Microchip Technology - PIC16F648AT-I/SS Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	20MHz
Connectivity	UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	7KB (4K x 14)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	-
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/pic16f648at-i-ss

NOTES:

TABLE 3-2: PIC16F627A/628A/648A PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
RA0/AN0	RA0	ST	CMOS	Bidirectional I/O port
	AN0	AN	—	Analog comparator input
RA1/AN1	RA1	ST	CMOS	Bidirectional I/O port
	AN1	AN	—	Analog comparator input
RA2/AN2/VREF	RA2	ST	CMOS	Bidirectional I/O port
	AN2	AN	—	Analog comparator input
	VREF	—	AN	VREF output
RA3/AN3/CMP1	RA3	ST	CMOS	Bidirectional I/O port
	AN3	AN	—	Analog comparator input
	CMP1	—	CMOS	Comparator 1 output
RA4/T0CKI/CMP2	RA4	ST	OD	Bidirectional I/O port
	T0CKI	ST	—	Timer0 clock input
	CMP2	—	OD	Comparator 2 output
RA5/$\overline{\text{MCLR}}$/VPP	RA5	ST	—	Input port
	$\overline{\text{MCLR}}$	ST	—	Master clear. When configured as $\overline{\text{MCLR}}$, this pin is an active low Reset to the device. Voltage on $\overline{\text{MCLR}}$/VPP must not exceed VDD during normal device operation.
	VPP	—	—	Programming voltage input
RA6/OSC2/CLKOUT	RA6	ST	CMOS	Bidirectional I/O port
	OSC2	—	XTAL	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
	CLKOUT	—	CMOS	In RC/INTOSC mode, OSC2 pin can output CLKOUT, which has 1/4 the frequency of OSC1.
RA7/OSC1/CLKIN	RA7	ST	CMOS	Bidirectional I/O port
	OSC1	XTAL	—	Oscillator crystal input
	CLKIN	ST	—	External clock source input. RC biasing pin.
RB0/INT	RB0	TTL	CMOS	Bidirectional I/O port. Can be software programmed for internal weak pull-up.
	INT	ST	—	External interrupt
RB1/RX/DT	RB1	TTL	CMOS	Bidirectional I/O port. Can be software programmed for internal weak pull-up.
	RX	ST	—	USART receive pin
	DT	ST	CMOS	Synchronous data I/O
RB2/TX/CK	RB2	TTL	CMOS	Bidirectional I/O port. Can be software programmed for internal weak pull-up.
	TX	—	CMOS	USART transmit pin
	CK	ST	CMOS	Synchronous clock I/O
RB3/CCP1	RB3	TTL	CMOS	Bidirectional I/O port. Can be software programmed for internal weak pull-up.
	CCP1	ST	CMOS	Capture/Compare/PWM I/O

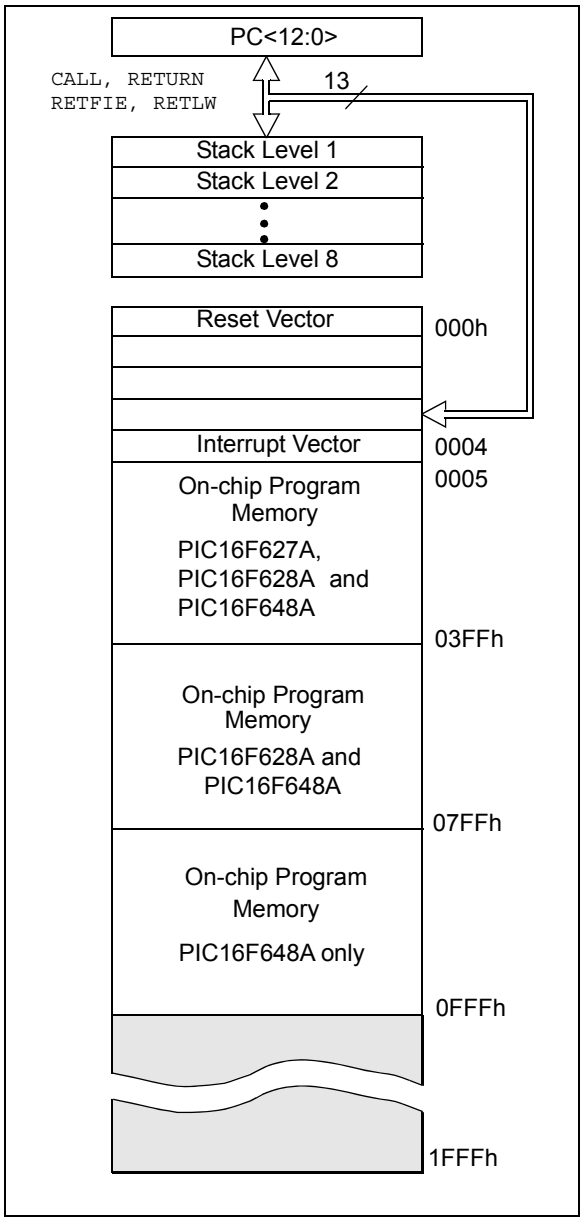
Legend: O = Output, — = Not used, TTL = TTL Input, CMOS = CMOS Output, I = Input, OD = Open Drain Output, P = Power, ST = Schmitt Trigger Input, AN = Analog

4.0 MEMORY ORGANIZATION

4.1 Program Memory Organization

The PIC16F627A/628A/648A has a 13-bit program counter capable of addressing an 8K x 14 program memory space. Only the first 1K x 14 (0000h-03FFh) for the PIC16F627A, 2K x 14 (0000h-07FFh) for the PIC16F628A and 4K x 14 (0000h-0FFFh) for the PIC16F648A are physically implemented. Accessing a location above these boundaries will cause a wrap-around within the first 1K x 14 space (PIC16F627A), 2K x 14 space (PIC16F628A) or 4K x 14 space (PIC16F648A). The Reset vector is at 0000h and the interrupt vector is at 0004h (Figure 4-1).

FIGURE 4-1: PROGRAM MEMORY MAP AND STACK



4.2 Data Memory Organization

The data memory (Figure 4-2 and Figure 4-3) is partitioned into four banks, which contain the General Purpose Registers (GPRs) and the Special Function Registers (SFRs). The SFRs are located in the first 32 locations of each bank. There are General Purpose Registers implemented as static RAM in each bank. Table 4-1 lists the General Purpose Register available in each of the four banks.

TABLE 4-1: GENERAL PURPOSE STATIC RAM REGISTERS

	PIC16F627A/628A	PIC16F648A
Bank0	20-7Fh	20-7Fh
Bank1	A0h-FF	A0h-FF
Bank2	120h-14Fh, 170h-17Fh	120h-17Fh
Bank3	1F0h-1FFh	1F0h-1FFh

Addresses F0h-FFh, 170h-17Fh and 1F0h-1FFh are implemented as common RAM and mapped back to addresses 70h-7Fh.

Table 4-2 lists how to access the four banks of registers via the Status register bits RP1 and RP0.

TABLE 4-2: ACCESS TO BANKS OF REGISTERS

Bank	RP1	RP0
0	0	0
1	0	1
2	1	0
3	1	1

4.2.1 GENERAL PURPOSE REGISTER FILE

The register file is organized as 224 x 8 in the PIC16F627A/628A and 256 x 8 in the PIC16F648A. Each is accessed either directly or indirectly through the File Select Register (FSR), See **Section 4.4 "Indirect Addressing, INDF and FSR Registers"**.

5.3 I/O Programming Considerations

5.3.1 BIDIRECTIONAL I/O PORTS

Any instruction that writes operates internally as a read followed by a write operation. The BCF and BSF instructions, for example, read the register into the CPU, execute the bit operation and write the result back to the register. Caution must be used when these instructions are applied to a port with both inputs and outputs defined. For example, a BSF operation on bit 5 of PORTB will cause all eight bits of PORTB to be read into the CPU. Then the BSF operation takes place on bit 5 and PORTB is written to the output latches. If another bit of PORTB is used as a bidirectional I/O pin (e.g., bit 0) and is defined as an input at this time, the input signal present on the pin itself would be read into the CPU and rewritten to the data latch of this particular pin, overwriting the previous content. As long as the pin stays in the Input mode, no problem occurs. However, if bit 0 is switched into Output mode later on, the content of the data latch may now be unknown.

Reading a port register reads the values of the port pins. Writing to the port register writes the value to the port latch. When using read-modify-write instructions (ex. BCF, BSF, etc.) on a port, the value of the port pins is read, the desired operation is done to this value, and this value is then written to the port latch.

Example 5-2 shows the effect of two sequential read-modify-write instructions (ex., BCF, BSF, etc.) on an I/O port.

A pin actively outputting a Low or High should not be driven from external devices at the same time in order to change the level on this pin ("wired-OR", "wired-AND"). The resulting high output currents may damage the chip.

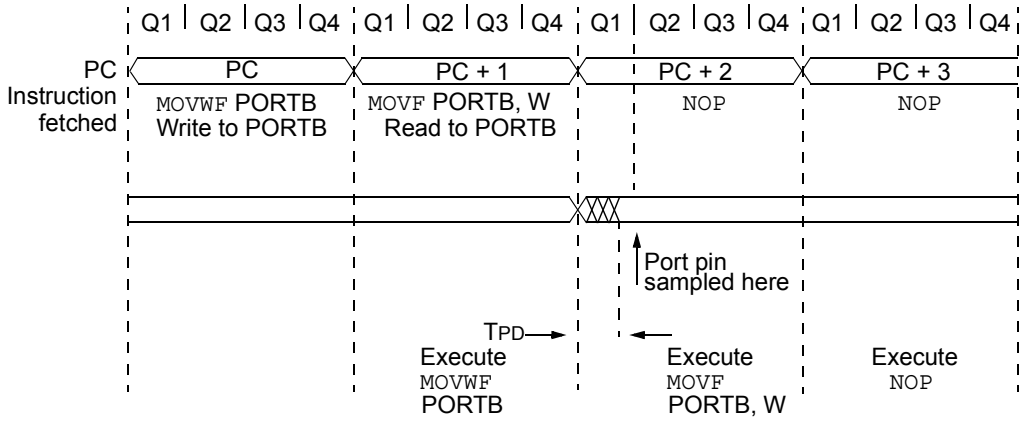
EXAMPLE 5-2: READ-MODIFY-WRITE INSTRUCTIONS ON AN I/O PORT

```
;Initial PORT settings:PORTB<7:4> Inputs
;                      PORTB<3:0> Outputs
;PORTB<7:6> have external pull-up and are
;not connected to other circuitry
;
;                      PORT latchPORT Pins
                       ---------- ----------
    BCF STATUS, RP0    ;
    BCF PORTB, 7       ;01pp pppp 11pp pppp
    BSF STATUS, RP0    ;
    BCF TRISB, 7       ;10pp pppp 11pp pppp
    BCF TRISB, 6       ;10pp pppp 10pp pppp
;
;Note that the user may have expected the
;pin values to be 00pp pppp. The 2nd BCF
;caused RB7 to be latched as the pin value
;(High).
```

5.3.2 SUCCESSIVE OPERATIONS ON I/O PORTS

The actual write to an I/O port happens at the end of an instruction cycle, whereas for reading, the data must be valid at the beginning of the instruction cycle (Figure 5-16). Therefore, care must be exercised if a write followed by a read operation is carried out on the same I/O port. The sequence of instructions should be such to allow the pin voltage to stabilize (load dependent) before the next instruction, which causes that file to be read into the CPU, is executed. Otherwise, the previous state of that pin may be read into the CPU rather than the new state. When in doubt, it is better to separate these instructions with a NOP or another instruction not accessing this I/O port.

FIGURE 5-16: SUCCESSIVE I/O OPERATION



Note 1: This example shows write to PORTB followed by a read from PORTB.

2: Data setup time = (0.25 TCY - TPD) where TCY = instruction cycle and TPD = propagation delay of Q1 cycle to output valid. Therefore, at higher clock frequencies, a write followed by a read may be problematic.

7.3 Timer1 Operation in Asynchronous Counter Mode

If control bit $\overline{\text{T1SYNC}}$ (T1CON<2>) is set, the external clock input is not synchronized. The timer continues to increment asynchronous to the internal phase clocks. The timer will continue to run during Sleep and can generate an interrupt on overflow, which will wake-up the processor. However, special precautions in software are needed to read/write the timer (**Section 7.3.2 "Reading and Writing Timer1 in Asynchronous Counter Mode"**).

Note:	In Asynchronous Counter mode, Timer1 cannot be used as a time base for capture or compare operations.

7.3.1 EXTERNAL CLOCK INPUT TIMING WITH UNSYNCHRONIZED CLOCK

If control bit $\overline{\text{T1SYNC}}$ is set, the timer will increment completely asynchronously. The input clock must meet certain minimum high and low time requirements. Refer to Table 17-8 in the Electrical Specifications Section, timing parameters 45, 46 and 47.

7.3.2 READING AND WRITING TIMER1 IN ASYNCHRONOUS COUNTER MODE

Reading the TMR1H or TMR1L register, while the timer is running from an external asynchronous clock, will produce a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself poses certain problems since the timer may overflow between the reads.

For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers while the register is incrementing. This may produce an unpredictable value in the timer register.

Reading the 16-bit value requires some care. Example 7-1 is an example routine to read the 16-bit timer value. This is useful if the timer cannot be stopped.

EXAMPLE 7-1: READING A 16-BIT FREE-RUNNING TIMER

```
; All interrupts are disabled
    MOVF    TMR1H, W    ;Read high byte
    MOVWF   TMPH        ;
    MOVF    TMR1L, W    ;Read low byte
    MOVWF   TMPL        ;
    MOVF    TMR1H, W    ;Read high byte
    SUBWF   TMPH, W     ;Sub 1st read with
                        ;2nd read
    BTFSC   STATUS,Z    ;Is result = 0
    GOTO    CONTINUE    ;Good 16-bit read
;
; TMR1L may have rolled over between the
; read of the high and low bytes. Reading
; the high and low bytes now will read a good
; value.
;
    MOVF    TMR1H, W    ;Read high byte
    MOVWF   TMPH        ;
    MOVF    TMR1L, W    ;Read low byte
    MOVWF   TMPL        ;
; Re-enable the Interrupts (if required)
CONTINUE                ;Continue with your
                        ;code
```

10.6 Comparator Interrupts

The comparator interrupt flag is set whenever there is a change in the output value of either comparator. Software will need to maintain information about the status of the output bits, as read from CMCON<7:6>, to determine the actual change that has occurred. The CMIF bit, PIR1<6>, is the comparator interrupt flag. The CMIF bit must be reset by clearing '0'. Since it is also possible to write a '1' to this register, a simulated interrupt may be initiated.

The CMIE bit (PIE1<6>) and the PEIE bit (INTCON<6>) must be set to enable the interrupt. In addition, the GIE bit must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMIF bit will still be set if an interrupt condition occurs.

Note:	If a change in the CMCON register (C1OUT or C2OUT) should occur when a read operation is being executed (start of the Q2 cycle), then the CMIF (PIR1<6>) interrupt flag may not get set.

The user, in the interrupt service routine, can clear the interrupt in the following manner:

a) Any write or read of CMCON. This will end the mismatch condition.

b) Clear flag bit CMIF.

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

10.7 Comparator Operation During Sleep

When a comparator is active and the device is placed in Sleep mode, the comparator remains active and the interrupt is functional if enabled. This interrupt will wake-up the device from Sleep mode when enabled. While the comparator is powered-up, higher Sleep currents than shown in the power-down current specification will occur. Each comparator that is operational will consume additional current as shown in the comparator specifications. To minimize power consumption while in Sleep mode, turn off the comparators, CM<2:0> = 111, before entering Sleep. If the device wakes up from Sleep, the contents of the CMCON register are not affected.

10.8 Effects of a Reset

A device Reset forces the CMCON register to its Reset state. This forces the Comparator module to be in the comparator Reset mode, CM<2:0> = 000. This ensures that all potential inputs are analog inputs. Device current is minimized when analog inputs are present at Reset time. The comparators will be powered-down during the Reset interval.

10.9 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 10-4. Since the analog pins are connected to a digital output, they have reverse biased diodes to V_{DD} and V_{SS}. The analog input therefore, must be between V_{SS} and V_{DD}. If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up may occur. A maximum source impedance of 10 kΩ is recommended for the analog sources. Any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current.

REGISTER 12-2: RCSTA – RECEIVE STATUS AND CONTROL REGISTER (ADDRESS: 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN**: Serial Port Enable bit
(Configures RB1/RX/DT and RB2/TX/CK pins as serial port pins when bits TRISB<2:1> are set)
1 = Serial port enabled
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode - master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - slave:
Unused in this mode

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADEN**: Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enable interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit
Asynchronous mode 8-bit (RX9 = 0):
Unused in this mode
Synchronous mode
Unused in this mode

bit 2 **FERR**: Framing Error bit
1 = Framing error (Can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (Can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D**: 9th bit of received data (Can be parity bit)

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

Follow these steps when setting up an Asynchronous Reception:

1. TRISB<1> and TRISB<2> should both be set to '1' to configure the RB1/RX/DT and RB2/TX/CK pins as inputs. Output drive, when required, is controlled by the peripheral circuitry.
2. Initialize the SPBRG register for the appropriate baud rate. If a high-speed baud rate is desired, set bit BRGH. (**Section 12.1 "USART Baud Rate Generator (BRG)"**).
3. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
4. If interrupts are desired, then set enable bit RCIE.
5. If 9-bit reception is desired, then set bit RX9.
6. Enable the reception by setting bit CREN.
7. Flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE was set.
8. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
9. Read the 8-bit received data by reading the RCREG register.
10. If an OERR error occurred, clear the error by clearing enable bit CREN.

TABLE 12-7: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other Resets
0Ch	PIR1	EEIF	CMIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF	0000 -000	0000 -000
18h	RCSTA	SPEN	RX9	SREN	CREN	ADEN	FERR	OERR	RX9D	0000 000x	0000 000x
1Ah	RCREG	USART Receive Data Register								0000 0000	0000 0000
8Ch	PIE1	EEIE	CMIE	RCIE	TXIE	—	CCP1IE	TMR2IE	TMR1IE	0000 -000	0000 -000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.

12.5.2 USART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical except in the case of the Sleep mode. Also, bit SREN is a “don’t care” in Slave mode.

If receive is enabled by setting bit CREN prior to the SLEEP instruction, then a word may be received during Sleep. On completely receiving the word, the RSR register will transfer the data to the RCREG register and if enable bit RCIE bit is set, the interrupt generated will wake the chip from Sleep. If the global interrupt is enabled, the program will branch to the interrupt vector (0004h).

Follow these steps when setting up a Synchronous Slave Reception:

1. TRISB<1> and TRISB<2> should both be set to ‘1’ to configure the RB1/RX/DT and RB2/TX/CK pins as inputs. Output drive, when required, is controlled by the peripheral circuitry.
2. Enable the synchronous master serial port by setting bits SYNC and SPEN and clearing bit CSRC.
3. If interrupts are desired, then set enable bit RCIE.
4. If 9-bit reception is desired, then set bit RX9.
5. To enable reception, set enable bit CREN.
6. Flag bit RCIF will be set when reception is complete and an interrupt will be generated, if enable bit RCIE was set.
7. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If an OERR error occurred, clear the error by clearing bit CREN.

TABLE 12-11: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other Resets
0Ch	PIR1	EEIF	CMIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF	0000 -000	0000 -000
18h	RCSTA	SPEN	RX9	SREN	CREN	ADEN	FERR	OERR	RX9D	0000 000x	0000 000x
19h	TXREG	USART Transmit Data Register								0000 0000	0000 0000
8Ch	PIE1	EEIE	CMIE	RCIE	TXIE	—	CCP1IE	TMR2IE	TMR1IE	0000 -000	0000 -000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

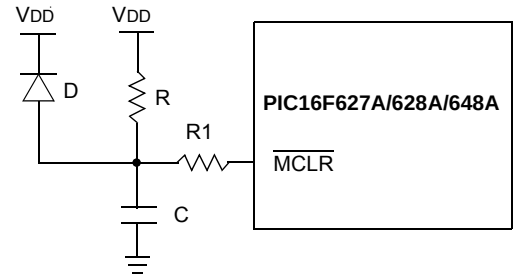
Legend: x = unknown, - = unimplemented read as ‘0’. Shaded cells are not used for synchronous slave transmission.

TABLE 12-12: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other Resets
0Ch	PIR1	EEIF	CMIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF	0000 -000	0000 -000
18h	RCSTA	SPEN	RX9	SREN	CREN	ADEN	FERR	OERR	RX9D	0000 000x	0000 000x
1Ah	RCREG	USART Receive Data Register								0000 0000	0000 0000
8Ch	PIE1	EEIE	CMIE	RCIE	TXIE	—	CCP1IE	TMR2IE	TMR1IE	0000 -000	0000 -000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented read as ‘0’. Shaded cells are not used for synchronous slave reception.

FIGURE 14-11: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)

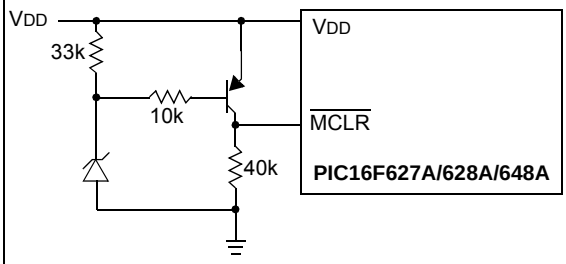


Note 1: External Power-on Reset circuit is required only if VDD power-up slope is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.

2: R < 40 kΩ is recommended to make sure that voltage drop across R does not violate the device's electrical specification.

3: R1 = 100Ω to 1 kΩ will limit any current flowing into MCLR from external capacitor C in the event of MCLR/VPP pin break-down due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

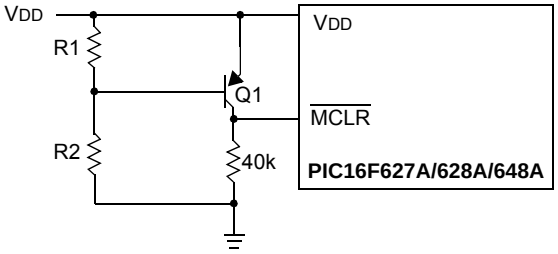
FIGURE 14-12: EXTERNAL BROWN-OUT PROTECTION CIRCUIT 1



Note 1: This circuit will activate Reset when VDD goes below (Vz + 0.7V) where Vz = Zener voltage.

2: Internal Brown-out Reset circuitry should be disabled when using this circuit.

FIGURE 14-13: EXTERNAL BROWN-OUT PROTECTION CIRCUIT 2



Note 1: This brown-out circuit is less expensive, albeit less accurate. Transistor Q1 turns off when VDD is below a certain level such that:

$$V_{DD} \times \frac{R1}{R1 + R2} = 0.7\ V$$

2: Internal Brown-out Reset should be disabled when using this circuit.

3: Resistors should be adjusted for the characteristics of the transistor.

14.5 Interrupts

The PIC16F627A/628A/648A has 10 sources of interrupt:

- External Interrupt RB0/INT
- TMR0 Overflow Interrupt
- PORTB Change Interrupts (pins RB<7:4>)
- Comparator Interrupt
- USART Interrupt TX
- USART Interrupt RX
- CCP Interrupt
- TMR1 Overflow Interrupt
- TMR2 Match Interrupt
- Data EEPROM Interrupt

The Interrupt Control register (INTCON) records individual interrupt requests in flag bits. It also has individual and global interrupt enable bits.

A Global Interrupt Enable bit, GIE (INTCON<7>) enables (if set) all un-masked interrupts or disables (if cleared) all interrupts. Individual interrupts can be disabled through their corresponding enable bits in INTCON register. GIE is cleared on Reset.

The "return-from-interrupt" instruction, RETFIE, exits interrupt routine as well as sets the GIE bit, which re-enables RB0/INT interrupts.

The INT pin interrupt, the RB port change interrupt and the TMR0 overflow interrupt flags are contained in the INTCON register.

The peripheral interrupt flag is contained in the special register PIR1. The corresponding interrupt enable bit is contained in special registers PIE1.

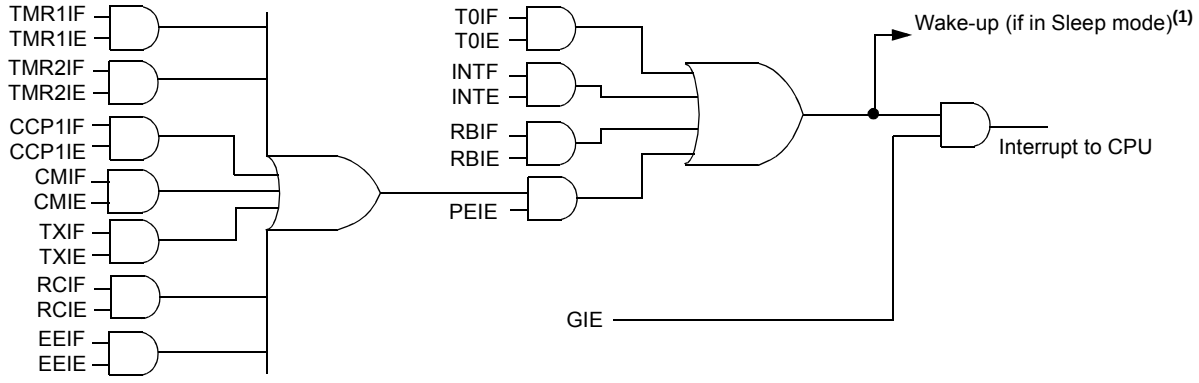
When an interrupt is responded to, the GIE is cleared to disable any further interrupt, the return address is pushed into the stack and the PC is loaded with 0004h. Once in the interrupt service routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid RB0/INT recursive interrupts.

For external interrupt events, such as the INT pin or PORTB change interrupt, the interrupt latency will be three or four instruction cycles. The exact latency depends when the interrupt event occurs (Figure 14-15). The latency is the same for one or two-cycle instructions. Once in the interrupt service routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid multiple interrupt requests.

Note 1: Individual interrupt flag bits are set regardless of the status of their corresponding mask bit or the GIE bit.

2: When an instruction that clears the GIE bit is executed, any interrupts that were pending for execution in the next cycle are ignored. The CPU will execute a NOP in the cycle immediately following the instruction which clears the GIE bit. The interrupts which were ignored are still pending to be serviced when the GIE bit is set again.

FIGURE 14-14: INTERRUPT LOGIC



Note 1: Some peripherals depend upon the system clock for operation. Since the system clock is suspended during Sleep, only those peripherals which do not depend upon the system clock will wake the part from Sleep. See **Section 14.8.1 "Wake-up from Sleep"**.

15.2 Instruction Descriptions

ADDLW — Add Literal and W

Syntax: [*label*] ADDLW k

Operands: $0 \le k \le 255$

Operation: (W) + k → (W)

Status Affected: C, DC, Z

Encoding:

11	111x	kkkk	kkkk

Description: The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.

Words: 1

Cycles: 1

Example

```
ADDLW   0x15
```

Before Instruction
W = 0x10
After Instruction
W = 0x25

ANDLW — AND Literal with W

Syntax: [*label*] ANDLW k

Operands: $0 \le k \le 255$

Operation: (W) .AND. (k) → (W)

Status Affected: Z

Encoding:

11	1001	kkkk	kkkk

Description: The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example

```
ANDLW   0x5F
```

Before Instruction
W = 0xA3
After Instruction
W = 0x03

ADDWF — Add W and f

Syntax: [*label*] ADDWF f,d

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

Operation: (W) + (f) → (dest)

Status Affected: C, DC, Z

Encoding:

00	0111	dfff	ffff

Description: Add the contents of the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example

```
ADDWF   REG1, 0
```

Before Instruction
W = 0x17
REG1 = 0xC2
After Instruction
W = 0xD9
REG1 = 0xC2
Z = 0
C = 0
DC = 0

ANDWF — AND W with f

Syntax: [*label*] ANDWF f,d

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

Operation: (W) .AND. (f) → (dest)

Status Affected: Z

Encoding:

00	0101	dfff	ffff

Description: AND the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example

```
ANDWF   REG1, 1
```

Before Instruction
W = 0x17
REG1 = 0xC2
After Instruction
W = 0x17
REG1 = 0x02

IORLW **Inclusive OR Literal with W**

Syntax:	[*label*] IORLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .OR. k → (W)
Status Affected:	Z

Encoding:	11	1000	kkkk	kkkk

Description: The contents of the W register is OR'ed with the eight-bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example

```
IORLW   0x35
```

Before Instruction
W = 0x9A
After Instruction
W = 0xBF
Z = 0

IORWF **Inclusive OR W with f**

Syntax:	[*label*] IORWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) .OR. (f) → (dest)
Status Affected:	Z

Encoding:	00	0100	dfff	ffff

Description: Inclusive OR the W register with register 'f'. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.

Words: 1

Cycles: 1

Example

```
IORWF   REG1, 0
```

Before Instruction
REG1 = 0x13
W = 0x91
After Instruction
REG1 = 0x13
W = 0x93
Z = 1

MOVLW **Move Literal to W**

Syntax:	[*label*] MOVLW k
Operands:	$0 \le k \le 255$
Operation:	k → (W)
Status Affected:	None

Encoding:	11	00xx	kkkk	kkkk

Description: The eight bit literal 'k' is loaded into W register. The "don't cares" will assemble as '0's.

Words: 1

Cycles: 1

Example

```
MOVLW   0x5A
```

After Instruction
W = 0x5A

MOVF **Move f**

Syntax:	[*label*] MOVF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) → (dest)
Status Affected:	Z

Encoding:	00	1000	dfff	ffff

Description: The contents of register 'f' is moved to a destination dependent upon the status of 'd'. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected.

Words: 1

Cycles: 1

Example

```
MOVF    REG1, 0
```

After Instruction
W = value in REG1 register
Z = 1

RETLW **Return with Literal in W**

Syntax: [*label*] RETLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow (W)$;
$TOS \rightarrow PC$

Status Affected: None

Encoding:

11	01xx	kkkk	kkkk

Description: The W register is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.

Words: 1

Cycles: 2

Example

```
CALL TABLE;W contains table
    ;offset value
•   ;W now has table value
•
•
TABLE
ADDWF PC;W = offset
RETLW k1;Begin table
RETLW k2;
•
•
•
RETLW kn; End of table
```

Before Instruction
W = 0x07
After Instruction
W = value of k8

RETURN **Return from Subroutine**

Syntax: [*label*] RETURN

Operands: None

Operation: $TOS \rightarrow PC$

Status Affected: None

Encoding:

00	0000	0000	1000

Description: Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction.

Words: 1

Cycles: 2

Example

```
RETURN
```

After Interrupt
PC = TOS

RLF **Rotate Left f through Carry**

Syntax: [*label*] RLF f,d

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

Operation: See description below

Status Affected: C

Encoding:

00	1101	dfff	ffff

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

Words: 1

Cycles: 1

Example

```
RLF     REG1, 0
```

Before Instruction
REG1=1110 0110
C = 0
After Instruction
REG1=1110 0110
W = 1100 1100
C = 1

17.0 ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings(†)

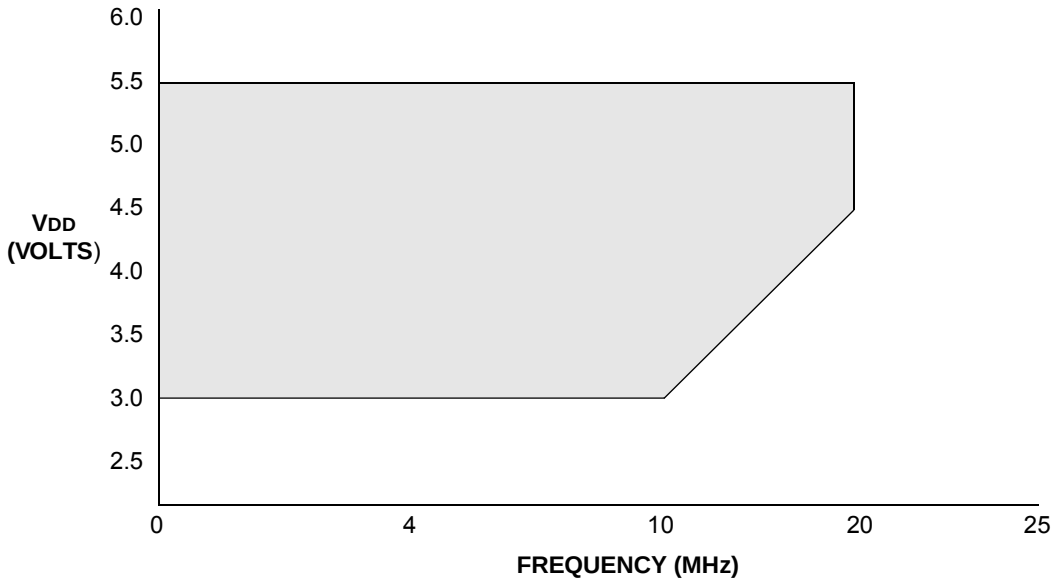
Ambient temperature under bias ..... -40 to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on V_{DD} with respect to V_{SS} ..... -0.3 to +6.5V
Voltage on $\overline{MCLR}$ and RA4 with respect to V_{SS} ..... -0.3 to +14V
Voltage on all other pins with respect to V_{SS} ..... -0.3V to V_{DD} + 0.3V
Total power dissipation(1) ..... 800 mW
Maximum current out of V_{SS} pin ..... 300 mA
Maximum current into V_{DD} pin ..... 250 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD}) ..... ± 20 mA
Output clamp current, I_{OK} (Vo < 0 or Vo > 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 and PORTB (Combined) ..... 200 mA
Maximum current sourced by PORTA and PORTB (Combined) ..... 200 mA

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

† **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.

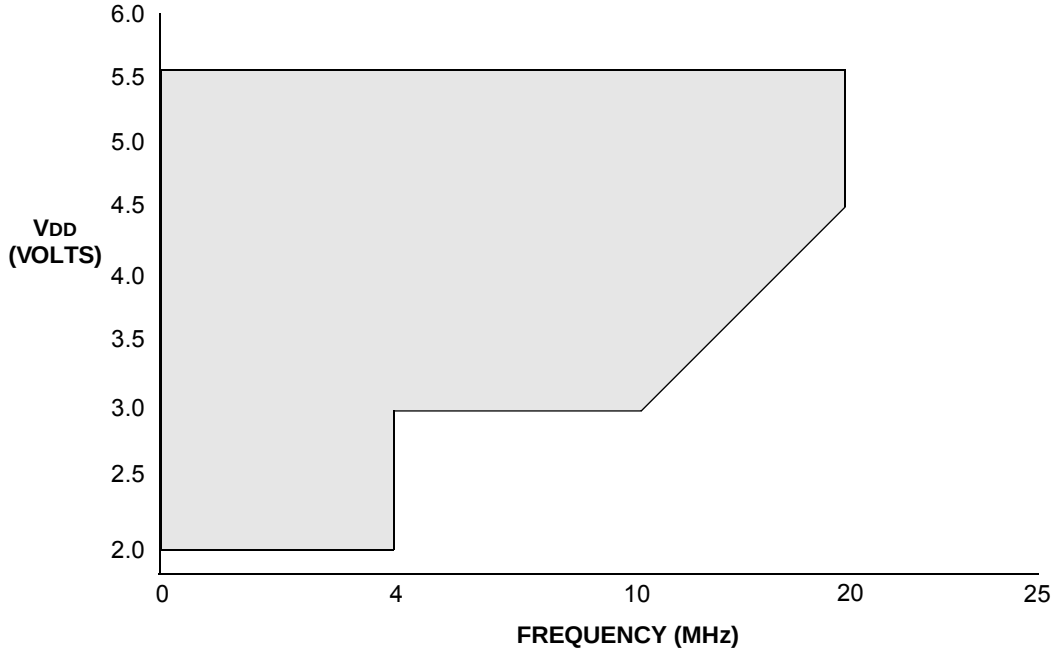
Note: Voltage spikes below V_{SS} at the $\overline{MCLR}$ pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100 Ω should be used when applying a "low" level to the $\overline{MCLR}$ pin rather than pulling this pin directly to V_{SS}.

FIGURE 17-1: PIC16F627A/628A/648A VOLTAGE-FREQUENCY GRAPH, -40°C ≤ TA ≤ +125°C



Note: The shaded region indicates the permissible combinations of voltage and frequency.

FIGURE 17-2: PIC16LF627A/628A/648A VOLTAGE-FREQUENCY GRAPH, -40°C ≤ TA ≤ +85°C



Note: The shaded region indicates the permissible combinations of voltage and frequency.

FIGURE 18-10: TYPICAL INTERNAL OSCILLATOR FREQUENCY vs. TEMPERATURE VDD = 3 VOLTS

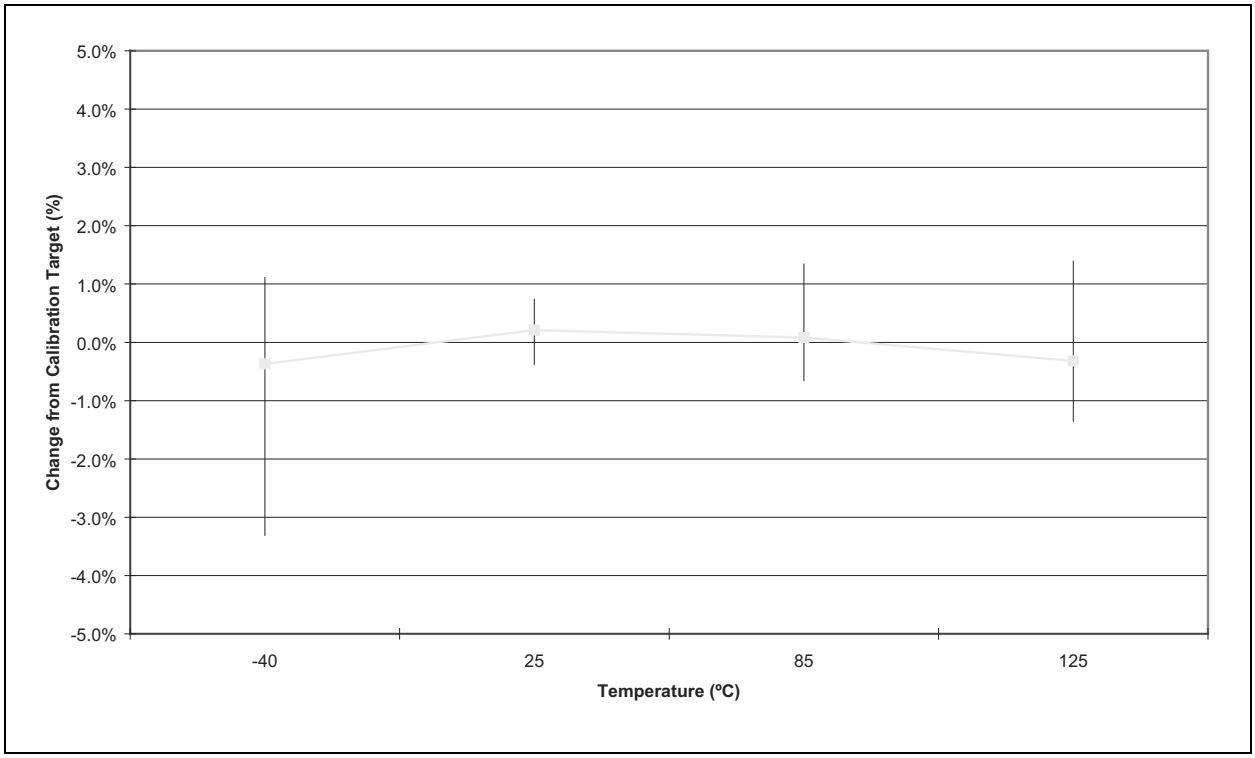
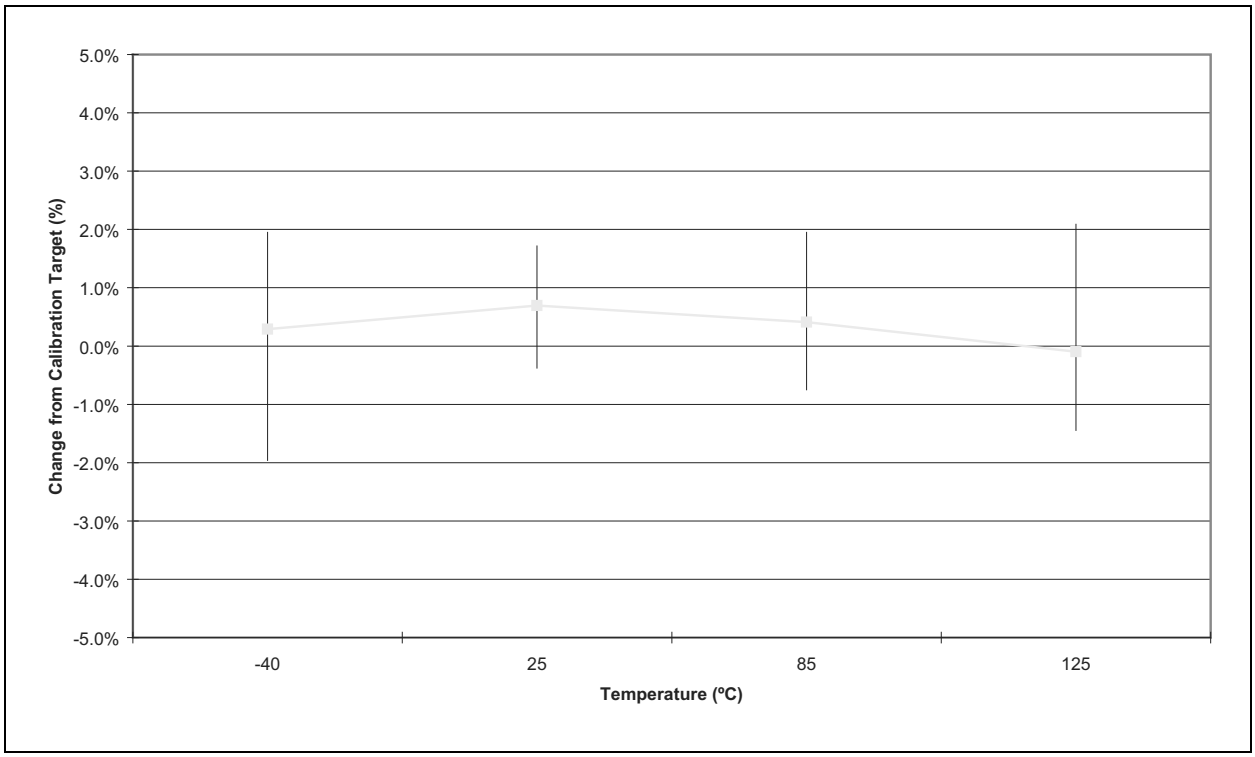
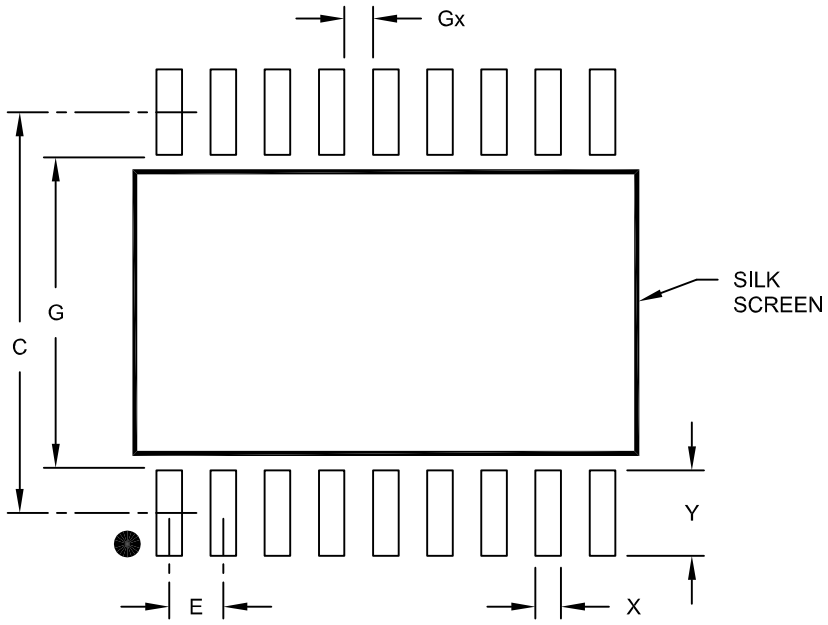


FIGURE 18-11: TYPICAL INTERNAL OSCILLATOR FREQUENCY vs. TEMPERATURE VDD = 2 VOLTS



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



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		1.27 BSC	
Contact Pad Spacing	C		9.40	
Contact Pad Width	X			0.60
Contact Pad Length	Y			2.00
Distance Between Pads	Gx	0.67		
Distance Between Pads	G	7.40		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

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

Microchip Technology Drawing No. C04-2051A

READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

To: Technical Publications Manager Total Pages Sent _____

RE: Reader Response

From: Name _____

Company _____

Address _____

City / State / ZIP / Country _____

Telephone: (_____) _____ - _____ FAX: (_____) _____ - _____

Application (optional):

Would you like a reply? ___Y ___N

Device: PIC16F627A/628A/648A Literature Number: DS40044G

Questions:

1. What are the best features of this document?

2. How does this document meet your hardware and software development needs?

3. Do you find the organization of this document easy to follow? If not, why?

4. What additions to the document do you think would enhance the structure and subject?

5. What deletions from the document could be made without affecting the overall usefulness?

6. Is there any incorrect or misleading information (what and where)?

7. How would you improve this document?