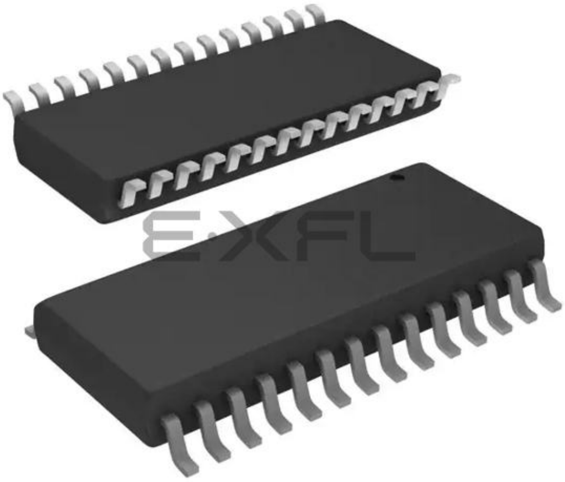
Microchip Technology - PIC18F26K42T-I/SO Datasheet



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, DMA, HLVD, 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	4K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 24x12b; 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/pic18f26k42t-i-so

4.5.5 STATUS REGISTER

The STATUS register, shown in Register 4-2, contains the arithmetic status of the ALU. As with any other SFR, it can be the operand for any instruction.

If the STATUS register is the destination for an instruction that affects the Z, DC, C, OV or N bits, the results of the instruction are not written; instead, the STATUS register is updated according to the instruction performed. Therefore, the result of an instruction with the STATUS register as its destination may be different than intended. As an example, `CLRF STATUS` will set the Z bit and leave the remaining Status bits unchanged (‘`0uuu u1uu`’).

It is recommended that only `BCF`, `BSF`, `SWAPF`, `MOVFF`, `MOVWF` and `MOVFFL` instructions are used to alter the STATUS register, because these instructions do not affect the Z, C, DC, OV or N bits in the STATUS register.

For other instructions that do not affect Status bits, see the instruction set summaries in **Section 41.2 “Extended Instruction Set”** and Table 41-3.

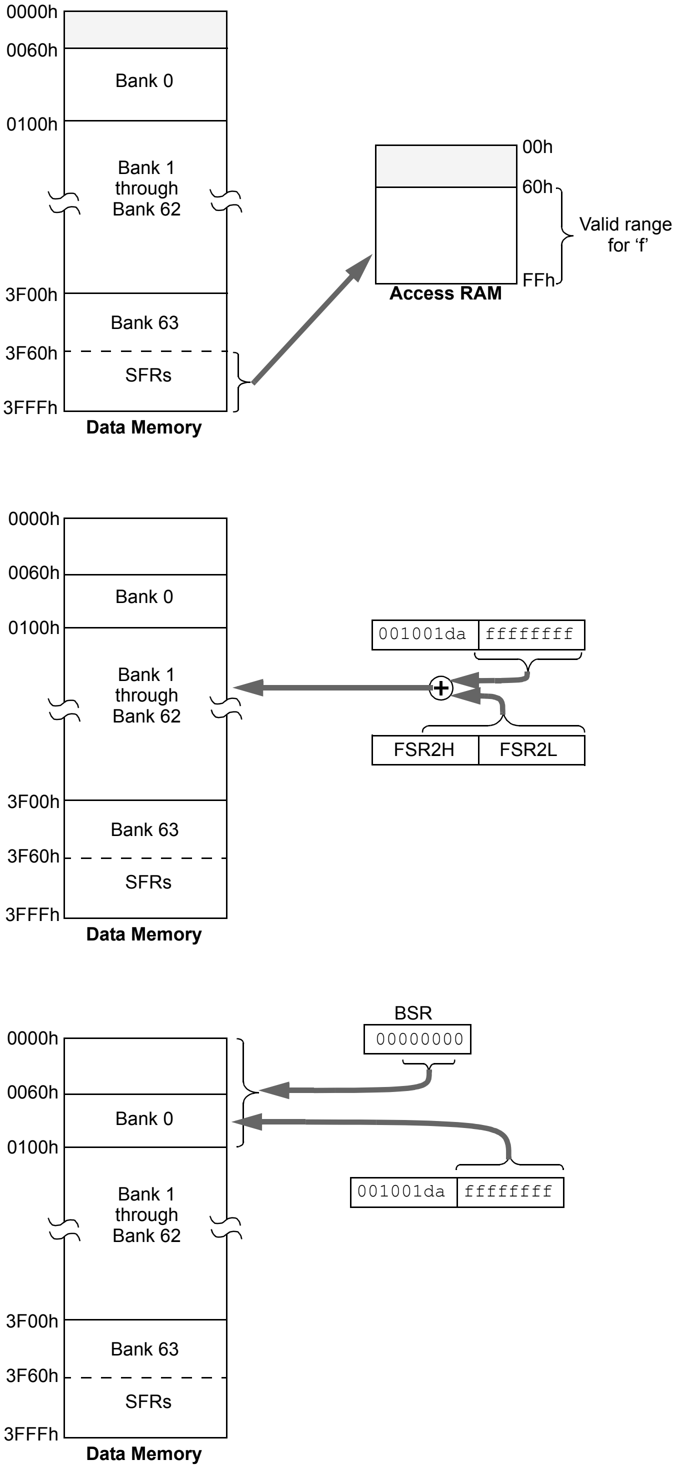
> **Note:** The C and DC bits operate as the borrow and digit borrow bits, respectively, in subtraction.

4.5.6 CALL SHADOW REGISTER

When `CALL`, `CALLW`, `RCALL` instructions are used, the WREG, BSR and STATUS are automatically saved in hardware and can be accessed using the WREG_C-SHAD, BSR_CSHAD and STATUS_CSHAD registers. The contents of these registers should be handled correctly to avoid erroneous code execution.

FIGURE 4-7: 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 3F60h to 3FFFh (Bank 63) 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 63 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.

REGISTER 6-3: PCON1: POWER CONTROL REGISTER 1

U-0	U-0	U-0	U-0	U-0	U-0	R/W/HC-1/u	U-0
—	—	—	—	—	—	$\overline{\text{MEMV}}$	—
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	-m/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	q = Value depends on condition

bit 7-2 **Unimplemented:** Read as ‘0’

bit 1 **$\overline{\text{MEMV}}$:** Memory Violation Flag bit

1 = No memory violation Reset occurred or set to ‘1’ by firmware
0 = A memory violation Reset occurred (set to ‘0’ in hardware when a Memory Violation occurs)

bit 0 **Unimplemented:** Read as ‘0’

TABLE 6-4: SUMMARY OF REGISTERS ASSOCIATED WITH RESETS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
BORCON	SBOREN	—	—	—	—	—	—	BORRDY	85
PCON0	STKOVF	STKUNF	$\overline{\text{WDTWV}}$	$\overline{\text{RWDT}}$	$\overline{\text{RMCLR}}$	$\overline{\text{RI}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	90
PCON1	—	—	—	—	—	—	$\overline{\text{MEMV}}$	—	91

Legend: — = unimplemented location, read as ‘0’. Shaded cells are not used by Resets.

REGISTER 7-2: OSCCON2: OSCILLATOR CONTROL REGISTER 2

U-0	R-f/f	R-f/f	R-f/f	R-f/f	R-f/f	R-f/f	R-f/f
—	COSC<2:0>			CDIV<3:0>			
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 **Unimplemented:** Read as ‘0’

bit 6-4 **COSC<2:0>:** Current Oscillator Source Select bits (read-only)[1]
Indicates the current source oscillator and PLL combination per Table 7-1.

bit 3-0 **CDIV<3:0>:** Current Divider Select bits (read-only)[1]
Indicates the current postscaler division ratio per Table 7-1.

Note 1: The POR value is the value present when user code execution begins.

REGISTER 7-3: OSCCON3: OSCILLATOR CONTROL REGISTER 3

R/W/HC-0/0	R/W-0/0	U-0	R-0/0	R-0/0	U-0	U-0	U-0
CSWHOLD	SOSCPWR	—	ORDY	NOSCR	—	—	—
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 **CSWHOLD:** Clock Switch Hold bit
1 = Clock switch will hold (with interrupt) when the oscillator selected by NOSC is ready
0 = Clock switch may proceed when the oscillator selected by NOSC is ready; NOSCR becomes ‘1’, the switch will occur

bit 6 **SOSCPWR:** Secondary Oscillator Power Mode Select bit
1 = Secondary oscillator operating in High-Power mode
0 = Secondary oscillator operating in Low-Power mode

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **ORDY:** Oscillator Ready bit (read-only)
1 = OSCCON1 = OSCCON2; the current system clock is the clock specified by NOSC
0 = A clock switch is in progress

bit 3 **NOSCR:** New Oscillator is Ready bit (read-only)[1]
1 = A clock switch is in progress and the oscillator selected by NOSC indicates a “ready” condition
0 = A clock switch is not in progress, or the NOSC-selected oscillator is not yet ready

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

Note 1: If CSWHOLD = 0, the user may not see this bit set because, when the oscillator becomes ready there may be a delay of one instruction clock before this bit is set. The clock switch occurs in the next instruction cycle and this bit is cleared.

REGISTER 9-7: PIR4: PERIPHERAL INTERRUPT REGISTER 4[1]

R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	U-0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0
CLC1IF	CWG1IF	NCO1IF	—	CCP1IF	TMR2IF	TMR1GIF	TMR1IF
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	HS = Bit is set in hardware

bit 7 **CLC1IF:** CLC1 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 6 **CWG1IF:** CWG1 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 5 **NCO1IF:** NCO1 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 4 **Unimplemented:** Read as ‘0’

bit 3 **CCP1IF:** CCP1 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 2 **TMR2IF:** TMR2 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 1 **TMR1GIF:** TMR1 Gate Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 0 **TMR1IF:** TMR1 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 9-32: IPR7: PERIPHERAL INTERRUPT PRIORITY REGISTER 7

U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1	U-0	R/W-1/1	R/W-1/1
—	—	INT2IP	CLC2IP	CWG2IP	—	CCP2IP	TMR4IP
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 **INT2IP:** External Interrupt 2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **CLC2IP:** CLC2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **CWG2IP:** CWG2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **Unimplemented:** Read as ‘0’

bit 1 **CCP2IP:** CRC Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR4IP:** TMR4 Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-33: IPR8: PERIPHERAL INTERRUPT PRIORITY REGISTER 8

R/W-1/1	R/W-1/1	U-0	U-0	U-0	U-0	U-0	U-0
TMR5GIP	TMR5IP	—	—	—	—	—	—
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 **TMR5GIP:** TMR5 Gate Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **TMR5IP:** TMR5 Interrupt Priority bit
1 = High priority
0 = Low priority

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

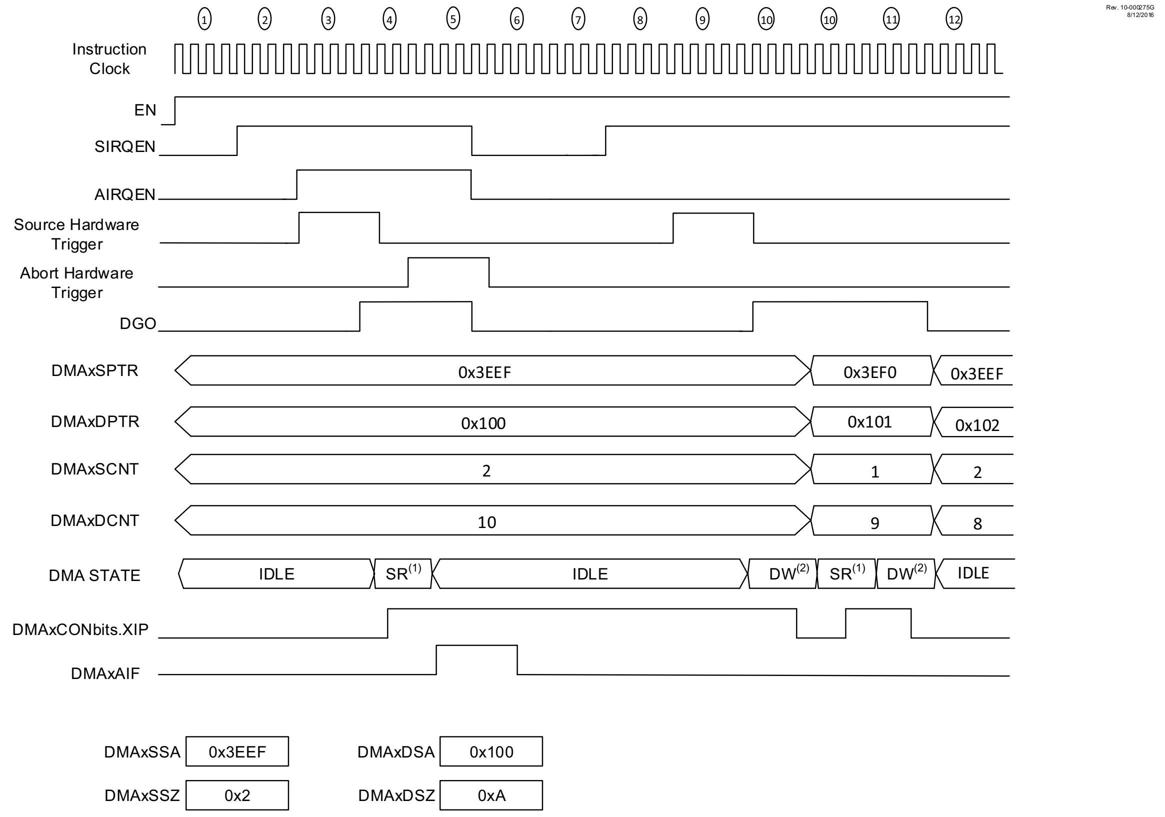
15.9.7 ABORT TRIGGER, MESSAGE IN PROGRESS

When an abort interrupt request is received in a DMA transaction, the DMA will perform a soft-stop by clearing the DGO (i.e., if the DMA was reading the source register, it will complete the read operation and then clear the DGO bit).

The SIREQEN bit is cleared to prevent any overrun and the AIRQEN bit is cleared to prevent any false aborts.

When the DGO bit is set again the DMA will resume operation from where it left off after the soft-stop.

FIGURE 15-11: ABORT DURING MESSAGE TRANSFER



Note 1: SR - Source Read

2: DW - Destination Write

The following table contains some of the cases in which the DMA module can be configured to.

REGISTER 23-3: CCPxCAP: CAPTURE INPUT SELECTION MULTIPLEXER REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/x	R/W-0/x	R/W-0/x
—	—	—	—	—	CTS<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 **CTS<2:0>:** Capture Trigger Input Selection bits

CTS<1:0>	**Connection**			
	CCP1	**CCP2**	**CCP3**	**CCP4**
111	CLC4_out			
110	CLC3_out			
101	CLC2_out			
100	CLC1_out			
011	IOC_Interrupt			
010	CMP2_output			
001	CMP1_output			
000	Pin selected by CCP1PPS	Pin selected by CCP2PPS	Pin selected by CCP3PPS	Pin selected by CCP4PPS

REGISTER 23-4: CCPRxL: CCPx REGISTER LOW BYTE

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
RL<7:0>							
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 7-0 MODE = Capture Mode:

RL<7:0>: LSB of captured TMR1 value

MODE = Compare Mode:

RL<7:0>: LSB compared to TMR1 value

MODE = PWM Mode && FMT = 0:

RL<7:0>: CCPW<7:0> – Pulse-Width LS 8 bits

MODE = PWM Mode && FMT = 1:

RL<7:6>: CCPW<1:0> – Pulse-Width LS 2 bits

RL<5:0>: Not used

REGISTER 24-3: PWMxDCH: PWM DUTY CYCLE HIGH BITS

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
DC<9:2>							
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 **DC<9:2>:** PWM Duty Cycle Most Significant bits
These bits are the MSbs of the PWM duty cycle. The two LSbs are found in PWMxDCL Register.

REGISTER 24-4: PWMxDCL: PWM DUTY CYCLE LOW BITS

R/W-x/u	R/W-x/u	U-0	U-0	U-0	U-0	U-0	U-0
DC<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 7-6 **DC<1:0>:** PWM Duty Cycle Least Significant bits
These bits are the LSbs of the PWM duty cycle. The MSbs are found in PWMxDCH Register.

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

TABLE 24-3: SUMMARY OF REGISTERS ASSOCIATED WITH PWM

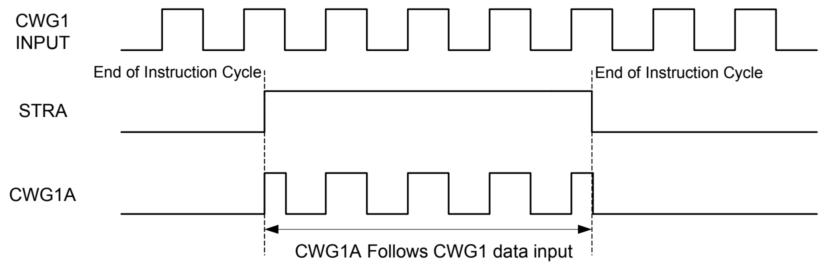
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
PWMxCON	EN	—	OUT	POL	—	—	—	—	358
PWMxDCH	DC<9:2>								360
PWMxDCL	DC<1:0>		—	—	—	—	—	—	360
CCPTMRS1	P8TSEL<1:0>		P7TSEL<1:0>		P6TSEL<1:0>		P5TSEL<1:0>		359

Legend: – = Unimplemented locations, read as ‘0’, u = unchanged, x = unknown. Shaded cells are not used by the PWM.

26.2.4.2 Asynchronous Steering Mode

In Asynchronous mode (MODE<2:0> bits = 000, Register 26-1), steering takes effect at the end of the instruction cycle that writes to STR. In Asynchronous Steering mode, the output signal may be an incomplete waveform (Figure 26-10). This operation may be useful when the user firmware needs to immediately remove a signal from the output pin.

FIGURE 26-10: EXAMPLE OF ASYNCHRONOUS STEERING (MODE<2:0>= 000)



26.2.4.3 Start-up Considerations

The application hardware must use the proper external pull-up and/or pull-down resistors on the CWG output pins. This is required because all I/O pins are forced to high-impedance at Reset.

The POLy bits (Register 26-2) allow the user to choose whether the output signals are active-high or active-low.

REGISTER 26-4: CWGxISM: CWGx INPUT SELECTION REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	ISM<4:0>				
bit 7							bit 0

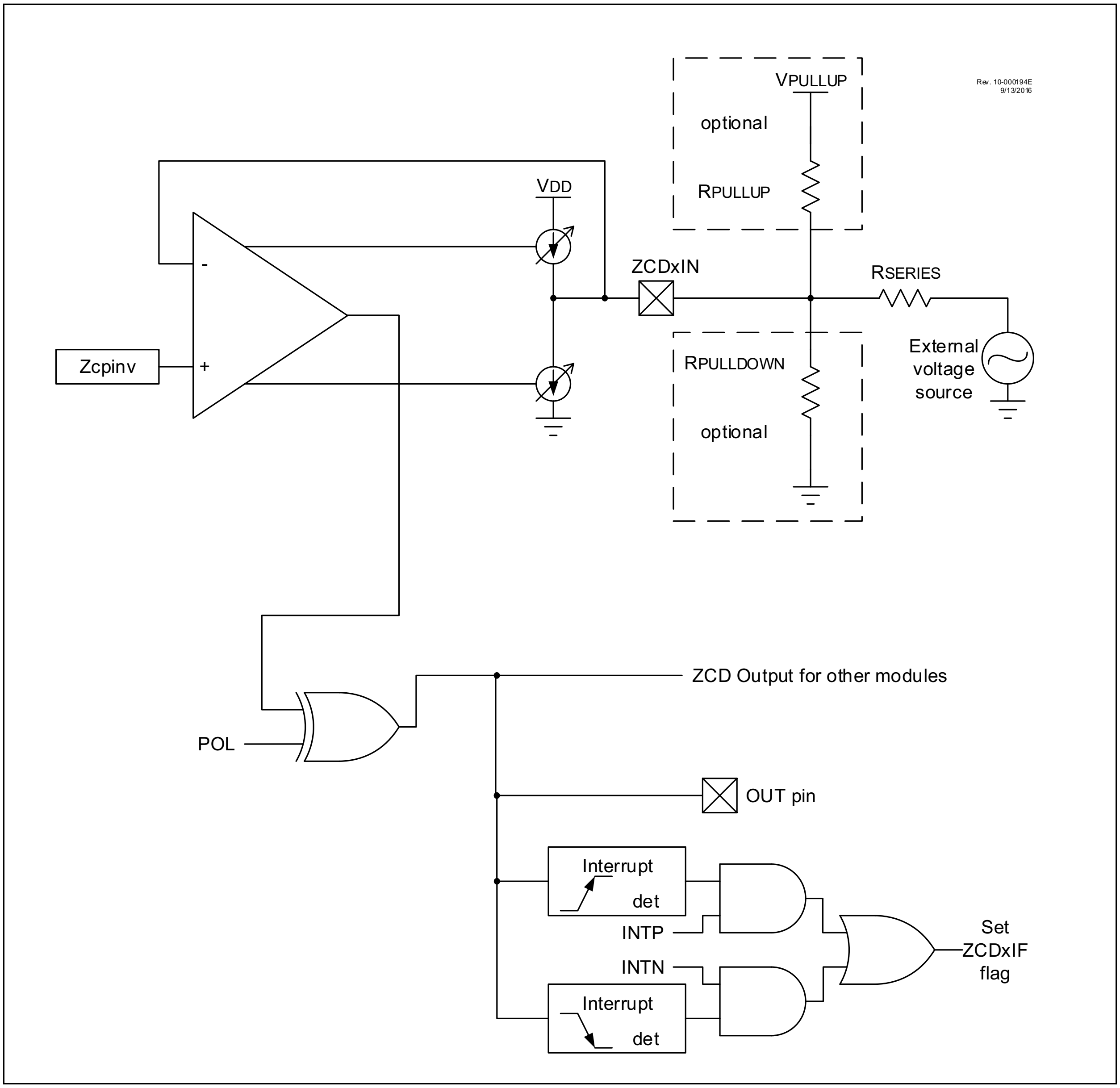
Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	q = Value depends on condition

bit 7-5 **Unimplemented** Read as ‘0’

bit 4-0 **ISM<4:0>:** CWG Data Input Selection Multiplexer Select bits

ISM<4:0>	**CWG1**	**CWG2**	**CWG3**
	Input Selection	**Input Selection**	**Input Selection**
11111-10011	Reserved	Reserved	Reserved
10010	CLC4_out	CLC4_out	CLC4_out
10001	CLC3_out	CLC3_out	CLC3_out
10000	CLC2_out	CLC2_out	CLC2_out
01111	CLC1_out	CLC1_out	CLC1_out
01110	DSM_out	DSM_out	DSM_out
01101	CMP2OUT	CMP2OUT	CMP2OUT
01100	CMP1OUT	CMP1OUT	CMP1OUT
01011	NCO1OUT	NCO1OUT	NCO1OUT
01010-01001	Reserved	Reserved	Reserved
01000	PWM8OUT	PWM8OUT	PWM8OUT
00111	PWM7OUT	PWM7OUT	PWM7OUT
00110	PWM6OUT	PWM6OUT	PWM6OUT
00101	PWM5OUT	PWM5OUT	PWM5OUT
00100	CCP4_out	CCP4_out	CCP4_out
00011	CCP3_out	CCP3_out	CCP3_out
00010	CCP2_out	CCP2_out	CCP2_out
00001	CCP1_out	CCP1_out	CCP1_out
00000	Pin selected by CWG1PPS	Pin selected by CWG2PPS	Pin selected by CWG3PPS

FIGURE 29-2: SIMPLIFIED ZCD BLOCK DIAGRAM



29.2 ZCD Logic Output

The ZCD module includes a Status bit, which can be read to determine whether the current source or sink is active. The OUT bit of the ZCDCON register is set when the current sink is active, and cleared when the current source is active. The OUT bit is affected by the polarity bit, even if the module is disabled.

The OUT signal can also be used as input to other modules. This is controlled by the registers of the corresponding module. OUT can be used as follows:

- Gate source for TMR1/3/5
- Clock source for TMR2/4/6
- Reset source for TMR2/4/6

29.3 ZCD Logic Polarity

The POL bit of the ZCDCON register inverts the OUT bit relative to the current source and sink output. When the POL bit is set, a OUT high indicates that the current source is active, and a low output indicates that the current sink is active.

The POL bit affects the ZCD interrupts.

REGISTER 30-3: MD1CARH: MODULATION HIGH CARRIER CONTROL REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	CH<4:0>(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 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **CH<4:0>:** Modulator Carrier High Selection bits(1)
See Table 30-1 for signal list

Note 1: Unused selections provide an input value.

REGISTER 30-4: MD1CARL: MODULATION LOW CARRIER CONTROL REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	CL<4:0>(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 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **CL<4:0>:** Modulator Carrier Low Input Selection bits(1)
See Table 30-1 for signal list

Note 1: Unused selections provide a zero as the input value.

31.16 Clock Accuracy with Asynchronous Operation

The factory calibrates the internal oscillator block output (INTOSC). However, the INTOSC frequency may drift as VDD or temperature changes, and this directly affects the asynchronous baud rate. Two methods may be used to adjust the baud rate clock, but both require a reference clock source of some kind.

The first (preferred) method uses the OSCTUNE register to adjust the INTOSC output. Adjusting the value of the OSCTUNE register allows for fine resolution changes to the system clock source. See **Section 7.2.2.3 “Internal Oscillator Frequency Adjustment”** for more information.

The other method adjusts the value of the Baud Rate Generator. This can be done automatically with the Auto-Baud Detect feature (see **Section 31.17.1 “Auto-Baud Detect”**). There may not be fine enough resolution when adjusting the Baud Rate Generator to compensate for a gradual change of the peripheral clock frequency.

31.17 UART Baud Rate Generator (BRG)

The Baud Rate Generator (BRG) is a 16-bit timer that is dedicated to the support of the UART operation.

The UxBRGH, UxBRGL register pair determines the period of the free running baud rate timer. The multiplier of the baud rate period is determined by the BRGS bit in the UxCON0 register.

Table 31-1 contains the formulas for determining the baud rate. Example 31-1 provides a sample calculation for determining the baud rate and baud rate error.

The high baud rate range (BRGS = `1`) is intended to extend the baud rate range up to a faster rate when the desired baud rate is not possible otherwise. Using the normal baud rate range (BRGS = `0`) is recommended when the desired baud rate is achievable with either range.

Writing a new value to the UxBRGH, UxBRGL register pair causes the BRG timer to be reset (or cleared). This ensures that the BRG does not wait for a timer overflow before outputting the new baud rate.

If the system clock is changed during an active receive operation, a receive error or data loss may result. To avoid this problem, check the status of the RXIDL bit to make sure that the receive operation is idle before changing the system clock.

EXAMPLE 31-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, BRGS = `0`:

$$\textit{Desired Baud Rate} = \frac{F_{OSC}}{16([UxBRG] + 1)}$$

$$X = \frac{\frac{F_{OSC}}{\textit{Desired Baud Rate}}}{16} - 1$$

$$= \frac{\frac{16000000}{9600}}{16} - 1$$

$$= [103.17] = 103$$

$$\textit{Calculated Baud Rate} = \frac{16000000}{16(103 + 1)}$$

$$= 9615$$

$$\textit{Error} = \frac{\textit{Calc. Baud Rate} - \textit{Desired Baud Rate}}{\textit{Desired Baud Rate}}$$

$$= \frac{(9615 - 9600)}{9600} = 0.16\%$$

TABLE 31-1: BAUD RATE FORMULAS

BRGS	BRG/UART Mode	Baud Rate Formula
1	High Rate	FOSC/[4 (n+1)]
0	Normal Rate	FOSC/[16(n+1)]

Legend: n = value of UxBRGH, UxBRGL register pair.

32.4 Transfer Counter

In all master modes, the transfer counter can be used to determine how many data transfers the SPI will send/receive. The transfer counter is comprised of the SPIxTCTH/L set of registers, and is also partially controlled by the SPIxTWIDTH register. The Transfer Counter has two primary modes, determined by the BMODE bit of the SPIxCON0 register. Each mode uses the SPIxTCTH/L and SPIxTWIDTH registers to determine the number and size of the transfers. In both modes, when the transfer counter reaches zero, the TCZIF interrupt flag is set.

Note:	When BMODE=1 in all master modes (and at all times in slave modes), the Transfer Counter will still decrement as transfers occur and can be used to count the number of messages sent/received, as well as to control SS(out) and to trigger TCZIF. Also when BMODE = 1, the SPIxTWIDTH register can be used in Master and Slave modes to determine the size of messages sent and received by the SPI, even if the Transfer Counter is not being actively used to control the number of messages being sent/received by the SPI module.

32.4.1 TOTAL BIT COUNT MODE (BMODE = 0)

In this mode, SPIxTCTH/L and SPIxTWIDTH are concatenated to determine the total number of bits to be transferred. These bits will be loaded from/into the transmit/receive FIFOs in 8-bit increments and the transfer counter will be decremented by eight until the total number of remaining bits is less than eight. If there are any remaining bits (SPIxTWIDTH ≠ 0), the transmit FIFO will send out one final message with any extra bits greater than the remainder ignored. The SPIxTWIDTH is the remaining bit count but the value does not change as it does for the SPIxTCT value. Similarly, the receiver will load a final byte into the receiver FIFO, and pad the extra bits with zeros. The LSBF bit of SPIxCON0 determines whether the Most Significant or Least Significant bits of this final byte are ignored/padded. For example, when LSBF = 0 and the final transfer contains only two bits then if the last byte sent was 5Fh then the RXB of the receiver will contain 40h which are the two MSbits of the final byte padded with zeros in the LSbits.

In this mode, the SPI master will only transmit messages when the SPIxTCT value is greater than zero, regardless of TXR and RXR settings. In Master Transmit mode, the transfer starts with the data write to the SPIxTXB register or the count value written to the SPIxTCTL register, which ever occurs last. In Master Receive-only mode, the transfer clocks start when the SPIxTCTL value is written. Transfer clocks are suspended when the receive FIFO is full and resume as the FIFO is read.

32.4.2 VARIABLE TRANSFER SIZE MODE (BMODE = 1)

In this mode, SPIxTWIDTH specifies the width of every individual piece of the data transfer in bits. SPIxTCTH/SPIxTCTL specifies the number of transfers of this bit length. If SPIxTWIDTH = 0, each piece is a full byte of data. If SPIxTWIDTH ≠ 0, then only the specified number of bits from the transmit FIFO are shifted out, with the unused bits ignored. Received data is padded with zeros in the unused bit areas when transfered into the receive FIFO. The LSBF bit of SPIxCON0 determines whether the Most Significant or Least Significant bits of the transfers are ignored/padded. In this mode, the transfer counter being zero only stops messages from being sent/received when in “Receive only” mode.

Note:	With BMODE = 1, it is possible for the transfer counter (SPIxTCTH/L) to decrement below zero, although when in “Receive only” Master mode, transfer clocks will cease when the transfer counter reaches zero.

REGISTER 36-27: ADSTPTH: ADC THRESHOLD SETPOINT REGISTER HIGH

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
STPT<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 **STPT<15:8>**: ADC Threshold Setpoint MSB. Upper byte of ADC threshold setpoint, depending on ADCALC, may be used to determine ERR, see Register 36-29 for more details.

REGISTER 36-28: ADSTPTL: ADC THRESHOLD SETPOINT REGISTER LOW

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
STPT<7:0>							
bit 7							bit 0

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

bit 7-0 **STPT<7:0>**: ADC Threshold Setpoint LSB. Lower byte of ADC threshold setpoint, depending on ADCALC, may be used to determine ERR, see Register 36-30 for more details.

RCALL	Relative Call
Syntax:	RCALL n
Operands:	$-1024 \leq n \leq 1023$
Operation:	$(PC) + 2 \rightarrow TOS$, $(PC) + 2 + 2n \rightarrow PC$
Status Affected:	None
Encoding:	`1101` `1nnn` `nnnn` `nnnn`
Description:	Subroutine call with a jump up to 1K from the current location. First, return address (PC + 2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a 2-cycle instruction.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n' PUSH PC to stack	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: `HERE     RCALL Jump`

Before Instruction
PC = Address `(HERE)`

After Instruction
PC = Address `(Jump)`
TOS = Address `(HERE + 2)`

RESET	Reset
Syntax:	RESET
Operands:	None
Operation:	Reset all registers and flags that are affected by a $\overline{\text{MCLR}}$ Reset.
Status Affected:	All
Encoding:	`0000` `0000` `1111` `1111`
Description:	This instruction provides a way to execute a $\overline{\text{MCLR}}$ Reset by software.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start Reset	No operation	No operation

Example: `RESET`

After Instruction
Registers = Reset Value
Flags* = Reset Value

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) → 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 41.2.3 “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
```

SUBWFB Subtract W from f with Borrow

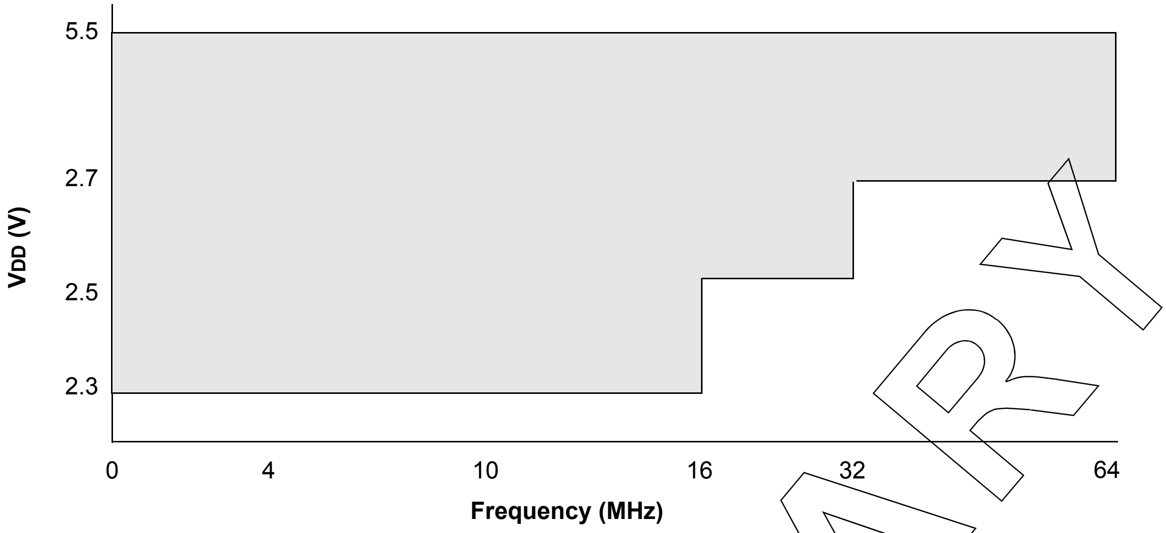
Syntax:	SUBWFB f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) – (W) – ($\overline{C}$) → dest
Status Affected:	N, OV, C, DC, Z
Encoding:	`0101` | `10da` | `ffff` | `ffff`
Description:	Subtract W and the CARRY flag (borrow) from register ‘f’ (2’s complement method). 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 41.2.3 “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

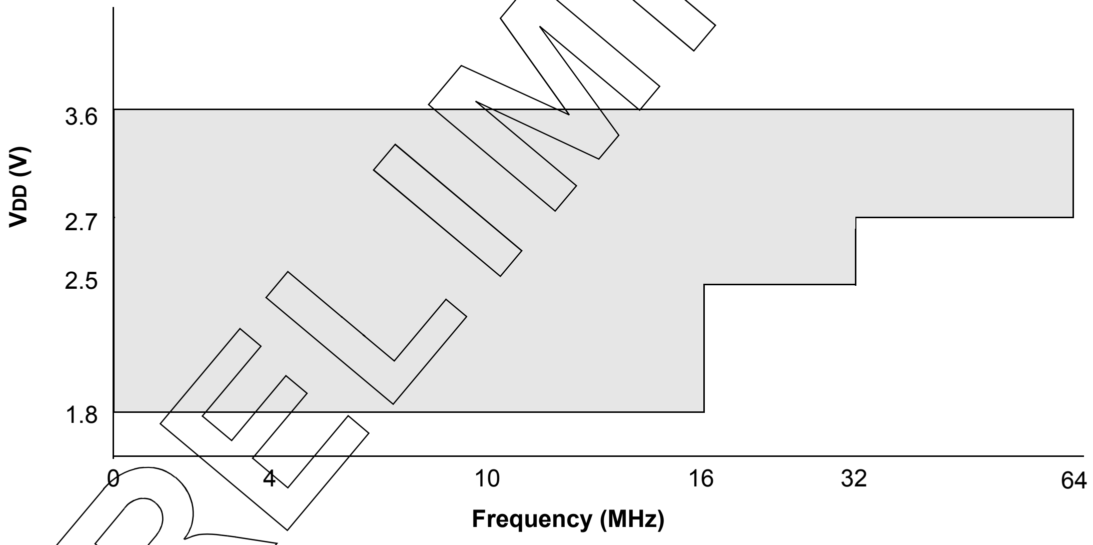
```
Example 1:        SUBWFB  REG, 1, 0
    Before Instruction
        REG    =   19h    (0001 1001)
        W      =   0Dh    (0000 1101)
        C      =   1
    After Instruction
        REG    =   0Ch    (0000 1100)
        W      =   0Dh    (0000 1101)
        C      =   1
        Z      =   0
        N      =   0      ; result is positive
Example 2:        SUBWFB  REG, 0, 0
    Before Instruction
        REG    =   1Bh    (0001 1011)
        W      =   1Ah    (0001 1010)
        C      =   0
    After Instruction
        REG    =   1Bh    (0001 1011)
        W      =   00h
        C      =   1
        Z      =   1      ; result is zero
        N      =   0
Example 3:        SUBWFB  REG, 1, 0
    Before Instruction
        REG    =   03h    (0000 0011)
        W      =   0Eh    (0000 1110)
        C      =   1
    After Instruction
        REG    =   F5h    (1111 0101)
                          ; [2’s comp]
        W      =   0Eh    (0000 1110)
        C      =   0
        Z      =   0
        N      =   1      ; result is negative
```

FIGURE 44-1: VOLTAGE FREQUENCY GRAPH, -40°C ≤ TA ≤ +125°C, PIC18F26/45/46/55/56K42 ONLY



Note 1: The shaded region indicates the permissible combinations of voltage and frequency.
2: Refer to Table 44-9 for each Oscillator mode's supported frequencies.

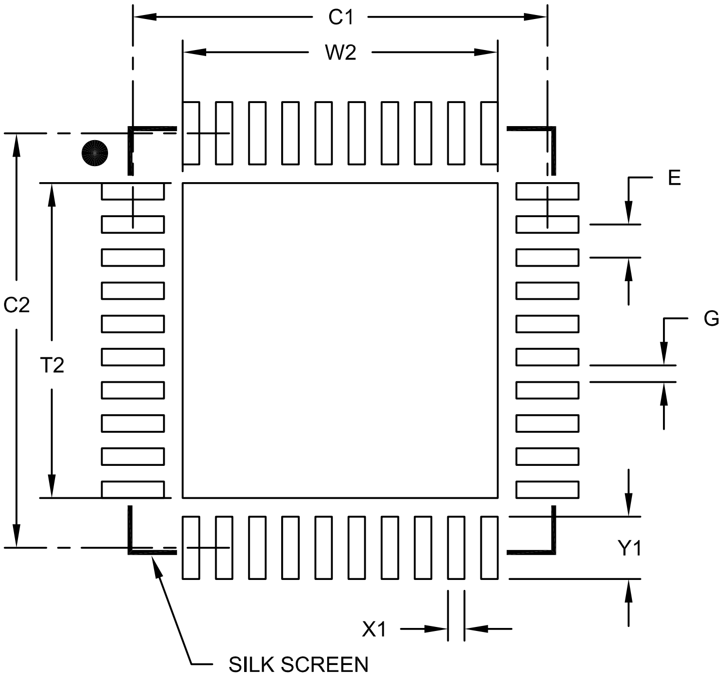
FIGURE 44-2: VOLTAGE FREQUENCY GRAPH, -40°C ≤ TA ≤ +125°C, PIC18(L)F26/27/45/46/47/55/56/57K42 ONLY



Note 1: The shaded region indicates the permissible combinations of voltage and frequency.
2: Refer to Table 44-9 for each Oscillator mode's supported frequencies.

40-Lead Plastic Ultra Thin Quad Flat, No Lead Package (MV) - 5x5 mm Body [UQFN]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

		Units	MILLIMETERS	
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.40 BSC	
Optional Center Pad Width	W2			3.80
Optional Center Pad Length	T2			3.80
Contact Pad Spacing	C1		5.00	
Contact Pad Spacing	C2		5.00	
Contact Pad Width (X40)	X1			0.20
Contact Pad Length (X40)	Y1			0.75
Distance Between Pads	G	0.20		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2156B