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	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 35x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f25k40t-i-ss

REGISTER 3-13: REVISION ID: REVISION ID REGISTER

R	R	R	R	R	R	R	R
1	0	1	0	MJRREV<5:2>			
bit 15							bit 8

R	R	R	R	R	R	R	R
MJRREV<1:0>		MNRREV<5:0>					
bit 7							bit 0

Legend:			
R = Readable bit	'1' = Bit is set	0' = Bit is cleared	x = Bit is unknown

bit 15-12 **Read as '1010'**
These bits are fixed with value '1010' for all devices in this family.

bit 11-6 **MJRREV<5:0>:** Major Revision ID bits
These bits are used to identify a major revision. A major revision is indicated by an all-layer revision (A0, B0, C0, etc.).
Revision A = 6'b00_0000

bit 5-0 **MNRREV<5:0>:** Minor Revision ID bits
These bits are used to identify a minor revision.

REGISTER 4-4: OSCSTAT: OSCILLATOR STATUS REGISTER 1

R-q/q	R-q/q	R-q/q	R-q/q	R-q/q	R-q/q	U-0	R-q/q
EXTOR	HFOR	MFOR	LFOR	SOR	ADOR	—	PLLR
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 = Reset value is determined by hardware

bit 7 **EXTOR:** EXTOSC (external) Oscillator Ready bit
1 = The oscillator is ready to be used
0 = The oscillator is not enabled, or is not yet ready to be used

bit 6 **HFOR:** HFINTOSC Oscillator Ready bit
1 = The oscillator is ready to be used
0 = The oscillator is not enabled, or is not yet ready to be used

bit 5 **MFOR:** MFINTOSC Oscillator Ready
1 = The oscillator is ready to be used
0 = The oscillator is not enabled, or is not yet ready to be used

bit 4 **LFOR:** LFINTOSC Oscillator Ready bit
1 = The oscillator is ready to be used
0 = The oscillator is not enabled, or is not yet ready to be used

bit 3 **SOR:** Secondary (Timer1) Oscillator Ready bit
1 = The oscillator is ready to be used
0 = The oscillator is not enabled, or is not yet ready to be used

bit 2 **ADOR:** ADC Oscillator Ready bit
1 = The oscillator is ready to be used
0 = The oscillator is not enabled, or is not yet ready to be used

bit 1 **Unimplemented:** Read as ‘0’

bit 0 **PLLR:** PLL is Ready bit
1 = The PLL is ready to be used
0 = The PLL is not enabled, the required input source is not ready, or the PLL is not locked.

5.5 Register Definitions: Reference Clock

Long bit name prefixes for the Reference Clock peripherals are shown in Table 5-1. Refer to **Section 1.4.2.2 "Long Bit Names"** for more information.

TABLE 5-1:

Peripheral	Bit Name Prefix
CLKR	CLKR

REGISTER 5-1: CLKRCON: REFERENCE CLOCK CONTROL REGISTER

R/W-0/0	U-0	U-0	R/W-1/1	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
EN	—	—	DC<1:0>		DIV<2:0>		
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 **EN:** Reference Clock Module Enable bit
- 1 = Reference clock module enabled
- 0 = Reference clock module is disabled

bit 6-5 **Unimplemented:** Read as '0'

bit 4-3 **DC<1:0>:** Reference Clock Duty Cycle bits(1)
- 11 = Clock outputs duty cycle of 75%
- 10 = Clock outputs duty cycle of 50%
- 01 = Clock outputs duty cycle of 25%
- 00 = Clock outputs duty cycle of 0%

bit 2-0 **DIV<2:0>:** Reference Clock Divider bits
- 111 = Base clock value divided by 128
- 110 = Base clock value divided by 64
- 101 = Base clock value divided by 32
- 100 = Base clock value divided by 16
- 011 = Base clock value divided by 8
- 010 = Base clock value divided by 4
- 001 = Base clock value divided by 2
- 000 = Base clock value

Note 1: Bits are valid for reference clock divider values of two or larger, the base clock cannot be further divided.

12.0 8x8 HARDWARE MULTIPLIER

12.1 Introduction

All PIC18 devices include an 8x8 hardware multiplier as part of the ALU. The multiplier performs an unsigned operation and yields a 16-bit result that is stored in the product register pair, PRODH:PRODL. The multiplier's operation does not affect any flags in the STATUS register.

Making multiplication a hardware operation allows it to be completed in a single instruction cycle. This has the advantages of higher computational throughput and reduced code size for multiplication algorithms and allows the PIC18 devices to be used in many applications previously reserved for digital signal processors. A comparison of various hardware and software multiply operations, along with the savings in memory and execution time, is shown in Table 12-1.

12.2 Operation

Example 12-1 shows the instruction sequence for an 8x8 unsigned multiplication. Only one instruction is required when one of the arguments is already loaded in the WREG register.

Example 12-2 shows the sequence to do an 8x8 signed multiplication. To account for the sign bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

EXAMPLE 12-1: 8x8 UNSIGNED MULTIPLY ROUTINE

```
MOVF   ARG1, W   ;
MULWF  ARG2      ; ARG1 * ARG2 ->
                 ; PRODH:PRODL
```

EXAMPLE 12-2: 8x8 SIGNED MULTIPLY ROUTINE

```
MOVF   ARG1, W
MULWF  ARG2      ; ARG1 * ARG2 ->
                 ; PRODH:PRODL
BTFSC  ARG2, SB  ; Test Sign Bit
SUBWF  PRODH, F  ; PRODH = PRODH
                 ;          - ARG1
MOVF   ARG2, W
BTFSC  ARG1, SB  ; Test Sign Bit
SUBWF  PRODH, F  ; PRODH = PRODH
                 ;          - ARG2
```

TABLE 12-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time			
				@ 64 MHz	@ 40 MHz	@ 10 MHz	@ 4 MHz
8x8 unsigned	Without hardware multiply	13	69	4.3 μs	6.9 μs	27.6 μs	69 μs
	Hardware multiply	1	1	62.5 ns	100 ns	400 ns	1 μs
8x8 signed	Without hardware multiply	33	91	5.7 μs	9.1 μs	36.4 μs	91 μs
	Hardware multiply	6	6	375 ns	600 ns	2.4 μs	6 μs
16x16 unsigned	Without hardware multiply	21	242	15.1 μs	24.2 μs	96.8 μs	242 μs
	Hardware multiply	28	28	1.8 μs	2.8 μs	11.2 μs	28 μs
16x16 signed	Without hardware multiply	52	254	15.9 μs	25.4 μs	102.6 μs	254 μs
	Hardware multiply	35	40	2.5 μs	4.0 μs	16.0 μs	40 μs

13.6 CRC Check Value

The CRC check value will be located in the CRCACC registers after the CRC calculation has finished. The check value will depend on two mode settings of the CRCCON register: ACCM and SHIFTM.

When the ACCM bit is set, the CRC module augments the data with a number of zeros equal to the length of the polynomial to align the final check value. When the ACCM bit is not set, the CRC will stop at the end of the data. A number of zeros equal to the length of the polynomial can then be entered into CRCDAT to find the same check value as augmented mode. Alternatively, the expected check value can be entered at this point to make the final result equal 0.

When the CRC check value is computed with the SHIFTM bit set, selecting LSb first, and the ACCM bit is set then the final value in the CRCACC registers will be reversed such that the LSb will be in the MSb position and vice versa. This is the expected check value in bit reversed form. If you are creating a check value to be appended to a data stream then a bit reversal must be performed on the final value to achieve the correct checksum. You can use the CRC to do this reversal by the following method:

- Save CRCACC value in user RAM space
- Clear the CRCACC registers
- Clear the CRCXOR registers
- Write the saved CRCACC value to the CRCDAT input

The properly oriented check value will be in the CRCACC registers as the result.

13.7 CRC Interrupt

The CRC will generate an interrupt when the BUSY bit transitions from 1 to 0. The CRCIF Interrupt Flag bit of the PIR7 register is set every time the BUSY bit transitions, regardless of whether or not the CRC interrupt is enabled. The CRCIF bit can only be cleared in software. The CRC interrupt enable is the CRCIE bit of the PIE7 register.

13.8 Configuring the CRC

The following steps illustrate how to properly configure the CRC.

1. Determine if the automatic program memory scan will be used with the scanner or manual calculation through the SFR interface and perform the actions specified in **Section 13.5 "CRC Data Sources"**, depending on which decision was made.
2. If desired, seed a starting CRC value into the CRCACCH/L registers.
3. Program the CRCXORH/L registers with the desired generator polynomial.
4. Program the DLEN<3:0> bits of the CRCCON1 register with the length of the data word - 1 (refer to Example 13-1). This determines how many times the shifter will shift into the accumulator for each data word.
5. Program the PLEN<3:0> bits of the CRCCON1 register with the length of the polynomial -2 (refer to Example 13-1).
6. Determine whether shifting in trailing zeros is desired and set the ACCM bit of the CRCCON0 register appropriately.
7. Likewise, determine whether the MSb or LSb should be shifted first and write the SHIFTM bit of the CRCCON0 register appropriately.
8. Write the CRCGO bit of the CRCCON0 register to begin the shifting process.

9a. If manual SFR entry is used, monitor the FULL bit of the CRCCON0 register. When FULL = 0, another word of data can be written to the CRCDATH/L registers, keeping in mind that CRCDATH should be written first if the data has more than eight bits, as the shifter will begin upon the CRCDATL register being written.

9b. If the scanner is used, the scanner will automatically stuff words into the CRCDATH/L registers as needed, as long as the SCANGO bit is set.

10a. If using the Flash memory scanner, monitor the SCANIF (or the SCANGO bit) for the scanner to finish pushing information into the CRCDATA registers. After the scanner is completed, monitor the BUSY bit to determine that the CRC has been completed and the check value can be read from the CRCACC registers. If both the interrupt flags are set (or both BUSY and SCANGO bits are cleared), the completed CRC calculation can be read from the CRCACCH/L registers.

10b. If manual entry is used, monitor the BUSY bit to determine when the CRCACC registers will hold the check value.

REGISTER 14-6: PIR4: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 4

U-0	U-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
—	—	TMR6IF	TMR5IF	TMR4IF	TMR3IF	TMR2IF	TMR1IF
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 7-6 **Unimplemented:** Read as '0'

bit 5 **TMR6IF:** TMR6 to PR6 Match Interrupt Flag bit
1 = TMR6 to PR6 match occurred (must be cleared in software)
0 = No TMR6 to PR6 match occurred

bit 4 **TMR5IF:** TMR5 Overflow Interrupt Flag bit
1 = TMR5 register overflowed (must be cleared in software)
0 = TMR5 register did not overflow

bit 3 **TMR4IF:** TMR4 to PR4 Match Interrupt Flag bit
1 = TMR4 to PR4 match occurred (must be cleared in software)
0 = No TMR4 to PR4 match occurred

bit 2 **TMR3IF:** TMR3 Overflow Interrupt Flag bit
1 = TMR3 register overflowed (must be cleared in software)
0 = TMR3 register did not overflow

bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

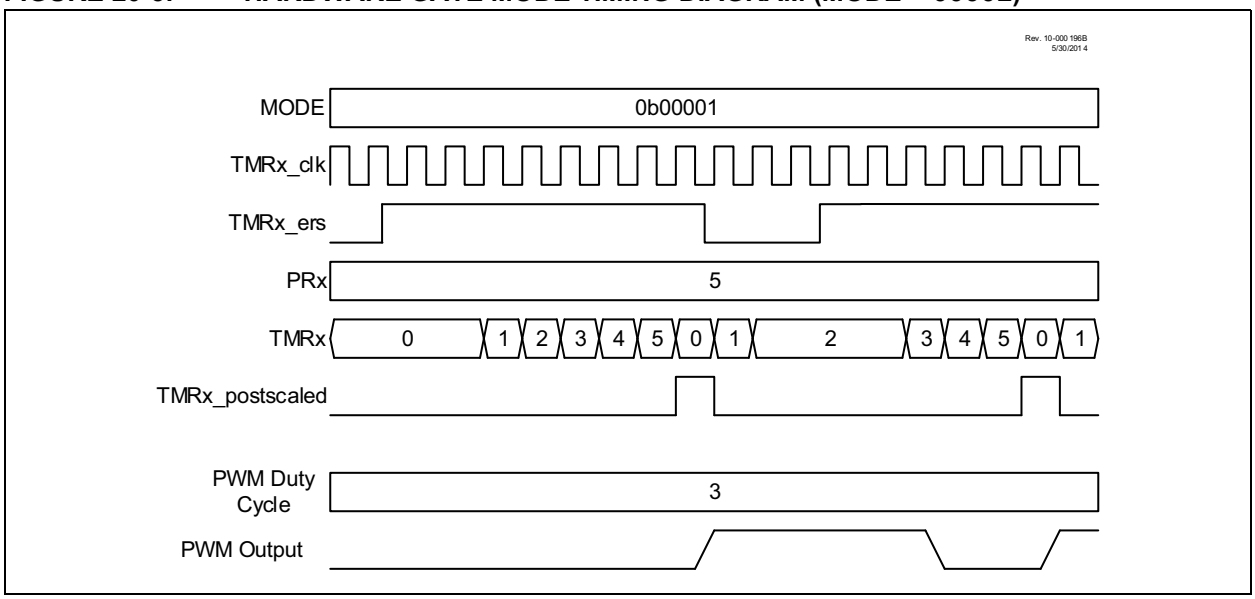
20.5.2 HARDWARE GATE MODE

The Hardware Gate modes operate the same as the Software Gate mode except the TMRx_ers external signal can also gate the timer. When used with the CCP the gating extends the PWM period. If the timer is stopped when the PWM output is high then the duty cycle is also extended.

When MODE<4:0> = 00001 then the timer is stopped when the external signal is high. When MODE<4:0> = 00010 then the timer is stopped when the external signal is low.

Figure 20-5 illustrates the Hardware Gating mode for MODE<4:0> = 00001 in which a high input level starts the counter.

FIGURE 20-5: HARDWARE GATE MODE TIMING DIAGRAM (MODE = 00001)



REGISTER 21-2: CCPTMRS: CCP TIMERS CONTROL REGISTER

R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1
P4TSEL<1:0>		P3TSEL<1:0>		C2TSEL<1:0>		C1TSEL<1:0>	
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	x = Bit is unknown

bit 7-6 **P4TSEL<1:0>:** PWM4 Timer Selection bits
11 = PWM4 based on TMR6
10 = PWM4 based on TMR4
01 = PWM4 based on TMR2
00 = Reserved

bit 5-4 **P3TSEL<1:0>:** PWM3 Timer Selection bits
11 = PWM3 based on TMR6
10 = PWM3 based on TMR4
01 = PWM3 based on TMR2
00 = Reserved

bit 3-2 **C2TSEL<1:0>:** CCP2 Timer Selection bits
11 = CCP2 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
10 = CCP2 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
01 = CCP2 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
00 = Reserved

bit 1-0 **C1TSEL<1:0>:** CCP1 Timer Selection bits
11 = CCP1 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
10 = CCP1 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
01 = CCP1 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
00 = Reserved

FIGURE 24-1: SIMPLIFIED CWG BLOCK DIAGRAM (HALF-BRIDGE MODE, MODE<2:0> = 100)

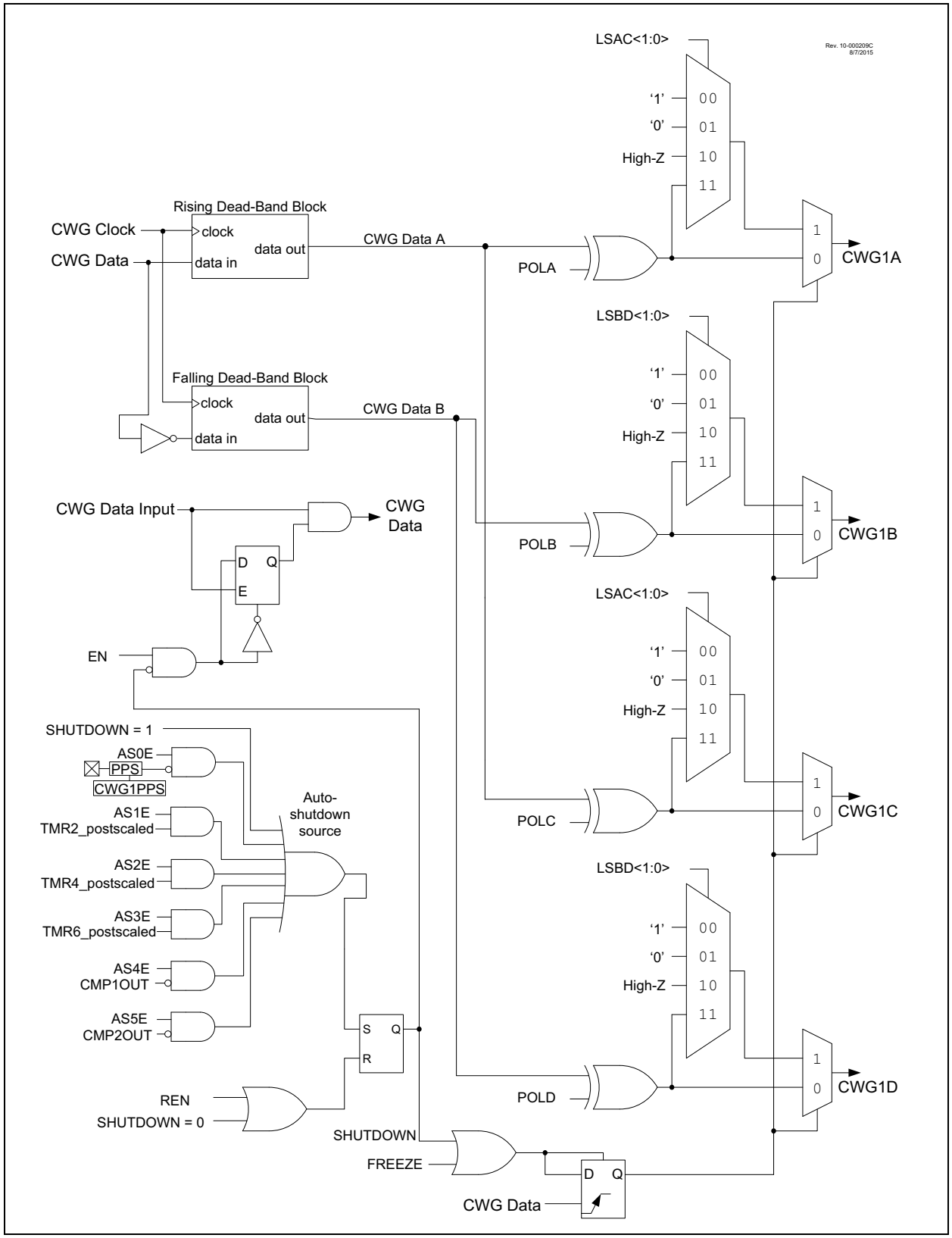
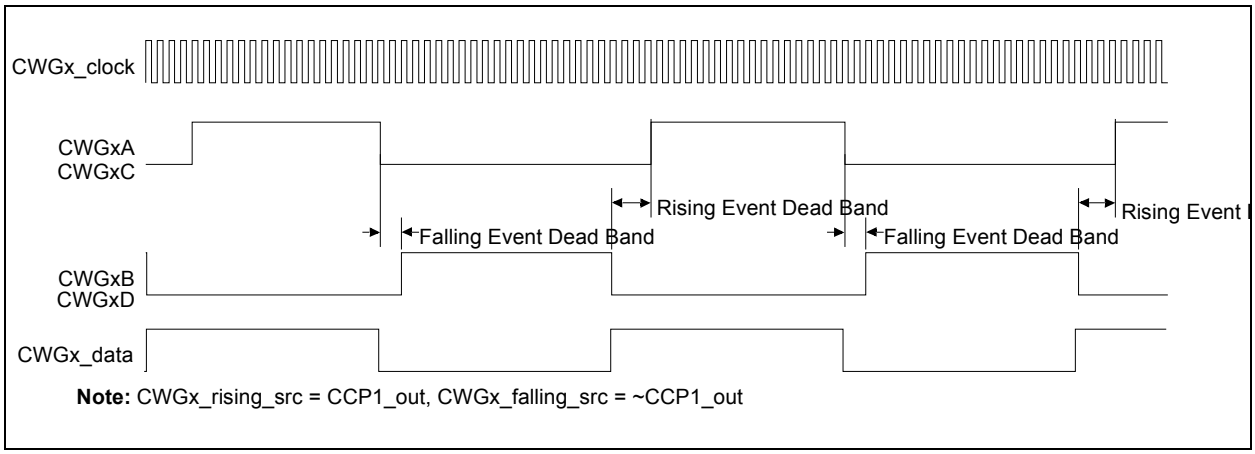


FIGURE 24-2: CWG1 HALF-BRIDGE MODE OPERATION



24.2.2 PUSH-PULL MODE

In Push-Pull mode, two output signals are generated, alternating copies of the input as illustrated in Figure 24-4. This alternation creates the push-pull effect required for driving some transformer-based power supply designs. Steering modes are not used in Push-Pull mode. A basic block diagram for the Push-Pull mode is shown in Figure 24-3.

The push-pull sequencer is reset whenever EN = 0 or if an auto-shutdown event occurs. The sequencer is clocked by the first input pulse, and the first output appears on CWG1A.

The unused outputs CWG1C and CWG1D drive copies of CWG1A and CWG1B, respectively, but with polarity controlled by the POLC and POLD bits of the CWG1CON1 register, respectively.

24.14 Register Definitions: CWG Control

Long bit name prefixes for the CWG peripheral is shown in Table 24-2. Refer to **Section 1.4.2.2 "Long Bit Names"** for more information.

TABLE 24-2:

Peripheral	Bit Name Prefix
CWG	CWG

I

REGISTER 24-1: CWG1CON0: CWG CONTROL REGISTER 0

R/W-0/0	R/W/HC-0/0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
EN	LD(1)	—	—	—	MODE<2:0>		
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	HC = Bit is cleared by hardware

bit 7 **EN:** CWG1 Enable bit
- 1 = Module is enabled
- 0 = Module is disabled

bit 6 **LD:** CWG1 Load Buffers bit(1)
- 1 = Dead-band count buffers to be loaded on CWG data rising edge, following first falling edge after this bit is set
- 0 = Buffers remain unchanged

bit 5-3 **Unimplemented**: Read as '0'

bit 2-0 **MODE<2:0>**: CWG1 Mode bits
- 111 = Reserved
- 110 = Reserved
- 101 = CWG outputs operate in Push-Pull mode
- 100 = CWG outputs operate in Half-Bridge mode
- 011 = CWG outputs operate in Reverse Full-Bridge mode
- 010 = CWG outputs operate in Forward Full-Bridge mode
- 001 = CWG outputs operate in Synchronous Steering mode
- 000 = CWG outputs operate in Asynchronous Steering mode

Note 1: This bit can only be set after EN = 1; it cannot be set in the same cycle when EN is set.

26.10.8 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN bit of the SSPxCON2 register. When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge data bit are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG. The SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSP module then goes into Idle mode (Figure 26-30).

26.10.8.1 WCOL Status Flag

If the user writes the SSPxBUF when an Acknowledge sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

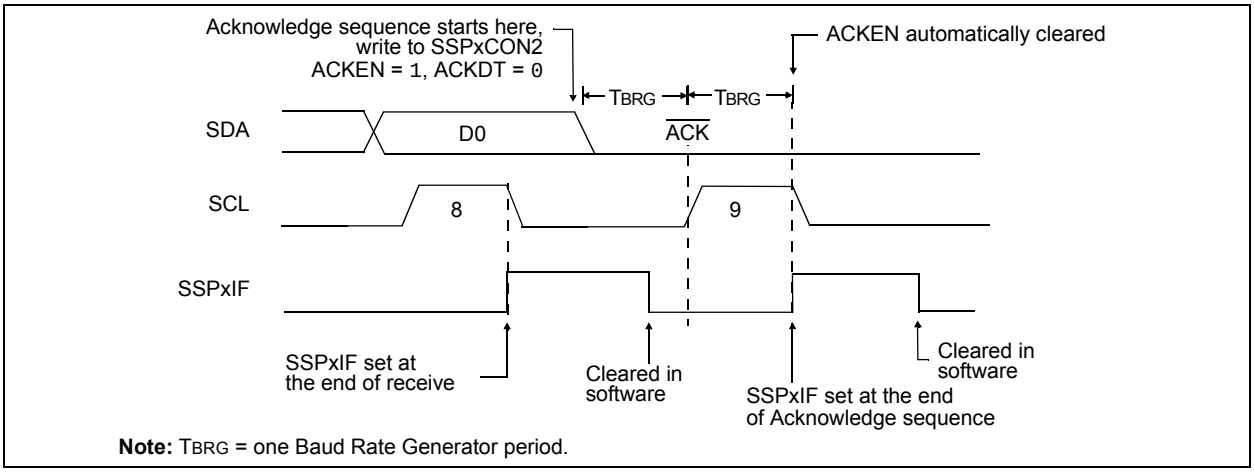
26.10.9 STOP CONDITION TIMING

A Stop bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit, PEN bit of the SSPxCON2 register. At the end of a receive/transmit, the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the Baud Rate Generator is reloaded and counts down to '0'. When the Baud Rate Generator times out, the SCL pin will be brought high and one TBRG (Baud Rate Generator rollover count) later, the SDA pin will be deasserted. When the SDA pin is sampled high while SCL is high, the P bit of the SSPxSTAT register is set. A TBRG later, the PEN bit is cleared and the SSPxIF bit is set (Figure 26-31).

26.10.9.1 WCOL Status Flag

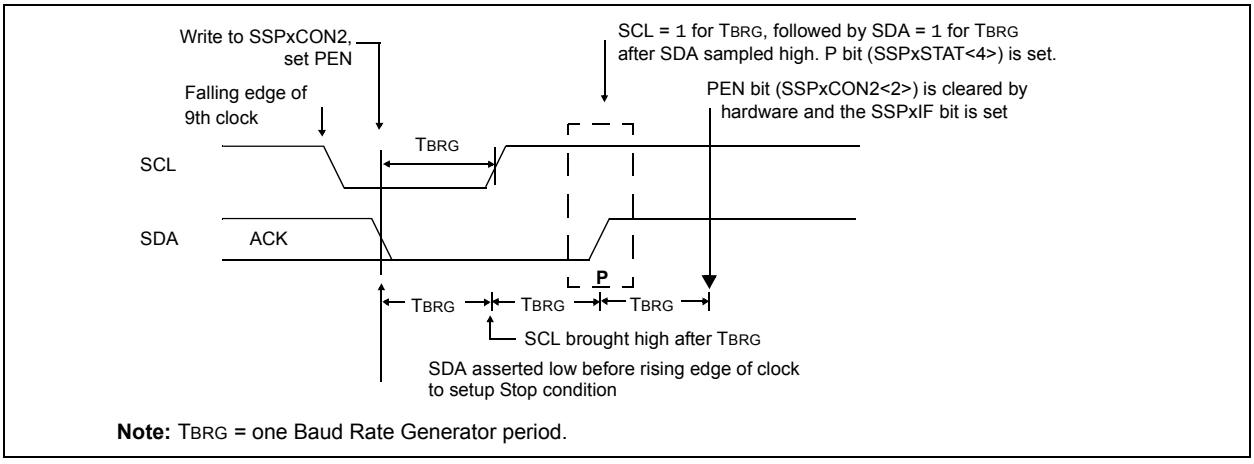
If the user writes the SSPxBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

FIGURE 26-30: ACKNOWLEDGE SEQUENCE WAVEFORM



Note: TBRG = one Baud Rate Generator period.

FIGURE 26-31: STOP CONDITION RECEIVE OR TRANSMIT MODE



Note: TBRG = one Baud Rate Generator period.

31.1.5 INTERRUPTS

The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital conversion. The ADC Interrupt Flag is the ADIF bit in the PIR1 register. The ADC Interrupt Enable is the ADIE bit in the PIE1 register. The ADIF bit must be cleared in software.

Note 1:	The ADIF bit is set at the completion of every conversion, regardless of whether or not the ADC interrupt is enabled.
2:	The ADC operates during Sleep only when the FRC oscillator is selected.

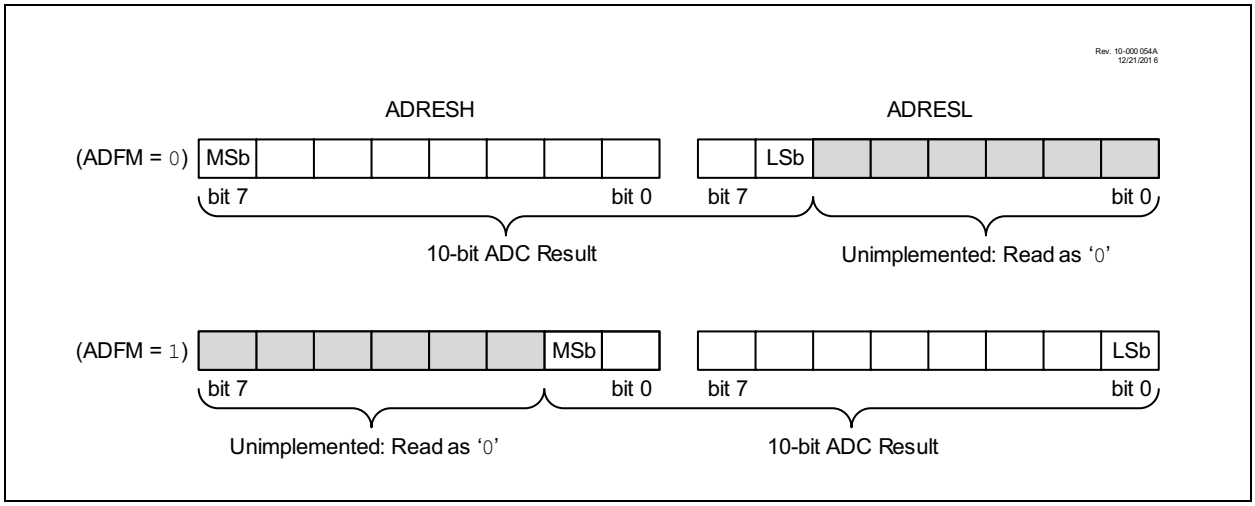
This interrupt can be generated while the device is operating or while in Sleep. If the device is in Sleep, the interrupt will wake-up the device. Upon waking from Sleep, the next instruction following the SLEEP instruction is always executed. If the user is attempting to wake-up from Sleep and resume in-line code execution, the ADIE bit of the PIE1 register and the PEIE bit of the INTCON register must both be set and the GIE bit of the INTCON register must be cleared. If all three of these bits are set, the execution will switch to the Interrupt Service Routine.

31.1.6 RESULT FORMATTING

The 10-bit ADC conversion result can be supplied in two formats, left justified or right justified. The ADFM bits of the ADCON0 register controls the output format.

Figure 31-3 shows the two output formats.

FIGURE 31-3: 10-BIT ADC CONVERSION RESULT FORMAT



REGISTER 31-6: ADCLK: ADC CLOCK SELECTION REGISTER

U-0	U-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
—	—	ADCS<5:0>					
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 7-6 **Unimplemented**: Read as '0'

bit 5-0 **ADCS<5:0>**: ADC Conversion Clock Select bits
111111 = FOSC/128
111110 = FOSC/126
111101 = FOSC/124
•
•
•
000000 = FOSC/2

REGISTER 31-7: ADREF: ADC REFERENCE SELECTION REGISTER

U-0	U-0	U-0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	—	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 7-5 **Unimplemented**: Read as '0'

bit 4 **ADNREF**: ADC Negative Voltage Reference Selection bit
1 = VREF- is connected to external VREF-
0 = VREF- is connected to AVSS

bit 3-2 **Unimplemented**: Read as '0'

bit 1-0 **ADPREF**: ADC Positive Voltage Reference Selection bits
11 = VREF+ is connected to internal Fixed Voltage Reference (FVR) module
10 = VREF+ is connected to external VREF+
01 = Reserved
00 = VREF+ is connected to VDD

32.3 Comparator Control

Each comparator has two control registers: CMxCON0 and CMxCON1.

The CMxCON0 register (see Register 32-1) contains Control and Status bits for the following:

- Enable
- Output
- Output polarity
- Hysteresis enable
- Timer1 output synchronization

The CMxCON1 register (see Register 32-2) contains Control bits for the following:

- Interrupt on positive/negative edge enables
- Positive input channel selection
- Negative input channel selection

32.3.1 COMPARATOR ENABLE

Setting the EN bit of the CMxCON0 register enables the comparator for operation. Clearing the CxEN bit disables the comparator resulting in minimum current consumption.

32.3.2 COMPARATOR OUTPUT

The output of the comparator can be monitored by reading either the CxOUT bit of the CMxCON0 register or the MCxOUT bit of the CMOUT register.

The comparator output can also be routed to an external pin through the RxyPPS register (Register 17-2). The corresponding TRIS bit must be clear to enable the pin as an output.

Note 1:	The internal output of the comparator is latched with each instruction cycle. Unless otherwise specified, external outputs are not latched.

32.3.3 COMPARATOR OUTPUT POLARITY

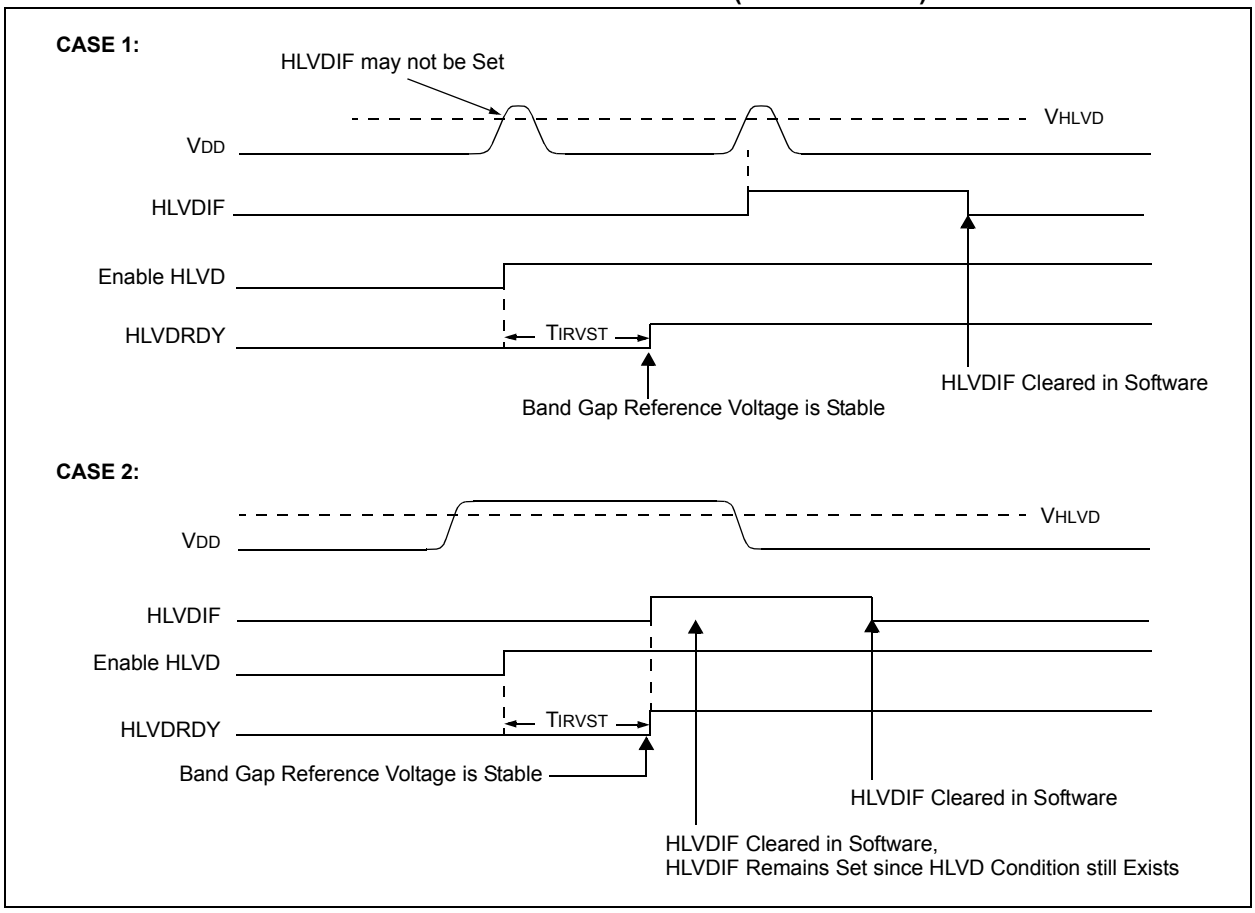
Inverting the output of the comparator is functionally equivalent to swapping the comparator inputs. The polarity of the comparator output can be inverted by setting the CxPOL bit of the CMxCON0 register. Clearing the CxPOL bit results in a non-inverted output.

Table 32-2 shows the output state versus input conditions, including polarity control.

TABLE 32-2: COMPARATOR OUTPUT STATE VS. INPUT CONDITIONS

Input Condition	CxPOL	CxOUT
CxVN > CxVP	0	0
CxVN < CxVP	0	1
CxVN > CxVP	1	1
CxVN < CxVP	1	0

FIGURE 33-3: HIGH-VOLTAGE DETECT OPERATION (HLVDINTH = 1)

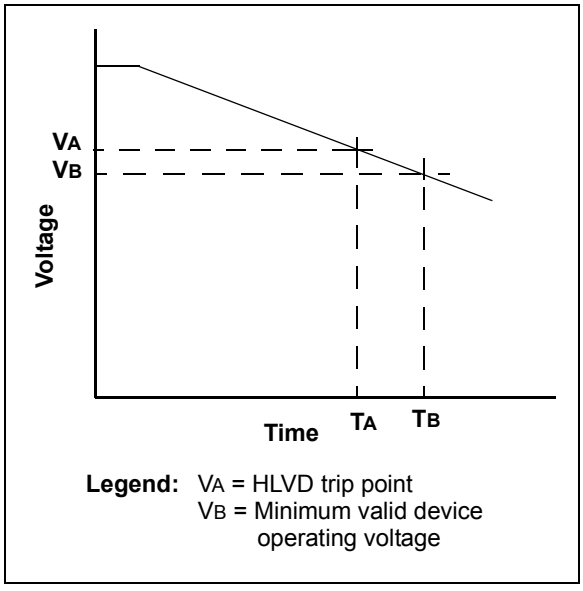


33.5 Applications

In many applications, it is desirable to detect a drop below, or rise above, a particular voltage threshold. For example, the HLVD module could be periodically enabled to detect Universal Serial Bus (USB) attach or detach. This assumes the device is powered by a lower voltage source than the USB when detached. An attach would indicate a High-Voltage Detect from, for example, 3.3V to 5V (the voltage on USB) and vice versa for a detach. This feature could save a design a few extra components and an attach signal (input pin).

For general battery applications, Figure 33-4 shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage, VA, the HLVD logic generates an interrupt at time, TA. The interrupt could cause the execution of an Interrupt Service Routine (ISR), which would allow the application to perform “housekeeping tasks” and a controlled shutdown before the device voltage exits the valid operating range at TB. This would give the application a time window, represented by the difference between TA and TB, to safely exit.

FIGURE 33-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION



RLNCF **Rotate Left f (No Carry)**

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

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

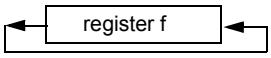
Operation: (f<n>) → dest<n + 1>,
(f<7>) → dest<0>

Status Affected: N, Z

Encoding:

0100	01da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the left. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored 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 ≤ 95 (5Fh). See **Section 35.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: RLNCF REG, 1, 0

Before Instruction
REG = 1010 1011
After Instruction
REG = 0101 0111

RRCF **Rotate Right f through Carry**

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

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

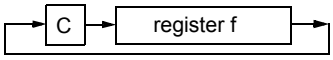
Operation: (f<n>) → dest<n – 1>,
(f<0>) → C,
(C) → dest<7>

Status Affected: C, N, Z

Encoding:

0011	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right through the CARRY flag. 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 ≤ 95 (5Fh). See **Section 35.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: RRCF REG, 0, 0

Before Instruction
REG = 1110 0110
C = 0
After Instruction
REG = 1110 0110
W = 0111 0011
C = 0

35.2.2 EXTENDED INSTRUCTION SET

ADDFSR **Add Literal to FSR**

Syntax: ADDFSR f, k

Operands: $0 \le k \le 63$
$f \in [0, 1, 2]$

Operation: FSR(f) + k → FSR(f)

Status Affected: None

Encoding:

1110	1000	ffkk	kkkk

Description: The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ADDFSR 2, 23h`

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 0422h

ADDULNK **Add Literal to FSR2 and Return**

Syntax: ADDULNK k

Operands: $0 \le k \le 63$

Operation: FSR2 + k → FSR2,
(TOS) → PC

Status Affected: None

Encoding:

1110	1000	11kk	kkkk

Description: The 6-bit literal 'k' is added to the contents of FSR2. A RETURN is then executed by loading the PC with the TOS.
The instruction takes two cycles to execute; a NOP is performed during the second cycle.
This may be thought of as a special case of the ADDFSR instruction, where f = 3 (binary '11'); it operates only on FSR2.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to FSR
No Operation	No Operation	No Operation	No Operation

Example: `ADDULNK 23h`

Before Instruction
FSR2 = 03FFh
PC = 0100h
After Instruction
FSR2 = 0422h
PC = (TOS)

Note: All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction syntax then becomes: {label} instruction argument(s).

TABLE 37-3: POWER-DOWN CURRENT (I_{PD})(1,2)

PIC18LF24/25K40				Standard Operating Conditions (unless otherwise stated)					
PIC18F24/25K40				Standard Operating Conditions (unless otherwise stated) VREGPM = 1					
Param. No.	Symbol	Device Characteristics	Min.	Typ.†	Max. +85°C	Max. +125°C	Units	Conditions	
								V_{DD}	Note
D200	I_{PD}	I_{PD} Base	—	0.05	2	9	µA	3.0V	
D200 D200A	I_{PD}	I_{PD} Base	—	0.4	4	12	µA	3.0V	
			—	20	—	—	µA	3.0V	VREGPM = 0
D201	I_{PD_WDT}	Low-Frequency Internal Oscillator/ WDT	—	0.4	3	10	µA	3.0V	
D201	I_{PD_WDT}	Low-Frequency Internal Oscillator/ WDT	—	0.6	5	13	µA	3.0V	
D202	I_{PD_SOSC}	Secondary Oscillator (S_{OSC})	—	0.6	5	13	µA	3.0V	
D202	I_{PD_SOSC}	Secondary Oscillator (S_{OSC})	—	0.8	8.5	15	µA	3.0V	
D203	I_{PD_FVR}	FVR	—	31	—	—	µA	3.0V	FVRCON = 0X81 or 0x84
D203	I_{PD_FVR}	FVR	—	32	—	—	µA	3.0V	FVRCON = 0X81 or 0x84
D204	I_{PD_BOR}	Brown-out Reset (BOR)	—	9	14	18	µA	3.0V	
D204	I_{PD_BOR}	Brown-out Reset (BOR)	—	14	19	21	µA	3.0V	
D205	I_{PD_LPBOR}	Low-Power Brown-out Reset (LPBOR)	—	0.5	—	—	µA	3.0V	
D205	I_{PD_LPBOR}	Low-Power Brown-out Reset (LPBOR)	—	0.7	—	—	µA	3.0V	
D206	I_{PD_HLVD}	High/Low Voltage Detect (HLVD)	—	31	—	—	µA	3.0V	
D206	I_{PD_HLVD}	High/Low Voltage Detect (HLVD)	—	32	—	—	µA	3.0V	
D207	I_{PD_ADCA}	ADC - Active	—	250	—	—	µA	3.0V	ADC is converting (4)
D207	I_{PD_ADCA}	ADC - Active	—	280	—	—	µA	3.0V	ADC is converting (4)
D208	I_{PD_CMP}	Comparator	—	25	38	40	µA	3.0V	
D208	I_{PD_CMP}	Comparator	—	28	50	60	µA	3.0V	

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

Note 1: The peripheral current is the sum of the base I_{DD} and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base I_{DD} or I_{PD} current from this limit. Max. values should be used when calculating total current consumption.

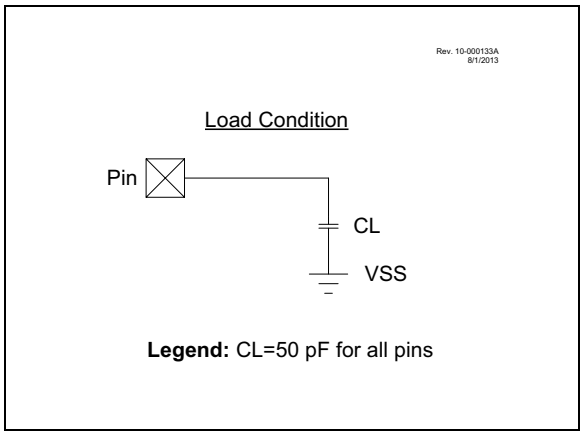
2: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode with all I/O pins in high-impedance state and tied to V_{SS}.

3: All peripheral currents listed are on a per-peripheral basis if more than one instance of a peripheral is available.

4: ADC clock source is FRC.

37.4 AC Characteristics

FIGURE 37-4: LOAD CONDITIONS



Legend: CL=50 pF for all pins