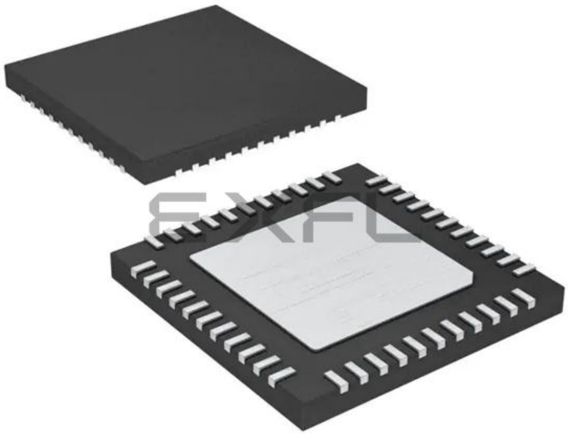


Welcome to E-XFL.COM

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, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	35
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1.5K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 30x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VQFN Exposed Pad
Supplier Device Package	44-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf45k22-i-ml

TABLE 5-2: REGISTER FILE SUMMARY FOR PIC18(L)F2X/4XK22 DEVICES (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
F3Ah	ANSELC	ANSC7	ANSC6	ANSC5	ANSC4	ANSC3	ANSC2	—	—	1111 11--
F39h	ANSELB	—	—	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0	--11 1111
F38h	ANSELA	—	—	ANSA5	—	ANSA3	ANSA2	ANSA1	ANSA0	--1- 1111

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: PIC18(L)F4XK22 devices only.
2: PIC18(L)F2XK22 devices only.
3: PIC18(L)F23/24K22 and PIC18(L)F43/44K22 devices only.
4: PIC18(L)F26K22 and PIC18(L)F46K22 devices only.

5.6 Data Addressing Modes

Note:	The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See **Section 5.7 "Data Memory and the Extended Instruction Set"** for more information.

While the program memory can be addressed in only one way – through the program counter – information in the data memory space can be addressed in several ways. For most instructions, the addressing mode is fixed. Other instructions may use up to three modes, depending on which operands are used and whether or not the extended instruction set is enabled.

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = `1`). Its operation is discussed in greater detail in **Section 5.7.1 "Indexed Addressing with Literal Offset"**.

5.6.1 INHERENT AND LITERAL ADDRESSING

Many PIC18 control instructions do not need any argument at all; they either perform an operation that globally affects the device or they operate implicitly on one register. This addressing mode is known as Inherent Addressing. Examples include `SLEEP`, `RESET` and `DAW`.

Other instructions work in a similar way but require an additional explicit argument in the opcode. This is known as Literal Addressing mode because they require some literal value as an argument. Examples include `ADDLW` and `MOVLW`, which respectively, add or move a literal value to the W register. Other examples include `CALL` and `GOTO`, which include a 20-bit program memory address.

5.6.2 DIRECT ADDRESSING

Direct addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of direct addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (**Section 5.4.3 "General Purpose Register File"**) or a location in the Access Bank (**Section 5.4.2 "Access Bank"**) as the data source for the instruction.

The Access RAM bit 'a' determines how the address is interpreted. When 'a' is '`1`', the contents of the BSR (**Section 5.4.1 "Bank Select Register (BSR)"**) are used with the address to determine the complete 12-bit address of the register. When 'a' is '`0`', the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as `MOVFF`, include the entire 12-bit address (either source or destination) in their opcodes. In these cases, the BSR is ignored entirely.

The destination of the operation's results is determined by the destination bit 'd'. When 'd' is '`1`', the results are stored back in the source register, overwriting its original contents. When 'd' is '`0`', the results are stored in the W register. Instructions without the 'd' argument have a destination that is implicit in the instruction; their destination is either the target register being operated on or the W register.

5.6.3 INDIRECT ADDRESSING

Indirect addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations which are to be read or written. Since the FSRs are themselves located in RAM as Special File Registers, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures, such as tables and arrays in data memory.

The registers for indirect addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code, using loops, such as the example of clearing an entire RAM bank in Example 5-5.

EXAMPLE 5-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
        LFSR    FSR0, 100h ;
NEXT    CLRF    POSTINC0   ; Clear INDF
                           ; register then
                           ; inc pointer
        BTFSS   FSR0H, 1   ; All done with
                           ; Bank1?
        BRA     NEXT       ; NO, clear next
CONTINUE                   ; YES, continue
```

EXAMPLE 6-3: WRITING TO FLASH PROGRAM MEMORY (CONTINUED)

```
                 DECFSZ  COUNTER              ; loop until holding registers are full
                 BRA     WRITE_WORD_TO_HREGS
PROGRAM_MEMORY
                 BSF     EECON1, EEPGD        ; point to Flash program memory
                 BCF     EECON1, CFGS         ; access Flash program memory
                 BSF     EECON1, WREN         ; enable write to memory
                 BCF     INTCON, GIE          ; disable interrupts
                 MOVLW   55h
    Required     MOVWF   EECON2               ; write 55h
    Sequence     MOVLW   0AAh
                 MOVWF   EECON2               ; write 0AAh
                 BSF     EECON1, WR           ; start program (CPU stall)
                 DCFSZ   COUNTER2             ; repeat for remaining write blocks
                 BRA     WRITE_BYTE_TO_HREGS  ;
                 BSF     INTCON, GIE          ; re-enable interrupts
                 BCF     EECON1, WREN         ; disable write to memory
```

6.6.2 WRITE VERIFY

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

6.6.3 UNEXPECTED TERMINATION OF WRITE OPERATION

If a write is terminated by an unplanned event, such as loss of power or an unexpected Reset, the memory location just programmed should be verified and reprogrammed if needed. If the write operation is interrupted by a $\overline{\text{MCLR}}$ Reset or a WDT Time-out Reset during normal operation, the WRERR bit will be set which the user can check to decide whether a rewrite of the location(s) is needed.

6.6.4 PROTECTION AGAINST SPURIOUS WRITES

To protect against spurious writes to Flash program memory, the write initiate sequence must also be followed. See **Section 24.0 “Special Features of the CPU”** for more detail.

6.7 Flash Program Operation During Code Protection

See **Section 24.5 “Program Verification and Code Protection”** for details on code protection of Flash program memory.

TABLE 6-2: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TBLPTRU	—	—	Program Memory Table Pointer Upper Byte (TBLPTR<21:16>)						—
TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								—
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								—
TABLAT	Program Memory Table Latch								—
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	109
EECON2	EEPROM Control Register 2 (not a physical register)								—
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	92
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCL1IP	HLVDIP	TMR3IP	CCP2IP	122
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCL1IF	HLVDIF	TMR3IF	CCP2IF	113
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCL1IE	HLVDIE	TMR3IE	CCP2IE	118

Legend: — = unimplemented, read as ‘0’. Shaded bits are not used during Flash/EEPROM access.

TABLE 10-1: PORTA I/O SUMMARY

Pin Name	Function	TRIS Setting	ANSEL Setting	Pin Type	Buffer Type	Description
RA0/C12IN0-/AN0	RA0	0	0	O	DIG	LATA<0> data output; not affected by analog input.
		1	0	I	TTL	PORTA<0> data input; disabled when analog input enabled.
	C12IN0-	1	1	I	AN	Comparators C1 and C2 inverting input.
	AN0	1	1	I	AN	Analog input 0.
RA1/C12IN1-/AN1	RA1	0	0	O	DIG	LATA<1> data output; not affected by analog input.
		1	0	I	TTL	PORTA<1> data input; disabled when analog input enabled.
	C12IN1-	1	1	I	AN	Comparators C1 and C2 inverting input.
	AN1	1	1	I	AN	Analog input 1.
RA2/C2IN+/AN2/DACOUT/VREF-	RA2	0	0	O	DIG	LATA<2> data output; not affected by analog input; disabled when DACOUT enabled.
		1	0	I	TTL	PORTA<2> data input; disabled when analog input enabled; disabled when DACOUT enabled.
	C2IN+	1	1	I	AN	Comparator C2 non-inverting input.
	AN2	1	1	I	AN	Analog output 2.
	DACOUT	x	1	O	AN	DAC Reference output.
	VREF-	1	1	I	AN	A/D reference voltage (low) input.
RA3/C1IN+/AN3/VREF+	RA3	0		O	DIG	LATA<3> data output; not affected by analog input.
		1	0	I	TTL	PORTA<3> data input; disabled when analog input enabled.
	C1IN+	1	1	I	AN	Comparator C1 non-inverting input.
	AN3	1	1	I	AN	Analog input 3.
	VREF+	1	1	I	AN	A/D reference voltage (high) input.
RA4/CCP5/C1OUT/SRQ/T0CKI	RA4	0	—	O	DIG	LATA<4> data output.
		1	—	I	ST	PORTA<4> data input; default configuration on POR.
	CCP5	0	—	O	DIG	CCP5 Compare output/PWM output, takes priority over RA4 output.
		1	—	I	ST	Capture 5 input/Compare 5 output/ PWM 5 output.
	C1OUT	0	—	O	DIG	Comparator C1 output.
	SRQ	0	—	O	DIG	SR latch Q output; take priority over CCP 5 output.
	T0CKI	1	—	I	ST	Timer0 external clock input.
RA5/C2OUT/SRNQ/$\overline{SS1}$/HLVDIN/AN4	RA5	0	0	O	DIG	LATA<5> data output; not affected by analog input.
		1	0	I	TTL	PORTA<5> data input; disabled when analog input enabled.
	C2OUT	0	0	O	DIG	Comparator C2 output.
	SRNQ	0	0	O	DIG	SR latch $\overline{Q}$ output.
	$\overline{SS1}$	1	0	I	TTL	SPI slave select input (MSSP1).
	HLVDIN	1	1	I	AN	High/Low-Voltage Detect input.
	AN4	1	1	I	AN	A/D input 4.
RA6/CLKO/OSC2	RA6	0	—	O	DIG	LATA<6> data output; enabled in INTOSC modes when CLKO is not enabled.
		1	—	I	TTL	PORTA<6> data input; enabled in INTOSC modes when CLKO is not enabled.
	CLKO	x	—	O	DIG	In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
	OSC2	x	—	O	XTAL	Oscillator crystal output; connects to crystal or resonator in Crystal Oscillator mode.
RA7/CLKI/OSC1	RA7	0	—	O	DIG	LATA<7> data output; disabled in external oscillator modes.
		1	—	I	TTL	PORTA<7> data input; disabled in external oscillator modes.
	CLKI	x	—	I	AN	External clock source input; always associated with pin function OSC1.
	OSC1	x	—	I	XTAL	Oscillator crystal input or external clock source input ST buffer when configured in RC mode; CMOS otherwise.

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

TABLE 10-12: REGISTERS ASSOCIATED WITH PORTD

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ANSELD(1)	ANSD7	ANSD6	ANSD5	ANSD4	ANSD3	ANSD2	ANSD1	ANSD0	150
BAUDCON2	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	271
CCP1CON	P1M<1:0>		DC1B<1:0>		CCP1M<3:0>				198
CCP2CON	P2M<1:0>		DC2B<1:0>		CCP2M<3:0>				198
CCP4CON	—	—	DC4B<1:0>		CCP4M<3:0>				198
LATD(1)	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0	152
PORTD(1)	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	148
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	270
SLRCON(1)	—	—	—	SLRE	SLRD	SLRC	SLRB	SLRA	153
SSP2CON1	WCOL	SSPOV	SSPEN	CKP	SSPM<3:0>				253
TRISD(1)	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	151

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

Note 1: Available on PIC18(L)F4XK22 devices.

TABLE 10-13: CONFIGURATION REGISTERS ASSOCIATED WITH PORTD

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
CONFIG3H	MCLRE	—	P2BMX	T3CMX	HFOFST	CCP3MX	PBADEN	CCP2MX	348

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

14.2 Compare Mode

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

Compare mode makes use of the 16-bit TimerX resources, Timer1, Timer3 and Timer5. The 16-bit value of the CCPRxH:CCPRxL register pair is constantly compared against the 16-bit value of the TMRxH:TMRxL register pair. When a match occurs, one of the following events can occur:

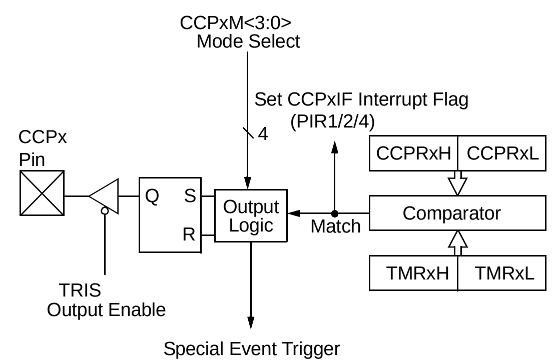
- Toggle the CCPx output
- Set the CCPx output
- Clear the CCPx output
- Generate a Special Event Trigger
- Generate a Software Interrupt

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 14-2 shows a simplified diagram of the Compare operation.

FIGURE 14-2: COMPARE MODE OPERATION BLOCK DIAGRAM



Special Event Trigger function on

- ECCP1, ECCP2, ECCP3, CCP4 and CCP5 will:
  - Reset TimerX – TMRxH:TMRxL = 0x0000
  - TimerX Interrupt Flag, (TMRxIF) is not set

Additional Function on

- CCP5 will
  - Set ADCON0<1>, GO/$\overline{\text{DONE}}$ bit to start an ADC Conversion if ADCON<0>, ADON = 1.

14.2.1 CCP PIN CONFIGURATION

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

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

Note:	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.

14.2.2 TimerX MODE RESOURCE

In Compare mode, 16-bit TimerX resource must be running in either Timer mode or Synchronized Counter mode. The compare operation may not work in Asynchronous Counter mode.

See **Section 12.0 “Timer1/3/5 Module with Gate Control”** for more information on configuring the 16-bit TimerX resources.

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

14.2.3 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).

REGISTER 14-2: CCPxCON: ENHANCED CCPx CONTROL REGISTER (CONTINUED)

bit 3-0 **CCPxM<3:0>:** ECCPx Mode Select bits

`0000` = Capture/Compare/PWM off (resets the module)
`0001` = Reserved
`0010` = Compare mode: toggle output on match
`0011` = Reserved

`0100` = Capture mode: every falling edge
`0101` = Capture mode: every rising edge
`0110` = Capture mode: every 4th rising edge
`0111` = Capture mode: every 16th rising edge

`1000` = Compare mode: set output on compare match (CCPx pin is set, CCPxIF is set)
`1001` = Compare mode: clear output on compare match (CCPx pin is cleared, CCPxIF is set)
`1010` = Compare mode: generate software interrupt on compare match (CCPx pin is unaffected, CCPxIF is set)
`1011` = Compare mode: Special Event Trigger (CCPx pin is unaffected, CCPxIF is set)
TimerX is reset

Half-Bridge ECCP Modules(1):
`1100` = PWM mode: PxA active-high; PxB active-high
`1101` = PWM mode: PxA active-high; PxB active-low
`1110` = PWM mode: PxA active-low; PxB active-high
`1111` = PWM mode: PxA active-low; PxB active-low

Full-Bridge ECCP Modules(1):
`1100` = PWM mode: PxA, PxC active-high; PxB, PxD active-high
`1101` = PWM mode: PxA, PxC active-high; PxB, PxD active-low
`1110` = PWM mode: PxA, PxC active-low; PxB, PxD active-high
`1111` = PWM mode: PxA, PxC active-low; PxB, PxD active-low

Note 1: See Table 14-1 to determine full-bridge and half-bridge ECCPs for the device being used.

REGISTER 14-6: PWMxCON: ENHANCED PWM CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PxRSEN	PxDC<6:0>						
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 **PxRSEN:** PWM Restart Enable bit
`1` = Upon auto-shutdown, the CCPxASE bit clears automatically once the shutdown event goes away; the PWM restarts automatically
`0` = Upon auto-shutdown, CCPxASE must be cleared in software to restart the PWM

bit 6-0 **PxDC<6:0>:** PWM Delay Count bits
PxDCx = Number of FOSC/4 (4 * TOSC) cycles between the scheduled time when a PWM signal **should** transition active and the **actual** time it transitions active

REGISTER 14-7: PSTRxCON: PWM STEERING CONTROL REGISTER(1)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
—	—	—	STRxSYNC	STRxD	STRxC	STRxB	STRxA
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 7-5 **Unimplemented:** Read as ‘`0`’

bit 4 **STRxSYNC:** Steering Sync bit
`1` = Output steering update occurs on next PWM period
`0` = Output steering update occurs at the beginning of the instruction cycle boundary

bit 3 **STRxD:** Steering Enable bit D
`1` = PxD pin has the PWM waveform with polarity control from CCPxM<1:0>
`0` = PxD pin is assigned to port pin

bit 2 **STRxC:** Steering Enable bit C
`1` = PxC pin has the PWM waveform with polarity control from CCPxM<1:0>
`0` = PxC pin is assigned to port pin

bit 1 **STRxB:** Steering Enable bit B
`1` = PxB pin has the PWM waveform with polarity control from CCPxM<1:0>
`0` = PxB pin is assigned to port pin

bit 0 **STRxA:** Steering Enable bit A
`1` = PxA pin has the PWM waveform with polarity control from CCPxM<1:0>
`0` = PxA pin is assigned to port pin

Note 1: The PWM Steering mode is available only when the CCPxCON register bits CCPxM<3:2> = `11` and PxM<1:0> = `00`.

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

The MSSPx consists of a transmit/receive shift register (SSPxSR) and a buffer register (SSPxBUF). The SSPxSR shifts the data in and out of the device, MSb first. The SSPxBUF holds the data that was written to the SSPxSR until the received data is ready. Once the 8 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.

When the application software is expecting to receive valid data, the SSPxBUF should be read before the next byte of data to transfer is written to the SSPxBUF. The Buffer Full bit, BF of the SSPxSTAT register, indicates when SSPxBUF has been loaded with the received data (transmission is complete). When the SSPxBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSPx interrupt is used to determine when the transmission/reception has completed. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur.

FIGURE 15-5: SPI MASTER/SLAVE CONNECTION

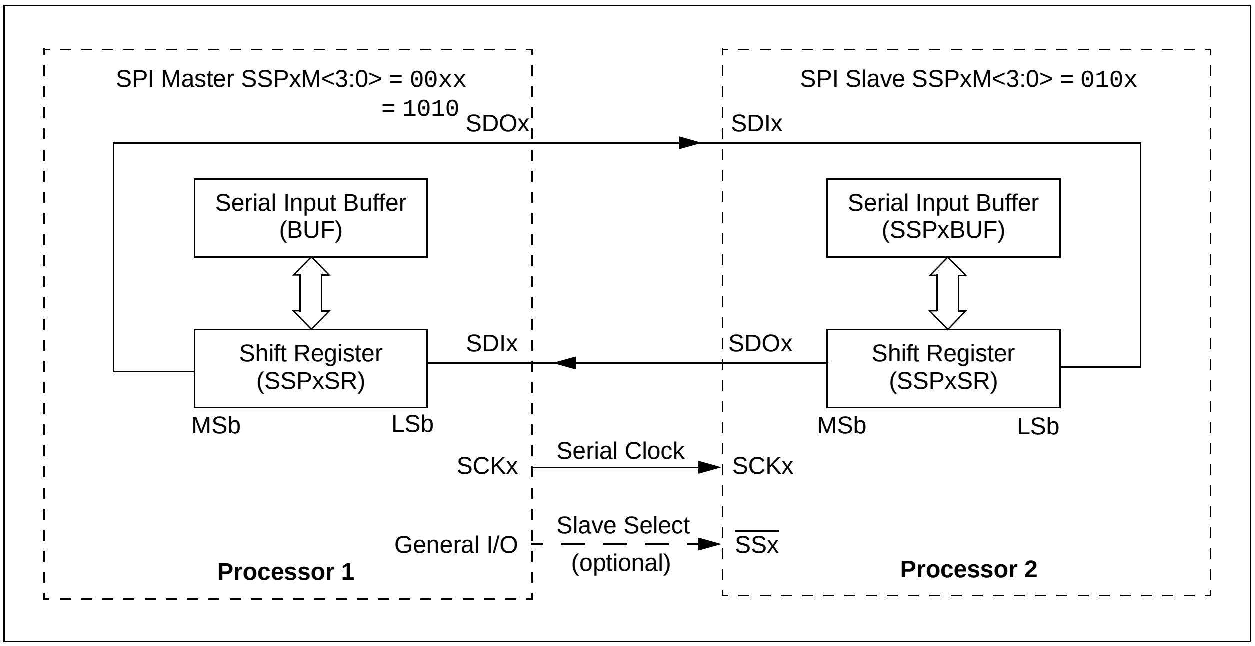


FIGURE 15-22: I²C SLAVE, 10-BIT ADDRESS, TRANSMISSION (SEN = 0, AHEN = 0, DHEN = 0)

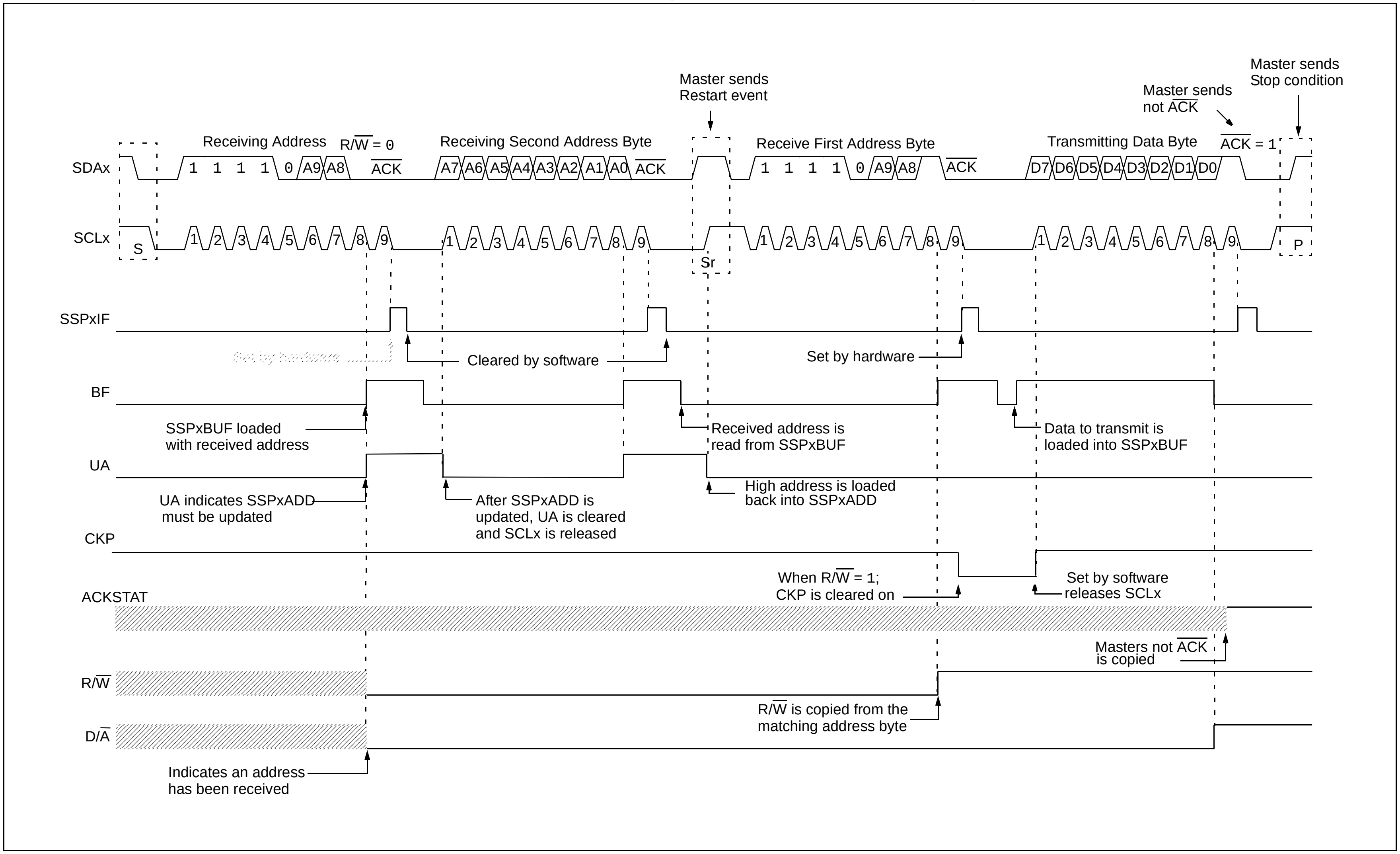


FIGURE 15-34: BUS COLLISION DURING START CONDITION (SCLx = 0)

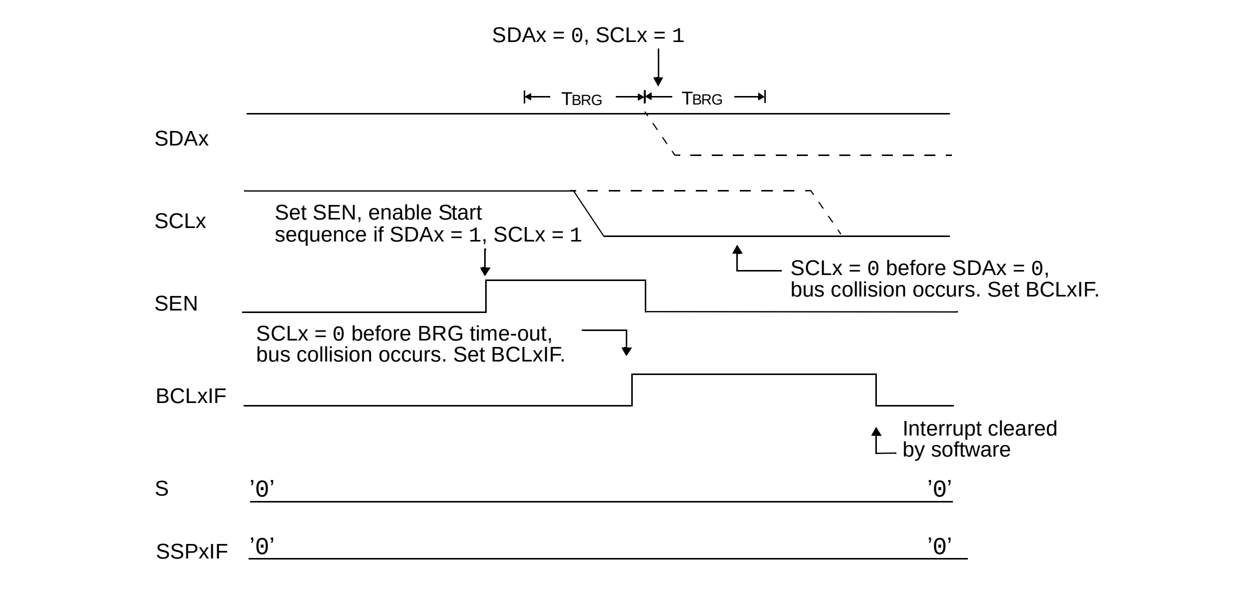
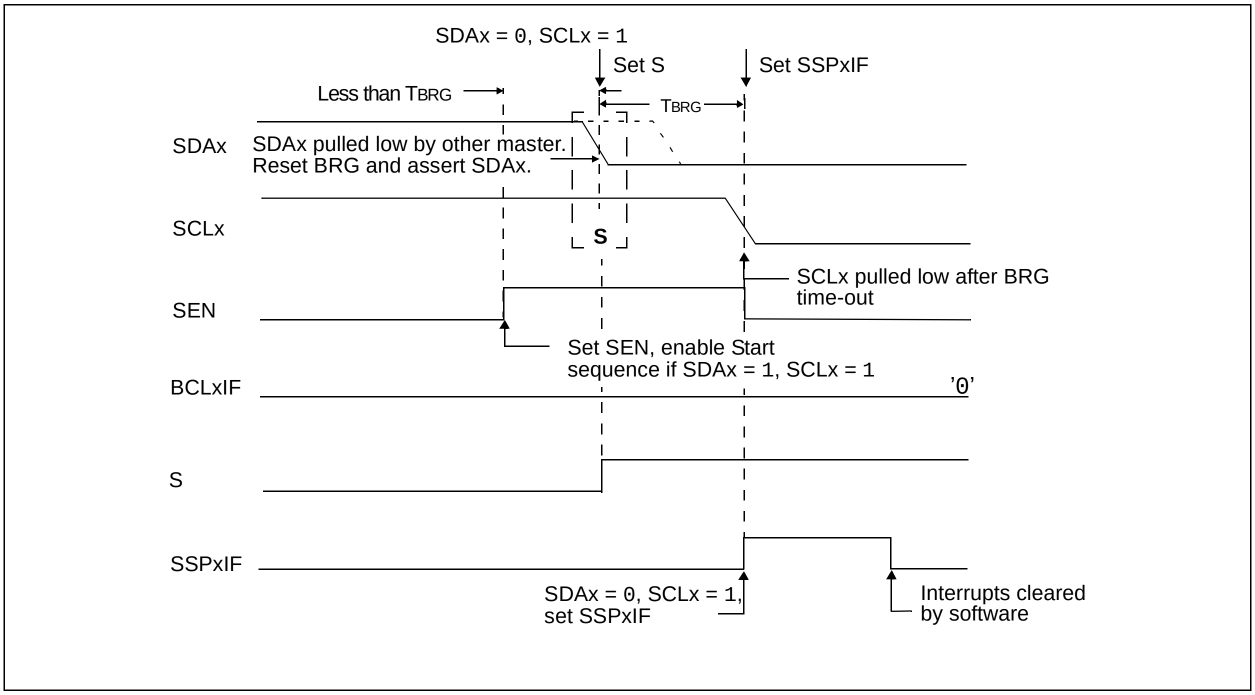


FIGURE 15-35: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION



16.4.1 AUTO-BAUD DETECT

The EUSART module supports automatic detection and calibration of the baud rate.

In the Auto-Baud Detect (ABD) mode, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RXx signal, the RXx signal is timing the BRG. The Baud Rate Generator is used to time the period of a received 55h (ASCII "U") which is the Sync character for the LIN bus. The unique feature of this character is that it has five rising edges including the Stop bit edge.

Setting the ABDEN bit of the BAUDCONx register starts the auto-baud calibration sequence (**Section 16.4.2 "Auto-baud Overflow"**). While the ABD sequence takes place, the EUSART state machine is held in Idle. On the first rising edge of the receive line, after the Start bit, the SPBRGx begins counting up using the BRG counter clock as shown in Table 16-6. The fifth rising edge will occur on the RXx/DTx pin at the end of the eighth bit period. At that time, an accumulated value totaling the proper BRG period is left in the SPBRGHx:SPBRGx register pair, the ABDEN bit is automatically cleared, and the RCxIF interrupt flag is set. A read operation on the RCREGx needs to be performed to clear the RCxIF interrupt. RCREGx content should be discarded. When calibrating for modes that do not use the SPBRGHx register the user can verify that the SPBRGx register did not overflow by checking for 00h in the SPBRGHx register.

The BRG auto-baud clock is determined by the BRG16 and BRGH bits as shown in Table 16-6. During ABD, both the SPBRGHx and SPBRGx registers are used as a 16-bit counter, independent of the BRG16 bit setting. While calibrating the baud rate period, the SPBRGHx and SPBRGx registers are clocked at 1/8th the BRG base clock rate. The resulting byte measurement is the average bit time when clocked at full speed.

Note 1: If the WUE bit is set with the ABDEN bit, auto-baud detection will occur on the byte following the Break character (see **Section 16.4.3 "Auto-Wake-up on Break"**).

2: It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and EUSART baud rates are not possible.

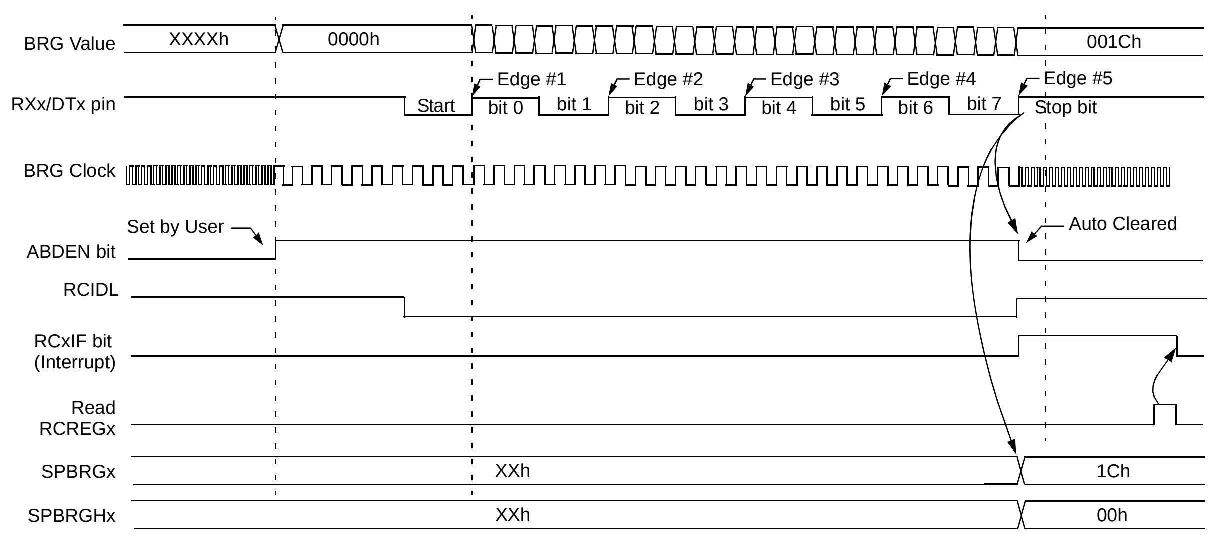
3: During the auto-baud process, the auto-baud counter starts counting at one. Upon completion of the auto-baud sequence, to achieve maximum accuracy, subtract one from the SPBRGHx:SPBRGx register pair.

TABLE 16-6: BRG COUNTER CLOCK RATES

BRG16	BRGH	BRG Base Clock	BRG ABD Clock
0	0	FOSC/64	FOSC/512
0	1	FOSC/16	FOSC/128
1	0	FOSC/16	FOSC/128
1	1	FOSC/4	FOSC/32

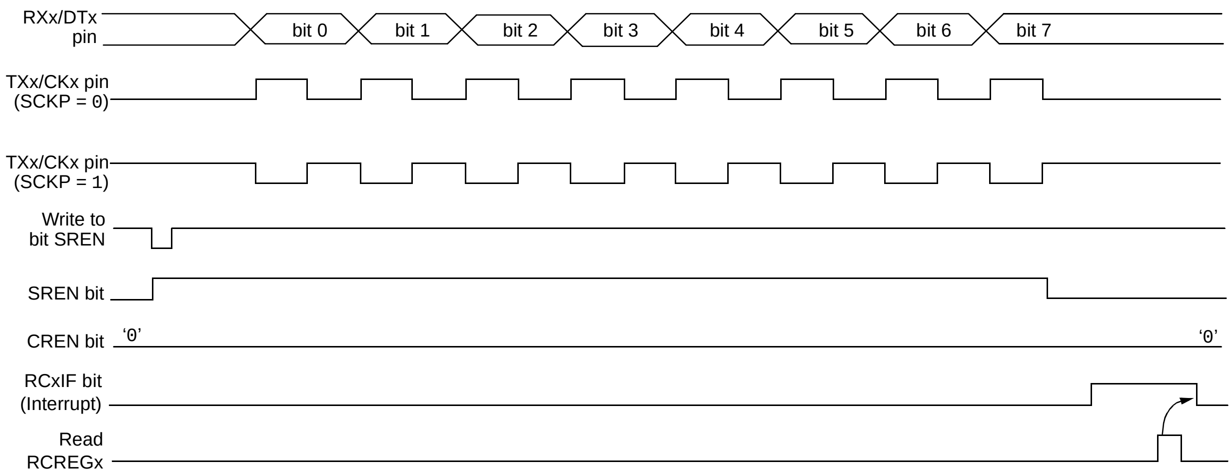
Note: During the ABD sequence, SPBRGx and SPBRGHx registers are both used as a 16-bit counter, independent of BRG16 setting.

FIGURE 16-6: AUTOMATIC BAUD RATE CALIBRATION



Note 1: The ABD sequence requires the **EUSART** module to be configured in Asynchronous mode.

FIGURE 16-12: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)



Note: Timing diagram demonstrates Sync Master mode with bit SREN = 1 and bit BRGH = 0.

TABLE 16-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
BAUDCON1	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	271
BAUDCON2	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	271
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	109
IPR1	—	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP	121
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	CTMUIP	TMR5GIP	TMR3GIP	TMR1GIP	123
PIE1	—	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	117
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	CTMUIE	TMR5GIE	TMR3GIE	TMR1GIE	119
PIR1	—	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	112
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	CTMUIF	TMR5GIF	TMR3GIF	TMR1GIF	114
PMD0	UART2MD	UART1MD	TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	52
RCREG1	EUSART1 Receive Register								—
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	270
RCREG2	EUSART2 Receive Register								—
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	270
SPBRG1	EUSART1 Baud Rate Generator, Low Byte								—
SPBRGH1	EUSART1 Baud Rate Generator, High Byte								—
SPBRG2	EUSART2 Baud Rate Generator, Low Byte								—
SPBRGH2	EUSART2 Baud Rate Generator, High Byte								—
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	269
TXSTA2	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	269

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for synchronous master reception.

18.0 COMPARATOR MODULE

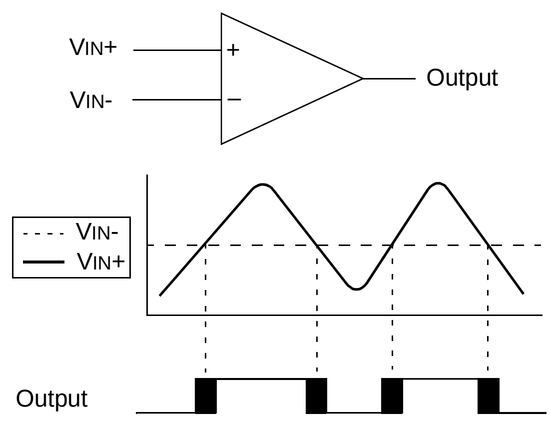
Comparators are used to interface analog circuits to a digital circuit by comparing two analog voltages and providing a digital indication of their relative magnitudes. The comparators are very useful mixed signal building blocks because they provide analog functionality independent of the program execution. The analog comparator module includes the following features:

- Independent comparator control
- Programmable input selection
- Comparator output is available internally/externally
- Programmable output polarity
- Interrupt-on-change
- Wake-up from Sleep
- Programmable Speed/Power optimization
- PWM shutdown
- Programmable and fixed voltage reference
- Selectable Hysteresis

18.1 Comparator Overview

A single comparator is shown in Figure 18-1 along with the relationship between the analog input levels and the digital output. When the analog voltage at VIN+ is less than the analog voltage at VIN-, the output of the comparator is a digital low level. When the analog voltage at VIN+ is greater than the analog voltage at VIN-, the output of the comparator is a digital high level.

FIGURE 18-1: SINGLE COMPARATOR



Note: The black areas of the output of the comparator represents the uncertainty due to input offsets and response time.

BNC Branch if Not Carry

Syntax:	BNC n
Operands:	-128 ≤ n ≤ 127
Operation:	if CARRY bit is '0' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 0011 nnnn nnnn
Description:	If the CARRY bit is '0', 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 BNC Jump`

Before Instruction
- PC = address (HERE)

After Instruction
- If CARRY = 0;
- PC = address (Jump)
- If CARRY = 1;
- PC = address (HERE + 2)

BNN Branch if Not Negative

Syntax:	BNN n
Operands:	-128 ≤ n ≤ 127
Operation:	if NEGATIVE bit is '0' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 0111 nnnn nnnn
Description:	If the NEGATIVE bit is '0', 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 BNN Jump`

Before Instruction
- PC = address (HERE)

After Instruction
- If NEGATIVE = 0;
- PC = address (Jump)
- If NEGATIVE = 1;
- PC = address (HERE + 2)

GOTO	Unconditional Branch			
Syntax:	GOTO k			
Operands:	$0 \le k \le 1048575$			
Operation:	$k \rightarrow PC<20:1>$			
Status Affected:	None			
Encoding: 1st word (k<7:0>) 2nd word(k<19:8>)	1110 1111	1111 k_{19}kkk	$k_7$kkk kkkk	kkkk$_0$ kkkk$_8$

Description: GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value ‘k’ is loaded into PC<20:1>. GOTO is always a 2-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’<7:0>,	No operation	Read literal ‘k’<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: `GOTO THERE`

After Instruction
PC = Address (THERE)

INCF	Increment f			
Syntax:	INCF f {,d {,a}}			
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$			
Operation:	$(f) + 1 \rightarrow dest$			
Status Affected:	C, DC, N, OV, Z			
Encoding:	0010	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 ‘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 25.2.3 “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: `INCF CNT, 1, 0`

Before Instruction
CNT = FFh
Z = 0
C = ?
DC = ?
After Instruction
CNT = 00h
Z = 1
C = 1
DC = 1

MOVFF **Move f to f**

Syntax:	MOVFF f_s,f_d
Operands:	$0 \le f_s \le 4095$ $0 \le f_d \le 4095$
Operation:	$(f_s) \rightarrow f_d$
Status Affected:	None

Encoding:

1st word (source)	`1100`	`ffff`	`ffff`	`ffff`$_s$
2nd word (destin.)	`1111`	`ffff`	`ffff`	`ffff`$_d$

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

Words:	2
Cycles:	2 (3)

Q Cycle Activity:

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

Example: `MOVFF REG1, REG2`

Before Instruction
REG1 = 33h
REG2 = 11h
After Instruction
REG1 = 33h
REG2 = 33h

MOVLB **Move literal to low nibble in BSR**

Syntax:	MOVLW k
Operands:	$0 \le k \le 255$
Operation:	$k \rightarrow$ BSR
Status Affected:	None

Encoding:

`0000`	`0001`	`kkkk`	`kkkk`

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

Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `MOVLB 5`

Before Instruction
BSR Register = 02h
After Instruction
BSR Register = 05h

26.11 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.

26.12 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®

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

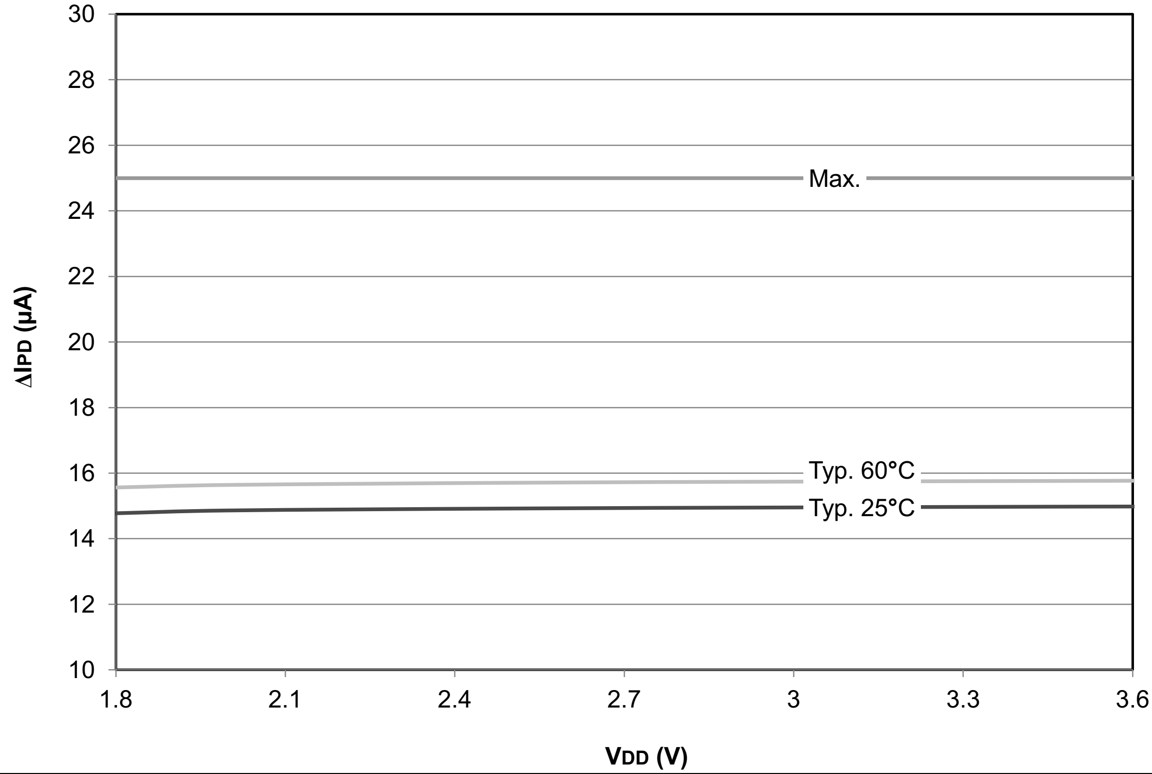
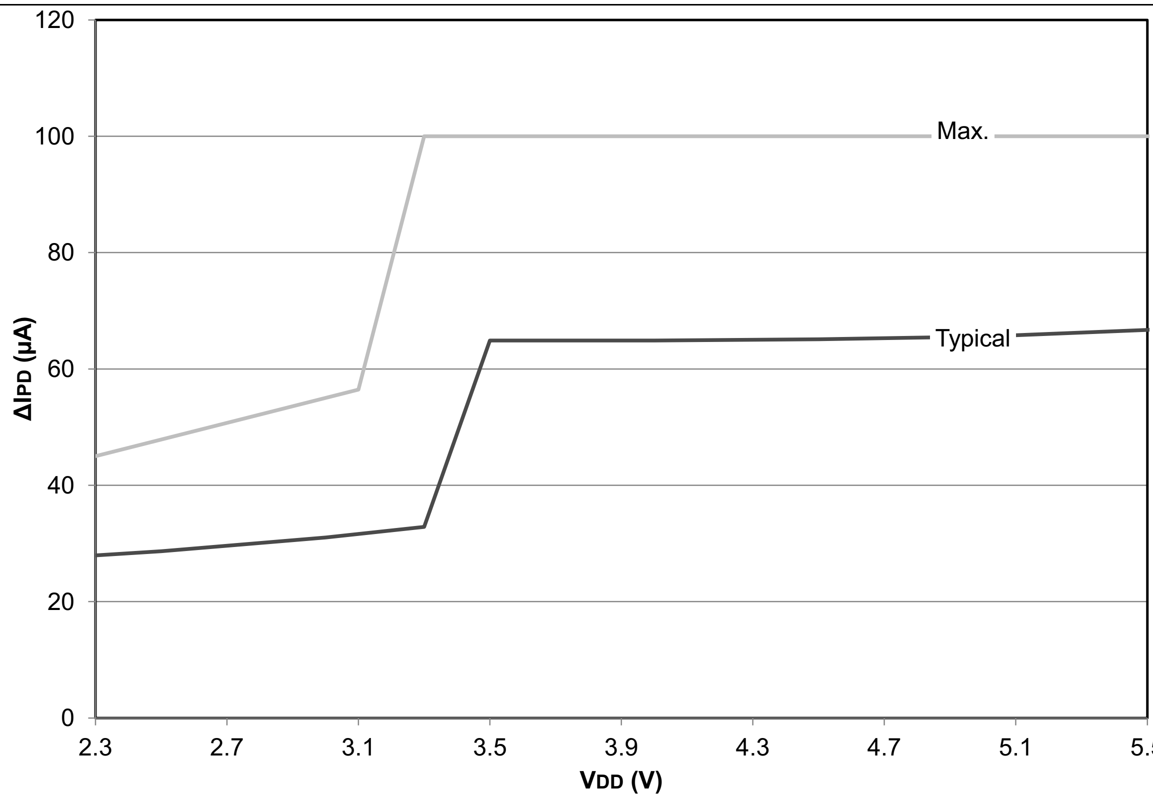


FIGURE 28-18: PIC18F2X/4XK22 DELTA IPD FVR



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

FIGURE 28-96: PIC18(L)F2X/4XK22 TYPICAL FIXED VOLTAGE REFERENCE 2x OUTPUT

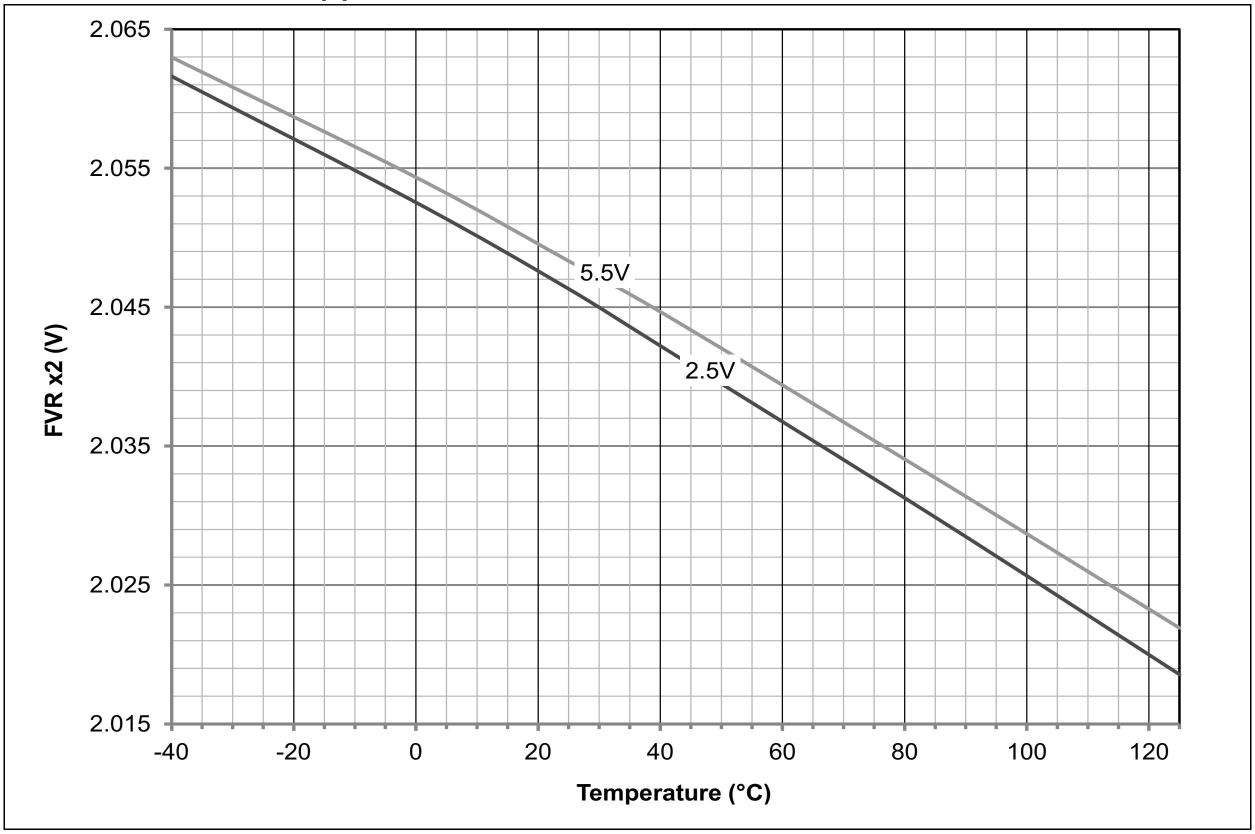


FIGURE 28-97: PIC18(L)F2X/4XK22 TYPICAL FIXED VOLTAGE REFERENCE 2x OUTPUT