E·XFL



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 "<u>Embedded -</u> <u>Microcontrollers</u>"

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	1.75KB (1K x 14)
Program Memory Type	FLASH
EEPROM Size	128 x 8
RAM Size	224 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/pic16f627t-20i-ss

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

3.3 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 3-3 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> \rightarrow PCH). The lower example in the figure shows how the PC is loaded during a CALL or GOTO instruction (PCLATH<4:3> \rightarrow PCH).

FIGURE 3-3: LOADING OF PC IN DIFFERENT SITUATIONS



3.3.1 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 *"Implementing a Table Read"* (AN556).

3.3.2 STACK

The PIC16F62X family has an 8-level deep x 13-bit wide hardware stack (Figure 3-1 and Figure 3-2). 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.

3.4 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 3-4.

A simple program to clear RAM location 20h-2Fh using indirect addressing is shown in Example 3-1.

EXAMPLE 3-1: 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



FIGURE 5-9: BLOCK DIAGRAM OF



FIGURE 5-15: BLOCK DIAGRAM OF THE RB7/T10SI PIN



7.3 Timer1 Operation in Asynchronous Counter Mode

If control bit $\overline{T1SYNC}$ (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).

In Asynchronous Counter mode, Timer1 can not be used as a time-base for capture or compare operations.

7.3.1 EXTERNAL CLOCK INPUT TIMING WITH UNSYNCHRONIZED CLOCK

If control bit $\overline{T1SYNC}$ is set, the timer will increment completely asynchronously. The input clock must meet certain minimum high-time and low-time requirements. Refer to the appropriate Electrical Specifications section, Timing Parameters 45, 46, and 47.

7.3.2 READING AND WRITING TIMER1 IN ASYNCHRONOUS COUNTER MODE

Reading TMR1H or TMR1L while the timer is running, from an external asynchronous clock, will ensure 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 with your code
CONTINUE
```

The code example in Example 9-1 depicts the steps required to configure the Comparator module. RA3 and RA4 are configured as digital output. RA0 and RA1 are configured as the V- inputs and RA2 as the V+ input to both comparators.

EXAMPLE 9-1: INITIALIZING COMPARATOR MODULE

FLAG_REG	EQU	0X20
CLRF	FLAG_REG	;Init flag register
CLRF	PORTA	;Init PORTA
MOVF	CMCON, W	;Load comparator bits
ANDLW	0xC0	;Mask comparator bits
IORWF	FLAG_REG,F	;Store bits in flag register
MOVLW	0x03	;Init comparator mode
MOVWF	CMCON	;CM<2:0> = 011
BSF	STATUS, RPO	;Select Bank1
MOVLW	0x07	;Initialize data direction
MOVWF	TRISA	;Set RA<2:0> as inputs
		;RA<4:3> as outputs
		;TRISA<7:5> always read `0'
BCF	STATUS, RPO	;Select Bank 0
CALL	DELAY10	;10µs delay
MOVF	CMCON, F	;Read CMCONtoend change condition
BCF	PIR1,CMIF	;Clear pending interrupts
BSF	STATUS, RPO	;Select Bank 1
BSF	PIE1,CMIE	;Enable comparator interrupts
BCF	STATUS, RPO	;Select Bank 0
BSF	INTCON, PEIE	;Enable peripheral interrupts
BSF	INTCON, GIE	;Global interrupt enable

9.2 Comparator Operation

A single comparator is shown in Figure 9-2 along with the relationship between the analog input levels and the digital output. When the analog input at VIN+ is less than the analog input VIN-, the output of the comparator is a digital low level. When the analog input at VIN+ is greater than the analog input VIN-, the output of the comparator is a digital high level. The shaded areas of the output of the comparator in Figure 9-2 represent the uncertainty due to input offsets and response time.

9.3 Comparator Reference

An external or internal reference signal may be used depending on the Comparator Operating mode. The analog signal that is present at VIN- is compared to the signal at VIN+, and the digital output of the comparator is adjusted accordingly (Figure 9-2).



SINGLE COMPARATOR



9.3.1 EXTERNAL REFERENCE SIGNAL

When external voltage references are used, the comparator module can be configured to have the comparators operate from the same or different reference sources. However, threshold detector applications may require the same reference. The reference signal must be between VSs and VDD, and can be applied to either pin of the comparator(s).

9.3.2 INTERNAL REFERENCE SIGNAL

The Comparator module also allows the selection of an internally generated voltage reference for the comparators. Section 10.0, Voltage Reference Manual, contains a detailed description of the Voltage Reference module that provides this signal. The internal reference signal is used when the comparators are in mode CM<2:0>=010 (Figure 9-1). In this mode, the internal voltage reference is applied to the VIN+ pin of both comparators.

9.4 Comparator Response Time

Response time is the minimum time, after selecting a new reference voltage or input source, before the comparator output is ensured to have a valid level. If the internal reference is changed, the maximum delay of the internal voltage reference must be considered when using the comparator outputs. Otherwise the maximum delay of the comparators should be used (Table 17-1).

FIGURE 9-4: ANALOG INPUT MODE



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
1Fh	CMCON	C2OUT	C10UT	C2INV	C1NV	CIS	CM2	CM1	CM0	0000 0000	0000 0000
0Bh/8Bh/ 10Bh/18Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	EEIF	CMIF	RCIF	TXIF	_	CCP1IF	TMR2IF	TMR1IF	0000 -000	0000 -000
8Ch	PIE1	EEIE	CMIE	RCIE	TXIE	_	CCP1IE	TMR2IE	TMR1IE	0000 -000	0000 -000
85h	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 1111	1111 1111

Legend: x = Unknown, u = Unchanged, - = Unimplemented, read as '0'

PIC16F62X

ER 12-2:	RCSTA: R	ECEIVE S	TATUS AI	ND CONTR	OL REGIST	ER (ADDR	ESS: 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: Seria (Configures 1 = Serial po 0 = Serial po	al Port Enable RB1/RX/DT a ort enabled ort disabled	e bit and RB2/TX/	CK pins as se	rial port pins wh	en bits TRISI	3<2:17> are	set)	
bit 6	RX9 : 9-bit R 1 = Selects 9 0 = Selects 9	eceive Enabl 9-bit receptio 8-bit receptio	le bit n n						
bit 5	SREN: Sing Asynchronou Don't care Synchronou 1 = Enab 0 = Disab This bit is Synchronou Unused in	le Receive Er us mode: e s mode - mas les single rec les single rec cleared after s mode - slav n this mode	nable bit ster: seive ceive r reception is <u>/e</u> :	complete.					
bit 4	CREN : Cont <u>Asynchronou</u> 1 = Enab 0 = Disat <u>Synchronou</u> 1 = Enab 0 = Disat	inuous Rece us mode: les continuou oles continuou <u>s mode</u> : les continuou oles continuou	ive Enable bi us receive us receive us receive un us receive	t til enable bit C	REN is cleared	(CREN overr	ides SREN)		
bit 3	ADEN: Addr Asynchronou 1 = Enabl 0 = Disab Asynchronou Unused ir Synchron Unused ir	ress Detect E us mode 9-bi es address d les address d us mode 8-bi n this mode ous mode n this mode	inable bit <u>t (RX9 = 1)</u> : letection, ena detection, all <u>t (RX9=0)</u> :	able interrupt a bytes are rece	ind load of the r ived, and ninth	eceive buffer bit can be us	when RSR< ed as PARIT	8> is set Ƴ bit	
bit 2	FERR: Fram 1 = Framing 0 = No frami	ning Error bit error (Can b ing error	e updated by	reading RCR	EG register and	receive next	valid byte)		
bit 1	OERR: Overrun Error bit 1 = Overrun error (Can be cleared by clearing bit CREN) 0 = No overrun error								
bit 0	RX9D : 9th b	it of received	data (Can b	e PARITY bit)					
	Legend:								
	R = Reada	ble bit	W = V	Vritable bit	U = Unimp	lemented b	it, read as '	0'	

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

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown





12.2 USART Asynchronous Mode

In this mode, the USART uses standard non-return to zero (NRZ) format (one START bit, eight or nine data bits and one STOP bit). The most common data format is 8 bits. A dedicated 8-bit baud rate generator is used to derive baud rate frequencies from the oscillator. The USART transmits and receives the LSb first. The USART's transmitter and receiver are functionally independent but use the same data format and baud rate. The baud rate generator produces a clock either x16 or x64 of the bit shift rate, depending on bit BRGH (TXSTA<2>). Parity is not supported by the hardware, but can be implemented in software (and stored as the ninth data bit). Asynchronous mode is stopped during SLEEP.

Asynchronous mode is selected by clearing bit SYNC (TXSTA<4>).

The USART Asynchronous module consists of the following important elements:

- · Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- · Asynchronous Receiver

12.2.1 USART ASYNCHRONOUS TRANSMITTER

The USART transmitter block diagram is shown in Figure 12-5. The heart of the transmitter is the transmitt (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the STOP bit has been transmitted from the previous load. As soon as the STOP bit is transmitted, the TSR is loaded with new data from the TXREG register (if available). Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and flag bit TXIF (PIR1<4>) is set. This 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 indicated the status of the TXREG register, another bit TRMT (TXSTA<1>) shows the status of the TSR register. STATUS bit TRMT is a read only bit which is set when the TSR register 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.

- **Note 1:** The TSR register is not mapped in data memory so it is not available to the user.
 - 2: Flag bit TXIF is set when enable bit TXEN is set.

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 and the baud rate generator (BRG) has produced a shift clock (Figure 12-5). The transmission can also be started by first loading the TXREG register and then setting enable bit TXEN. 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. A back-to-back transfer is thus possible (Figure 12-7). Clearing enable bit TXEN during a transmission will cause the transmission to be aborted and will RESET the transmitter. As a result the RB2/TX/CK pin will revert to hi-impedance.

In order to select 9-bit transmission, transmit bit TX9 (TXSTA<6>) should be set and the ninth bit should be written to 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 register can result in an immediate transfer of the data to the TSR register (if the TSR is empty). In such a case, an incorrect ninth data bit may be loaded in the TSR register.

12.3 USART Function

The USART function is similar to that on the PIC16C74B, which includes the BRGH = 1 fix.

12.3.1 USART 9-BIT RECEIVER WITH ADDRESS DETECT

When the RX9 bit is set in the RCSTA register, 9 bits are received and the ninth bit is placed in the RX9D bit of the RCSTA register. The USART module has a special provision for multiprocessor communication. Multiprocessor communication is enabled by setting the ADEN bit (RCSTA<3>) along with the RX9 bit. The port is now programmed so when the last bit is received, the contents of the Receive Shift Register (RSR) are transferred to the receive buffer. The ninth bit of the RSR (RSR<8>) is transferred to RX9D, and the receive interrupt is set if, and only, if RSR<8> = 1. This feature can be used in a multiprocessor system as follows:

A master processor intends to transmit a block of data to one of many slaves. It must first send out an address byte that identifies the target slave. An address byte is identified by setting the ninth bit (RSR<8>) to a '1' (instead of a '0' for a data byte). If the ADEN and RX9 bits are set in the slave's RCSTA register, enabling multiprocessor communication, all data bytes will be ignored. However, if the ninth received bit is equal to a '1', indicating that the received byte is an address, the slave will be interrupted and the contents of the RSR register will be transferred into the receive buffer. This allows the slave to be interrupted only by addresses, so that the slave can examine the received byte to see if it is being addressed. The addressed slave will then clear its ADEN bit and prepare to receive data bytes from the master.

When ADEN is enabled (='1'), all data bytes are ignored. Following the STOP bit, the data will not be loaded into the receive buffer, and no interrupt will occur. If another byte is shifted into the RSR register, the previous data byte will be lost. The ADEN bit will only take effect when the receiver is configured in 9-bit mode (RX9 = '1'). When ADEN is disabled (='0'), all data bytes are received and the 9th bit can be used as the PARITY bit.

The USART Receive Block Diagram is shown in Figure 12-8.

Reception is enabled by setting bit CREN (RCSTA<4>).

12.3.1.1 Setting up 9-bit mode with Address Detect

Steps to follow when setting up an Asynchronous or Synchronous Reception with Address Detect Enabled:

- 1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH.
- Enable asynchronous or synchronous communication by setting or clearing bit SYNC and setting bit SPEN.
- 3. If interrupts are desired, then set enable bit RCIE.
- 4. Set bit RX9 to enable 9-bit reception.
- 5. Set ADEN to enable address detect.
- 6. Enable the reception by setting enable bit CREN or SREN.
- 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 8-bit received data by reading the RCREG register to determine if the device is being addressed.
- 9. If any error occurred, clear the error by clearing enable bit CREN if it was already set.
- If the device has been addressed (RSR<8> = 1 with address match enabled), clear the ADEN and RCIF bits to allow data bytes and address bytes to be read into the receive buffer and interrupt the CPU.

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 -00x	0000 -00x
1Ah	RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	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	e Genera	ator Regist	er					0000 0000	0000 0000

TABLE 12-8: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

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

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 12-5. 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 12-12). The transmission can also be started by first loading the TXREG register and then setting bit TXEN (Figure 12-13). 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 hiimpedance. If either bit CREN or bit SREN is set, during a transmission, the transmission is aborted and the DT pin reverts to a hi-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 Hi-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.

Steps to follow when setting up a Synchronous Master Transmission:

- 1. Initialize the SPBRG register for the appropriate baud rate (Section 12.1).
- 2. Enable the synchronous master serial port by setting bits SYNC, SPEN, and CSRC.
- 3. If interrupts are desired, then set enable bit TXIE.
- 4. If 9-bit transmission is desired, then set bit TX9.
- 5. Enable the transmission by setting bit TXEN.
- 6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
- 7. Start transmission by loading data to the TXREG register.

TABLE 13-1: REGISTERS/BITS ASSOCIATED WITH DATA EEPROM

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
9Ah	EEDATA	EEPROM	data regi	XXXX XXXX	uuuu uuuu						
9Bh	EEADR	EEPROM	address	register						XXXX XXXX	uuuu uuuu
9Ch	EECON1	—	— — — WRERR WREN WR RD							x000	q000
9Dh	EECON2 ⁽¹⁾	EEPROM	EEPROM control register 2								

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

Note 1: EECON2 is not a physical register

TABLE 14-7: INITIALIZATION CONDITION FOR SPECIAL REGISTERS

Condition	Program Counter	STATUS Register	PCON Register
Power-on Reset	000h	0001 1xxx	1-0x
MCLR Reset during normal operation	000h	000u uuuu	1-uu
MCLR Reset during SLEEP	000h	0001 0uuu	1-uu
WDT Reset	000h	0000 uuuu	1-uu
WDT Wake-up	PC + 1	uuu0 0uuu	u-uu
Brown-out Detect Reset	000h	000x xuuu	1-u0
Interrupt Wake-up from SLEEP	PC + 1 ⁽¹⁾	uuul 0uuu	u-uu

Legend: u = unchanged, x = unknown, - = unimplemented bit, reads as '0'.

Note 1: When the wake-up is due to an interrupt and global enable bit, GIE is set, the PC is loaded with the interrupt vector (0004h) after execution of PC+1.

Register	Address	Power-on Reset	 MCLR Reset during normal operation MCLR Reset during SLEEP WDT Reset Brown-out Detect Reset ⁽¹⁾ 	 Wake-up from SLEEP through interrupt Wake-up from SLEEP through WDT timeout
W		xxxx xxxx	<u>uuuu</u> uuuu	<u>uuuu</u> uuuu
INDF	00h	-	-	-
TMR0	01h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	02h	0000 0000	0000 0000	PC + 1 ⁽³⁾
STATUS	03h	0001 1xxx	000q quuu ⁽⁴⁾	uuuq quuu ⁽⁴⁾
FSR	04h	XXXX XXXX	սսսս սսսս	սսսս սսսս
PORTA	05h	xxxx 0000	xxxx u000	xxxx 0000
PORTB	06h	xxxx xxxx	սսսս սսսս	սսսս սսսս
T1CON	10h	00 0000	uu uuuu	uu uuuu
T2CON	12h	-000 0000	-000 0000	-uuu uuuu
CCP1CON	17h	00 0000	00 0000	uu uuuu
RCSTA	18h	0000 -00x	0000 -00x	uuuu -uuu
CMCON	1Fh	0000 0000	0000 0000	uu uuuu
PCLATH	0Ah	0 0000	0 0000	u uuuu
INTCON	0Bh	0000 000x	0000 000u	uuuu uqqq ⁽²⁾
PIR1	0Ch	0000 -000	0000 -000	-q (2,5)
OPTION	81h	1111 1111	1111 1111	uuuu uuuu
TRISA	85h	11-1 1111	11 1111	uu-u uuuu
TRISB	86h	1111 1111	1111 1111	uuuu uuuu
PIE1	8Ch	0000 -000	0000 -000	uuuu -uuu
PCON	8Eh	1-0x	1-uq ^(1,6)	
TXSTA	98h	0000 -010	0000 -010	นนนน -นนน
EECON1	9Ch	x000	q000	uuuu
VRCON	9Fh	000- 0000	000- 0000	นนน- นนนน

 TABLE 14-8:
 INITIALIZATION CONDITION FOR REGISTERS

Legend: u = unchanged, x = unknown, - = unimplemented bit, reads as '0', q = value depends on condition.

Note 1: If VDD goes too low, Power-on Reset will be activated and registers will be affected differently.

2: One or more bits in INTCON, PIR1 and/or PIR2 will be affected (to cause wake-up).

3: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

4: See Table 14-7 for RESET value for specific condition.

5: If wake-up was due to comparator input changing, then Bit 6 = 1. All other interrupts generating a wake-up will cause Bit 6 = u.

6: If RESET was due to brown-out, then Bit 0 = 0. All other RESETS will cause Bit 0 = u.





FIGURE 14-9: TIMEOUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2



FIGURE 14-10: TIMEOUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD)



© 2003 Microchip Technology Inc.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR Reset	Value on all other RESETS ⁽¹⁾
0Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	EEIF	CMIF	RCIF	TXIF	_	CCP1IF	TMR2IF	TMR1IF	0000 -000	0000 -000
8Ch	PIE1	EEIE	CMIE	RCIE	TXIE	_	CCP1IE	TMR2IE	TMR1IE	0000 -000	0000 -000

TABLE 14-9:SUMMARY OF INTERRUPT REGISTERS

Note 1: Other (non Power-up) Resets include MCLR Reset, Brown-out Detect Reset and Watchdog Timer Reset during normal operation.

14.7 Context Saving During Interrupts

During an interrupt, only the return PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt (e.g., W register and STATUS register). This will have to be implemented in software.

Example 14-2 stores and restores the STATUS and W registers. The user register, W_TEMP, must be defined in a common memory location (i.e., W_TEMP is defined at 0x70 in Bank 0 and is therefore, accessible at 0xF0, 0x17 and 0xIFD). The Example 14-2:

- Stores the W register
- Stores the STATUS register
- Executes the ISR code
- Restores the STATUS (and bank select bit register)
- Restores the W register

EXAMPLE 14-2: SAVING THE STATUS AND W REGISTERS IN RAM

MOVWF	W_TEMP	;copy W to temp register, could be in either bank
SWAPF	STATUS,W	;swap status to be saved into W
BCF	STATUS, RPO	;change to bank 0 regardless of current bank
MOVWF	STATUS_TEMP	;save status to bank 0 register
:		
:	(ISR)	
:		
SWAPF	STATUS_TEMP,W	;swap STATUS_TEMP register into W, sets bank to origi- nal state
MOVWF	STATUS	;move W into STATUS register
SWAPF	W_TEMP,F	;swap W_TEMP
SWAPF	W_TEMP,W	;swap W_TEMP into W

14.8 Watchdog Timer (WDT)

The Watchdog Timer is a free running on-chip RC oscillator which does not require any external components. This RC oscillator is separate from the ER oscillator of the CLKIN pin. That means that the WDT will run, even if the clock on the OSC1 and OSC2 pins of the device has been stopped, for example, by execution of a SLEEP instruction. During normal operation, a WDT timeout generates a device RESET. If the device is in SLEEP mode, a WDT timeout causes the device to wake-up and continue with normal operation. The WDT can be permanently disabled by programming the configuration bit WDTE as clear (Section 14.1).

14.8.1 WDT PERIOD

The WDT has a nominal timeout period of 18 ms (with no prescaler). The timeout periods vary with temperature, VDD and process variations from part to part (see DC specs). If longer timeout periods are desired, a postscaler with a division ratio of up to 1:128 can be assigned to the WDT under software control by writing to the OPTION register. Thus, timeout periods up to 2.3 seconds can be realized.

The CLRWDT and SLEEP instructions clear the WDT and the postscaler, if assigned to the WDT, and prevent it from timing out and generating a device RESET.

The $\overline{\text{TO}}$ bit in the STATUS register will be cleared upon a Watchdog Timer timeout.

14.8.2 WDT PROGRAMMING CONSIDERATIONS

It should also be taken in account that under worst case conditions (VDD = Min., Temperature = Max., max. WDT prescaler), it may take several seconds before a WDT timeout occurs.

14.12 In-Circuit Serial Programming

The PIC16F62X microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data, and three other lines for power, ground, and the programming voltage. This allows customers to manufacture boards with unprogrammed devices, and then program the microcontroller just before shipping the product. This also allows the most recent firmware, or a custom firmware to be programmed.

The device is placed into a Program/Verify mode by holding the RB6 and RB7 pins low while raising the $\overline{\text{MCLR}}$ (VPP) pin from VIL to VIHH (see programming specification). RB6 becomes the programming clock and RB7 becomes the programming data. Both RB6 and RB7 are Schmitt Trigger inputs in this mode.

After RESET, to place the device into Programming/ Verify mode, the program counter (PC) is at location 00h. A 6-bit command is then supplied to the device. Depending on the command, 14 bits of program data are then supplied to or from the device, depending if the command was a load or a read. For complete details of serial programming, please refer to the Programming Specifications.

A typical in-circuit serial programming connection is shown in Figure 14-18.

FIGURE 14-18: TYPICAL IN-CIRCUIT SERIAL PROGRAMMING CONNECTION



14.13 Low Voltage Programming

The LVP bit of the configuration word, enables the low voltage programming. This mode allows the microcontroller to be programmed via ICSP using only a 5V source. This mode removes the requirement of VIHH to be placed on the MCLR pin. The LVP bit is normally erased to '1', which enables the low voltage programming. In this mode, the RB4/PGM pin is dedicated to the programming function and ceases to be a general purpose I/O pin. The device will enter Programming mode when a '1' is placed on the RB4/PGM pin. The HV Programming mode is still available by placing VIHH on the MCLR pin.

- Note 1: While in this mode, the RB4 pin can no longer be used as a general purpose I/O pin.
 - 2: VDD must be 5.0V <u>+</u>10% during erase/ program operations while in low voltage Programming mode.

If Low voltage Programming mode is not used, the LVP bit can be programmed to a '0', and RB4/PGM becomes a digital I/O pin. To program the device, VIHH must be placed onto MCLR during programming. The LVP bit may only be programmed when programming is entered with VIHH on MCLR. The LVP bit cannot be programmed when programming is entered with RB4/PGM.

It should be noted, that once the LVP bit is programmed to 0, High voltage Programming mode can be used to program the device.

RETLW	Return with Literal in W	RLF	Rotate Left f through Carry
Syntax:	[<i>label</i>] RETLW k	Syntax:	[<i>label</i>] RLF f,d
Operands:	$0 \leq k \leq 255$	Operands:	$0 \leq f \leq 127$
Operation:	$k \rightarrow (W);$ TOS \rightarrow PC	Operation:	d ∈ [0,1] See description below
Status Affected:	None	Status Affected:	c .
Encoding:	11 01xx kkkk kkkk	Encoding:	00 1101 dfff ffff
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.	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:	2	Words	1
Example	CALL TABLE;W contains table	Cycles:	1
	 ;offset value ;W now has table 	Eycles. Example	RLF REGIO
TABLE	<pre>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</pre>		Before Instruction REG1 = 1110 0110 C = 0 After Instruction REG1 = 1110 0110 W = 1100 1100 C = 1
RETURN	Return from Subroutine		
Syntax:	[label] RETURN		
Operands:	None		
Operation:	$TOS\toPC$		
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		

16.9 MPLAB ICE 2000 High Performance Universal In-Circuit Emulator

The MPLAB ICE 2000 universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PICmicro microcontrollers. Software control of the MPLAB ICE 2000 in-circuit emulator is advanced by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the MPLAB ICE in-circuit emulator allows expansion to support new PICmicro microcontrollers.

The MPLAB ICE 2000 in-circuit emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft[®] Windows 32-bit operating system were chosen to best make these features available in a simple, unified application.

16.10 MPLAB ICE 4000 High Performance Universal In-Circuit Emulator

The MPLAB ICE 4000 universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for highend PICmicro microcontrollers. Software control of the MPLAB ICE in-circuit emulator is provided by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICD 4000 is a premium emulator system, providing the features of MPLAB ICE 2000, but with increased emulation memory and high speed performance for dsPIC30F and PIC18XXXX devices. Its advanced emulator features include complex triggering and timing, up to 2 Mb of emulation memory, and the ability to view variables in real-time.

The MPLAB ICE 4000 in-circuit emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft Windows 32-bit operating system were chosen to best make these features available in a simple, unified application.

16.11 MPLAB ICD 2 In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD 2, is a powerful, low cost, run-time development tool, connecting to the host PC via an RS-232 or high speed USB interface. This tool is based on the FLASH PICmicro MCUs and can be used to develop for these and other PICmicro microcontrollers. The MPLAB ICD 2 utilizes the in-circuit debugging capability built into the FLASH devices. This feature, along with Microchip's In-Circuit Serial Programming[™] (ICSP[™]) protocol, offers cost effective in-circuit FLASH debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by setting breakpoints, single-stepping and watching variables, CPU status and peripheral registers. Running at full speed enables testing hardware and applications in real-time. MPLAB ICD 2 also serves as a development programmer for selected PICmicro devices.

16.12 PRO MATE II Universal Device Programmer

The PRO MATE II is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX for maximum reliability. It features an LCD display for instructions and error messages and a modular detachable socket assembly to support various package types. In Stand-Alone mode, the PRO MATE II device programmer can read, verify, and program PICmicro devices without a PC connection. It can also set code protection in this mode.

16.13 PICSTART Plus Development Programmer

The PICSTART Plus development programmer is an easy-to-use, low cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient. The PICSTART Plus development programmer supports most PICmicro devices up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus development programmer is CE compliant.

17.0 ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings†

Ambient temperature under bias	40 to +125°C
Storage temperature	65°C to +150°C
Voltage on VDD with respect to VSS	-0.3 to +6.5V
Voltage on MCLR and RA4 with respect to Vss	0.3 to +14V
Voltage on all other pins with respect to Vss	-0.3V to VDD + 0.3V
Total power dissipation ⁽¹⁾	
Maximum current out of Vss pin	
Maximum current into VDD pin	250 mA
Input clamp current, Iık (VI < 0 or VI > VDD)	±20 mA
Output clamp current, loк (Vo < 0 or Vo >VDD)	±20 mA
Maximum output current sunk by any I/O pin	25 mA
Maximum output current sourced by any I/O pin	
Maximum current sunk by PORTA and PORTB	
Maximum current sourced by PORTA and PORTB	
Note 1: Power dissipation is calculated as follows: PDIS = VDD x {IDD - \sum IOH} + \sum {(VDD-	VOH) x IOH} + Σ (VOI x IOL)

† 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 Vss at the \overline{MCLR} pin, inducing currents greater than 80 mA, may cause latchup. 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 Vss

DC Characteristics		Standard Operating Conditions (unless otherwise stated)					
Parameter No.	Sym	Characteristic	Min	Тур†	Max	Units	Conditions
		Data EEPROM Memory					
D120	ED	Endurance	1M*	10M		E/W	25°C at 5V
D121	Vdrw	VDD for read/write	VMIN	—	5.5	V	VMIN = Minimum operating
							voltage
D122	TDEW	Erase/Write cycle time	—	4	8*	ms	
		Program FLASH Memory	•		•	•	•
D130	Eр	Endurance	1000*	10000	—	E/W	
D131	Vpr	VDD for read	Vmin	—	5.5	V	VMIN = Minimum operating
							voltage
D132	VPEW	VDD for erase/write	4.5	—	5.5	V	
D133	TPEW	Erase/Write cycle time	—	4	8*	ms	

TABLE 17-3: DC CHARACTERISTICS: PIC16F62X, PIC16LF62X

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

17.4 Timing Diagrams and Specifications

FIGURE 17-6: EXTERNAL CLOCK TIMING



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					
Fron	ו: Name					
	Company					
	Address					
	City / State / ZIP / Country					
	Telephone: ()	FAX: ()				
Appl	Application (optional):					
Wou	Would you like a reply?YN					
Device: PIC16F62X Literature Number: DS40300C						
Que	Questions:					
1 \	What are the best features of this do	cument?				
-						
2.	2. How does this document meet your hardware and software development needs?					
-						
-						
3. I	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 \	Nhat deletions from the document or	ould be made without affecting the overall usefulness?				
5.		Sud be made without anecting the overall usefulness?				
-						
6. I	s there any incorrect or misleading in	nformation (what and where)?				
	. 0					
-						
7. I	How would you improve this docume	nt?				
-						