**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 | 20MHz |
| Connectivity | I²C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 25 |
| Program Memory Size | 3.5KB (2K x 14) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 128 x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 3.6V |
| Data Converters | A/D 11x8b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 125°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/pic16lf722a-e-ss |

**TABLE 1: 28-PIN SPDIP/SOIC/SSOP/QFN/UQFN SUMMARY (PIC16(L)F722A/723A)**

| I/O | 28-Pin SPDIP, SOIC, SSOP | 28-Pin QFN, UQFN | A/D | Cap Sensor | Timers | CCP | AUSART | SSP | Interrupt | Pull Up | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RA0 | 2 | 27 | AN0 | — | — | — | — | $\overline{SS}$(3) | — | — | $V_{CAP}$(4) |
| RA1 | 3 | 28 | AN1 | — | — | — | — | — | — | — | — |
| RA2 | 4 | 1 | AN2 | — | — | — | — | — | — | — | — |
| RA3 | 5 | 2 | AN3/$V_{REF}$ | — | — | — | — | — | — | — | — |
| RA4 | 6 | 3 | — | CPS6 | T0CKI | — | — | — | — | — | — |
| RA5 | 7 | 4 | AN4 | CPS7 | — | — | — | $\overline{SS}$(3) | — | — | $V_{CAP}$(4) |
| RA6 | 10 | 7 | — | — | — | — | — | — | — | — | OSC2/CLKOUT/$V_{CAP}$(4) |
| RA7 | 9 | 6 | — | — | — | — | — | — | — | — | OSC1/CLKIN |
| RB0 | 21 | 18 | AN12 | CPS0 | — | — | — | — | IOC/INT | Y | — |
| RB1 | 22 | 19 | AN10 | CPS1 | — | — | — | — | IOC | Y | — |
| RB2 | 23 | 20 | AN8 | CPS2 | — | — | — | — | IOC | Y | — |
| RB3 | 24 | 21 | AN9 | CPS3 | — | CCP2(2) | — | — | IOC | Y | — |
| RB4 | 25 | 22 | AN11 | CPS4 | — | — | — | — | IOC | Y | — |
| RB5 | 26 | 23 | AN13 | CPS5 | T1G | — | — | — | IOC | Y | — |
| RB6 | 27 | 24 | — | — | — | — | — | — | IOC | Y | ICSPCLK/ICDCLK |
| RB7 | 28 | 25 | — | — | — | — | — | — | IOC | Y | ICSPDAT/ICDDAT |
| RC0 | 11 | 8 | — | — | T1OSO/T1CKI | — | — | — | — | — | — |
| RC1 | 12 | 9 | — | — | T1OSI | CCP2(2) | — | — | — | — | — |
| RC2 | 13 | 10 | — | — | — | CCP1 | — | — | — | — | — |
| RC3 | 14 | 11 | — | — | — | — | — | SCK/SCL | — | — | — |
| RC4 | 15 | 12 | — | — | | — | — | SDI/SDA | — | — | — |
| RC5 | 16 | 13 | — | — | — | — | — | SDO | — | — | — |
| RC6 | 17 | 14 | — | — | — | — | TX/CK | — | — | — | — |
| RC7 | 18 | 15 | — | — | — | — | RX/DT | — | — | — | — |
| RE3 | 1 | 26 | — | — | — | — | — | — | — | Y(1) | $\overline{MCLR}$/$V_{PP}$ |
| — | 20 | 17 | — | — | — | — | — | — | — | — | $V_{DD}$ |
| — | 8,19 | 5,16 | — | — | — | — | — | — | — | — | $V_{SS}$ |

**Note 1:** Pull up enabled only with external $\overline{MCLR}$ configuration.
**2:** RC1 is the default pin location for CCP2. RB3 may be selected by changing the CCP2SEL bit in the APFCON register.
**3:** RA5 is the default pin location for $\overline{SS}$. RA0 may be selected by changing the SSSEL bit in the APFCON register.
**4:** PIC16F722A/723A devices only.

**Note:** The PIC16F722A/723A devices have an internal low dropout voltage regulator. An external capacitor must be connected to one of the available $V_{CAP}$ pins to stabilize the regulator. For more information, see **Section 5.0 "Low Dropout (LDO) Voltage Regulator"**. The PIC16LF722A/723A devices do not have the voltage regulator and therefore no external capacitor is required.

#### 2.2.2.1 STATUS Register

The STATUS register, shown in Register 2-1, contains:

- the arithmetic status of the ALU
- the Reset status
- the bank select bits for data memory (SRAM)

The STATUS register can be the destination for any instruction, like any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper three bits and set the Z bit. This leaves the STATUS register as '`000u u1uu`' (where u = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect any Status bits. For other instructions not affecting any Status bits (Refer to **Section 21.0 "Instruction Set Summary"**).

**Note 1:** The C and DC bits operate as $\overline{Borrow}$ and $\overline{Digit\ Borrow}$ out bits, respectively, in subtraction.

**REGISTER 2-1: STATUS: STATUS REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R-1 | R-1 | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| IRP | RP1 | RP0 | $\overline{TO}$ | $\overline{PD}$ | Z | DC(1) | C(1) |
| 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 **IRP:** Register Bank Select bit (used for indirect addressing)
`1` = Bank 2, 3 (100h-1FFh)
`0` = Bank 0, 1 (00h-FFh)

bit 6-5 **RP<1:0>:** Register Bank Select bits (used for direct addressing)
`00` = Bank 0 (00h-7Fh)
`01` = Bank 1 (80h-FFh)
`10` = Bank 2 (100h-17Fh)
`11` = Bank 3 (180h-1FFh)

bit 4 **$\overline{TO}$:** Time-out bit
`1` = After power-up, `CLRWDT` instruction or `SLEEP` instruction
`0` = A WDT time-out occurred

bit 3 **$\overline{PD}$:** Power-down bit
`1` = After power-up or by the `CLRWDT` instruction
`0` = By execution of the `SLEEP` instruction

bit 2 **Z:** Zero bit
`1` = The result of an arithmetic or logic operation is zero
`0` = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit Carry/$\overline{Digit\ Borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)(1)
`1` = A carry-out from the 4th low-order bit of the result occurred
`0` = No carry-out from the 4th low-order bit of the result

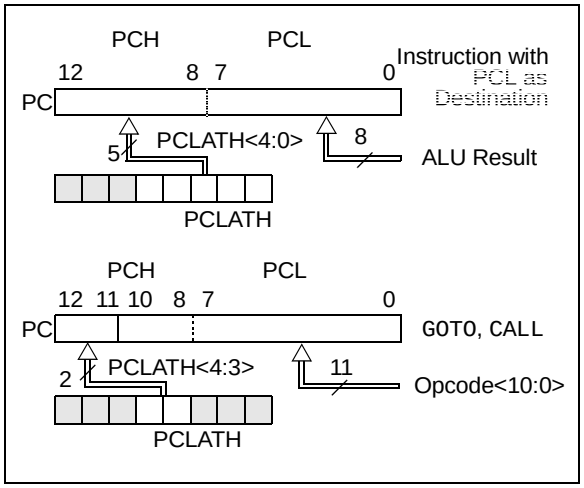
bit 0 **C:** Carry/$\overline{Borrow}$ bit(1) (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)(1)
`1` = A carry-out from the Most Significant bit of the result occurred
`0` = No carry-out from the Most Significant bit of the result occurred

**Note 1:** For $\overline{Borrow}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high-order or low-order bit of the source register.

## 2.3 PCL and PCLATH

The Program Counter (PC) is 13 bits wide. The low byte comes from the PCL register, which is a readable and writable register. The high byte (PC<12:8>) is not directly readable or writable and comes from PCLATH. On any Reset, the PC is cleared. Figure 2-5 shows the two situations for the loading of the PC. The upper example in Figure 2-5 shows how the PC is loaded on a write to PCL (PCLATH<4:0> → PCH). The lower example in Figure 2-5 shows how the PC is loaded during a CALL or GOTO instruction (PCLATH<4:3> → PCH).

**FIGURE 2-5: LOADING OF PC IN DIFFERENT SITUATIONS**




### 2.3.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When performing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block). Refer to Application Note *AN556, Implementing a Table Read* (DS00556).

### 2.3.2 STACK

All devices have an 8-level x 13-bit wide hardware stack (refer to Figures 2-1 and 2-2). The stack space is not part of either program or data space and the Stack Pointer is not readable or writable. The PC is PUSHed onto the stack when a CALL instruction is executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer. This means that after the stack has been PUSHed eight times, the ninth PUSH overwrites the value that was stored from the first PUSH. The tenth PUSH overwrites the second PUSH (and so on).

**Note 1:** There are no Status bits to indicate stack overflow or stack underflow conditions.

**2:** There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, RETURN, RETLW and RETFIE instructions or the vectoring to an interrupt address.

## 2.4 Program Memory Paging

All devices are capable of addressing a continuous 8K word block of program memory. The CALL and GOTO instructions provide only 11 bits of address to allow branching within any 2K program memory page. When doing a CALL or GOTO instruction, the upper 2 bits of the address are provided by PCLATH<4:3>. When doing a CALL or GOTO instruction, the user must ensure that the page select bits are programmed so that the desired program memory page is addressed. If a return from a CALL instruction (or interrupt) is executed, the entire 13-bit PC is POPed off the stack. Therefore, manipulation of the PCLATH<4:3> bits is not required for the RETURN instructions (which POPs the address from the stack).

**Note:** The contents of the PCLATH register are unchanged after a RETURN or RETFIE instruction is executed. The user must rewrite the contents of the PCLATH register for any subsequent subroutine calls or GOTO instructions.

Example 2-1 shows the calling of a subroutine in page 1 of the program memory. This example assumes that PCLATH is saved and restored by the Interrupt Service Routine (if interrupts are used).

**EXAMPLE 2-1: CALL OF A SUBROUTINE IN PAGE 1 FROM PAGE 0**

```
        ORG 500h
        PAGESEL SUB_P1  ;Select page 1
                        ;(800h-FFFh)
        CALL    SUB1_P1 ;Call subroutine in
        :               ;page 1 (800h-FFFh)
        :
        ORG     900h    ;page 1 (800h-FFFh)
SUB1_P1
        :               ;called subroutine
                        ;page 1 (800h-FFFh)
        :
        RETURN          ;return to
                        ;Call subroutine
                        ;in page 0
                        ;(000h-7FFh)
```

**TABLE 3-4: INITIALIZATION CONDITION FOR REGISTERS (CONTINUED)**

| Register | Address | Power-on Reset/ Brown-out Reset(1) | $\overline{\text{MCLR}}$ Reset/ WDT Reset | Wake-up from Sleep through Interrupt/Time out |
|---|---|---|---|---|
| PIE1 | 8Ch | 0000 0000 | 0000 0000 | uuuu uuuu |
| PIE2 | 8Dh | ---- ---0 | ---- ---0 | ---- ---u |
| PCON | 8Eh | ---- --qq | ---- --uu(1,5) | ---- --uu |
| T1GCON | 8Fh | 0000 0x00 | uuuu uxuu | uuuu uxuu |
| OSCCON | 90h | --10 qq-- | --10 qq-- | --uu qq-- |
| OSCTUNE | 91h | --00 0000 | --uu uuuu | --uu uuuu |
| PR2 | 92h | 1111 1111 | 1111 1111 | uuuu uuuu |
| SSPADD | 93h | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPMSK | 93h | 1111 1111 | 1111 1111 | uuuu uuuu |
| SSPSTAT | 94h | 0000 0000 | 0000 0000 | uuuu uuuu |
| WPUB | 95h | 1111 1111 | 1111 1111 | uuuu uuuu |
| IOCB | 96h | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXSTA | 98h | 0000 -010 | 0000 -010 | uuuu -uuu |
| SPBRG | 99h | 0000 0000 | 0000 0000 | uuuu uuuu |
| APFCON | 9Ch | ---- --00 | ---- --00 | ---- --uu |
| FVRCON | 9Dh | q000 --00 | q000 --00 | uuuu --uu |
| ADCON1 | 9Fh | -000 --00 | -000 --00 | -uuu --uu |
| CPSCON0 | 108h | 0--- 0000 | 0--- 0000 | u--- uuuu |
| CPSCON1 | 109h | ---- 0000 | ---- 0000 | ---- uuuu |
| PMDATL | 10Ch | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| PMADRL | 10Dh | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| PMDATH | 10Eh | --xx xxxx | --xx xxxx | --uu uuuu |
| PMADRH | 10Fh | ---x xxxx | ---x xxxx | ---u uuuu |
| ANSELA | 185h | --11 1111 | --11 1111 | --uu uuuu |
| ANSELB | 186h | --11 1111 | --11 1111 | --uu uuuu |
| PMCON1 | 18Ch | 1--- ---0 | 1--- ---0 | u--- ---u |

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, reads as '0', q = value depends on condition.

**Note 1:** If VDD goes too low, Power-on Reset will be activated and registers will be affected differently.

**2:** One or more bits in INTCON and/or PIR1 and PIR2 will be affected (to cause wake-up).

**3:** When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

**4:** See Table 3-5 for Reset value for specific condition.

**5:** If Reset was due to brown-out, then bit 0 = 0. All other Resets will cause bit 0 = u.

## 7.2 Clock Source Modes

Clock source modes can be classified as external or internal.

- Internal clock source (INTOSC) is contained within the oscillator module and derived from a 500 kHz high precision oscillator. The oscillator module has eight selectable output frequencies, with a maximum internal frequency of 16 MHz.
- External clock modes rely on external circuitry for the clock source. Examples are: oscillator modules (EC mode), quartz crystal resonators or ceramic resonators (LP, XT and HS modes) and Resistor-Capacitor (RC) mode circuits.

The system clock can be selected between external or internal clock sources via the FOSC bits of the Configuration Word 1.

## 7.3 Internal Clock Modes

The oscillator module has eight output frequencies derived from a 500 kHz high precision oscillator. The IRCF bits of the OSCCON register select the postscaler applied to the clock source dividing the frequency by 1, 2, 4 or 8. Setting the PLLEN bit of the Configuration Word 1 locks the internal clock source to 16 MHz before the postscaler is selected by the IRCF bits. The PLLEN bit must be set or cleared at the time of programming; therefore, only the upper or low four clock source frequencies are selectable in software.

### 7.3.1 INTOSC AND INTOSCIO MODES

The INTOSC and INTOSCIO modes configure the internal oscillators as the system clock source when the device is programmed using the oscillator selection or the FOSC<2:0> bits in the CONFIG1 register. See **Section 8.0 “Device Configuration”** for more information.

In INTOSC mode, OSC1/CLKIN is available for general purpose I/O. OSC2/CLKOUT outputs the selected internal oscillator frequency divided by 4. The CLKOUT signal may be used to provide a clock for external circuitry, synchronization, calibration, test or other application requirements.

In INTOSCIO mode, OSC1/CLKIN and OSC2/CLKOUT are available for general purpose I/O.

### 7.3.2 FREQUENCY SELECT BITS (IRCF)

The output of the 500 kHz INTOSC and 16 MHz INTOSC, with Phase-Locked Loop enabled, connect to a postscaler and multiplexer (see Figure 7-1). The Internal Oscillator Frequency Select bits (IRCF) of the OSCCON register select the frequency output of the internal oscillator. Depending upon the PLLEN bit, one of four frequencies of two frequency sets can be selected via software:

If PLLEN = 1, frequency selection is as follows:

- 16 MHz
- 8 MHz (Default after Reset)
- 4 MHz
- 2 MHz

If PLLEN = 0, frequency selection is as follows:

- 500 kHz
- 250 kHz (Default after Reset)
- 125 kHz
- 62.5 kHz

| | |
|---|---|
| **Note:** | Following any Reset, the IRCF<1:0> bits of the OSCCON register are set to ‘10’ and the frequency selection is set to 8 MHz or 250 kHz. The user can modify the IRCF bits to select a different frequency. |

There is no start-up delay before a new frequency selected in the IRCF bits takes effect. This is because the old and new frequencies are derived from INTOSC via the postscaler and multiplexer.

Start-up delay specifications are located in Table 23-2 in **Section 23.0 “Electrical Specifications”**.

**REGISTER 8-1: CONFIG1: CONFIGURATION WORD REGISTER 1 (CONTINUED)**

bit 2-0 **FOSC<2:0>:** Oscillator Selection bits
111 = RC oscillator: CLKOUT function on RA6/OSC2/CLKOUT pin, RC on RA7/OSC1/CLKIN
110 = RCIO oscillator: I/O function on RA6/OSC2/CLKOUT pin, RC on RA7/OSC1/CLKIN
101 = INTOSC oscillator: CLKOUT function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLKIN
100 = INTOSCIO oscillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLKIN
011 = EC: I/O function on RA6/OSC2/CLKOUT pin, CLKIN on RA7/OSC1/CLKIN
010 = HS oscillator: High-speed crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN
001 = XT oscillator: Crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN
000 = LP oscillator: Low-power crystal on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN

**Note 1:** Enabling Brown-out Reset does not automatically enable Power-up Timer.
**2:** The entire program memory will be erased when the code protection is turned off.
**3:** When $\overline{MCLR}$ is asserted in INTOSC or RC mode, the internal clock oscillator is disabled.
**4:** MPLAB® X IDE masks unimplemented Configuration bits to '0'.

**REGISTER 8-2: CONFIG2: CONFIGURATION WORD REGISTER 2**

| U-1(1) | U-1(1) | U-1(1) | U-1(1) | U-1(1) | U-1(1) |
|---|---|---|---|---|---|
| — | — | — | — | — | — |
| bit 13 | | | | | bit 8 |

| U-1(1) | U-1(1) | R/P-1 | R/P-1 | U-1(1) | U-1(1) | U-1(1) | U-1(1) |
|---|---|---|---|---|---|---|---|
| — | — | VCAPEN1 | VCAPEN0 | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | P = Programmable bit | | |
|---|---|---|---|
| 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 13-6 **Unimplemented**: Read as '1'

bit 5-4 **VCAPEN<1:0>:** Voltage Regulator Capacitor Enable bits
For the PIC16LF722A/723A:
These bits are ignored. All VCAP pin functions are disabled.
For the PIC16F722A/723A:
00 = VCAP functionality is enabled on RA0
01 = VCAP functionality is enabled on RA5
10 = VCAP functionality is enabled on RA6
11 = All VCAP functions are disabled (not recommended)

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

**Note 1:** MPLAB® X IDE masks unimplemented Configuration bits to '0'.

**FIGURE 12-4: TIMER1 GATE COUNT ENABLE MODE**

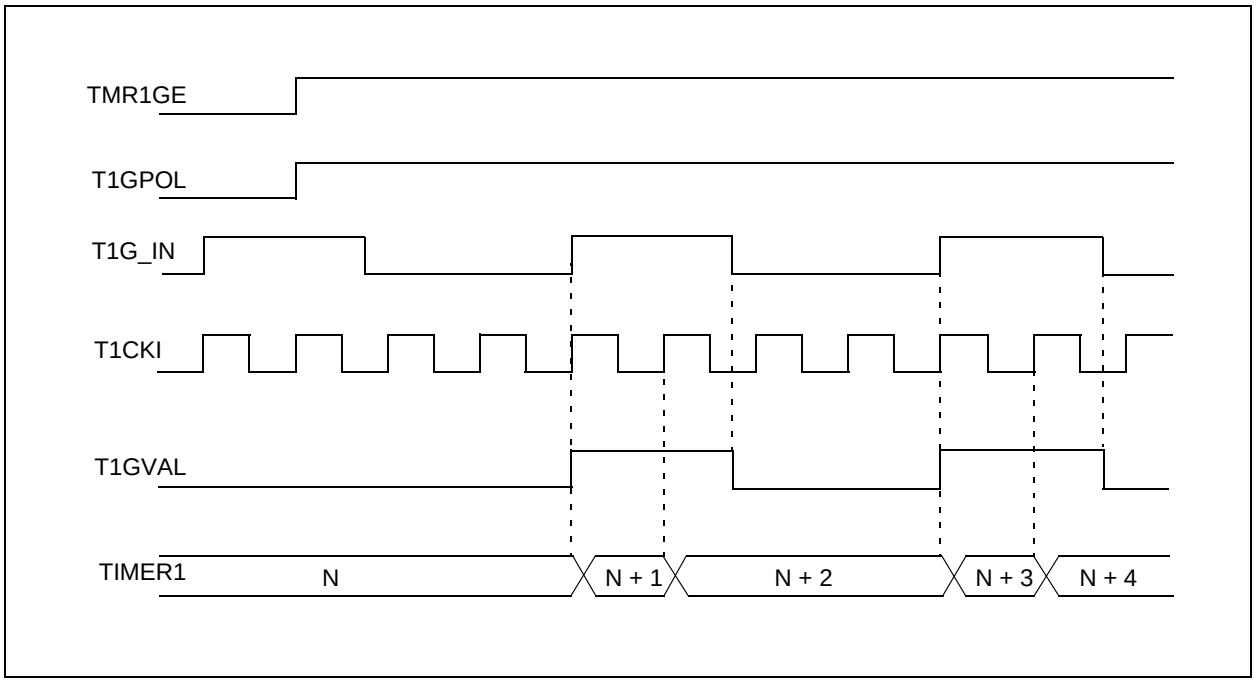
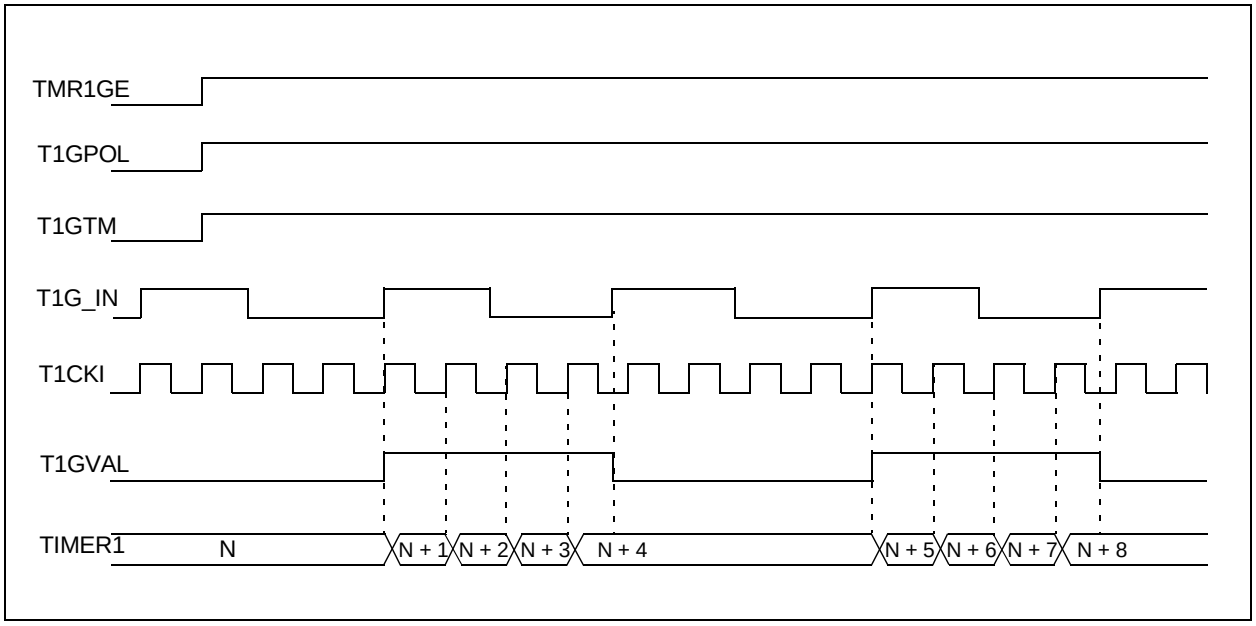



**FIGURE 12-5: TIMER1 GATE TOGGLE MODE**



### 15.3.4 PWM RESOLUTION

The resolution determines the number of available duty cycles for a given period. For example, a 10-bit resolution will result in 1024 discrete duty cycles, whereas an 8-bit resolution will result in 256 discrete duty cycles.

The maximum PWM resolution is ten bits when PR2 is 255. The resolution is a function of the PR2 register value as shown by Equation 15-4.

**EQUATION 15-4: PWM RESOLUTION**

$$Resolution = \frac{log[4(PR2 + 1)]}{log(2)} \ bits$$

| Note: | If the pulse-width value is greater than the period, the assigned PWM pin(s) will remain unchanged. |
|---|---|

**TABLE 15-5: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 20 MHz)**

| PWM Frequency | 1.22 kHz | 4.88 kHz | 19.53 kHz | 78.12 kHz | 156.3 kHz | 208.3 kHz |
|---|---|---|---|---|---|---|
| Timer Prescale (1, 4, 16) | 16 | 4 | 1 | 1 | 1 | 1 |
| PR2 Value | 0xFF | 0xFF | 0xFF | 0x3F | 0x1F | 0x17 |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 6.6 |

**TABLE 15-6: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 8 MHz)**

| PWM Frequency | 1.22 kHz | 4.90 kHz | 19.61 kHz | 76.92 kHz | 153.85 kHz | 200.0 kHz |
|---|---|---|---|---|---|---|
| Timer Prescale (1, 4, 16) | 16 | 4 | 1 | 1 | 1 | 1 |
| PR2 Value | 0x65 | 0x65 | 0x65 | 0x19 | 0x0C | 0x09 |
| Maximum Resolution (bits) | 8 | 8 | 8 | 6 | 5 | 5 |

### 15.3.5 OPERATION IN SLEEP MODE

In Sleep mode, the TMR2 register will not increment and the state of the module will not change. If the CCPx pin is driving a value, it will continue to drive that value. When the device wakes up, TMR2 will continue from its previous state.

### 15.3.6 CHANGES IN SYSTEM CLOCK FREQUENCY

The PWM frequency is derived from the system clock frequency (FOSC). Any changes in the system clock frequency will result in changes to the PWM frequency. Refer to **Section 7.0 "Oscillator Module"** for additional details.

### 15.3.7 EFFECTS OF RESET

Any Reset will force all ports to Input mode and the CCP registers to their Reset states.

### 15.3.8 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for PWM operation:

1. Disable the PWM pin (CCPx) output driver(s) by setting the associated TRIS bit(s).
2. Load the PR2 register with the PWM period value.
3. Configure the CCP module for the PWM mode by loading the CCPxCON register with the appropriate values.
4. Load the CCPRxL register and the DCxBx bits of the CCPxCON register, with the PWM duty cycle value.
5. Configure and start Timer2:
   - Clear the TMR2IF interrupt flag bit of the PIR1 register. See Note below.
   - Configure the T2CKPS bits of the T2CON register with the Timer2 prescale value.
   - Enable Timer2 by setting the TMR2ON bit of the T2CON register.
6. Enable PWM output pin:
   - Wait until Timer2 overflows, TMR2IF bit of the PIR1 register is set. See Note below.
   - Enable the PWM pin (CCPx) output driver(s) by clearing the associated TRIS bit(s).

| **Note:** | In order to send a complete duty cycle and period on the first PWM output, the above steps must be included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then step 6 may be ignored. |
|---|---|

### 16.3.2 SYNCHRONOUS SLAVE MODE

The following bits are used to configure the AUSART for Synchronous slave operation:

- SYNC = 1
- CSRC = 0
- SREN = 0 (for transmit); SREN = 1 (for receive)
- CREN = 0 (for transmit); CREN = 1 (for receive)
- SPEN = 1

Setting the SYNC bit of the TXSTA register configures the device for synchronous operation. Clearing the CSRC bit of the TXSTA register configures the device as a slave. Clearing the SREN and CREN bits of the RCSTA register ensures that the device is in the Transmit mode, otherwise the device will be configured to receive. Setting the SPEN bit of the RCSTA register enables the AUSART.

#### 16.3.2.1 AUSART Synchronous Slave Transmit

The operation of the Synchronous Master and Slave modes are identical (refer to **Section 16.3.1.2 "Synchronous Master Transmission")**, except in the case of the Sleep mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

1. The first character will immediately transfer to the TSR register and transmit.
2. The second word will remain in TXREG register.
3. The TXIF bit will not be set.
4. After the first character has been shifted out of TSR, the TXREG register will transfer the second character to the TSR and the TXIF bit will now be set.
5. If the PEIE and TXIE bits are set, the interrupt will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will call the Interrupt Service Routine.

#### 16.3.2.2 Synchronous Slave Transmission Setup:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the CREN and SREN bits.
3. If using interrupts, ensure that the GIE and PEIE bits of the INTCON register are set and set the TXIE bit.
4. If 9-bit transmission is desired, set the TX9 bit.
5. Enable transmission by setting the TXEN bit.
6. Verify address detection is disabled by clearing the ADDEN bit of the RCSTA register.
7. If 9-bit transmission is selected, insert the Most Significant bit into the TX9D bit.
8. Start transmission by writing the Least Significant eight bits to the TXREG register.

**TABLE 16-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE | PEIE | T0IE | INTE | RBIE | T0IF | INTF | RBIF | 0000 000x | 0000 000x |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000X | 0000 000X |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 1111 1111 | 1111 1111 |
| TXREG | AUSART Transmit Data Register | | | | | | | | 0000 0000 | 0000 0000 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 |

**Legend:** x = unknown, - = unimplemented read as '0'. Shaded cells are not used for synchronous slave transmission.

### 17.1.2 SLAVE MODE

For any SPI device acting as a slave, the data is transmitted and received as external clock pulses appear on SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

#### 17.1.2.1 Slave Mode Operation

The SSP consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR until the received data is ready.

The slave has no control as to when data will be clocked in or out of the device. All data that is to be transmitted, to a master or another slave, must be loaded into the SSPBUF register before the first clock pulse is received.

Once eight bits of data have been received:

- Received byte is moved to the SSPBUF register
- BF bit of the SSPSTAT register is set
- SSPIF bit of the PIR1 register is set

Any write to the SSPBUF register during transmission/ reception of data will be ignored and the Write Collision Detect bit, WCOL of the SSPCON register, will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPBUF register completed successfully.

The user's firmware must read SSPBUF, clearing the BF flag, or the SSPOV bit of the SSPCON register will be set with the reception of the next byte and communication will be disabled.

A SPI module transmits and receives at the same time, occasionally causing dummy data to be transmitted/ received. It is up to the user to determine which data is to be used and what can be discarded.

#### 17.1.2.2 Enabling Slave I/O

To enable the serial port, the SSPEN bit of the SSPCON register must be set. If a Slave mode of operation is selected in the SSPM bits of the SSPCON register, the SDI, SDO and SCK pins will be assigned as serial port pins.

For these pins to function as serial port pins, they must have their corresponding data direction bits set or cleared in the associated TRIS register as follows:

- SDI configured as input
- SDO configured as output
- SCK configured as input

Optionally, a fourth pin, Slave Select ($\overline{SS}$) may be used in Slave mode. Slave Select may be configured to operate on one of the following pins via the SSSEL bit in the APFCON register.

- RA5/AN4/$\overline{SS}$
- RA0/AN0/$\overline{SS}$

Upon selection of a Slave Select pin, the appropriate bits must be set in the ANSELA and TRISA registers. Slave Select must be set as an input by setting the corresponding bit in TRISA, and digital I/O must be enabled on the $\overline{SS}$ pin by clearing the corresponding bit of the ANSELA register.

#### 17.1.2.3 Slave Mode Setup

When initializing the SSP module to SPI Slave mode, compatibility must be ensured with the master device. This is done by programming the appropriate control bits of the SSPCON and SSPSTAT registers. These control bits allow the following to be specified:

- SCK as clock input
- Idle state of SCK (CKP bit)
- Data input sample phase (SMP bit)
- Output data on rising/falling edge of SCK (CKE bit)

Figure 17-4 and Figure 17-5 show example waveforms of Slave mode operation.

### 17.2.7 CLOCK STRETCHING

During any SCL low phase, any device on the $I^2C$ bus may hold the SCL line low and delay, or pause, the transmission of data. This “stretching” of a transmission allows devices to slow down communication on the bus. The SCL line must be constantly sampled by the master to ensure that all devices on the bus have released SCL for more data.

Stretching usually occurs after an $\overline{ACK}$ bit of a transmission, delaying the first bit of the next byte. The SSP module hardware automatically stretches for two conditions:

- After a 10-bit address byte is received (update SSPADD register)
- Anytime the CKP bit of the SSPCON register is cleared by hardware

The module will hold SCL low until the CKP bit is set. This allows the user slave software to update SSPBUF with data that may not be readily available. In 10-bit addressing modes, the SSPADD register must be updated after receiving the first and second address bytes. The SSP module will hold the SCL line low until the SSPADD has a byte written to it. The UA bit of the SSPSTAT register will be set, along with SSPIF, indicating an address update is needed.

### 17.2.8 FIRMWARE MASTER MODE

Master mode of operation is supported in firmware using interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits of the SSPSTAT register are cleared from a Reset or when the SSP module is disabled (SSPEN cleared). The Stop (P) and Start (S) bits will toggle based on the Start and Stop conditions. Control of the $I^2C$ bus may be taken when the P bit is set or the bus is Idle and both the S and P bits are clear.

In Firmware Master mode, the SCL and SDA lines are manipulated by setting/clearing the corresponding TRIS bit(s). The output level is always low, irrespective of the value(s) in the corresponding PORT register bit(s). When transmitting a ‘1’, the TRIS bit must be set (input) and a ‘0’, the TRIS bit must be clear (output).

The following events will cause the SSP Interrupt Flag bit, SSPIF, to be set (SSP Interrupt will occur if enabled):

- Start condition
- Stop condition
- Data transfer byte transmitted/received

Firmware Master mode of operation can be done with either the Slave mode Idle (SSPM<3:0> = 1011), or with either of the Slave modes in which interrupts are enabled. When both master and slave functionality is enabled, the software needs to differentiate the source(s) of the interrupt.

Refer to Application Note *AN554, Software Implementation of $I^2C$™ Bus Master* (DS00554) for more information.

### 17.2.9 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allow the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the SSP module is disabled. The Stop (P) and Start (S) bits will toggle based on the Start and Stop conditions. Control of the $I^2C$ bus may be taken when the P bit of the SSPSTAT register is set or when the bus is Idle, and both the S and P bits are clear. When the bus is busy, enabling the SSP Interrupt will generate the interrupt when the Stop condition occurs.

In Multi-Master operation, the SDA line must be monitored to see if the signal level is the expected output level. This check only needs to be done when a high level is output. If a high level is expected and a low level is present, the device needs to release the SDA and SCL lines (set TRIS bits). There are two stages where this arbitration of the bus can be lost. They are the address transfer and data transfer stages.

When the slave logic is enabled, the slave continues to receive. If arbitration was lost during the address transfer stage, communication to the device may be in progress. If addressed, an $\overline{ACK}$ pulse will be generated. If arbitration was lost during the data transfer stage, the device will need to re-transfer the data at a later time.

Refer to Application Note *AN578, Use of the SSP Module in the $I^2C$™ Multi-Master Environment* (DS00578) for more information.

## 21.2 Instruction Descriptions

| ADDLW | Add literal and W |
|---|---|
| Syntax: | [ *label* ] ADDLW k |
| Operands: | $0 \le k \le 255$ |
| Operation: | (W) + k → (W) |
| Status Affected: | C, DC, Z |
| Description: | The contents of the W register are added to the 8-bit literal 'k' and the result is placed in the W register. |

| ADDWF | Add W and f |
|---|---|
| Syntax: | [ *label* ] ADDWF f,d |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | (W) + (f) → (destination) |
| Status Affected: | C, DC, Z |
| Description: | Add the contents of the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'. |

| ANDLW | AND literal with W |
|---|---|
| Syntax: | [ *label* ] ANDLW k |
| Operands: | $0 \le k \le 255$ |
| Operation: | (W) .AND. (k) → (W) |
| Status Affected: | Z |
| Description: | The contents of W register are AND'ed with the 8-bit literal 'k'. The result is placed in the W register. |

| ANDWF | AND W with f |
|---|---|
| Syntax: | [ *label* ] ANDWF f,d |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | (W) .AND. (f) → (destination) |
| Status Affected: | Z |
| Description: | AND the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'. |

| BCF | Bit Clear f |
|---|---|
| Syntax: | [ *label* ] BCF f,b |
| Operands: | $0 \le f \le 127$<br>$0 \le b \le 7$ |
| Operation: | 0 → (f<b>) |
| Status Affected: | None |
| Description: | Bit 'b' in register 'f' is cleared. |

| BSF | Bit Set f |
|---|---|
| Syntax: | [ *label* ] BSF f,b |
| Operands: | $0 \le f \le 127$<br>$0 \le b \le 7$ |
| Operation: | 1 → (f<b>) |
| Status Affected: | None |
| Description: | Bit 'b' in register 'f' is set. |

| BTFSC | Bit Test f, Skip if Clear |
|---|---|
| Syntax: | [ *label* ] BTFSC f,b |
| Operands: | $0 \le f \le 127$<br>$0 \le b \le 7$ |
| Operation: | skip if (f<b>) = 0 |
| Status Affected: | None |
| Description: | If bit 'b' in register 'f' is '1', the next instruction is executed.<br>If bit 'b', in register 'f', is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2-cycle instruction. |

| RETFIE | Return from Interrupt |
|---|---|
| Syntax: | [ *label* ] RETFIE |
| Operands: | None |
| Operation: | TOS → PC,<br>1 → GIE |
| Status Affected: | None |
| Description: | Return from Interrupt. Stack is POPed and Top-of-Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a 2-cycle instruction. |
| Words: | 1 |
| Cycles: | 2 |
| Example: | `RETFIE`<br>After Interrupt<br>PC = TOS<br>GIE = 1 |

| RETLW | Return with literal in W |
|---|---|
| Syntax: | [ *label* ] RETLW k |
| Operands: | 0 ≤ k ≤ 255 |
| Operation: | k → (W);<br>TOS → PC |
| Status Affected: | None |
| Description: | The W register is loaded with the 8-bit literal ‘k’. The program counter is loaded from the top of the stack (the return address). This is a 2-cycle instruction. |
| Words: | 1 |
| Cycles: | 2 |
| Example: | `CALL TABLE;W contains table`<br>`;offset value`<br>`TABLE ;W now has table value`<br>`•`<br>`•`<br>`ADDWF PC ;W = offset`<br>`RETLW k1 ;Begin table`<br>`RETLW k2 ;`<br>`•`<br>`•`<br>`•`<br>`RETLW kn ; End of table`<br><br>Before Instruction<br>W = 0x07<br>After Instruction<br>W = value of k8 |

| RETURN | Return from Subroutine |
|---|---|
| Syntax: | [ *label* ] RETURN |
| Operands: | None |
| Operation: | TOS → PC |
| Status Affected: | None |
| Description: | Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a 2-cycle instruction. |

## 22.11 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

## 22.12 Third-Party Development Tools

Microchip also offers a great collection of tools from third-party vendors. These tools are carefully selected to offer good value and unique functionality.

- Device Programmers and Gang Programmers from companies, such as SoftLog and CCS
- Software Tools from companies, such as Gimpel and Trace Systems
- Protocol Analyzers from companies, such as Saleae and Total Phase
- Demonstration Boards from companies, such as MikroElektronika, Digilent® and Olimex
- Embedded Ethernet Solutions from companies, such as EZ Web Lynx, WIZnet and IPLogika®

**TABLE 23-14: CAP SENSE OSCILLATOR SPECIFICATIONS**

| Param. No. | Symbol | Characteristic | | Min. | Typ† | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| CS01 | ISRC | Current Source | High | — | -5.8 | -6 | µA | -40, -85°C |
| | | | Medium | — | -1.1 | -3.2 | µA | |
| | | | Low | — | -0.2 | -0.9 | µA | |
| CS02 | ISNK | Current Sink | High | — | 6.6 | 6 | µA | -40, -85°C |
| | | | Medium | — | 1.3 | 3.2 | µA | |
| | | | Low | — | 0.24 | 0.9 | µA | |
| CS03 | VCHYST | Cap Hysteresis | High | — | 525 | — | mV | VCTH-VCTL |
| | | | Medium | — | 375 | — | mV | |
| | | | Low | — | 280 | — | mV | |

* 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.

**FIGURE 23-22: CAP SENSE OSCILLATOR**

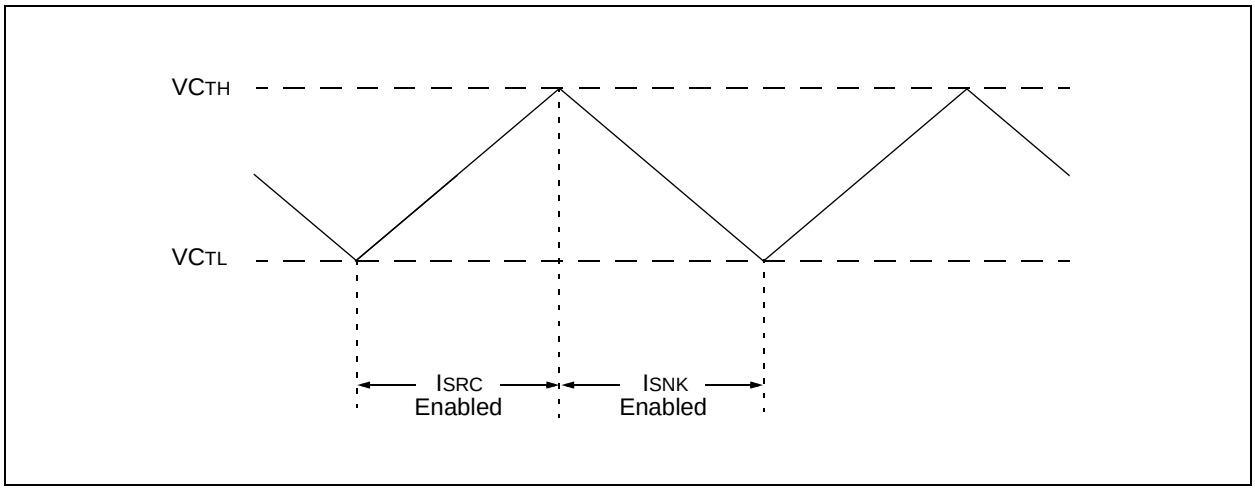


**FIGURE 24-22: PIC16LF722A/723A MAXIMUM IDD vs. FOSC OVER VDD, INTOSC MODE**

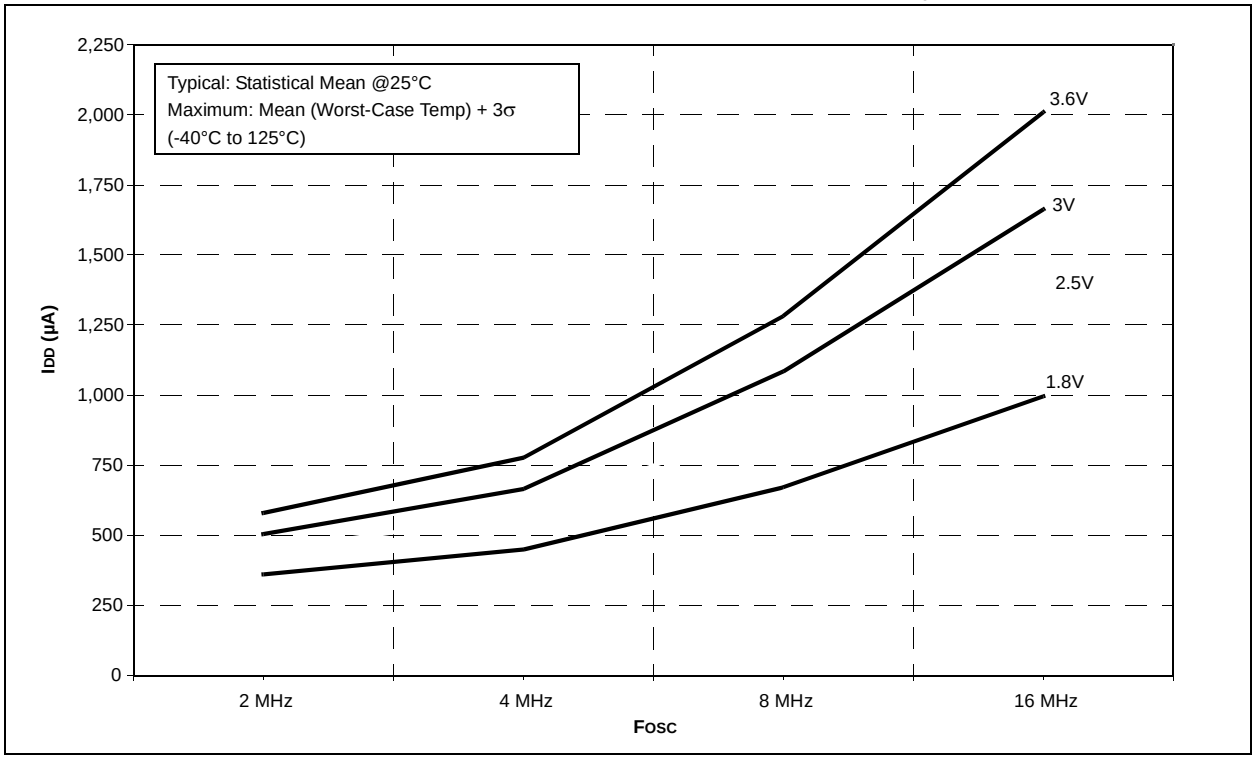



**FIGURE 24-23: PIC16F722A/723A TYPICAL IDD vs. FOSC OVER VDD, INTOSC MODE, VCAP = 0.1µF**

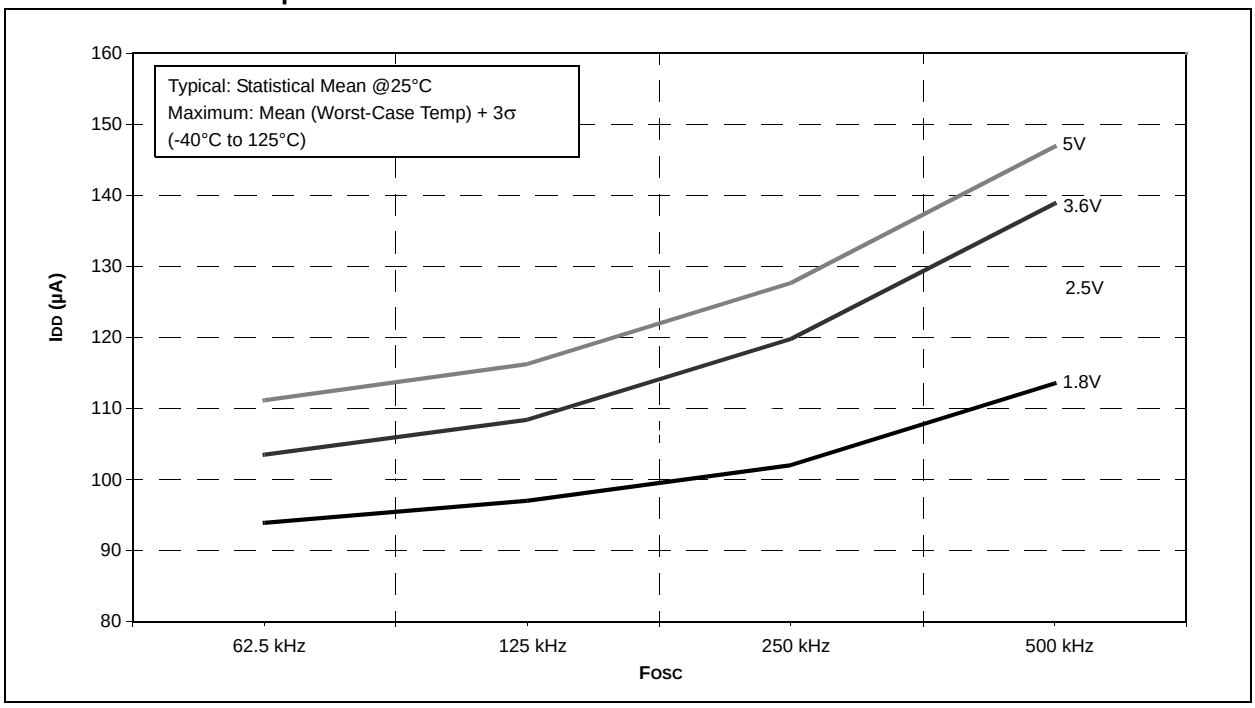


**FIGURE 24-28: PIC16LF722A/723A MAXIMUM BASE IPD vs. VDD**

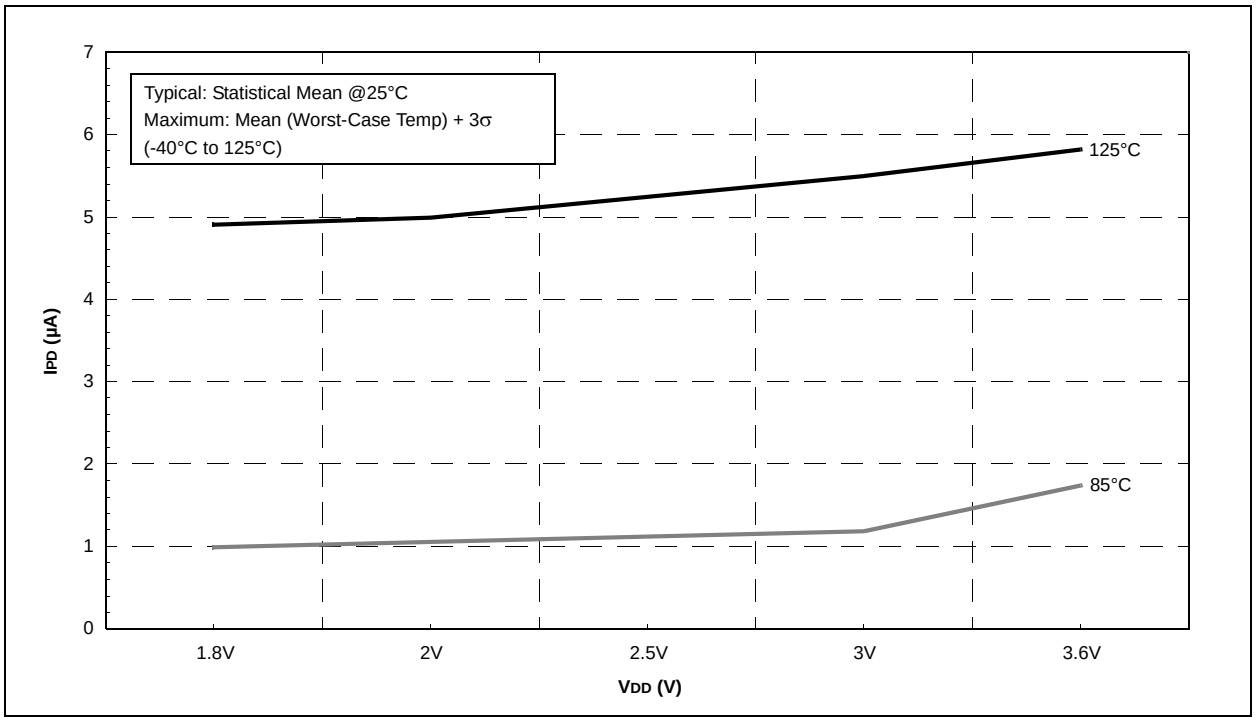
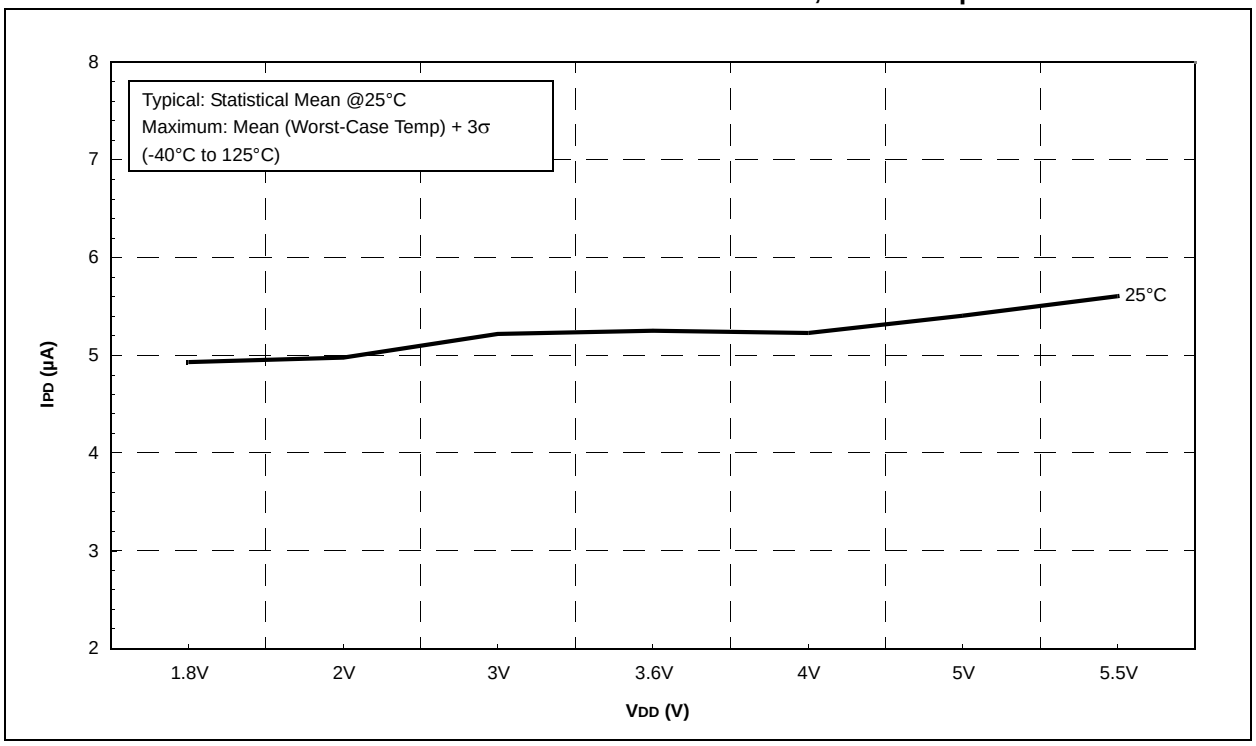
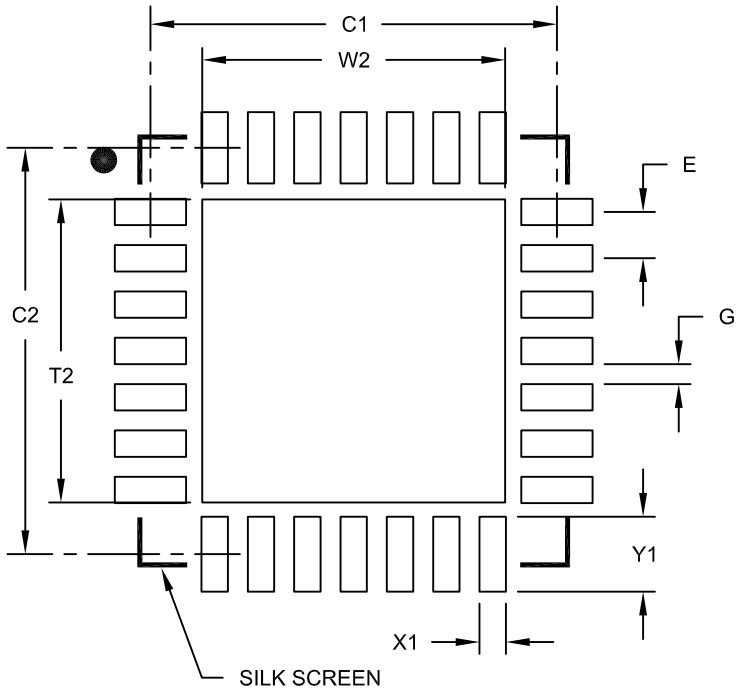



**FIGURE 24-29: PIC16F722A/723A TYPICAL BASE IPD vs. VDD, VCAP = 0.1µF**



### 28-Lead Plastic Quad Flat, No Lead Package (ML) – 6x6 mm Body [QFN] with 0.55 mm Contact Length

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




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.65 BSC | |
| Optional Center Pad Width | W2 | | | 4.25 |
| Optional Center Pad Length | T2 | | | 4.25 |
| Contact Pad Spacing | C1 | | 5.70 | |
| Contact Pad Spacing | C2 | | 5.70 | |
| Contact Pad Width (X28) | X1 | | | 0.37 |
| Contact Pad Length (X28) | Y1 | | | 1.00 |
| Distance Between Pads | G | 0.20 | | |

Notes:

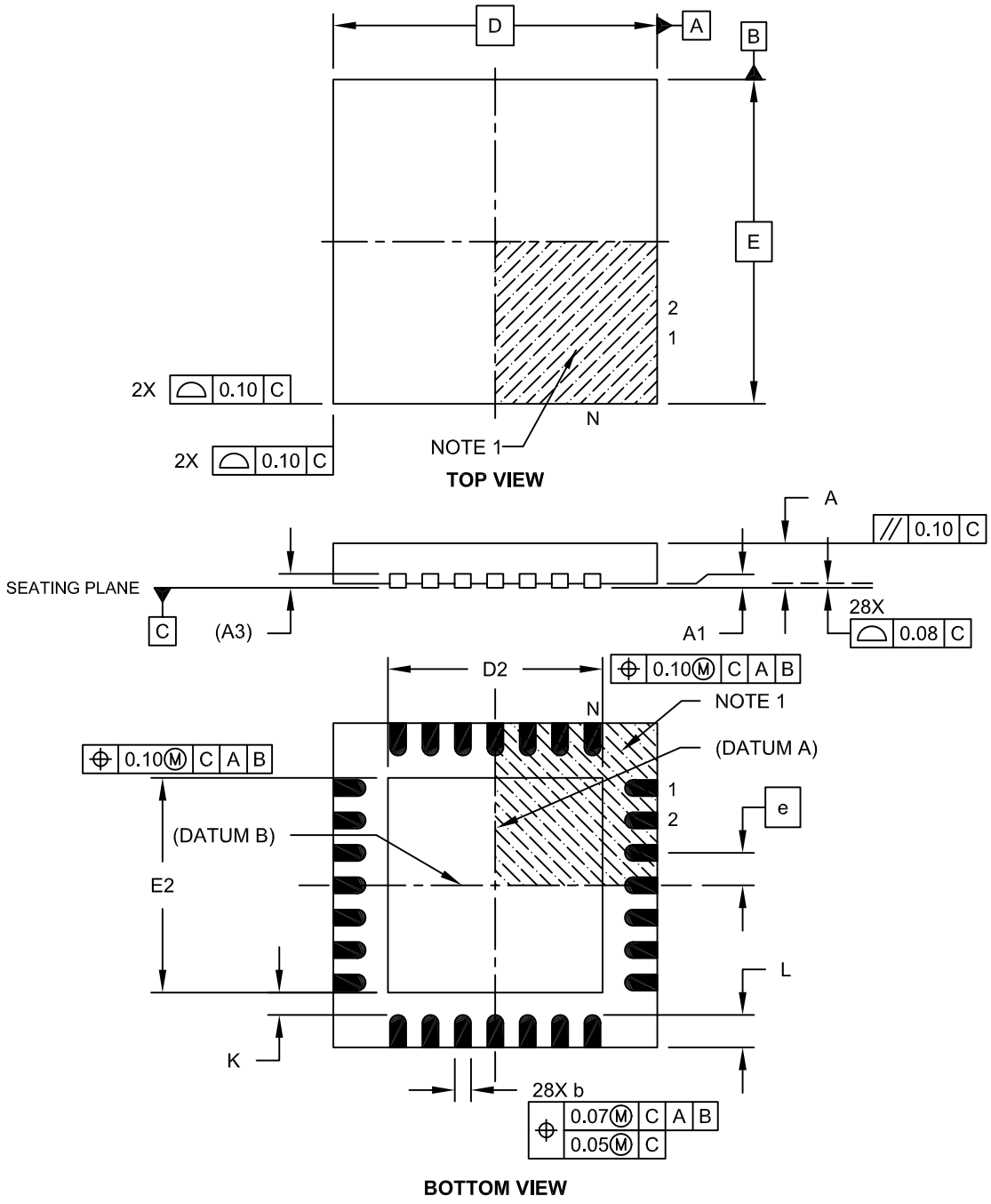
1. Dimensioning and tolerancing per ASME Y14.5M

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

Microchip Technology Drawing No. C04-2105A

### 28-Lead Plastic Ultra Thin Quad Flat, No Lead Package (MV) – 4x4x0.5 mm Body [UQFN]

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




Microchip Technology Drawing C04-152A Sheet 1 of 2

## APPENDIX A: DATA SHEET REVISION HISTORY

### Revision A (April 2010)

Original release of this data sheet.

### Revision B (January 2012)

Updated the data sheet to new format; Updated Figure 9-1 and Register 9-1; Updated the Packaging Information section; Updated the Product Identification System section; Other minor corrections.

### Revision C (03/2016)

Updated Table 2-1, Table 6-1 and Table 6-3; Updated Register 14-2; Other minor corrections.

## APPENDIX B: MIGRATING FROM OTHER PIC® DEVICES

This discusses some of the issues in migrating from other PIC® devices to the PIC16F722A/723A family of devices.

| | |
|---|---|
| **Note:** | This device has been designed to perform to the parameters of its data sheet. It has been tested to an electrical specification designed to determine its conformance with these parameters. Due to process differences in the manufacture of this device, this device may have different performance characteristics than its earlier version. These differences may cause this device to perform differently in your application than the earlier version of this device. |

| | |
|---|---|
| **Note:** | The user should verify that the device oscillator starts and performs as expected. Adjusting the loading capacitor values and/or the oscillator mode may be required. |

### B.1 PIC16F77 to PIC16F722A/723A

**TABLE B-1: FEATURE COMPARISON**

| Feature | PIC16F77 | PIC16F722A/723A |
|---|---|---|
| Max. Operating Speed | 20 MHz | 20 MHz |
| Max. Program Memory (Words) | 8K | 4K |
| Max. SRAM (Bytes) | 368 | 192 |
| A/D Resolution | 8-bit | 8-bit |
| Timers (8/16-bit) | 2/1 | 2/1 |
| Oscillator Modes | 4 | 8 |
| Brown-out Reset | Y | Y |
| Internal Pull ups | RB<7:0> | RB<7:0> |
| Interrupt-on-change | RB<7:4> | RB<7:0> |
| Comparator | 0 | 0 |
| USART | Y | Y |
| Extended WDT | N | N |
| Software Control Option of WDT/BOR | N | N |
| INTOSC Frequencies | None | 500 kHz - 16 MHz |
| Clock Switching | N | N |