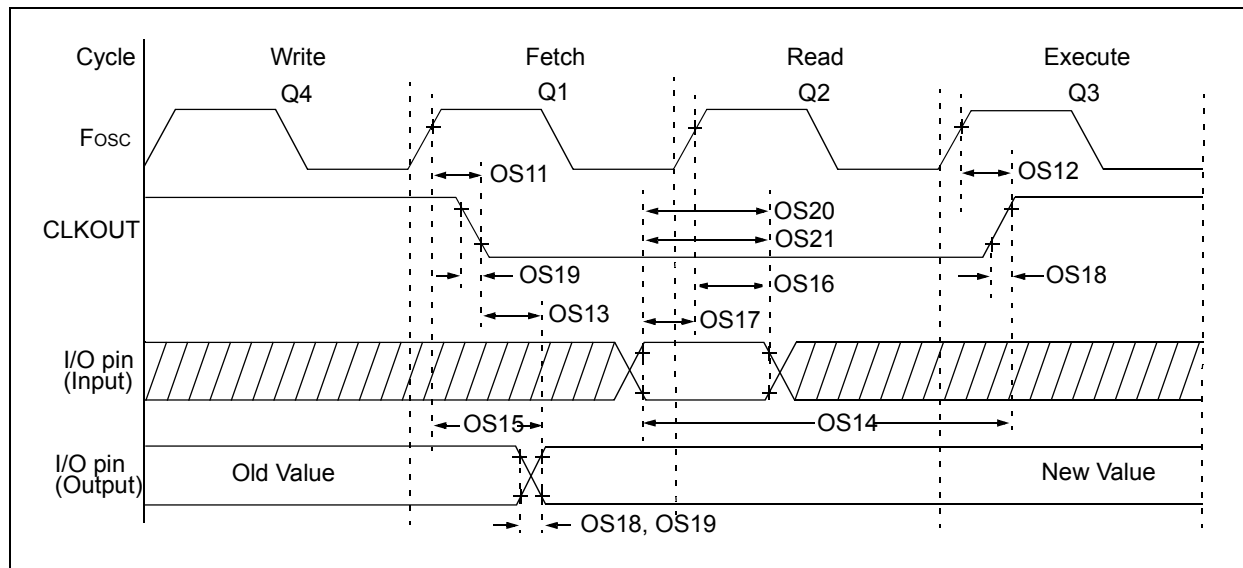
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ANSELB	—	—	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0	126
ANSELC	ANSC7	ANSC6	ANSC5	ANSC4	ANSC3	ANSC2	—	—	131
BAUD1CON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	349
CKPPS	—	—	—	CKPPS<4:0>					136
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	83
PIE1	TMR1GIE	ADIE	RCIE	TXIE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	84
PIR1	TMR1GIF	ADIF	RCIF	TXIF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	87
RC1REG	EUSART Receive Data Register								342*
RC1STA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	348
RXPPS	—	—	—	RXPPS<4:0>					136
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	125
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISA0	130
TX1STA	CSRC	TX9	TXEN	SYNC	SEnDB	BRGH	TRMT	TX9D	347

**Legend:** — = unimplemented location, read as ‘0’. Shaded cells are not used for synchronous slave reception.

\* Page provides register information.

# PIC16(L)F1713/6

**FIGURE 34-7: CLKOUT AND I/O TIMING**



**TABLE 34-10: CLKOUT AND I/O TIMING PARAMETERS**

Standard Operating Conditions (unless otherwise stated)							
Param No.	Sym.	Characteristic	Min.	Typ†	Max.	Units	Conditions
OS11	TosH2ckL	Fosc↑ to CLKOUT↓ <sup>(1)</sup>	—	—	70	ns	3.3V ≤ VDD ≤ 5.0V
OS12	TosH2ckH	Fosc↑ to CLKOUT↑ <sup>(1)</sup>	—	—	72	ns	3.3V ≤ VDD ≤ 5.0V
OS13	TckL2ioV	CLKOUT↓ to Port out valid <sup>(1)</sup>	—	—	20	ns	
OS14	TioV2ckH	Port input valid before CLKOUT↑ <sup>(1)</sup>	Tosc + 200 ns	—	—	ns	
OS15	TosH2ioV	Fosc↑ (Q1 cycle) to Port out valid	—	50	70*	ns	3.3V ≤ VDD ≤ 5.0V
OS16	TosH2ioI	Fosc↑ (Q2 cycle) to Port input invalid (I/O in hold time)	50	—	—	ns	3.3V ≤ VDD ≤ 5.0V
OS17	TioV2osH	Port input valid to Fosc↑ (Q2 cycle) (I/O in setup time)	20	—	—	ns	
OS18*	TioR	Port output rise time <sup>(2)</sup>	—	40 15	72 32	ns	VDD = 1.8V 3.3V ≤ VDD ≤ 5.0V
OS19*	TioF	Port output fall time <sup>(2)</sup>	—	28 15	55 30	ns	VDD = 1.8V 3.3V ≤ VDD ≤ 5.0V
OS20*	Tinp	INT pin input high or low time	25	—	—	ns	
OS21*	Tioc	Interrupt-on-change new input level time	25	—	—	ns	

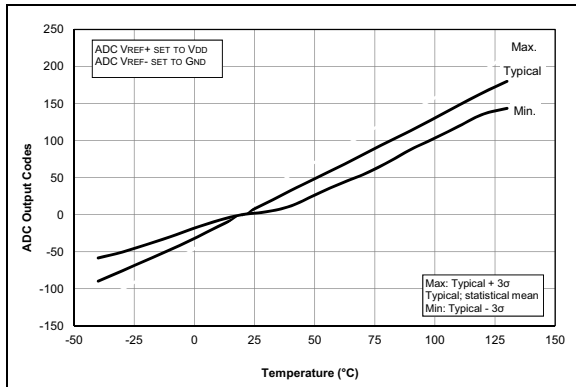
\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated.

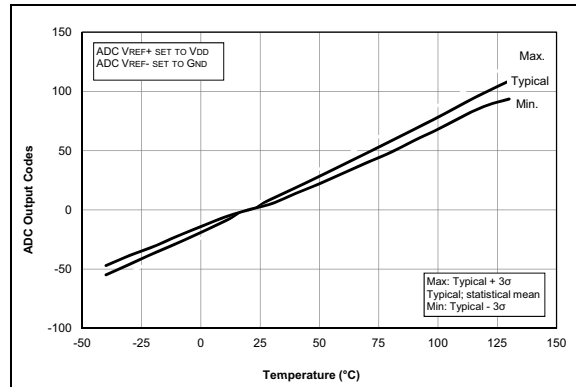
**Note 1:** Measurements are taken in EXTRC mode where CLKOUT output is 4 x Tosc.

**Note 2:** Slew rate limited.

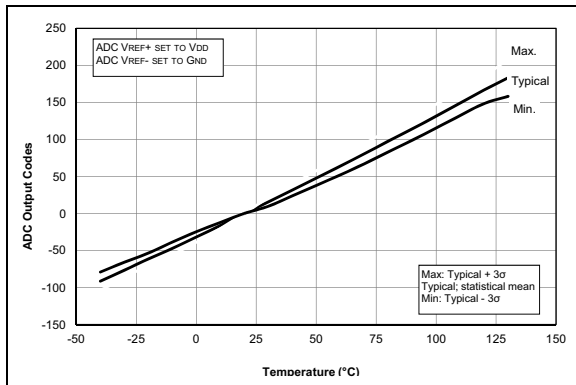
**Note:** Unless otherwise noted,  $V_{IN} = 5V$ ,  $F_{OSC} = 300\text{ kHz}$ ,  $C_{IN} = 0.1\text{ }\mu\text{F}$ ,  $T_A = 25^\circ\text{C}$ .



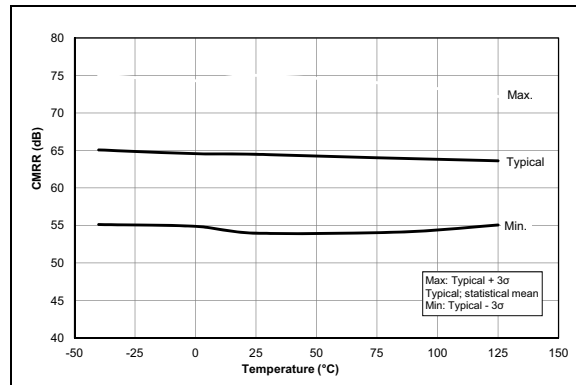
**FIGURE 35-91:** Temp. Indicator Slope Normalized to  $20^\circ\text{C}$ , Low Range,  $V_{DD} = 1.8V$ , PIC16LF1713/6 Only.



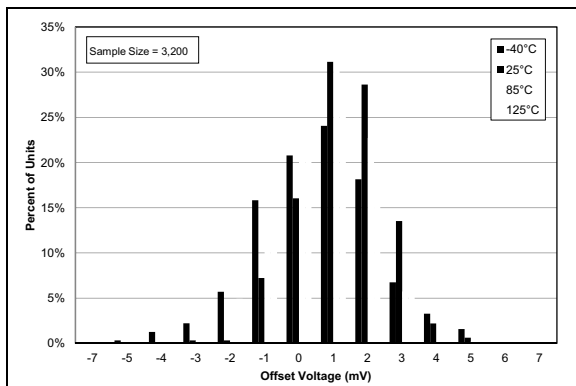
**FIGURE 35-92:** Temp. Indicator Slope Normalized to  $20^\circ\text{C}$ , Low Range,  $V_{DD} = 3.0V$ , PIC16LF1713/6 Only.



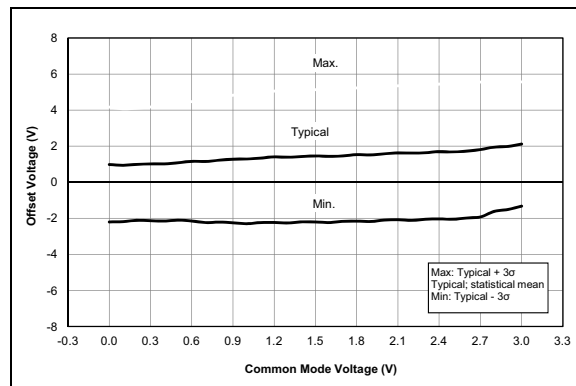
**FIGURE 35-93:** Temp. Indicator Slope Normalized to  $20^\circ\text{C}$ , High Range,  $V_{DD} = 3.6V$ , PIC16LF1713/6 Only.



**FIGURE 35-94:** Op Amp, Common Mode Rejection Ratio (CMRR),  $V_{DD} = 3.0V$ .



**FIGURE 35-95:** Op Amp, Input Offset Voltage Histogram,  $V_{DD} = 3.0V$ ,  $V_{CM} = V_{DD}/2$ .



**FIGURE 35-96:** Op Amp, Offset Over Common Mode Voltage,  $V_{DD} = 3.0V$ , Temp. =  $25^\circ\text{C}$ .

## 36.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers (MCU) and dsPIC® digital signal controllers (DSC) are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® X IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB XC Compiler
  - MPASM™ Assembler
  - MPLINK™ Object Linker/  
MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for  
Various Device Families
- Simulators
  - MPLAB X SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers/Programmers
  - MPLAB ICD 3
  - PICKit™ 3
- Device Programmers
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards,  
Evaluation Kits and Starter Kits
- Third-party development tools

## 36.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:

- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker