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	10MHz
Connectivity	I²C, SPI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	1.75KB (1K x 14)
Program Memory Type	FLASH
EEPROM Size	128 x 8
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 5x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SSOP (0.209", 5.30mm Width)
Supplier Device Package	20-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lf818t-i-ss

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 2-1.

The Special Function Registers can be classified into two sets: core (CPU) and peripheral. Those registers associated with the core functions are described in detail in this section. Those related to the operation of the peripheral features are described in detail in the peripheral feature section.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
Bank 0											
00h(1)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	23
01h	TMR0	Timer0 Module Register								xxxx xxxx	53, 17
02h(1)	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	23
03h(1)	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxx	16
04h(1)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	23
05h	PORTA	PORTA Data Latch when written; PORTA pins when read								xxx0 0000	39
06h	PORTB	PORTB Data Latch when written; PORTB pins when read								xxxx xxxx	43
07h	—	Unimplemented								—	—
08h	—	Unimplemented								—	—
09h	—	Unimplemented								—	—
0Ah(1,2)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	23
0Bh(1)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	18
0Ch	PIR1	—	ADIF	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF	-0-- 0000	20
0Dh	PIR2	—	—	—	EEIF	—	—	—	—	---0 ----	21
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	57
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	57
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	57
11h	TMR2	Timer2 Module Register								0000 0000	63
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	64
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	71, 76
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	73
15h	CCPR1L	Capture/Compare/PWM Register (LSB)								xxxx xxxx	66, 67, 68
16h	CCPR1H	Capture/Compare/PWM Register (MSB)								xxxx xxxx	66, 67, 68
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	65
18h	—	Unimplemented								—	—
19h	—	Unimplemented								—	—
1Ah	—	Unimplemented								—	—
1Bh	—	Unimplemented								—	—
1Ch	—	Unimplemented								—	—
1Dh	—	Unimplemented								—	—
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	81
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON	0000 00-0	81

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

Note 1: These registers can be addressed from any bank.

2: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

3: Pin 5 is an input only; the state of the TRISA5 bit has no effect and will always read '1'.

2.2.2.2 OPTION_REG Register

The OPTION_REG register is a readable and writable register that contains various control bits to configure the TMR0 prescaler/WDT postscaler (single assignable register known also as the prescaler), the external INT interrupt, TMR0 and the weak pull-ups on PORTB.

Note: To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

REGISTER 2-2: OPTION_REG: OPTION REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG:** Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS:** TMR0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE:** TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0:** Prescaler Rate Select bits

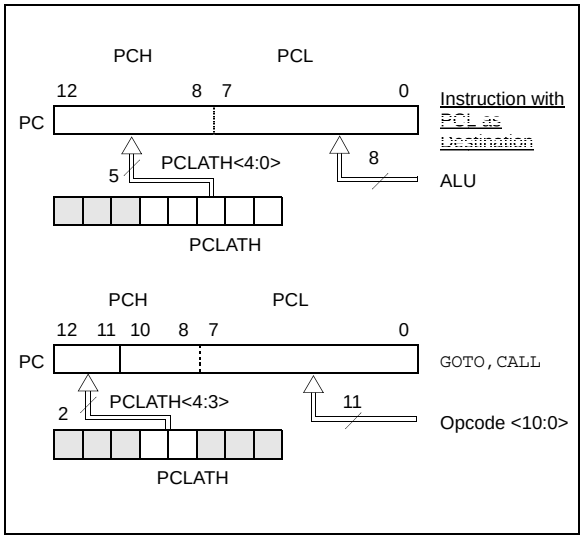
Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

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

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 upper bits (PC<12:8>) are not readable but are indirectly writable through the PCLATH register. On any Reset, the upper bits of the PC will be cleared. Figure 2-5 shows the two situations for the loading of the PC. The upper example in the figure shows how the PC is loaded on a write to PCL (PCLATH<4:0> → PCH). The lower example in the figure 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 doing 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 the application note *AN556, "Implementing a Table Read"* (DS00556).

2.3.2 STACK

The PIC16F818/819 family has an 8-level deep x 13-bit wide hardware stack. 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 Indirect Addressing: INDF and FSR Registers

The INDF register is not a physical register. Addressing INDF actually addresses the register whose address is contained in the FSR register (FSR is a *pointer*). This is indirect addressing.

EXAMPLE 2-1: INDIRECT ADDRESSING

- Register file 05 contains the value 10h
- Register file 06 contains the value 0Ah
- Load the value 05 into the FSR register
- A read of the INDF register will return the value of 10h
- Increment the value of the FSR register by one (FSR = 06)
- A read of the INDF register now will return the value of 0Ah

Reading INDF itself indirectly (FSR = 0) will produce 00h. Writing to the INDF register indirectly results in a no operation (although status bits may be affected).

A simple program to clear RAM locations, 20h-2Fh, using indirect addressing is shown in Example 2-2.

EXAMPLE 2-2: HOW TO CLEAR RAM USING INDIRECT ADDRESSING

```
        MOVLW  0x20    ;initialize pointer
        MOVWF  FSR     ;to RAM
NEXT    CLRF   INDF    ;clear INDF register
        INCF   FSR     ;inc pointer
        BTFSS  FSR, 4  ;all done?
        GOTO   NEXT    ;NO, clear next
CONTINUE
       :               ;YES, continue
```

An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (Status<7>) as shown in Figure 2-6.

3.7 Writing to Flash Program Memory

Flash program memory may only be written to if the destination address is in a segment of memory that is not write-protected, as defined in bits WRT1:WRT0 of the device Configuration Word (Register 12-1). Flash program memory must be written in four-word blocks. A block consists of four words with sequential addresses, with a lower boundary defined by an address, where EEADR<1:0> = 00. At the same time, all block writes to program memory are done as write-only operations. The program memory must first be erased. The write operation is edge-aligned and cannot occur across boundaries.

To write to the program memory, the data must first be loaded into the buffer registers. There are four 14-bit buffer registers and they are addressed by the low 2 bits of EEADR.

The following sequence of events illustrate how to perform a write to program memory:

- Set the EEPGD and WREN bits in the EECON1 register
- Clear the FREE bit in EECON1
- Write address to EEADRH:EEADR
- Write data to EEDATH:EEDATA
- Write 55 to EECON2
- Write AA to EECON2
- Set WR bit in EECON 1

The user must follow the same specific sequence to initiate the write for each word in the program block by writing each program word in sequence (00, 01, 10, 11).

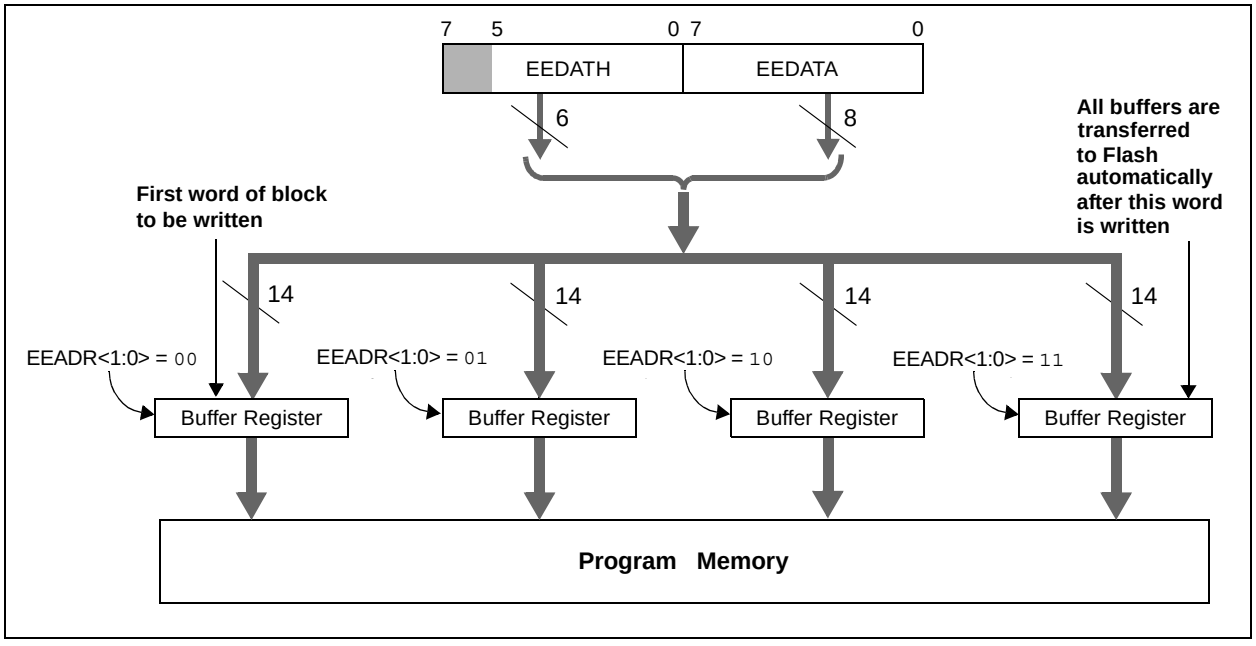
There are 4 buffer register words and all four locations **MUST** be written to with correct data.

After the "BSF EECON1, WR" instruction, if EEADR ≠ xxxxxx11, then a short write will occur. This short write-only transfers the data to the buffer register. The WR bit will be cleared in hardware after one cycle.

After the "BSF EECON1, WR" instruction, if EEADR = xxxxxx11, then a long write will occur. This will simultaneously transfer the data from EEDATH:EEDATA to the buffer registers and begin the write of all four words. The processor will execute the next instruction and then ignore the subsequent instruction. The user should place NOP instructions into the second words. The processor will then halt internal operations for typically 2 msec in which the write takes place. This is not a Sleep mode, as the clocks and peripherals will continue to run. After the write cycle, the processor will resume operation with the 3rd instruction after the EECON1 write instruction.

After each long write, the 4 buffer registers will be reset to 3FFF.

FIGURE 3-1: BLOCK WRITES TO FLASH PROGRAM MEMORY



An example of the complete four-word write sequence is shown in Example 3-5. The initial address is loaded into the EEADRH:EEADR register pair; the four words of data are loaded using indirect addressing, assuming that a row erase sequence has already been performed.

EXAMPLE 3-5: WRITING TO FLASH PROGRAM MEMORY

```
; This write routine assumes the following:

; 1. The 32 words in the erase block have already been erased.
; 2. A valid starting address (the least significant bits = '00') is loaded into EEADRH:EEADR
; 3. This example is starting at 0x100, this is an application dependent setting.
; 4. The 8 bytes (4 words) of data are loaded, starting at an address in RAM called ARRAY.
; 5. This is an example only, location of data to program is application dependent.
; 6. word_block is located in data memory.

        BANKSEL  EECON1            ;prepare for WRITE procedure
        BSF      EECON1, EEPGD     ;point to program memory
        BSF      EECON1, WREN      ;allow write cycles
        BCF      EECON1, FREE      ;perform write only

        BANKSEL  word_block
        MOVLW    .4
        MOVWF    word_block        ;prepare for 4 words to be written

        BANKSEL  EEADRH            ;Start writing at 0x100
        MOVLW    0x01
        MOVWF    EEADRH            ;load HIGH address
        MOVLW    0x00
        MOVWF    EEADR             ;load LOW address
        BANKSEL  ARRAY
        MOVLW    ARRAY             ;initialize FSR to start of data
        MOVWF    FSR
LOOP
        BANKSEL  EEDATA
        MOVF     INDF, W           ;indirectly load EEDATA
        MOVWF    EEDATA
        INCF     FSR, F            ;increment data pointer
        MOVF     INDF, W           ;indirectly load EEDATH
        MOVWF    EEDATH
        INCF     FSR, F            ;increment data pointer

        BANKSEL  EECON1
        MOVLW    0x55              ;required sequence
        MOVWF    EECON2
        MOVLW    0xAA
        MOVWF    EECON2
        BSF      EECON1, WR        ;set WR bit to begin write
        NOP                        ;instructions here are ignored as processor
        NOP

        BANKSEL  EEADR
        INCF     EEADR, f          ;load next word address
        BANKSEL  word_block
        DECFSZ   word_block, f     ;have 4 words been written?
        GOTO     loop              ;NO, continue with writing

        BANKSEL  EECON1
        BCF      EECON1, WREN      ;YES, 4 words complete, disable writes
        BSF      INTCON, GIE       ;enable interrupts
```

Required Sequence

3.8 Protection Against Spurious Write

There are conditions when the device should not write to the data EEPROM memory. To protect against spurious EEPROM writes, various mechanisms have been built-in. On power-up, WREN is cleared. Also, the Power-up Timer (72 ms duration) prevents an EEPROM write.

The write initiate sequence and the WREN bit together help prevent an accidental write during brown-out, power glitch or software malfunction.

3.9 Operation During Code-Protect

When the data EEPROM is code-protected, the microcontroller can read and write to the EEPROM normally. However, all external access to the EEPROM is disabled. External write access to the program memory is also disabled.

When program memory is code-protected, the microcontroller can read and write to program memory normally as well as execute instructions. Writes by the device may be selectively inhibited to regions of the memory depending on the setting of bits, WRT1:WRT0, of the Configuration Word (see **Section 12.1 "Configuration Bits"** for additional information). External access to the memory is also disabled.

TABLE 3-1: REGISTERS/BITS ASSOCIATED WITH DATA EEPROM AND FLASH PROGRAM MEMORIES

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other Resets
10Ch	EEDATA	EEPROM/Flash Data Register Low Byte								xxxx xxxx	uuuu uuuu
10Dh	EEADR	EEPROM/Flash Address Register Low Byte								xxxx xxxx	uuuu uuuu
10Eh	EEDATH	—	—	EEPROM/Flash Data Register High Byte						--xx xxxx	--uu uuuu
10Fh	EEADRH	—	—	—	—	—	EEPROM/Flash Address Register High Byte			---- -xxx	---- -uuu
18Ch	EECON1	EEPGD	—	—	FREE	WRERR	WREN	WR	RD	x--x x000	x--x q000
18Dh	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	---- ----
0Dh	PIR2	—	—	—	EEIF	—	—	—	—	---0 ----	---0 ----
8Dh	PIE2	—	—	—	EEIE	—	—	—	—	---0 ----	---0 ----

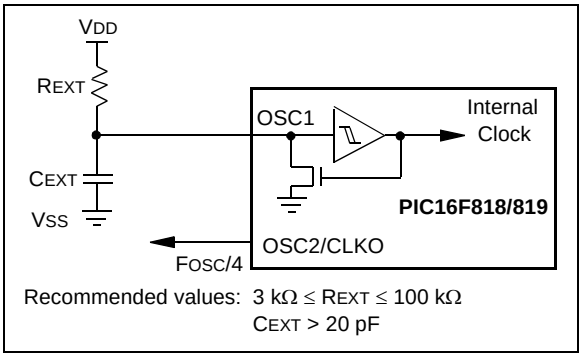
Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends upon condition.
Shaded cells are not used by data EEPROM or Flash program memory.

4.4 RC Oscillator

For timing insensitive applications, the “RC” and “RCIO” device options offer additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (REXT) and capacitor (CEXT) values and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit due to normal manufacturing variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low CEXT values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 4-4 shows how the R/C combination is connected.

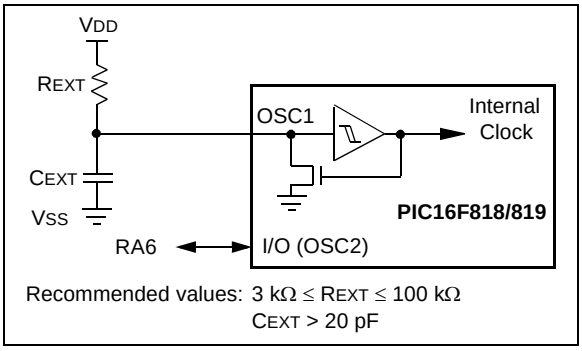
In the RC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic.

FIGURE 4-4: RC OSCILLATOR MODE



The RCIO Oscillator mode (Figure 4-5) functions like the RC mode except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6).

FIGURE 4-5: RCIO OSCILLATOR MODE



4.5 Internal Oscillator Block

The PIC16F818/819 devices include an internal oscillator block which generates two different clock signals; either can be used as the system’s clock source. This can eliminate the need for external oscillator circuits on the OSC1 and/or OSC2 pins.

The main output (INTOSC) is an 8 MHz clock source which can be used to directly drive the system clock. It also drives the INTOSC postscaler which can provide a range of clock frequencies from 125 kHz to 4 MHz.

The other clock source is the internal RC oscillator (INTRC) which provides a 31.25 kHz (32 μs nominal period) output. The INTRC oscillator is enabled by selecting the INTRC as the system clock source or when any of the following are enabled:

- Power-up Timer
- Watchdog Timer

These features are discussed in greater detail in **Section 12.0 “Special Features of the CPU”**.

The clock source frequency (INTOSC direct, INTRC direct or INTOSC postscaler) is selected by configuring the IRCF bits of the OSCCON register (Register 4-2).

Note:	Throughout this data sheet, when referring *specifically* to a generic clock source, the term “INTRC” may also be used to refer to the clock modes using the internal oscillator block. This is regardless of whether the actual frequency used is INTOSC (8 MHz), the INTOSC postscaler or INTRC (31.25 kHz).

4.5.1 INTRC MODES

Using the internal oscillator as the clock source can eliminate the need for up to two external oscillator pins, which can then be used for digital I/O. Two distinct configurations are available:

- In INTIO1 mode, the OSC2 pin outputs FOSC/4 while OSC1 functions as RA7 for digital input and output.
- In INTIO2 mode, OSC1 functions as RA7 and OSC2 functions as RA6, both for digital input and output.

5.0 I/O PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Additional information on I/O ports may be found in the *"PIC® Mid-Range MCU Family Reference Manual"* (DS33023).

5.1 PORTA and the TRISA Register

PORTA is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Note:	On a Power-on Reset, the pins PORTA<4:0> are configured as analog inputs and read as '0'.

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the port data latch.

Pin RA4 is multiplexed with the Timer0 module clock input and with an analog input to become the RA4/AN4/T0CKI pin. The RA4/AN4/T0CKI pin is a Schmitt Trigger input and full CMOS output driver.

Pin RA5 is multiplexed with the Master Clear module input. The RA5/MCLR/VPP pin is a Schmitt Trigger input.

Pin RA6 is multiplexed with the oscillator module input and external oscillator output. Pin RA7 is multiplexed with the oscillator module input and external oscillator input. Pin RA6/OSC2/CLKO and pin RA7/OSC1/CLKI are Schmitt Trigger inputs and full CMOS output drivers.

Pins RA<1:0> are multiplexed with analog inputs. Pins RA<3:2> are multiplexed with analog inputs and VREF inputs. Pins RA<3:0> have TTL inputs and full CMOS output drivers.

EXAMPLE 5-1: INITIALIZING PORTA

```
BANKSEL  PORTA     ; select bank of PORTA
CLRF     PORTA     ; Initialize PORTA by
                   ; clearing output
                   ; data latches
BANKSEL  ADCON1    ; Select Bank of ADCON1
MOVLW    0x06      ; Configure all pins
MOVWF    ADCON1    ; as digital inputs
MOVLW    0xFF      ; Value used to
                   ; initialize data
                   ; direction
MOVWF    TRISA     ; Set RA<7:0> as inputs
```

TABLE 5-1: PORTA FUNCTIONS

Name	Bit#	Buffer	Function
RA0/AN0	bit 0	TTL	Input/output or analog input.
RA1/AN1	bit 1	TTL	Input/output or analog input.
RA2/AN2/VREF-	bit 2	TTL	Input/output, analog input or VREF-.
RA3/AN3/VREF+	bit 3	TTL	Input/output, analog input or VREF+.
RA4/AN4/T0CKI	bit 4	ST	Input/output, analog input or external clock input for Timer0.
RA5/MCLR/VPP	bit 5	ST	Input, Master Clear (Reset) or programming voltage input.
RA6/OSC2/CLKO	bit 6	ST	Input/output, connects to crystal or resonator, oscillator output or 1/4 the frequency of OSC1 and denotes the instruction cycle in RC mode.
RA7/OSC1/CLKI	bit 7	ST/CMOS(1)	Input/output, connects to crystal or resonator or oscillator input.

Legend: TTL = TTL input, ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

TABLE 5-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Address	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
05h	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxx0 0000	uuu0 0000
85h	TRISA	TRISA7	TRISA6	TRISA5(1)	PORTA Data Direction Register					1111 1111	1111 1111
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by PORTA.

Note 1: Pin 5 is an input only; the state of the TRISA5 bit has no effect and will always read '1'.

5.2 PORTB and the TRISB Register

PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit RBPU (OPTION_REG<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

Four of PORTB's pins, RB7:RB4, have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are ORed together to generate the RB Port Change Interrupt with Flag bit, RBIF (INTCON<0>).

This interrupt can wake the device from Sleep. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB. This will end the mismatch condition.
b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

RB0/INT is an external interrupt input pin and is configured using the INTEDG bit (OPTION_REG<6>).

PORTB is multiplexed with several peripheral functions (see Table 5-3). PORTB pins have Schmitt Trigger input buffers.

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTB pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. Since the TRIS bit override is in effect while the peripheral is enabled, read-modify-write instructions (BSF, BCF, XORWF) with TRISB as the destination should be avoided. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

8.0 TIMER2 MODULE

Timer2 is an 8-bit timer with a prescaler and a postscaler. It can be used as the PWM time base for the PWM mode of the CCP1 module. The TMR2 register is readable and writable and is cleared on any device Reset.

The input clock (FOSC/4) has a prescale option of 1:1, 1:4 or 1:16, selected by control bits, T2CKPS1:T2CKPS0 (T2CON<1:0>).

The Timer2 module has an 8-bit period register, PR2. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register. The PR2 register is initialized to FFh upon Reset.

The match output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt (latched in flag bit, TMR2IF (PIR1<1>)).

Timer2 can be shut-off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption.

Register 8-1 shows the Timer2 Control register.

Additional information on timer modules is available in the *"PIC® Mid-Range MCU Family Reference Manual"* (DS33023).

8.1 Timer2 Prescaler and Postscaler

The prescaler and postscaler counters are cleared when any of the following occurs:

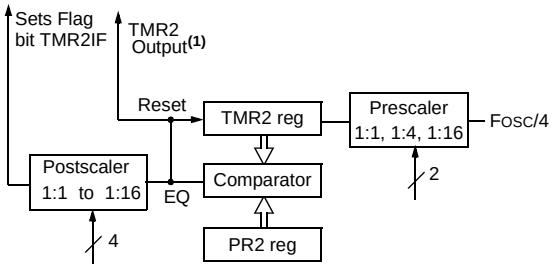
- A write to the TMR2 register
- A write to the T2CON register
- Any device Reset (Power-on Reset, $\overline{\text{MCLR}}$, WDT Reset or Brown-out Reset)

TMR2 is not cleared when T2CON is written.

8.2 Output of TMR2

The output of TMR2 (before the postscaler) is fed to the Synchronous Serial Port module which optionally uses it to generate a shift clock.

FIGURE 8-1: TIMER2 BLOCK DIAGRAM



Note 1: TMR2 register output can be software selected by the SSP module as a baud clock.

The maximum PWM resolution (bits) for a given PWM frequency is given by the following formula.

EQUATION 9-3:

$$\text{Resolution} = \frac{\log\left(\frac{FOSC}{FPWM}\right)}{\log(2)} \text{ bits}$$

Note:	If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

9.3.3 SETUP FOR PWM OPERATION

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

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCPR1L register and CCP1CON<5:4> bits.
3. Make the CCP1 pin an output by clearing the TRISB<x> bit.
4. Set the TMR2 prescale value and enable Timer2 by writing to T2CON.
5. Configure the CCP1 module for PWM operation.

Note:	The TRISB bit (2 or 3) is dependant upon the setting of configuration bit 12 (CCPMX).

TABLE 9-3: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 20 MHz

PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.12 kHz	156.3 kHz	208.3 kHz
Timer Prescaler (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	5.5

TABLE 9-4: REGISTERS ASSOCIATED WITH PWM AND TIMER2

Address	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
0Bh,8Bh 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	ADIF	—	—	SSPIF	CCP1IF	TMR2IF	TMR1IF	-0-- 0000	-0-- 0000
8Ch	PIE1	—	ADIE	—	—	SSPIE	CCP1IE	TMR2IE	TMR1IE	-0-- 0000	-0-- 0000
86h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
11h	TMR2	Timer2 Module Register								0000 0000	0000 0000
92h	PR2	Timer2 Module Period Register								1111 1111	1111 1111
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PWM and Timer2.

The ADRESH:ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the A/D Result register pair, the GO/DONE bit (ADCON0<2>) is cleared and A/D Interrupt Flag bit, ADIF, is set. The block diagram of the A/D module is shown in Figure 11-1.

After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as inputs.

To determine sample time, see **Section 11.1 "A/D Acquisition Requirements"**. After this sample time has elapsed, the A/D conversion can be started.

These steps should be followed for doing an A/D conversion:

1. Configure the A/D module:
   - Configure analog pins/voltage reference and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D conversion clock (ADCON0)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. Wait the required acquisition time.
4. Start conversion:
   - Set GO/DONE bit (ADCON0)
5. Wait for A/D conversion to complete by either:
   - Polling for the GO/DONE bit to be cleared (with interrupts disabled); OR
   - Waiting for the A/D interrupt
6. Read A/D Result register pair (ADRESH:ADRESL), clear bit ADIF if required.
7. For next conversion, go to step 1 or step 2 as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2 TAD is required before the next acquisition starts.

FIGURE 11-1: A/D BLOCK DIAGRAM

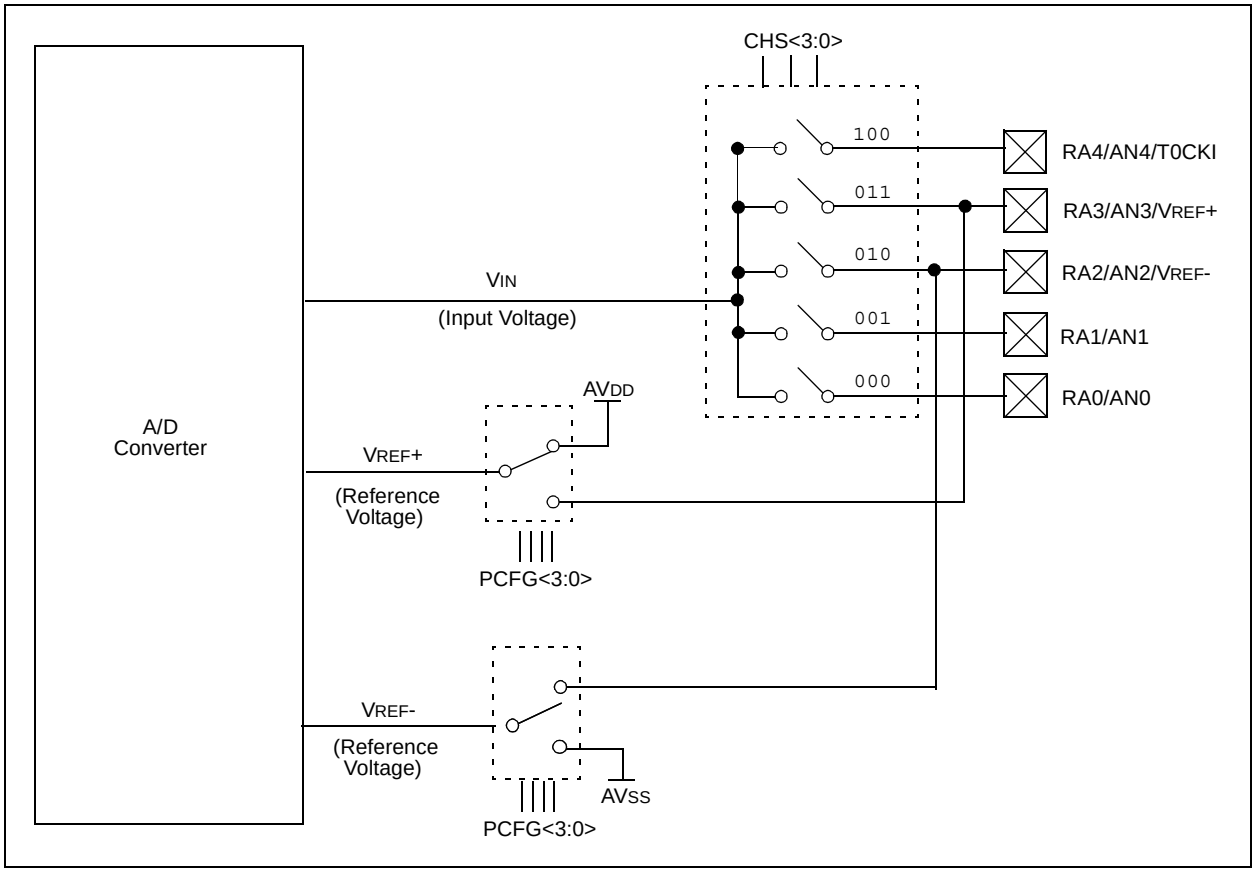


TABLE 13-2: PIC16F818/819 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	**f, d**	Add W and f	1	00	0111	dfff	ffff	C, DC, Z	1, 2
ANDWF	**f, d**	AND W with f	1	00	0101	dfff	ffff	Z	1, 2
CLRF	**f**	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	**-**	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	**f, d**	Complement f	1	00	1001	dfff	ffff	Z	1, 2
DECF	**f, d**	Decrement f	1	00	0011	dfff	ffff	Z	1, 2
DECFSZ	**f, d**	Decrement f, Skip if 0	1 (2)	00	1011	dfff	ffff		1, 2, 3
INCF	**f, d**	Increment f	1	00	1010	dfff	ffff	Z	1, 2
INCFSZ	**f, d**	Increment f, Skip if 0	1 (2)	00	1111	dfff	ffff		1, 2, 3
IORWF	**f, d**	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1, 2
MOVF	**f, d**	Move f	1	00	1000	dfff	ffff	Z	1, 2
MOVWF	**f**	Move W to f	1	00	0000	lfff	ffff		
NOP	**-**	No Operation	1	00	0000	0xx0	0000		
RLF	**f, d**	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1, 2
RRF	**f, d**	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1, 2
SUBWF	**f, d**	Subtract W from f	1	00	0010	dfff	ffff	C, DC, Z	1, 2
SWAPF	**f, d**	Swap nibbles in f	1	00	1110	dfff	ffff		1, 2
XORWF	**f, d**	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1, 2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	**f, b**	Bit Clear f	1	01	00bb	bfff	ffff		1, 2
BSF	**f, b**	Bit Set f	1	01	01bb	bfff	ffff		1, 2
BTFSC	**f, b**	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	**f, b**	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	**k**	Add literal and W	1	11	111x	kkkk	kkkk	C, DC, Z	
ANDLW	**k**	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	**k**	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	**-**	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$, $\overline{PD}$	
GOTO	**k**	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	**k**	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	**k**	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	**-**	Return from interrupt	2	00	0000	0000	1001		
RETLW	**k**	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	**-**	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	**-**	Go into Standby mode	1	00	0000	0110	0011	$\overline{TO}$, $\overline{PD}$	
SUBLW	**k**	Subtract W from literal	1	11	110x	kkkk	kkkk	C, DC, Z	
XORLW	**k**	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 module.

3: If the Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

Note: Additional information on the mid-range instruction set is available in the *"PIC® Mid-Range MCU Family Reference Manual"* (DS33023).

15.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

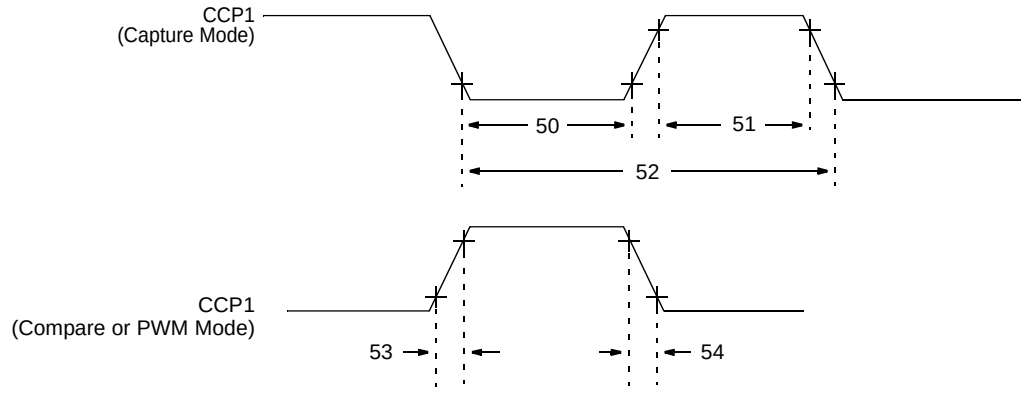
Ambient temperature under bias ..... -40°C to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on any pin with respect to V_{SS} (except V_{DD} and $\overline{MCLR}$) ..... -0.3V to (V_{DD} + 0.3V)
Voltage on V_{DD} with respect to V_{SS} ..... -0.3 to +7.5V
Voltage on $\overline{MCLR}$ with respect to V_{SS} **(Note 2)** ..... -0.3 to +14V
Total power dissipation **(Note 1)** 1W
Maximum current out of V_{SS} pin ..... 200 mA
Maximum current into V_{DD} pin ..... 200 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD}) ..... ±20 mA
Output clamp current, I_{OK} (V_O < 0 or V_O > V_{DD}) ..... ±20 mA
Maximum output current sunk by any I/O pin ..... 25 mA
Maximum output current sourced by any I/O pin ..... 25 mA
Maximum current sunk by PORTA ..... 100 mA
Maximum current sourced by PORTA ..... 100 mA
Maximum current sunk by PORTB ..... 100 mA
Maximum current sourced by PORTB ..... 100 mA

Note 1: Power dissipation is calculated as follows: $P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$

2: Voltage spikes at the $\overline{MCLR}$ pin may cause latch-up. A series resistor of greater than 1 kΩ should be used to pull $\overline{MCLR}$ to V_{DD}, rather than tying the pin directly to V_{DD}.

† NOTICE: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

FIGURE 15-9: CAPTURE/COMPARE/PWM TIMINGS (CCP1)



Note: Refer to Figure 15-3 for load conditions.

TABLE 15-5: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP1)

Param No.	Symbol	Characteristic			Min	Typ†	Max	Units	Conditions
50*	TccL	CCP1 Input Low Time	No Prescaler		0.5 TCY + 20	—	—	ns	
			With Prescaler	PIC16**F**818/819	10	—	—	ns	
				PIC16**LF**818/819	20	—	—	ns	
51*	TccH	CCP1 Input High Time	No Prescaler		0.5 TCY + 20	—	—	ns	
			With Prescaler	PIC16**F**818/819	10	—	—	ns	
				PIC16**LF**818/819	20	—	—	ns	
52*	TccP	CCP1 Input Period			$\frac{3\ T_{CY} + 40}{N}$	—	—	ns	N = prescale value (1,4 or 16)
53*	TccR	CCP1 Output Rise Time		PIC16**F**818/819	—	10	25	ns	
				PIC16**LF**818/819	—	25	50	ns	
54*	TccF	CCP1 Output Fall Time		PIC16**F**818/819	—	10	25	ns	
				PIC16**LF**818/819	—	25	45	ns	

* These parameters are characterized but not tested.

† Data in “Typ” column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 15-12: SPI SLAVE MODE TIMING (CKE = 0)

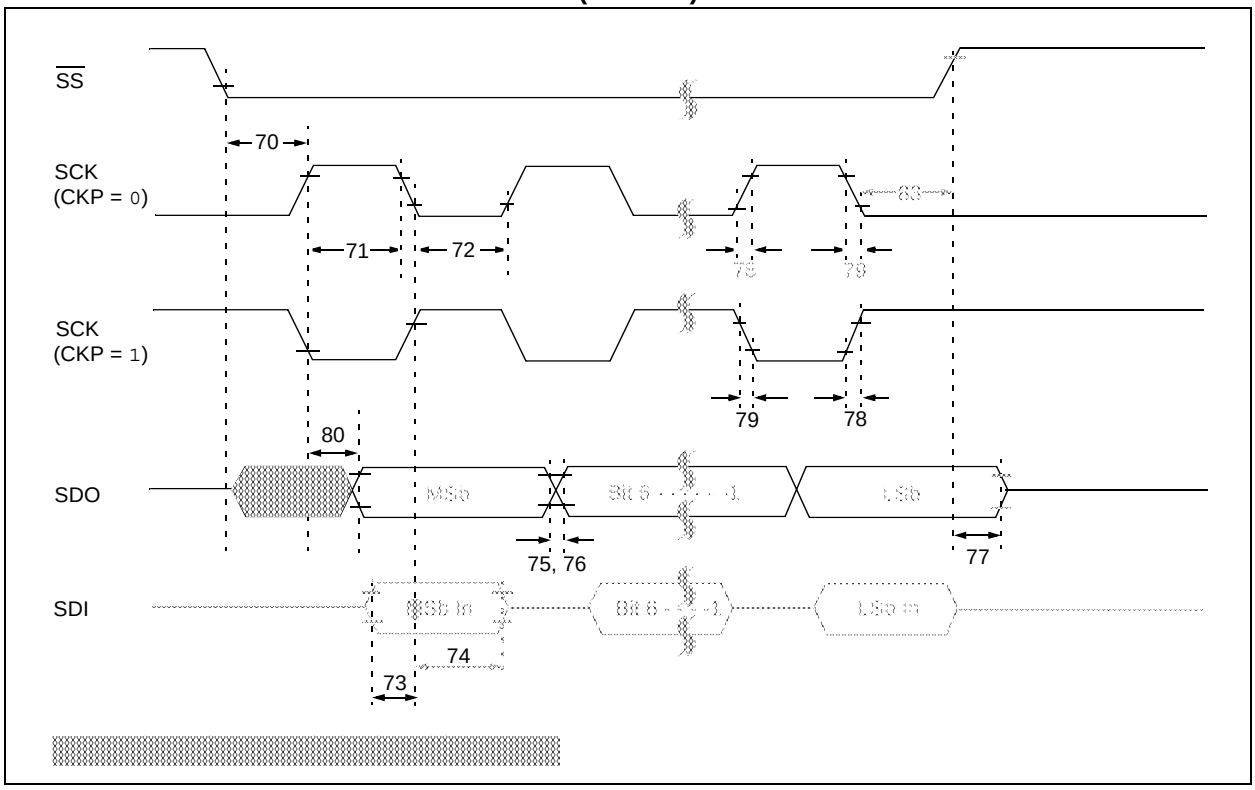
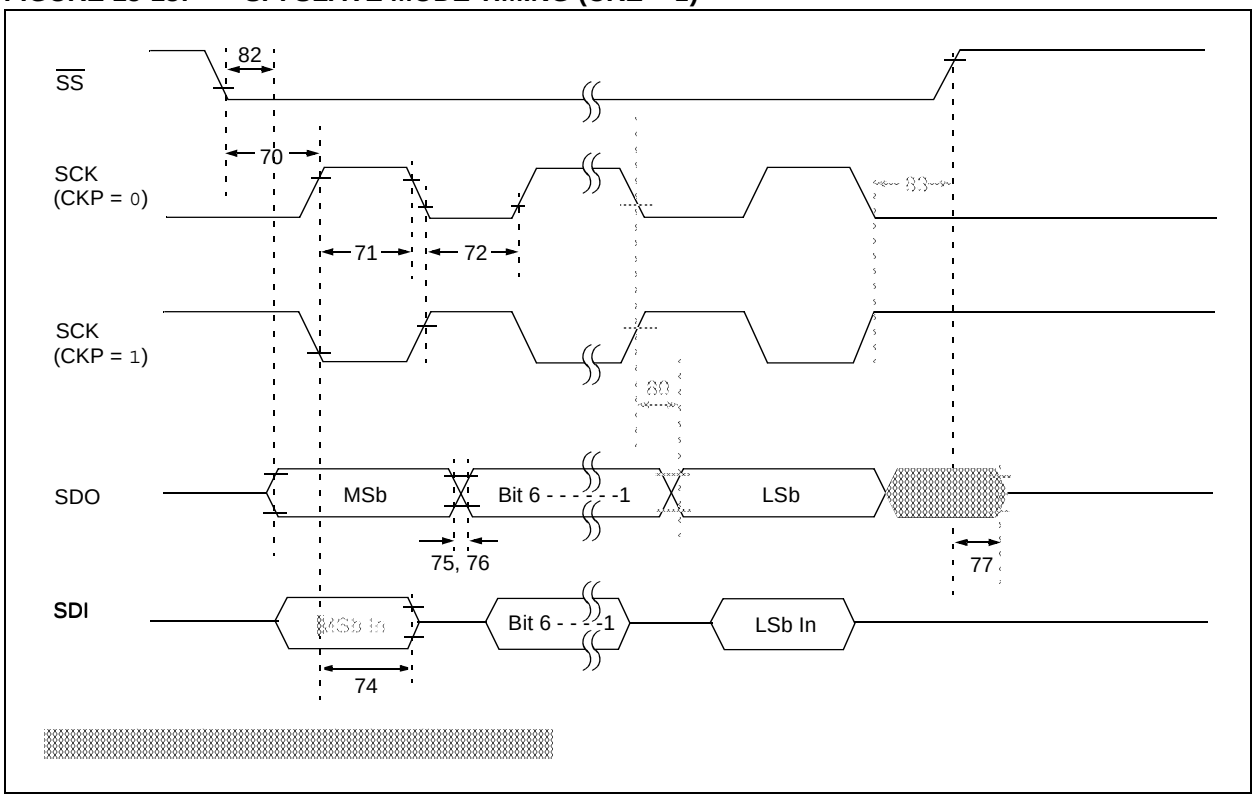


FIGURE 15-13: SPI SLAVE MODE TIMING (CKE = 1)



16.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

Note:	The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range.

"Typical" represents the mean of the distribution at 25°C. "Maximum" or "minimum" represents (mean + 3σ) or (mean – 3σ) respectively, where σ is a standard deviation, over the whole temperature range.

FIGURE 16-1: TYPICAL IDD vs. FOSC OVER VDD (HS MODE)

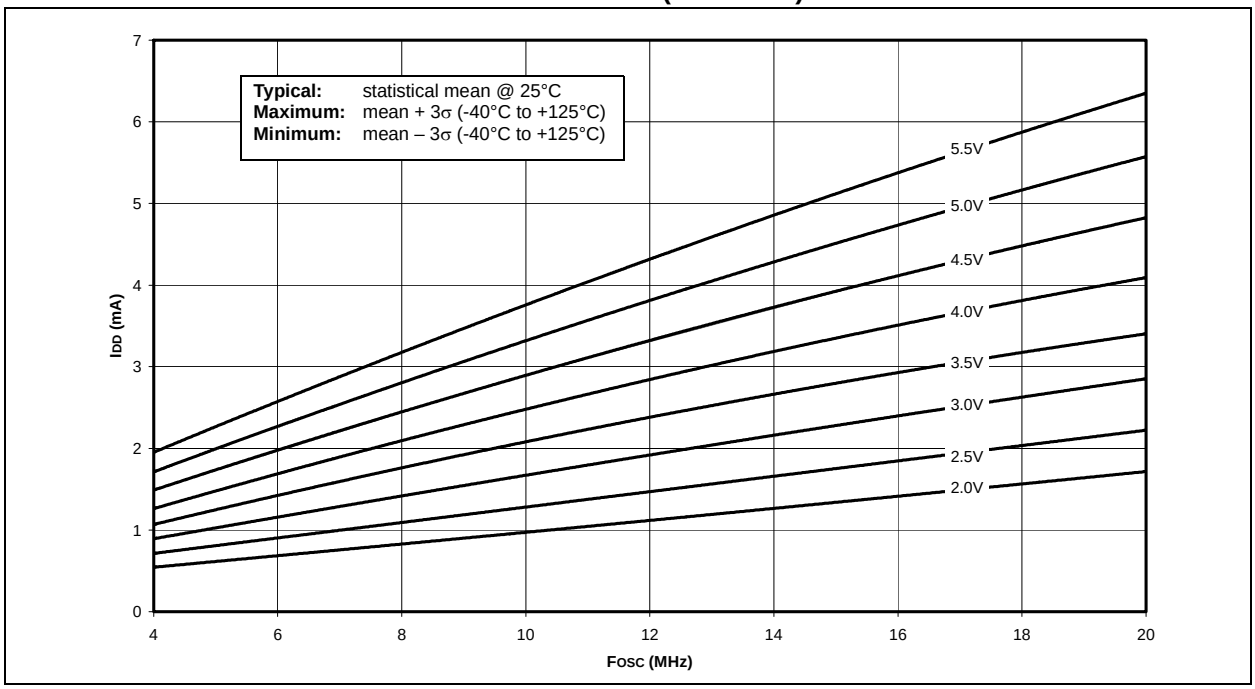


FIGURE 16-2: MAXIMUM IDD vs. FOSC OVER VDD (HS MODE)

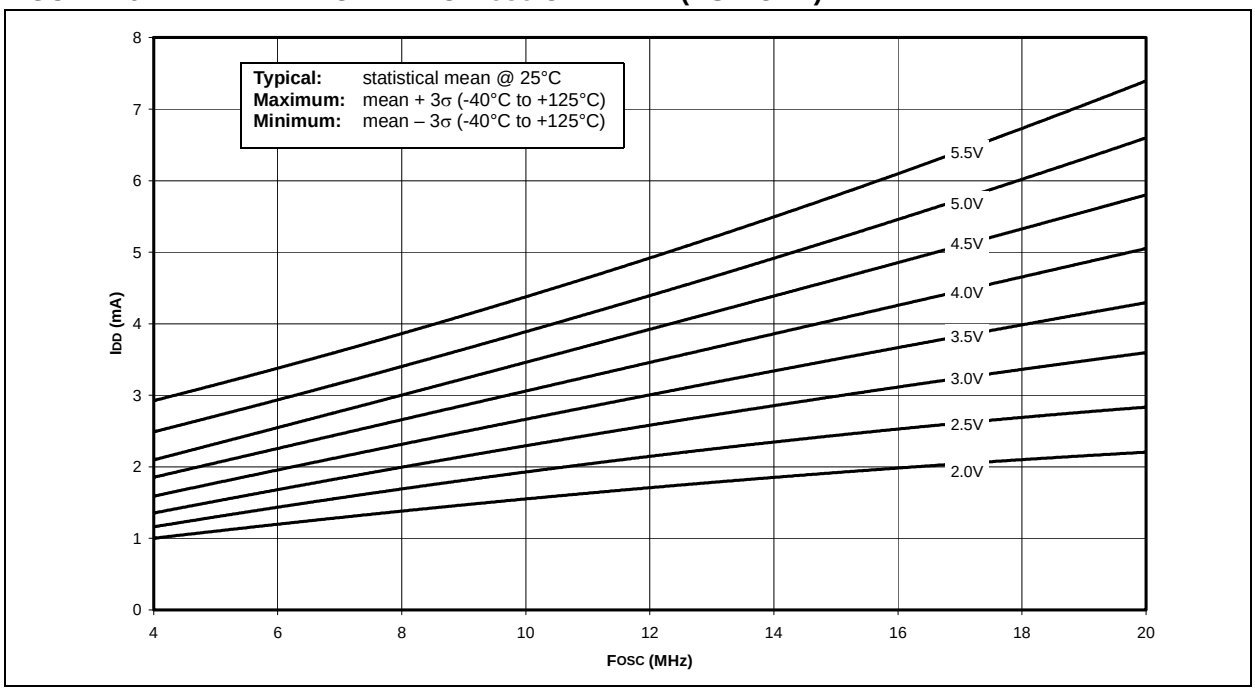


FIGURE 16-13: ΔIPD TIMER1 OSCILLATOR, -10°C TO +70°C (SLEEP MODE, TMR1 COUNTER DISABLED)

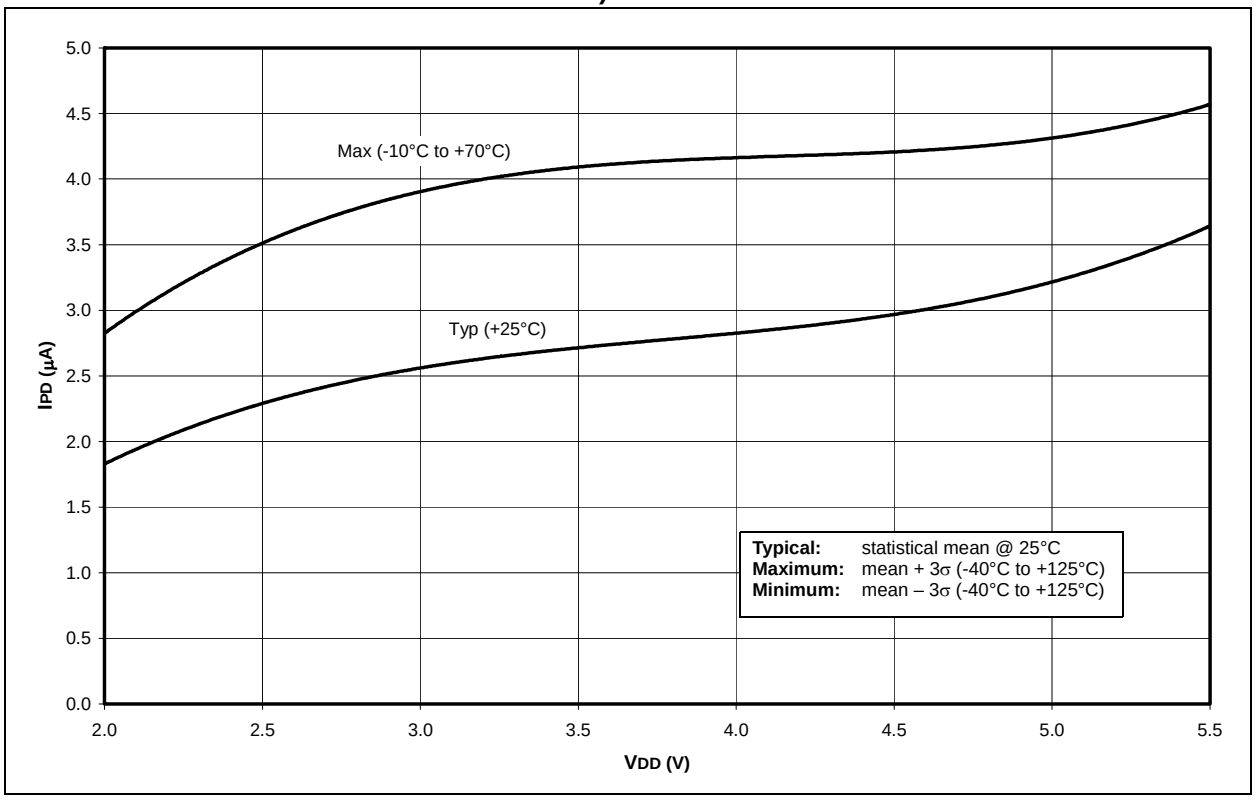
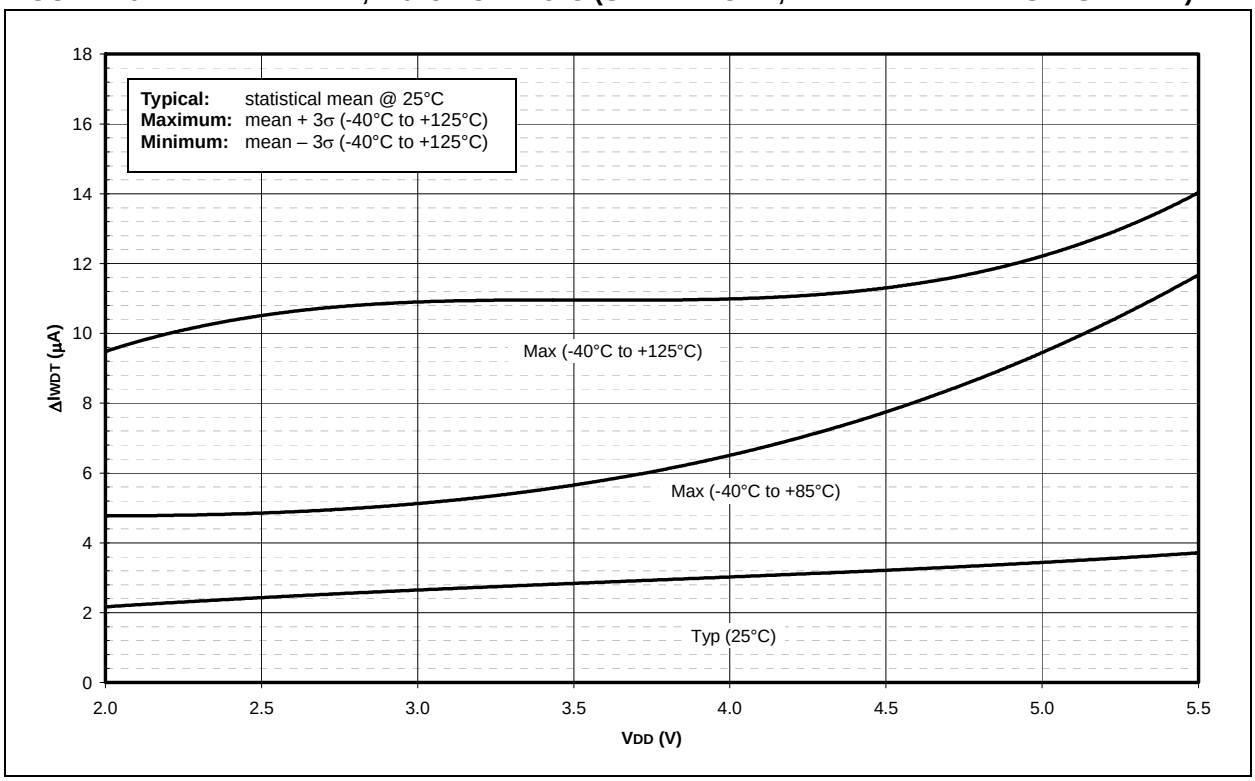


FIGURE 16-14: ΔIPD WDT, -40°C TO +125°C (SLEEP MODE, ALL PERIPHERALS DISABLED)



INDEX

A

A/D
- Acquisition Requirements ... 84
- ADIF Bit ... 83
- Analog-to-Digital Converter ... 81
- Associated Registers ... 87
- Calculating Acquisition Time ... 84
- Configuring Analog Port Pins ... 85
- Configuring the Interrupt ... 83
- Configuring the Module ... 83
- Conversion Clock ... 85
- Conversion Requirements ... 140
- Conversions ... 86
- Converter Characteristics ... 139
- Delays ... 84
- Effects of a Reset ... 87
- GO/DONE Bit ... 83
- Internal Sampling Switch (Rss) Impedance ... 84
- Operation During Sleep ... 87
- Result Registers ... 86
- Source Impedance ... 84
- Time Delays ... 84
- Use of the CCP Trigger ... 87

Absolute Maximum Ratings ... 115
ACK ... 77
ADCON0 Register ... 81
ADCON1 Register ... 81
ADRESH Register ... 13, 81
ADRESH, ADRESL Register Pair ... 83
ADRESL Register ... 14, 81
Application Notes
- AN556 (Implementing a Table Read) ... 23
- AN578 (Use of the SSP Module in the I^2C Multi-Master Environment) ... 71
- AN607 (Power-up Trouble Shooting) ... 92

Assembler
- MPASM Assembler ... 112

B

BF Bit ... 77
Block Diagrams
- A/D ... 83
- Analog Input Model ... 84
- Capture Mode Operation ... 66
- Compare Mode Operation ... 67
- In-Circuit Serial Programming Connections ... 101
- Interrupt Logic ... 96
- On-Chip Reset Circuit ... 91
- PIC16F818/819 ... 6
- PWM ... 68
- RA0/AN0:RA1/AN1 Pins ... 40
- RA2/AN2/Vref- Pin ... 40
- RA3/AN3/Vref+ Pin ... 40
- RA4/AN4/T0CKI Pin ... 40
- RA5/MCLR/Vpp Pin ... 41
- RA6/OSC2/CLKO Pin ... 41
- RA7/OSC1/CLKI Pin ... 42
- RB0 Pin ... 45
- RB1 Pin ... 46
- RB2 Pin ... 47
- RB3 Pin ... 48
- RB4 Pin ... 49
- RB5 Pin ... 50
- RB6 Pin ... 51
- RB7 Pin ... 52
- Recommended MCLR Circuit ... 92
- SSP in I^2C Mode ... 76
- SSP in SPI Mode ... 74
- System Clock ... 38
- Timer0/WDT Prescaler ... 53
- Timer1 ... 58
- Timer2 ... 63
- Watchdog Timer (WDT) ... 98

BOR. *See* Brown-out Reset.
Brown-out Reset (BOR) ... 89, 91, 92, 93, 94

C

C Compilers
- MPLAB C18 ... 112

Capture/Compare/PWM (CCP) ... 65
- Capture Mode ... 66
  - CCP Prescaler ... 66
  - Pin Configuration ... 66
  - Software Interrupt ... 66
  - Timer1 Mode Selection ... 66
- Capture, Compare and Timer1
  - Associated Registers ... 67
- CCP1IF ... 66
- CCPR1 ... 66
- CCPR1H:CCPR1L ... 66
- Compare Mode ... 67
  - Pin Configuration ... 67
  - Software Interrupt Mode ... 67
  - Special Event Trigger ... 67
  - Special Event Trigger Output of CCP1 ... 67
  - Timer1 Mode Selection ... 67
- PWM and Timer2
  - Associated Registers ... 69
- PWM Mode ... 68
  - Duty Cycle ... 68
  - Example Frequencies/Resolutions ... 69
  - Period ... 68
  - Setup for Operation ... 69
- Timer Resources ... 65

CCP1M0 Bit ... 65
CCP1M1 Bit ... 65
CCP1M2 Bit ... 65
CCP1M3 Bit ... 65
CCP1X Bit ... 65
CCP1Y Bit ... 65
CCPR1H Register ... 65
CCPR1L Register ... 65
Code Examples
- Changing Between Capture Prescalers ... 66
- Changing Prescaler Assignment from Timer0 to WDT ... 55
- Changing Prescaler Assignment from WDT to Timer0 ... 55
- Clearing RAM Using Indirect Addressing ... 23
- Erasing a Flash Program Memory Row ... 29
- Implementing a Real-Time Clock Using a Timer1 Interrupt Service ... 62
- Initializing PORTA ... 39
- Reading a 16-Bit Free Running Timer ... 59
- Reading Data EEPROM ... 27
- Reading Flash Program Memory ... 28
- Saving Status and W Registers in RAM ... 97
- Writing a 16-Bit Free Running Timer ... 59
- Writing to Data EEPROM ... 27

NOTES: