



Welcome to [E-XFL.COM](https://www.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 | I ² C, LINbus, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 36 |
| Program Memory Size | 14KB (8K x 14) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 1K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 3.6V |
| Data Converters | A/D 34x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 125°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 44-TQFP |
| Supplier Device Package | 44-TQFP (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic16lf1567-e-pt |

PIN DIAGRAMS

FIGURE 1: 28-PIN SPDIP, SOIC, SSOP DIAGRAM FOR PIC16LF1566

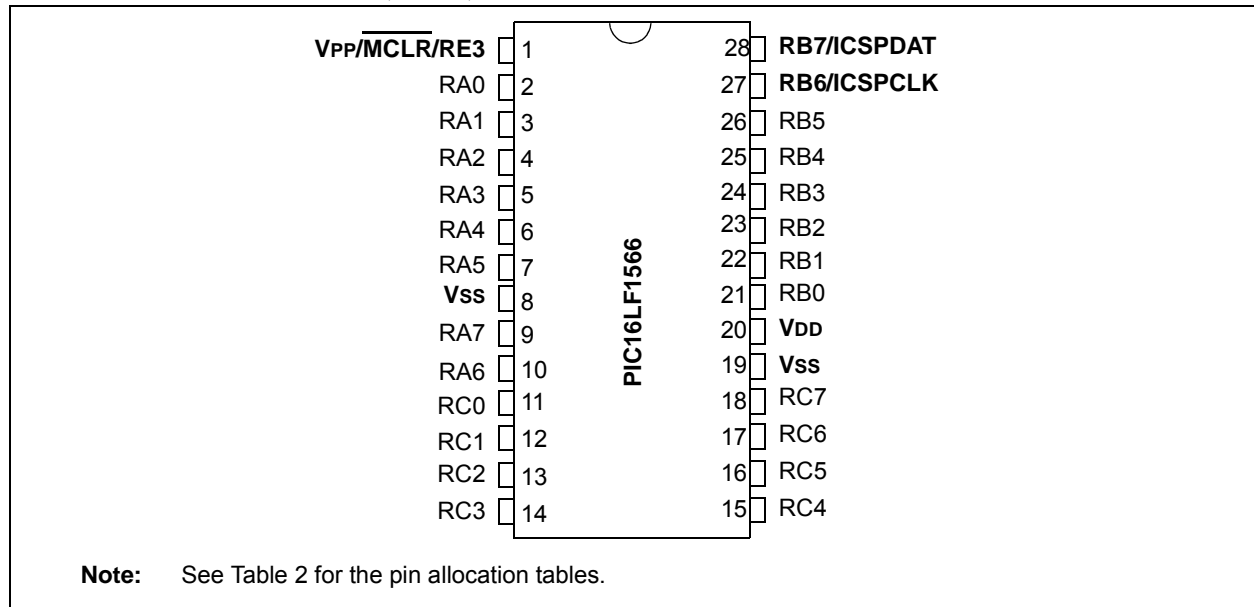
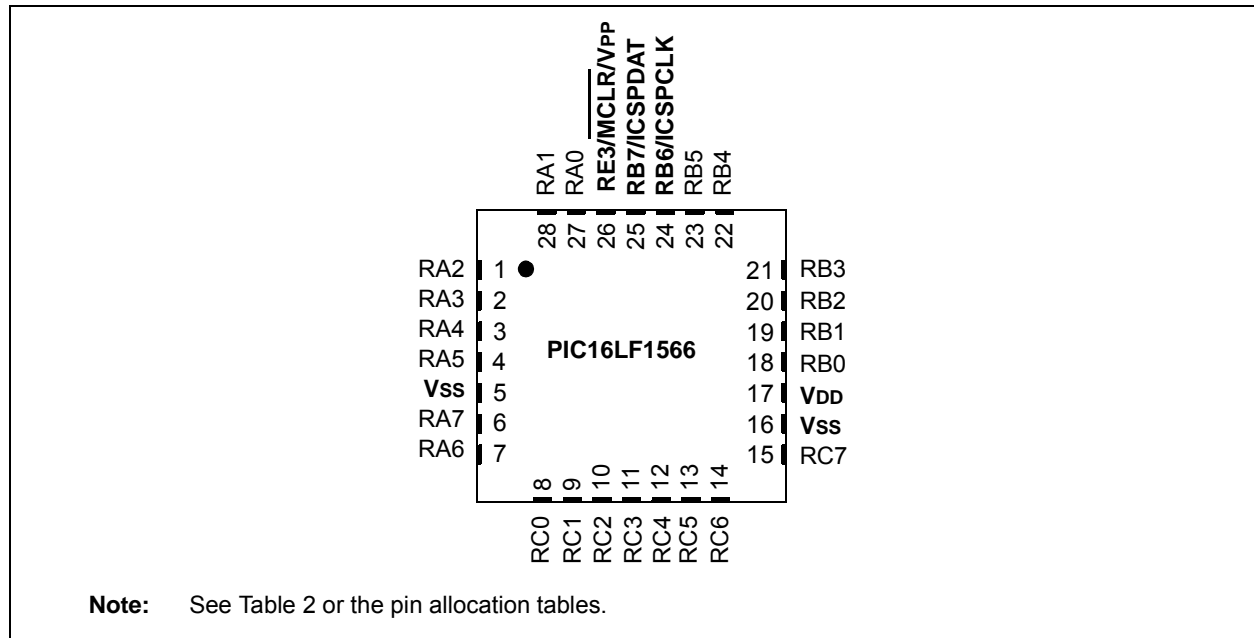


FIGURE 2: 28-PIN UQFN DIAGRAM FOR PIC16LF1566



1.0 DEVICE OVERVIEW

The PIC16LF1566/1567 devices are described within this data sheet. The block diagram of these devices is shown in Figure 1-1, the available peripherals are shown in Table 1-1 and the pinout descriptions are shown in Table 1-2 and Table 1-3.

TABLE 1-1: DEVICE PERIPHERAL SUMMARY

| Peripheral | | PIC16LF1566 | PIC16LF1567 |
|---|--------|-------------|-------------|
| Analog-to-Digital Converter (ADC) | | | |
| | ADC1 | • | • |
| | ADC2 | • | • |
| Hardware Capacitive Voltage Divider (CVD) | | • | • |
| Enhanced Universal Synchronous/Asynchronous Receiver/Transmitter (EUSART) | | • | • |
| Fixed Voltage Reference (FVR) | | • | • |
| Temperature Indicator | | • | • |
| Master Synchronous Serial Ports | | | |
| | MSSP1 | • | • |
| | MSSP2 | • | • |
| PWM Modules | | | |
| | PWM1 | • | • |
| | PWM2 | • | • |
| Timers | | | |
| | Timer0 | • | • |
| | Timer1 | • | • |
| | Timer2 | • | • |
| | Timer4 | • | • |

TABLE 1-2: PIC16LF1566 PINOUT DESCRIPTION (CONTINUED)

| Name | Function | Input Type | Output Type | Description |
|---------------------------|-----------------|------------------|-------------|---------------------------------------|
| RC4/AN14/SDA1/SDI1 | RC4 | TTL | CMOS | General Purpose I/O. |
| | AN14 | AN | — | ADC Channel Input for ADC1. |
| | SDA1 | I ² C | OD | I ² C Data for MSSP1. |
| | SDI1 | CMOS | — | SPI Data Input for MSSP1. |
| RC5/AN25/SDO1/I2CLVL | RC5 | TTL | CMOS | General Purpose I/O. |
| | AN25 | AN | — | ADC Channel Input for ADC2. |
| | SDO1 | — | CMOS | SPI Data Output for MSSP1. |
| | I2CLVL | AN | — | I ² C Voltage Level Input. |
| RC6/AN15/TX/CK | RC6 | TTL | — | General Purpose I/O. |
| | AN15 | AN | — | ADC Channel Input for ADC1. |
| | TX | — | CMOS | EUSART Asynchronous Transmit. |
| | CK | ST | CMOS | EUSART Synchronous Clock. |
| RC7/AN26/RX/DT | RC7 | TTL | CMOS | General Purpose I/O. |
| | AN26 | AN | — | ADC Channel Input for ADC2. |
| | RX | ST | — | EUSART Asynchronous Input. |
| | DT | ST | CMOS | EUSART Synchronous Data. |
| RE3/V _{PP} /MCLR | RE3 | TTL | — | General Purpose Input with WPU. |
| | V _{PP} | HV | — | Programming Voltage. |
| | MCLR | ST | — | Master Clear with Internal Pull-up. |

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 I²C = Schmitt Trigger input with I²C levels
HV = High Voltage XTAL = Crystal

Note 1: Alternate pin function selected with the APFCON (Register 11-1) register.

PIC16LF1566/1567

TABLE 1-3: PIC16LF1567 PINOUT DESCRIPTION (CONTINUED)

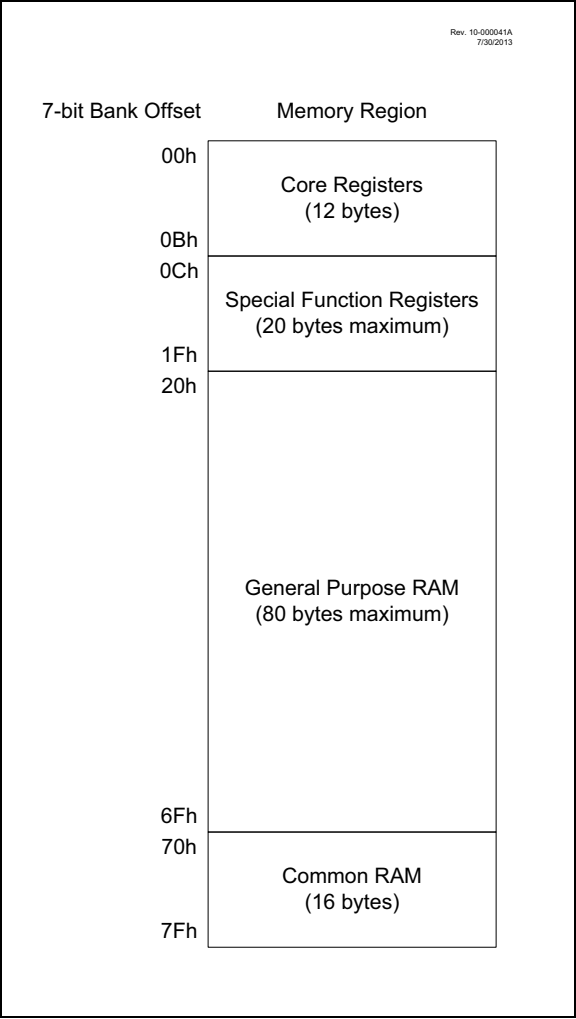
| Name | Function | Input Type | Output Type | Description |
|------|----------|------------|-------------|-------------|
|------|----------|------------|-------------|-------------|

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 I²C = Schmitt Trigger input with I²C levels
HV = High Voltage XTAL = Crystal

Note 1: Alternate pin function selected with the APFCON (Register 11-1) register.

PIC16LF1566/1567

FIGURE 3-2: BANKED MEMORY PARTITIONING



PIC16LF1566/1567

TABLE 3-11: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

| Addr. | 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 | |
|--------------------|-----------------------|--|--|------------|-----------------|-----------------|-------------|---------|--------|-----------------------|---------------------------------|-----------|
| Bank 31 | | | | | | | | | | | | |
| F80h | INDF0 ⁽¹⁾ | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu | |
| F81h | INDF1 ⁽¹⁾ | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu | |
| F82h | PCL ⁽¹⁾ | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 | |
| F83h | STATUS ⁽¹⁾ | — | — | — | \overline{TO} | \overline{PD} | Z | DC | C | ---1 1000 | ---q quuu | |
| F84h | FSR0L ⁽¹⁾ | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu | |
| F85h | FSR0H ⁽¹⁾ | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 | |
| F86h | FSR1L ⁽¹⁾ | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu | |
| F87h | FSR1H ⁽¹⁾ | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 | |
| F88h | BSR ⁽¹⁾ | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 | |
| F89h | WREG ⁽¹⁾ | Working Register | | | | | | | | 0000 0000 | uuuu uuuu | |
| F8Ah | PCLATH ⁽¹⁾ | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | | -000 0000 | -000 0000 |
| F8Bh | INTCON ⁽¹⁾ | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 | |
| F8Ch | ICDIO | PORT_ICDDAT | PORT_ICDCLK | LAT_ICDDAT | LAT_ICDCLK | TRIS_ICDDAT | TRIS_ICDCLK | — | — | xxxxxx-- | | |
| F8Dh | ICDCON0 | INBUG | FREEZ | SSTEP | — | DBGINEX | — | — | RSTVEC | xxx-x--x | | |
| F8Eh to F90h | — | Unimplemented | | | | | | | | — | — | |
| F91h | ICDSTAT | TRP1HLTF | TRP0HLTF | — | — | — | — | USRHLTF | — | xx----x- | | |
| F92h to F95h | — | Unimplemented | | | | | | | | — | — | |
| F96h | ICDINSTL | DBGIN7 | DBGIN6 | DBGIN5 | DBGIN4 | DBGIN3 | DBGIN2 | DBGIN1 | DBGIN0 | xxxxxxxx | | |
| F97h | ICDINSTH | — | — | DBGIN13 | DBGIN12 | DBGIN11 | DBGIN10 | DBGIN9 | DBGIN8 | --xxxxxx | | |
| F98h to F9Bh | — | Unimplemented | | | | | | | | — | — | |
| F9Ch | ICDBK0CON | BKEN | — | — | — | — | — | — | BKHLT | x-----x | | |
| F9Dh | ICDBK0L | BAK7 | BAK6 | BAK5 | BAK4 | BAK3 | BAK2 | BAK1 | BAK0 | xxxxxxxx | | |
| F9Eh | ICDBK0H | — | BAK14 | BAK13 | BAK12 | BAK11 | BAK10 | BAK9 | BAK8 | -xxxxxxxx | | |
| F9Fh | — | Unimplemented | | | | | | | | — | — | |
| FA0h to FBFh | — | Unimplemented | | | | | | | | — | — | |
| FC0h to FCFh | — | Unimplemented | | | | | | | | — | — | |
| FD0h to FE2h | — | Unimplemented | | | | | | | | — | — | |
| FE3h | BSRICDSHAD | — | — | — | BSR_ICDSHAD | | | | | ---xxxxx | — | |
| FE4h | STATUS_-SHAD | — | — | — | — | — | Z_SHAD | DC_SHAD | C_SHAD | ---- -xxx | ---- -uuu | |
| FE5h | WREG_SHAD | WREG_SHAD | | | | | | | | xxxx xxxx | uuuu uuuu | |
| FE6h | BSR_SHAD | — | — | — | BSR_SHAD | | | | | ---x xxxx | ---u uuuu | |
| FE7h | PCLATH_-SHAD | — | PCLATH_SHAD | | | | | | | | -xxx xxxx | uuuu uuuu |
| FE8h | FSR0L_SHAD | FSR0L_SHAD | | | | | | | | xxxx xxxx | uuuu uuuu | |
| FE9h | FSR0H_SHAD | FSR0H_SHAD | | | | | | | | xxxx xxxx | uuuu uuuu | |
| FEAh | FSR1L_SHAD | FSR1L_SHAD | | | | | | | | xxxx xxxx | uuuu uuuu | |
| FEBh | FSR1H_SHAD | FSR1H_SHAD | | | | | | | | xxxx xxxx | uuuu uuuu | |

Legend: x = unknown, u = unchanged, q = depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations unimplemented, read as '0'.

- Note**
- 1: These registers can be accessed from any bank.
 - 2: PIC16LF1567.
 - 3: These registers/bits are available at two address locations, in Bank 1 and Bank 14.
 - 4: PIC16LF1566 only.
 - 5: Unimplemented, read as '1'.

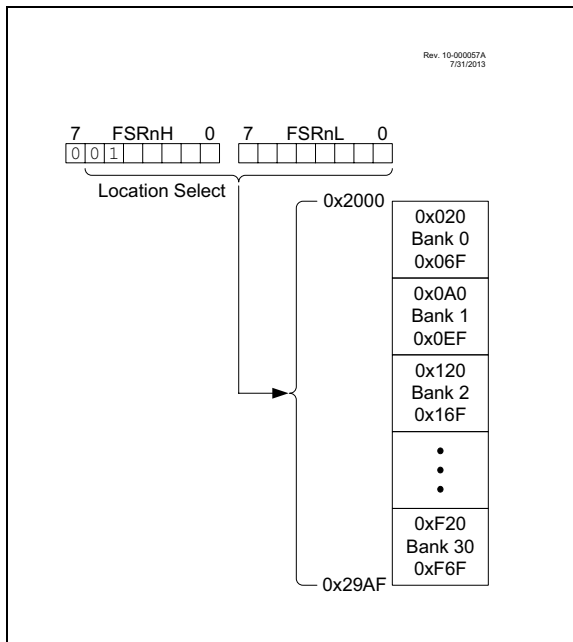
3.5.2 LINEAR DATA MEMORY

The linear data memory is the region from FSR address 0x2000 to FSR address 0x29AF. This region is a virtual region that points back to the 80-byte blocks of GPR memory in all the banks.

Unimplemented memory reads as 0x00. Use of the linear data memory region allows buffers to be larger than 80 bytes because incrementing the FSR beyond one bank will go directly to the GPR memory of the next bank.

The 16 bytes of common memory are not included in the linear data memory region.

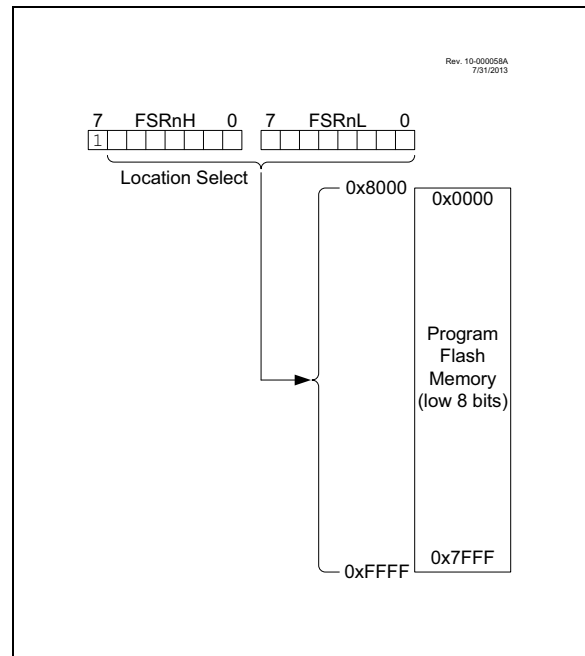
FIGURE 3-10: LINEAR DATA MEMORY MAP



3.5.3 PROGRAM FLASH MEMORY

To make constant data access easier, the entire program Flash memory is mapped to the upper half of the FSR address space. When the MSb of FSRnH is set, the lower 15 bits are the address in program memory which will be accessed through INDF. Only the lower eight bits of each memory location is accessible via INDF. Writing to the program Flash memory cannot be accomplished via the FSR/INDF interface. All instructions that access program Flash memory via the FSR/INDF interface will require one additional instruction cycle to complete.

FIGURE 3-11: PROGRAM FLASH MEMORY MAP



PIC16LF1566/1567

5.2 Clock Source Types

Clock sources can be classified as external or internal.

External clock sources rely on external circuitry for the clock source to function. Examples are: oscillator modules (EC mode).

Internal clock sources are contained within the oscillator module. The oscillator block has two internal oscillators that are used to generate two system clock sources: the 16 MHz High-Frequency Internal Oscillator (HFINTOSC) and the 31 kHz Low-Frequency Internal Oscillator (LFINTOSC).

The system clock can be selected between external or internal clock sources via the System Clock Select (SCS) bits in the OSCCON register. See **Section 5.3 “Clock Switching”** for additional information.

5.2.1 EXTERNAL CLOCK SOURCES

An external clock source can be used as the device system clock by performing one of the following actions:

- Program the FOSC<1:0> bits in the Configuration Words to select an external clock source that will be used as the default system clock upon a device Reset.
- Clear the SCS<1:0> bits in the OSCCON register to switch the system clock source to:
 - An external clock source determined by the value of the FOSC bits.

See **Section 5.3 “Clock Switching”** for more information.

5.2.1.1 EC Mode

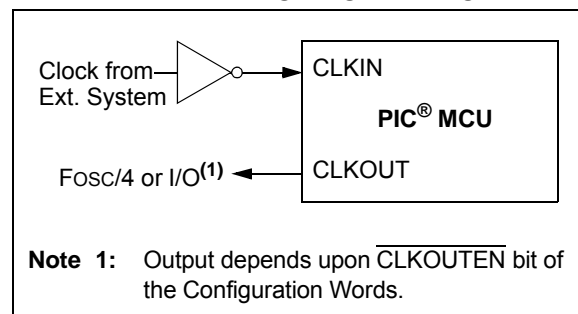
The External Clock (EC) mode allows an externally generated logic level signal to be the system clock source. When operating in this mode, an external clock source is connected to the CLKIN input. CLKOUT is available for general purpose I/O or CLKOUT. Figure 5-2 shows the pin connections for EC mode.

EC mode has three power modes to select from through Configuration Words:

- High power, 4-20 MHz (FOSC = 11)
- Medium power, 0.5-4 MHz (FOSC = 10)
- Low power, 0-0.5 MHz (FOSC = 01)

When EC mode is selected, there is no delay in operation after a Power-on Reset (POR) or wake-up from Sleep. Because the PIC® MCU design is fully static, stopping the external clock input will have the effect of halting the device while leaving all data intact. Upon restarting the external clock, the device will resume operation as if no time had elapsed.

FIGURE 5-2: EXTERNAL CLOCK (EC) MODE OPERATION



5.2.2 INTERNAL CLOCK SOURCES

The device may be configured to use the internal oscillator block as the system clock by performing either of the following actions:

- Program the FOSC<1:0> bits in Configuration Words to select the INTOSC clock source, which will be used as the default system clock upon a device Reset.
- Set the SCS<1:0> bits in the OSCCON register to '1x' to switch the system clock source to the internal oscillator during run-time. See **Section 5.3 "Clock Switching"** for more information.

In INTOSC mode, the CLKIN pin is available for general purpose I/O. The CLKOUT pin is available for general purpose I/O or CLKOUT.

The function of the CLKOUT pin is determined by the CLKOUTEN bit in Configuration Words.

The internal oscillator block has two independent oscillators.

1. The HFINTOSC (High-Frequency Internal Oscillator) is factory calibrated and operates at 16 MHz.
2. The LFINTOSC (Low-Frequency Internal Oscillator) is uncalibrated and operates at 31 kHz.

5.2.2.1 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a factory calibrated 16 MHz internal clock source.

The outputs of the HFINTOSC connects to a prescaler and multiplexer (see Figure 5-1). One of multiple frequencies derived from the HFINTOSC can be selected via software using the IRCF<3:0> bits of the OSCCON register. See **Section 5.2.2.4 "Internal Oscillator Clock Switch Timing"** for more information.

The HFINTOSC is enabled by:

- Configure the IRCF<3:0> bits of the OSCCON register for the desired HF frequency, and
- FOSC<1:0> = 00, or
- Set the System Clock Source (SCS) bits of the OSCCON register to '1x'.

A fast start-up oscillator allows internal circuits to power-up and stabilize before switching to HFINTOSC.

The High-Frequency Internal Oscillator Ready bit (HFIOFR) of the OSCSTAT register indicates when the HFINTOSC is running.

The High-Frequency Internal Oscillator Stable bit (HFIOFS) of the OSCSTAT register indicates when the HFINTOSC is running within 0.5% of its final value.

5.2.2.2 LFINTOSC

The Low-Frequency Internal Oscillator (LFINTOSC) is an uncalibrated 31 kHz internal clock source.

The output of the LFINTOSC connects to a multiplexer (see Figure 5-1). Select 31 kHz, via software, using the IRCF<3:0> bits of the OSCCON register. See **Section 5.2.2.4 "Internal Oscillator Clock Switch Timing"** for more information. The LFINTOSC is also the source for the Power-up Timer (PWRT) and Watchdog Timer (WDT).

The LFINTOSC is enabled by selecting 31 kHz (IRCF<3:0> bits of the OSCCON register = 000x) as the system clock source (SCS bits of the OSCCON register = 1x), or when any of the following are enabled:

- Configure the IRCF<3:0> bits of the OSCCON register for the LF frequency, and
- FOSC<1:0> = 00, or
- Set the System Clock Source (SCS) bits of the OSCCON register to '1x'

Peripherals that use the LFINTOSC are:

- Power-up Timer (PWRT)
- Watchdog Timer (WDT)

The Low-Frequency Internal Oscillator Ready bit (LFIOFR) of the OSCSTAT register indicates when the LFINTOSC is running.

PIC16LF1566/1567

5.2.2.3 Internal Oscillator Frequency Selection

The system clock speed can be selected via software using the Internal Oscillator Frequency Select bits $IRCF<3:0>$ of the $OSCCON$ register.

The outputs of the 16 MHz HFINTOSC postscaler and the LFINTOSC connect to a multiplexer (see Figure 5-1). The Internal Oscillator Frequency Select bits $IRCF<3:0>$ of the $OSCCON$ register select the frequency. One of the following frequencies can be selected via software:

- 32 MHz (requires 4x PLL)
- 16 MHz
- 8 MHz
- 4 MHz
- 2 MHz
- 1 MHz
- 500 kHz (default after Reset)
- 250 kHz
- 125 kHz
- 62.5 kHz
- 31.25 kHz
- 31 kHz (LFINTOSC)

Note: Following any Reset, the $IRCF<3:0>$ bits of the $OSCCON$ register are set to '0111' and the frequency selection is set to 500 kHz. The user can modify the $IRCF$ bits to select a different frequency.

The $IRCF<3:0>$ bits of the $OSCCON$ register allow duplicate selections for some frequencies. These duplicate choices can offer system design trade-offs. Lower power consumption can be obtained when changing oscillator sources for a given frequency. Faster transition times can be obtained between frequency changes that use the same oscillator source.

5.2.2.4 Internal Oscillator Clock Switch Timing

When switching between the HFINTOSC and the LFINTOSC, the new oscillator may already be shut down to save power (see Figure 5-3). If this is the case, there is a delay after the $IRCF<3:0>$ bits of the $OSCCON$ register are modified before the frequency selection takes place. The $OSCSTAT$ register will reflect the current active status of the HFINTOSC and LFINTOSC oscillators. The sequence of a frequency selection is as follows:

1. $IRCF<3:0>$ bits of the $OSCCON$ register are modified.
2. If the new clock is shut down, a clock start-up delay is started.
3. Clock switch circuitry waits for a falling edge of the current clock.
4. Clock switch is complete.

See Figure 5-3 for more details.

If the internal oscillator speed is switched between two clocks of the same source, there is no start-up delay before the new frequency is selected.

Start-up delay specifications are located in the oscillator tables of **Section 25.0 "Electrical Specifications"**.

5.2.2.5 32 MHz Internal Oscillator Frequency Selection

The Internal Oscillator Block can be used with the 4x PLL to produce a 32 MHz internal system clock source. The following settings are required to use the 32 MHz internal clock source:

- The $FOSC$ bits in Configuration Word 1 must be set to use the INTOSC source as the device system clock ($FOSC<1:0> = 00$).
- The SCS bits in the $OSCCON$ register must be cleared to use the clock determined by $FOSC<1:0>$ in Configuration Word 1 ($SCS<1:0> = 00$).
- The $IRCF$ bits in the $OSCCON$ register must be set to the 8 MHz HFINTOSC set to use ($IRCF<3:0> = 1110$).
- The $SPLLEN$ bit in the $OSCCON$ register must be set to enable the 4x PLL.

The 4x PLL is not available for use with the internal oscillator when the SCS bits of the $OSCCON$ register are set to '1x'. The SCS bits must be set to '00' to use the 4x PLL with the internal oscillator.

PIC16LF1566/1567

7.1 Operation

Interrupts are disabled upon any device Reset. They are enabled by setting the following bits:

- GIE bit of the INTCON register
- Interrupt enable bit(s) for the specific interrupt event(s)
- PEIE bit of the INTCON register (if the interrupt enable bit of the interrupt event is contained in the PIE1 and PIE2 registers)

The INTCON, PIR1 and PIR2 registers record individual interrupts via interrupt flag bits. Interrupt flag bits will be set, regardless of the status of the GIE, PEIE and individual interrupt enable bits.

The following events happen when an interrupt event occurs while the GIE bit is set:

- Current prefetched instruction is flushed
- GIE bit is cleared
- Current Program Counter (PC) is pushed onto the stack
- Critical registers are automatically saved to the shadow registers (See “**Section 7.5 “Automatic Context Saving”**.”)
- PC is loaded with the interrupt vector 0004h

The firmware within the Interrupt Service Routine (ISR) should determine the source of the interrupt by polling the interrupt flag bits. The interrupt flag bits must be cleared before exiting the ISR to avoid repeated interrupts. Because the GIE bit is cleared, any interrupt that occurs while executing the ISR will be recorded through its interrupt flag, but will not cause the processor to redirect to the interrupt vector.

The RETFIE instruction exits the ISR by popping the previous address from the stack, restoring the saved context from the shadow registers and setting the GIE bit.

For additional information on a specific interrupt's operation, refer to its peripheral chapter.

Note 1: Individual interrupt flag bits are set, regardless of the state of any other enable bits.

2: All interrupts will be ignored while the GIE bit is cleared. Any interrupt occurring while the GIE bit is clear will be serviced when the GIE bit is set again.

10.0 FLASH PROGRAM MEMORY CONTROL

The Flash program memory is readable and writable during normal operation over the full V_{DD} range. Program memory is indirectly addressed using Special Function Registers (SFRs). The SFRs used to access program memory are:

- PMCON1
- PMCON2
- PMDATL
- PMDATH
- PMADRL
- PMADRH

When accessing the program memory, the PMDATH:PMDATL register pair forms a 2-byte word that holds the 14-bit data for read/write, and the PMADRH:PMADRL register pair forms a 2-byte word that holds the 15-bit address of the program memory location being read.

The write time is controlled by an on-chip timer. The write/erase voltages are generated by an on-chip charge pump rated to operate over the operating voltage range of the device.

The Flash program memory can be protected in two ways; by code protection (\overline{CP} bit in Configuration Words) and write protection ($WRT<1:0>$ bits in Configuration Words).

Code protection ($\overline{CP} = 0$)⁽¹⁾, disables access, reading and writing, to the Flash program memory via external device programmers. Code protection does not affect the self-write and erase functionality. Code protection can only be reset by a device programmer performing a Bulk Erase to the device, clearing all Flash program memory, Configuration bits and User IDs.

Write protection prohibits self-write and erase to a portion or all of the Flash program memory, as defined by the bits $WRT<1:0>$. Write protection does not affect a device programmers ability to read, write or erase the device.

Note 1: Code protection of the entire Flash program memory array is enabled by clearing the \overline{CP} bit of Configuration Words.

10.1 PMADRL and PMADRH Registers

The PMADRH:PMADRL register pair can address up to a maximum of 32K words of program memory. When selecting a program address value, the MSB of the address is written to the PMADRH register and the LSB is written to the PMADRL register.

10.1.1 PMCON1 AND PMCON2 REGISTERS

PMCON1 is the control register for Flash program memory accesses.

Control bits RD and WR initiate read and write, respectively. These bits cannot be cleared, only set, in software. They are cleared by hardware at completion of the read or write operation. The inability to clear the WR bit in software prevents the accidental, premature termination of a write operation.

The WREN bit, when set, will allow a write operation to occur. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a Reset during normal operation. In these situations, following Reset, the user can check the WRERR bit and execute the appropriate error handling routine.

The PMCON2 register is a write-only register. Attempting to read the PMCON2 register will return all '0's.

To enable writes to the program memory, a specific pattern (the unlock sequence), must be written to the PMCON2 register. The required unlock sequence prevents inadvertent writes to the program memory write latches and Flash program memory.

10.2 Flash Program Memory Overview

It is important to understand the Flash program memory structure for erase and programming operations. Flash program memory is arranged in rows. A row consists of a fixed number of 14-bit program memory words. A row is the minimum size that can be erased by user software.

After a row has been erased, the user can reprogram all or a portion of this row. Data to be written into the program memory row is written to 14-bit wide data write latches. These write latches are not directly accessible to the user, but may be loaded via sequential writes to the PMDATH:PMDATL register pair.

Note: If the user wants to modify only a portion of a previously programmed row, then the contents of the entire row must be read and saved in RAM prior to the erase. Then, new data and retained data can be written into the write latches to reprogram the row of Flash program memory. However, any unprogrammed locations can be written without first erasing the row. In this case, it is not necessary to save and rewrite the other previously programmed locations.

See Table 10-1 for erase row size and the number of write latches for Flash program memory.

TABLE 10-1: FLASH MEMORY ORGANIZATION BY DEVICE

| Device | Row Erase (words) | Write Latches (words) |
|-------------|-------------------|-----------------------|
| PIC16LF1566 | 32 | 32 |
| PIC16LF1567 | | |

PIC16LF1566/1567

10.2.2 FLASH MEMORY UNLOCK SEQUENCE

The unlock sequence is a mechanism that protects the Flash program memory from unintended self-write programming or erasing. The sequence must be executed and completed without interruption to successfully complete any of the following operations:

- Row Erase
- Load program memory write latches
- Write of program memory write latches to program memory
- Write of program memory write latches to User IDs

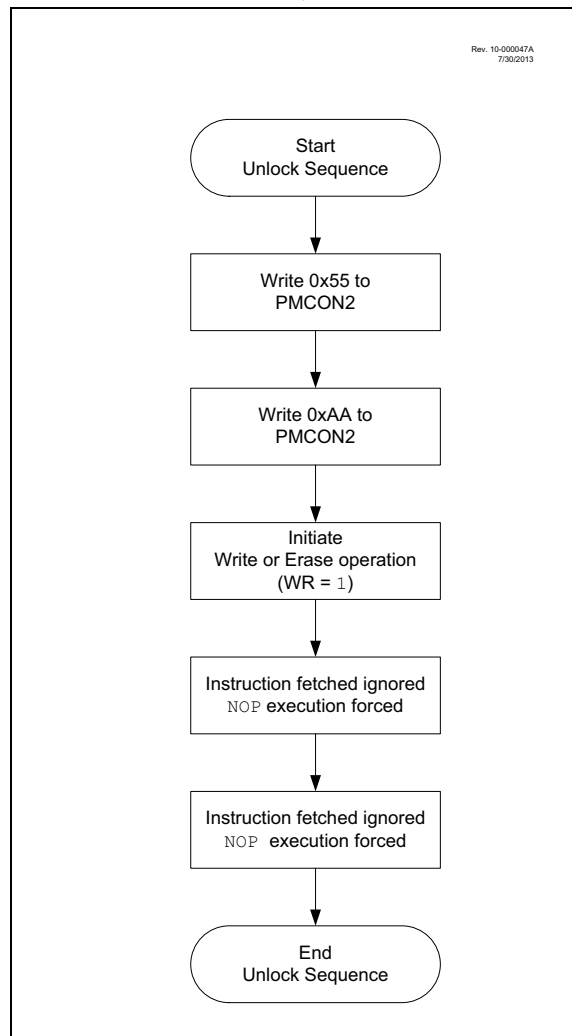
The unlock sequence consists of the following steps:

1. Write 55h to PMCON2
2. Write AAh to PMCON2
3. Set the WR bit in PMCON1
4. NOP instruction
5. NOP instruction

Once the WR bit is set, the processor will always force two NOP instructions. When an Erase Row or Program Row operation is being performed, the processor will stall internal operations (typical 2 ms), until the operation is complete and then resume with the next instruction. When the operation is loading the program memory write latches, the processor will always force the two NOP instructions and continue uninterrupted with the next instruction.

Since the unlock sequence must not be interrupted, global interrupts should be disabled prior to the unlock sequence and re-enabled after the unlock sequence is completed.

FIGURE 10-3: FLASH PROGRAM MEMORY UNLOCK SEQUENCE FLOWCHART



11.6 Register Definitions: PORTB

REGISTER 11-6: PORTB: PORTB REGISTER

| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
|---------|---------|---------|---------|---------|---------|---------|---------|
| RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 |
| 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 7-0 **RB<7:0>**: PORTB I/O Value bits⁽¹⁾
 1 = Port pin is $\geq V_{IH}$
 0 = Port pin is $\leq V_{IL}$

Note 1: Writes to PORTB are actually written to corresponding LATB register. Reads from PORTB register is return of actual I/O pin values.

REGISTER 11-7: TRISB: PORTB TRI-STATE REGISTER

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 |
| 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 7-0 **TRISB<7:0>**: PORTB Tri-State Control bits
 1 = PORTB pin configured as an input (tri-stated)
 0 = PORTB pin configured as an output

20.4.9 ACKNOWLEDGE SEQUENCE

The ninth SCLx pulse for any transferred byte in I²C is dedicated as an Acknowledge. It allows receiving devices to respond back to the transmitter by pulling the SDAx line low. The transmitter must release control of the line during this time to shift in the response. The Acknowledge (ACK) is an active-low signal, pulling the SDAx line low indicated to the transmitter that the device has received the transmitted data and is ready to receive more.

The result of an $\overline{\text{ACK}}$ is placed in the ACKSTAT bit of the SSPxCON2 register.

Slave software, when the AHEN and DHEN bits are set, allow the user to set the $\overline{\text{ACK}}$ value sent back to the transmitter. The ACKDT bit of the SSPxCON2 register is set/cleared to determine the response.

Slave hardware will generate an $\overline{\text{ACK}}$ response if the AHEN and DHEN bits of the SSPxCON3 register are clear.

There are certain conditions where an $\overline{\text{ACK}}$ will not be sent by the slave. If the BF bit of the SSPxSTAT register or the SSPOV bit of the SSPxCON1 register are set when a byte is received.

When the module is addressed, after the eighth falling edge of SCLx on the bus, the ACKTIM bit of the SSPxCON3 register is set. The ACKTIM bit indicates the acknowledge time of the active bus. The ACKTIM Status bit is only active when the AHEN bit or DHEN bit is enabled.

20.5 I²C SLAVE MODE OPERATION

The MSSPx Slave mode operates in one of four modes selected in the SSPM bits of SSPxCON1 register. The modes can be divided into 7-bit and 10-bit Addressing mode. 10-bit Addressing modes operate the same as 7-bit with some additional overhead for handling the larger addresses.

Modes with Start and Stop bit interrupts operated the same as the other modes with SSPxIF additionally getting set upon detection of a Start, Restart, or Stop condition.

20.5.1 SLAVE MODE ADDRESSES

The SSPxADD register (Register 20-6) contains the Slave mode address. The first byte received after a Start or Restart condition is compared against the value stored in this register. If the byte matches, the value is loaded into the SSPxBUF register and an interrupt is generated. If the value does not match, the module goes idle and no indication is given to the software that anything happened.

The SSPx Mask register (Register 20-5) affects the address matching process. See **Section 20.5.9 “SSPx Mask Register”** for more information.

20.5.1.1 I²C Slave 7-bit Addressing Mode

In 7-bit Addressing mode, the LSb of the received data byte is ignored when determining if there is an address match.

20.5.1.2 I²C Slave 10-bit Addressing Mode

In 10-bit Addressing mode, the first received byte is compared to the binary value of '1 1 1 1 0 A9 A8 0'. A9 and A8 are the two MSb's of the 10-bit address and stored in bits 2 and 1 of the SSPxADD register.

After the acknowledge of the high byte the UA bit is set and SCLx is held low until the user updates SSPxADD with the low address. The low address byte is clocked in and all eight bits are compared to the low address value in SSPxADD. Even if there is not an address match; SSPxIF and UA are set, and SCLx is held low until SSPxADD is updated to receive a high byte again. When SSPxADD is updated the UA bit is cleared. This ensures the module is ready to receive the high address byte on the next communication.

A high and low address match as a write request is required at the start of all 10-bit addressing communication. A transmission can be initiated by issuing a Restart once the slave is addressed, and clocking in the high address with the $\overline{\text{R/W}}$ bit set. The slave hardware will then acknowledge the read request and prepare to clock out data. This is only valid for a slave after it has received a complete high and low address byte match.

21.1.2.4 Receive Framing Error

Each character in the receive FIFO buffer has a corresponding framing error Status bit. A framing error indicates that a Stop bit was not seen at the expected time. The framing error status is accessed via the FERR bit of the RCSTA register. The FERR bit represents the status of the top unread character in the receive FIFO. Therefore, the FERR bit must be read before reading the RCREG.

The FERR bit is read-only and only applies to the top unread character in the receive FIFO. A framing error (FERR = 1) does not preclude reception of additional characters. It is not necessary to clear the FERR bit. Reading the next character from the FIFO buffer will advance the FIFO to the next character and the next corresponding framing error.

The FERR bit can be forced clear by clearing the SPEN bit of the RCSTA register, which resets the EUSART. Clearing the CREN bit of the RCSTA register does not affect the FERR bit. A framing error by itself does not generate an interrupt.

| |
|--|
| Note: If all receive characters in the receive FIFO have framing errors, repeated reads of the RCREG will not clear the FERR bit. |
|--|

21.1.2.5 Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before the FIFO is accessed. When this happens the OERR bit of the RCSTA register is set. The characters already in the FIFO buffer can be read but no additional characters will be received until the error is cleared. The error must be cleared by either clearing the CREN bit of the RCSTA register or by resetting the EUSART by clearing the SPEN bit of the RCSTA register.

21.1.2.6 Receiving 9-Bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCSTA register is set, the EUSART will shift nine bits into the RSR for each character received. The RX9D bit of the RCSTA register is the ninth and Most Significant data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the eight Least Significant bits from the RCREG.

21.1.2.7 Address Detection

A special Address Detection mode is available for use when multiple receivers share the same transmission line, such as in RS-485 systems. Address detection is enabled by setting the ADDEN bit of the RCSTA register.

Address detection requires 9-bit character reception. When address detection is enabled, only characters with the ninth data bit set will be transferred to the receive FIFO buffer, thereby setting the RCIF interrupt bit. All other characters will be ignored.

Upon receiving an address character, user software determines if the address matches its own. Upon address match, user software must disable address detection by clearing the ADDEN bit before the next Stop bit occurs. When user software detects the end of the message, determined by the message protocol used, software places the receiver back into the Address Detection mode by setting the ADDEN bit.

FIGURE 21-12: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)

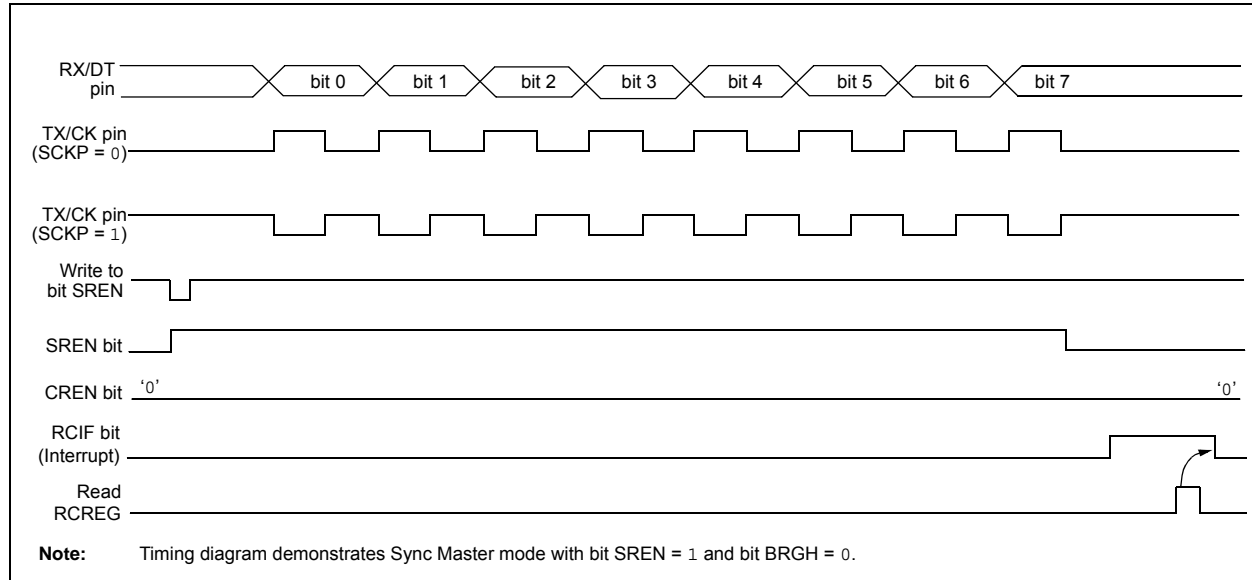


TABLE 21-8: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---------|------------------------------|--------|--------|--------|--------|--------|--------|--------|------------------|
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 266 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| RCREG | EUSART Receive Data Register | | | | | | | | 260* |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 265 |
| SPBRGL | BRG<7:0> | | | | | | | | 267* |
| SPBRGH | BRG<15:8> | | | | | | | | 267* |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 119 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDER | BRGH | TRMT | TX9D | 264 |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for synchronous master reception.

* Page provides register information.

PIC16LF1566/1567

TABLE 22-4: SUMMARY OF REGISTERS ASSOCIATED WITH PWM

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---------|-------------------------------|--------------|---------|---------|--------|--------|-------------|--------|------------------|
| PR2 | Timer2 module Period Register | | | | | | | | 193* |
| PWM1CON | PWM1EN | PWM1OE | PWM1OUT | PWM1POL | — | — | — | — | 286 |
| PWM1DCH | PWM1DCH<7:0> | | | | | | | | 287 |
| PWM1DCL | PWM1DCL<7:6> | | — | — | — | — | — | — | 287 |
| PWM2CON | PWM2EN | PWM2OE | PWM2OUT | PWM2POL | — | — | — | — | 286 |
| PWM2DCH | PWM2DCH<7:0> | | | | | | | | 287 |
| PWM2DCL | PWM2DCL<7:6> | | — | — | — | — | — | — | 287 |
| T2CON | — | T2OUTPS<3:0> | | | | TMR2ON | T2CKPS<1:0> | | 195 |
| TMR2 | Timer2 module Register | | | | | | | | 193* |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 115 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 123 |

Legend: — = Unimplemented locations, read as '0', u = unchanged, x = unknown. Shaded cells are not used by the PWM.

* Page provides register information.

RRF Rotate Right f through Carry

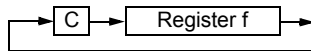
Syntax: [*label*] RRF f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: See description below

Status Affected: C

Description: The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.



SUBLW Subtract W from literal

Syntax: [*label*] SUBLW k

Operands: $0 \leq k \leq 255$

Operation: $k - (W) \rightarrow (W)$

Status Affected: C, DC, Z

Description: The W register is subtracted (2's complement method) from the 8-bit literal 'k'. The result is placed in the W register.

| | |
|--------|----------------------|
| C = 0 | $W > k$ |
| C = 1 | $W \leq k$ |
| DC = 0 | $W<3:0> > k<3:0>$ |
| DC = 1 | $W<3:0> \leq k<3:0>$ |

SLEEP Enter Sleep mode

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h \rightarrow WDT,
0 \rightarrow WDT prescaler,
1 $\rightarrow \overline{TO}$,
0 $\rightarrow \overline{PD}$

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

Description: The power-down Status bit, \overline{PD} is cleared. Time-out Status bit, \overline{TO} is set. Watchdog Timer and its prescaler are cleared. The processor is put into Sleep mode with the oscillator stopped.

SUBWF Subtract W from f

Syntax: [*label*] SUBWF f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: $(f) - (W) \rightarrow (\text{destination})$

Status Affected: C, DC, Z

Description: Subtract (2's complement method) W register from 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'.

| | |
|--------|----------------------|
| C = 0 | $W > f$ |
| C = 1 | $W \leq f$ |
| DC = 0 | $W<3:0> > f<3:0>$ |
| DC = 1 | $W<3:0> \leq f<3:0>$ |

SUBWFB Subtract W from f with Borrow

Syntax: SUBWFB f {,d}

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: $(f) - (W) - (\overline{B}) \rightarrow \text{dest}$

Status Affected: C, DC, Z

Description: Subtract W and the BORROW flag (CARRY) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.

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>