

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	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	28-VQFN Exposed Pad
Supplier Device Package	28-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16f648at-i-ml

PIC16F627A/628A/648A

Table of Contents

1.0 General Description	7
2.0 PIC16F627A/628A/648A Device Varieties	9
3.0 Architectural Overview	11
4.0 Memory Organization	17
5.0 I/O Ports	33
6.0 Timer0 Module	47
7.0 Timer1 Module	50
8.0 Timer2 Module	54
9.0 Capture/Compare/PWM (CCP) Module	57
10.0 Comparator Module	63
11.0 Voltage Reference Module	69
12.0 Universal Synchronous Asynchronous Receiver Transmitter (USART) Module.....	73
13.0 Data EEPROM Memory	91
14.0 Special Features of the CPU	97
15.0 Instruction Set Summary	117
16.0 Development Support	131
17.0 Electrical Specifications	135
18.0 DC and AC Characteristics Graphs and Tables	151
19.0 Packaging Information	163
Appendix A: Data Sheet Revision History.....	171
Appendix B: Device Differences	171
Appendix C: Device Migrations	172
Appendix D: Migrating from other PIC® Devices	172
The Microchip Web Site	173
Customer Change Notification Service	173
Customer Support.....	173
Reader Response	174
Product Identification System	179

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at docerrors@microchip.com or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

<http://www.microchip.com>

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; <http://www.microchip.com>
- Your local Microchip sales office (see last page)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our web site at www.microchip.com to receive the most current information on all of our products.

PIC16F627A/628A/648A

NOTES:

PIC16F627A/628A/648A

NOTES:

PIC16F627A/628A/648A

4.2.2 SPECIAL FUNCTION REGISTERS

The SFRs are registers used by the CPU and Peripheral functions for controlling the desired operation of the device (Table 4-3). These registers are static RAM.

The special registers can be classified into two sets (core and peripheral). The SFRs associated with the “core” functions are described in this section. Those related to the operation of the peripheral features are described in the section of that peripheral feature.

TABLE 4-3: SPECIAL REGISTERS SUMMARY BANK0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR Reset ⁽¹⁾	Details on Page
Bank 0											
00h	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	30
01h	TMR0	Timer0 Module's Register								xxxx xxxx	47
02h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	30
03h	STATUS	IRP	RP1	RP0	\overline{TO}	\overline{PD}	Z	DC	C	0001 1xxx	24
04h	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	30
05h	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxx 0000	33
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	38
07h	—	Unimplemented								—	—
08h	—	Unimplemented								—	—
09h	—	Unimplemented								—	—
0Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of Program Counter					--0 0000	30
0Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	26
0Ch	PIR1	EEIF	CMIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF	0000 -000	28
0Dh	—	Unimplemented								—	—
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	50
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	50
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	--00 0000	50
11h	TMR2	TMR2 Module's Register								0000 0000	54
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	54
13h	—	Unimplemented								—	—
14h	—	Unimplemented								—	—
15h	CCPR1L	Capture/Compare/PWM Register (LSB)								xxxx xxxx	57
16h	CCPR1H	Capture/Compare/PWM Register (MSB)								xxxx xxxx	57
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	57
18h	RCSTA	SPEN	RX9	SREN	CREN	ADEN	FERR	OERR	RX9D	0000 000x	74
19h	TXREG	USART Transmit Data Register								0000 0000	79
1Ah	RCREG	USART Receive Data Register								0000 0000	82
1Bh	—	Unimplemented								—	—
1Ch	—	Unimplemented								—	—
1Dh	—	Unimplemented								—	—
1Eh	—	Unimplemented								—	—
1Fh	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0000	63

Legend: — = Unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: For the initialization condition for registers tables, refer to Table 14-6 and Table 14-7.

PIC16F627A/628A/648A

4.2.2.3 INTCON Register

The INTCON register is a readable and writable register, which contains the various enable and flag bits for all interrupt sources except the comparator module. See **Section 4.2.2.4 “PIE1 Register”** and **Section 4.2.2.5 “PIR1 Register”** for a description of the comparator enable and flag bits.

Note: Interrupt flag bits get set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>).

REGISTER 4-3: INTCON – INTERRUPT CONTROL REGISTER (ADDRESS: 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
bit 7				bit 0			

- bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all un-masked interrupts
0 = Disables all interrupts
- bit 6 **PEIE:** Peripheral Interrupt Enable bit
1 = Enables all un-masked peripheral interrupts
0 = Disables all peripheral interrupts
- bit 5 **TOIE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt
- bit 4 **INTE:** RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt
- bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt
- bit 2 **TOIF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow
- bit 1 **INTF:** RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur
- bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = When at least one of the RB<7:4> pins changes state (must be cleared in software)
0 = None of the RB<7:4> pins have changed state

Legend:

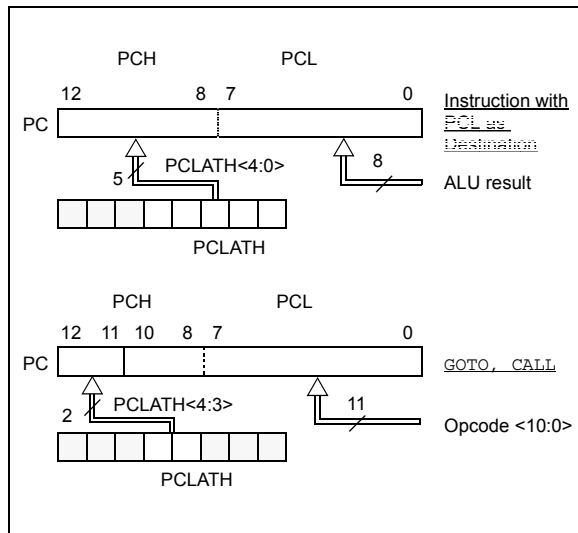
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

PIC16F627A/628A/648A

4.3 PCL and PCLATH

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

FIGURE 4-4: LOADING OF PC IN DIFFERENT SITUATIONS



4.3.1 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).

4.3.2 STACK

The PIC16F627A/628A/648A family has an 8-level deep x 13-bit wide hardware stack (Figure 4-1). 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.

4.4 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing.

Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses data pointed to by the File Select Register (FSR). Reading INDF itself indirectly will produce 00h. Writing to the INDF register indirectly results in a no-operation (although Status bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 4-5.

A simple program to clear RAM location 20h-2Fh using indirect addressing is shown in Example 4-1.

EXAMPLE 4-1: INDIRECT ADDRESSING

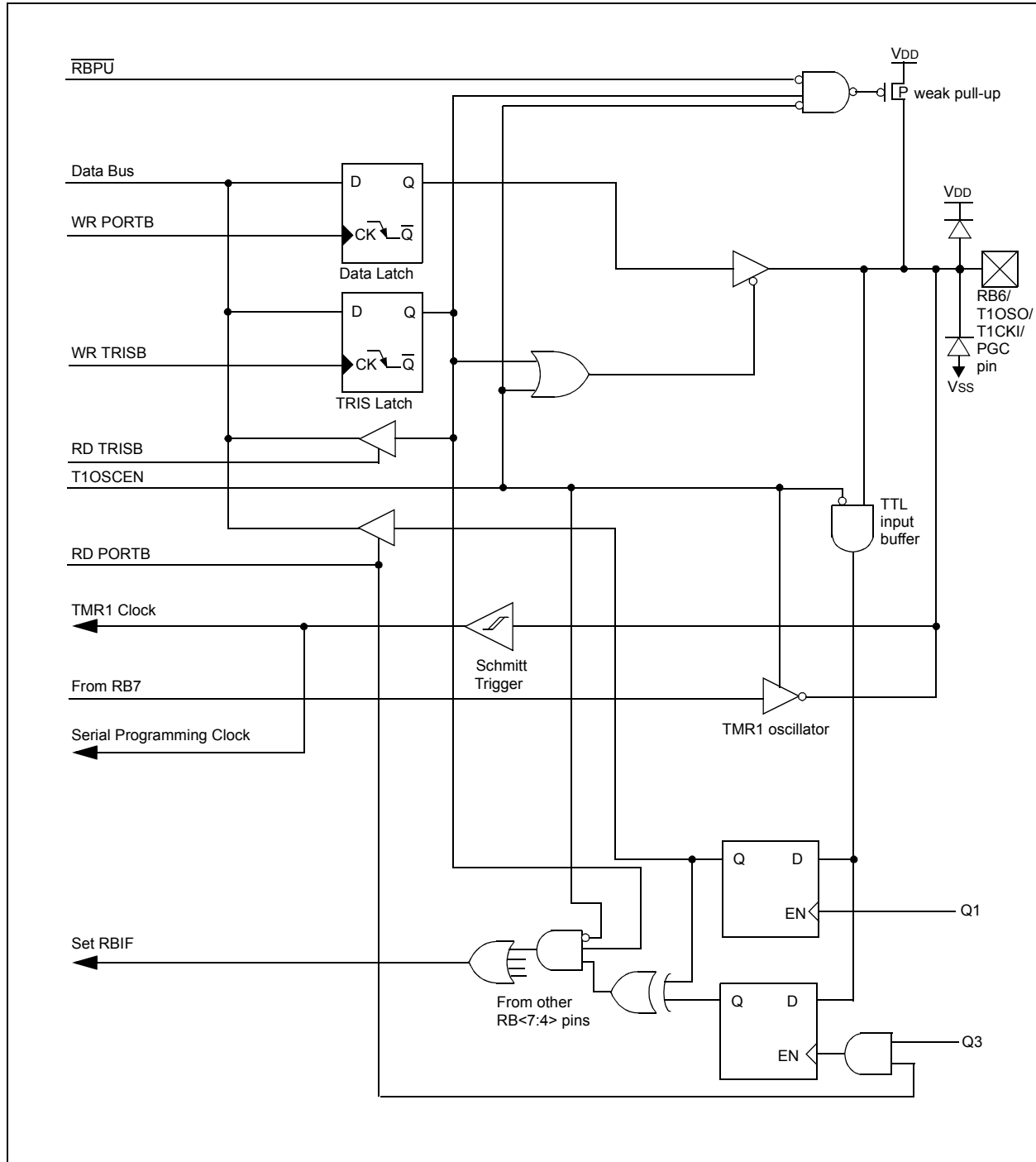
	MOVLW	0x20	;initialize pointer
	MOVWF	FSR	;to RAM
NEXT	CLRF	INDF	;clear INDF register
	INCF	FSR	;inc pointer
	BTFSS	FSR,4	;all done?
	GOTO	NEXT	;no clear next
			;yes continue

PIC16F627A/628A/648A

NOTES:

PIC16F627A/628A/648A

FIGURE 5-14: BLOCK DIAGRAM OF RB6/T1OSO/T1CKI/PGC PIN



7.1 Timer1 Operation in Timer Mode

Timer mode is selected by clearing the TMR1CS (T1CON<1>) bit. In this mode, the input clock to the timer is $F_{osc}/4$. The synchronize control bit T1SYNC (T1CON<2>) has no effect since the internal clock is always in sync.

7.2 Timer1 Operation in Synchronized Counter Mode

Counter mode is selected by setting bit TMR1CS. In this mode, the TMR1 register pair value increments on every rising edge of clock input on pin RB7/T1OSI/PGD when bit T1OSCEN is set or pin RB6/T1OSO/T1CKI/PGC when bit T1OSCEN is cleared.

If T1SYNC is cleared, then the external clock input is synchronized with internal phase clocks. The synchronization is done after the prescaler stage. The prescaler stage is an asynchronous ripple-counter.

In this configuration, during Sleep mode, the TMR1 register pair value will not increment even if the external clock is present, since the synchronization circuit is shut off. The prescaler however will continue to increment.

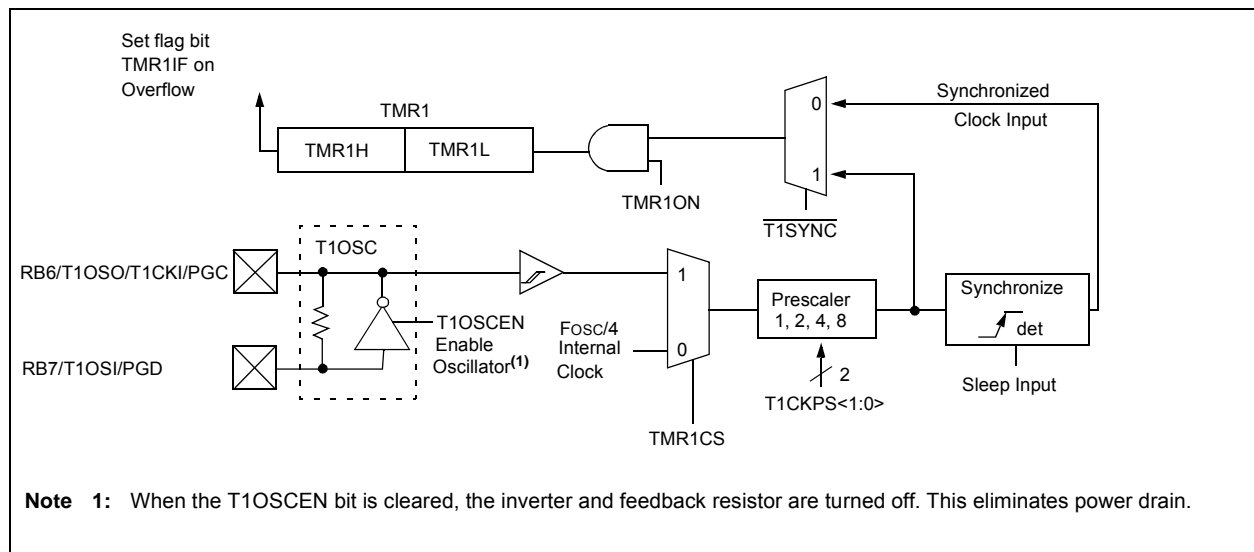
7.2.1 EXTERNAL CLOCK INPUT TIMING FOR SYNCHRONIZED COUNTER MODE

When an external clock input is used for Timer1 in Synchronized Counter mode, it must meet certain requirements. The external clock requirement is due to internal phase clock (Tosc) synchronization. Also, there is a delay in the actual incrementing of the TMR1 register pair value after synchronization.

When the prescaler is 1:1, the external clock input is the same as the prescaler output. The synchronization of T1CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks. Therefore, it is necessary for T1CKI to be high for at least 2 T_{osc} (and a small RC delay of 20 ns) and low for at least 2 T_{osc} (and a small RC delay of 20 ns). Refer to Table 17-8 in the Electrical Specifications Section, timing parameters 45, 46 and 47.

When a prescaler other than 1:1 is used, the external clock input is divided by the asynchronous ripple-counter type prescaler so that the prescaler output is symmetrical. In order for the external clock to meet the sampling requirement, the ripple-counter must be taken into account. Therefore, it is necessary for T1CKI to have a period of at least 4 T_{osc} (and a small RC delay of 40 ns) divided by the prescaler value. The only requirement on T1CKI high and low time is that they do not violate the minimum pulse width requirements of 10 ns). Refer to the appropriate electrical specifications in Table 17-8, parameters 45, 46 and 47.

FIGURE 7-1: TIMER1 BLOCK DIAGRAM



PIC16F627A/628A/648A

NOTES:

12.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. In Asynchronous mode, bit BRGH (TXSTA<2>) also controls the baud rate. In Synchronous mode, bit BRGH is ignored. Table 12-1 shows the formula for computation of the baud rate for different USART modes, which only apply in Master mode (internal clock).

Given the desired baud rate and Fosc, the nearest integer value for the SPBRG register can be calculated using the formula in Table 12-1. From this, the error in baud rate can be determined.

Example 12-1 shows the calculation of the baud rate error for the following conditions:

Fosc = 16 MHz

Desired Baud Rate = 9600

BRGH = 0

SYNC = 0

EQUATION 12-1: CALCULATING BAUD RATE ERROR

$$\text{Desired Baud Rate} = \frac{F_{osc}}{64(x + 1)}$$

$$9600 = \frac{16000000}{64(x + 1)}$$

$$x = 25.042$$

$$\text{Calculated Baud Rate} = \frac{16000000}{64(25 + 1)} = 9615$$

$$\text{Error} = \frac{(\text{Calculated Baud Rate} - \text{Desired Baud Rate})}{\text{Desired Baud Rate}}$$

$$= \frac{9615 - 9600}{9600} = 0.16\%$$

It may be advantageous to use the high baud rate (BRGH = 1) even for slower baud clocks. This is because the $F_{osc}/(16(X + 1))$ equation can reduce the baud rate error in some cases.

Writing a new value to the SPBRG register causes the BRG timer to be reset (or cleared) and ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

The data on the RB1/RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

TABLE 12-1: BAUD RATE FORMULA

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = $F_{osc}/(64(X+1))$	Baud Rate = $F_{osc}/(16(X+1))$
1	(Synchronous) Baud Rate = $F_{osc}/(4(X+1))$	NA

Legend: X = value in SPBRG (0 to 255)

TABLE 12-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

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
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
18h	RCSTA	SPEN	RX9	SREN	CREN	ADEN	FERR	OERR	RX9D	0000 000x	0000 000x
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented read as '0'. Shaded cells are not used for the BRG.

PIC16F627A/628A/648A

TABLE 12-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

BAUD RATE (K)	Fosc = 20 MHz			16 MHz			10 MHz		
	KBAUD	ERROR	SPBRG value (decimal)	KBAUD	ERROR	SPBRG value (decimal)	KBAUD	ERROR	SPBRG value (decimal)
0.3	NA	—	—	NA	—	—	NA	—	—
1.2	1.221	+1.73%	255	1.202	+0.16%	207	1.202	+0.16%	129
2.4	2.404	+0.16%	129	2.404	+0.16%	103	2.404	+0.16%	64
9.6	9.469	-1.36%	32	9.615	+0.16%	25	9.766	+1.73%	15
19.2	19.53	+1.73%	15	19.23	+0.16%	12	19.53	+1.73V	7
76.8	78.13	+1.73%	3	83.33	+8.51%	2	78.13	+1.73%	1
96	104.2	+8.51%	2	NA	—	—	NA	—	—
300	312.5	+4.17%	0	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	312.5	—	0	250	—	0	156.3	—	0
LOW	1.221	—	255	0.977	—	255	0.6104	—	255

BAUD RATE (K)	Fosc = 7.15909 MHz			5.0688 MHz			4 MHz		
	KBAUD	ERROR	SPBRG value (decimal)	KBAUD	ERROR	SPBRG value (decimal)	KBAUD	ERROR	SPBRG value (decimal)
0.3	NA	—	—	0.31	+3.13%	255	0.3005	-0.17%	207
1.2	1.203	+0.23%	92	1.2	0	65	1.202	+1.67%	51
2.4	2.380	-0.83%	46	2.4	0	32	2.404	+1.67%	25
9.6	9.322	-2.90%	11	9.9	+3.13%	7	NA	—	—
19.2	18.64	-2.90%	5	19.8	+3.13%	3	NA	—	—
76.8	NA	—	—	79.2	+3.13%	0	NA	—	—
96	NA	—	—	NA	—	—	NA	—	—
300	NA	—	—	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	111.9	—	0	79.2	—	0	62.500	—	0
LOW	0.437	—	255	0.3094	—	255	3.906	—	255

BAUD RATE (K)	Fosc = 3.579545 MHz			1 MHz			32.768 kHz		
	KBAUD	ERROR	SPBRG value (decimal)	KBAUD	ERROR	SPBRG value (decimal)	KBAUD	ERROR	SPBRG value (decimal)
0.3	0.301	+0.23%	185	0.300	+0.16%	51	0.256	-14.67%	1
1.2	1.190	-0.83%	46	1.202	+0.16%	12	NA	—	—
2.4	2.432	+1.32%	22	2.232	-6.99%	6	NA	—	—
9.6	9.322	-2.90%	5	NA	—	—	NA	—	—
19.2	18.64	-2.90%	2	NA	—	—	NA	—	—
76.8	NA	—	—	NA	—	—	NA	—	—
96	NA	—	—	NA	—	—	NA	—	—
300	NA	—	—	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	55.93	—	0	15.63	—	0	0.512	—	0
LOW	0.2185	—	255	0.0610	—	255	0.0020	—	255

FIGURE 12-5: ASYNCHRONOUS RECEPTION WITH ADDRESS DETECT

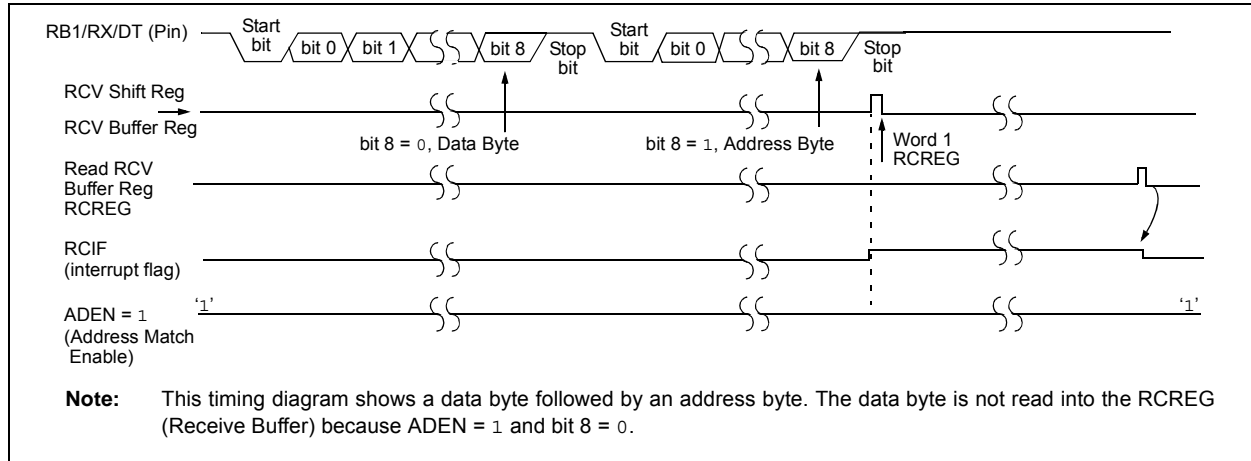


FIGURE 12-6: ASYNCHRONOUS RECEPTION WITH ADDRESS BYTE FIRST

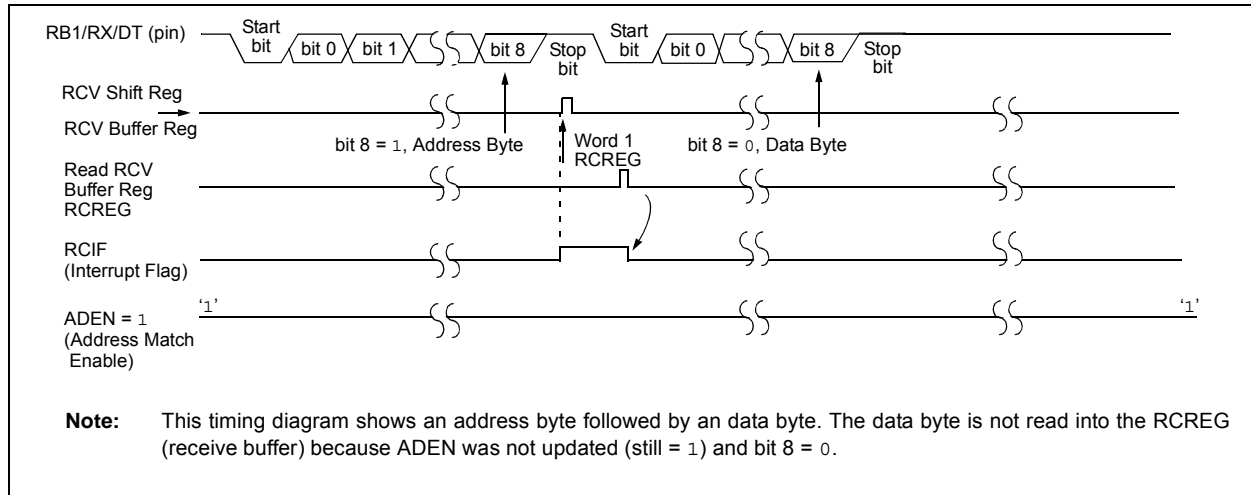
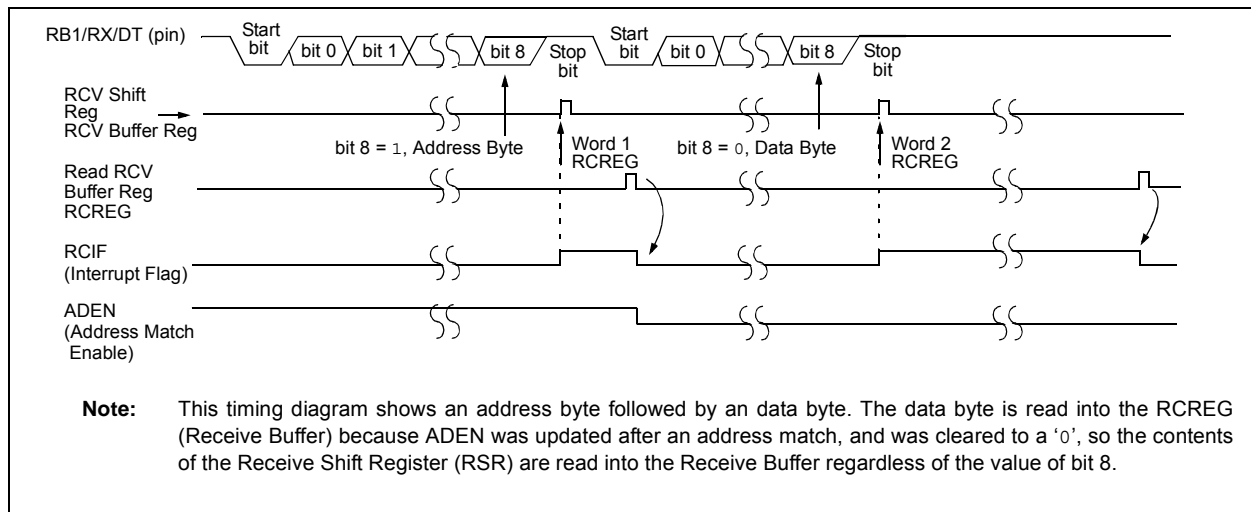


FIGURE 12-7: ASYNCHRONOUS RECEPTION WITH ADDRESS BYTE FIRST FOLLOWED BY VALID DATA BYTE



12.4 USART Synchronous Master Mode

In Synchronous Master mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit SYNC (TXSTA<4>). In addition enable bit SPEN (RCSTA<7>) is set in order to configure the RB2/TX/CK and RB1/RX/DT I/O pins to CK (clock) and DT (data) lines, respectively. The Master mode indicates that the processor transmits the master clock on the CK line. The Master mode is entered by setting bit CSRC (TXSTA<7>).

12.4.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 12-1. The heart of the transmitter is the Transmit (serial) Shift Register (TSR). The shift register obtains its data from the read/write transmit buffer register, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREG (if available). Once the TXREG register transfers the data to the TSR register (occurs in one Tcycle), the TXREG is empty and interrupt bit, TXIF (PIR1<4>) is set. The interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE1<4>). Flag bit TXIF will be set regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicates the status of the TXREG register, another bit TRMT (TXSTA<1>) shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory so it is not available to the user.

Transmission is enabled by setting enable bit TXEN (TXSTA<5>). The actual transmission will not occur until the TXREG register has been loaded with data. The first data bit will be shifted out on the next available rising edge of the clock on the CK line. Data out is stable around the falling edge of the synchronous clock (Figure 12-8). The transmission can also be started by first loading the TXREG register and then setting bit TXEN (Figure 12-9). This is advantageous when slow baud rates are selected, since the BRG is kept in Reset when bits TXEN, CREN and SREN are clear. Setting enable bit TXEN will start the BRG, creating a shift clock immediately. Normally, when transmission is first started, the TSR register is empty, so a transfer to the TXREG register will result in an immediate transfer to TSR resulting in an empty TXREG. Back-to-back transfers are possible.

Clearing enable bit TXEN during a transmission will cause the transmission to be aborted and will reset the transmitter. The DT and CK pins will revert to high-impedance. If either bit CREN or bit SREN is set during a transmission, the transmission is aborted and the DT pin reverts to a high-impedance state (for a reception). The CK pin will remain an output if bit CSRC is set (internal clock). The transmitter logic however is not reset although it is disconnected from the pins. In order to reset the transmitter, the user has to clear bit TXEN. If bit SREN is set (to interrupt an on-going transmission and receive a single word), then after the single word is received, bit SREN will be cleared and the serial port will revert back to transmitting since bit TXEN is still set. The DT line will immediately switch from high-impedance Receive mode to transmit and start driving. To avoid this, bit TXEN should be cleared.

In order to select 9-bit transmission, the TX9 (TXSTA<6>) bit should be set and the ninth bit should be written to bit TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to the TXREG register. This is because a data write to the TXREG can result in an immediate transfer of the data to the TSR register (if the TSR is empty). If the TSR was empty and the TXREG was written before writing the “new” TX9D, the “present” value of bit TX9D is loaded.

Follow these steps when setting up a Synchronous Master Transmission:

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 (**Section 12.1 “USART Baud Rate Generator (BRG)”**).
3. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
4. If interrupts are desired, then set enable bit TXIE.
5. If 9-bit transmission is desired, then set bit TX9.
6. Enable the transmission by setting bit TXEN.
7. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
8. Start each transmission by loading data to the TXREG register.

PIC16F627A/628A/648A

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.

PIC16F627A/628A/648A

INCF	Increment f	INCFSZ	Increment f, Skip if 0								
Syntax:	[<i>label</i>] INCF f,d	Syntax:	[<i>label</i>] INCFSZ f,d								
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$	Operands:	$0 \leq f \leq 127$ $d \in [0,1]$								
Operation:	$(f) + 1 \rightarrow (\text{dest})$	Operation:	$(f) + 1 \rightarrow (\text{dest})$, skip if result = 0								
Status Affected:	Z	Status Affected:	None								
Encoding:	<table><tr><td>00</td><td>1010</td><td>dfff</td><td>ffff</td></tr></table>	00	1010	dfff	ffff	Encoding:	<table><tr><td>00</td><td>1111</td><td>dfff</td><td>ffff</td></tr></table>	00	1111	dfff	ffff
00	1010	dfff	ffff								
00	1111	dfff	ffff								
Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.	Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'. If the result is '0', the next instruction, which is already fetched, is discarded. A NOP is executed instead making it a two-cycle instruction.								
Words:	1	Words:	1								
Cycles:	1	Cycles:	1(2)								
<u>Example</u>	<pre>INCF REG1, 1 Before Instruction REG1 = 0xFF Z = 0 After Instruction REG1 = 0x00 Z = 1</pre>	<u>Example</u>	<pre>HERE INCFSZ REG1, 1 GOTO LOOP CONTINUE • • • Before Instruction PC = address HERE After Instruction REG1 = REG1 + 1 if CNT = 0, PC = address CONTINUE if REG1≠ 0, PC = address HERE +1</pre>								

PIC16F627A/628A/648A

MOVWF Move W to f

Syntax:	[<i>label</i>]	MOVWF	f	
Operands:	$0 \leq f \leq 127$			
Operation:	(W) → (f)			
Status Affected:	None			
Encoding:	00	0000	1fff	ffff

Description: Move data from W register to register 'f'.

Words: 1

Cycles: 1

Example

MOVWF REG1
 Before Instruction
 REG1 = 0xFF
 W = 0x4F
 After Instruction
 REG1 = 0x4F
 W = 0x4F

OPTION Load Option Register

Syntax:	[<i>label</i>] OPTION			
Operands:	None			
Operation:	(W) → OPTION			
Status Affected:	None			
Encoding:	00	0000	0110	0010

Description: The contents of the W register are loaded in the OPTION register. This instruction is supported for code compatibility with PIC16C5X products. Since OPTION is a readable/writable register, the user can directly address it. Using only register instruction such as MOVWF.

Words: 1

Cycles: 1

Example

To maintain upward compatibility with future PIC® MCU products, do not use this instruction.

NOP No Operation

Syntax:	[<i>label</i>] NOP			
Operands:	None			
Operation:	No operation			
Status Affected:	None			
Encoding:	00	0000	0xx0	0000

Description: No operation.

Words: 1

Cycles: 1

Example

NOP

RETFIE Return from Interrupt

Syntax:	[<i>label</i>] RETFIE			
Operands:	None			
Operation:	TOS → PC, 1 → GIE			
Status Affected:	None			
Encoding:	00	0000	0000	1001

Description: Return from Interrupt. Stack is POPed and Top-of-Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a two-cycle instruction.

Words: 1

Cycles: 2

Example

RETFIE
 After Interrupt
 PC = TOS
 GIE = 1

16.7 MPLAB SIM Software Simulator

The MPLAB SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC® DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB SIM Software Simulator fully supports symbolic debugging using the MPLAB C Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

16.8 MPLAB REAL ICE In-Circuit Emulator System

MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs PIC® Flash MCUs and dsPIC® Flash DSCs with the easy-to-use, powerful graphical user interface of the MPLAB Integrated Development Environment (IDE), included with each kit.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

16.9 MPLAB ICD 3 In-Circuit Debugger System

MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost effective high-speed hardware debugger/programmer for Microchip Flash Digital Signal Controller (DSC) and microcontroller (MCU) devices. It debugs and programs PIC® Flash microcontrollers and dsPIC® DSCs with the powerful, yet easy-to-use graphical user interface of MPLAB Integrated Development Environment (IDE).

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

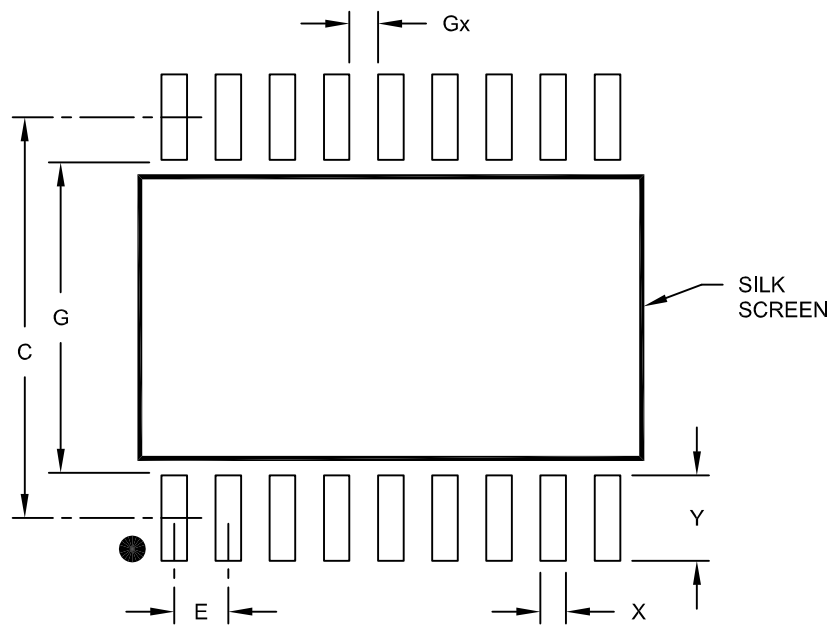
16.10 PICkit 3 In-Circuit Debugger/Programmer and PICkit 3 Debug Express

The MPLAB PICkit 3 allows debugging and programming of PIC® and dsPIC® Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB Integrated Development Environment (IDE). The MPLAB PICkit 3 is connected to the design engineer's PC using a full speed USB interface and can be connected to the target via an Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the reset line to implement in-circuit debugging and In-Circuit Serial Programming™.

The PICkit 3 Debug Express include the PICkit 3, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

PIC16F627A/628A/648A

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

PIC16F627A/628A/648A

APPENDIX C: DEVICE MIGRATIONS

This section describes the functional and electrical specification differences when migrating between functionally similar devices. (such as from a PIC16F627 to a PIC16F627A).

C.1 PIC16F627/628 to a PIC16F627A/628A

1. ER mode is now RC mode.
2. Code protection for the program memory has changed from code-protect sections of memory to code-protect of the whole memory. The configuration bits CP0 and CP1 in the PIC16F627/628 do not exist in the PIC16F627A/628A. They have been replaced with one configuration bit $\overline{\text{CP}}$.
3. "Brown-out Detect (BOD)" terminology has changed to "Brown-out Reset (BOR)" to better represent the function of the Brown-out circuitry.
4. Enabling Brown-out Reset (BOR) does not automatically enable the Power-up Timer (PWRT) the way it did in the PIC16F627/628.
5. INTRC is now called INTOSC.
6. Timer1 Oscillator is now designed for 32.768 kHz operation. In the PIC16F627/628, the Timer1 oscillator was designed to run up to 200 kHz.
7. The Dual-Speed Oscillator mode only works in the INTOSC oscillator mode. In the PIC16F627/628, the Dual-Speed Oscillator mode worked in both the INTRC and ER oscillator modes.

APPENDIX D: MIGRATING FROM OTHER PIC[®] DEVICES

This discusses some of the issues in migrating from other PIC MCU devices to the PIC16F627A/628A/648A family of devices.

D.1 PIC16C62X/CE62X to PIC16F627A/628A/648A Migration

See Microchip web site for availability (www.microchip.com).

D.2 PIC16C622A to PIC16F627A/628A/648A Migration

See Microchip web site for availability (www.microchip.com).

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