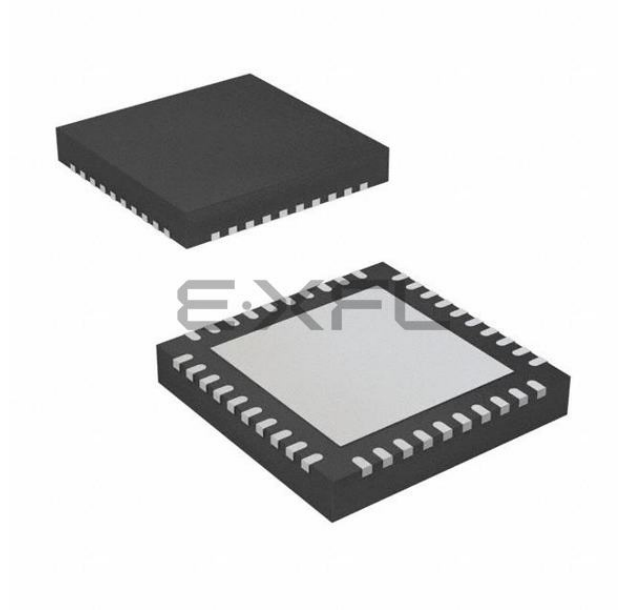


**Welcome to E-XFL.COM**

## 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, DMA, HLVD, POR, PWM, WDT |
| Number of I/O | 36 |
| Program Memory Size | 128KB (64K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | 1K x 8 |
| RAM Size | 8K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.3V ~ 5.5V |
| Data Converters | A/D 35x12b; D/A 1x5b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 125°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 40-UFQFN Exposed Pad |
| Supplier Device Package | 40-UQFN (5x5) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic18f47k42-e-mv |

**TABLE 1-1: DEVICE FEATURES (CONTINUED)**

| Features | PIC18(L)F26K42 | PIC18(L)F27K42 | PIC18(L)F45K42 | PIC18(L)F46K42 | PIC18(L)F47K42 | PIC18(L)F55K42 | PIC18(L)F56K42 | PIC18(L)F57K42 |
|---|---|---|---|---|---|---|---|---|
| Comparator Module | 2 | | | | | | | |
| Direct Memory Access (DMA) | 2 | | | | | | | |
| Configurable Logic Cell (CLC) | 4 | | | | | | | |
| Peripheral Pin Select (PPS) | Yes | | | | | | | |
| Peripheral Module Disable (PMD) | Yes | | | | | | | |
| 16-bit CRC with Scanner | Yes | | | | | | | |
| Programmable High/ Low-Voltage Detect (HLVD) | Yes | | | | | | | |
| Resets (and Delays) | POR, Programmable BOR, RESET Instruction, Stack Overflow, Stack Underflow (PWRT, OST), $\overline{\text{MCLR}}$, WDT, $\overline{\text{MEMV}}$ | | | | | | | |
| Instruction Set | 81 Instructions; 87 with Extended Instruction Set enabled | | | | | | | |
| Maximum Operating Frequency | 64 MHz | | | | | | | |

**Note 1:** PORTE is partially implemented. Pin RE3 is an input-only pin on 28/40/44/48-pin variants. In addition to that, on 40/44/48-pin variants, PORTE also consists of RE0, RE1 and RE2 pins.

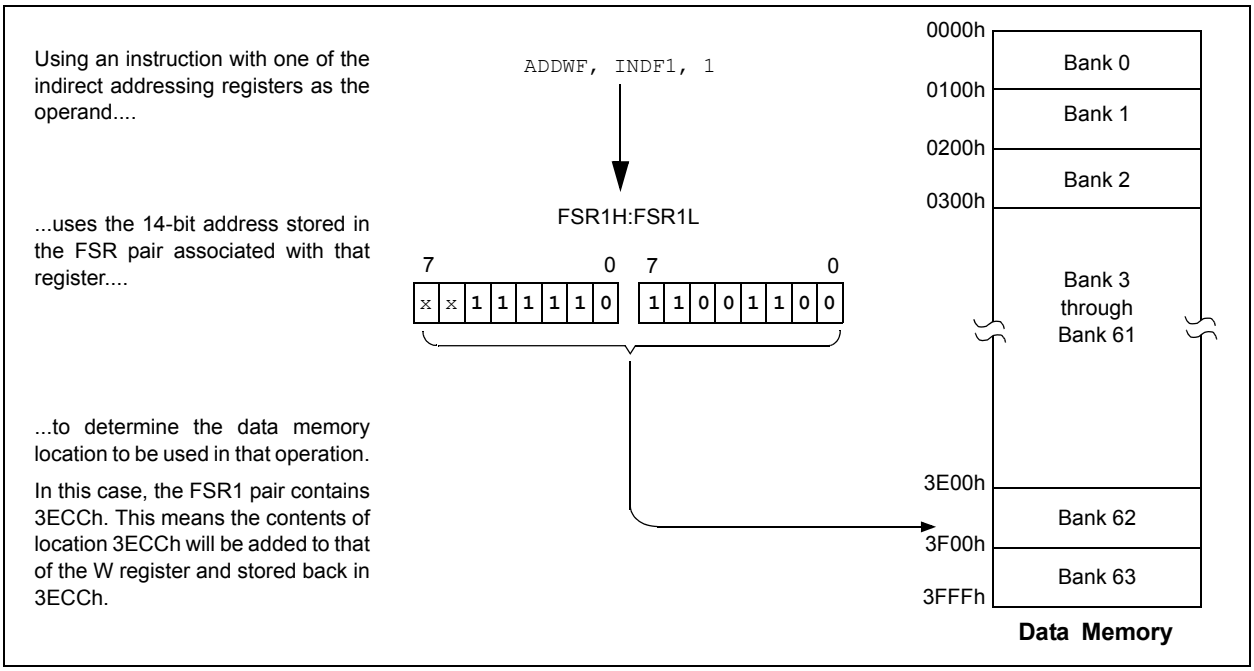
#### 4.7.3.2 FSR Registers, POSTINC, POSTDEC, PREINC and PLUSW

In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are “virtual” registers which cannot be directly read or written. Accessing these registers actually accesses the location to which the associated FSR register pair points, and also performs a specific action on the FSR value. They are:

- POSTDEC: accesses the location to which the FSR points, then automatically decrements the FSR by 1 afterwards
- POSTINC: accesses the location to which the FSR points, then automatically increments the FSR by 1 afterwards
- PREINC: automatically increments the FSR by 1, then uses the location to which the FSR points in the operation
- PLUSW: adds the signed value of the W register (range of -127 to 128) to that of the FSR and uses the location to which the result points in the operation.

In this context, accessing an INDF register uses the value in the associated FSR register without changing it. Similarly, accessing a PLUSW register gives the FSR value an offset by that in the W register; however, neither W nor the FSR is actually changed in the operation. Accessing the other virtual registers changes the value of the FSR register.

**FIGURE 4-6: INDIRECT ADDRESSING**



### 9.3.2 NATURAL ORDER (HARDWARE) PRIORITY

When more than one interrupt with the same user specified priority level are requested, the priority conflict is resolved by using a method called “Natural Order Priority”. Natural order priority is a fixed priority scheme that is based on the Interrupt Vector Table. Table 9-2 shows the natural order priority and the interrupt vector number assigned for each source.

**TABLE 9-2: INTERRUPT VECTOR PRIORITY TABLE**

| Vector Number | Interrupt Source | Vector Number | Interrupt Source |
|---|---|---|---|
| 0 | Software Interrupt | 42 | DMA2SCNT |
| 1 | HLVD | 43 | DMA2DCNT |
| 2 | OSF | 44 | DMA2OR |
| 3 | CSW | 45 | DMA2A |
| 4 | NVM | 46 | I2C2RX |
| 5 | SCAN | 47 | I2C2TX |
| 6 | CRC | 48 | I2C2 |
| 7 | IOC | 49 | I2C2E |
| 8 | INT0 | 50 | U2RX |
| 9 | ZCD | 51 | U2TX |
| 10 | AD | 52 | U2E |
| 11 | ADT | 53 | U2 |
| 12 | C1 | 54 | TMR3 |
| 13 | SMT1 | 55 | TMR3G |
| 14 | SMT1PRA | 56 | TMR4 |
| 15 | SMT1PWA | 57 | CCP2 |
| 16 | DMA1SCNT | 58 | — |
| 17 | DMA1DCNT | 59 | CWG2 |
| 18 | DMA1OR | 60 | CLC2 |
| 19 | DMA1A | 61 | INT2 |
| 20 | SPI1RX | 62 | — |
| 21 | SPI1TX | 63 | — |
| 22 | SPI1 | 64 | — |
| 23 | I2C1RX | 65 | — |
| 24 | I2C1TX | 66 | — |
| 25 | I2C1 | 67 | — |
| 26 | I2C1E | 68 | — |
| 27 | U1RX | 69 | — |
| 28 | U1TX | 70 | TMR5 |
| 29 | U1E | 71 | TMR5G |
| 30 | U1 | 72 | TMR6 |
| 31 | TMR0 | 73 | CCP3 |
| 32 | TMR1 | 74 | CWG3 |
| 33 | TMR1G | 75 | CLC3 |
| 34 | TMR2 | 76 | — |
| 35 | CCP1 | 77 | — |
| 36 | — | 78 | — |
| 37 | NCO | 79 | — |
| 38 | CWG1 | 80 | CCP4 |
| 39 | CLC1 | 81 | CLC4 |
| 40 | INT1 | | |
| 41 | C2 | | |

The natural order priority scheme has vector interrupt 0 as the highest priority and vector interrupt 81 as the lowest priority.

For example, when two concurrently occurring interrupt sources that are both designated high priority using the IPRx register will be resolved using the natural order priority (i.e., the interrupt with a lower corresponding vector number will preempt the interrupt with the higher vector number).

The ability for the user to assign every interrupt source to high or low priority levels means that the user program can give an interrupt with a low natural order priority a higher overall priority level.

## 9.4 Interrupt Operation

All pending interrupts are indicated by the flag bit being equal to a ‘1’ in the PIRx register. All pending interrupts are resolved using the priority scheme explained in **Section 9.3 “Interrupt Priority”**.

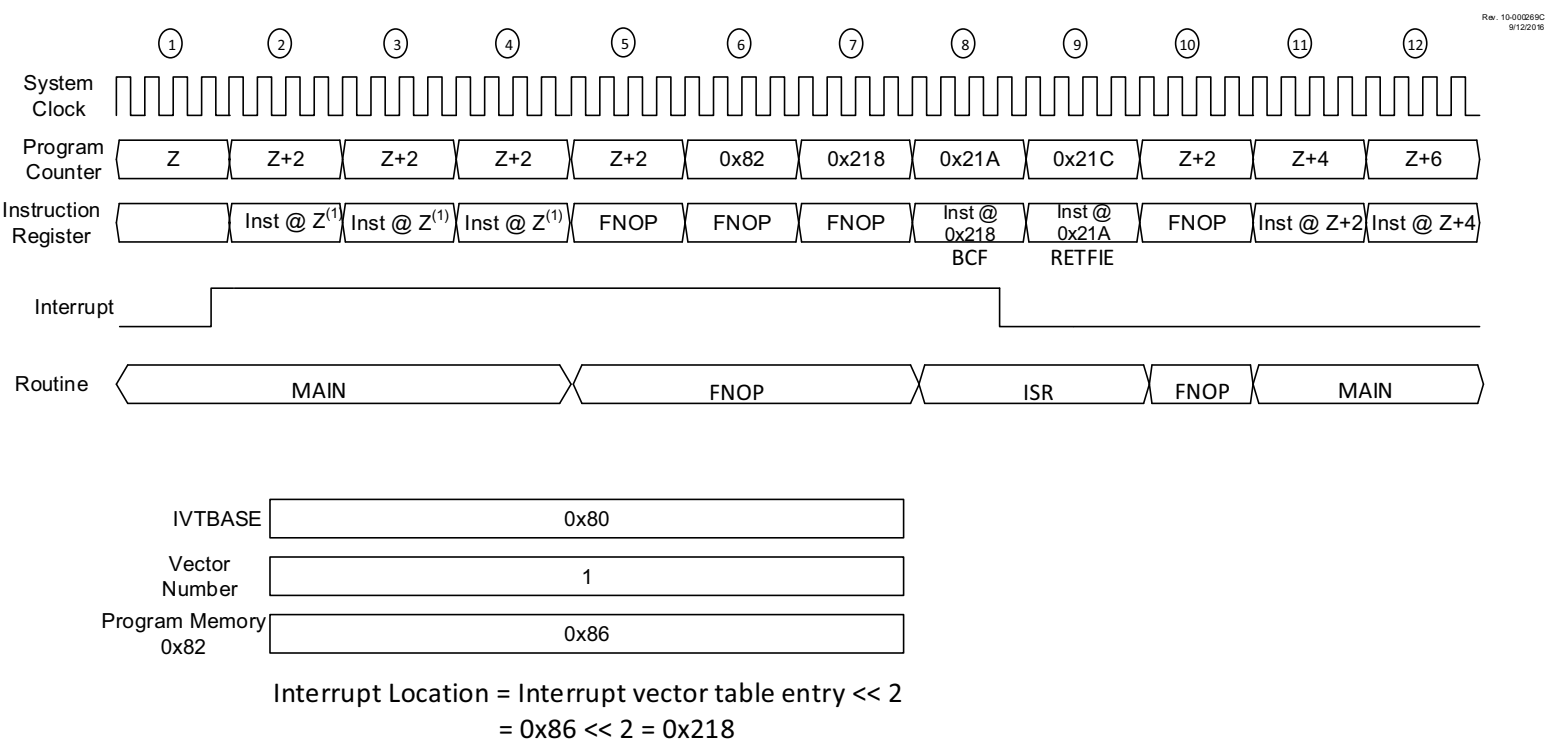
Once the interrupt source to be serviced is resolved, the program execution vectors to the resolved interrupt vector addresses, as explained in **Section 9.2 “Interrupt Vector Table (IVT)”**. The vector number is also stored in the WREG register. Most of the flag bits are required to be cleared by the application software, but in some cases, device hardware clears the interrupt automatically. Some flag bits are read-only in the PIRx registers, these flags are a summary of the source interrupts and the corresponding interrupt flags of the source must be cleared.

A valid interrupt can be either a high or low priority interrupt when in main routine or a high priority interrupt when in low priority Interrupt Service Routine. Depending on order of interrupt requests received and their relative timing, the CPU will be in the state of execution indicated by the STAT bits of the INTCON1 register (Register 9-2).

The State machine shown in Figure 9-1 and the subsequent sections detail the execution of interrupts when received in different orders.

| | |
|---|---|
| **Note:** | The state of GIEH/L is not changed by the hardware when servicing an interrupt. The internal state machine is used to keep track of execution states. These bits can be manipulated in the user code resulting in transferring execution to the main routine and ignoring existing interrupts. |

**FIGURE 9-9: INTERRUPT TIMING DIAGRAM - THREE CYCLE INSTRUCTION**

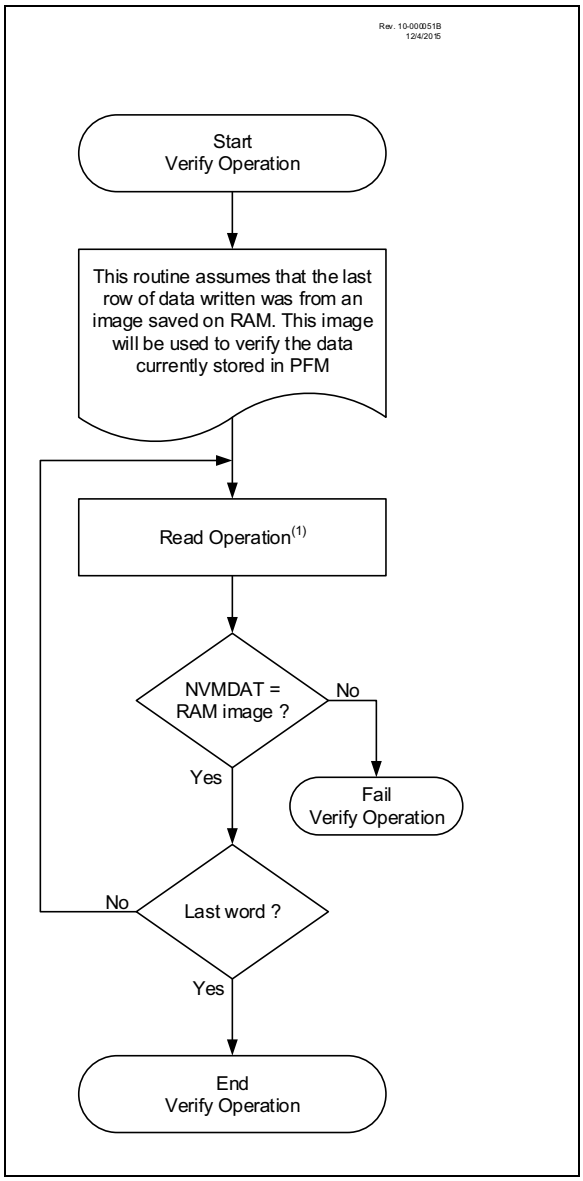



**Note 1:** Instruction @ Z is a Three-cycle instruction.

#### 13.1.6.2 Write Verify

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit. Since program memory is stored as a full page, the stored program memory contents are compared with the intended data stored in RAM after the last write is complete.

**FIGURE 13-10: PROGRAM FLASH MEMORY VERIFY FLOWCHART**




#### 13.1.6.3 Unexpected Termination of Write Operation

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

#### 13.1.6.4 Protection Against Spurious Writes

A write sequence is valid only when both the following conditions are met, this prevents spurious writes which might lead to data corruption.

1. The WR bit is gated through the WREN bit. It is suggested to have the WREN bit cleared at all times except during memory writes. This prevents memory writes if the WR bit gets set accidentally.
2. The NVM unlock sequence must be performed each time before a write operation.

## 13.2 Device Information Area, Device Configuration Area, User ID, Device ID and Configuration Word Access

When REG<1:0> = 0b01 or 0b11 in the NVMCON1 register, the Device Information Area, the Device Configuration Area, the User IDs, Device ID/ Revision ID and Configuration Words can be accessed. Different access may exist for reads and writes (see Table 13-1).

### 13.2.1 Reading Access

The user can read from these blocks by setting the REG bits to 0b01 or 0b11. The user needs to load the address into the TBLPTR registers. Executing a TBLRD after that moves the byte pointed to the TABLAT register. The CPU operation is suspended during the read and resumes after. When read access is initiated on an address outside the parameters listed in Table 13-1, the TABLAT register is cleared, reading back ‘0’s.

### 13.2.2 Writing Access

The WREN bit in NVMCON1 must be set to enable writes. This prevents accidental writes to the CONFIG words due to errant (unexpected) code execution. The WREN bit should be kept clear at all times, except when updating the CONFIG words. The WREN bit is not cleared by hardware. The WR bit will be inhibited from being set unless the WREN bit is set.

### 23.2.3 SOFTWARE INTERRUPT MODE

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE Interrupt Priority bit of the respective PIE register clear to avoid false interrupts. Additionally, the user should clear the CCPxIF interrupt flag bit of the respective PIR register following any change in Operating mode.

### 23.2.4 CAPTURE DURING SLEEP

Capture mode depends upon the Timer1 module for proper operation. There are two options for driving the Timer1 module in Capture mode. It can be driven by the instruction clock (FOSC/4), or by an external clock source.

When Timer1 is clocked by FOSC/4, Timer1 will not increment during Sleep. When the device wakes from Sleep, Timer1 will continue from its previous state.

Capture mode will operate during Sleep as long as the clock source for Timer1 is active in Sleep.

## 23.3 Compare Mode

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

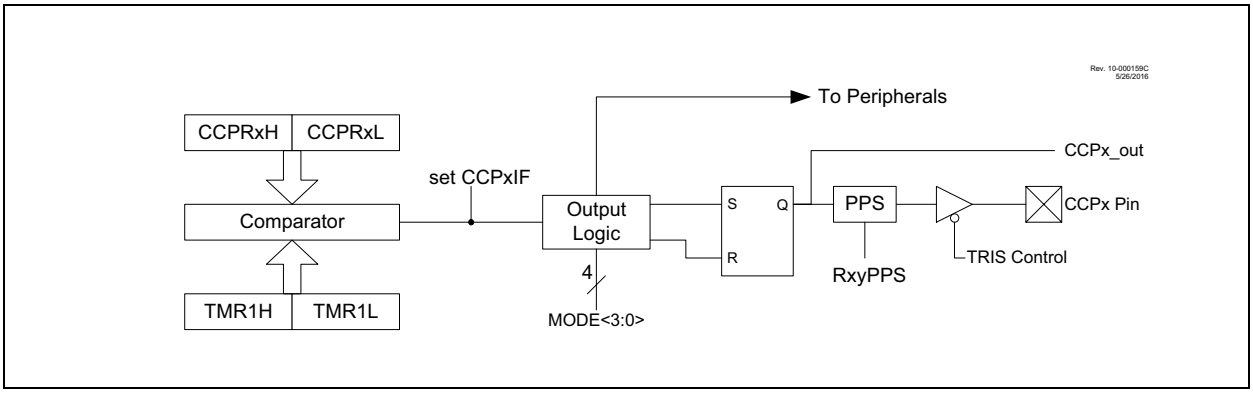
- Toggle the CCPx output, clear TMRx
- Toggle the CCPx output
- Set the CCPx output
- Clear the CCPx output
- Pulse output
- Pulse output, clear TMRx

The action on the pin is based on the value of the MODE<3:0> control bits of the CCPxCON register. At the same time, the interrupt flag CCPxIF bit is set, and an ADC conversion can be triggered, if selected.

All Compare modes can generate an interrupt and trigger an ADC conversion. When MODE = `0b0001` or `0b1011`, the CCP resets the TMR register pair.

Figure 23-2 shows a simplified diagram of the compare operation.

**FIGURE 23-2: COMPARE MODE OPERATION BLOCK DIAGRAM**



**REGISTER 24-2: CCPTMRS1: CCP TIMERS CONTROL REGISTER 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 | R/W-0/0 | R/W-1/1 |
|---|---|---|---|---|---|---|---|
| P8TSEL<1:0> | | P7TSEL<1:0> | | P6TSEL<1:0> | | P5TSEL<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 **P8TSEL<1:0>:** PWM8 Timer Selection bits
11 = PWM8 based on TMR6
10 = PWM8 based on TMR4
01 = PWM8 based on TMR2
00 = Reserved

bit 5-4 **P7TSEL<1:0>:** PWM7 Timer Selection bits
11 = PWM7 based on TMR6
10 = PWM7 based on TMR4
01 = PWM7 based on TMR2
00 = Reserved

bit 3-2 **P6TSEL<1:0>:** PWM6 Timer Selection bits
11 = PWM6 based on TMR6
10 = PWM6 based on TMR4
01 = PWM6 based on TMR2
00 = Reserved

bit 1-0 **P5TSEL<1:0>:** PWM5 Timer Selection bits
11 = PWM5 based on TMR6
10 = PWM5 based on TMR4
01 = PWM5 based on TMR2
00 = Reserved

### 25.6.2 GATED TIMER MODE

Gated Timer mode uses the SMTSIGx input to control whether or not the SMT1TMR will increment. Upon a falling edge of the external signal, the SMT1CPW register will update to the current value of the SMT1TMR. Example waveforms for both repeated and single acquisitions are provided in Figure 25-4 and Figure 25-5.

#### 26.2.3.1 Direction Change in Full-Bridge Mode

In Full-Bridge mode, changing MODE<2:0> controls the forward/reverse direction. Changes to MODE<2:0> change to the new direction on the next rising edge of the modulated input.

A direction change is initiated in software by changing the MODE<2:0> bits of the CWGxCON0 register. The sequence is illustrated in Figure 26-8.

- The associated active output CWGxA and the inactive output CWGxC are switched to drive in the opposite direction.
- The previously modulated output CWGxD is switched to the inactive state, and the previously inactive output CWGxB begins to modulate.
- CWG modulation resumes after the direction-switch dead band has elapsed.

#### 26.2.3.2 Dead-Band Delay in Full-Bridge Mode

Dead-band delay is important when either of the following conditions is true:

1. The direction of the CWG output changes when the duty cycle of the data input is at or near 100%, or
2. The turn-off time of the power switch, including the power device and driver circuit, is greater than the turn-on time.

The dead-band delay is inserted only when changing directions, and only the modulated output is affected. The statically-configured outputs (CWGxA and CWGxC) are not afforded dead band, and switch essentially simultaneously.

Figure 26-8 shows an example of the CWG outputs changing directions from forward to reverse, at near 100% duty cycle. In this example, at time t1, the output of CWGxA and CWGxD become inactive, while output CWGxC becomes active. Since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current will flow through power devices QC and QD for the duration of 't'. The same phenomenon will occur to power devices QA and QB for the CWG direction change from reverse to forward.

When changing the CWG direction at high duty cycle is required for an application, two possible solutions for eliminating the shoot-through current are:

1. Reduce the CWG duty cycle for one period before changing directions.
2. Use switch drivers that can drive the switches off faster than they can drive them on.

**FIGURE 26-8: EXAMPLE OF PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE**

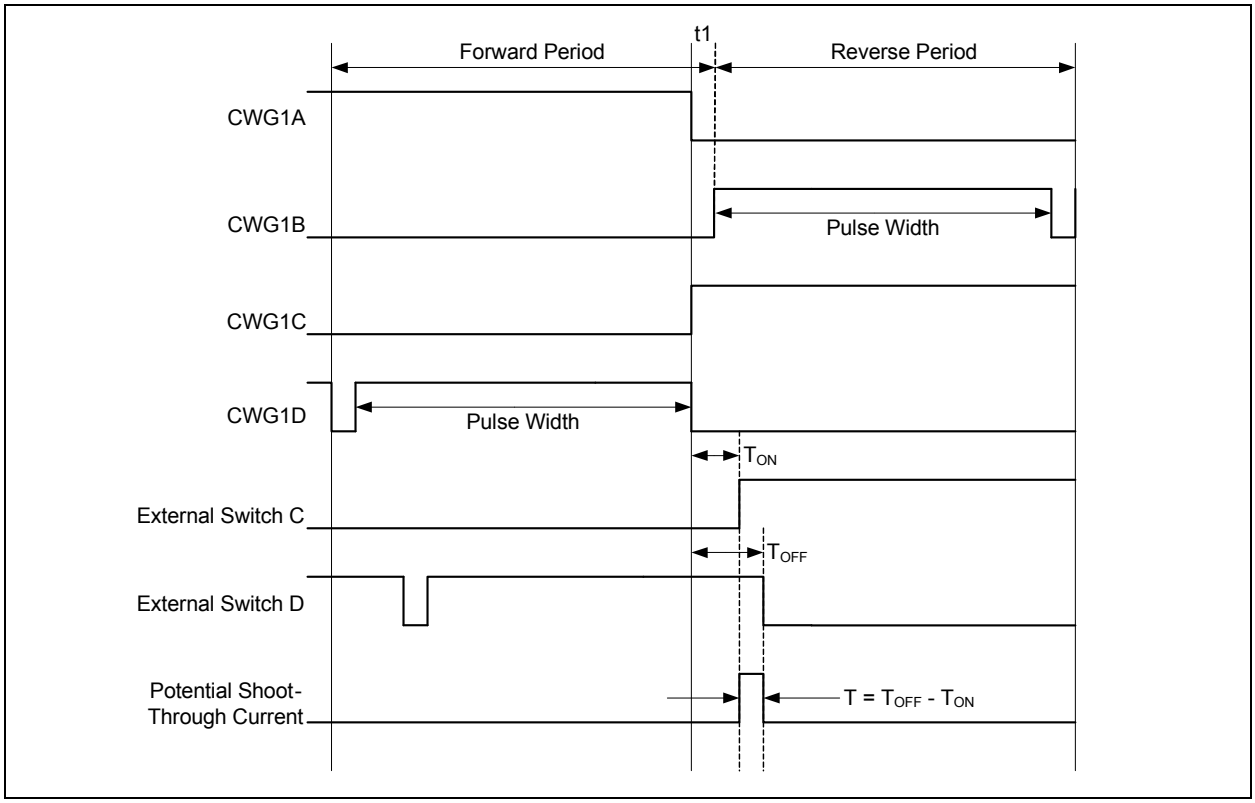
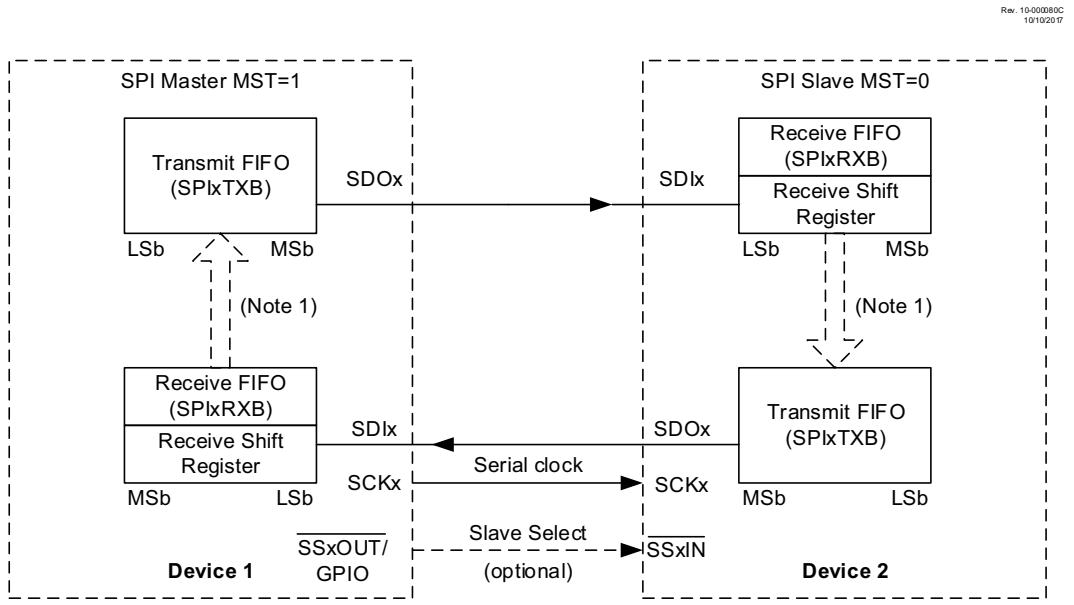


**FIGURE 32-2: SPI MASTER/SLAVE CONNECTION WITH FIFOs**




**Note 1:** In some modes, if the Transmit FIFO is empty, the most recently received byte of data will be transmitted.

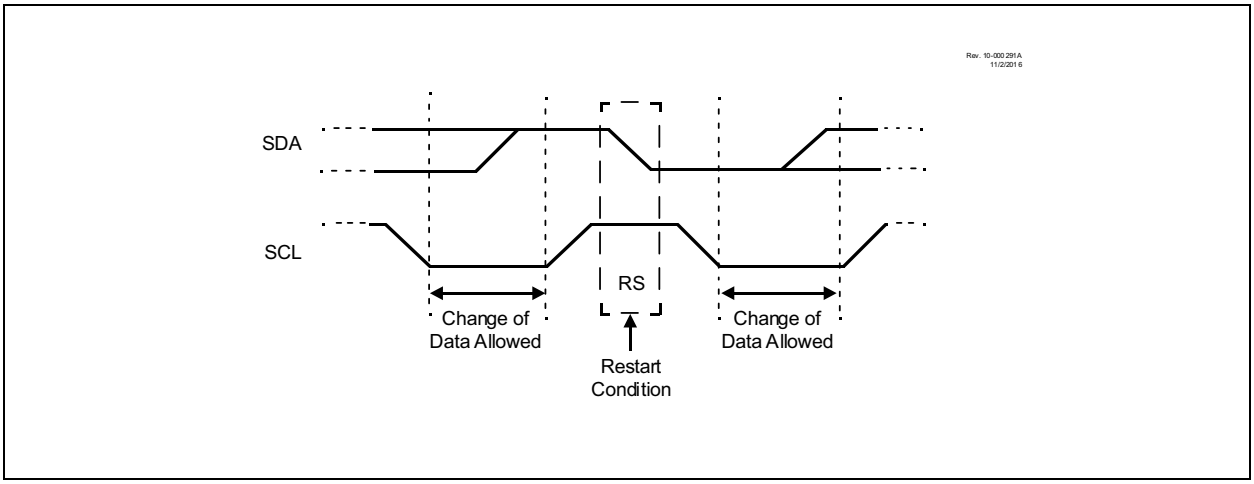
**2:** This diagram assumes that the LSBF bit is cleared (communications are MSb-first). If LSBF is set, the communications will be LSb-first.

### 33.3.7 RESTART CONDITION

A Restart is valid any time that a Stop would be valid. A master can issue a Restart if it wishes to hold the bus after terminating the current transfer. A Restart has the same effect on the slave that a Start would, resetting all slave logic and preparing it to clock in an address. The master may want to address the same or another slave. Figure 33-4 shows the waveform for a Restart condition.

In 10-bit Addressing Slave mode a Restart is required for the master to clock data out of the addressed slave. Once a slave has been fully addressed, matching both high and low address bytes (SMA = 1), the master can issue a Restart and the high address byte with the R/W bit set. The slave logic will then hold the clock and prepare to clock out data.

**FIGURE 33-4: RESTART CONDITION**




### 33.3.8 ACKNOWLEDGE SEQUENCE

The ninth SCL pulse for any transferred byte in $I^2C$ is dedicated as an Acknowledge. It allows receiving devices to respond back to the transmitter by pulling the SDA line low. The transmitter must release control of the line during this time to shift in the response. The Acknowledge (ACK) is an active-low signal, pulling the SDA line low indicates to the transmitter that the device has received the transmitted data and is ready to receive more.

The result of an ACK is placed in the ACKSTAT bit of the I2CxCON1 register. The ACKSTAT bit is cleared when the receiving device sends an Acknowledge and is set when the receiving device does not Acknowledge. A slave sends an Acknowledge when it has recognized its address. When in a mode that is receiving data, the ACK data being sent to the transmitter depends on the value of I2CxCNT register. ACKDT is the value sent when I2CxCNT! = 0. When I2CxCNT = 0, the ACKCNT value is used instead.

In Slave mode, if the ADRIE or WRIE bits are set, clock stretching is initiated when there is an address match or when there is an attempt to write to slave. This allows the user to set the ACK value sent back to the transmitter. The ACKDT bit of the I2CxCON1 register is set/cleared to determine the response. Slave hardware will generate an ACK response if the ADRIE or WRIE bits are clear.

Certain conditions will cause a not-ACK (NACK) to be sent automatically. If any of the RXRE, TXRE, RXO, or TXU bits is set, the hardware response is forced to NACK. All subsequent responses from the device for address matches or data will be a NACK response.

### 33.3.9 BUS TIME-OUT

The I2CxBTO register can be used to select the timeout source for the module. The $I^2C$ module is reset when the selected bus time out signal goes high. This feature is useful for SMBus and PMBus™ compatibility.

For example, Timer2 can be selected as the bus timeout source and configured to count when the SCL pin is low. If the timer runs over before the SCL pin transitioned high, the timer-out pulse will reset the module.

| **Note:** | The bus time-out source should produce a rising edge. |
|---|---|

If the module is configured as a slave and a BTO event occurs when the slave is active, i.e., the SMA bit is set, the module is immediately reset. The SMA and CSTR bits are also cleared, and the BTOIF bit is set.

If a BTO event occurs when the module is configured as a master and is active, (i.e., MMA bit is set), and the module immediately tries to assert a Stop condition and also sets the BTOIF bit. The actual generation of the Stop condition may be delayed if the bus is been clock stretched by some slave device. The MMA bit will be cleared only after the Stop condition is generated.

### 33.3.10 ADDRESS BUFFERS

The $I^2C$ module has two address buffer registers, I2CxADB0 and I2CxADB1. Depending on the mode, these registers are used as either receive or transmit address buffers. See Table 33-2 for data flow directions in these registers. In Slave modes, these registers are only updated when there is an address match. The ADB bit in the I2CxCON2 register is used to enable/ disable the address buffer functionality. When disabled, the address data is sourced from the transmit buffer and is stored in the receive buffer.

**TABLE 33-2: ADDRESS BUFFER DIRECTION AS PER $I^2C$ MODE**

| Modes | MODE<2:0> | I2CxADB0 | I2CxADB1 |
|---|---|---|---|
| Slave (7-bit) | 000 | RX | — |
| | 001 | RX | — |
| Slave (10-bit) | 010 | RX | RX |
| | 011 | RX | RX |
| Master (7-bit) | 100 | — | TX |
| Master (10-bit) | 101 | TX | TX |
| Multi-Master (7-bit) | 110 | RX | TX |
| | 111 | RX | TX |

#### 33.3.10.1 Slave Mode (7-bit)

In 7-bit Slave mode, I2CxADB0 is loaded with the received matching address and R/W data. The I2CxADB1 register is ignored in this mode.

#### 33.3.10.2 Slave Mode (10-bit)

In 10-bit Slave mode, I2CxADB0 is loaded with the lower eight bits of the matching received address. I2CxADB1 is loaded with full eight bits of the high address byte, including the R/W bit.

#### 33.3.10.3 Master Mode (7-bit)

The I2CxADB0 register is ignored in this mode. In 7-bit Master mode, the I2CxADB1 register is used to copy address data byte, including the R/W value, to the shift register.

#### 33.3.10.4 Master Mode (10-bit)

In 10-bit Master mode, the I2CxADB0 register stores the low address data byte value that will be copied to the shift register after the high address byte is shifted out. The I2CxADB1 register stores the high address byte value that will be copied to the shift register. It is up to the user to specify all eight of these bits, even though the $I^2C$ specification defines the upper five bits as a constant.

#### 33.3.10.5 Multi-Master Mode (7-bit only)

In Multi-Master mode, the device can be both master and slave depending on the sequence of events on the bus. If being addressed as a slave, the I2CxADB0 register stores the received matching slave address byte. If the device is trying to communicate as a master on the bus, the contents of the I2CxADB1 register are copied to the shift register for addressing a slave device.

### 33.3.11 RECEIVE AND TRANSMIT BUFFER

The receive buffer holds one byte of data while another is shifted into the SDA pin. The user can access the buffer by software (or DMA) through the I2CxRXB register. When new data is loaded into the I2CxRXB register, the receive buffer full Status bit (RXBF) is set and reading the I2CxRXB register clears this bit.

If the user tries to read I2CxRXB when it is empty (i.e., RXBF = 0), receive read error bit (RXRE) is set and a NACK will be generated. The user must clear the error bit to resume normal operation.

The transmit buffer holds one byte of data while another can be shifted out through the SDA pin. The user can access the buffer by software (or DMA) through the I2CxTXB register. When the I2CxTXB does not contain any transmit data, the transmit buffer empty status bit (TXBE) is set. At this point, the user can load another byte into the buffer.

If the user tries to write I2CxTXB when it is NOT empty (i.e. TXBE = 0), transmit write error flag bit (TXRE) is set and the new data is discarded. When TXRE is set, the user must clear this error condition to resume normal operation.

By setting the CLRBF bit in the I2CxSTAT1 register, the user can clear both receive and transmit buffers. CLRBF will also clear the I2CxRXIF and I2CxTXIF bits.

### 33.3.12 CLOCK STRETCHING

When a slave device has not completed processing data, it can delay the transfer of more data through the process of clock stretching. An addressed slave device may hold the SCL clock line low after receiving or sending a bit, indicating that it is not yet ready to continue. The master will attempt to raise the SCL line in order to transfer the next bit, but will detect that the clock line has not yet been released. Since the SCL connection is open-drain, the slave has the ability to hold the line low until it is ready to continue communicating. Clock stretching allows receivers that cannot keep up with a transmitter to control the flow of incoming data.

### 33.5.11 MASTER TRANSMISSION IN 10-BIT ADDRESSING MODE

This section describes the sequence of events for the $I^2C$ module configured as an $I^2C$ master in 10-bit Addressing mode and is transmitting data. Figure 33-21 is used as a visual reference for this description

1. If ABD = 0; i.e., Address buffers are enabled

Master software loads number of bytes to be transmitted in one sequence in I2CxCNT, high address byte of slave address in I2CxADB1 with R/W = 0, low address byte in I2CxADB0 and the first byte of data in I2CxTXB. Master software has to set the Start (S) bit to initiate communication.

If ABD = 1; i.e., Address buffers are disabled

Master software loads the number of bytes to be transmitted in one sequence in I2CxCNT and the high address byte of the slave address with R/W = 0 into the I2CxTXB register. Writing to the I2CxTXB will assert the start condition on the bus and sets the S bit. Software writes to the S bit are ignored in this case.

2. Master hardware waits for BFRE bit to be set; then shifts out the start and high address and waits for acknowledge.
3. If NACK, master hardware sends Stop.
4. If ABD = 0; i.e., Address buffer are enabled

If ACK, master hardware sends the low address byte from I2CxADB0.

If ABD = 1; i.e., Address buffer are disabled

If ACK, master hardware sets TXIF and MDR bits and the software has to write the low address byte into I2CxTXB. Writing to I2CxTXB sends the low address on the bus.

5. If TXBE = 1 and I2CxCNT! = 0, I2CxTXIF and MDR bits are set. Clock is stretched on 8th falling SCL edge until master software writes next data byte to I2CxTXB.
6. Master hardware sends ninth SCL pulse for ACK from slave and loads the shift register from I2CxTXB. I2CxCNT is decremented.
7. If slave sends a NACK, master hardware sends Stop and ends transmission.
8. If slave sends an ACK, master hardware outputs data in the shift register on SDA. I2CxCNT value is checked on the 8th falling SCL edge. If I2CxCNT = 0; master hardware sends 9th SCL pulse for ACK and CNTIF is set.
9. If I2CxCNT! = 0; go to step 5.

**BRA** **Unconditional Branch**

Syntax: BRA n

Operands: -1024 ≤ n ≤ 1023

Operation: (PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1101 | 0nnn | nnnn | nnnn |

Description: 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' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example: `HERE BRA Jump`

Before Instruction
PC = address `(HERE)`
After Instruction
PC = address `(Jump)`

**BSF** **Bit Set f**

Syntax: BSF f, b {,a}

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

Operation: 1 → f<b>

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1000 | bbba | ffff | ffff |

Description: Bit 'b' in register 'f' is set.
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 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 register 'f' |

Example: `BSF FLAG_REG, 7, 1`

Before Instruction
FLAG_REG = 0Ah
After Instruction
FLAG_REG = 8Ah

**NEGF** **Negate f**

Syntax: NEGF f {,a}

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

Operation: $(\overline{f}) + 1 \rightarrow f$

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

Encoding:

| 0110 | 110a | ffff | ffff |
|---|---|---|---|

Description: Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'.
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 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 register 'f' |

Example:
```
NEGF    REG, 1
```
Before Instruction
REG = 0011 1010 [3Ah]
After Instruction
REG = 1100 0110 [C6h]

**NOP** **No Operation**

Syntax: NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding:

| 0000 1111 | 0000 xxxx | 0000 xxxx | 0000 xxxx |
|---|---|---|---|

Description: No operation.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |

Example:

None.

## 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

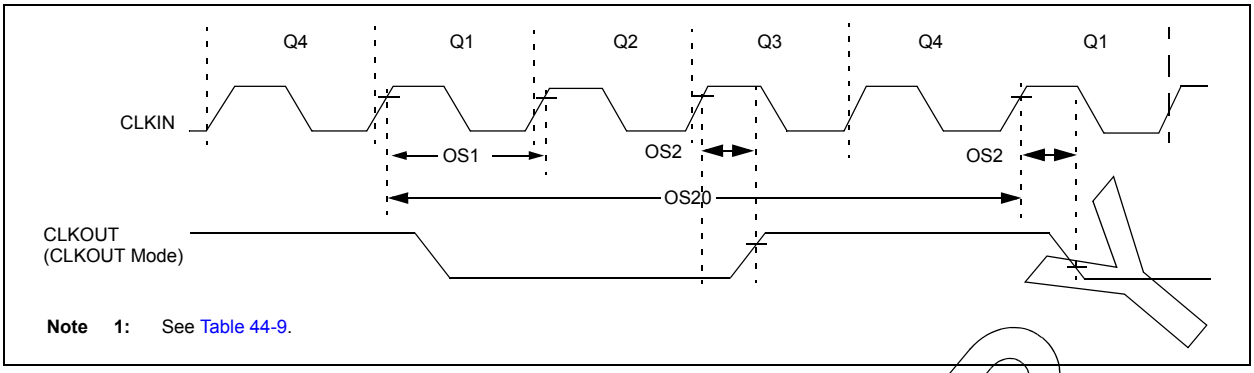
**TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|---|---|---|---|---|---|---|---|---|---|---|
| 3BC8h - 3AEBh | — | Unimplemented | | | | | | | | |
| 3AEAh | U2CTSPPS | — | — | — | U2CTSPPS | | | | | 277 |
| 3AE8h | U2RXPPS | — | — | — | U2RXPPS | | | | | 277 |
| 3AE7h | U1CTSPPS | — | — | — | U1CTSPPS | | | | | 277 |
| 3AE5h | U1RXPPS | — | — | — | U1RXPPS | | | | | 277 |
| 3AE4h | I2C2SDAPPS | — | — | — | I2C2SDAPPS | | | | | 277 |
| 3AE3h | I2C2SCLPPS | — | — | — | I2C2SCLPPS | | | | | 277 |
| 3AE2h | I2C1SDAPPS | — | — | — | I2C1SDAPPS | | | | | 277 |
| 3AE1h | I2C1SCLPPS | — | — | — | I2C1SCLPPS | | | | | 277 |
| 3AE0h | SPI1SSPPS | — | — | — | SPI1SSPPS | | | | | 277 |
| 3ADFh | SPI1SDIPPS | — | — | — | SPI1SDIPPS | | | | | 277 |
| 3ADEh | SPI1SCKPPS | — | — | — | SPI1SCKPPS | | | | | 277 |
| 3ADDh | ADACTPPS | — | — | — | ADACTPPS | | | | | 277 |
| 3ADCh | CLCIN3PPS | — | — | — | CLCIN3PPS | | | | | 277 |
| 3ADBh | CLCIN2PPS | — | — | — | CLCIN2PPS | | | | | 277 |
| 3ADAh | CLCIN1PPS | — | — | — | CLCIN1PPS | | | | | 277 |
| 3AD9h | CLCIN0PPS | — | — | — | CLCIN0PPS | | | | | 277 |
| 3AD8h | MD1SRCPPS | — | — | — | MD1SRCPPS | | | | | 277 |
| 3AD7h | MD1CARHPPS | — | — | — | MD1CARHPPS | | | | | 277 |
| 3AD6h | MD1CARLPPS | — | — | — | MD1CARLPPS | | | | | 277 |
| 3AD5h | CWG3INPPS | — | — | — | CWG3INPPS | | | | | 277 |
| 3AD4h | CWG2INPPS | — | — | — | CWG2INPPS | | | | | 277 |
| 3AD3h | CWG1INPPS | — | — | — | CWG1INPPS | | | | | 277 |
| 3AD2h | SMT1SIGPPS | — | — | — | SMT1SIGPPS | | | | | 277 |
| 3AD1h | SMT1WINPPS | — | — | — | SMT1WINPPS | | | | | 277 |
| 3AD0h | CCP4PPS | — | — | — | CCP4PPS | | | | | 277 |
| 3ACFh | CCP3PPS | — | — | — | CCP3PPS | | | | | 277 |
| 3ACEh | CCP2PPS | — | — | — | CCP2PPS | | | | | 277 |
| 3ACDh | CCP1PPS | — | — | — | CCP1PPS | | | | | 277 |
| 3ACCh | T6INPPS | — | — | — | T6INPPS | | | | | 277 |
| 3ACBh | T4INPPS | — | — | — | T4INPPS | | | | | 277 |
| 3ACAh | T2INPPS | — | — | — | T2INPPS | | | | | 277 |
| 3AC9h | T5GPPS | — | — | — | T5GPPS | | | | | 277 |
| 3AC8h | T5CLKIPPS | — | — | — | T5CLKIPPS | | | | | 277 |
| 3AC7h | T3GPPS | — | — | — | T3GPPS | | | | | 277 |
| 3AC6h | T3CLKIPPS | — | — | — | T3CLKIPPS | | | | | 277 |
| 3AC5h | T1GPPS | — | — | — | T1GPPS | | | | | 277 |
| 3AC4h | T1CLKIPPS | — | — | — | T1CLKIPPS | | | | | 277 |
| 3AC3h | T0CLKIPPS | — | — | — | T0CLKIPPS | | | | | 277 |
| 3AC2h | INT2PPS | — | — | — | INT2PPS | | | | | 277 |
| 3AC1h | INT1PPS | — | — | — | INT1PPS | | | | | 277 |
| 3AC0h | INT0PPS | — | — | — | INT0PPS | | | | | 277 |
| 3ABFh | PPSLOCK | — | — | — | — | — | — | — | PPSLOCKED | 283 |
| 3ABEh | — | Reserved, maintain as '0' | | | | | | | | |
| 3ABDh - 3A9Ah | — | Unimplemented | | | | | | | | |
| 3A99h | — | Reserved, maintain as '0' | | | | | | | | |

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

**Note 1:** Unimplemented in LF devices.
**2:** Unimplemented in PIC18(L)F26/27K42.
**3:** Unimplemented on PIC18(L)F26/27/45/46/47K42 devices.
**4:** Unimplemented in PIC18(L)F45/55K42.

**FIGURE 44-5: CLOCK TIMING**




**Note 1:** See Table 44-9.

**TABLE 44-9: EXTERNAL CLOCK/OSCILLATOR TIMING REQUIREMENTS**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ†** | **Max.** | **Units** | **Conditions** |
| **ECL Oscillator** | | | | | | | |
| OS1 | $F_{ECL}$ | Clock Frequency | — | — | 500 | kHz | |
| OS2 | $T_{ECL\_DC}$ | Clock Duty Cycle | 40 | — | 60 | % | |
| **ECM Oscillator** | | | | | | | |
| OS3 | $F_{ECM}$ | Clock Frequency | — | — | 8 | MHz | |
| OS4 | $T_{ECM\_DC}$ | Clock Duty Cycle | 40 | — | 60 | % | |
| **ECH Oscillator** | | | | | | | |
| OS5 | $F_{ECH}$ | Clock Frequency | — | — | 64 | MHz | |
| OS6 | $T_{ECH\_DC}$ | Clock Duty Cycle | 40 | — | 60 | % | |
| **LP Oscillator** | | | | | | | |
| OS7 | $F_{LP}$ | Clock Frequency | — | — | 100 | kHz | **Note 4** |
| **XT Oscillator** | | | | | | | |
| OS8 | $F_{XT}$ | Clock Frequency | — | — | 4 | MHz | **Note 4** |
| **HS Oscillator** | | | | | | | |
| OS9 | $F_{HS}$ | Clock Frequency | — | — | 20 | MHz | **Note 4** |
| **Secondary Oscillator** | | | | | | | |
| OS10 | $F_{SEC}$ | Clock Frequency | 32.4 | 32.768 | 33.1 | kHz | |
| **System Oscillator** | | | | | | | |
| OS20 | $F_{OSC}$ | System Clock Frequency | — | — | 64 | MHz | **(Note 2, Note 3)** |

* These parameters are characterized but not tested.

† 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:** Instruction cycle period (TCY) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min" values with an external clock applied to OSC1 pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

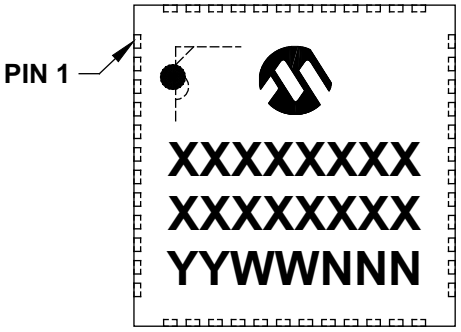
**2:** The system clock frequency (FOSC) is selected by the "main clock switch controls" as described in **Section 10.0 "Power-Saving Operation Modes"**.

**3:** The system clock frequency (FOSC) must meet the voltage requirements defined in the **Section 44.2 "Standard Operating Conditions"**.

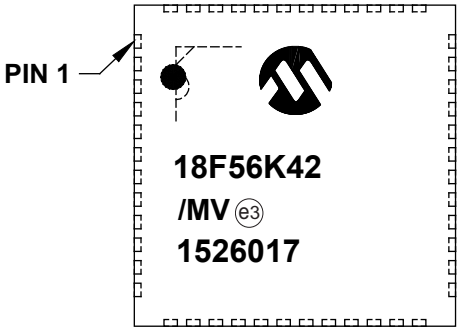
**4:** LP, XT and HS oscillator modes require an appropriate crystal or resonator to be connected to the device. For clocking the device with the external square wave, one of the EC mode selections must be used.

**Package Marking Information (Continued)**

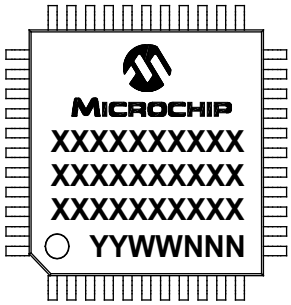
48-Lead UQFN (6x6x0.5 mm)



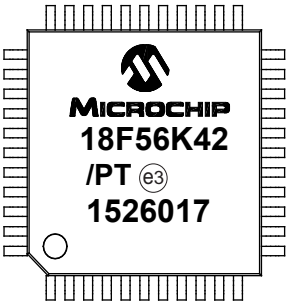

Example




48-Lead TQFP (7x7x1 mm)




Example

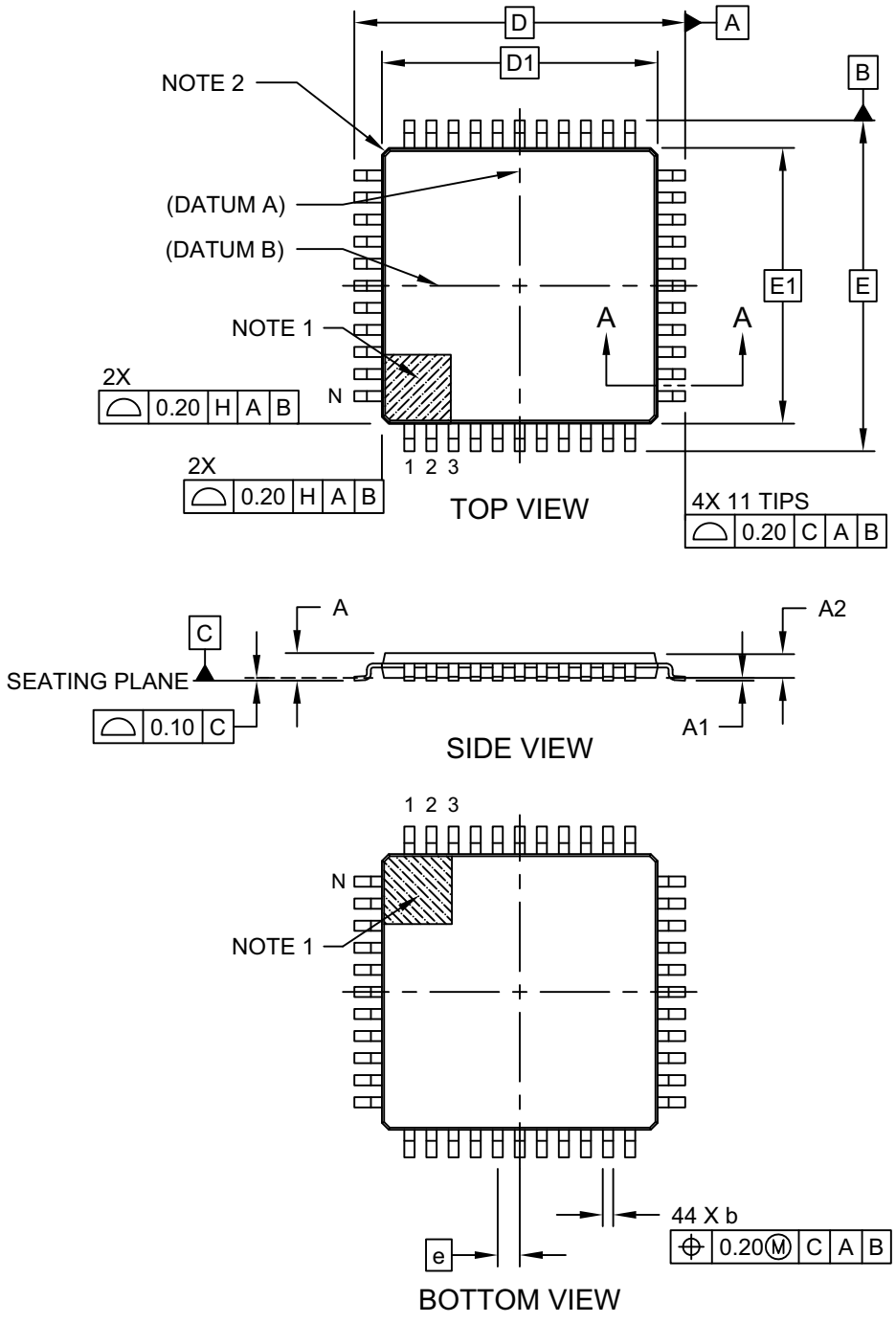



| | | |
|---|---|---|
| **Legend:** | XX...X | Customer-specific information or Microchip part number |
| | Y | Year code (last digit of calendar year) |
| | YY | Year code (last 2 digits of calendar year) |
| | WW | Week code (week of January 1 is week ‘01’) |
| | NNN | Alphanumeric traceability code |
| | (e3) | Pb-free JEDEC® designator for Matte Tin (Sn) |
| | * | This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package. |

**Note**: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

### 44-Lead Plastic Thin Quad Flatpack (PT) - 10x10x1.0 mm Body [TQFP]

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




Microchip Technology Drawing C04-076C Sheet 1 of 2