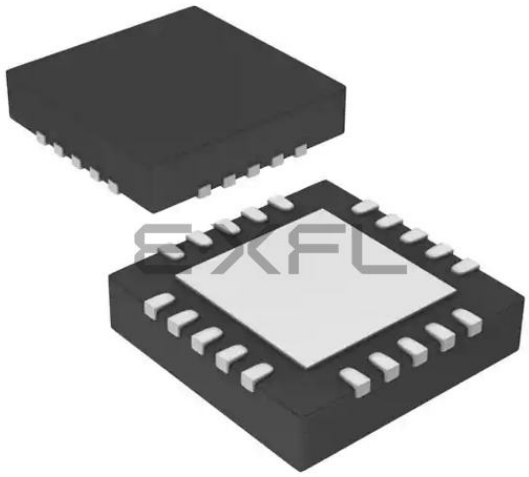


**Welcome to E-XFL.COM**

## 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 | 32MHz |
| Connectivity | I²C, LINbus, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 18 |
| Program Memory Size | 7KB (4K x 14) |
| Program Memory Type | FLASH |
| EEPROM Size | 128 x 8 |
| RAM Size | 512 x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.3V ~ 5.5V |
| Data Converters | A/D 12x10b; D/A 2x5b, 2x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 20-VFQFN Exposed Pad |
| Supplier Device Package | 20-QFN (4x4) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic16f1768t-i-ml |

**TABLE 1-2: PIC16(L)F1764/5 PINOUT DESCRIPTION (CONTINUED)**

| Name | Function | Input Type | Output Type | Description |
|---|---|---|---|---|
| RA5/T1CKI/T2IN/CLCIN3/ MD1MOD/SOSCI/OSC1/CLKIN | RA5 | TTL/ST | CMOS | General purpose I/O. |
| | T1CKI(1) | TTL/ST | — | Timer1 clock input. |
| | T2IN(1) | TTL/ST | — | Timer2 clock input. |
| | CLCIN3(1) | TTL/ST | — | CLC Input 3. |
| | MD1MOD(1) | TTL/ST | — | Data Signal Modulator modulation input. |
| | SOSCI | — | XTAL | Secondary Oscillator connection. |
| | OSC1 | XTAL | — | Crystal/Resonator (LP, XT, HS modes). |
| | CLKIN | ST | — | External Clock input (EC mode). |
| RC0/AN4/OPA1IN+/C2IN0+/ T5CKI/SCL/SCK | RC0 | TTL/ST | CMOS | General purpose I/O. |
| | AN4 | AN | | ADC Channel 4 input. |
| | OPA1IN+ | AN | — | Operational Amplifier 1 non-inverting input. |
| | C2IN0+ | AN | — | Comparator 2 positive input. |
| | T5CKI(1) | TTL/ST | — | Timer5 clock input. |
| | SCL(1,3) | $I^2C$ | — | $I^2C$ clock output. |
| | SCK(1) | TTL/ST | — | SPI clock input. |
| RC1/AN5/OPA1IN-/C1IN1-/ C2IN1-/T4IN/CLCIN2/SDI/SDA | RC1 | TTL/ST | CMOS | General purpose I/O. |
| | AN5 | AN | XTAL | ADC Channel 5 input. |
| | OPA1IN- | AN | — | Operational Amplifier 1 inverting input. |
| | C1IN1- | AN | — | Comparator 1 negative input. |
| | C2IN1- | AN | — | Comparator 2 negative input. |
| | T4IN(1) | TTL/ST | — | Timer4 clock input. |
| | CLCIN2(1) | TTL/ST | — | CLC Input 2. |
| | SDI(1) | TTL/ST | — | SPI data input. |
| | SDA(1) | $I^2C$ | — | $I^2C$ data output. |
| RC2/AN6/OPA1OUT/C1IN2-/ C2IN2-/PRG1IN0 | RC2 | TTL/ST | CMOS | General purpose I/O. |
| | AN6 | AN | — | ADC Channel 6 input. |
| | OPA1OUT | — | AN | Operational Amplifier 1 output. |
| | C1IN2- | AN | — | Comparator 1 negative input. |
| | C2IN2- | AN | — | Comparator 2 negative input. |
| | PRG1IN0 | AN | — | Ramp Generator 1 reference voltage input. |
| RC3/AN7/C1IN3-/C2IN3-/T5G/ CLCIN0/$\overline{SS}$ | RC3 | TTL/ST | CMOS | General purpose I/O. |
| | AN7 | AN | — | ADC Channel 7 input. |
| | C1IN3- | AN | — | Comparator 1 negative input. |
| | C2IN3- | AN | — | Comparator 2 negative input. |
| | T5G(1) | TTL/ST | — | Timer5 gate input. |
| | CLCIN0(1) | TTL/ST | — | CLC Input 0. |
| | $\overline{SS}$(1) | TTL/ST | — | SPI Slave Select input. |

**Legend:** AN = Analog input or output CMOS = CMOS compatible input or output OD = Open-Drain
TTL = TTL compatible input ST = Schmitt Trigger input with CMOS levels $I^2C$ = Schmitt Trigger input with $I^2C$
HV = High Voltage XTAL = Crystal levels

**Note 1:** Default peripheral input. Alternate pins can be selected as the peripheral input with the PPS Input Selection registers.

**2:** All pin digital outputs default to PORT latch data. Alternate outputs can be selected as peripheral digital outputs with the PPS Output Selection registers.

**3:** These peripheral functions are bidirectional. The output pin selections must be the same as the input pin selections.

**EXAMPLE 3-2: ACCESSING PROGRAM MEMORY VIA FSRn**

```
constants
   DW DATA0              ;First constant
   DW DATA1              ;Second constant
   DW DATA2
   DW DATA3
my_function
   ;… LOTS OF CODE…
   MOVLW   DATA_INDEX
   ADDLW   LOW constants
   MOVWF   FSR1L
   MOVLW   HIGH constants ;MSb sets
                           automatically
   MOVWF   FSR1H
   BTFSC   STATUS, C      ;carry from ADDLW?
   INCF    FSR1H, f       ;yes
   MOVIW   0[FSR1]
;THE PROGRAM MEMORY IS IN W
```

## 3.3 Data Memory Organization

The data memory is partitioned in 32 memory banks with 128 bytes in a bank. Each bank consists of (Figure 3-2):

- 12 core registers
- 20 Special Function Registers (SFRs)
- Up to 80 bytes of General Purpose RAM (GPR)
- 16 bytes of common RAM

The active bank is selected by writing the bank number into the Bank Select Register (BSR). Unimplemented memory will read as '0'. All data memory can be accessed either directly (via instructions that use the file registers) or indirectly via the two File Select Registers (FSRs). See **Section 3.6 "Indirect Addressing"** for more information.

Data memory uses a 12-bit address. The upper five bits of the address define the Bank address and the lower seven bits select the registers/RAM in that bank.

### 3.3.1 CORE REGISTERS

The core registers contain the registers that directly affect the basic operation. The core registers occupy the first 12 addresses of every data memory bank (addresses, x00h/x08h through x0Bh/x8Bh). These registers are listed below in Table 3-2. For detailed information, see Table 3-15.

**TABLE 3-2: CORE REGISTERS**

| Addresses | BANKx |
|---|---|
| x00h or x80h | INDF0 |
| x01h or x81h | INDF1 |
| x02h or x82h | PCL |
| x03h or x83h | STATUS |
| x04h or x84h | FSR0L |
| x05h or x85h | FSR0H |
| x06h or x86h | FSR1L |
| x07h or x87h | FSR1H |
| x08h or x88h | BSR |
| x09h or x89h | WREG |
| x0Ah or x8Ah | PCLATH |
| x0Bh or x8Bh | INTCON |

#### 3.3.1.1 STATUS Register

The STATUS register, shown in Register 3-1, contains:

- The arithmetic status of the ALU
- The Reset status

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 35.0 "Instruction Set Summary"**.

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

## TABLE 3-7: PIC16(L)F1764 MEMORY MAP (BANKS 8-23)

| BANK 8 | | BANK 9 | | BANK 10 | | BANK 11 | | BANK 12 | | BANK 13 | | BANK 14 | | BANK 15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 400h | Core Registers (Table 3-2) | 480h | Core Registers (Table 3-2) | 500h | Core Registers (Table 3-2) | 580h | Core Registers (Table 3-2) | 600h | Core Registers (Table 3-2) | 680h | Core Registers (Table 3-2) | 700h | Core Registers (Table 3-2) | 780h | Core Registers (Table 3-2) |
| 40Bh | | 48Bh | | 50Bh | | 58Bh | | 60Bh | | 68Bh | | 70Bh | | 78Bh | |
| 40Ch | — | 48Ch | — | 50Ch | — | 58Ch | — | 60Ch | — | 68Ch | — | 70Ch | — | 78Ch | — |
| 40Dh | — | 48Dh | — | 50Dh | — | 58Dh | — | 60Dh | — | 68Dh | COG1PHR | 70Dh | COG2PHR | 78Dh | — |
| 40Eh | HIDRVC | 48Eh | — | 50Eh | — | 58Eh | — | 60Eh | — | 68Eh | COG1PHF | 70Eh | COG2PHF | 78Eh | — |
| 40Fh | — | 48Fh | — | 50Fh | OPA1NCHS | 58Fh | — | 60Fh | — | 68Fh | COG1BLKR | 70Fh | COG2BLKR | 78Fh | — |
| 410h | — | 490h | — | 510h | OPA1PCHS | 590h | DACLD | 610h | — | 690h | COG1BLKF | 710h | COG2BLKF | 790h | — |
| 411h | — | 491h | — | 511h | OPA1CON | 591h | DAC1CON0 | 611h | — | 691h | COG1DBR | 711h | COG2DBR | 791h | — |
| 412h | — | 492h | — | 512h | OPA1ORS | 592h | DAC1REFL | 612h | — | 692h | COG1DBF | 712h | COG2DBF | 792h | — |
| 413h | T4TMR | 493h | TMR3L | 513h | — | 593h | DAC1REFH | 613h | — | 693h | COG1CON0 | 713h | COG2CON0 | 793h | — |
| 414h | T4PR | 494h | TMR3H | 514h | — | 594h | — | 614h | — | 694h | COG1CON1 | 714h | COG2CON1 | 794h | PRG1RTSS |
| 415h | T4CON | 495h | T3CON | 515h | — | 595h | — | 615h | — | 695h | COG1RIS0 | 715h | COG2RIS0 | 795h | PRG1FTSS |
| 416h | T4HLT | 496h | T3GCON | 516h | — | 596h | — | 616h | — | 696h | COG1RIS1 | 716h | COG2RIS1 | 796h | PRG1INS |
| 417h | T4CLKCON | 497h | — | 517h | — | 597h | DAC3CON0 | 617h | PWM3DCL | 697h | COG1RSIM0 | 717h | COG2RSIM0 | 797h | PRG1CON0 |
| 418h | T4RST | 498h | — | 518h | — | 598h | DAC3REF | 618h | PWM3DCH | 698h | COG1RSIM1 | 718h | COG2RSIM1 | 798h | PRG1CON1 |
| 419h | — | 499h | — | 519h | — | 599h | — | 619h | PWM3CON | 699h | COG1FIS0 | 719h | COG2FIS0 | 799h | PRG1CON2 |
| 41Ah | T6TMR | 49Ah | TMR5L | 51Ah | — | 59Ah | — | 61Ah | — | 69Ah | COG1FIS1 | 71Ah | COG2FIS1 | 79Ah | — |
| 41Bh | T6PR | 49Bh | TMR5H | 51Bh | — | 59Bh | — | 61Bh | — | 69Bh | COG1FSIM0 | 71Bh | COG2FSIM0 | 79Bh | — |
| 41Ch | T6CON | 49Ch | T5CON | 51Ch | — | 59Ch | — | 61Ch | — | 69Ch | COG1FSIM1 | 71Ch | COG2FSIM1 | 79Ch | — |
| 41Dh | T6HLT | 49Dh | T5GCON | 51Dh | — | 59Dh | — | 61Dh | — | 69Dh | COG1ASD0 | 71Dh | COG2ASD0 | 79Dh | — |
| 41Eh | T6CLKCON | 49Eh | — | 51Eh | — | 59Eh | — | 61Eh | — | 69Eh | COG1ASD1 | 71Eh | COG2ASD1 | 79Eh | — |
| 41Fh | T6RST | 49Fh | — | 51Fh | — | 59Fh | — | 61Fh | — | 69Fh | COG1STR | 71Fh | COG2STR | 79Fh | — |
| 420h | Unimplemented Read as '0' | 4A0h | Unimplemented Read as '0' | 520h | Unimplemented Read as '0' | 5A0h | Unimplemented Read as '0' | 620h | Unimplemented Read as '0' | 6A0h | Unimplemented Read as '0' | 720h | Unimplemented Read as '0' | 7A0h | Unimplemented Read as '0' |
| 46Fh | | 4EFh | | 56Fh | | 5EFh | | 66Fh | | 6EFh | | 76Fh | | 7EFh | |
| 470h | Accesses 70h-7Fh | 4F0h | Accesses 70h-7Fh | 570h | Accesses 70h-7Fh | 5F0h | Accesses 70h-7Fh | 670h | Accesses 70h-7Fh | 6F0h | Accesses 70h-7Fh | 770h | Accesses 70h-7Fh | 7F0h | Accesses 70h-7Fh |
| 47Fh | | 4FFh | | 57Fh | | 5FFh | | 67Fh | | 6FFh | | 77Fh | | 7FFh | |

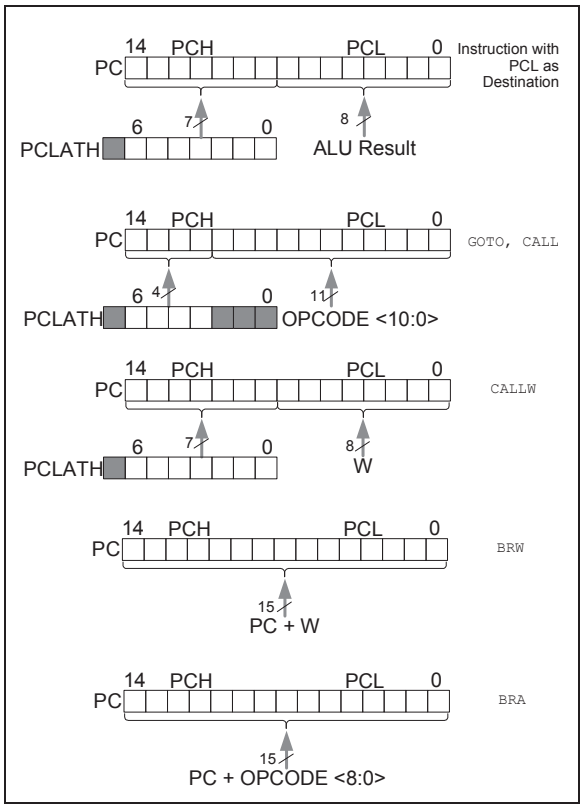
| BANK 16 | | BANK 17 | | BANK 18 | | BANK 19 | | BANK 20 | | BANK 21 | | BANK 22 | | BANK 23 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 800h | Core Registers (Table 3-2) | 880h | Core Registers (Table 3-2) | 900h | Core Registers (Table 3-2) | 980h | Core Registers (Table 3-2) | A00h | Core Registers (Table 3-2) | A80h | Core Registers (Table 3-2) | B00h | Core Registers (Table 3-2) | B80h | Core Registers (Table 3-2) |
| 80Bh | | 88Bh | | 90Bh | | 98Bh | | A0Bh | | A8Bh | | B0Bh | | B8Bh | |
| 80Ch | Unimplemented Read as '0' | 88Ch | Unimplemented Read as '0' | 90Ch | Unimplemented Read as '0' | 98Ch | Unimplemented Read as '0' | A0Ch | Unimplemented Read as '0' | A8Ch | Unimplemented Read as '0' | B0Ch | Unimplemented Read as '0' | B8Ch | Unimplemented Read as '0' |
| 86Fh | | 8EFh | | 96Fh | | 9EFh | | A6Fh | | AEFh | | B6Fh | | BEFh | |
| 870h | Accesses 70h-7Fh | 8F0h | Accesses 70h-7Fh | 970h | Accesses 70h-7Fh | 9F0h | Accesses 70h-7Fh | A70h | Accesses 70h-7Fh | AF0h | Accesses 70h-7Fh | B70h | Accesses 70h-7Fh | BF0h | Accesses 70h-7Fh |
| 87Fh | | 8FFh | | 97Fh | | 9FFh | | A7Fh | | AFFh | | B7Fh | | BFFh | |

**Legend:** ☐ = Unimplemented data memory locations, read as '0'.

## 3.4 PCL and PCLATH

The Program Counter (PC) is 15 bits wide. The low byte comes from the PCL register, which is a readable and writable register. The high byte (PC<14:8>) is not directly readable or writable and comes from PCLATH. On any Reset, the PC is cleared. Figure 3-3 shows the five situations for the loading of the PC.

**FIGURE 3-3: LOADING OF PC IN DIFFERENT SITUATIONS**




### 3.4.1 MODIFYING PCL

Executing any instruction with the PCL register as the destination simultaneously causes the Program Counter PC<14:8> bits (PCH) to be replaced by the contents of the PCLATH register. This allows the entire contents of the Program Counter to be changed by writing the desired upper seven bits to the PCLATH register. When the lower eight bits are written to the PCL register, all 15 bits of the Program Counter will change to the values contained in the PCLATH register and those being written to the PCL register.

### 3.4.2 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).

### 3.4.3 COMPUTED FUNCTION CALLS

A computed function CALL allows programs to maintain tables of functions and provide another way to execute state machines or look-up tables. When performing a table read using a computed function CALL, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block).

If using the CALL instruction, the PCH<2:0> and PCL registers are loaded with the operand of the CALL instruction. PCH<6:3> is loaded with PCLATH<6:3>.

The CALLW instruction enables computed calls by combining PCLATH and W to form the destination address. A computed CALLW is accomplished by loading the W register with the desired address and executing CALLW. The PCL register is loaded with the value of W and PCH is loaded with PCLATH.

### 3.4.4 BRANCHING

The branching instructions add an offset to the PC. This allows relocatable code and code that crosses page boundaries. There are two forms of branching, BRW and BRA. The PC will have incremented to fetch the next instruction in both cases. When using either branching instruction, a PCL memory boundary may be crossed.

If using BRW, load the W register with the desired unsigned address and execute BRW. The entire PC will be loaded with the address PC + 1 + W.

If using BRA, the entire PC will be loaded with PC + 1 +, the signed value of the operand of the BRA instruction.

## 5.4 Two-Speed Clock Start-up Mode

Two-Speed Clock Start-up mode provides additional power savings by minimizing the latency between external oscillator start-up and code execution. In applications that make heavy use of the Sleep mode, Two-Speed Start-up will remove the external oscillator start-up time from the time spent awake and can reduce the overall power consumption of the device. This mode allows the application to wake-up from Sleep, perform a few instructions using the Internal Oscillator Block, INTOSC, as the clock source and go back to Sleep without waiting for the external oscillator to become stable.

Two-Speed Start-up provides benefits when the oscillator module is configured for LP, XT or HS modes. The Oscillator Start-up Timer (OST) is enabled for these modes and must count 1024 oscillations before the oscillator can be used as the system clock source.

If the oscillator module is configured for any mode other than LP, XT or HS mode, then Two-Speed Start-up is disabled. This is because the external clock oscillator does not require any stabilization time after POR or an exit from Sleep.

If the OST count reaches 1024 before the device enters Sleep mode, the OSTS bit of the OSCSTAT register is set and program execution switches to the external oscillator. However, the system may never operate from the external oscillator if the time spent awake is very short.

| | |
|---|---|
| **Note:** | Executing a `SLEEP` instruction will abort the oscillator start-up time and will cause the OSTS bit of the OSCSTAT register to remain clear. |

### 5.4.1 TWO-SPEED START-UP MODE CONFIGURATION

Two-Speed Start-up mode is configured by the following settings:

- IESO (of the Configuration Words) = `1`; Internal/External Switchover bit (Two-Speed Start-up mode enabled).
- SCS<1:0> (of the OSCCON register) = `00`.
- FOSC<2:0> bits in the Configuration Words configured for LP, XT or HS mode.

Two-Speed Start-up mode is entered after:

- Power-on Reset (POR) and, if enabled, after Power-up Timer (PWRT) has expired, or
- Wake-up from Sleep.

**TABLE 5-1: OSCILLATOR SWITCHING DELAYS**

| Switch From | Switch To | Frequency | Oscillator Delay |
|---|---|---|---|
| Sleep | LFINTOSC<br>MFINTOSC<br>HFINTOSC(1) | 31 kHz<br>31.25 kHz-500 kHz<br>31.25 kHz-16 MHz | Oscillator Warm-up Delay (TWARM)(2) |
| Sleep | EC, RC(1) | DC- 32 MHz | 2 cycles |
| LFINTOSC | EC, RC(1) | DC-32 MHz | 1 Cycle of Each |
| Sleep | Secondary Oscillator<br>LP, XT, HS(1) | 32 kHz-20 MHz | 1024 Clock Cycles (OST) |
| Any Clock Source | MFINTOSC<br>HFINTOSC(1) | 31.25 kHz-500 kHz<br>31.25 kHz-16 MHz | 2 μs (approx.) |
| Any Clock Source | LFINTOSC | 31 kHz | 1 Cycle of Each |
| Any Clock Source | Secondary Oscillator | 32 kHz | 1024 Clock Cycles (OST) |
| PLL Inactive | PLL Active | 16-32 MHz | 2 ms (approx.) |

**Note 1:** PLL is inactive.

**2:** See Table 36-8.

**REGISTER 5-2: OSCSTAT: OSCILLATOR STATUS REGISTER**

| R-1/q | R-0/q | R-q/q | R-0/q | R-0/q | R-q/q | R-0/0 | R-0/q |
|---|---|---|---|---|---|---|---|
| SOSCR | PLLR | OSTS | HFIOFR | HFIOFL | MFIOFR | LFIOFR | HFIOFS |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **SOSCR:** Secondary Oscillator Ready bit
If T1OSCEN = 1:
1 = Secondary oscillator is ready
0 = Secondary oscillator is not ready
If T1OSCEN = 0:
1 = Secondary clock source is always ready

bit 6 **PLLR** 4x PLL Ready bit
1 = 4x PLL is ready
0 = 4x PLL is not ready

bit 5 **OSTS:** Oscillator Start-up Timer Status bit
1 = Running from the clock defined by the FOSC<2:0> bits of the Configuration Words
0 = Running from an internal oscillator (FOSC<2:0> = 100)

bit 4 **HFIOFR:** High-Frequency Internal Oscillator Ready bit
1 = HFINTOSC is ready
0 = HFINTOSC is not ready

bit 3 **HFIOFL:** High-Frequency Internal Oscillator Locked bit
1 = HFINTOSC is at least 2% accurate
0 = HFINTOSC is not 2% accurate

bit 2 **MFIOFR:** Medium Frequency Internal Oscillator Ready bit
1 = MFINTOSC is ready
0 = MFINTOSC is not ready

bit 1 **LFIOFR:** Low-Frequency Internal Oscillator Ready bit
1 = LFINTOSC is ready
0 = LFINTOSC is not ready

bit 0 **HFIOFS:** High-Frequency Internal Oscillator Stable bit
1 = HFINTOSC is at least 0.5% accurate
0 = HFINTOSC is not 0.5% accurate

## 7.3 Interrupts During Sleep

Some interrupts can be used to wake from Sleep. To wake from Sleep, the peripheral must be able to operate without the system clock. The interrupt source must have the appropriate interrupt enable bit(s) set prior to entering Sleep.

On waking from Sleep, if the GIE bit is also set, the processor will branch to the interrupt vector. Otherwise, the processor will continue executing instructions after the SLEEP instruction. The instruction directly after the SLEEP instruction will always be executed before branching to the ISR. Refer to **Section 8.0 "Power-Down Mode (Sleep)"** for more details.

## 7.4 INT Pin

The INT pin can be used to generate an asynchronous edge-triggered interrupt. This interrupt is enabled by setting the INTE bit of the INTCON register. The INTEDG bit of the OPTION_REG register determines on which edge the interrupt will occur. When the INTEDG bit is set, the rising edge will cause the interrupt. When the INTEDG bit is clear, the falling edge will cause the interrupt. The INTF bit of the INTCON register will be set when a valid edge appears on the INT pin. If the GIE and INTE bits are also set, the processor will redirect program execution to the interrupt vector.

## 7.5 Automatic Context Saving

Upon entering an interrupt, the return PC address is saved on the stack. Additionally, the following registers are automatically saved in the shadow registers:

- W register
- STATUS register (except for $\overline{TO}$ and $\overline{PD}$)
- BSR register
- FSR registers
- PCLATH register

Upon exiting the Interrupt Service Routine, these registers are automatically restored. Any modifications to these registers during the ISR will be lost. If modifications to any of these registers are desired, the corresponding shadow register should be modified and the value will be restored when exiting the ISR. The shadow registers are available in Bank 31 and are readable and writable. Depending on the user's application, other registers may also need to be saved.

**REGISTER 12-3: PPSLOCK: PPS LOCK REGISTER**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | PPSLOCKED |
| bit 7 | | | | | | | bit 0 |

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

bit 7-1 **Unimplemented:** Read as ‘0’

bit 0 **PPSLOCKED:** PPS Locked bit

1 = PPS is locked; PPS selections cannot be changed

0 = PPS is not locked; PPS selections can be changed

## 24.2 Compare Mode

The Compare mode function described in this section is available and identical for all CCP modules.

Compare mode makes use of the 16-bit Timer1 resource. The 16-bit value of the CCPRxH:CCPRxL register pair is constantly compared against the 16-bit value of the TMR1H:TMR1L register pair. When a match occurs, one of the following events can occur:

- Toggle the CCPx output
- Set the CCPx output
- Clear the CCPx output
- Pulse the CCPx output
- Generate a software interrupt
- Auto-conversion trigger

The action on the pin is based on the value of the MODE<3:0> control bits of the CCPxCON register. At the same time, the interrupt flag CCPxIF bit is set.

All Compare modes can generate an interrupt.

Figure 24-2 shows a simplified diagram of the compare operation.

### 24.2.1 AUTO-CONVERSION TRIGGER

When Auto-Conversion Trigger mode is chosen (CCPxM<3:0> = 1011), the CCPx module does the following:

- Resets Timer1
- Starts an ADC conversion if ADC is enabled

The CCPx module does not assert control of the CCPx pin in this mode.

The auto-conversion trigger output of the CCP occurs immediately upon a match between the TMR1H, TMR1L register pair and the CCPRxH, CCPRxL register pair. The TMR1H, TMR1L register pair is not reset until the next rising edge of the Timer1 clock. The auto-conversion trigger output starts an ADC conversion (if the ADC module is enabled). This allows the CCPRxH, CCPRxL register pair to effectively provide a 16-bit programmable period register for Timer1.

Refer to **Section 16.2.5 "Auto-Conversion Trigger"** for more information.

| | |
|---|---|
| **Note 1:** | The auto-conversion trigger from the CCP module does not set interrupt flag bit, TMR1IF of the PIR1 register. |
| **2:** | Removing the match condition by changing the contents of the CCPRxH and CCPRxL register pair, between the clock edge that generates the auto-conversion trigger and the clock edge that generates the Timer1 Reset, will preclude the Reset from occurring. |

### 24.2.2 CCPx PIN CONFIGURATION

The user must configure the CCPx pin as an output by clearing the associated TRISx bit.

The CCPx pin function can be moved to alternate pins using the PPS controls. See **Section 12.0 "Peripheral Pin Select (PPS) Module"** for more detail.

| | |
|---|---|
| **Note:** | Clearing the CCPxCON register will force the CCPx compare output latch to the default low level. This is not the PORT I/O data latch. |

**FIGURE 24-2: COMPARE MODE OPERATION BLOCK DIAGRAM**

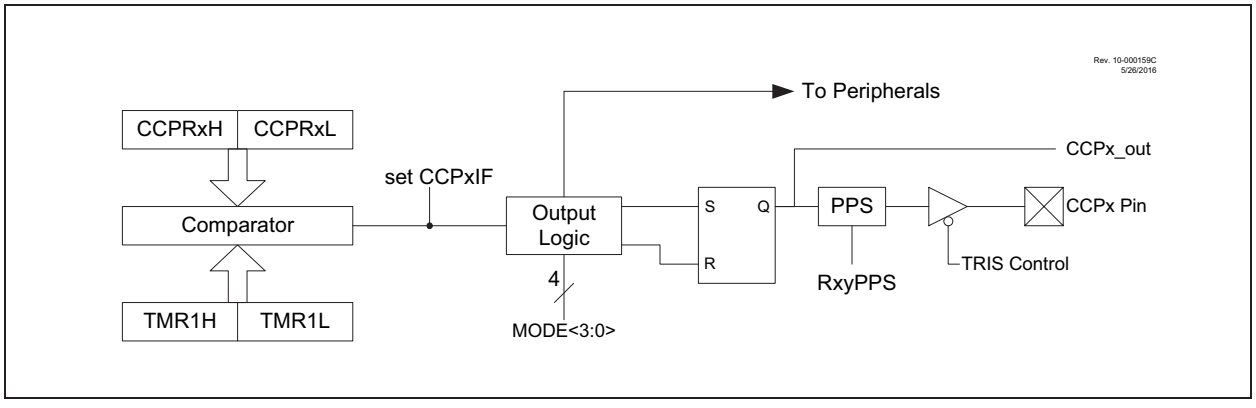


**FIGURE 26-4: STANDARD PWMx MODE TIMING DIAGRAM**

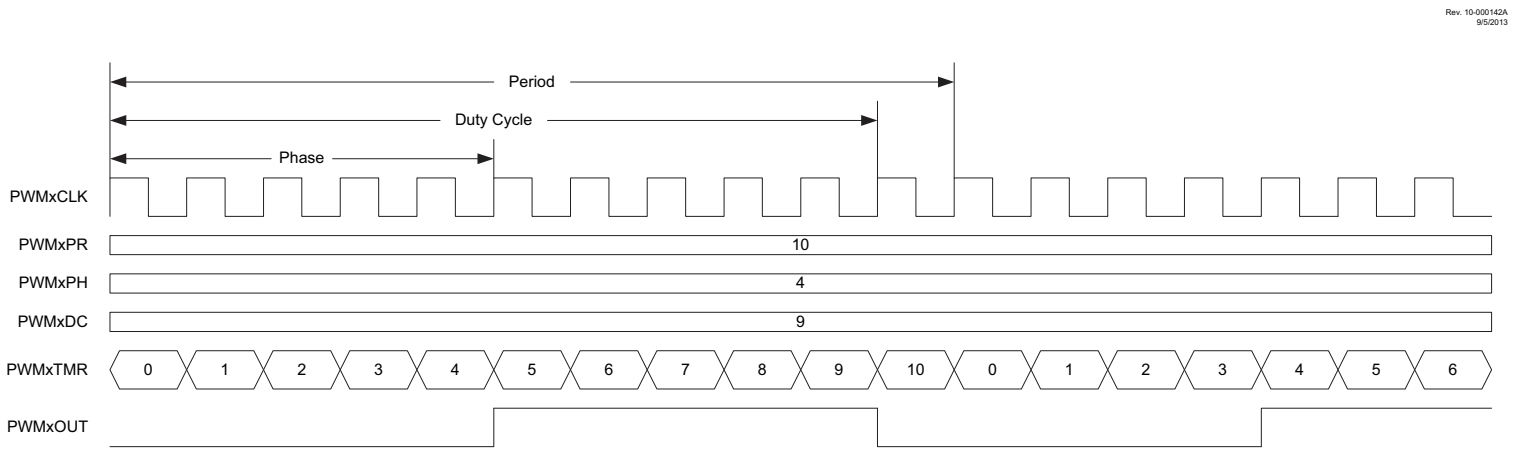
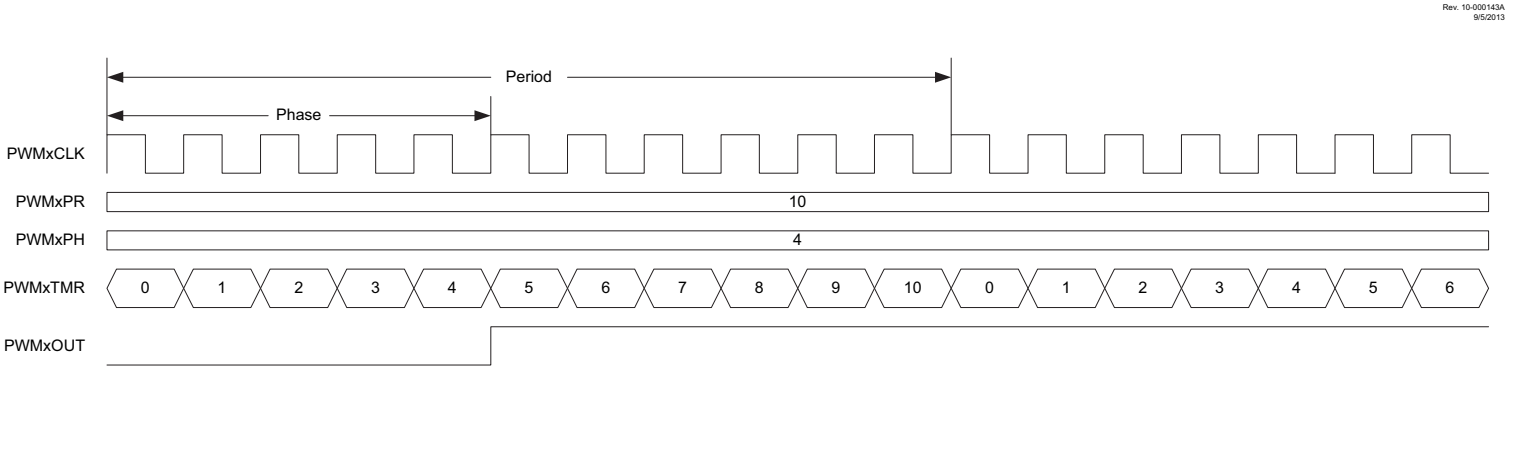



**FIGURE 26-5: SET ON MATCH PWMx MODE TIMING DIAGRAM**



**REGISTER 26-11: PWMxPRH: PWMx PERIOD COUNT HIGH 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 |
|---|---|---|---|---|---|---|---|
| PR<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

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

bit 7-0 **PR<15:8>**: PWMx Period High bits
Upper eight bits of PWMx period count.

**REGISTER 26-12: PWMxPRL: PWMx PERIOD COUNT LOW 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 |
|---|---|---|---|---|---|---|---|
| PR<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

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

bit 7-0 **PR<7:0>**: PWMx Period Low bits
Lower eight bits of PWMx period count.

**REGISTER 27-6: COGxRSIM1: COGx RISING EVENT SOURCE INPUT MODE REGISTER 1 (CONTINUED)**

bit 1 **RSIM9:** COGx Rising Event Input Source 9 Mode bit

RIS9 = 1:

1 = PWM5 output low-to-high transition will cause a rising event after rising event phase delay
0 = PWM5 output high level will cause an immediate rising event

RIS9 = 0:

PWM5 output has no effect on rising event.

bit 0 **RSIM8:** COGx Rising Event Input Source 8 Mode bit

RIS8 = 1:

1 = PWM4 output low-to-high transition will cause a rising event after rising event phase delay
0 = PWM4 output high level will cause an immediate rising event

RIS8 = 0:

PWM4 output has no effect on rising event.

**Note 1:** PIC16(L)F1768/9 only. Otherwise unimplemented, read as '0'.

**FIGURE 31-2: ON-OFF KEYING (OOK) SYNCHRONIZATION**

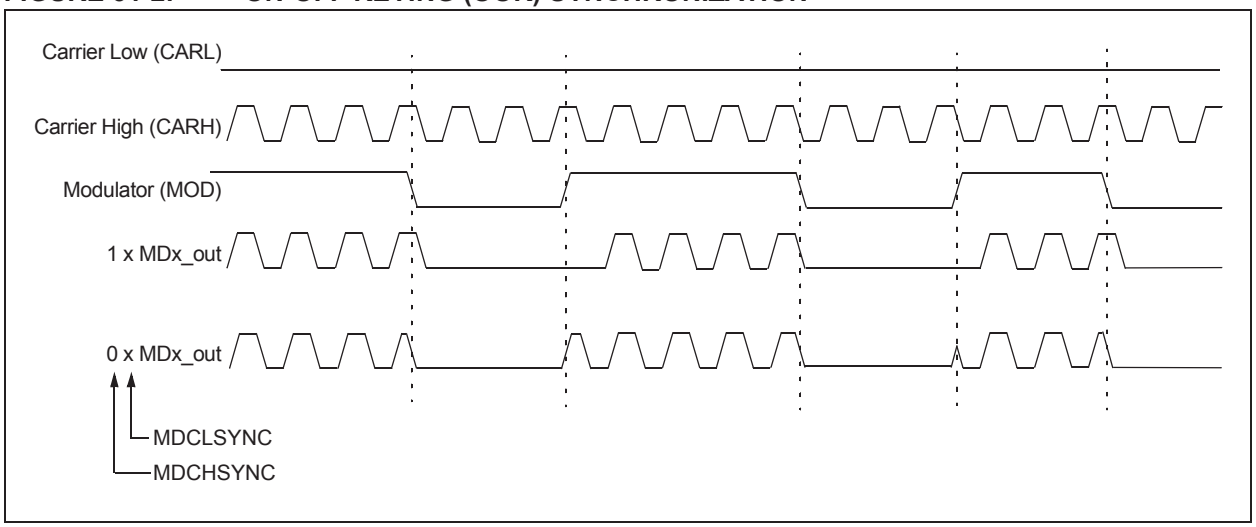



**FIGURE 31-3: NO SYNCHRONIZATION (MDCHSYNC = 0, MDCLSYNC = 0)**

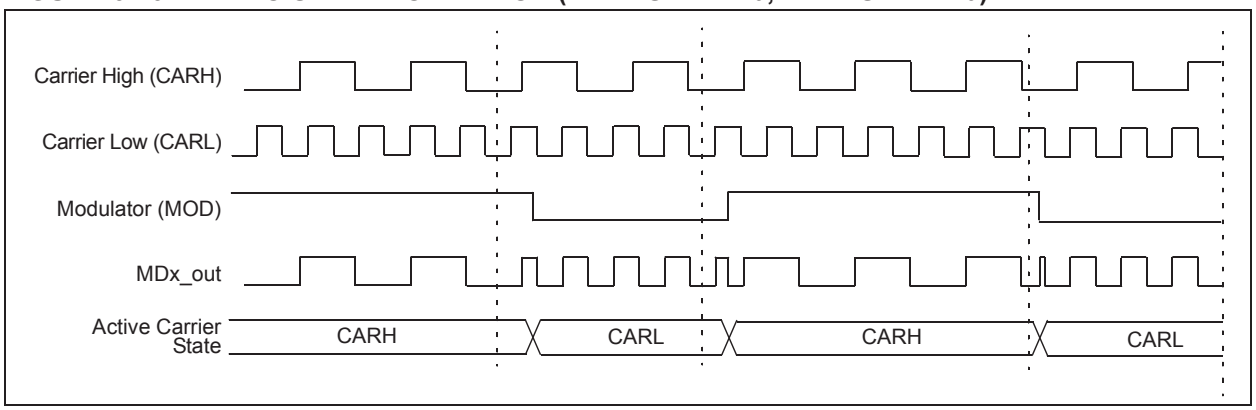
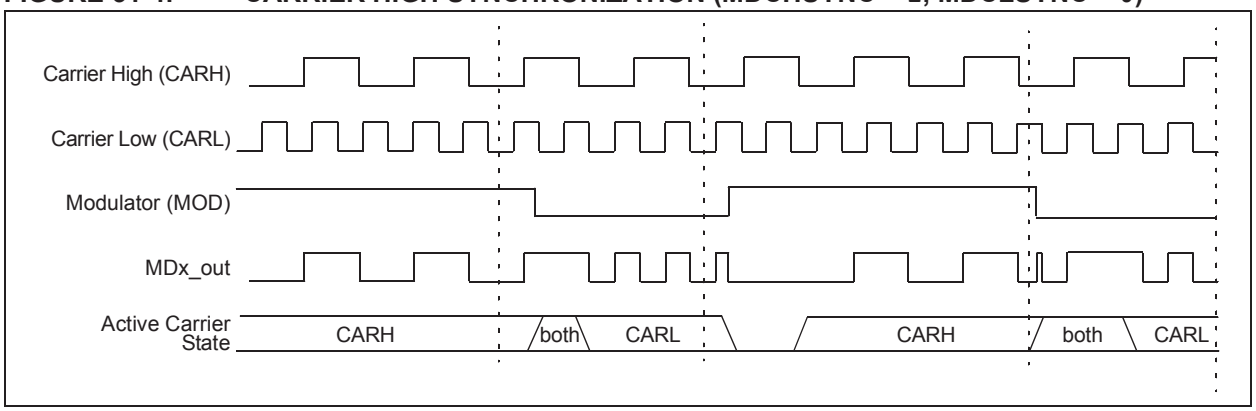



**FIGURE 31-4: CARRIER HIGH SYNCHRONIZATION (MDCHSYNC = 1, MDCLSYNC = 0)**



## 32.2 SPI Mode Overview

The Serial Peripheral Interface (SPI) bus is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is controlled through a chip select known as Slave Select.

The SPI bus specifies four signal connections:

- Serial Clock (SCK)
- Serial Data Out (SDO)
- Serial Data In (SDI)
- Slave Select ($\overline{SS}$)

Figure 32-1 shows the block diagram of the MSSP module when operating in SPI mode.

The SPI bus operates with a single master device and one or more slave devices. When multiple slave devices are used, an independent Slave Select connection is required from the master device to each slave device.

Figure 32-4 shows a typical connection between a master device and multiple slave devices.

The master selects only one slave at a time. Most slave devices have tri-state outputs so their output signal appears disconnected from the bus when they are not selected.

Transmissions involve two shift registers, eight bits in size, one in the master and one in the slave. With either the master or the slave device, data is always shifted out one bit at a time, with the Most Significant bit (MSb) shifted out first. At the same time, a new Least Significant bit (LSb) is shifted into the same register.

Figure 32-5 shows a typical connection between two processors configured as master and slave devices.

Data is shifted out of both shift registers on the programmed clock edge and latched on the opposite edge of the clock.

The master device transmits information out on its SDO output pin, which is connected to, and received by, the slave's SDI input pin. The slave device transmits information out on its SDO output pin, which is connected to, and received by, the master's SDI input pin.

To begin communication, the master device first sends out the clock signal. Both the master and the slave devices should be configured for the same clock polarity.

The master device starts a transmission by sending out the MSb from its shift register. The slave device reads this bit from that same line and saves it into the LSb position of its shift register.

During each SPI clock cycle, a full-duplex data transmission occurs. This means that while the master device is sending out the MSb from its shift register (on its SDO pin), and the slave device is reading this bit and saving it as the LSb of its shift register, that the slave device is also sending out the MSb from its shift register (on its SDO pin) and the master device is reading this bit and saving it as the LSb of its shift register.

After eight bits have been shifted out, the master and slave have exchanged register values.

If there is more data to exchange, the shift registers are loaded with new data and the process repeats itself.

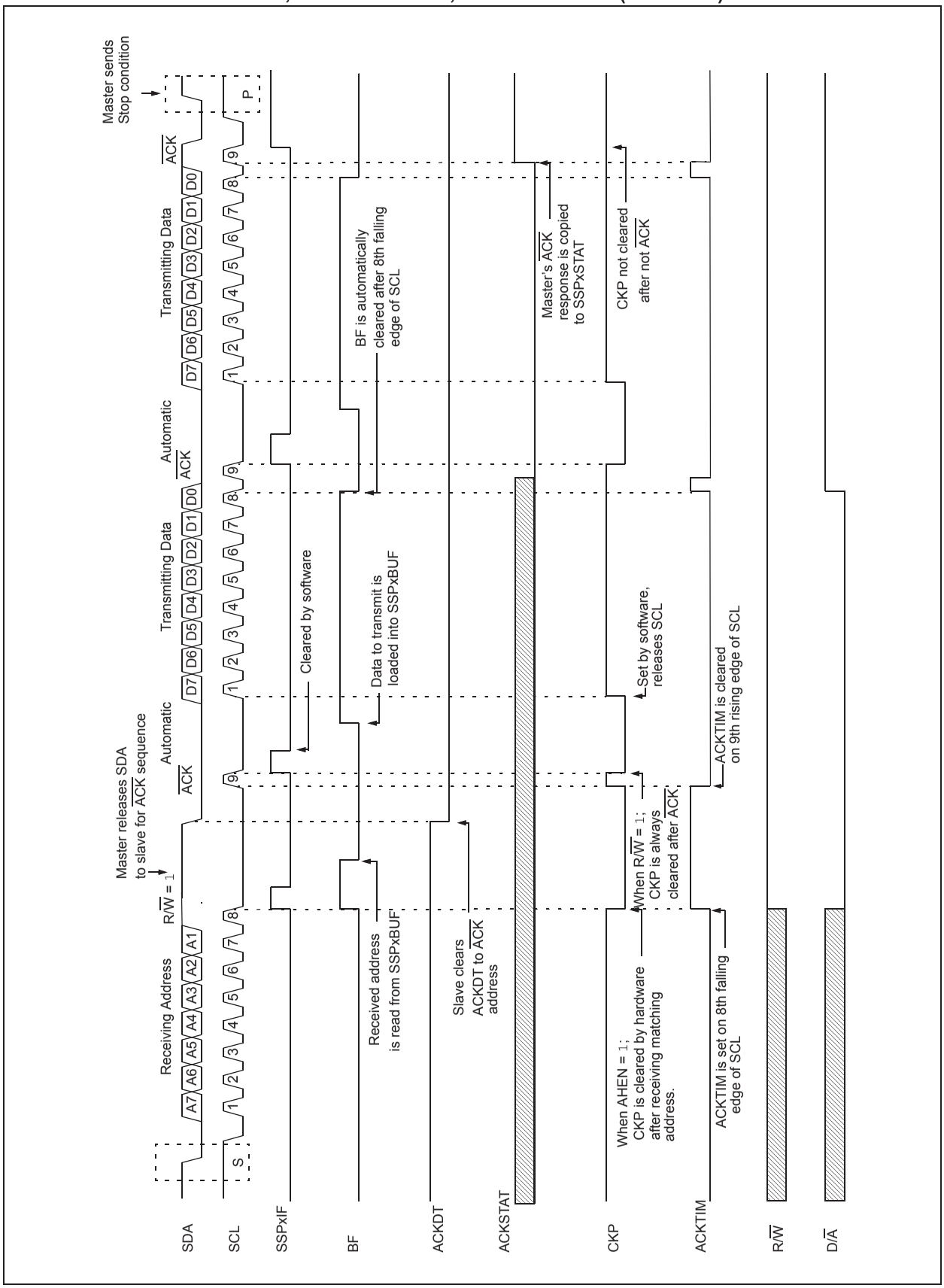
Whether the data is meaningful or not (dummy data), depends on the application software. This leads to three scenarios for data transmission:

- Master sends useful data and slave sends dummy data.
- Master sends useful data and slave sends useful data.
- Master sends dummy data and slave sends useful data.

Transmissions may involve any number of clock cycles. When there is no more data to be transmitted, the master stops sending the clock signal and it deselects the slave.

Every slave device connected to the bus that has not been selected through its Slave Select line must disregard the clock and transmission signals and must not transmit out any data of its own.

**FIGURE 32-19: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 1)**



## 32.7 BAUD RATE GENERATOR

The MSSP module has a Baud Rate Generator available for clock generation in both $I^2C$ and SPI Master modes. The Baud Rate Generator (BRG) reload value is placed in the SSPxADD register (Register 32-6). When a write occurs to SSPxBUF, the Baud Rate Generator will automatically begin counting down.

Once the given operation is complete, the internal clock will automatically stop counting and the clock pin will remain in its last state.

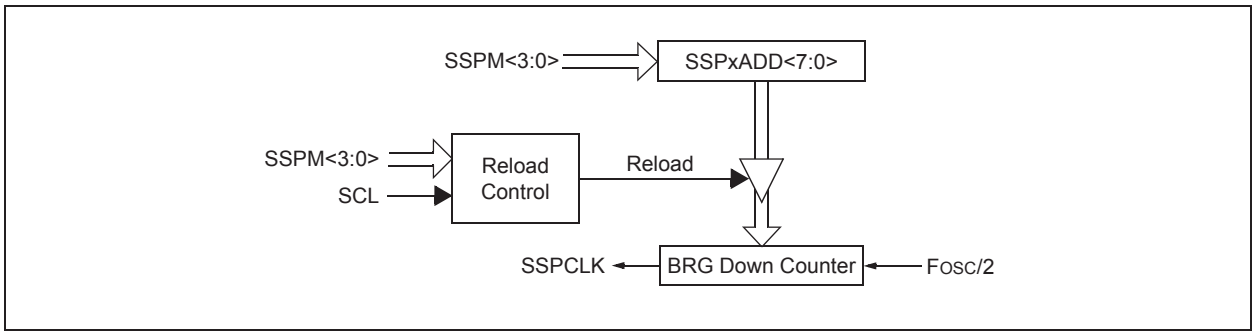
An internal signal “Reload” in Figure 32-40 triggers the value from SSPxADD to be loaded into the BRG counter. This occurs twice for each oscillation of the module clock line. The logic dictating when the reload signal is asserted depends on the mode the MSSP is being operated in.

Table 32-4 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD.

**EQUATION 32-1:**

$$F_{CLOCK} = \frac{F_{OSC}}{(SSPxADD + 1)(4)}$$

**FIGURE 32-40: BAUD RATE GENERATOR BLOCK DIAGRAM**




**Note:** Values of 0x00, 0x01 and 0x02 are not valid for SSPxADD when used as a Baud Rate Generator for $I^2C$; this is an implementation limitation.

**TABLE 32-4: MSSP CLOCK RATE w/BRG**

| FOSC | FCY | BRG Value | FCLOCK (2 Rollovers of BRG) |
|---|---|---|---|
| 32 MHz | 8 MHz | 13h | 400 kHz |
| 32 MHz | 8 MHz | 19h | 308 kHz |
| 32 MHz | 8 MHz | 4Fh | 100 kHz |
| 16 MHz | 4 MHz | 09h | 400 kHz |
| 16 MHz | 4 MHz | 0Ch | 308 kHz |
| 16 MHz | 4 MHz | 27h | 100 kHz |
| 4 MHz | 1 MHz | 09h | 100 kHz |

**Note:** Refer to the I/O port electrical specifications in Table 36-10 and Figure 36-7 to ensure the system is designed to support I/O timing requirements.

**REGISTER 33-3: BAUD1CON: BAUD RATE CONTROL REGISTER**

| R-0/0 | R-1/1 | U-0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **ABDOVF:** Auto-Baud Detect Overflow bit
Asynchronous mode:
1 = Auto-baud timer overflowed
0 = Auto-baud timer did not overflow
Synchronous mode:
Don't care.

bit 6 **RCIDL**: Receive Idle Flag bit
Asynchronous mode:
1 = Receiver is Idle
0 = Start bit has been received and the receiver is receiving
Synchronous mode:
Don't care.

bit 5 **Unimplemented:** Read as '0'

bit 4 **SCKP**: Synchronous Clock Polarity Select bit
Asynchronous mode:
1 = Transmit inverted data to the TX/CK pin
0 = Transmit non-inverted data to the TX/CK pin
Synchronous mode:
1 = Data is clocked on rising edge of the clock
0 = Data is clocked on falling edge of the clock

bit 3 **BRG16:** 16-bit Baud Rate Generator bit
1 = 16-bit Baud Rate Generator is used
0 = 8-bit Baud Rate Generator is used

bit 2 **Unimplemented:** Read as '0'

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = Receiver is waiting for a falling edge, no character will be received, byte RCIF will be set; WUE will automatically clear after RCIF is set
0 = Receiver is operating normally
Synchronous mode:
Don't care.

bit 0 **ABDEN**: Auto-Baud Detect Enable bit
Asynchronous mode:
1 = Auto-Baud Detect mode is enabled (clears when auto-baud is complete)
0 = Auto-Baud Detect mode is disabled
Synchronous mode:
Don't care.

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

| BRA | Relative Branch |
|---|---|
| Syntax: | [ *label* ] BRA label<br>[ *label* ] BRA $+k |
| Operands: | -256 ≤ label - PC + 1 ≤ 255<br>-256 ≤ k ≤ 255 |
| Operation: | (PC) + 1 + k → PC |
| Status Affected: | None |
| Description: | Add the signed 9-bit literal 'k' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 1 + k. This instruction is a 2-cycle instruction. This branch has a limited range. |

| BRW | Relative Branch with W |
|---|---|
| Syntax: | [ *label* ] BRW |
| Operands: | None |
| Operation: | (PC) + (W) → PC |
| Status Affected: | None |
| Description: | Add the contents of W (unsigned) to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 1 + (W). This instruction is a 2-cycle instruction. |

| BSF | Bit Set f |
|---|---|
| Syntax: | [ *label* ] BSF f,b |
| Operands: | 0 ≤ f ≤ 127<br>0 ≤ b ≤ 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 ≤ f ≤ 127<br>0 ≤ b ≤ 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. |

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

## 38.11 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.

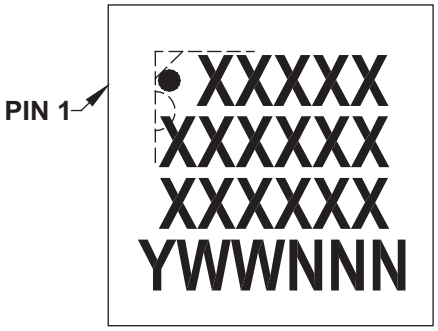
## 38.12 Third-Party Development Tools

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®

**Package Marking Information (Continued)**

20-Lead QFN (4x4x0.9 mm)




Example