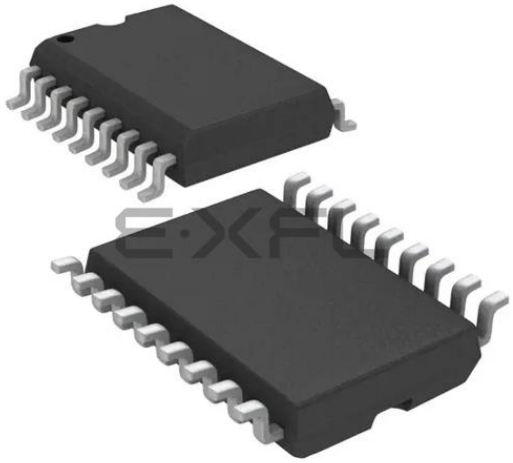


Welcome to E-XFL.COM

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	4MHz
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)	2V ~ 5.5V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	18-SOIC (0.295", 7.50mm Width)
Supplier Device Package	18-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lf627-04-so

Note the following details of the code protection feature on Microchip devices:

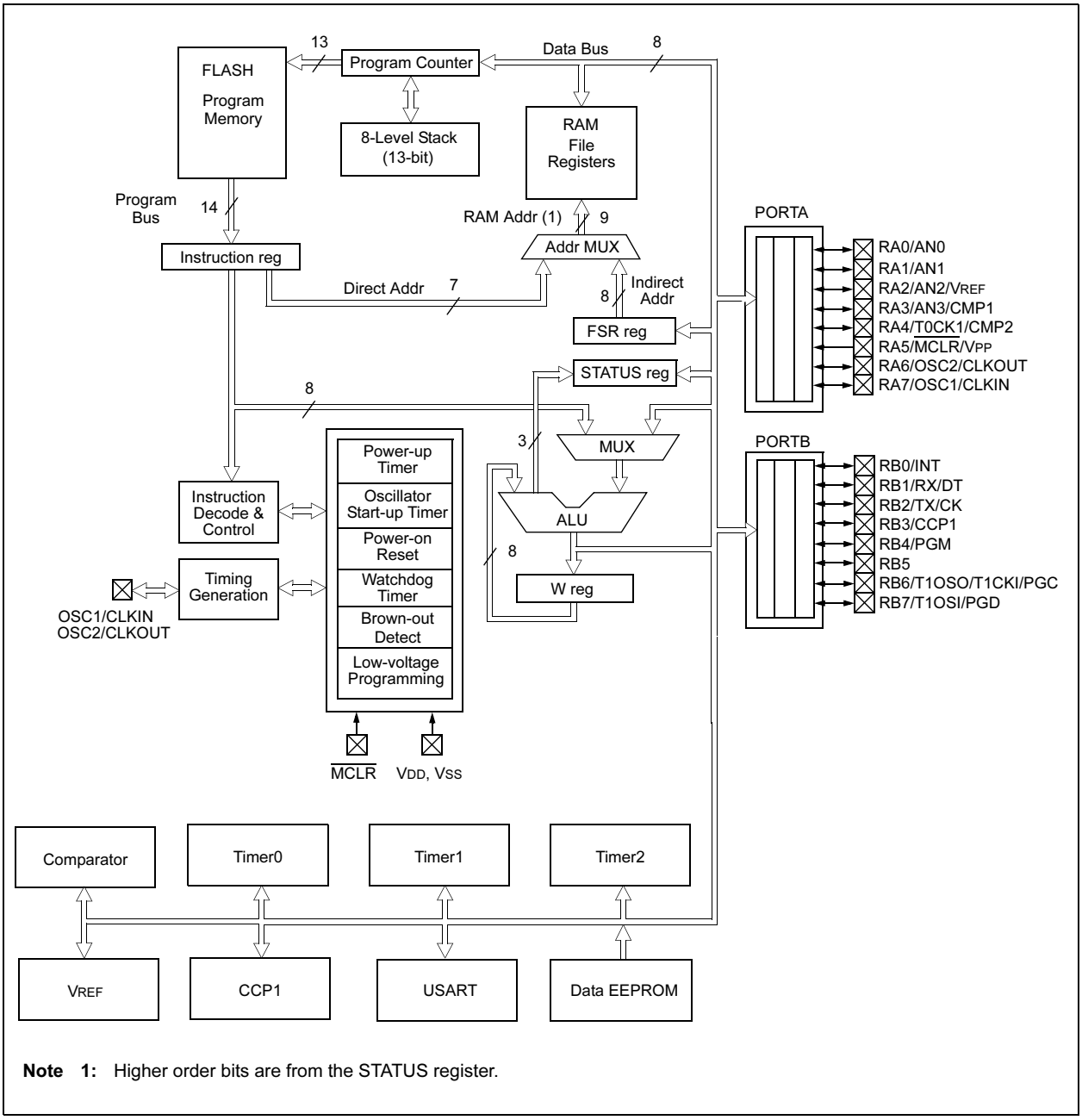
- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products.

DNV Certification, Inc. USA — ANSI-RAB QMS ACCREDITED — DNV MSC The Netherlands Accredited by the RvA — DNV — ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.

FIGURE 2-1: BLOCK DIAGRAM



Note 1: Higher order bits are from the STATUS register.

TABLE 2-1: PIC16F62X PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
RA0/AN0	RA0	ST	CMOS	Bi-directional I/O port
	AN0	AN	—	Analog comparator input
RA1/AN1	RA1	ST	CMOS	Bi-directional I/O port
	AN1	AN	—	Analog comparator input
RA2/AN2/VREF	RA2	ST	CMOS	Bi-directional I/O port
	AN2	AN	—	Analog comparator input
	VREF	—	AN	VREF output
RA3/AN3/CMP1	RA3	ST	CMOS	Bi-directional I/O port
	AN3	AN	—	Analog comparator input
	CMP1	—	CMOS	Comparator 1 output
RA4/T0CKI/CMP2	RA4	ST	OD	Bi-directional 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
	VPP	—	—	Programming voltage input. 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.
RA6/OSC2/CLKOUT	RA6	ST	CMOS	Bi-directional I/O port
	OSC2	XTAL	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
	CLKOUT	—	CMOS	In ER/INTRC mode, OSC2 pin can output CLKOUT, which has 1/4 the frequency of OSC1
RA7/OSC1/CLKIN	RA7	ST	CMOS	Bi-directional I/O port
	OSC1	XTAL	—	Oscillator crystal input
	CLKIN	ST	—	External clock source input. ER biasing pin.
RB0/INT	RB0	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	INT	ST	—	External interrupt.
RB1/RX/DT	RB1	TTL	CMOS	Bi-directional 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	Bi-directional I/O port.
	TX	—	CMOS	USART transmit pin
	CK	ST	CMOS	Synchronous clock I/O. Can be software programmed for internal weak pull-up.
RB3/CCP1	RB3	TTL	CMOS	Bi-directional 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

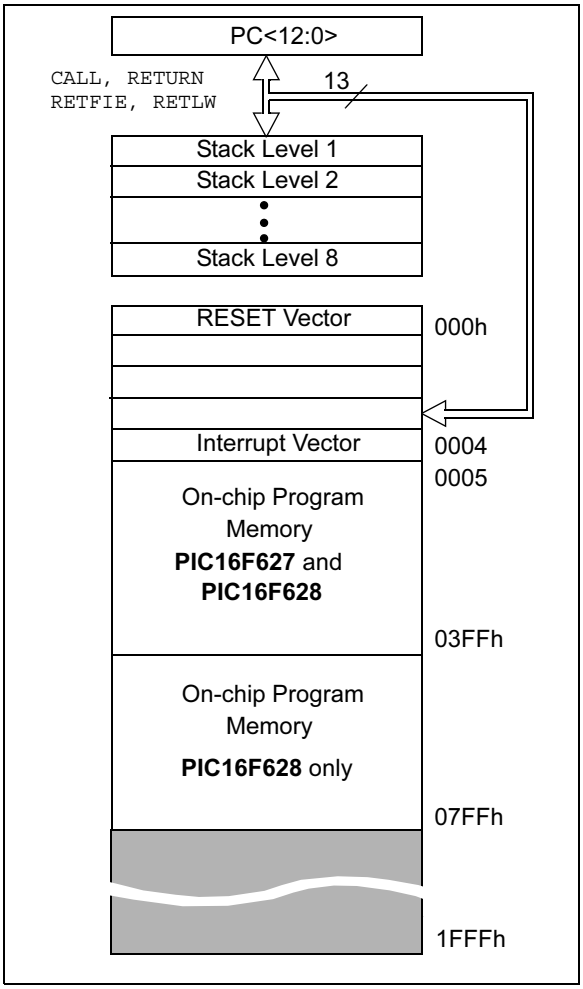
NOTES:

3.0 MEMORY ORGANIZATION

3.1 Program Memory Organization

The PIC16F62X 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 PIC16F627 and 2K x 14 (0000h - 07FFh) for the PIC16F628 are physically implemented. Accessing a location above these boundaries will cause a wrap-around within the first 1K x 14 space (PIC16F627) or 2K x 14 space (PIC16F628). The RESET vector is at 0000h and the interrupt vector is at 0004h (Figure 3-1).

FIGURE 3-1: PROGRAM MEMORY MAP AND STACK



3.2 Data Memory Organization

The data memory (Figure 3-2) is partitioned into four banks, which contain the general purpose registers and the Special Function Registers (SFR). The SFR's are located in the first 32 locations of each Bank. Register locations 20-7Fh, A0h-FFh, 120h-14Fh, 170h-17Fh and 1F0h-1FFh are general purpose registers implemented as static RAM.

The Table below lists how to access the four banks of registers:

	RP1	RP0
Bank0	0	0
Bank1	0	1
Bank2	1	0
Bank3	1	1

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

3.2.1 GENERAL PURPOSE REGISTER FILE

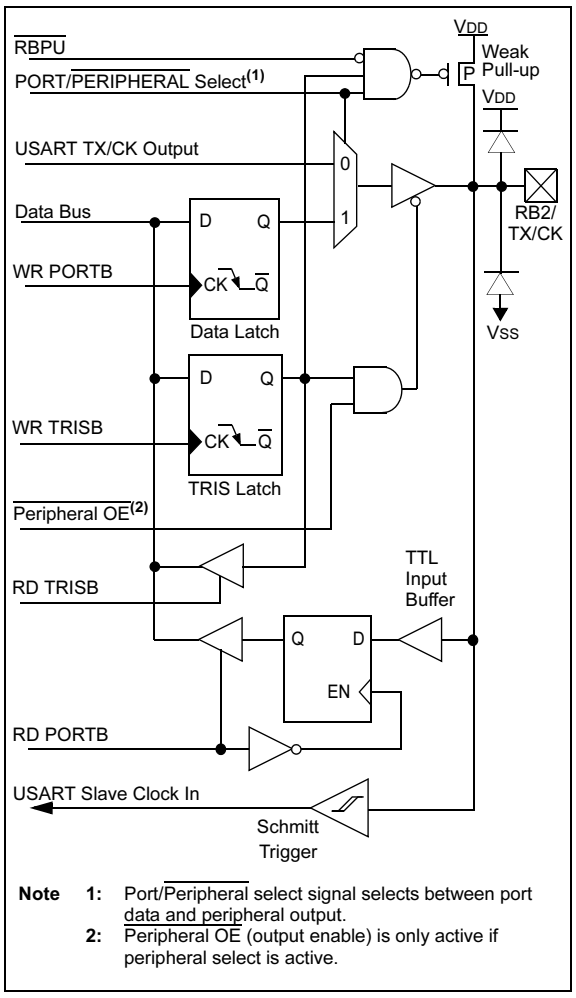
The register file is organized as 224 x 8 in the PIC16F62X. Each is accessed either directly or indirectly through the File Select Register FSR (See Section 3.4).

TABLE 5-1: PORTA FUNCTIONS

Name	Function	Input Type	Output Type	Description
RA0/AN0	RA0	ST	CMOS	Bi-directional I/O port
	AN0	AN	—	Analog comparator input
RA1/AN1	RA1	ST	CMOS	Bi-directional I/O port
	AN1	AN	—	Analog comparator input
RA2/AN2/VREF	RA2	ST	CMOS	Bi-directional I/O port
	AN2	AN	—	Analog comparator input
	VREF	—	AN	VREF output
RA3/AN3/CMP1	RA3	ST	CMOS	Bi-directional I/O port
	AN3	AN	—	Analog comparator input
	CMP1	—	CMOS	Comparator 1 output
RA4/T0CKI/CMP2	RA4	ST	OD	Bi-directional I/O port
	T0CKI	ST	—	External clock input for TMR0 or comparator output. Output is open drain type
	CMP2	—	OD	Comparator 2 output
RA5/$\overline{\text{MCLR}}$/VPP	RA5	ST	—	Input port
	$\overline{\text{MCLR}}$	ST	—	Master clear
	VPP	HV	—	Programming voltage input. 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
RA6/OSC2/CLKOUT	RA6	ST	CMOS	Bi-directional I/O port.
	OSC2	XTAL	—	Oscillator crystal output. Connects to crystal resonator in Crystal Oscillator mode.
	CLKOUT	—	CMOS	In ER/INTRC mode, OSC2 pin can output CLKOUT, which has 1/4 the frequency of OSC1
RA7/OSC1/CLKIN	RA7	ST	CMOS	Bi-directional I/O port
	OSC1	XTAL	—	Oscillator crystal input
	CLKIN	ST	—	External clock source input. ER biasing pin.

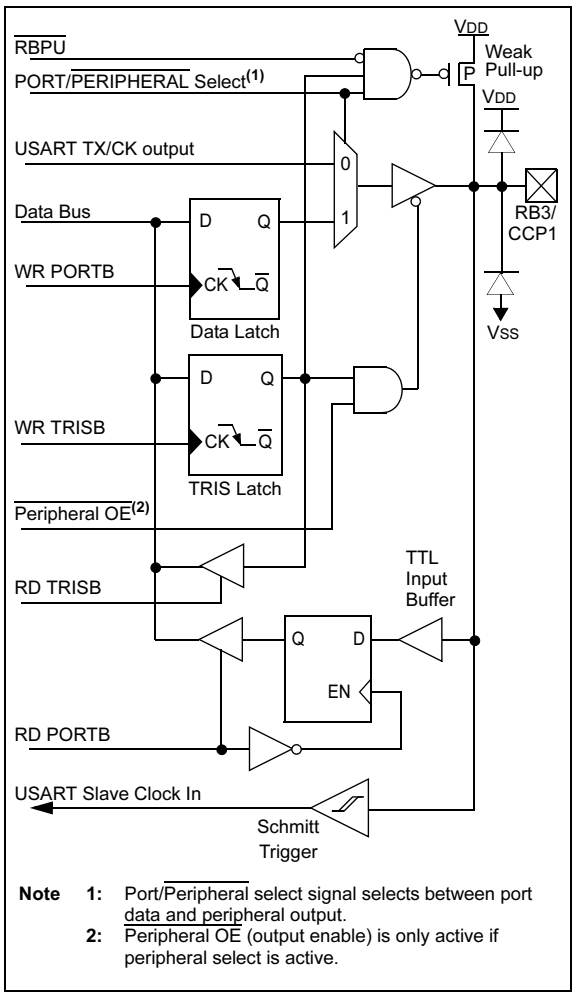
Legend: ST = Schmitt Trigger input HV = High Voltage OD = Open Drain AN = Analog

FIGURE 5-10: BLOCK DIAGRAM OF RB2/TX/CK PIN



Note 1: Port/Peripheral select signal selects between port data and peripheral output.
2: Peripheral OE (output enable) is only active if peripheral select is active.

FIGURE 5-11: BLOCK DIAGRAM OF RB3/CCP1 PIN



Note 1: Port/Peripheral select signal selects between port data and peripheral output.
2: Peripheral OE (output enable) is only active if peripheral select is active.

5.3 I/O Programming Considerations

5.3.1 BI-DIRECTIONAL I/O PORTS

Any instruction which 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 bi-directional I/O pin (e.g., Bit 0) and it 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.

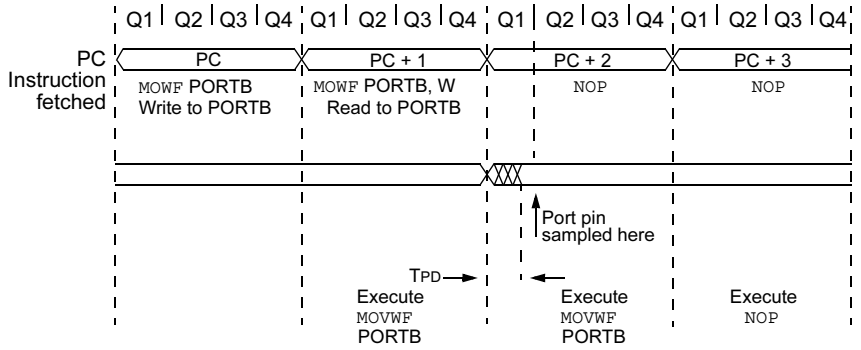
EXAMPLE 5-2: READ-MODIFY-WRITE INSTRUCTIONS ON AN I/O PORT

```
;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.

6.3.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control (i.e., it can be changed “on the fly” during program execution). Use the instruction sequences, shown in Example 6-1, when changing the prescaler assignment from Timer0 to WDT, to avoid an unintended device RESET.

EXAMPLE 6-1: CHANGING PRESCALER (TIMER0→WDT)

```
BCF     STATUS, RP0   ;Skip if already in
                      ;Bank 0
CLRWDT                ;Clear WDT
CLRF    TMR0          ;Clear TMR0 & Prescaler
BSF     STATUS, RP0   ;Bank 1
MOVLW   '00101111'b   ;These 3 lines
                      ;(5, 6, 7)
MOVWF   OPTION_REG    ;are required only
                      ;if desired PS<2:0>
                      ;are
CLRWDT                ;000 or 001
MOVLW   '00101xxx'b   ;Set Postscaler to
MOVWF   OPTION_REG    ;desired WDT rate
BCF     STATUS, RP0   ;Return to Bank 0
```

To change prescaler from the WDT to the TMR0 module use the sequence shown in Example 6-2. This precaution must be taken even if the WDT is disabled.

EXAMPLE 6-2: CHANGING PRESCALER (WDT→TIMER0)

```
CLRWDT                ;Clear WDT and
                      ;prescaler
BSF     STATUS, RP0
MOVLW   b'xxxx0xxx'   ;Select TMR0, new
                      ;prescale value and
                      ;clock source
MOVWF   OPTION_REG
BCF     STATUS, RP0
```

TABLE 6-1: REGISTERS ASSOCIATED WITH TIMER0

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
01h	TMR0	Timer0 module register								xxxx xxxx	uuuu uuuu
0Bh/8Bh/ 10Bh/18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
81h, 181h	OPTION(2)	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 1111	1111 1111

Legend: — = Unimplemented locations, read as ‘0’, u = unchanged, x = unknown

Note 1: Shaded bits are not used by TMR0 module.

2: Option is referred by OPTION_REG in MPLAB.

REGISTER 8-1: T2CON: TIMER CONTROL REGISTER (ADDRESS: 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits

0000 = 1:1 Postscale Value
0001 = 1:2 Postscale Value
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit

1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits

00 = 1:1 Prescaler Value
01 = 1:4 Prescaler Value
1x = 1:16 Prescaler Value

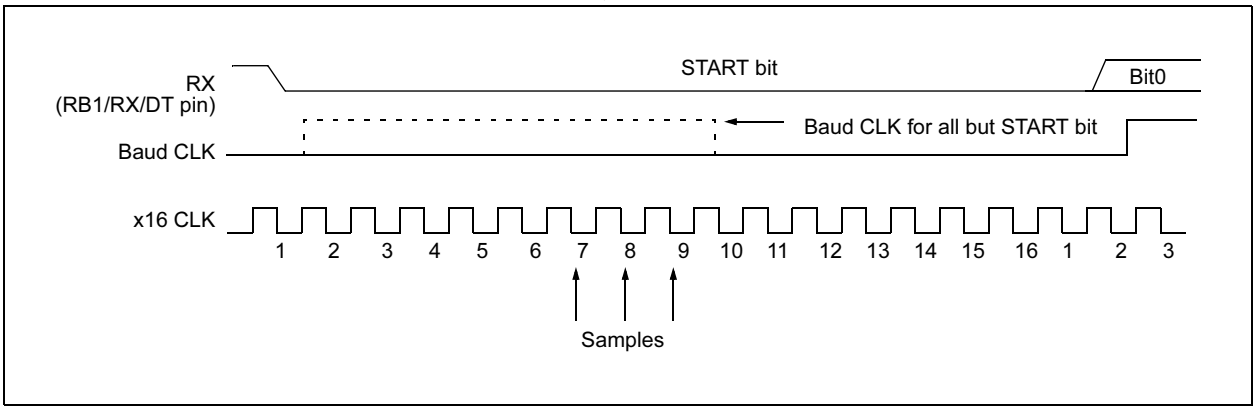
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

TABLE 8-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

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
0Bh/8Bh/ 10Bh/18Bh	INTCON	GIE	PEIE	T0IE	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
11h	TMR2	Timer2 module's register								0000 0000	0000 0000
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
92h	PR2	Timer2 Period Register								1111 1111	1111 1111

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

FIGURE 12-4: RX PIN SAMPLING SCHEME, BRGH = 0 OR BRGH = 1



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

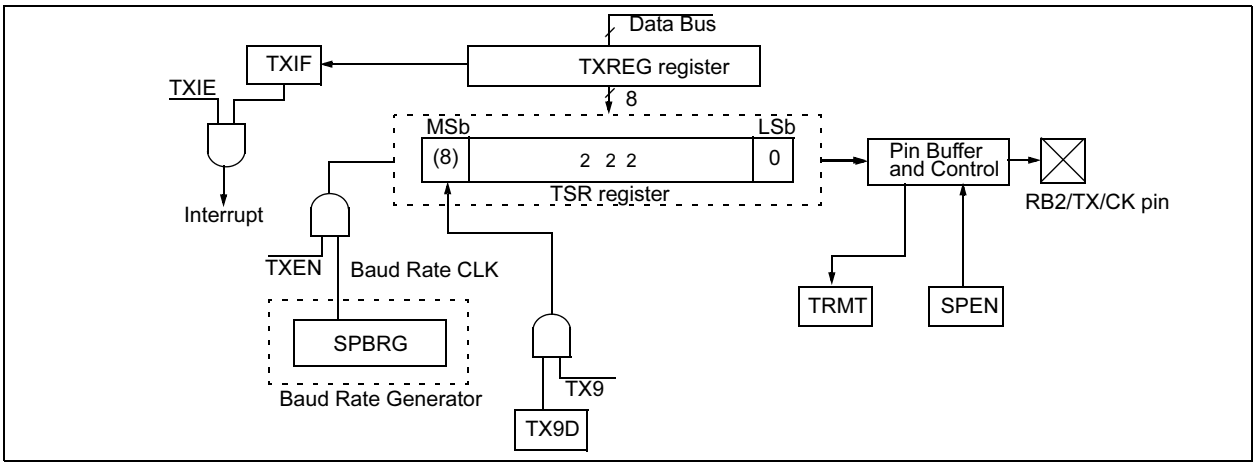
The USART transmitter block diagram is shown in Figure 12-5. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until 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.

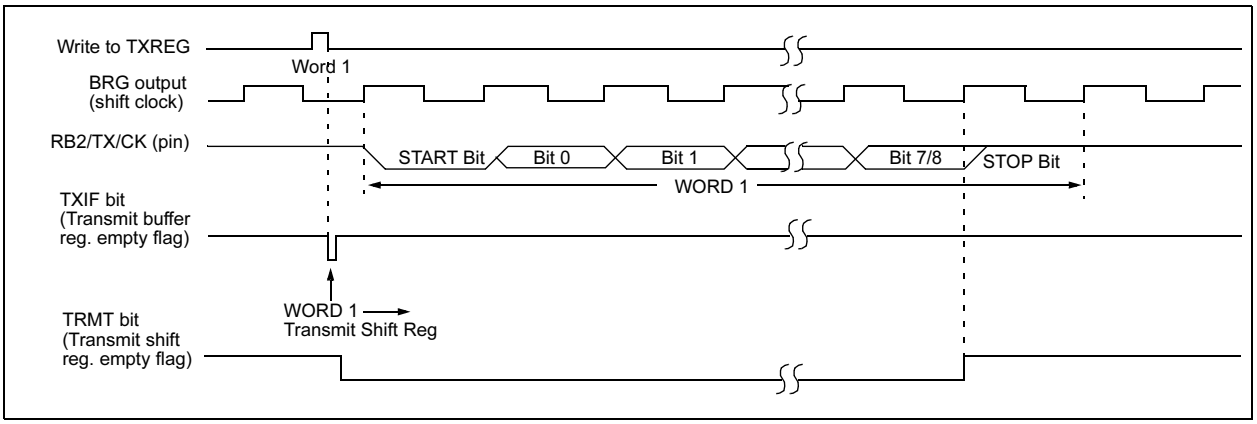
FIGURE 12-5: USART TRANSMIT BLOCK DIAGRAM



Steps to follow when setting up an Asynchronous Transmission:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH. (Section 12.1)
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, then set enable bit TXIE.
4. If 9-bit transmission is desired, then set transmit bit TX9.
5. Enable the transmission by setting bit TXEN, which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).

FIGURE 12-6: ASYNCHRONOUS TRANSMISSION



14.2 Oscillator Configurations

14.2.1 OSCILLATOR TYPES

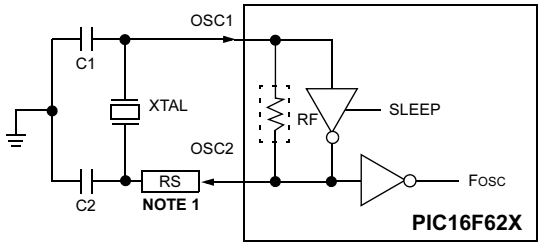
The PIC16F62X can be operated in eight different oscillator options. The user can program three configuration bits (FOSC2 thru FOSC0) to select one of these eight modes:

- LP Low Power Crystal
- XT Crystal/Resonator
- HS High Speed Crystal/Resonator
- ER External Resistor (2 modes)
- INTRC Internal Resistor/Capacitor (2 modes)
- EC External Clock In

14.2.2 CRYSTAL OSCILLATOR / CERAMIC RESONATORS

In XT, LP or HS modes a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation (Figure 14-1). The PIC16F62X oscillator design requires the use of a parallel cut crystal. Use of a series cut crystal may give a frequency out of the crystal manufacturers specifications. When in XT, LP or HS modes, the device can have an external clock source to drive the OSC1 pin (Figure 14-4).

FIGURE 14-1: CRYSTAL OPERATION (OR CERAMIC RESONATOR) (HS, XT OR LP OSC CONFIGURATION)



Note 1: A series resistor may be required for some crystals.
2: See Table 14-1 and Table 14-2 for recommended values of C1 and C2.

TABLE 14-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS

Ranges Characterized:			
Mode	**Freq**	**OSC1(C1)**	**OSC2(C2)**
XT	455 kHz 2.0 MHz 4.0 MHz	22 - 100 pF 15 - 68 pF 15 - 68 pF	22 - 100 pF 15 - 68 pF 15 - 68 pF
HS	8.0 MHz 16.0 MHz	10 - 68 pF 10 - 22 pF	10 - 68 pF 10 - 22 pF

Note 1: Higher capacitance increases the stability of the oscillator but also increases the start-up time. These values are for design guidance only. Since each resonator has its own characteristics, the user should consult the resonator manufacturer for appropriate values of external components.

TABLE 14-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Mode	Freq	OSC1(C1)	OSC2(C2)
LP	32 kHz 200 kHz	68 - 100 pF 15 - 30 pF	68 - 100 pF 15 - 30 pF
XT	100 kHz 2 MHz 4 MHz	68 - 150 pF 15 - 30 pF 15 - 30 pF	150 - 200 pF 15 - 30 pF 15 - 30 pF
HS	8 MHz 10 MHz 20 MHz	15 - 30 pF 15 - 30 pF 15 - 30 pF	15 - 30 pF 15 - 30 pF 15 - 30 pF

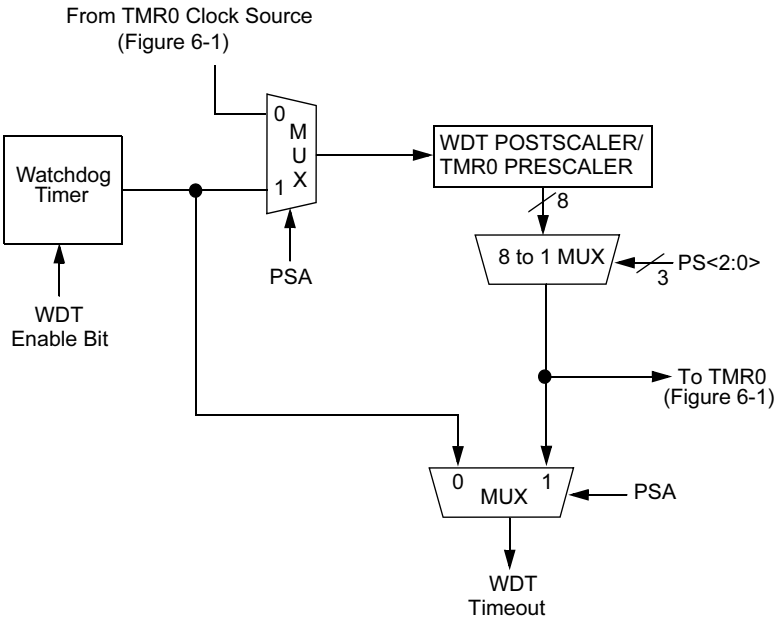
Note 1: Higher capacitance increases the stability of the oscillator but also increases the start-up time. These values are for design guidance only. Rs may be required in HS mode as well as XT mode to avoid overdriving crystals with low drive level specification. Since each crystal has its own characteristics, the user should consult the crystal manufacturer for appropriate values of external components.

14.2.3 EXTERNAL CRYSTAL OSCILLATOR CIRCUIT

Either a prepackaged oscillator can be used, or a simple oscillator circuit with TTL gates can be built. Prepackaged oscillators provide a wide operating range and better stability. A well-designed crystal oscillator will provide good performance with TTL gates. Two types of crystal oscillator circuits can be used; one with series resonance, or one with parallel resonance.

Figure 14-2 shows implementation of a parallel resonant oscillator circuit. The circuit is designed to use the fundamental frequency of the crystal. The 74AS04 inverter performs the 180° phase shift that a parallel oscillator requires. The 4.7 kΩ resistor provides the negative feedback for stability. The 10 kΩ potentiometers bias the 74AS04 in the linear region. This could be used for external oscillator designs.

FIGURE 14-16: WATCHDOG TIMER BLOCK DIAGRAM



Note 1: T0SE, T0CS, PSA, PS0-PS2 are bits in the OPTION register.

TABLE 14-10: SUMMARY OF WATCHDOG TIMER REGISTERS

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
2007h	Config. bits	LVP	BODEN	MCLRE	FOSC2	$\overline{\text{PWRTE}}$	WDTE	FOSC1	FOSC0	uuuu uuuu	uuuu uuuu
81h	OPTION	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: – = Unimplemented location, read as "0", + = Reserved for future use
Note 1: Shaded cells are not used by the Watchdog Timer.

14.9 Power-Down Mode (SLEEP)

The Power-down mode is entered by executing a SLEEP instruction.

If enabled, the Watchdog Timer will be cleared but keeps running, the $\overline{\text{PD}}$ bit in the STATUS register is cleared, the $\overline{\text{TO}}$ bit is set, and the oscillator driver is turned off. The I/O ports maintain the status they had, before SLEEP was executed (driving high, low, or hi-impedance).

For lowest current consumption in this mode, all I/O pins should be either at VDD, or VSS, with no external circuitry drawing current from the I/O pin and the comparators, and VREF should be disabled. I/O pins that are hi-impedance inputs should be pulled high or low externally to avoid switching currents caused by floating inputs. The T0CKI input should also be at VDD or VSS for lowest current consumption. The contribution from on-chip pull-ups on PORTB should be considered.

The $\overline{\text{MCLR}}$ pin must be at a logic high level (VIHMC).

Note:	It should be noted that a RESET generated by a WDT timeout does not drive $\overline{\text{MCLR}}$ pin low.

MOVWF	Move W to f
Syntax:	[*label*] MOVWF f
Operands:	$0 \le f \le 127$
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:	[*label*] 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 PICmicro® products, do not use this instruction.**

NOP	No Operation
Syntax:	[*label*] 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:	[*label*] 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

SUBWF **Subtract W from f**

Syntax: [*label*] SUBWF f,d

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

Operation: (f) - (W) → (dest)

Status Affected: C, DC, Z

Encoding:

00	0010	dfff	ffff

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

Words: 1

Cycles: 1

Example 1:
```
SUBWF   REG1, 1
```
Before Instruction
REG1 = 3
W = 2
C = ?
After Instruction
REG1 = 1
W = 2
C = 1; result is positive
Z = DC = 1

Example 2: Before Instruction
REG1 = 2
W = 2
C = ?
After Instruction
REG1 = 0
W = 2
C = 1; result is zero
Z = DC = 1

Example 3: Before Instruction
REG1 = 1
W = 2
C = ?
After Instruction
REG1 = 0xFF
W = 2
C = 0; result is negative
Z = DC = 0

SWAPF **Swap Nibbles in f**

Syntax: [*label*] SWAPF f,d

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

Operation: (f<3:0>) → (dest<7:4>),
(f<7:4>) → (dest<3:0>)

Status Affected: None

Encoding:

00	1110	dfff	ffff

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0 the result is placed in W register. If 'd' is 1 the result is placed in register 'f'.

Words: 1

Cycles: 1

Example
```
SWAPF   REG1, 0
```
Before Instruction
REG1 = 0xA5
After Instruction
REG1 = 0xA5
W = 0x5A

TRIS **Load TRIS Register**

Syntax: [*label*] TRIS f

Operands: $5 \le f \le 7$

Operation: (W) → TRIS register f;

Status Affected: None

Encoding:

00	0000	0110	0fff

Description: The instruction is supported for code compatibility with the PIC16C5X products. Since TRIS registers are readable and writable, the user can directly address them.

Words: 1

Cycles: 1

Example

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

16.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI C compilers for Microchip's PIC17CXXX and PIC18CXXX family of microcontrollers. These compilers provide powerful integration capabilities, superior code optimization and ease of use not found with other compilers.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

16.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK object linker combines relocatable objects created by the MPASM assembler and the MPLAB C17 and MPLAB C18 C compilers. It can link relocatable objects from pre-compiled libraries, using directives from a linker script.

The MPLIB object librarian manages the creation and modification of library files of pre-compiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

16.5 MPLAB C30 C Compiler

The MPLAB C30 C compiler is a full-featured, ANSI compliant, optimizing compiler that translates standard ANSI C programs into dsPIC30F assembly language source. The compiler also supports many command-line options and language extensions to take full advantage of the dsPIC30F device hardware capabilities, and afford fine control of the compiler code generator.

MPLAB C30 is distributed with a complete ANSI C standard library. All library functions have been validated and conform to the ANSI C library standard. The library includes functions for string manipulation, dynamic memory allocation, data conversion, time-keeping, and math functions (trigonometric, exponential and hyperbolic). The compiler provides symbolic information for high level source debugging with the MPLAB IDE.

16.6 MPLAB ASM30 Assembler, Linker, and Librarian

MPLAB ASM30 assembler produces relocatable machine code from symbolic assembly language for dsPIC30F devices. MPLAB C30 compiler uses the assembler to produce it's object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire dsPIC30F instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

16.7 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC hosted environment by simulating the PICmicro series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user defined key press, to any pin. The execution can be performed in Single-Step, Execute Until Break, or Trace mode.

The MPLAB SIM simulator fully supports symbolic debugging using the MPLAB C17 and MPLAB C18 C Compilers, as well as the MPASM assembler. The software simulator offers the flexibility to develop and debug code outside of the laboratory environment, making it an excellent, economical software development tool.

16.8 MPLAB SIM30 Software Simulator

The MPLAB SIM30 software simulator allows code development in a PC hosted environment by simulating the dsPIC30F series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user defined key press, to any of the pins.

The MPLAB SIM30 simulator fully supports symbolic debugging using the MPLAB C30 C Compiler and MPLAB ASM30 assembler. The simulator runs in either a Command Line mode for automated tasks, or from MPLAB IDE. This high speed simulator is designed to debug, analyze and optimize time intensive DSP routines.

TABLE 17-8: CAPTURE/COMPARE/PWM REQUIREMENTS

Param No.	Sym	Characteristic			Min	Typ†	Max	Units	Conditions
50*	TccL	CCP input low time	No Prescaler		0.5TCY + 20	—	—	ns	
			With Prescaler	16F62X	10	—	—	ns	
				16LF62X	20	—	—	ns	
51*	TccH	CCP input high time	No Prescaler		0.5TCY + 20	—	—	ns	
			With Prescaler	16F62X	10	—	—	ns	
				16LF62X	20	—	—	ns	
52*	TccP	CCP input period			$\frac{3T_{CY} + 40}{N}$	—	—	ns	N = prescale value (1,4 or 16)
53*	TccR	CCP output rise time		16F62X		10	25	ns	
				16LF62X		25	45	ns	
54*	TccF	CCP output fall time		16F62X		10	25	ns	
				16LF62X		25	45	ns	

* These parameters are characterized but not tested.

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

FIGURE 17-12: TIMER0 CLOCK TIMING

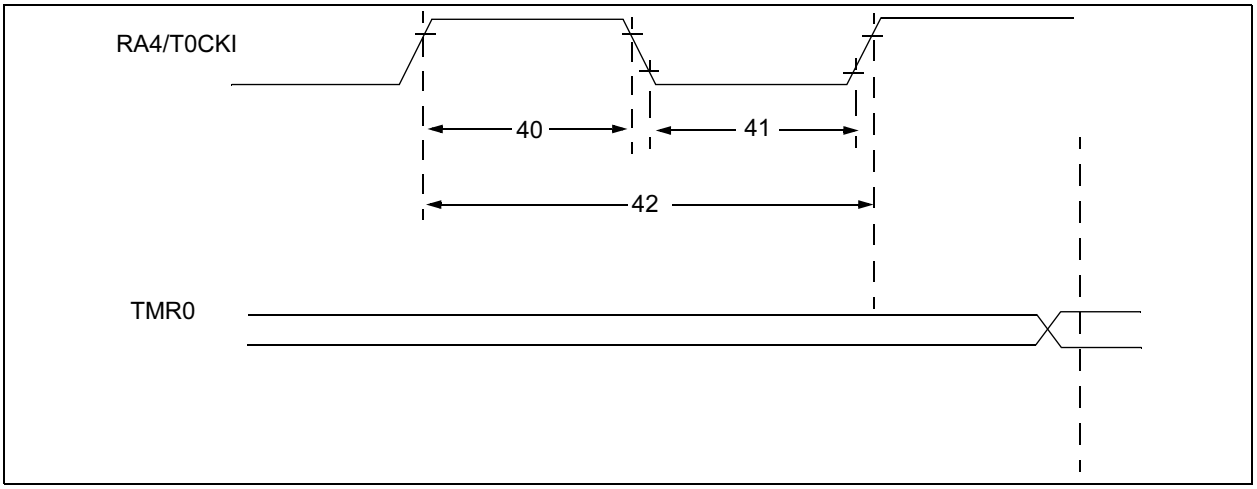


TABLE 17-9: TIMER0 CLOCK REQUIREMENTS

Param No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width	No Prescaler	0.5 TCY + 20*	—	—	ns	
			With Prescaler	10*	—	—	ns	
41	Tt0L	T0CKI Low Pulse Width	No Prescaler	0.5 TCY + 20*	—	—	ns	
			With Prescaler	10*	—	—	ns	
42	Tt0P	T0CKI Period		$\frac{T_{CY} + 40}{N}$*	—	—	ns	N = prescale value (1, 2, 4, ..., 256)

* 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.

Note: The graphs and tables provided in this section are for design guidance and are not tested.

FIGURE 18-22: VIN vs VDD TTL

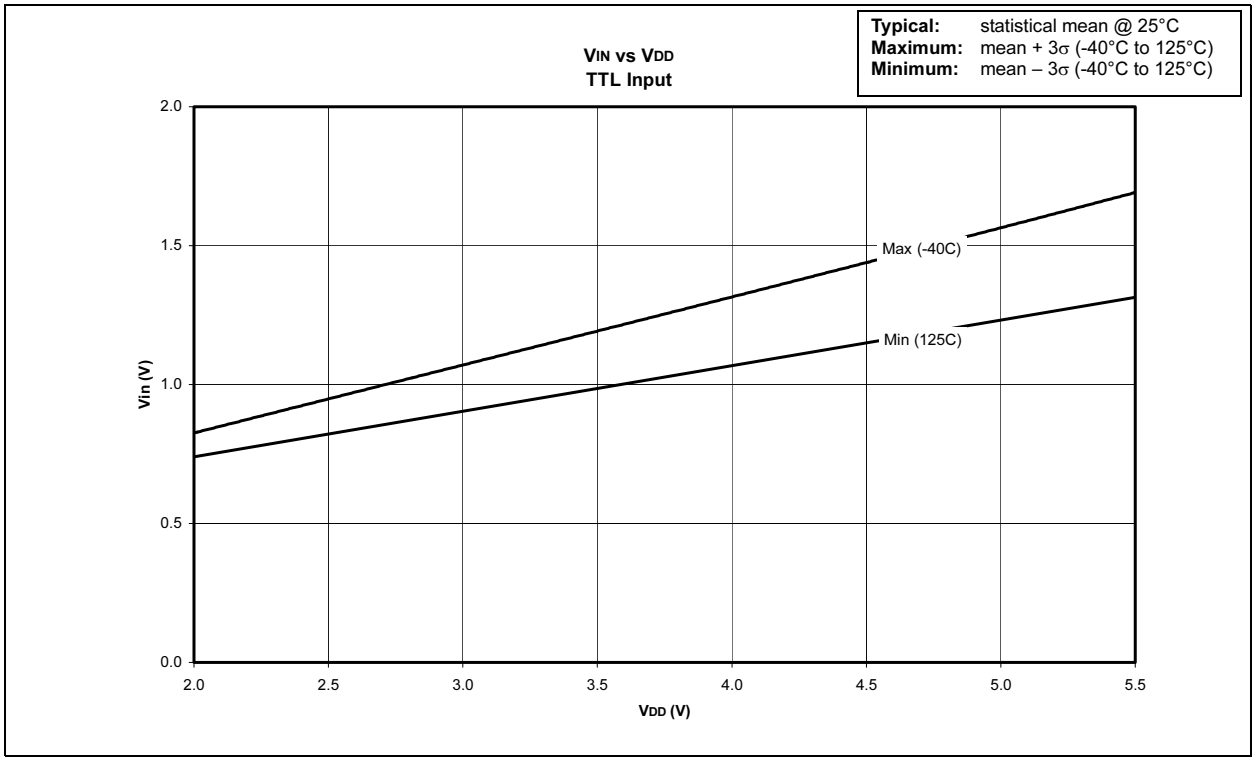
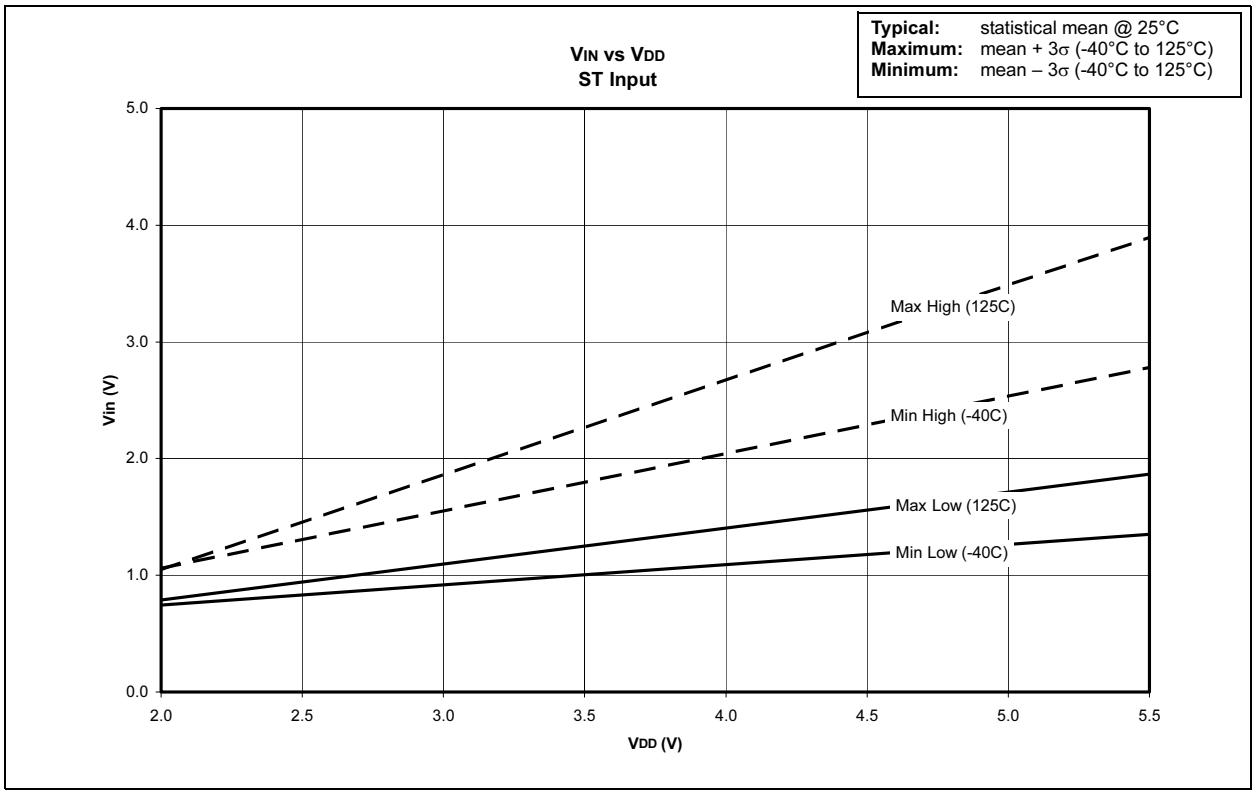


FIGURE 18-23: VIN vs VDD ST INPUT



INDEX