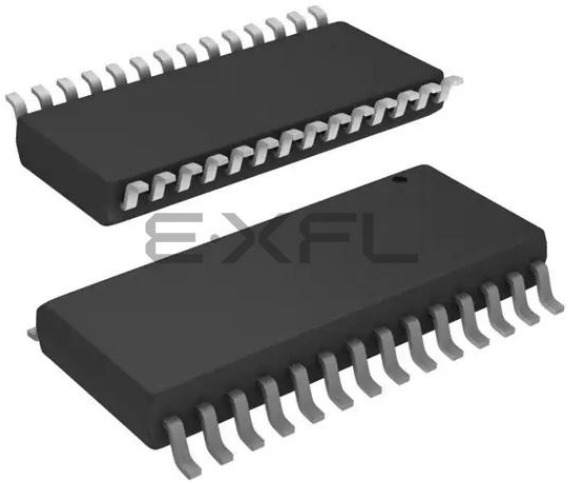


Welcome to E-XFL.COM

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	128KB (64K 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 ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf27k40-i-so

REGISTER 3-3: Configuration Word 2L (30 0002h): Supervisor

R/W-1	R/W-1	R/W-1	U-1	U-1	U-1	R/W-1	R/W-1
BOREN<1:0>		$\overline{\text{LPBOREN}}$	—	—	—	$\overline{\text{PWRTE}}$	MCLRE
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 **BOREN<1:0>:** Brown-out Reset Enable bits
When enabled, Brown-out Reset Voltage (V_{BOR}) is set by BORV bit
11 = Brown-out Reset enabled, SBOREN bit is ignored
10 = Brown-out Reset enabled while running, disabled in Sleep; SBOREN is ignored
01 = Brown-out Reset enabled according to SBOREN
00 = Brown-out Reset disabled

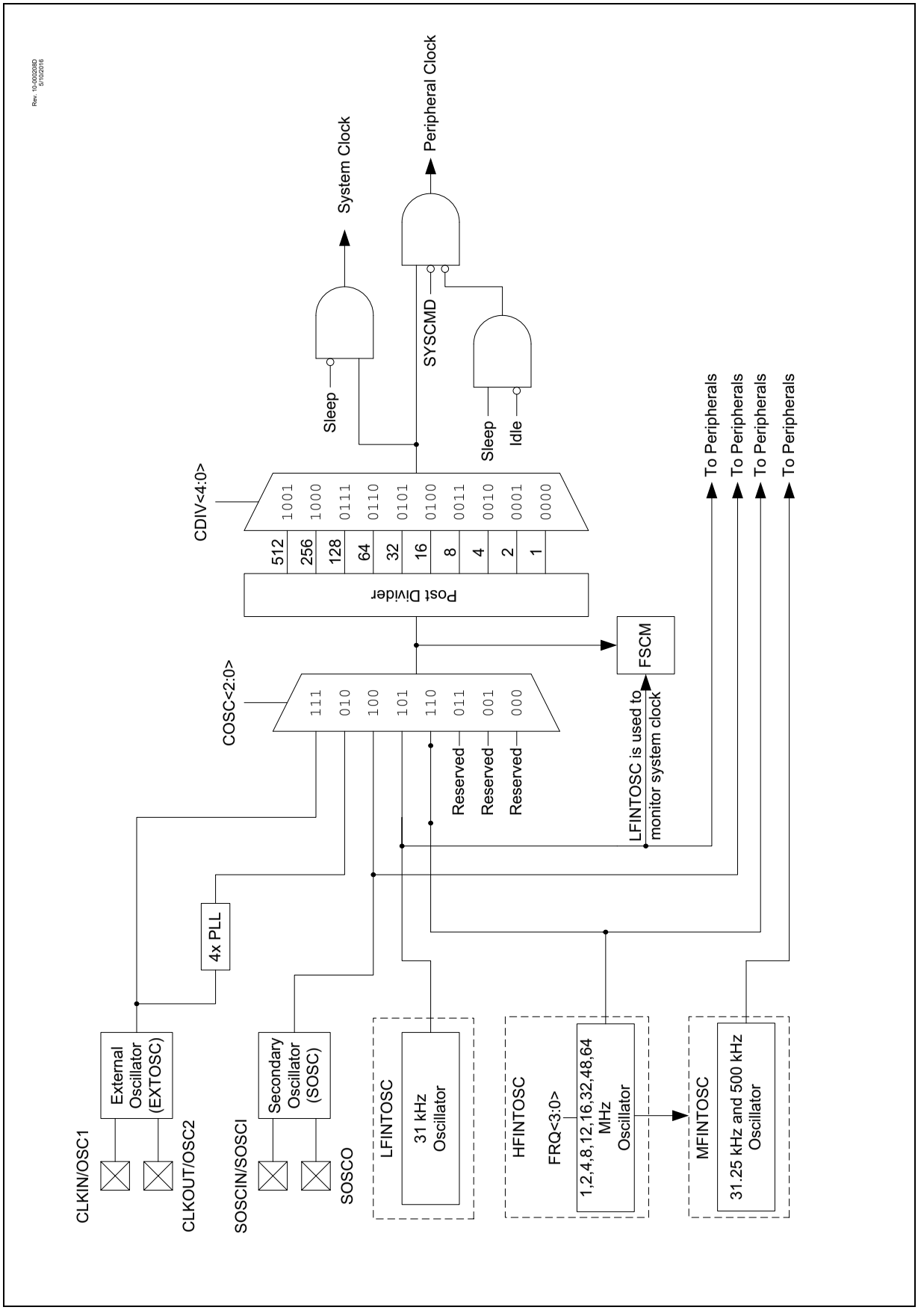
bit 5 **$\overline{\text{LPBOREN}}$**: Low-Power BOR Enable bit
1 = Low-Power Brown-out Reset is disabled
0 = Low-Power Brown-out Reset is enabled

bit 4-2 **Unimplemented:** Read as '1'

bit 1 **$\overline{\text{PWRTE}}$:** Power-up Timer Enable bit
1 = PWRT disabled
0 = PWRT enabled

bit 0 **MCLRE:** Master Clear ($\overline{\text{MCLR}}$) Enable bit
If LVP = 1
RE3 pin function is $\overline{\text{MCLR}}$
If LVP = 0
1 = $\overline{\text{MCLR}}$ pin is $\overline{\text{MCLR}}$
0 = $\overline{\text{MCLR}}$ pin function is port defined function

FIGURE 4-1: SIMPLIFIED PIC® MCU CLOCK SOURCE BLOCK DIAGRAM



REGISTER 4-6: OSCTUNE: HFINTOSC TUNING 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
—	—	HFTUN<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 **HFTUN<5:0>:** HFINTOSC Frequency Tuning bits

01 1111 = Maximum frequency

•

•

•

00 0000 = Center frequency. Oscillator module is running at the calibrated frequency (default value).

•

•

•

10 0000 = Minimum frequency

6.2.1 WAKE-UP FROM SLEEP

The device can wake-up from Sleep through one of the following events:

1. External Reset input on $\overline{\text{MCLR}}$ pin, if enabled
2. BOR Reset, if enabled
3. Low-Power Brown-Out Reset (LPBOR), if enabled
4. POR Reset
5. Windowed Watchdog Timer, if enabled
6. All interrupt sources except clock switch interrupt can wake-up the part.

The first five events will cause a device Reset. The last one event is considered a continuation of program execution. To determine whether a device Reset or wake-up event occurred, refer to **Section 8.13 "Determining the Cause of a Reset"**.

When the SLEEP instruction is being executed, the next instruction (PC + 2) is prefetched. For the device to wake-up through an interrupt event, the corresponding Interrupt Enable bit must be enabled, as well as the Peripheral Interrupt Enable bit (PEIE = 1), for every interrupt not in PIR0. Wake-up will occur regardless of the state of the GIE bit. If the GIE bit is disabled, the device continues execution at the instruction after the SLEEP instruction. If the GIE bit is enabled, the device executes the instruction after the SLEEP instruction, the device will then call the Interrupt Service Routine. In cases where the execution of the instruction following SLEEP is not desirable, the user should have a NOP after the SLEEP instruction.

The WDT is cleared when the device wakes-up from Sleep, regardless of the source of wake-up.

Upon a wake from a Sleep event, the core will wait for a combination of three conditions before beginning execution. The conditions are:

- PFM Ready
- COSC-Selected Oscillator Ready
- BOR Ready (unless BOR is disabled)

6.2.2 WAKE-UP USING INTERRUPTS

When global interrupts are disabled (GIE cleared) and any interrupt source, with the exception of the clock switch interrupt, has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If the interrupt occurs **before** the execution of a SLEEP instruction
  - SLEEP instruction will execute as a NOP
  - WDT and WDT prescaler will not be cleared
  - $\overline{\text{TO}}$ bit of the STATUS register will not be set
  - $\overline{\text{PD}}$ bit of the STATUS register will not be cleared
- If the interrupt occurs **during or after** the execution of a SLEEP instruction
  - SLEEP instruction will be completely executed
  - Device will immediately wake-up from Sleep
  - WDT and WDT prescaler will be cleared
  - $\overline{\text{TO}}$ bit of the STATUS register will be set
  - $\overline{\text{PD}}$ bit of the STATUS register will be cleared

Even if the flag bits were checked before executing a SLEEP instruction, it may be possible for flag bits to become set before the SLEEP instruction completes. To determine whether a SLEEP instruction executed, test the $\overline{\text{PD}}$ bit. If the $\overline{\text{PD}}$ bit is set, the SLEEP instruction was executed as a NOP.

TABLE 10-4: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F27/47K40 DEVICES

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

Note 1: Not available on LF parts

2: Not available on PIC18(L)F27K40 (28-pin variants).

REGISTER 13-9: CRCXORH: CRC XOR HIGH BYTE REGISTER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
X<15:8>							
bit 7							bit 0

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

bit 7-0 **X<15:8>**: XOR of Polynomial Term XN Enable bits

REGISTER 13-10: CRCXORL: CRC XOR LOW BYTE REGISTER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	U-1
X<7:1>							—
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 7-1 **X<7:1>**: XOR of Polynomial Term XN Enable bits

bit 0 **Unimplemented:** Read as ‘1’

REGISTER 14-4: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2

R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
HLVDIF	ZCDIF	—	—	—	—	C2IF	C1IF
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 **HLVDIF:** HLVD Interrupt Flag bit
1 = HLVD interrupt event has occurred
0 = HLVD interrupt event has not occurred or has not been set up

bit 6 **ZCDIF:** Zero-Cross Detect Interrupt Flag bit
1 = ZCD Output has changed (must be cleared in software)
0 = ZCD Output has not changed

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

bit 1 **C2IF:** Comparator 2 Interrupt Flag bit
1 = Comparator C2 output has changed (must be cleared by software)
0 = Comparator C2 output has not changed

bit 0 **C1IF:** Comparator 1 Interrupt Flag bit
1 = Comparator C1 output has changed (must be cleared by software)
0 = Comparator C1 output has not changed

REGISTER 14-17: PIE7: PERIPHERAL INTERRUPT ENABLE REGISTER 7

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	R/W-0/0
SCANIE	CRCIE	NVMIE	—	—	—	—	CWG1IE
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 **SCANIE:** SCAN Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **CRCIE:** CRC Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **NVMIE:** NVM Interrupt Enable bit
1 = Enabled
0 = Disabled

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

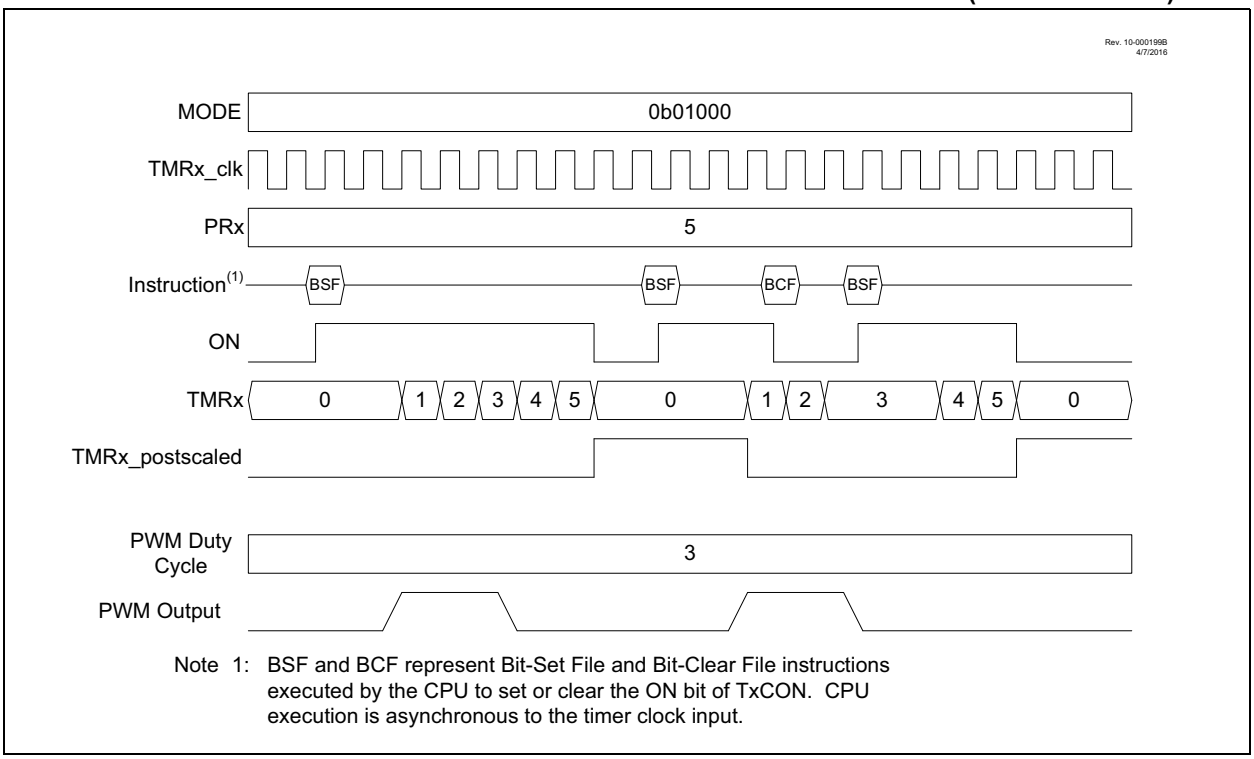
bit 0 **CWG1IE:** CWG Interrupt Enable bit
1 = Enabled
0 = Disabled

20.5.5 SOFTWARE START ONE-SHOT MODE

In One-Shot mode the timer resets and the ON bit is cleared when the timer value matches the PRx period value. The ON bit must be set by software to start another timer cycle. Setting MODE<4:0> = 01000 selects One-Shot mode which is illustrated in Figure 20-8. In the example, ON is controlled by BSF and BCF instructions. In the first case, a BSF instruction sets ON and the counter runs to completion and clears ON. In the second case, a BSF instruction starts the cycle, BCF/BSF instructions turn the counter off and on during the cycle, and then it runs to completion.

When One-Shot mode is used in conjunction with the CCP PWM operation the PWM pulse drive starts concurrent with setting the ON bit. Clearing the ON bit while the PWM drive is active will extend the PWM drive. The PWM drive will terminate when the timer value matches the CCPRx pulse width value. The PWM drive will remain off until software sets the ON bit to start another cycle. If software clears the ON bit after the CCPRx match but before the PRx match then the PWM drive will be extended by the length of time the ON bit remains cleared. Another timing cycle can only be initiated by setting the ON bit after it has been cleared by a PRx period count match.

FIGURE 20-8: SOFTWARE START ONE-SHOT MODE TIMING DIAGRAM (MODE = 01000)



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.

20.5.10 LEVEL-TRIGGERED HARDWARE LIMIT ONE-SHOT MODES

The Level-Triggered Hardware Limit One-Shot modes hold the timer in Reset on an external Reset level and start counting when both the ON bit is set and the external signal is not at the Reset level. If one of either the external signal is not in Reset or the ON bit is set then the other signal being set/made active will start the timer. Reset levels are selected as follows:

- Low Reset level (MODE<4:0> = `10110`)
- High Reset level (MODE<4:0> = `10111`)

When the timer count matches the PRx period count, the timer is reset and the ON bit is cleared. When the ON bit is cleared by either a PRx match or by software control the timer will stay in Reset until both the ON bit is set and the external signal is not at the Reset level.

When Level-Triggered Hardware Limit One-Shot modes are used in conjunction with the CCP PWM operation the PWM drive goes active with either the external signal edge or the setting of the ON bit, whichever of the two starts the timer.

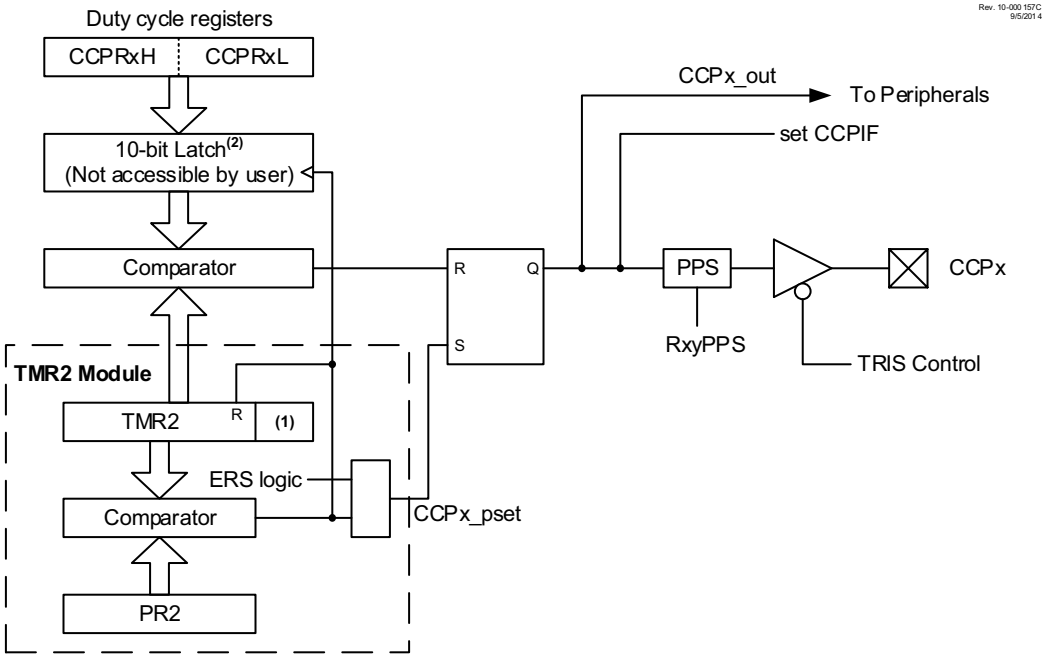
20.7 Register Definitions: Timer2/4/6 Control

Long bit name prefixes for the Timer2/4/6 peripherals are shown in Table 20-2. Refer to **Section 1.4.2.2 "Long Bit Names"** for more information.

TABLE 20-2:

Peripheral	Bit Name Prefix
Timer2	T2
Timer4	T4
Timer6	T6

FIGURE 21-4: SIMPLIFIED PWM BLOCK DIAGRAM



Notes: 1. 8-bit timer is concatenated with two bits generated by Fosc or two bits of the internal prescaler to create 10-bit time-base.
2. The alignment of the 10 bits from the CCPR register is determined by the FMT bit.

24.0 COMPLEMENTARY WAVEFORM GENERATOR (CWG) MODULE

The Complementary Waveform Generator (CWG) produces half-bridge, full-bridge, and steering of PWM waveforms. It is backwards compatible with previous CCP functions. The PIC18(L)F2x/4xK40 family has one instance of the CWG module.

The CWG has the following features:

- Six operating modes:
  - Synchronous Steering mode
  - Asynchronous Steering mode
  - Full-Bridge mode, Forward
  - Full-Bridge mode, Reverse
  - Half-Bridge mode
  - Push-Pull mode
- Output polarity control
- Output steering
- Independent 6-bit rising and falling event dead-band timers
  - Clocked dead band
  - Independent rising and falling dead-band enables
- Auto-shutdown control with:
  - Selectable shutdown sources
  - Auto-restart option
  - Auto-shutdown pin override control

24.1 Fundamental Operation

The CWG generates two output waveforms from the selected input source.

The off-to-on transition of each output can be delayed from the on-to-off transition of the other output, thereby, creating a time delay immediately where neither output is driven. This is referred to as dead time and is covered in **Section 24.6 “Dead-Band Control”**.

It may be necessary to guard against the possibility of circuit faults or a feedback event arriving too late or not at all. In this case, the active drive must be terminated before the Fault condition causes damage. This is referred to as auto-shutdown and is covered in **Section 24.10 “Auto-Shutdown”**.

24.2 Operating Modes

The CWG module can operate in six different modes, as specified by the MODE<2:0> bits of the CWG1CON0 register:

- Half-Bridge mode
- Push-Pull mode
- Asynchronous Steering mode
- Synchronous Steering mode
- Full-Bridge mode, Forward
- Full-Bridge mode, Reverse

All modes accept a single pulse data input, and provide up to four outputs as described in the following sections.

All modes include auto-shutdown control as described in **Section 24.10 “Auto-Shutdown”**

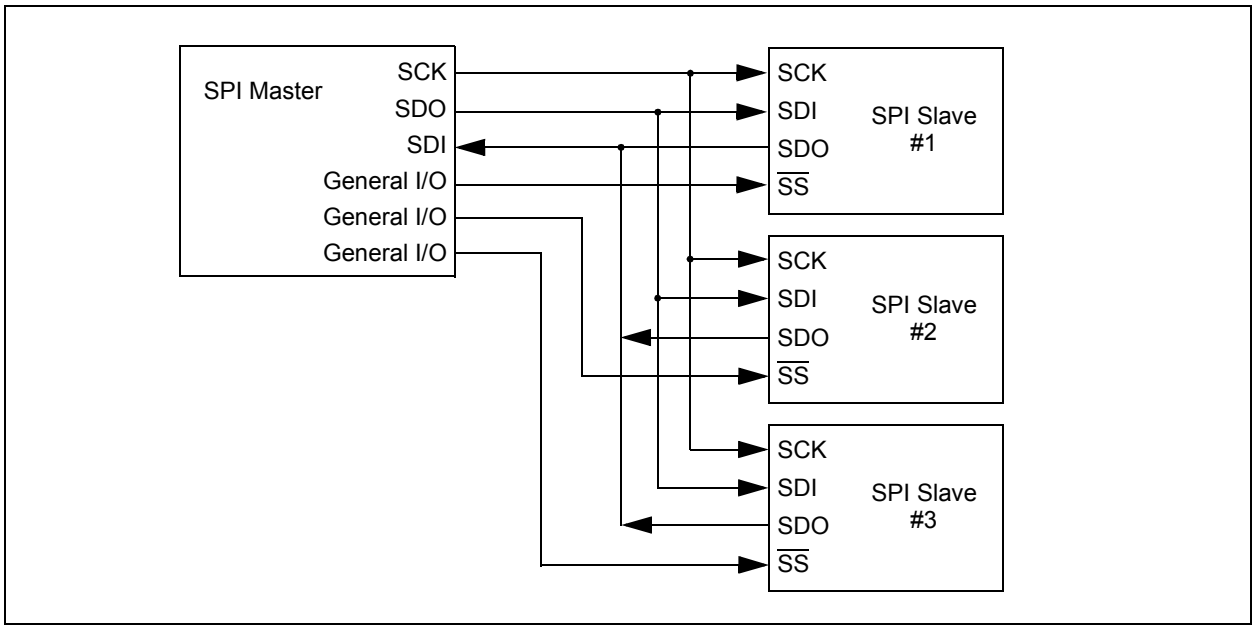
Note:	Except as noted for Full-bridge mode (**Section 24.2.3 “Full-Bridge Modes”**), mode changes should only be performed while EN = 0 (Register 24-1).

24.2.1 HALF-BRIDGE MODE

In Half-Bridge mode, two output signals are generated as true and inverted versions of the input as illustrated in Figure 24-2. A non-overlap (dead-band) time is inserted between the two outputs to prevent shoot through current in various power supply applications. Dead-band control is described in **Section 24.6 “Dead-Band Control”**. The output steering feature cannot be used in this mode. A basic block diagram of this mode is shown in Figure 24-1.

The unused outputs CWG1C and CWG1D drive similar signals, with polarity independently controlled by the POLC and POLD bits of the CWG1CON1 register, respectively.

FIGURE 26-2: SPI MASTER AND MULTIPLE SLAVE CONNECTION



26.3 SPI Mode Registers

The MSSP module has five registers for SPI mode operation. These are:

- MSSP STATUS register (SSPxSTAT)
- MSSP Control register 1 (SSPxCON1)
- MSSP Control register 3 (SSPxCON3)
- MSSP Data Buffer register (SSPxBUF)
- MSSP Address register (SSPxADD)
- MSSP Shift register (SSPSR)
  (Not directly accessible)

SSPxCON1 and SSPxSTAT are the control and STATUS registers in SPI mode operation. The SSPxCON1 register is readable and writable. The lower six bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write.

In one SPI master mode, SSPxADD can be loaded with a value used in the Baud Rate Generator. More information on the Baud Rate Generator is available in **Section 26.11 "Baud Rate Generator"**.

SSPSR is the shift register used for shifting data in and out. SSPxBUF provides indirect access to the SSPSR register. SSPxBUF is the buffer register to which data bytes are written, and from which data bytes are read.

In receive operations, SSPSR and SSPxBUF together create a buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set.

During transmission, the SSPxBUF is not buffered. A write to SSPxBUF will write to both SSPxBUF and SSPSR.

26.9.3 SLAVE TRANSMISSION

When the R/$\overline{W}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{W}$ bit of the SSPxSTAT register is set. The received address is loaded into the SSPxBUF register, and an $\overline{ACK}$ pulse is sent by the slave on the ninth bit.

Following the $\overline{ACK}$, slave hardware clears the CKP bit and the SCL pin is held low (see **Section 26.9.6 "Clock Stretching"** for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data.

The transmit data must be loaded into the SSPxBUF register which also loads the SSPSR register. Then the SCL pin should be released by setting the CKP bit of the SSPxCON1 register. The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time.

The $\overline{ACK}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. This $\overline{ACK}$ value is copied to the ACKSTAT bit of the SSPxCON2 register. If ACKSTAT is set (not $\overline{ACK}$), then the data transfer is complete. In this case, when the not $\overline{ACK}$ is latched by the slave, the slave goes idle and waits for another occurrence of the Start bit. If the SDA line was low ($\overline{ACK}$), the next transmit data must be loaded into the SSPxBUF register. Again, the SCL pin must be released by setting bit CKP.

An MSSP interrupt is generated for each data transfer byte. The SSPxIF bit must be cleared by software and the SSPxSTAT register is used to determine the status of the byte. The SSPxIF bit is set on the falling edge of the ninth clock pulse.

26.9.3.1 Slave Mode Bus Collision

A slave receives a Read request and begins shifting data out on the SDA line. If a bus collision is detected and the SBCDE bit of the SSPxCON3 register is set, the BCLxIF bit of the PIR register is set. Once a bus collision is detected, the slave goes idle and waits to be addressed again. User software can use the BCLxIF bit to handle a slave bus collision.

26.9.3.2 7-bit Transmission

A master device can transmit a read request to a slave, and then clock data out of the slave. The list below outlines what software for a slave will need to do to accomplish a standard transmission. Figure 26-18 can be used as a reference to this list.

1. Master sends a Start condition on SDA and SCL.
2. S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Matching address with R/$\overline{W}$ bit set is received by the Slave setting SSPxIF bit.
4. Slave hardware generates an $\overline{ACK}$ and sets SSPxIF.
5. SSPxIF bit is cleared by user.
6. Software reads the received address from SSPxBUF, clearing BF.
7. R/$\overline{W}$ is set so CKP was automatically cleared after the $\overline{ACK}$.
8. The slave software loads the transmit data into SSPxBUF.
9. CKP bit is set releasing SCL, allowing the master to clock the data out of the slave.
10. SSPxIF is set after the $\overline{ACK}$ response from the master is loaded into the ACKSTAT register.
11. SSPxIF bit is cleared.
12. The slave software checks the ACKSTAT bit to see if the master wants to clock out more data.

Note 1: If the master $\overline{ACK}$s the clock will be stretched.

2: ACKSTAT is the only bit updated on the rising edge of SCL (9th) rather than the falling.

13. Steps 9-13 are repeated for each transmitted byte.
14. If the master sends a not $\overline{ACK}$; the clock is not held, but SSPxIF is still set.
15. The master sends a Restart condition or a Stop.
16. The slave is no longer addressed.

TABLE 31-5: SUMMARY OF REGISTERS ASSOCIATED WITH ADC

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
PIE1	OSCFIE	CSWIE	—	—	—	—	ADTIE	ADIE	180
PIR1	OSCFIF	CSWIF	—	—	—	—	ADTIF	ADIF	172
ADCON0	ADON	ADCON	–	ADCS	–	ADFM	–	ADGO	448
ADCON1	ADPPOL	ADIPEN	ADGPOL	—	—	—	—	ADDSEN	449
ADCON2	ADPSIS	ADCRS<2:0>			ADACLR	ADMD<2:0>			450
ADCON3	–	ADCALC<2:0>			ADSOI	ADTMD<2:0>			451
ADACT	—	—	—	—	ADACT<4:0>				450
ADRESH	ADRESH<7:0>								458, 458
ADRESL	ADRESL<7:0>								458, 459
ADPREVH	ADPREV<15:8>								459
ADPREVL	ADPREV<7:0>								460
ADACCH	ADACC<15:8>								460
ADACCL	ADACC<7:0>								460
ADSTPTH	ADSTPT<15:8>								461
ADSTPT	ADSTPT<7:0>								461
ADERRL	ADERR<7:0>								462
ADLTHH	ADLTH<15:8>								462
ADLTHL	ADLTH<7:0>								462
ADUTHH	ADUTH<15:8>								463
ADUTHL	ADUTH<7:0>								463
ADSTAT	ADAOV	ADUTHR	ADLTHR	ADMATH	ADSTAT<3:0>				452
ADCLK	—	—	ADCS<5:0>						453
ADREF	—	—	—	ADNREF	—	—	ADPREF<1:0>		453
ADPCH	—	—	ADPCH<5:0>						454
ADPRE	ADPRE<7:0>								455
ADACQ	ADACQ<7:0>								455
ADCAP	—	—	—	ADCAP<4:0>					456
ADRPT	ADRPT<7:0>								456
ADCNT	ADCNT<7:0>								457
ADFLTRH	ADFLTR<15:8>								457
ADFLTRL	ADFLTR<7:0>								457
FVRCON	FVREN	FVRRDY	TSEN	TSRNG	CDAFVR<1:0>		ADFVR<1:0>		423
DAC1CON1	—	—	—	DAC1R<4:0>					429
OSCSTAT	EXTOR	HFOR	MFOR	LFOR	SOR	ADOR	—	PLLR	39

Legend: – = unimplemented read as '0'. Shaded cells are not used for the ADC module.

REGISTER 32-4: CMxPCH: COMPARATOR x NON-INVERTING CHANNEL SELECT REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	PCH<2:0>		
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 7-3 **Unimplemented:** Read as '0'

bit 2-0 **PCH<2:0>:** Comparator Non-Inverting Input Channel Select bits

111 = AVSS
110 = FVR_Buffer2
101 = DAC_Output
100 = CxPCH not connected
011 = CxPCH not connected
010 = CxPCH not connected
001 = CxIN1+
000 = CxIN0+

REGISTER 32-5: CMOUT: COMPARATOR OUTPUT REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	R-0/0	R-0/0
—	—	—	—	—	—	MC2OUT	MC1OUT
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 7-2 **Unimplemented:** Read as '0'

bit 1 **MC2OUT:** Mirror copy of C2OUT bit

bit 0 **MC1OUT:** Mirror copy of C1OUT bit

MULLW **Multiply literal with W**

Syntax: MULLW k

Operands: $0 \le k \le 255$

Operation: (W) x k → PRODH:PRODL

Status Affected: None

Encoding:

0000	1101	kkkk	kkkk

Description: An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged.
None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL

Example: `MULLW 0C4h`

Before Instruction
W = E2h
PRODH = ?
PRODL = ?
After Instruction
W = E2h
PRODH = ADh
PRODL = 08h

MULWF **Multiply W with f**

Syntax: MULWF f {,a}

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

Operation: (W) x (f) → PRODH:PRODL

Status Affected: None

Encoding:

0000	001a	ffff	ffff

Description: An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged.
None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 35.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write registers PRODH: PRODL

Example: `MULWF REG, 1`

Before Instruction
W = C4h
REG = B5h
PRODH = ?
PRODL = ?
After Instruction
W = C4h
REG = B5h
PRODH = 8Ah
PRODL = 94h

SLEEP **Enter Sleep mode**

Syntax: SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT postscaler,
1 → $\overline{TO}$,
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0000	0011

Description: The Power-down Status bit ($\overline{PD}$) is cleared. The Time-out Status bit ($\overline{TO}$) is set. Watchdog Timer and its postscaler are cleared.
The processor is put into Sleep mode with the oscillator stopped.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	Go to Sleep

Example: SLEEP

Before Instruction
$\overline{TO}$ = ?
$\overline{PD}$ = ?
After Instruction
$\overline{TO}$ = 1 †
$\overline{PD}$ = 0

† If WDT causes wake-up, this bit is cleared.

SUBFWB **Subtract f from W with borrow**

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

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (W) – (f) – ($\overline{C}$) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

0101	01da	ffff	ffff

Description: Subtract register ‘f’ and CARRY flag (borrow) from W (2’s complement method). If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored 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 1:
```
SUBFWB   REG, 1, 0
```
Before Instruction
REG = 3
W = 2
C = 1
After Instruction
REG = FF
W = 2
C = 0
Z = 0
N = 1 ; result is negative

Example 2:
```
SUBFWB   REG, 0, 0
```
Before Instruction
REG = 2
W = 5
C = 1
After Instruction
REG = 2
W = 3
C = 1
Z = 0
N = 0 ; result is positive

Example 3:
```
SUBFWB   REG, 1, 0
```
Before Instruction
REG = 1
W = 2
C = 0
After Instruction
REG = 0
W = 2
C = 1
Z = 1 ; result is zero
N = 0

SUBLW **Subtract W from literal**

Syntax: SUBLW k

Operands: $0 \le k \le 255$

Operation: $k - (W) \rightarrow W$

Status Affected: N, OV, C, DC, Z

Encoding:

0000	1000	kkkk	kkkk

Description: W is subtracted from the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1: `SUBLW 02h`

Before Instruction
W = 01h
C = ?
After Instruction
W = 01h
C = 1 ; result is positive
Z = 0
N = 0

Example 2: `SUBLW 02h`

Before Instruction
W = 02h
C = ?
After Instruction
W = 00h
C = 1 ; result is zero
Z = 1
N = 0

Example 3: `SUBLW 02h`

Before Instruction
W = 03h
C = ?
After Instruction
W = FFh ; (2's complement)
C = 0 ; result is negative
Z = 0
N = 1

SUBWF **Subtract W from f**

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

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

Operation: $(f) - (W) \rightarrow$ dest

Status Affected: N, OV, C, DC, Z

Encoding:

0101	11da	ffff	ffff

Description: Subtract W from register 'f' (2's complement method). If 'd' is '0', the result is stored 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 $f \le 95$ (5Fh). See **Section 35.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1: `SUBWF REG, 1, 0`

Before Instruction
REG = 3
W = 2
C = ?
After Instruction
REG = 1
W = 2
C = 1 ; result is positive
Z = 0
N = 0

Example 2: `SUBWF REG, 0, 0`

Before Instruction
REG = 2
W = 2
C = ?
After Instruction
REG = 2
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: `SUBWF REG, 1, 0`

Before Instruction
REG = 1
W = 2
C = ?
After Instruction
REG = FFh ;(2's complement)
W = 2
C = 0 ; result is negative
Z = 0
N = 1