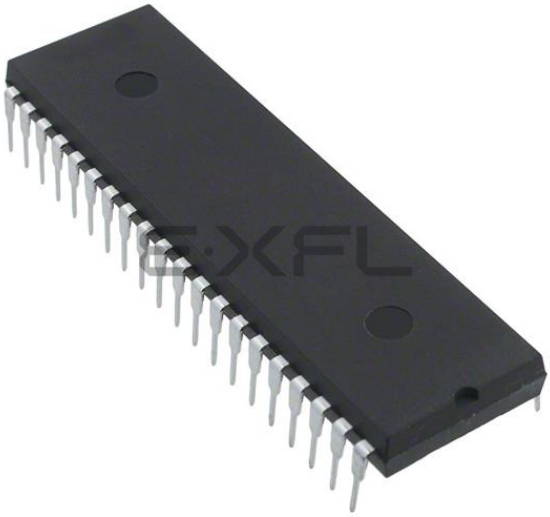


**Welcome to E-XFL.COM**

## 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 | 64KB (64K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | 1K x 8 |
| RAM Size | 4K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 3.6V |
| Data Converters | A/D 35x12b; D/A 1x5b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Through Hole |
| Package / Case | 40-DIP (0.600", 15.24mm) |
| Supplier Device Package | 40-PDIP |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic18lf46k42-i-p |

## 1.3 Register and Bit naming conventions

### 1.3.1 REGISTER NAMES

When there are multiple instances of the same peripheral in a device, the peripheral control registers will be depicted as the concatenation of a peripheral identifier, peripheral instance, and control identifier. The control registers section will show just one instance of all the register names with an 'x' in the place of the peripheral instance number. This naming convention may also be applied to peripherals when there is only one instance of that peripheral in the device to maintain compatibility with other devices in the family that contain more than one.

### 1.3.2 BIT NAMES

There are two variants for bit names:

- Short name: Bit function abbreviation
- Long name: Peripheral abbreviation + short name

#### 1.3.2.1 Short Bit Names

Short bit names are an abbreviation for the bit function. For example, some peripherals are enabled with the EN bit. The bit names shown in the registers are the short name variant.

Short bit names are useful when accessing bits in C programs. The general format for accessing bits by the short name is *RegisterName*bits.*ShortName*. For example, the enable bit, EN, in the T0CON0 register can be set in C programs with the instruction `T0CON0bits.EN = 1`.

Short names are generally not useful in assembly programs because the same name may be used by different peripherals in different bit positions. When this occurs, during the include file generation, all instances of that short bit name are appended with an underscore plus the name of the register in which the bit resides to avoid naming contentions.

#### 1.3.2.2 Long Bit Names

Long bit names are constructed by adding a peripheral abbreviation prefix to the short name. The prefix is unique to the peripheral thereby making every long bit name unique. The long bit name for the Timer0 enable bit is the Timer0 prefix, T0, appended with the enable bit short name, EN, resulting in the unique bit name T0EN.

Long bit names are useful in both C and assembly programs. For example, in C the T0CON0 enable bit can be set with the `T0EN = 1` instruction. In assembly, this bit can be set with the `BSF  T0CON0,T0EN` instruction.

#### 1.3.2.3 Bit Fields

Bit fields are two or more adjacent bits in the same register. For example, the four Least Significant bits of the T0CON0 register contain the output prescaler select bits. The short name for this field is OUTPS and the long name is T0OUTPS. Bit field access is only possible in C programs. The following example demonstrates a C program instruction for setting the Timer0 output prescaler to the 1:6 Postscaler:

```
T0CON0bits.OUTPS = 0x5;
```

Individual bits in a bit field can also be accessed with long and short bit names. Each bit is the field name appended with the number of the bit position within the field. For example, the Most Significant mode bit has the short bit name OUTPS3. The following two examples demonstrate assembly program sequences for setting the Timer0 output prescaler to 1:6 Postscaler:

Example 1:

```
MOVLW  ~(1<<OUTPS3 | 1<<OUTPS1)
ANDWF  T0CON0,F
MOVLW  1<<OUTPS2 | 1<<OUTPS0
IORWF  T0CON0,F
```

Example 2:

```
BCF    T0CON0,OUTPS3
BSF    T0CON0,OUTPS2
BCF    T0CON0,OUTPS1
BSF    T0CON0,OUTPS0
```

### 1.3.3 REGISTER AND BIT NAMING EXCEPTIONS

#### 1.3.3.1 Status, Interrupt, and Mirror Bits

Status, interrupt enables, interrupt flags, and mirror bits are contained in registers that span more than one peripheral. In these cases, the bit name shown is unique so there is no prefix or short name variant.

**TABLE 4-5: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 62**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3EFFh | ADCLK | 3EDFh | ADLTHH | 3EBFh | CM1PCH | 3E9Fh | — | 3E7Fh | — | 3E5Fh | — | 3E3Fh | — | 3E1Fh | — |
| 3EFEh | ADACT | 3EDEh | ADLTHL | 3EBEh | CM1NCH | 3E9Eh | DAC1CON0 | 3E7Eh | — | 3E5Eh | — | 3E3Eh | — | 3E1Eh | — |
| 3EFDh | ADREF | 3EDDh | — | 3EBDh | CM1CON1 | 3E9Dh | — | 3E7Dh | — | 3E5Dh | — | 3E3Dh | — | 3E1Dh | — |
| 3EFCh | ADSTAT | 3EDCh | — | 3EBCh | CM1CON0 | 3E9Ch | DAC1CON1 | 3E7Ch | — | 3E5Ch | — | 3E3Ch | — | 3E1Ch | — |
| 3EFBh | ADCON3 | 3EDBh | — | 3EBBh | CM2PCH | 3E9Bh | — | 3E7Bh | — | 3E5Bh | — | 3E3Bh | — | 3E1Bh | — |
| 3EFAh | ADCON2 | 3EDAh | — | 3EBAh | CM2NCH | 3E9Ah | — | 3E7Ah | — | 3E5Ah | — | 3E3Ah | — | 3E1Ah | — |
| 3EF9h | ADCON1 | 3ED9h | — | 3EB9h | CM2CON1 | 3E99h | — | 3E79h | — | 3E59h | — | 3E39h | — | 3E19h | — |
| 3EF8h | ADCON0 | 3ED8h | — | 3EB8h | CM2CON0 | 3E98h | — | 3E78h | — | 3E58h | — | 3E38h | — | 3E18h | — |
| 3EF7h | ADPREH | 3ED7h | ADCP | 3EB7h | — | 3E97h | — | 3E77h | — | 3E57h | — | 3E37h | — | 3E17h | — |
| 3EF6h | ADPREL | 3ED6h | — | 3EB6h | — | 3E96h | — | 3E76h | — | 3E56h | — | 3E36h | — | 3E16h | — |
| 3EF5h | ADCAP | 3ED5h | — | 3EB5h | — | 3E95h | — | 3E75h | — | 3E55h | — | 3E35h | — | 3E15h | — |
| 3EF4h | ADACQH | 3ED4h | — | 3EB4h | — | 3E94h | — | 3E74h | — | 3E54h | — | 3E34h | — | 3E14h | — |
| 3EF3h | ADACQL | 3ED3h | — | 3EB3h | — | 3E93h | — | 3E73h | — | 3E53h | — | 3E33h | — | 3E13h | — |
| 2EF2h | — | 3ED2h | — | 3EB2h | — | 3E92h | — | 3E72h | — | 3E52h | — | 3E32h | — | 3E12h | — |
| 3EF1h | ADPCH | 3ED1h | — | 3EB1h | — | 3E91h | — | 3E71h | — | 3E51h | — | 3E31h | — | 3E11h | — |
| 3EF0h | ADRESH | 3ED0h | — | 3EB0h | — | 3E90h | — | 3E70h | — | 3E50h | — | 3E30h | — | 3E10h | — |
| 3EEFh | ADRESL | 3ECFh | — | 3EAFh | — | 3E8Fh | — | 3E6Fh | — | 3E4Fh | — | 3E2Fh | — | 3E0Fh | — |
| 3EEEh | ADPREVH | 3ECEh | — | 3EAEh | — | 3E8Eh | — | 3E6Eh | — | 3E4Eh | — | 3E2Eh | — | 3E0Eh | — |
| 3EEDh | ADPREVL | 3ECDh | — | 3EADh | — | 3E8Dh | — | 3E6Dh | — | 3E4Dh | — | 3E2Dh | — | 3E0Dh | — |
| 3EECh | ADRPT | 3ECCh | — | 3EACh | — | 3E8Ch | — | 3E6Ch | — | 3E4Ch | — | 3E2Ch | — | 3E0Ch | — |
| 3EEBh | ADCNT | 3ECBh | — | 3EABh | — | 3E8Bh | — | 3E6Bh | — | 3E4Bh | — | 3E2Bh | — | 3E0Bh | — |
| 3EEAh | ADACCU | 3ECAh | HLVDCON1 | 3EAAh | — | 3E8Ah | — | 3E6Ah | — | 3E4Ah | — | 3E2Ah | — | 3E0Ah | — |
| 3EE9h | ADACCH | 3EC9h | HLVDCON0 | 3EA9h | — | 3E89h | — | 3E69h | — | 3E49h | — | 3E29h | — | 3E09h | — |
| 3EE8h | ADACCL | 3EC8h | — | 3EA8h | — | 3E88h | — | 3E68h | — | 3E48h | — | 3E28h | — | 3E08h | — |
| 3EE7h | ADFLTRH | 3EC7h | — | 3EA7h | — | 3E87h | — | 3E67h | — | 3E47h | — | 3E27h | — | 3E07h | — |
| 3EE6h | ADFLTRL | 3EC6h | — | 3EA6h | — | 3E86h | — | 3E66h | — | 3E46h | — | 3E26h | — | 3E06h | — |
| 3EE5h | ADSTPTH | 3EC5h | — | 3EA5h | — | 3E85h | — | 3E65h | — | 3E45h | — | 3E25h | — | 3E05h | — |
| 3EE4h | ADSTPTL | 3EC4h | — | 3EA4h | — | 3E84h | — | 3E64h | — | 3E44h | — | 3E24h | — | 3E04h | — |
| 3EE3h | ADERRH | 3EC3h | ZCDCON | 3EA3h | — | 3E83h | — | 3E63h | — | 3E43h | — | 3E23h | — | 3E03h | — |
| 3EE2h | ADERRL | 3EC2h | — | 3EA2h | — | 3E82h | — | 3E62h | — | 3E42h | — | 3E22h | — | 3E02h | — |
| 3EE1h | ADUTHH | 3EC1h | FVRCON | 3EA1h | — | 3E81h | — | 3E61h | — | 3E41h | — | 3E21h | — | 3E01h | — |
| 3EE0h | ADUTHL | 3EC0h | CMOUT | 3EA0h | — | 3E80h | — | 3E60h | — | 3E40h | — | 3E20h | — | 3E00h | — |

**Legend:** Unimplemented data memory locations and registers, read as '0'.

**Note 1:** Unimplemented in LF devices.

**2:** Unimplemented in PIC18(L)F26/27K42.

**3:** Unimplemented in PIC18(L)F26/27/45/46/47K42.

**TABLE 9-3: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| INTCON0 | GIE/GIEH | GIEL | IPEN | — | — | INT2EDG | INT1EDG | INT0EDG | 135 |
| INTCON1 | STAT<1:0> | | — | — | — | — | — | — | 136 |
| PIE0 | IOCIE | CRCIE | SCANIE | NVMIE | CSWIE | OSFIE | HLVDIE | SWIE | 147 |
| PIE1 | SMT1PWAIE | SMT1PRAIE | SMT1IE | C1IE | ADTIE | ADIE | ZCDIE | INT0IE | 148 |
| PIE2 | I2C1RXIE | SPI1IE | SPI1TXIE | SPI1RXIE | DMA1AIE | DMA1ORIE | DMA1DCNTIE | DMA1SCNTIE | 149 |
| PIE3 | TMR0IE | U1IE | U1EIE | U1TXIE | U1RXIE | I2C1EIE | I2C1IE | I2C1TXIE | 150 |
| PIE4 | CLC1IE | CWG1IE | NCO1IE | — | CCP1IE | TMR2IE | TMR1GIE | TMR1IE | 151 |
| PIE5 | I2C2TXIE | I2C2RXIE | DMA2AIE | DMA2ORIE | DMA2DCNTIE | DMA2SCNTIE | C2IE | INT1IE | 152 |
| PIE6 | TMR3GIE | TMR3IE | U2IE | U2EIE | U2TXIE | U2RXIE | I2C2EIE | I2C2IE | 153 |
| PIE7 | — | — | INT2IE | CLC2IE | CWG2IE | — | CCP2IE | TMR4IE | 154 |
| PIE8 | TMR5GIE | TMR5IE | — | — | — | — | — | — | 155 |
| PIE9 | — | — | — | — | CLC3IE | CWG3IE | CCP3IE | TMR6IE | 155 |
| PIE10 | — | — | — | — | — | — | CLC4IE | CCP4IE | 156 |
| PIR0 | IOCIF | CRCIF | SCANIF | NVMIF | CSWIF | OSFIF | HLVDIF | SWIF | 137 |
| PIR1 | SMT1PWAIF | SMT1PRAIF | SMT1IF | C1IF | ADTIF | ADIF | ZCDIF | INT0IF | 138 |
| PIR2 | I2C1RXIF | SPI1IF | SPI1TXIF | SPI1RXIF | DMA1AIF | DMA1ORIF | DMA1DCNTIF | DMA1SCNTIF | 139 |
| PIR3 | TMR0IF | U1IF | U1EIF | U1TXIF | U1RXIF | I2C1EIF | I2C1IF | I2C1TXIF | 140 |
| PIR4 | CLC1IF | CWG1IF | NCO1IF | — | CCP1IF | TMR2IF | TMR1GIF | TMR1IF | 141 |
| PIR5 | I2C2TXF | I2C2RXF | DMA2AIF | DMA2ORIF | DMA2DCNTIF | DMA2SCNTIF | C2IF | INT1IF | 142 |
| PIR6 | TMR3GIF | TMR3IF | U2IF | U2EIF | U2TXIF | U2RXIF | I2C2EIF | I2C2IF | 143 |
| PIR7 | — | — | INT2IF | CLC2IF | CWG2IF | — | CCP2IF | TMR4IF | 144 |
| PIR8 | TMR5GIF | TMR5IF | — | — | — | — | — | — | 145 |
| PIR9 | — | — | — | — | CLC3IF | CWG3IF | CCP3IF | TMR6IF | 145 |
| PIR10 | — | — | — | — | — | — | CLC4IF | CCP4IF | 146 |
| IPR0 | IOCIP | CRCIP | SCANIP | NVMIP | CSWIP | OSFIP | HLVDIP | SWIP | 157 |
| IPR1 | SMT1PWAIP | SMT1PRAIP | SMT1IP | C1IP | ADTIP | ADIP | ZCDIP | INT0IP | 158 |
| IPR2 | I2C1RIP | SPI1IP | SPI1TIP | SPI1RIP | DMA1AIP | DMA1ORIP | DMA1DCNTIP | DMA1SCNTIP | 159 |
| IPR3 | TMR0IP | U1IP | U1EIP | U1TXIP | U1RXIP | I2C1EIP | I2C1IP | I2C1TXIP | 160 |
| IPR4 | CLC1IP | CWG1IP | NCO1IP | — | CCP1IP | TMR2IP | TMR1GIP | TMR1IP | 161 |
| IPR5 | I2C2TXP | I2C2RXP | DMA2AIP | DMA2ORIP | DMA2DCNTIP | DMA2SCNTIP | C2IP | INT1IP | 162 |
| IPR6 | TMR3GIP | TMR3IP | U2IP | U2EIP | U2TXIP | U2RXIP | I2C2EIP | I2C2IP | 163 |
| IPR7 | — | — | INT2IP | CLC2IP | CWG2IP | — | CCP2IP | TMR4IP | 164 |
| IPR8 | TMR5GIP | TMR5IP | — | — | — | — | — | — | 164 |
| IPR9 | — | — | — | — | CLC3IP | CWG3IP | CCP3IP | TMR6IP | 165 |
| IPR10 | — | — | — | — | — | — | CLC4IP | CCP4IP | 165 |
| IVTBASEU | — | — | — | BASE<20:16> | | | | | 166 |
| IVTBASEH | BASE<15:8> | | | | | | | | 166 |
| IVTBASEL | BASE<7:0> | | | | | | | | 166 |
| IVTADU | | | | AD<20:16> | | | | | 167 |
| IVTADH | AD<15:8> | | | | | | | | 167 |
| IVTADL | AD<7:0> | | | | | | | | 167 |
| IVTLOCK | — | — | — | — | — | — | — | IVTLOCKED | 168 |

**Legend:** — = unimplemented locations, read as '0'. Shaded bits are not used for interrupts.

**FIGURE 15-3: DMA COUNTERS BLOCK DIAGRAM**

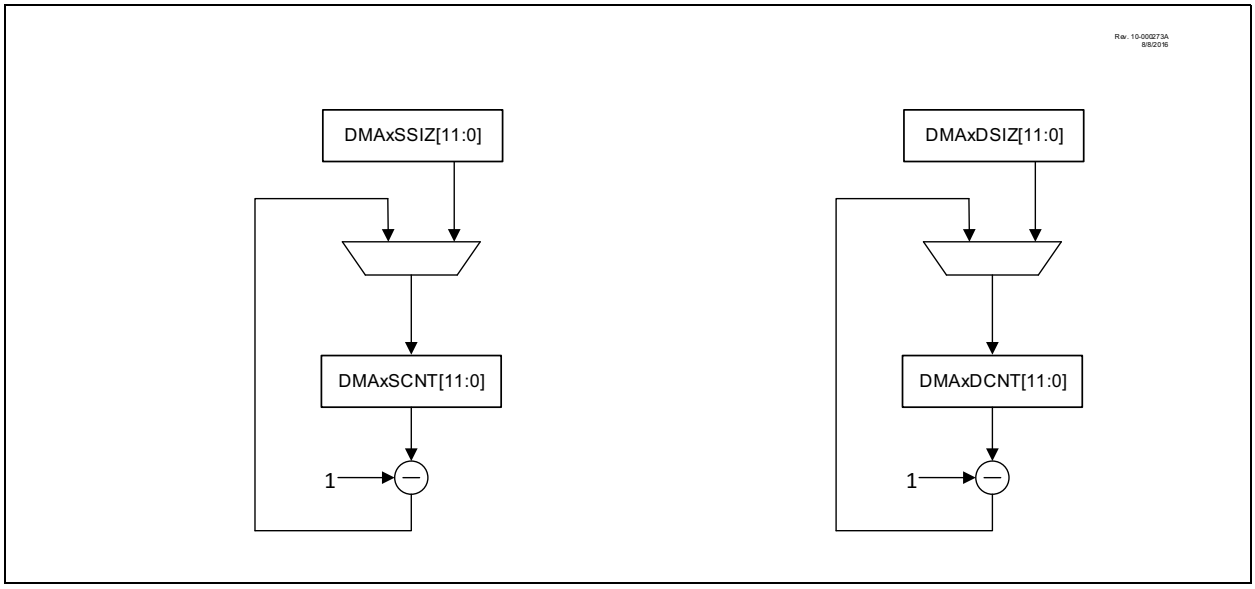



Table 15-2 has a few examples of configuring DMA Message sizes.

**TABLE 15-2: EXAMPLE MESSAGE SIZE TABLE**

| Operation | Example | SCNT | DCNT | Comments |
|---|---|---|---|---|
| Read from single SFR location to RAM | U1RXB | 1 | N | N equals the number of bytes desired in the destination buffer. N >= 1. |
| Write to single SFR location from RAM | U1TXB | N | 1 | N equals the number of bytes desired in the source buffer. N >= 1. |
| Read from multiple SFR location | ADRES[H:L] | 2 | 2*N | N equals the number of ADC results to be stored in memory. N>= 1 |
| | TMR1[H:L] | 2 | 2*N | N equals the number of TMR1 Acquisition results to be stored in memory. N>= 1 |
| | SMT1CPR[U:H:L] | 3 | 3*N | N equals the number of Capture Pulse Width measurements to be stored in memory. N>= 1 |
| Write to Multiple SFR registers | PWMDC[H:L] | 2*N | 2 | N equals the number of PWM duty cycle values to be loaded from a memory table. N>= 1 |
| | All ADC registers | N*31 | 31 | Using the DMA to transfer a complete ADC context from RAM to the ADC registers.N>= 1 |

**REGISTER 19-5: PMD4: PMD CONTROL REGISTER 4**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| CWG3MD | CWG2MD | CWG1MD | — | — | — | — | — |
| 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 |

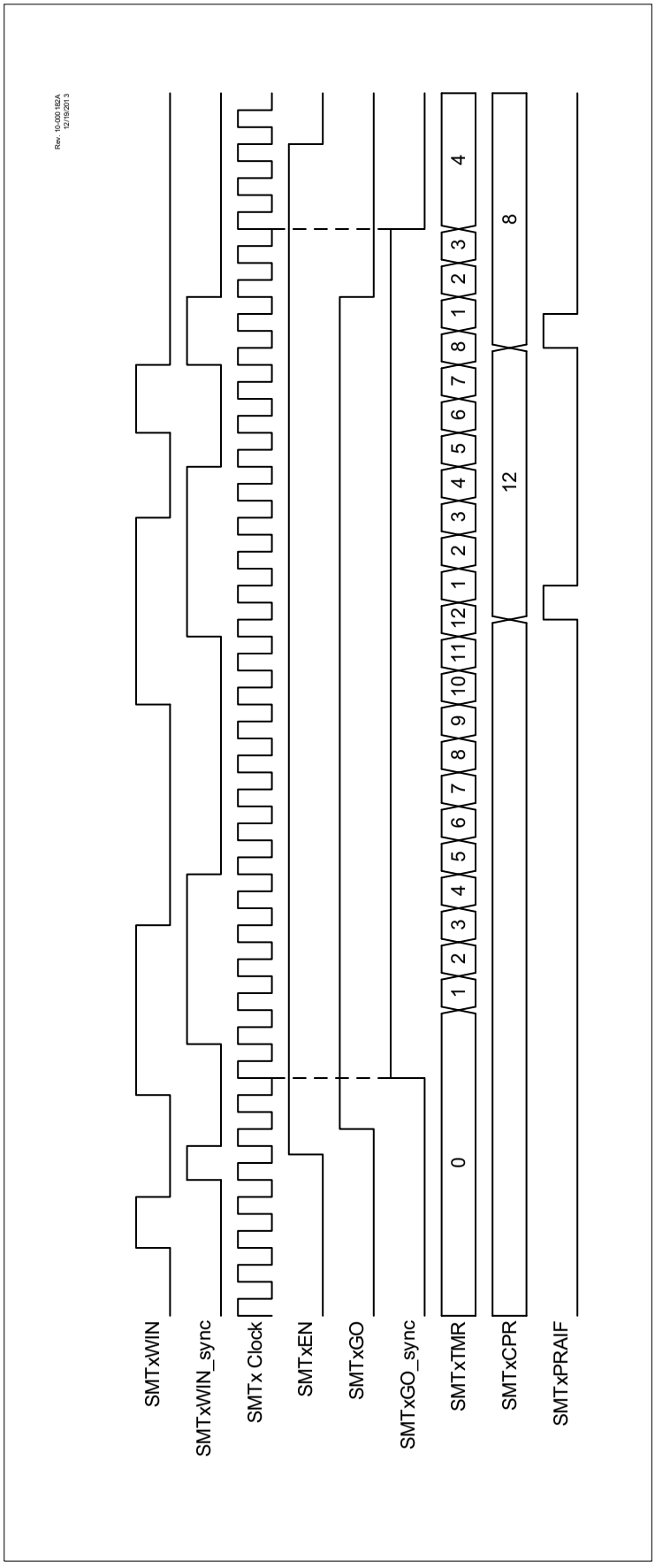
bit 7 **CWG3MD:** Disable CWG3 Module bit
1 = CWG3 module disabled
0 = CWG3 module enabled

bit 6 **CWG2MD:** Disable CWG2 Module bit
1 = CWG2 module disabled
0 = CWG2 module enabled

bit 5 **CWG1MD:** Disable CWG1 Module bit
1 = CWG1 module disabled
0 = CWG1 module enabled

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

**FIGURE 25-10: WINDOWED MEASURE MODE REPEAT ACQUISITION TIMING DIAGRAM**



## 25.8 Register Definitions: SMT Control

Long bit name prefixes for the Signal Measurement Timer peripherals are shown in **Section 1.3 "Register and Bit naming conventions"**.

**TABLE 25-2: LONG BIT NAMES PREFIXES FOR SMT PERIPHERALS**

| Peripheral | Bit Name Prefix |
|---|---|
| SMT1 | SMT1 |

**REGISTER 25-1: SMT1CON0: SMT CONTROL REGISTER 0**

| R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| EN(1) | — | STP | WPOL | SPOL | CPOL | PS<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 **EN:** SMT Enable bit(1)
1 = SMT is enabled
0 = SMT is disabled; internal states are reset, clock requests are disabled

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

bit 5 **STP:** SMT Counter Halt Enable bit
When SMT1TMR = SMT1PR:
1 = Counter remains SMT1PR; period match interrupt occurs when clocked
0 = Counter resets to 24'h000000; period match interrupt occurs when clocked

bit 4 **WPOL:** SMT1WIN Input Polarity Control bit
1 = SMT1WIN signal is active-low/falling edge enabled
0 = SMT1WIN signal is active-high/rising edge enabled

bit 3 **SPOL:** SMT1SIG Input Polarity Control bit
1 = SMT1_signal is active-low/falling edge enabled
0 = SMT1_signal is active-high/rising edge enabled

bit 2 **CPOL:** SMT Clock Input Polarity Control bit
1 = SMT1TMR increments on the falling edge of the selected clock signal
0 = SMT1TMR increments on the rising edge of the selected clock signal

bit 1-0 **PS<1:0>:** SMT Prescale Select bits
11 = Prescaler = 1:8
10 = Prescaler = 1:4
01 = Prescaler = 1:2
00 = Prescaler = 1:1

**Note 1:** Setting EN to '0' does not affect the register contents.

**REGISTER 27-9: CLCxGLS2: GATE 2 LOGIC SELECT REGISTER**

| 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 |
|---|---|---|---|---|---|---|---|
| G3D4T | G3D4N | G3D3T | G3D3N | G3D2T | G3D2N | G3D1T | G3D1N |
| 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 **G3D4T:** Gate 2 Data 4 True (noninverted) bit
1 = CLCIN3 (true) is gated into CLCx Gate 2
0 = CLCIN3 (true) is not gated into CLCx Gate 2

bit 6 **G3D4N:** Gate 2 Data 4 Negated (inverted) bit
1 = CLCIN3 (inverted) is gated into CLCx Gate 2
0 = CLCIN3 (inverted) is not gated into CLCx Gate 2

bit 5 **G3D3T:** Gate 2 Data 3 True (noninverted) bit
1 = CLCIN2 (true) is gated into CLCx Gate 2
0 = CLCIN2 (true) is not gated into CLCx Gate 2

bit 4 **G3D3N:** Gate 2 Data 3 Negated (inverted) bit
1 = CLCIN2 (inverted) is gated into CLCx Gate 2
0 = CLCIN2 (inverted) is not gated into CLCx Gate 2

bit 3 **G3D2T:** Gate 2 Data 2 True (noninverted) bit
1 = CLCIN1 (true) is gated into CLCx Gate 2
0 = CLCIN1 (true) is not gated into CLCx Gate 2

bit 2 **G3D2N:** Gate 2 Data 2 Negated (inverted) bit
1 = CLCIN1 (inverted) is gated into CLCx Gate 2
0 = CLCIN1 (inverted) is not gated into CLCx Gate 2

bit 1 **G3D1T:** Gate 2 Data 1 True (noninverted) bit
1 = CLCIN0 (true) is gated into CLCx Gate 2
0 = CLCIN0 (true) is not gated into CLCx Gate 2

bit 0 **G3D1N:** Gate 2 Data 1 Negated (inverted) bit
1 = CLCIN0 (inverted) is gated into CLCx Gate 2
0 = CLCIN0 (inverted) is not gated into CLCx Gate 2

## 28.1 NCO Operation

The NCO operates by repeatedly adding a fixed value to an accumulator. Additions occur at the input clock rate. The accumulator will overflow with a carry periodically, which is the raw NCO output (NCO_overflow). This effectively reduces the input clock by the ratio of the addition value to the maximum accumulator value. See Equation 28-1.

The NCO output can be further modified by stretching the pulse or toggling a flip-flop. The modified NCO output is then distributed internally to other peripherals and can be optionally output to a pin. The accumulator overflow also generates an interrupt (NCO_overflow).

The NCO period changes in discrete steps to create an average frequency. This output depends on the ability of the receiving circuit (i.e., CWG or external resonant converter circuitry) to average the NCO output to reduce uncertainty.

**EQUATION 28-1: NCO OVERFLOW FREQUENCY**

$$F_{OVERFLOW} = \frac{NCO\ Clock\ Frequency \times Increment\ Value}{2^{20}}$$

### 28.1.1 NCO CLOCK SOURCES

Clock sources available to the NCO include:

- FOSC
- HFINTOSC
- LFINTOSC
- MFINTOSC/4 (32 kHz)
- MFINTOSC (500 kHz)
- CLC1/2/3/4_out
- CLKREF
- SOSC

The NCO clock source is selected by configuring the N1CKS<2:0> bits in the NCO1CLK register.

### 28.1.2 ACCUMULATOR

The accumulator is a 20-bit register. Read and write access to the accumulator is available through three registers:

- NCO1ACCL
- NCO1ACCH
- NCO1ACCU

### 28.1.3 ADDER

The NCO Adder is a full adder, which operates independently from the source clock. The addition of the previous result and the increment value replaces the accumulator value on the rising edge of each input clock.

### 28.1.4 INCREMENT REGISTERS

The increment value is stored in three registers making up a 20-bit incrementer. In order of LSB to MSB they are:

- NCO1INCL
- NCO1INCH
- NCO1INCU

When the NCO module is enabled, the NCO1INCU and NCO1INCH registers should be written first, then the NCO1INCL register. Writing to the NCO1INCL register initiates the increment buffer registers to be loaded simultaneously on the second rising edge of the NCO_clk signal.

The registers are readable and writable. The increment registers are double-buffered to allow value changes to be made without first disabling the NCO module.

When the NCO module is disabled, the increment buffers are loaded immediately after a write to the increment registers.

**Note:** The increment buffer registers are not user-accessible.

#### 32.8.3.4 Receiver Overflow and Transmitter Underflow Interrupts

The receiver overflow interrupt triggers if data is received when the RXFIFO is already full and RXR = 1. In this case, the data will be discarded and the RXOIF bit will be set. The receiver overflow interrupt flag is the RXOIF bit of SPIxINTF. The receiver overflow interrupt enable bit is the RXOIE bit of SPIxINTE.

The Transmitter Underflow interrupt flag triggers if a data transfer begins when the TXFIFO is empty and TXR = 1. In this case, the most recently received data will be transmitted and the TXUIF bit will be set. The transmitter underflow interrupt flag is the TXUIF bit of SPIxINTF. The transmitter underflow interrupt enable bit is the TXUIE bit of SPIxINTE.

Both of these interrupts will only occur in Slave mode, as Master mode will not allow the RXFIFO to overflow or the TXFIFO to underflow.

**REGISTER 32-4: SPIxTCNTH: SPI TRANSFER COUNTER MSB REGISTER**

| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | TCNT10 | TCNT9 | TCNT8 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |

bit 7-3 **Unimplemented**: Read as '0'

bit 2-0 TCNT<10:8>:

BMODE = 0

Bits 13-11 of the Transfer Counter, counting the total number of bits to transfer

BMODE = 1

Bits 10-8 of the Transfer Counter, counting the total number of bytes to transfer

**Note:** This register should not be written to while a transfer is in progress (BUSY bit of SPIxCON2 is set).

**REGISTER 32-5: SPIxTWIDTH: SPI TRANSFER WIDTH REGISTER**

| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | TWIDTH2 | TWIDTH1 | TWIDTH0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |

bit 7-3 **Unimplemented**: Read as '0'

bit 2-0 TWIDTH<2:0>:

BMODE = 0

Bits 2-0 of the Transfer Counter, counting the total number of bits to transfer

BMODE = 1

Size (in bits) of each transfer counted by the transfer counter

111 = 7 bits

110 = 6 bits

101 = 5 bits

100 = 4 bits

011 = 3 bits

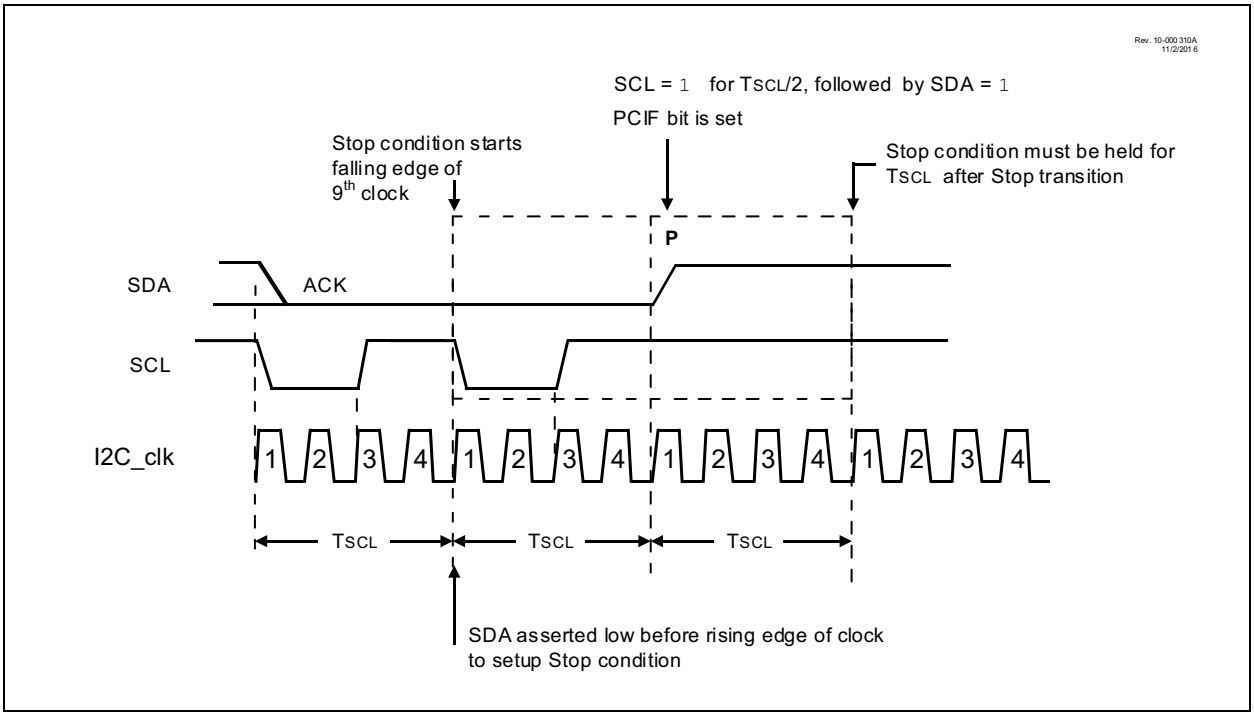
010 = 2 bits

001 = 1 bit

000 = 8 bits

**Note:** This register should not be written to while a transfer is in progress (BUSY bit of SPIxCON2 is set).

**FIGURE 33-18: STOP CONDITION DURING RECEIVE OR TRANSMIT**




### 33.5.9 MASTER TRANSMISSION IN 7-BIT ADDRESSING MODE

This section describes the sequence of events for the $I^2C$ module configured as an $I^2C$ master in 7-bit Addressing mode and is transmitting data. Figure 33-19 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, slave address in I2CxADB1 with R/W = 0 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 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 start and address.
3. If the transmit buffer is empty (i.e., TXBE = 1) and I2CxCNT!= 0, the I2CxTXIF and MDR bits are set and the clock is stretched on the 8th falling SCL edge. Clock can be started by loading the next data byte in I2CxTXB register.
4. Master sends out the 9th SCL pulse for ACK.
5. If the Master hardware receives ACK from Slave device, it loads the next byte from the transmit buffer (I2CxTXB) into the shift register and the value of I2CxCNT register is decremented.
6. If a NACK was received, Master hardware asserts Stop or Restart
7. If ABD = 0; i.e., Address buffers are enabled

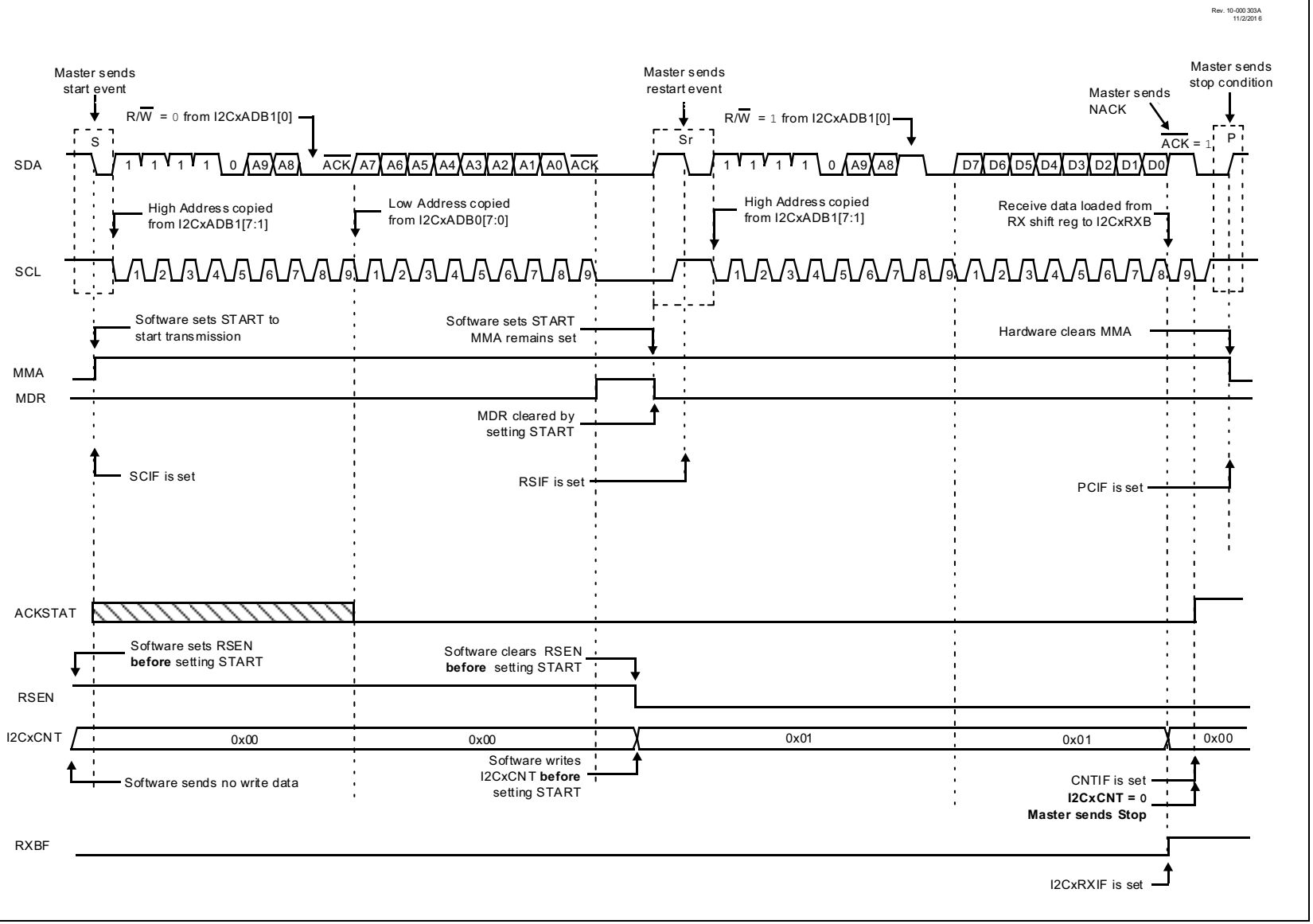
If I2CxCNT = 0, Master hardware sends Stop or sets MDR if RSEN = 1 and waits for the software to set the Start bit again to issue a restart condition.

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

If I2CxCNT = 0, Master hardware sends Stop or sets MDR if RSEN = 1 and waits for the software to write the new address to the I2CxTXB register. Software writes to the S bit are ignored in this case.

8. Master hardware outputs data on SDA.
9. If TXBE = 1 and I2CxCNT! = 0, I2CxTXIF and MDR bits are set and the clock is stretched on 8th falling SCL edge. The user can release the clock by writing the next data byte to I2CxTXB register.
10. Master hardware clocks in ACK from Slave, and loads the next data byte from I2CTXB to the shift register. The value of I2CxCNT is decremented.
11. Go to step 7.

**FIGURE 33-22: I²C MASTER, 10-BIT ADDRESS, RECEPTION (USING RSTEN BIT)**



**RETURN** **Return from Subroutine**

Syntax: RETURN {s}

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
if s = 1
(WS) → W,
(STATUSS) → Status,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0001 | 001s |

Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If 's'= 1, the contents of the shadow registers, WS, STATUSS and BSRS, are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | POP PC from stack |
| No operation | No operation | No operation | No operation |

Example: RETURN

After Instruction:
PC = TOS

**RLCF** **Rotate Left f through Carry**

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

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

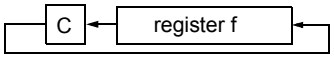
Operation: (f<n>) → dest<n + 1>,
(f<7>) → C,
(C) → dest<0>

Status Affected: C, N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 01da | ffff | ffff |

Description: The contents of register 'f' are rotated one bit to the left through the CARRY flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 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: RLCF REG, 0, 0

Before Instruction
REG = 1110 0110
C = 0
After Instruction
REG = 1110 0110
W = 1100 1100
C = 1

**MOVSS** **Move Indexed to Indexed**

Syntax: MOVSS [$z_s$], [$z_d$]

Operands: $0 \le z_s \le 127$
$0 \le z_d \le 127$

Operation: ((FSR2) + $z_s$) → ((FSR2) + $z_d$)

Status Affected: None

Encoding:

| | | | | |
|---|---|---|---|---|
| 1st word (source) | 1110 | 1011 | 1zzz | $zzzz_s$ |
| 2nd word (dest.) | 1111 | xxxx | xzzz | $zzzz_d$ |

Description: The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets '$z_s$' or '$z_d$', respectively, to the value of FSR2. Both registers can be located anywhere in the 4096-byte data memory space (000h to FFFh).
The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.
If the resultant source address points to an indirect addressing register, the value returned will be 00h. If the resultant destination address points to an indirect addressing register, the instruction will execute as a NOP.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Determine source addr | Determine source addr | Read source reg |
| Decode | Determine dest addr | Determine dest addr | Write to dest reg |

Example: MOVSS [05h], [06h]

Before Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 11h
After Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 33h

**PUSHL** **Store Literal at FSR2, Decrement FSR2**

Syntax: PUSHL k

Operands: $0 \le k \le 255$

Operation: k → (FSR2),
FSR2 – 1 → FSR2

Status Affected: None

Encoding:

| 1111 | 1010 | kkkk | kkkk |
|---|---|---|---|

Description: The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation. This instruction allows users to push values onto a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read 'k' | Process data | Write to destination |

Example: PUSHL 08h

Before Instruction
FSR2H:FSR2L = 01ECh
Memory (01ECh) = 00h

After Instruction
FSR2H:FSR2L = 01EBh
Memory (01ECh) = 08h

## 42.0 REGISTER SUMMARY

### 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 |
|---|---|---|---|---|---|---|---|---|---|---|
| 3FFFh | TOSU | — | — | — | Top of Stack Upper byte | | | | | 37 |
| 3FFEh | TOSH | Top of Stack High byte | | | | | | | | 37 |
| 3FFDh | TOSL | Top of Stack Low byte | | | | | | | | 37 |
| 3FFCh | STKPTR | — | — | — | Stack Pointer | | | | | 39 |
| 3FFBh | PCLATU | — | — | — | Holding Register for PC Upper byte | | | | | 36 |
| 3FFAh | PCLATH | Holding Register for PC High byte | | | | | | | | 36 |
| 3FF9h | PCL | PC Low byte | | | | | | | | 36 |
| 3FF8h | TBLPTRU | — | — | Program Memory Table Pointer Upper byte | | | | | | 192 |
| 3FF7h | TBLPTRH | Program Memory Table Pointer High byte | | | | | | | | 192 |
| 3FF6h | TBLPTRL | Program Memory Table Pointer Low byte | | | | | | | | 192 |
| 3FF5h | TABLAT | Table Latch | | | | | | | | 192 |
| 3FF4h | PRODH | Product Register High byte | | | | | | | | 187 |
| 3FF3h | PRODL | Product Register Low byte | | | | | | | | 187 |
| 3FF2h | — | Unimplemented | | | | | | | | |
| 3FF1h | PCON1 | — | — | — | — | — | — | $\overline{\text{MEMV}}$ | — | 91 |
| 3FF0h | PCON0 | STKOVF | STKUNF | $\overline{\text{WDTWV}}$ | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | 90 |
| 3FEFh | INDF0 | Uses contents of FSR0 to address data memory – value of FSR0 not changed | | | | | | | | 60 |
| 3FEEh | POSTINC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-incremented | | | | | | | | 61 |
| 3FEDh | POSTDEC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-decremented | | | | | | | | 61 |
| 3FECh | PREINC0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented | | | | | | | | 61 |
| 3FEBh | PLUSW0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented – value of FSR0 offset by W | | | | | | | | 61 |
| 3FEAh | FSR0H | — | — | Indirect Data Memory Address Pointer 0 High | | | | | | 61 |
| 3FE9h | FSR0L | Indirect Data Memory Address Pointer 0 Low | | | | | | | | 61 |
| 3FE8h | WREG | Working Register | | | | | | | | |
| 3FE7h | INDF1 | Uses contents of FSR1 to address data memory – value of FSR1 not changed | | | | | | | | 61 |
| 3FE6h | POSTINC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-incremented | | | | | | | | 61 |
| 3FE5h | POSTDEC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-decremented | | | | | | | | 61 |
| 3FE4h | PREINC1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented | | | | | | | | 61 |
| 3FE3h | PLUSW1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented – value of FSR1 offset by W | | | | | | | | 61 |
| 3FE2h | FSR1H | — | — | Indirect Data Memory Address Pointer 1 High | | | | | | 61 |
| 3FE1h | FSR1L | Indirect Data Memory Address Pointer 1 Low | | | | | | | | 61 |
| 3FE0h | BSR | — | — | Bank Select Register | | | | | | 44 |
| 3FDFh | INDF2 | Uses contents of FSR2 to address data memory – value of FSR2 not changed | | | | | | | | 61 |
| 3FDEh | POSTINC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-incremented | | | | | | | | 61 |
| 3FDDh | POSTDEC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-decremented | | | | | | | | 61 |
| 3FDCh | PREINC2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented | | | | | | | | 61 |
| 3FDBh | PLUSW2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented – value of FSR2 offset by W | | | | | | | | 61 |
| 3FDAh | FSR2H | — | — | Indirect Data Memory Address Pointer 2 High | | | | | | 61 |
| 3FD9h | FSR2L | Indirect Data Memory Address Pointer 2 Low | | | | | | | | 61 |
| 3FD8h | STATUS | — | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | N | OV | Z | DC | C | 58 |
| 3FD7h | IVTBASEU | — | — | — | BASE20 | BASE19 | BASE18 | BASE17 | BASE16 | 166 |
| 3FD6h | IVTBASEH | BASE15 | BASE14 | BASE13 | BASE12 | BASE11 | BASE10 | BASE9 | BASE8 | 166 |
| 3FD5h | IVTBASEL | BASE7 | BASE6 | BASE5 | BASE4 | BASE3 | BASE2 | BASE1 | BASE0 | 166 |
| 3FD4h | IVTLOCK | — | — | — | — | — | — | — | IVTLOCKED | 168 |
| 3FD3h | INTCON1 | STAT | | — | — | — | — | — | — | 136 |
| 3FD2h | INTCON0 | GIE | GIEL | IPEN | — | — | INT2EDG | INT1EDG | INT0EDG | 135 |

**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.

**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 |
|---|---|---|---|---|---|---|---|---|---|---|
| 3A16h | RC6PPS | — | — | — | RC6PPS4 | RC6PPS3 | RC6PPS2 | RC6PPS1 | RC6PPS0 | 280 |
| 3A15h | RC5PPS | — | — | — | RC5PPS4 | RC5PPS3 | RC5PPS2 | RC5PPS1 | RC5PPS0 | 280 |
| 3A14h | RC4PPS | — | — | — | RC4PPS4 | RC4PPS3 | RC4PPS2 | RC4PPS1 | RC4PPS0 | 280 |
| 3A13h | RC3PPS | — | — | — | RC3PPS4 | RC3PPS3 | RC3PPS2 | RC3PPS1 | RC3PPS0 | 280 |
| 3A12h | RC2PPS | — | — | — | RC2PPS4 | RC2PPS3 | RC2PPS2 | RC2PPS1 | RC2PPS0 | 280 |
| 3A11h | RC1PPS | — | — | — | RC1PPS4 | RC1PPS3 | RC1PPS2 | RC1PPS1 | RC1PPS0 | 280 |
| 3A10h | RC0PPS | — | — | — | RC0PPS4 | RC0PPS3 | RC0PPS2 | RC0PPS1 | RC0PPS0 | 280 |
| 3A0Fh | RB7PPS | — | — | — | RB7PPS4 | RB7PPS3 | RB7PPS2 | RB7PPS1 | RB7PPS0 | 280 |
| 3A0Eh | RB6PPS | — | — | — | RB6PPS4 | RB6PPS3 | RB6PPS2 | RB6PPS1 | RB6PPS0 | 280 |
| 3A0Dh | RB5PPS | — | — | — | RB5PPS4 | RB5PPS3 | RB5PPS2 | RB5PPS1 | RB5PPS0 | 280 |
| 3A0Ch | RB4PPS | — | — | — | RB4PPS4 | RB4PPS3 | RB4PPS2 | RB4PPS1 | RB4PPS0 | 280 |
| 3A0Bh | RB3PPS | — | — | — | RB3PPS4 | RB3PPS3 | RB3PPS2 | RB3PPS1 | RB3PPS0 | 280 |
| 3A0Ah | RB2PPS | — | — | — | RB2PPS4 | RB2PPS3 | RB2PPS2 | RB2PPS1 | RB2PPS0 | 280 |
| 3A09h | RB1PPS | — | — | — | RB1PPS4 | RB1PPS3 | RB1PPS2 | RB1PPS1 | RB1PPS0 | 280 |
| 3A08h | RB0PPS | — | — | — | RB0PPS4 | RB0PPS3 | RB0PPS2 | RB0PPS1 | RB0PPS0 | 280 |
| 3A07h | RA7PPS | — | — | — | RA7PPS4 | RA7PPS3 | RA7PPS2 | RA7PPS1 | RA7PPS0 | 280 |
| 3A06h | RA6PPS | — | — | — | RA6PPS4 | RA6PPS3 | RA6PPS2 | RA6PPS1 | RA6PPS0 | 280 |
| 3A05h | RA5PPS | — | — | — | RA5PPS4 | RA5PPS3 | RA5PPS2 | RA5PPS1 | RA5PPS0 | 280 |
| 3A04h | RA4PPS | — | — | — | RA4PPS4 | RA4PPS3 | RA4PPS2 | RA4PPS1 | RA4PPS0 | 280 |
| 3A03h | RA3PPS | — | — | — | RA3PPS4 | RA3PPS3 | RA3PPS2 | RA3PPS1 | RA3PPS0 | 280 |
| 3A02h | RA2PPS | — | — | — | RA2PPS4 | RA2PPS3 | RA2PPS2 | RA2PPS1 | RA2PPS0 | 280 |
| 3A01h | RA1PPS | — | — | — | RA1PPS4 | RA1PPS3 | RA1PPS2 | RA1PPS1 | RA1PPS0 | 280 |
| 3A00h | RA0PPS | — | — | — | RA0PPS4 | RA0PPS3 | RA0PPS2 | RA0PPS1 | RA0PPS0 | 280 |
| 39FFh - 39F8h | — | Unimplemented | | | | | | | | |
| 39F7h | SCANPR | — | — | — | — | — | PR | | | 31 |
| 39F6h - 39F5h | — | Unimplemented | | | | | | | | |
| 39F4h | DMA2PR | — | — | — | — | — | PR | | | 31 |
| 39F3h | DMA1PR | — | — | — | — | — | PR | | | 30 |
| 39F2h | MAINPR | — | — | — | — | — | PR | | | 30 |
| 39F1h | ISRPR | — | — | — | — | — | PR | | | 30 |
| 39F0h | — | Unimplemented | | | | | | | | |
| 39EFh | PRLOCK | — | — | — | — | — | — | — | PRLOCKED | 31 |
| 39EEh - 39E7h | — | Unimplemented | | | | | | | | |
| 39E6h | NVMCON2 | NVMCON2 | | | | | | | | 211 |
| 39E5h | NVMCON1 | REG | | — | FREE | WRERR | WREN | WR | RD | 210 |
| 39E4h | — | Unimplemented | | | | | | | | |
| 39E3h | NVMDAT | DAT | | | | | | | | 212 |
| 39E2h | — | Unimplemented | | | | | | | | |
| 39E1h | NVMADRH(4) | — | — | — | — | — | — | ADR | | 211 |
| 39E0h | NVMADRL | ADR | | | | | | | | 211 |
| 39DFh | OSCFRQ | — | — | — | — | FRQ | | | | 107 |
| 39DEh | OSCTUNE | — | — | TUN | | | | | | 108 |
| 39DDh | OSCEN | EXTOEN | HFOEN | MFOEN | LFOEN | SOSCEN | ADOEN | — | — | 109 |
| 39DCh | OSCSTAT | EXTOR | HFOR | MFOR | LFOR | SOR | ADOR | — | PLLR | 106 |
| 39DBh | OSCCON3 | CSWHOLD | SOSCPWR | — | ORDY | NOSCR | — | — | — | 105 |

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

**Note 1:** Unimplemented in LF devices.

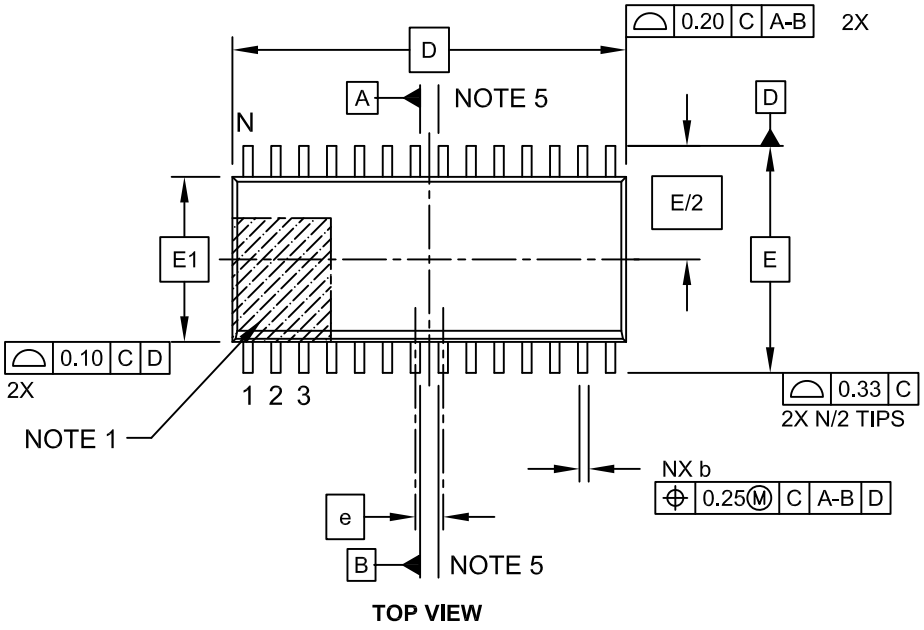
**2:** Unimplemented in PIC18(L)F26/27K42.

**3:** Unimplemented on PIC18(L)F26/27/45/46/47K42 devices.

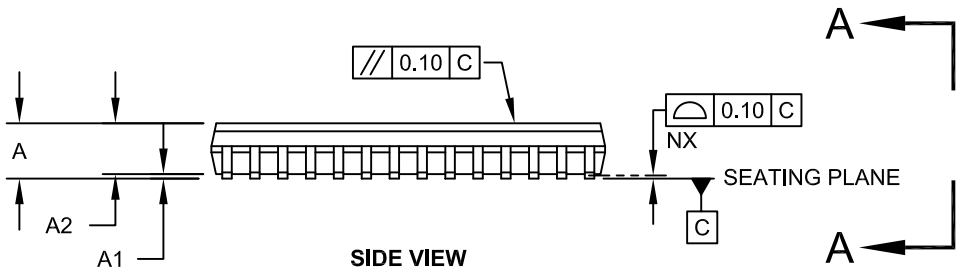
**4:** Unimplemented in PIC18(L)F45/55K42.

### 28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

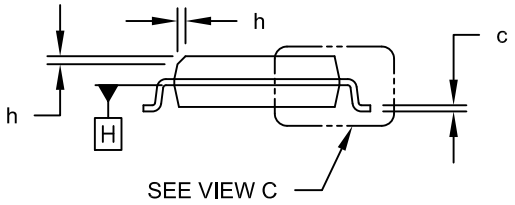
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




**TOP VIEW**




**SIDE VIEW**




**VIEW A-A**

Microchip Technology Drawing C04-052C Sheet 1 of 2

## APPENDIX A: REVISION HISTORY

### Revision A (6/2017)

Initial release of the document.

### Revision B (12/2017)

Standard operating conditions updated in Section 44.0, Electrical Specifications. Other minor corrections.

## THE MICROCHIP WEBSITE

Microchip provides online support via our website at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

**Technical support is available through the website at: http://www.microchip.com/support**