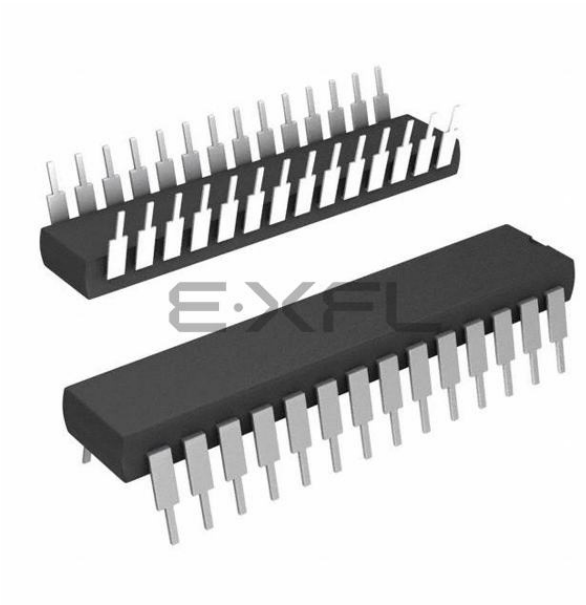


Welcome to E-XFL.COM

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	25
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.6K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 24x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-SPDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf26k40-e-sp

Register 3-7: Configuration Word 4L (30 0006h): Memory Write Protection

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
WRT7	WRT6	WRT5	WRT4	WRT3	WRT2	WRT1	WRT0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘1’	
-n = Value for blank device	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7-0 **WRT<7:0>:** User NVM Self-Write Protection bits[(1)]
1 = Corresponding Memory Block NOT write-protected
0 = Corresponding Memory Block write-protected

Note 1: Refer to Table 10-2 for details on implementation of the individual WRT bits.

Register 3-8: Configuration Word 4H (30 0007h): Memory Write Protection

U-1	U-1	R/W-1	R/W-1	U-1	R/W-1	R/W-1	R/W-1
—	—	LVP	SCANE	—	WRTD	WRTB	WRTC
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘1’	
-n = Value for blank device	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7-6 **Unimplemented:** Read as ‘1’

bit 5 **LVP:** Low-Voltage Programming Enable bit
1 = Low-voltage programming enabled. $\overline{MCLR}$/VPP pin function is $\overline{MCLR}$. MCLRE Configuration bit is ignored.
The LVP bit cannot be written (to zero) while operating from the LVP programming interface. The purpose of this rule is to prevent the user from dropping out of LVP mode while programming from LVP mode, or accidentally eliminating LVP mode from the Configuration state.
0 = HV on $\overline{MCLR}$/VPP must be used for programming

bit 4 **SCANE:** Scanner Enable bit
1 = Scanner module is available for use, SCANMD bit enables the module
0 = Scanner module is NOT available for use, SCANMD bit is ignored

bit 3 **Unimplemented:** Read as ‘1’

bit 2 **WRTD:** Data EEPROM Write Protection bit
1 = Data EEPROM NOT write-protected
0 = Data EEPROM write-protected

bit 1 **WRTB:** Boot Block Write Protection bit
1 = Boot Block NOT write-protected
0 = Boot Block write-protected

bit 0 **WRTC:** Configuration Register Write Protection bit
1 = Configuration Register NOT write-protected
0 = Configuration Register write-protected

6.1.2 INTERRUPTS DURING DOZE

If an interrupt occurs and the Recover-On-Interrupt bit is clear (ROI = 0) at the time of the interrupt, the Interrupt Service Routine (ISR) continues to execute at the rate selected by DOZE<2:0>. Interrupt latency is extended by the DOZE<2:0> ratio.

If an interrupt occurs and the ROI bit is set (ROI = 1) at the time of the interrupt, the DOZEN bit is cleared and the CPU executes at full speed. The prefetched instruction is executed and then the interrupt vector sequence is executed. In Figure 6-1, the interrupt occurs during the 2nd instruction cycle of the Doze period, and immediately brings the CPU out of Doze. If the Doze-On-Exit (DOE) bit is set (DOE = 1) when the RETFIE operation is executed, DOZEN is set, and the CPU executes at the reduced rate based on the DOZE<2:0> ratio.

EXAMPLE 6-1: DOZE SOFTWARE EXAMPLE

```
//Mainline operation
bool somethingToDo = FALSE:
void main()
{
    initializeSystem();
            // DOZE = 64:1 (for example)
            // ROI = 1;
    GIE = 1; // enable interrupts
    while (1)
    {
        // If ADC completed, process data
        if (somethingToDo)
        {
            doSomething();
            DOZEN = 1; // resume low-power
        }
    }
}

// Data interrupt handler
void interrupt()
{
    // DOZEN = 0 because ROI = 1
    if (ADIF)
    {
        somethingToDo = TRUE;
        DOE = 0; // make main() go fast
        ADIF = 0;
    }
    // else check other interrupts...
    if (TMR0IF)
    {
        timerTick++;
        DOE = 1; // make main() go slow
        TMR0IF = 0;
    }
}
```

6.2 Sleep Mode

Sleep mode is entered by executing the SLEEP instruction, while the Idle Enable (IDLEN) bit of the CPUDOZE register is clear (IDLEN = 0).

Upon entering Sleep mode, the following conditions exist:

1. WDT will be cleared but keeps running if enabled for operation during Sleep
2. The $\overline{\text{PD}}$ bit of the STATUS register is cleared (Register 10-2)
3. The $\overline{\text{TO}}$ bit of the STATUS register is set (Register 10-2)
4. The CPU clock is disabled
5. LFINTOSC, SOSC, HFINTOSC and ADCRC are unaffected and peripherals using them may continue operation in Sleep.
6. I/O ports maintain the status they had before Sleep was executed (driving high, low, or high-impedance)
7. Resets other than WDT are not affected by Sleep mode

Refer to individual chapters for more details on peripheral operation during Sleep.

To minimize current consumption, the following conditions should be considered:

- I/O pins should not be floating
- External circuitry sinking current from I/O pins
- Internal circuitry sourcing current from I/O pins
- Current draw from pins with internal weak pull-ups
- Modules using any oscillator

I/O pins that are high-impedance inputs should be pulled to V_{DD} or V_{SS} externally to avoid switching currents caused by floating inputs.

Examples of internal circuitry that might be sourcing current include modules such as the DAC and FVR modules. See **Section 30.0 “5-Bit Digital-to-Analog Converter (DAC) Module”** and **Section 28.0 “Fixed Voltage Reference (FVR)”** for more information on these modules.

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F26/45/46K40 DEVICES (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
ED8h	OSCCON1	—	NOSC<2:0>			NDIV<3:0>				-qqqqqqq
ED7h	CPUDOZE	IDLEN	DOZEN	ROI	DOE	—	DOZE<2:0>			0000-000
ED6h	WDTTMR	WDTTMR<4:0>					STATE	PSCNT<17:16>		xxxxx000
ED5h	WDTPSH	PSCNT<7:0>								00000000
ED4h	WDTPSL	PSCNT<15:8>								00000000
ED3h	WDTCON1	—	WDTCS<2:0>			—	WINDOW<2:0>			-qqq-qqq
ED2h	WDTCON0	—	—	WDTPS<4:0>					SEN	--qqqqq0
ED1h	PIR7	SCANIF	CRCIF	NVMIF	—	—	—	—	CWG1IF	000----0
ED0h	PIR6	—	—	—	—	—	—	CCP2IF	CCP1IF	------00
ECFh	PIR5	—	—	—	—	—	TMR5GIF	TMR3GIF	TMR1GIF	-----000
ECEh	PIR4	—	—	TMR6IF	TMR5IF	TMR4IF	TMR3IF	TMR2IF	TMR1IF	--000000
ECDh	PIR3	RC2IF	TX2IF	RC1IF	TX1IF	BCL2IF	SSP2IF	BCL1IF	SSP1IF	00000000
ECCh	PIR2	HLVDIF	ZCDIF	—	—	—	—	C2IF	C1IF	00----00
ECBh	PIR1	OSCFIF	CSWIF	—	—	—	—	ADTIF	ADIF	00----00
ECAh	PIR0	—	—	TMR0IF	IOCIF	—	INT2IF	INT1IF	INT0IF	--00-000
EC9h	PIE7	SCANIE	CRCIE	NVMIE	—	—	—	—	CWG1IE	000----0
EC8h	PIE6	—	—	—	—	—	—	CCP2IE	CCP1IE	------00
EC7h	PIE5	—	—	—	—	—	TMR5GIE	TMR3GIE	TMR1GIE	-----000
EC6h	PIE4	—	—	TMR6IE	TMR5IE	TMR4IE	TMR3IE	TMR2IE	TMR1IE	--000000
EC5h	PIE3	RC2IE	TX2IE	RC1IE	TX1IE	BCL2IE	SSP2IE	BCL1IE	SSP1IE	00000000
EC4h	PIE2	HLVDIE	ZCDIE	—	—	—	—	C2IE	C1IE	00----00
EC3h	PIE1	OSCFIE	CSWIE	—	—	—	—	ADTIE	ADIE	00----00
EC2h	PIE0	—	—	TMR0IE	IOCIE	—	INT2IE	INT1IE	INT0IE	--00-000
EC1h	IPR7	SCANIP	CRCIP	NVMIP	—	—	—	—	CWG1IP	111----1
EC0h	IPR6	—	—	—	—	—	—	CCP2IP	CCP1IP	------11
EBFh	IPR5	—	—	—	—	—	TMR5GIP	TMR3GIP	TMR1GIP	-----111
EBEh	IPR4	—	—	TMR6IP	TMR5IP	TMR4IP	TMR3IP	TMR2IP	TMR1IP	--111111
EBDh	IPR3	RC2IP	TX2IP	RC1IP	TX1IP	BCL2IP	SSP2IP	BCL1IP	SSP1IP	11111111
EBCh	IPR2	HLVDIP	ZCDIP	—	—	—	—	C2IP	C1IP	11----11
EBBh	IPR1	OSCFIP	CSWIP	—	—	—	—	ADTIP	ADIP	11----11
EBAh	IPR0	—	—	TMR0IP	IOCIP	—	INT2IP	INT1IP	INT0IP	--11-111
EB9h	SSP1SSPPS	—	—	—	SSPSSPPS<4:0>					---00101
EB8h	SSP1DATPPS	—	—	—	SSPDATPPS<4:0>					---10100
EB7h	SSP1CLKPPS	—	—	—	SSPCLKPPS<4:0>					---10011
EB6h	TX1PPS	—	—	—	TXPPS<4:0>					---10110
EB5h	RX1PPS	—	—	—	RXPPS<4:0>					---10111
EB4h	MDSRCPPS	—	—	—	MDSRCPPS<4:0>					---00101
EB3h	MDCARHPPS	—	—	—	MDCARHPPS<4:0>					---00100
EB2h	MDCARLPPS	—	—	—	MDCARLPPS<4:0>					---00011

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition
Note 1: Not available on LF devices.
2: Not available on PIC18(L)F26K40 (28-pin variants).
3: Not available on PIC18(L)F45K40 devices.

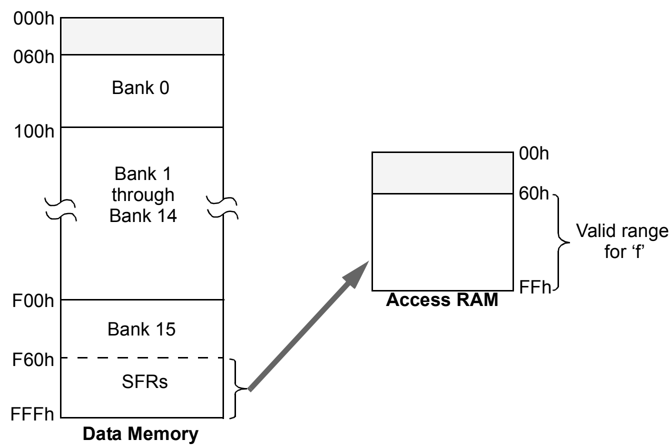
FIGURE 10-7: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)

EXAMPLE INSTRUCTION: `ADDWF, f, d, a` (Opcode: `0010 01da ffff ffff`)

When 'a' = `0` and f ≥ 60h:

The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and 0FFh. This is the same as locations F60h to FFFh (Bank 15) of data memory.

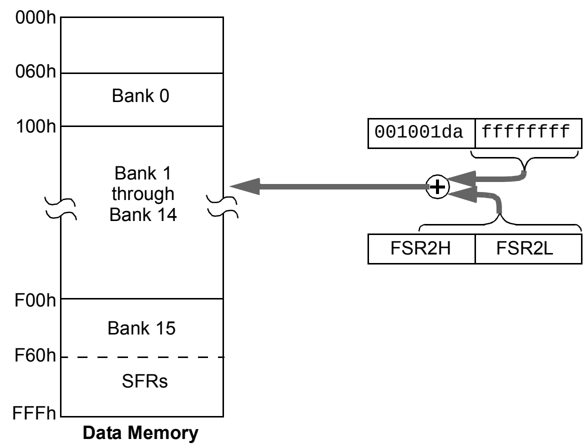
Locations below 60h are not available in this addressing mode.



When 'a' = `0` and f ≤ 5Fh:

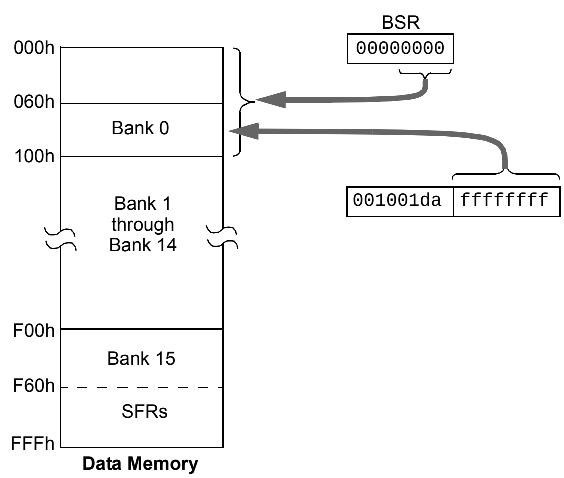
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

Note that in this mode, the correct syntax is now:
`ADDWF [k], d`
where 'k' is the same as 'f'.



When 'a' = `1` (all values of f):

The instruction executes in Direct mode (also known as Direct Long mode). 'f' is interpreted as a location in one of the 16 banks of the data memory space. The bank is designated by the Bank Select Register (BSR). The address can be in any implemented bank in the data memory space.



11.1.5 ERASING PROGRAM FLASH MEMORY

The minimum erase block is 32 or 64 words (refer to Table 11-3). Only through the use of an external programmer, or through ICSP™ control, can larger blocks of program memory be bulk erased. Word erase in the Flash array is not supported.

For example, when initiating an erase sequence from a microcontroller with erase row size of 32 words, a block of 32 words (64 bytes) of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:6> point to the block being erased. The TBLPTR<5:0> bits are ignored.

The NVMCON1 register commands the erase operation. The NVMREG<1:0> bits must be set to point to the Program Flash Memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation.

The NVM unlock sequence described in **Section 11.1.4 "NVM Unlock Sequence"** should be used to guard against accidental writes. This is sometimes referred to as a long write.

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

11.1.5.1 Program Flash Memory Erase Sequence

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

1. NVMREG bits of the NVMCON1 register point to PFM
2. Set the FREE and WREN bits of the NVMCON1 register
3. Perform the unlock sequence as described in **Section 11.1.4 "NVM Unlock Sequence"**

If the PFM address is write-protected, the WR bit will be cleared and the erase operation will not take place, WRERR is signaled in this scenario.

The operation erases the memory row indicated by masking the LSBs of the current TBLPTR.

While erasing PFM, CPU operation is suspended and it resumes when the operation is complete. Upon completion the WR bit is cleared in hardware, the NVMIF is set and an interrupt will occur if the NVMIE bit is also set.

Write latch data is not affected by erase operations and WREN will remain unchanged.

Note 1:	If a write or erase operation is terminated by an unexpected event, WRERR bit will be set which the user can check to decide whether a rewrite of the location(s) is needed.
2:	WRERR is set if WR is written to '1' while TBLPTR points to a write-protected address.
3:	WRERR is set if WR is written to '1' while TBLPTR points to an invalid address location (Table 10-2 and Table 11-1).

REGISTER 14-11: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
OSCFIE	CSWIE	—	—	—	—	ADTIE	ADIE
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 **OSCFIE:** Oscillator Fail Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **CSWIE:** Clock-Switch Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5-2 **Unimplemented:** Read as ‘0’

bit 1 **ADTIE:** ADC Threshold Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **ADIE:** ADC Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 15-8: INLVLx: INPUT LEVEL CONTROL REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
INLVLx7	INLVLx6	INLVLx5	INLVLx4	INLVLx3	INLVLx2	INLVLx1	INLVLx0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown
-n/n = Value at POR and BOR/Value at all other Resets		

bit 7-0 **INLVLx<7:0>**: Input Level Select on Pins Rx<7:0>, respectively
`1` = ST input used for port reads and interrupt-on-change
`0` = TTL input used for port reads and interrupt-on-change

TABLE 15-9: INPUT LEVEL PORT REGISTERS

Name	Device 28 Pins	Device 40/44 Pins	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INLVLA	X	X	INLVLA7	INLVLA6	INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0
INLVLB	X	X	INLVLB7	INLVLB6	INLVLB5	INLVLB4	INLVLB3	INLVLB2(1)	INLVLB1(1)	INLVLB0
INLVLC	X	X	INLVLC7	INLVLC6	INLVLC5	INLVLC4(1)	INLVLC3(1)	INLVLC2	INLVLC1	INLVLC0
INLVLD	X		—	—	—	—	—	—	—	—
		X	INLVLD7	INLVLD6	INLVLD5	INLVLD4	INLVLD3	INLVLD2	INLVLD1(1)	INLVLD0(1)
INLVLE	X		—	—	—	—	INLVLE3	—	—	—
		X	—	—	—	—	INLVLE3	INLVLE2	INLVLE1	INLVLE0

Note 1: Pins read the I^2C ST inputs when MSSP inputs select these pins, and I^2C mode is enabled.

REGISTER 19-4: TMRxGATE: TIMERx GATE ISM REGISTER

U-0	U-0	U-0	U-0	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u
—	—	—	—	GSS<3:0>			
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	u = unchanged

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **GSS<3:0>:** Timerx Gate Source Selection bits

GSS	Timer1	Timer3	Timer5
	Gate Source	Gate Source	Gate Source
1111	Reserved	Reserved	Reserved
1110	ZCDOUT	ZCDOUT	ZCDOUT
1101	CMP2OUT	CMP2OUT	CMP2OUT
1100	CMP1OUT	CMP1OUT	CMP1OUT
1011	PWM4OUT	PWM4OUT	PWM4OUT
1010	PWM3OUT	PWM3OUT	PWM3OUT
1001	CCP2OUT	CCP2OUT	CCP2OUT
1000	CCP1OUT	CCP1OUT	CCP1OUT
0111	TMR6OUT (post-scaled)	TMR6OUT (post-scaled)	TMR6OUT (post-scaled)
0110	TMR5 overflow	TMR5 overflow	Reserved
0101	TMR4OUT (post-scaled)	TMR4OUT (post-scaled)	TMR4OUT (post-scaled)
0100	TMR3 overflow	Reserved	TMR3 overflow
0011	TMR2OUT (post-scaled)	TMR2OUT (post-scaled)	TMR2OUT (post-scaled)
0010	Reserved	TMR1 overflow	TMR1 overflow
0001	TMR0 overflow	TMR0 overflow	TMR0 overflow
0000	Pin selected by T1GPPS	Pin selected by T3GPPS	Pin selected by T5GPPS

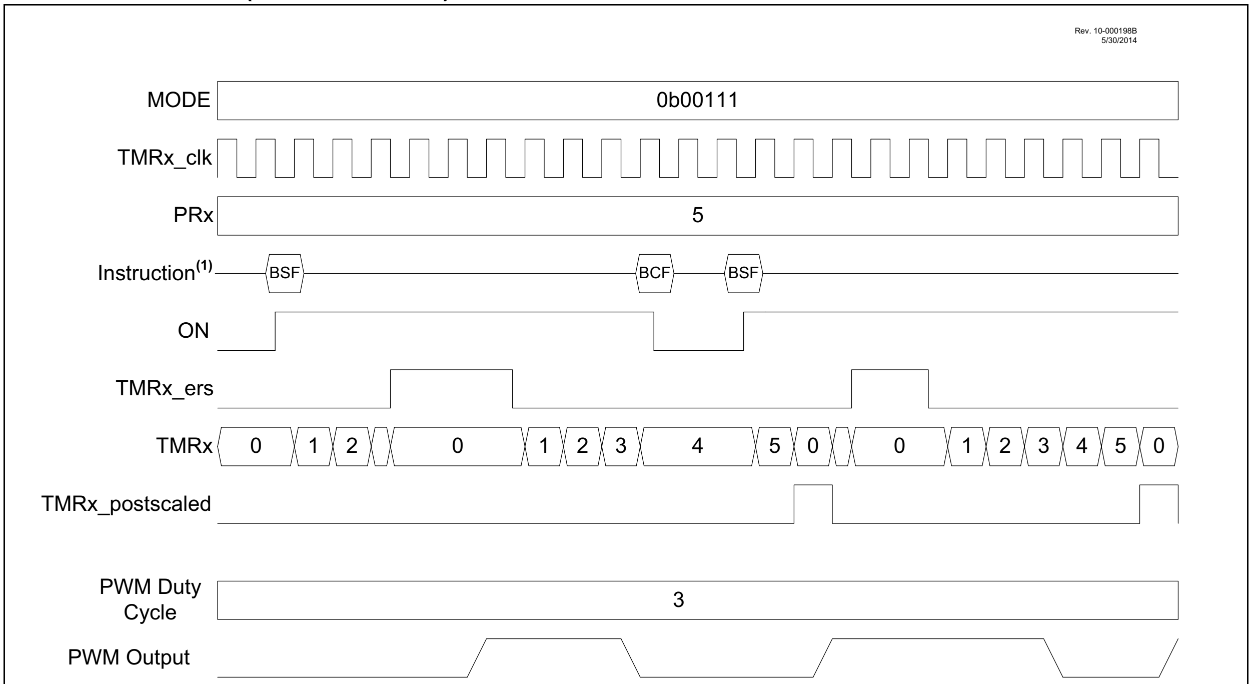
20.5.4 LEVEL-TRIGGERED HARDWARE LIMIT MODE

In the Level-Triggered Hardware Limit Timer modes the counter is reset by high or low levels of the external signal TMRx_ers, as shown in Figure 20-7. Selecting MODE<4:0> = `00110` will cause the timer to reset on a low level external signal. Selecting MODE<4:0> = `00111` will cause the timer to reset on a high level external signal. In the example, the counter is reset while TMRx_ers = `1`. ON is controlled by BSF and BCF instructions. When ON = `0` the external signal is ignored.

When the CCP uses the timer as the PWM time base then the PWM output will be set high when the timer starts counting and then set low only when the timer count matches the CCPRx value. The timer is reset when either the timer count matches the PRx value or two clock periods after the external Reset signal goes true and stays true.

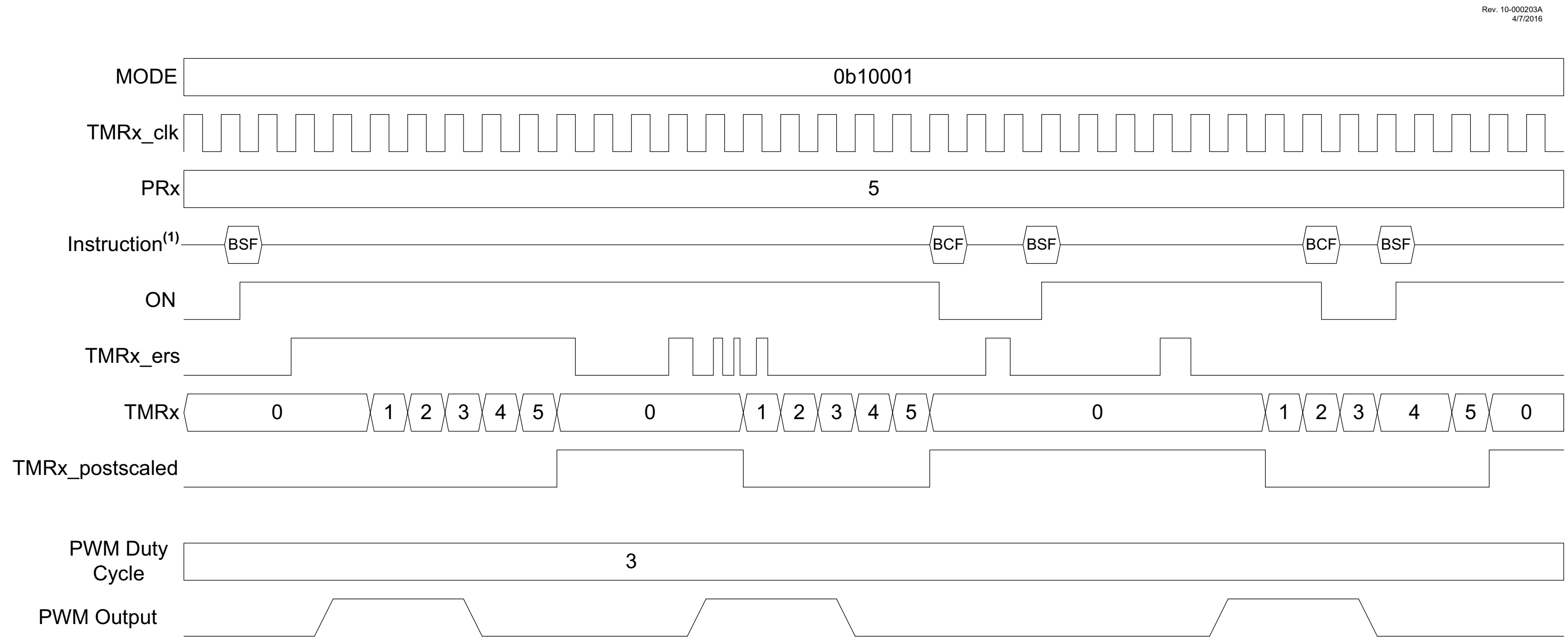
The timer starts counting, and the PWM output is set high, on either the clock following the PRx match or two clocks after the external Reset signal relinquishes the Reset. The PWM output will remain high until the timer counts up to match the CCPRx pulse width value. If the external Reset signal goes true while the PWM output is high then the PWM output will remain high until the Reset signal is released allowing the timer to count up to match the CCPRx value.

FIGURE 20-7: LEVEL-TRIGGERED HARDWARE LIMIT MODE TIMING DIAGRAM (MODE = `00111`)



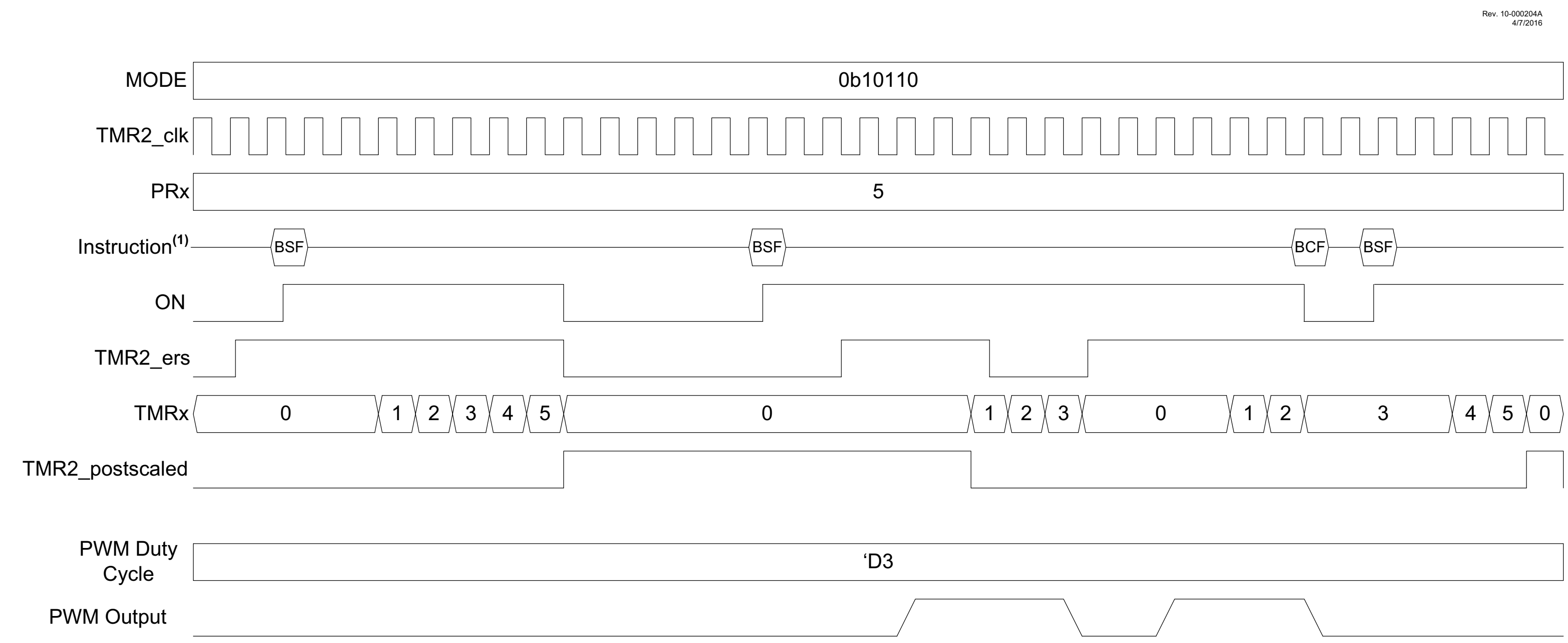
Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

FIGURE 20-12: RISING EDGE-TRIGGERED MONOSTABLE MODE TIMING DIAGRAM (MODE = `10001`)



Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

FIGURE 20-13: LEVEL-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE TIMING DIAGRAM (MODE = `10110`)



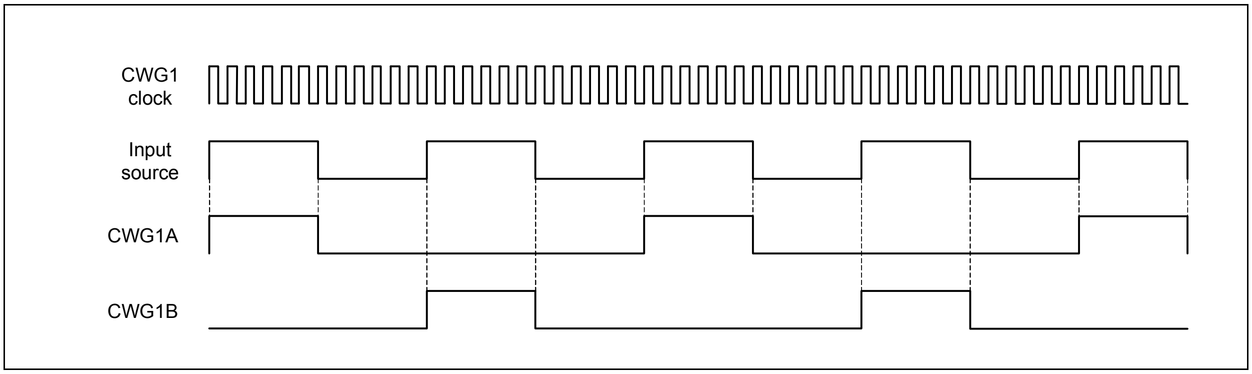
Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

TABLE 21-5: SUMMARY OF REGISTERS ASSOCIATED WITH CCPx

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
INTCON	GIE/GIEH	PEIE/GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG	170
PIE6	—	—	—	—	—	—	CCP2IE	CCP1IE	185
PIR6	—	—	—	—	—	—	CCP2IF	CCP1IF	177
IPR6	—	—	—	—	—	—	CCP2IP	CCP1IP	193
PMD3	—	—	—	—	PWM4MD	PWM3MD	CCP2MD	CCP1MD	71
CCPxCON	EN	—	OUT	FMT	MODE<3:0>				268
CCPxCAP	—	—	—	—	—	—	CTS<1:0>		271
CCPRxL	CCPRx<7:0>								271
CCPRxH	CCPRx<15:8>								272
CCPTMRS	P4TSEL<1:0>		P3TSEL<1:0>		C2TSEL<1:0>		C1TSEL<1:0>		270
CCPxPPS	—	—	—	CCPxPPS<4:0>					216
RxyPPS	—	—	—	RxyPPS<4:0>					218
T1CON	—	—	T1CKPS<1:0>		—	$\overline{\text{T1SYNC}}$	T1RD16	TMR1ON	229
T1GCON	TMR1GE	T1GPOL	T1GTM	T1GSPM	T1GO/$\overline{\text{DONE}}$	T1GVAL	—	—	230
T1CLK	—	—	—	—	CS<3:0>				231
T1GATE	—	—	—	—	GSS<3:0>				232
TMR1L	TMR1L7	TMR1L6	TMR1L5	TMR1L4	TMR1L3	TMR1L2	TMR1L1	TMR1L0	233
TMR1H	TMR1H7	TMR1H6	TMR1H5	TMR1H4	TMR1H3	TMR1H2	TMR1H1	TMR1H0	233
TMR2	TMR2<7:0>								244*
T2PR	PR2<7:0>								244*
T2CON	ON	CKPS<2:0>			OUTPS<3:0>				262
T2HLT	PSYNC	CPOL	CSYNC	MODE<4:0>					263
T2CLKCON	—	—	—	—	CS<3:0>				264
T2RST	—	—	—	—	RSEL<3:0>				265

Legend: — = Unimplemented location, read as '0'. Shaded cells are not used by the CCP module.
* Not a physical register.

FIGURE 24-4: CWG1 PUSH-PULL MODE OPERATION



24.2.3 FULL-BRIDGE MODES

In Forward and Reverse Full-Bridge modes, three outputs drive static values while the fourth is modulated by the input data signal. The mode selection may be toggled between forward and reverse by toggling the MODE<0> bit of the CWG1CON0 while keeping MODE<2:1> static, without disabling the CWG module. When connected as shown in Figure 24-5, the outputs are appropriate for a full-bridge motor driver. Each CWG output signal has independent polarity control, so the circuit can be adapted to high-active and low-active drivers. A simplified block diagram for the Full-Bridge modes is shown in Figure 24-6.

FIGURE 24-5: EXAMPLE OF FULL-BRIDGE APPLICATION

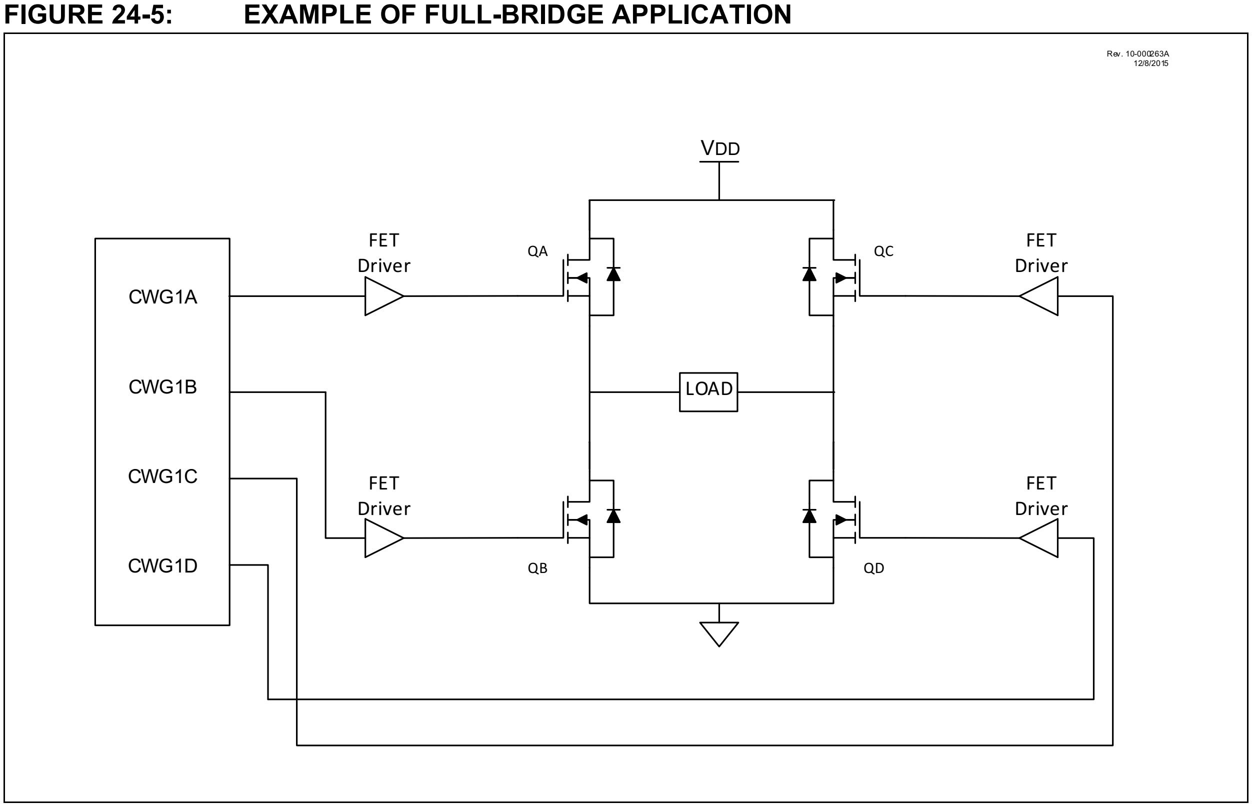


FIGURE 24-6: SIMPLIFIED CWG BLOCK DIAGRAM (FORWARD AND REVERSE FULL-BRIDGE MODES)

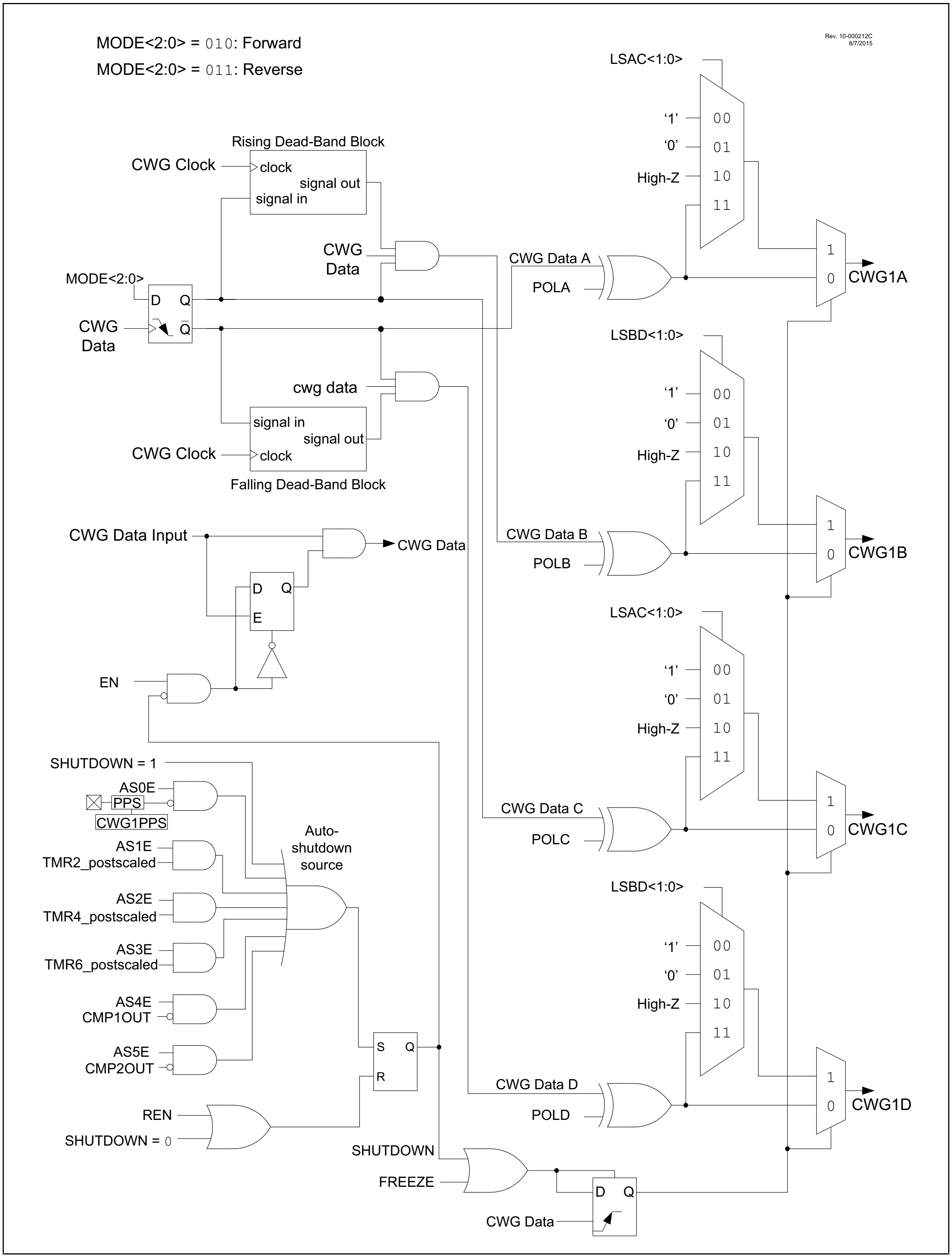
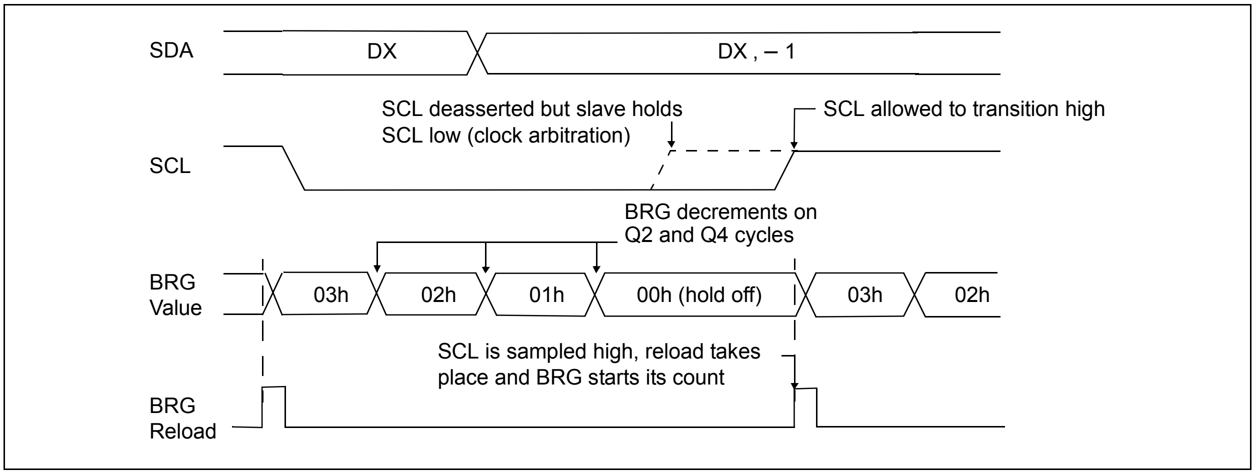


FIGURE 26-25: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION



26.10.3 WCOL STATUS FLAG

If the user writes the SSPxBUF when a Start, Restart, Stop, Receive or Transmit sequence is in progress, the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur). Any time the WCOL bit is set it indicates that an action on SSPxBUF was attempted while the module was not idle.

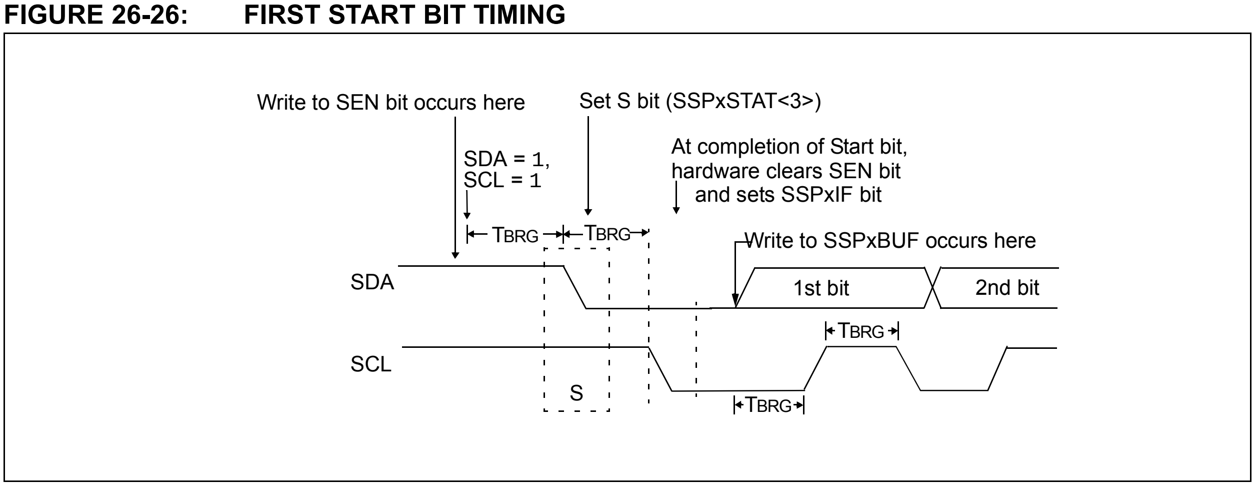
Note:	Because queuing of events is not allowed, writing to the lower five bits of SSPxCON2 is disabled until the Start condition is complete.

26.10.4 I²C MASTER MODE START CONDITION TIMING

To initiate a Start condition (Figure 26-26), the user sets the Start Enable bit, SEN bit of the SSPxCON2 register. If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and starts its count. If SCL and SDA are both sampled high when the Baud Rate Generator times out (TBRG), the SDA pin is driven low. The action of the SDA being driven low while SCL is high is the Start condition and causes the S bit of the SSPxSTAT1 register to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit of the SSPxCON2 register will be automatically cleared by hardware; the Baud Rate Generator is suspended, leaving the SDA line held low and the Start condition is complete.

Note 1:	If at the beginning of the Start condition, the SDA and SCL pins are already sampled low, or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLxIF, is set, the Start condition is aborted and the I²C module is reset into its Idle state.
2:	The Philips I²C specification states that a bus collision cannot occur on a Start.

FIGURE 26-26: FIRST START BIT TIMING



27.2.2.4 Receive Framing Error

Each character in the receive FIFO buffer has a corresponding framing error Status bit. A framing error indicates that a Stop bit was not seen at the expected time. The framing error status is accessed via the FERR bit of the RCxSTA register. The FERR bit represents the status of the top unread character in the receive FIFO. Therefore, the FERR bit must be read before reading the RCxREG.

The FERR bit is read-only and only applies to the top unread character in the receive FIFO. A framing error (FERR = 1) does not preclude reception of additional characters. It is not necessary to clear the FERR bit. Reading the next character from the FIFO buffer will advance the FIFO to the next character and the next corresponding framing error.

The FERR bit can be forced clear by clearing the SPEN bit of the RCxSTA register which resets the EUSART. Clearing the CREN bit of the RCxSTA register does not affect the FERR bit. A framing error by itself does not generate an interrupt.

Note:	If all receive characters in the receive FIFO have framing errors, repeated reads of the RCxREG will not clear the FERR bit.

27.2.2.5 Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before the FIFO is accessed. When this happens the OERR bit of the RCxSTA register is set. The characters already in the FIFO buffer can be read but no additional characters will be received until the error is cleared. The error must be cleared by either clearing the CREN bit of the RCxSTA register or by resetting the EUSART by clearing the SPEN bit of the RCxSTA register.

27.2.2.6 Receiving 9-Bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCxSTA register is set the EUSART will shift nine bits into the RSR for each character received. The RX9D bit of the RCxSTA register is the ninth and Most Significant data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the eight Least Significant bits from the RCxREG.

27.2.2.7 Address Detection

A special Address Detection mode is available for use when multiple receivers share the same transmission line, such as in RS-485 systems. Address detection is enabled by setting the ADDEN bit of the RCxSTA register.

Address detection requires 9-bit character reception. When address detection is enabled, only characters with the ninth data bit set will be transferred to the receive FIFO buffer, thereby setting the RCxIF interrupt bit. All other characters will be ignored.

Upon receiving an address character, user software determines if the address matches its own. Upon address match, user software must disable address detection by clearing the ADDEN bit before the next Stop bit occurs. When user software detects the end of the message, determined by the message protocol used, software places the receiver back into the Address Detection mode by setting the ADDEN bit.

27.5.2.3 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical (**Section 27.5.1.5 "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 RCxREG register. If the RCxIE 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.

27.5.2.4 Synchronous Slave Reception Setup:

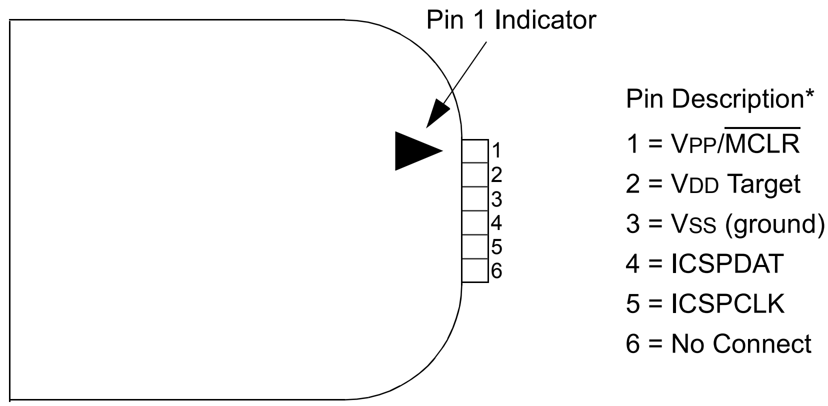
1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the ANSEL bit for both the CKx and DTx pins (if applicable).
3. If interrupts are desired, set the RCxIE bit of the PIE3 register and the GIE and PEIE bits of the INTCON register.
4. If 9-bit reception is desired, set the RX9 bit.
5. Set the CREN bit to enable reception.
6. The RCxIF bit will be set when reception is complete. An interrupt will be generated if the RCxIE bit was set.
7. If 9-bit mode is enabled, retrieve the Most Significant bit from the RX9D bit of the RCxSTA register.
8. Retrieve the eight Least Significant bits from the receive FIFO by reading the RCxREG register.
9. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCxSTA register or by clearing the SPEN bit which resets the EUSART.

TABLE 27-10: 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
BAUDxCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	395
INTCON	GIE/GIEH	PEIE/GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG	170
PIE3	RC2IE	TX2IE	RC1IE	TX1IE	BCL2IE	SSP2IE	BCL1IE	SSP1IE	182
PIR3	RC2IF	TX2IF	RC1IF	TX1IF	BCL2IF	SSP2IF	BCL1IF	SSP1IF	174
IPR3	RC2IP	TX2IP	RC1IP	TX1IP	BCL2IP	SSP2IP	BCL1IP	SSP1IP	190
RCxREG	EUSART Receive Data Register								399*
RCxSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	394
RxyPPS	—	—	—	RxyPPS<4:0>					218
RXxPPS	—	—	—	RXPPS<4:0>					216
TXxSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	393

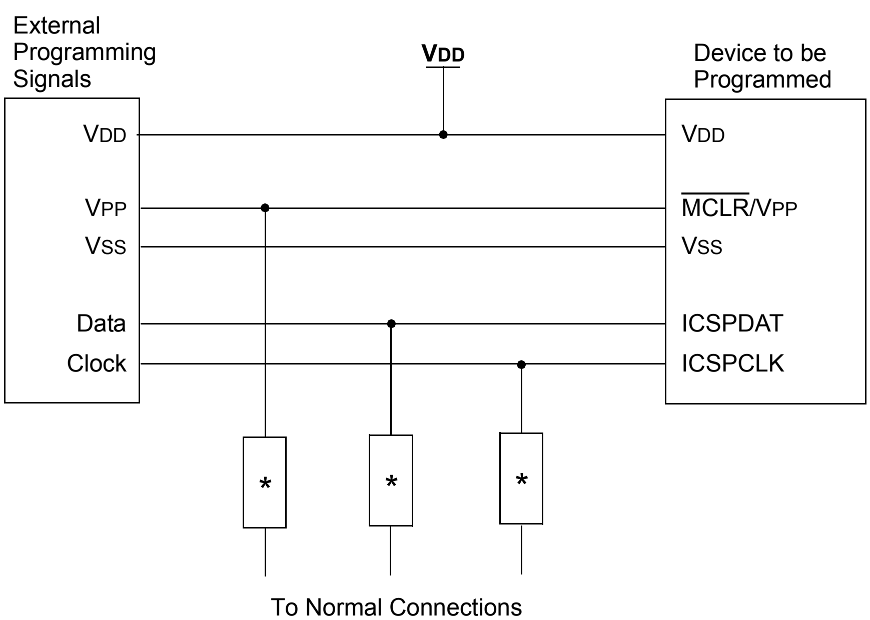
Legend: — = unimplemented location, read as '0'. Shaded cells are not used for synchronous slave reception.
* Page provides register information.

FIGURE 34-2: PICkit™ PROGRAMMER STYLE CONNECTOR INTERFACE



* The 6-pin header (0.100" spacing) accepts 0.025" square pins.

FIGURE 34-3: TYPICAL CONNECTION FOR ICSP™ PROGRAMMING



* Isolation devices (as required).

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

37.4 AC Characteristics

FIGURE 37-4: LOAD CONDITIONS