**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 | CANbus, I²C, LINbus, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, DMA, POR, PWM, WDT |
| Number of I/O | 25 |
| Program Memory Size | 32KB (16K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | 1K x 8 |
| RAM Size | 2K 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-SSOP (0.209", 5.30mm Width) |
| Supplier Device Package | 28-SSOP |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic18f25k83t-i-ss |

**TABLE 3-2: SUMMARY OF REGISTERS ASSOCIATED WITH CPU**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|---|---|---|---|---|---|---|---|---|---|
| ISRPR | — | — | — | — | — | ISRPR2 | ISRPR1 | ISRPR0 | 20 |
| MAINPR | — | — | — | — | — | MAINPR2 | MAINPR1 | MAINPR0 | 20 |
| DMA1PR | — | — | — | — | — | DMA1PR2 | DMA1PR1 | DMA1PR0 | 20 |
| DMA2PR | — | — | — | — | — | DMA2PR2 | DMA2PR1 | DMA2PR0 | 21 |
| SCANPR | — | — | — | — | — | SCANPR2 | SCANPR1 | SCANPR0 | 21 |
| PRLOCK | — | — | — | — | — | — | — | PRLOCKED | 21 |

**Legend:** — = Unimplemented location, read as '0'.

### 9.4.3 PREEMPTING LOW PRIORITY INTERRUPTS

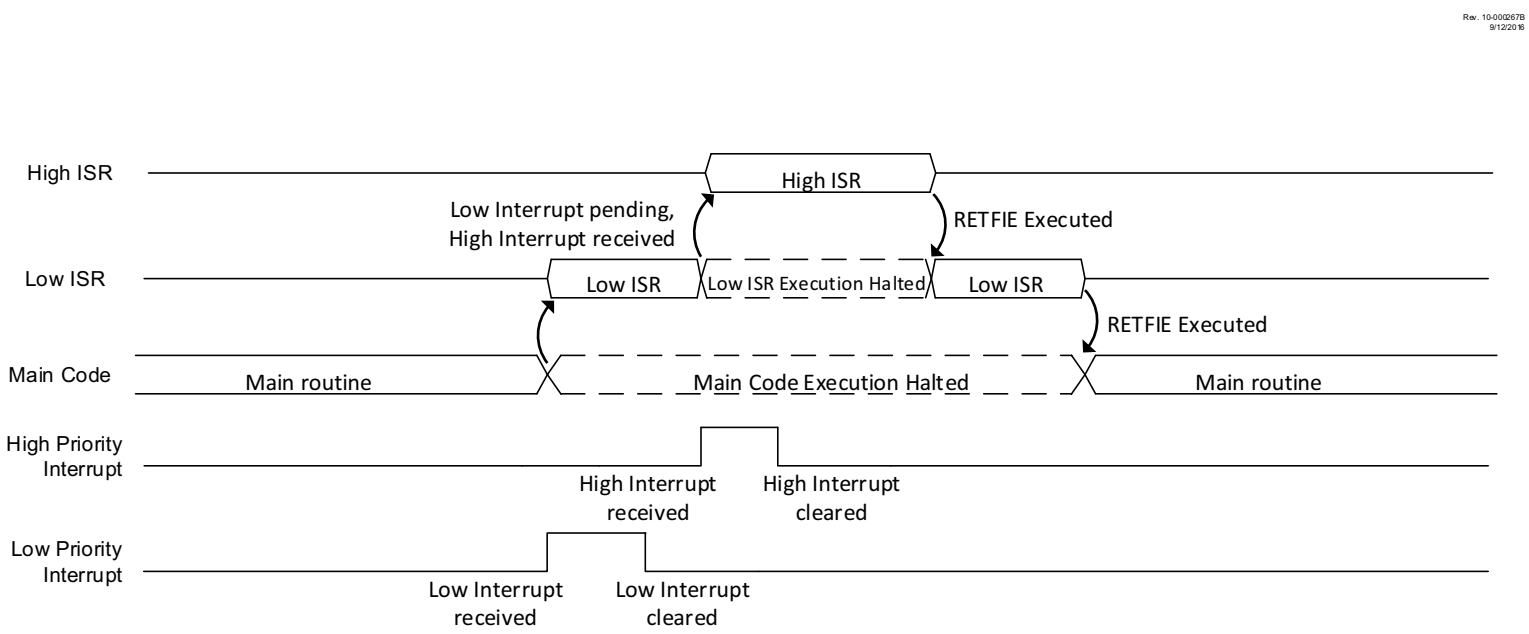
Low-priority interrupts can be preempted by high priority interrupts. While in the low priority ISR, if a high-priority interrupt arrives, the high priority interrupt request is generated and the low priority ISR is suspended, while the high priority ISR is executed, see Figure 9-4.

After the high priority ISR is complete and if any other high priority interrupt requests are not active, the execution returns to the preempted low priority ISR.

**Note 1:** The high priority interrupt flag must be cleared to avoid recursive interrupts.

**2:** If a low-priority ISR was already serviced halfway before moving on to a high priority ISR, then the low priority ISR is completely serviced even if user code clears GIEL.

**FIGURE 9-4: INTERRUPT EXECUTION: HIGH PRIORITY INTERRUPT PREEMPTING LOW PRIORITY INTERRUPTS**



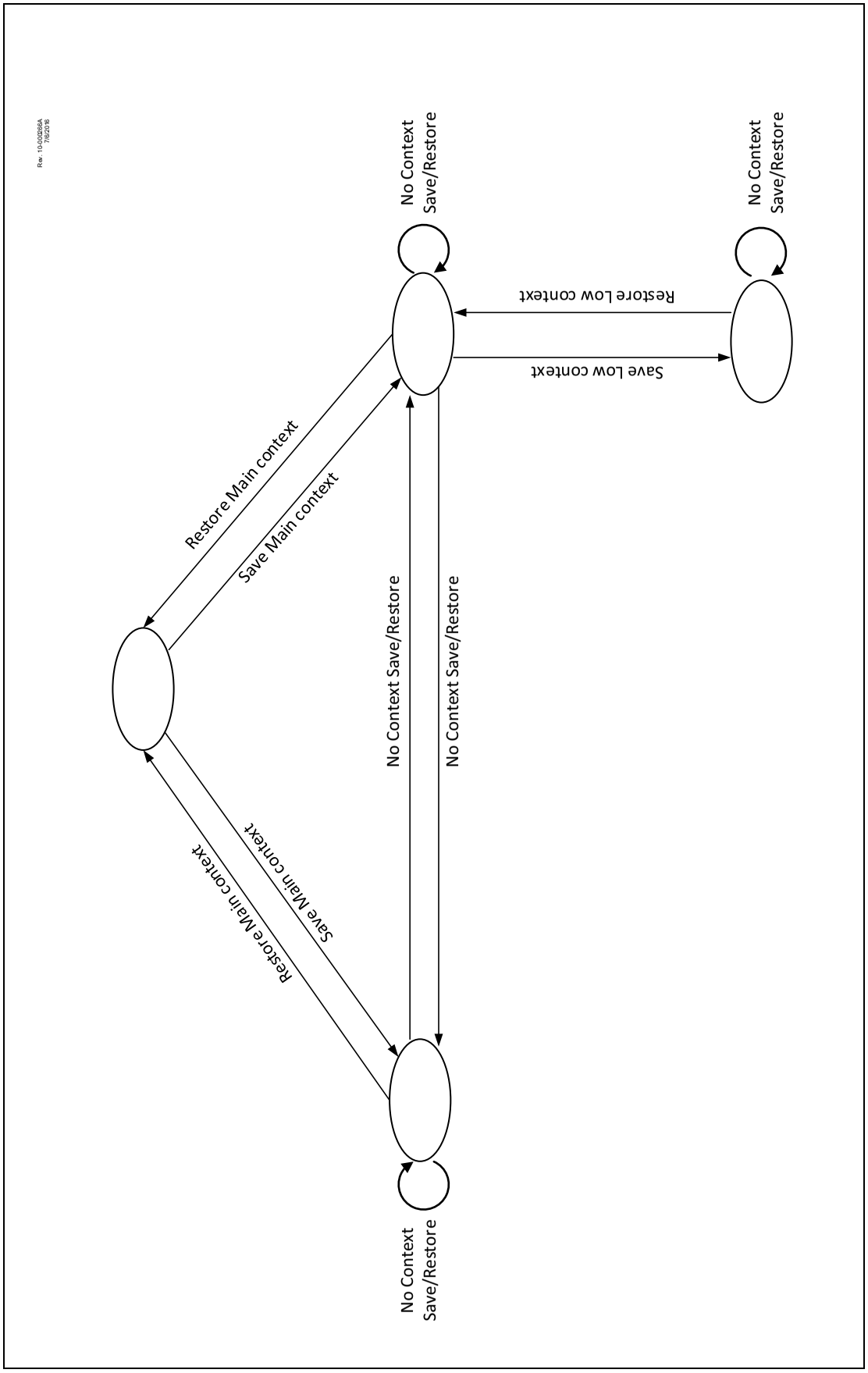



**FIGURE 9-6: CONTEXT SAVE STATE MACHINE DIAGRAM**

### 10.1.2 INTERRUPTS DURING DOZE

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

If an interrupt occurs and the ROI bit is set (ROI = 1) at the time of the interrupt, the DOZEN bit is cleared and the CPU executes at full speed. The prefetched instruction is executed and then the interrupt vector sequence is executed. In Figure 10-1, the interrupt occurs during the 2nd instruction cycle of the Doze period, and immediately brings the CPU out of Doze. If the Doze-On-Exit (DOE) bit is set (DOE = 1) when the RETFIE operation is executed, DOZEN is set, and the CPU executes at the reduced rate based on the DOZE<2:0> ratio.

**EXAMPLE 10-1: DOZE SOFTWARE EXAMPLE**

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

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

## 10.2 Sleep Mode

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

Upon entering Sleep mode, the following conditions exist:

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

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

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

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

I/O pins that are high-impedance inputs should be pulled to VDD or VSS externally to avoid switching currents caused by floating inputs.

Examples of internal circuitry that might be sourcing current include modules such as the DAC and FVR modules. See **Section 38.0 “5-Bit Digital-to-Analog Converter (DAC) Module”** and **Section 35.0 “Fixed Voltage Reference (FVR)”** for more information on these modules.

# 12.0 8x8 HARDWARE MULTIPLIER

## 12.1 Introduction

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

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

## 12.2 Operation

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

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

**EXAMPLE 12-1: 8x8 UNSIGNED MULTIPLY ROUTINE**

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

**EXAMPLE 12-2: 8x8 SIGNED MULTIPLY ROUTINE**

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

**TABLE 12-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS**

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

### 13.1.5 ERASING PROGRAM FLASH MEMORY

The minimum erase block is 64 words (refer to Table 5-4). Only through the use of an external programmer, or through ICSP™ control, can larger blocks of program memory be bulk erased. Word erase in the program memory array is not supported.

For example, when initiating an erase sequence from a microcontroller with erase row size of 64 words, a block of 64 words (128 bytes) of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:6> point to the block being erased. The TBLPTR<5:0> bits are ignored.

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

The NVM unlock sequence described in **Section 13.1.4 "NVM Unlock Sequence"** should be used to guard against accidental writes. This is sometimes referred to as a long write.

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

#### 13.1.5.1 Program Flash Memory Erase Sequence

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

1. REG bits of the NVMCON1 register point to PFM
2. Set the FREE and WREN bits of the NVMCON1 register
3. Perform the unlock sequence as described in **Section 13.1.4 "NVM Unlock Sequence"**

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

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

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

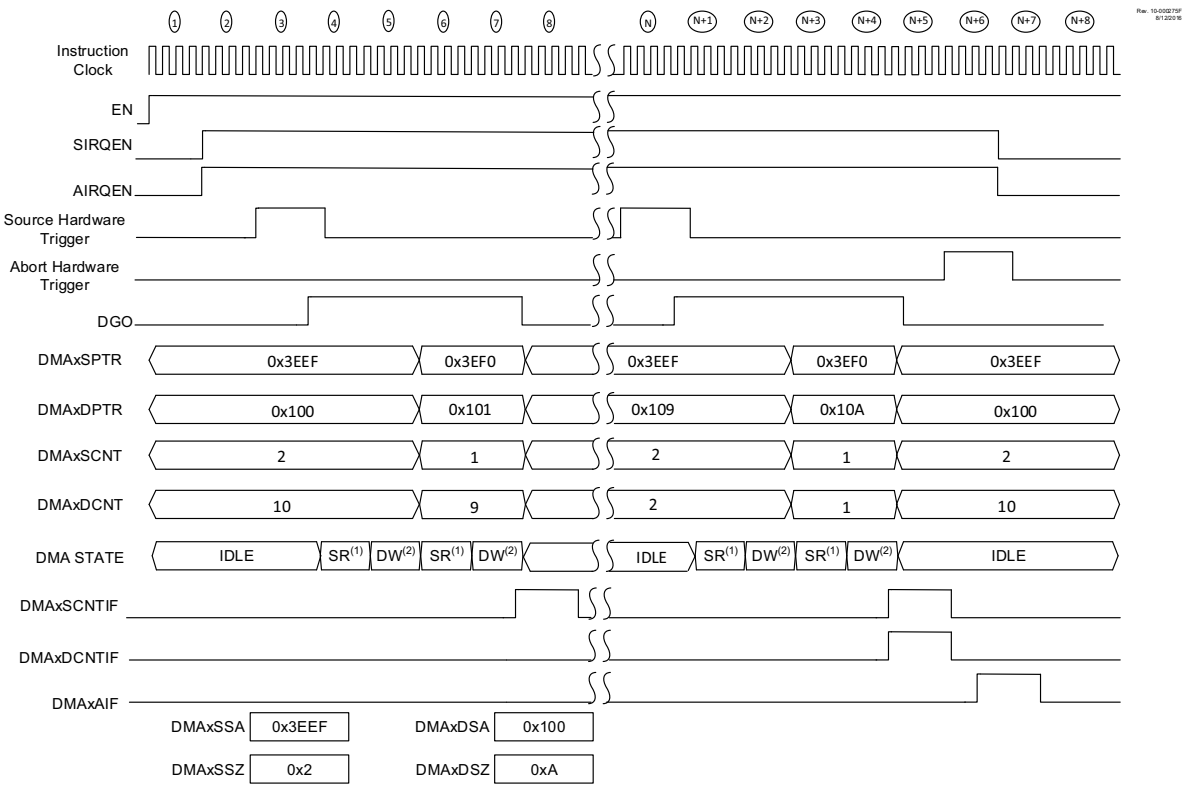
Write latch data is not affected by erase operations and WREN will remain unchanged.

| | |
|---|---|
| **Note 1:** | If a write or erase operation is terminated by an unexpected event, WRERR bit will be set which the user can check to decide whether a rewrite of the location(s) is needed. |
| **2:** | WRERR is set if WR is written to '1' while TBLPTR points to a write-protected address. |
| **3:** | WRERR is set if WR is written to '1' while TBLPTR points to an invalid address location (Table 13-1). |

### 15.9.6 ABORT TRIGGER, MESSAGE COMPLETE

The AIRQEN needs to be set in order for the DMA to sample Abort Interrupt sources. When an abort interrupt is received the SIRQEN bit is cleared and the AIRQEN bit is cleared to avoid receiving further abort triggers.

**FIGURE 15-10: ABORT AT THE END OF MESSAGE**




**Note 1:** SR – Source Read

**2:** DW – Destination Write

**REGISTER 16-5: WPUx: WEAK PULL-UP REGISTER**

| 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 |
|---|---|---|---|---|---|---|---|
| WPUx7 | WPUx6 | WPUx5 | WPUx4 | WPUx3 | WPUx2 | WPUx1 | WPUx0 |
| bit 7 | | | | | | | bit 0 |

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

bit 7-0 **WPUx<7:0>:** Weak Pull-up PORTx Control bits
1 = Weak Pull-up enabled
0 = Weak Pull-up disabled

**TABLE 16-6: WEAK PULL-UP PORT REGISTERS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| WPUA | WPUA7 | WPUA6 | WPUA5 | WPUA4 | WPUA3 | WPUA2 | WPUA1 | WPUA0 |
| WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 |
| WPUC | WPUC7 | WPUC6 | WPUC5 | WPUC4 | WPUC3 | WPUC2 | WPUC1 | WPUC0 |
| WPUE | — | — | — | — | WPUE3(1) | — | — | — |

**Note 1:** If MCLRE = 1, the weak pull-up in RE3 is always enabled; bit WPUE3 is not affected.

**FIGURE 21-5: TIMER1/3/5 GATE TOGGLE MODE**

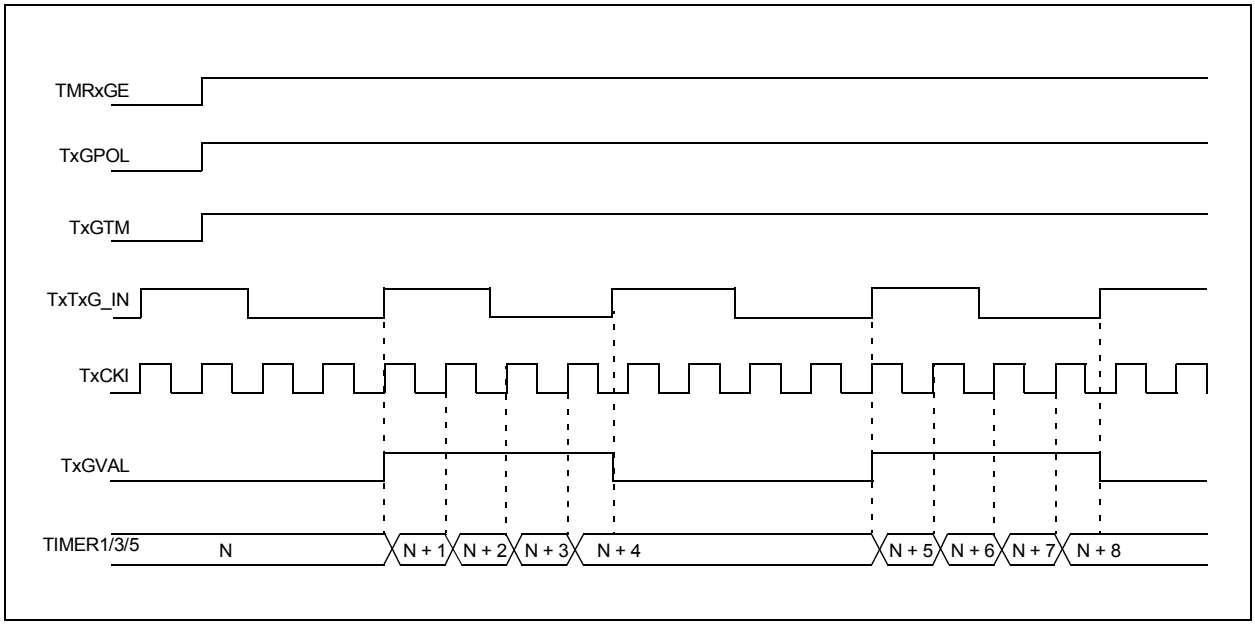



**FIGURE 21-6: TIMER1/3/5 GATE SINGLE-PULSE MODE**

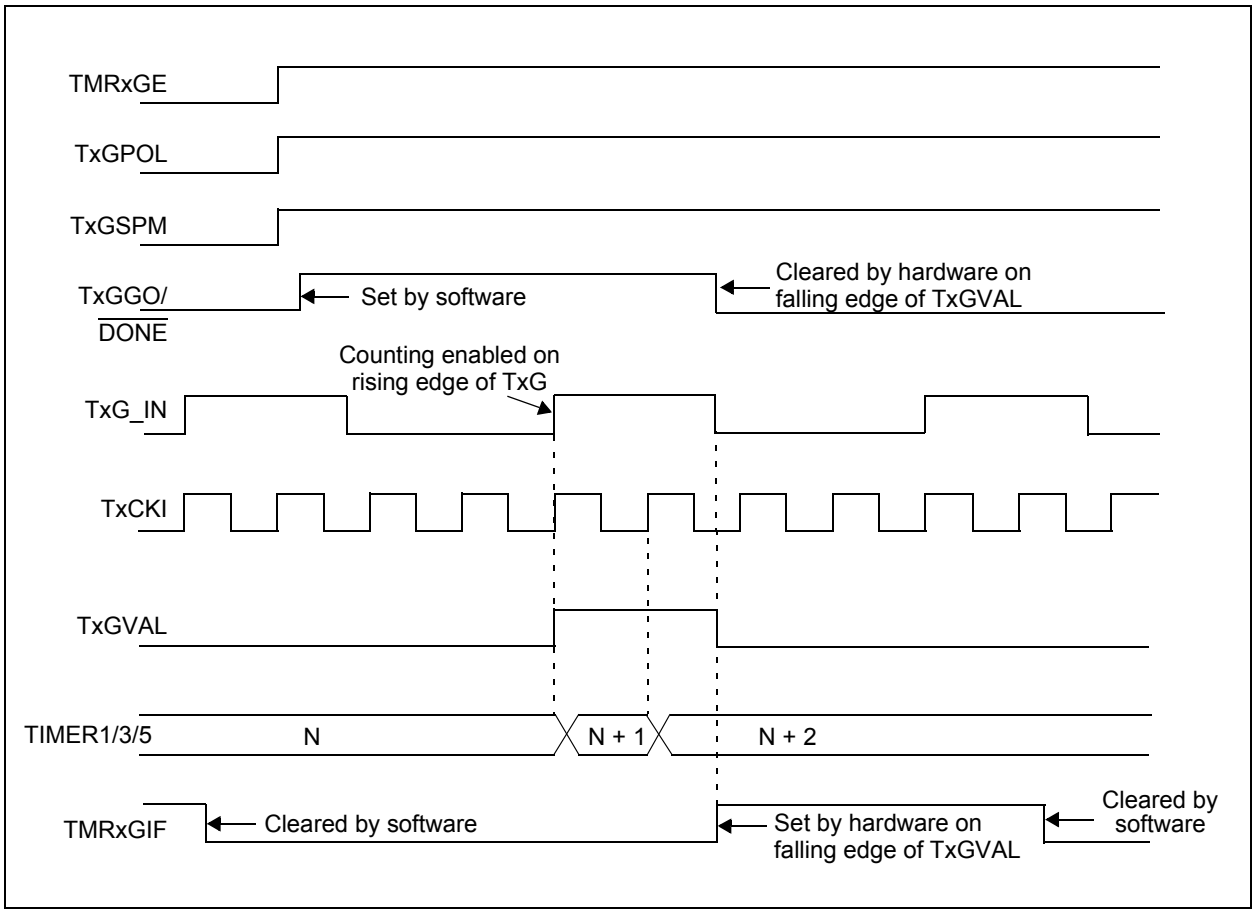
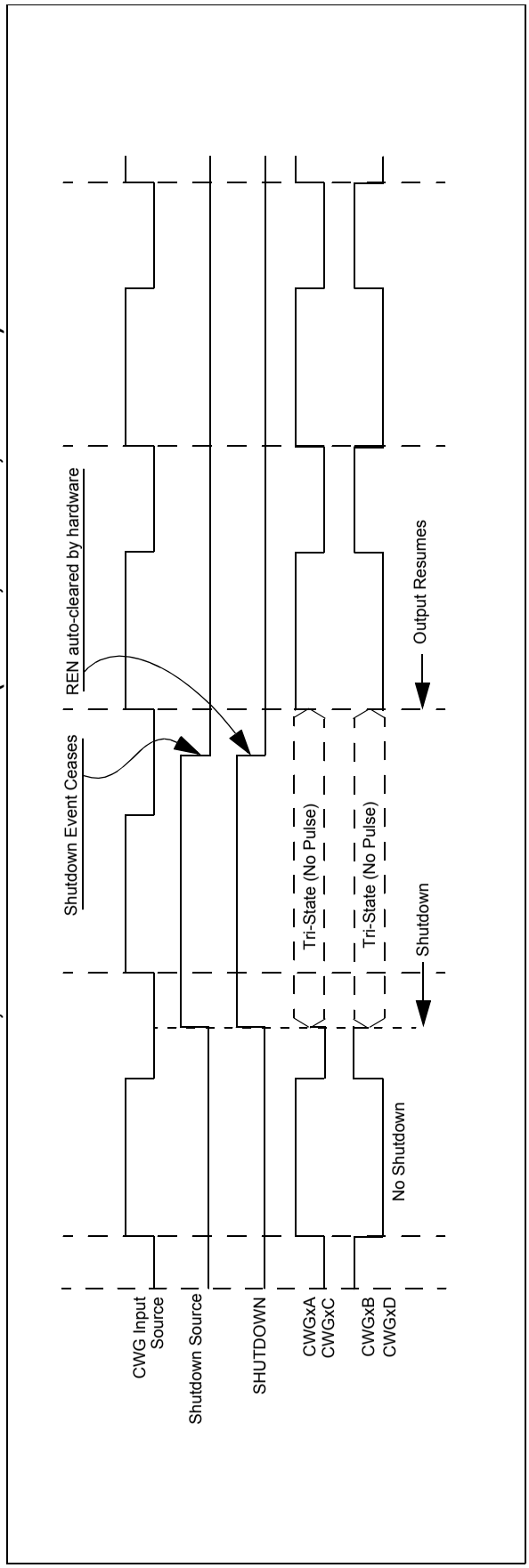


**FIGURE 26-16: SHUTDOWN FUNCTIONALITY, AUTO-RESTART ENABLED (REN = 1, LSAC = 01, LSBD = 01)**



## 28.8 NCO Control Registers

**REGISTER 28-1: NCO1CON: NCO CONTROL REGISTER**

| R/W-0/0 | U-0 | R-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| EN | — | OUT | POL | — | — | — | PFM |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **EN:** NCO1 Enable bit
1 = NCO1 module is enabled
0 = NCO1 module is disabled

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

bit 5 **OUT:** NCO1 Output bit
Displays the current output value of the NCO1 module.

bit 4 **POL:** NCO1 Polarity
1 = NCO1 output signal is inverted
0 = NCO1 output signal is not inverted

bit 3-1 **Unimplemented**: Read as '0'

bit 0 **PFM:** NCO1 Pulse Frequency Mode bit
1 = NCO1 operates in Pulse Frequency mode
0 = NCO1 operates in Fixed Duty Cycle mode, divide by 2

**FIGURE 32-12: TRADITIONAL SPI DAISY – CHAIN CONNECTION**

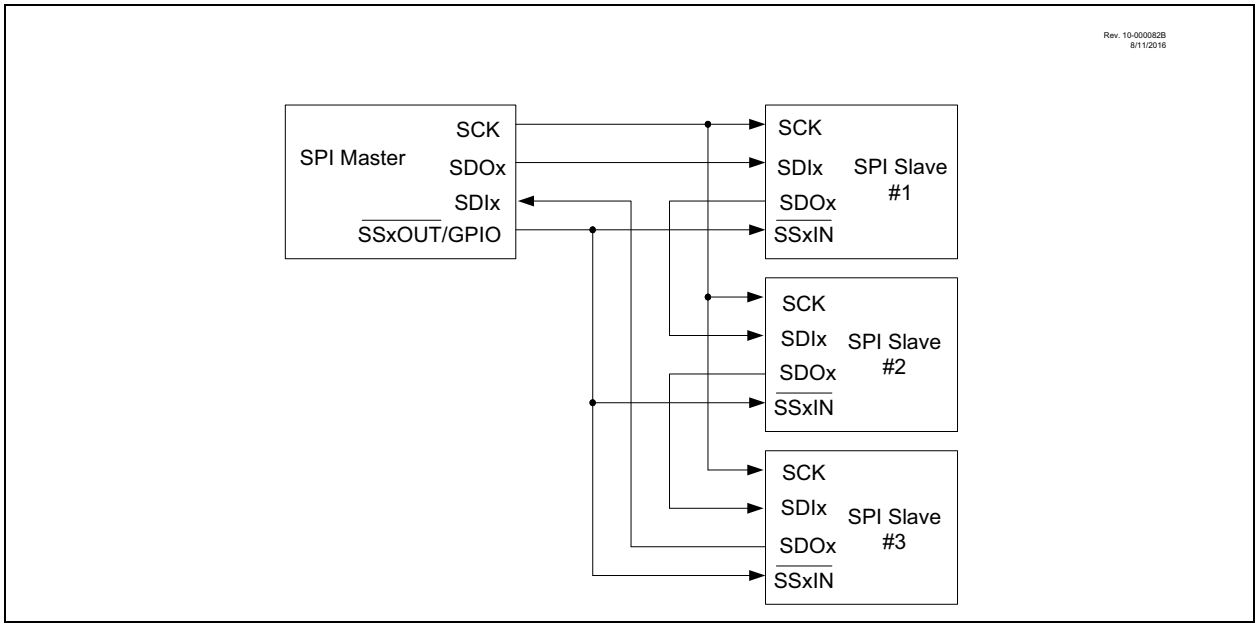
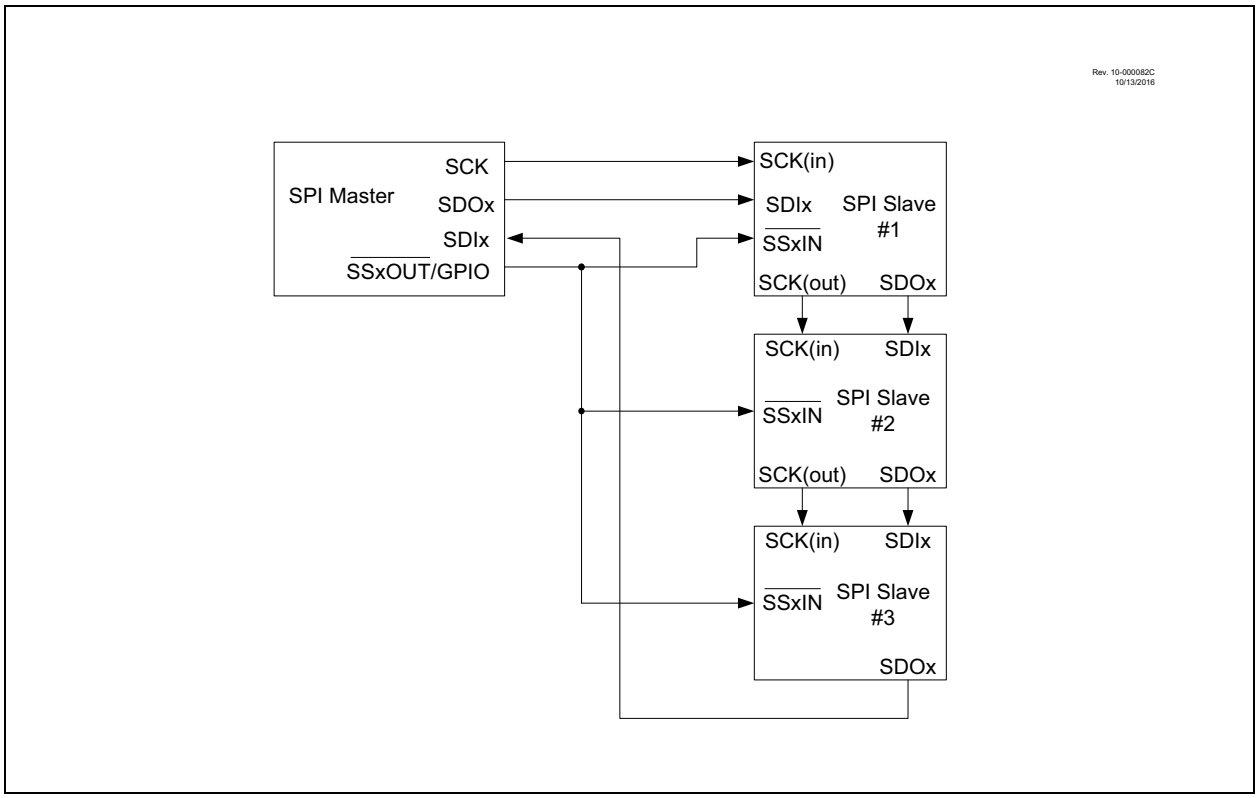



**FIGURE 32-13: SPI DAISY-CHAIN CONNECTION WITH CHAINED SCK**



**REGISTER 34-60: BIE0: BUFFER INTERRUPT ENABLE REGISTER 0(1)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| B5IE(2) | B4IE(2) | B3IE(2) | B2IE(2) | B1IE(2) | B0IE(2) | RXB1IE(2) | RXB0IE(2) |
| 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 **B<5:0>IE:** Programmable Transmit/Receive Buffer 5-0 Interrupt Enable bits(2)
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 1-0 **RXB<1:0>IE:** Dedicated Receive Buffer 1-0 Interrupt Enable bits(2)
1 = Interrupt is enabled
0 = Interrupt is disabled

**Note 1:** This register is available in Mode 1 and 2 only.
**2:** Either TXBnIE or RXBnIE, in the PIE5 register, must be set to get an interrupt.

**TABLE 42-2: INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| **CONTROL INSTRUCTIONS** | | | | | | | | | |
| BC | n | Branch if Carry | 1 (2) | 1110 | 0010 | nnnn | nnnn | None | 1 |
| BN | n | Branch if Negative | 1 (2) | 1110 | 0110 | nnnn | nnnn | None | 1 |
| BNC | n | Branch if Not Carry | 1 (2) | 1110 | 0011 | nnnn | nnnn | None | 1 |
| BNN | n | Branch if Not Negative | 1 (2) | 1110 | 0111 | nnnn | nnnn | None | 1 |
| BNOV | n | Branch if Not Overflow | 1 (2) | 1110 | 0101 | nnnn | nnnn | None | 1 |
| BNZ | n | Branch if Not Zero | 2 | 1110 | 0001 | nnnn | nnnn | None | 1 |
| BOV | n | Branch if Overflow | 1 (2) | 1110 | 0100 | nnnn | nnnn | None | 1 |
| BRA | n | Branch Unconditionally | 1 (2) | 1101 | 0nnn | nnnn | nnnn | None | |
| BZ | n | Branch if Zero | 1 (2) | 1110 | 0000 | nnnn | nnnn | None | 1 |
| CALL | n, s | Call subroutine 1st word | 2 | 1110 | 110s | nnnn | nnnn | None | 2 |
| | | 2nd word | | 1111 | nnnn | nnnn | nnnn | | |
| GOTO | n | Go to address 1st word | 2 | 1110 | 1111 | nnnn | nnnn | None | 2 |
| | — | 2nd word | | 1111 | nnnn | nnnn | nnnn | | |
| CALLW | — | W -> PCL and Call subroutine | 2 | 0000 | 0000 | 0001 | 0100 | None | 1 |
| RCALL | n | Relative Call | 2 | 1101 | 1nnn | nnnn | nnnn | None | 1 |
| RETFIE | s | Return from interrupt enable | 2 | 0000 | 0000 | 0001 | 000s | None | 1 |
| RETLW | k | Return with literal in WREG | 2 | 0000 | 1100 | kkkk | kkkk | None | 1 |
| RETURN | s | Return from Subroutine | 2 | 0000 | 0000 | 0001 | 001s | None | 1 |
| **INHERENT INSTRUCTIONS** | | | | | | | | | |
| CLRWDT | — | Clear Watchdog Timer | 1 | 0000 | 0000 | 0000 | 0100 | None | |
| DAW | — | Decimal Adjust WREG | 1 | 0000 | 0000 | 0000 | 0111 | C | |
| NOP | — | No Operation | 1 | 0000 | 0000 | 0000 | 0000 | None | |
| NOP | — | No Operation | 1 | 1111 | xxxx | xxxx | xxxx | None | 2 |
| POP | — | Pop top of return stack (TOS) | 1 | 0000 | 0000 | 0000 | 0110 | None | |
| PUSH | — | Push top of return stack (TOS) | 1 | 0000 | 0000 | 0000 | 0101 | None | |
| RESET | | Software device Reset | 1 | 0000 | 0000 | 1111 | 1111 | All | |
| SLEEP | — | Go into Standby mode | 1 | 0000 | 0000 | 0000 | 0011 | None | |

**Note 1:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires an additional cycle. The extra cycle is executed as a NOP.

**2:** Some instructions are multi word instructions. The second/third words of these instructions will be decoded as a NOP, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.

**3:** $f_s$ and $f_d$ do not cover the full memory range. 2 MSBs of bank selection are forced to ‘b00 to limit the range of these instructions to lower 4k addressing space.

**CLRWDT** **Clear Watchdog Timer**

Syntax: CLRWDT

Operands: None

Operation: 000h → WDT,
000h → WDT postscaler,
1 → $\overline{TO}$,
1 → $\overline{PD}$

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

Encoding:

| 0000 | 0000 | 0000 | 0100 |
|---|---|---|---|

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits, $\overline{TO}$ and $\overline{PD}$, are set.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: CLRWDT

Before Instruction
WDT Counter = ?
After Instruction
WDT Counter = 00h
WDT Postscaler = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

**COMF** **Complement f**

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

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

Operation: ($\bar{f}$) → dest

Status Affected: N, Z

Encoding:

| 0001 | 11da | ffff | ffff |
|---|---|---|---|

Description: The contents of register 'f' are complemented. 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 ≤ 95 (5Fh). See **Section 42.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

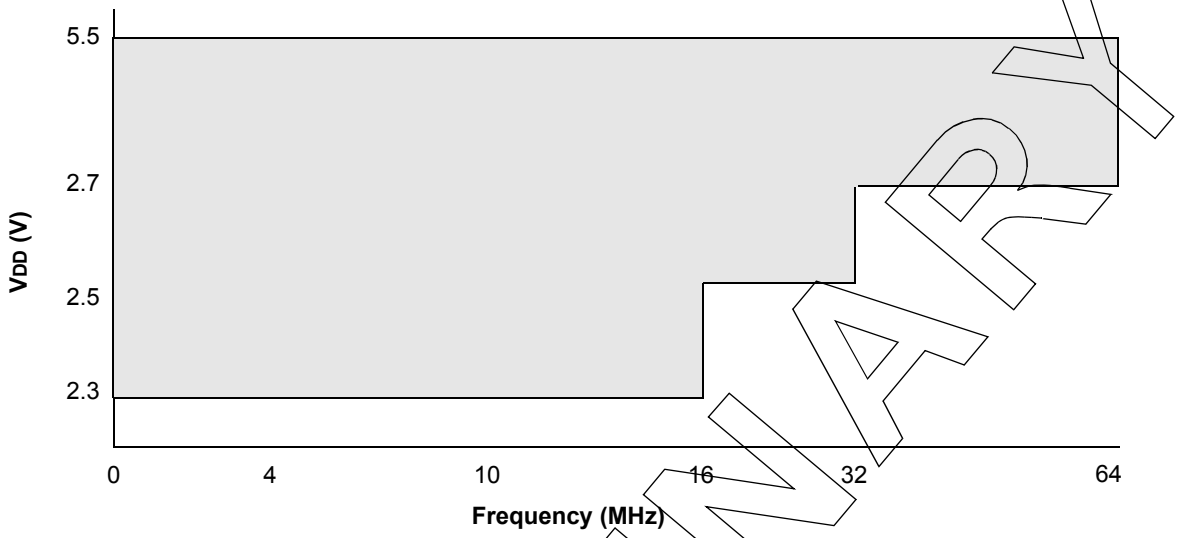
Q Cycle Activity:

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

Example: COMF REG, 0, 0

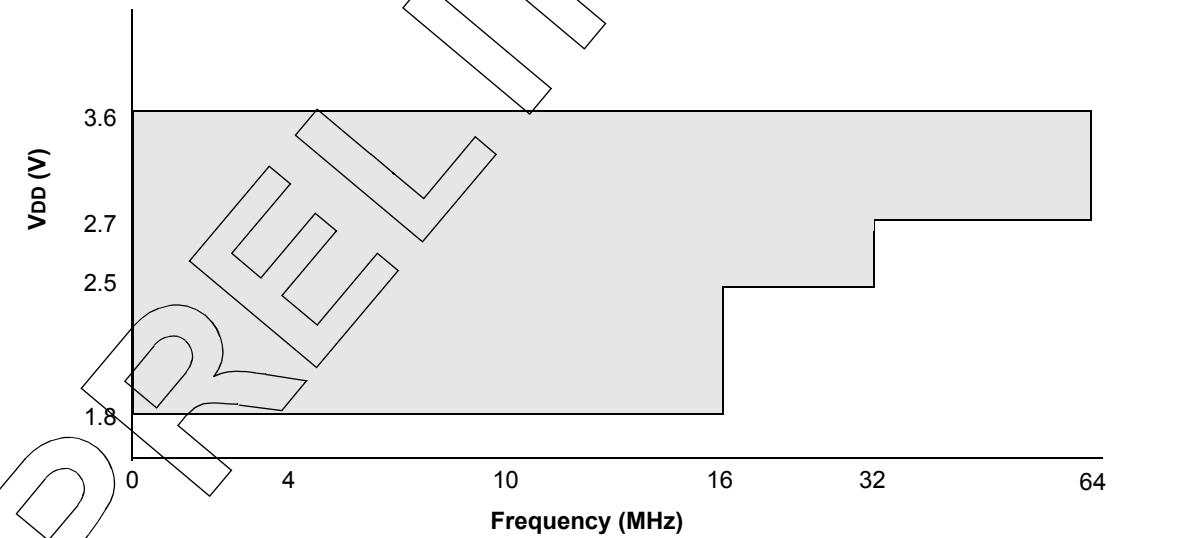
Before Instruction
REG = 13h
After Instruction
REG = 13h
W = ECh

**FIGURE 45-1: VOLTAGE FREQUENCY GRAPH, -40°C ≤ TA ≤ +125°C, PIC18F25/26K83 ONLY**



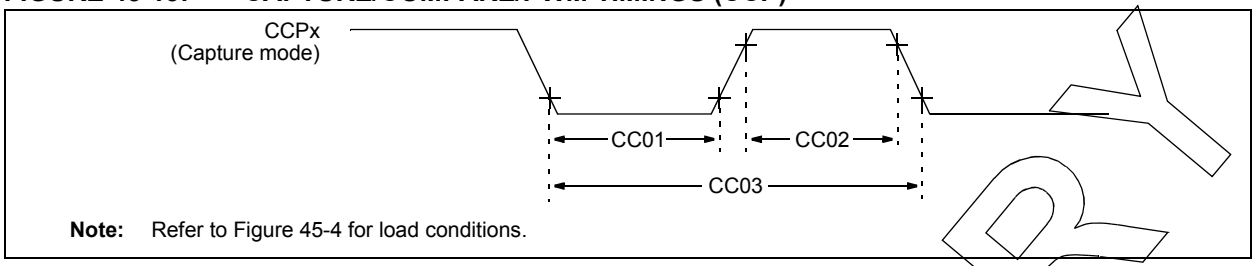

**Note 1:** The shaded region indicates the permissible combinations of voltage and frequency.
**2:** Refer to Table 45-7 for each Oscillator mode's supported frequencies.

**FIGURE 45-2: VOLTAGE FREQUENCY GRAPH, -40°C ≤ TA ≤ +125°C, PIC18LF25/26K83 ONLY**




**Note 1:** The shaded region indicates the permissible combinations of voltage and frequency.
**2:** Refer to Table 45-7 for each Oscillator mode's supported frequencies.

**FIGURE 45-13: CAPTURE/COMPARE/PWM TIMINGS (CCP)**




**Note:** Refer to Figure 45-4 for load conditions.

**TABLE 45-20: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP)**

| **Standard Operating Conditions (unless otherwise stated)**<br>Operating Temperature -40°C ≤ TA ≤ +125°C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | | **Min.** | **Typ†** | **Max.** | **Units** | **Conditions** |
| CC01* | TccL | CCPx Input Low Time | No Prescaler | 0.5TCY + 20 | — | — | ns | |
| | | | With Prescaler | 20 | — | — | ns | |
| CC02* | TccH | CCPx Input High Time | No Prescaler | 0.5TCY + 20 | — | — | ns | |
| | | | With Prescaler | 20 | — | — | ns | |
| CC03* | TccP | CCPx Input Period | | 3TCY + 40<br>N | — | — | ns | N = prescale value |

* These parameters are characterized but not tested.

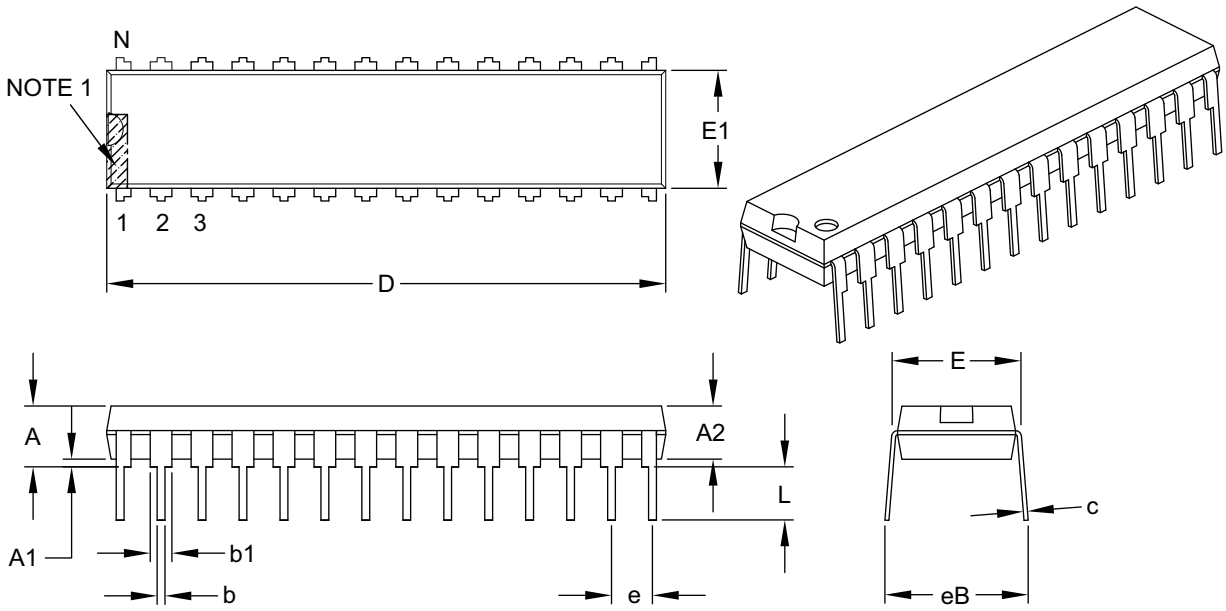
† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

## 47.1 Package Details

The following sections give the technical details of the packages.

### 28-Lead Skinny Plastic Dual In-Line (SP) – 300 mil Body [SPDIP]

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




| Units | | INCHES | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 28 | |
| Pitch | e | | .100 BSC | |
| Top to Seating Plane | A | – | – | .200 |
| Molded Package Thickness | A2 | .120 | .135 | .150 |
| Base to Seating Plane | A1 | .015 | – | – |
| Shoulder to Shoulder Width | E | .290 | .310 | .335 |
| Molded Package Width | E1 | .240 | .285 | .295 |
| Overall Length | D | 1.345 | 1.365 | 1.400 |
| Tip to Seating Plane | L | .110 | .130 | .150 |
| Lead Thickness | c | .008 | .010 | .015 |
| Upper Lead Width | b1 | .040 | .050 | .070 |
| Lower Lead Width | b | .014 | .018 | .022 |
| Overall Row Spacing § | eB | – | – | .430 |

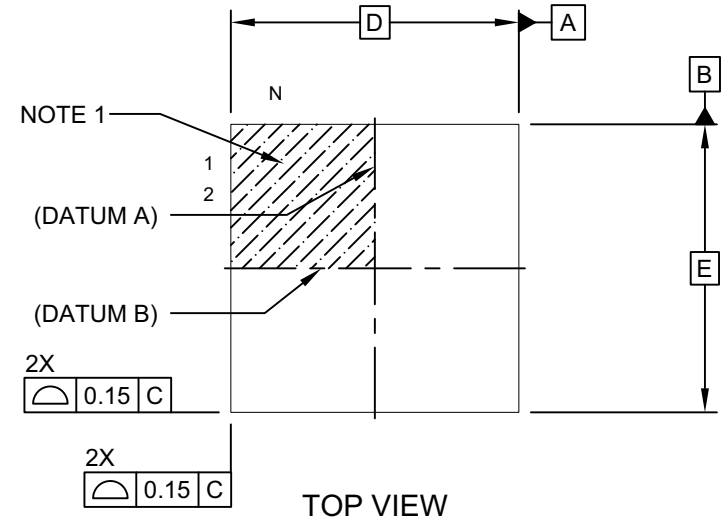
**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

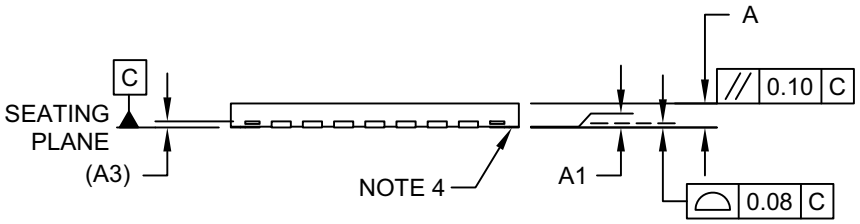
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-070B

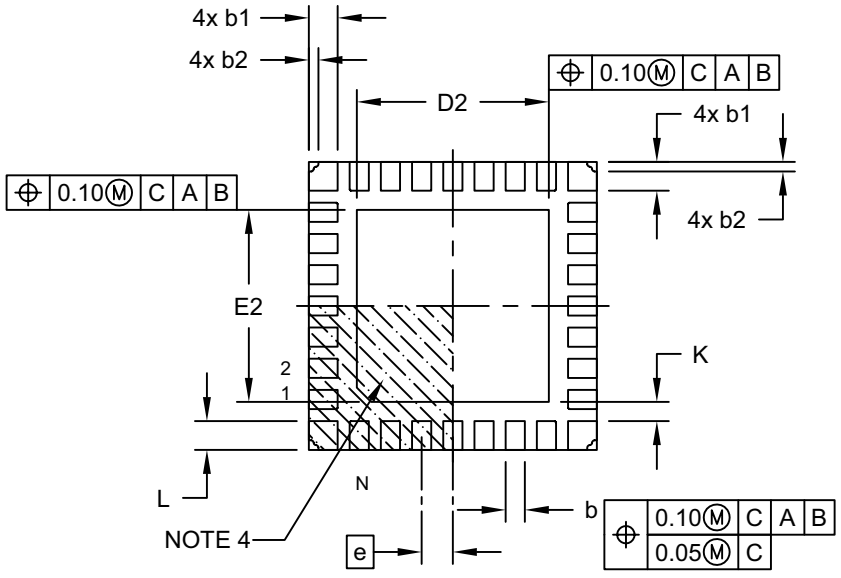
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




TOP VIEW




SIDE VIEW




BOTTOM VIEW

Microchip Technology Drawing C04-0209 Rev C Sheet 1 of 2

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.



*Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

**QUALITY MANAGEMENT SYSTEM**
**CERTIFIED BY DNV**
**═ ISO/TS 16949 ═**

**Trademarks**



ISBN: 978-1-5224-2047-7