

# EE 361L Digital Systems and Computer Design Laboratory

University of Hawaii

Department of Electrical Engineering

by Galen Sasaki and Ashok Balusubramaniam

## Quick Overview of PIC16F8X

Version 1.0

Date: 9/4/01

This document provides a quick overview of the PIC16F8X (which includes 16F84 and 16F84A) microcontrollers by Microchip. This is an 18-pin chip as shown in Figure 1.

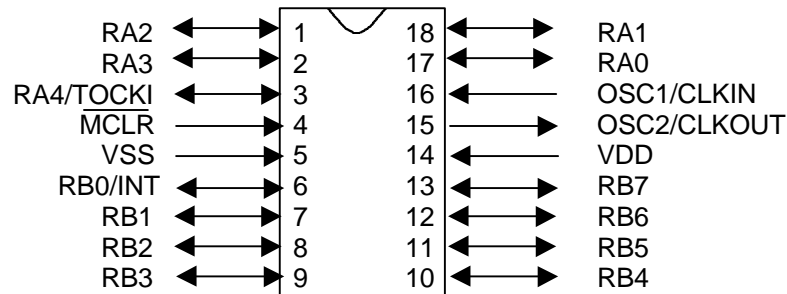


Figure 1. Pin Diagram

Features of the chip are

- 5-bit I/O port PORTA (pins RA4-RA0)
- 8-bit I/O port PORTB (pins RB7-RB0)
- Interrupts, including RB0/INT pin and PORTB<7:4>. Brief description of interrupt: An "interrupt" is a mechanism that allows a microprocessor to execute an alternate job from its main routine. This second job is often referred to as an *interrupt handler*. The interrupt handler may be invoked through hardware or software. If it is by hardware then it often corresponds to an electrical signal on a pin. If it is by software then it often corresponds to a machine instruction.

Other pins of the chip include

- $\overline{\text{MCLR}}$ : Master clear (reset) input/programming voltage input. This pin is an active low reset to the device.
- VSS: Ground reference for logic and I/O pins.
- VDD: Positive supply for logic and I/O pins.
- RA4/TOCKI: An interrupt input.
- OSC1/CLKIN: Oscillator crystal input/external clock source input.

- ❑ OSC2/CLKOUT: oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 frequency of OSC1, and denotes the instruction rate (for this processor, each instruction takes four clock cycles to execute).

Important registers for programming

- ❑ TMRO: 8-bit real-time clock/counter
- ❑ OPTION: Used to select options for controller
- ❑ TRISA: Selects data direction of PORTA
- ❑ TRISB: Selects data direction of PORTB
- ❑ PORTA: RA4-RA0
- ❑ PORTB: RB7-RB0
- ❑ INTCON: Used to select interrupt options
- ❑ Configuration word: Used to select options for the controller

## 2. Registers

The following is a description of some of the important registers. Included are the Value on Power-on Reset and Value on all other resets. Legend: x = unknown and u = unchanged. Also included are the addresses of the register in hexadecimal (prefix "0x").

### 2.1 TMRO

Description: 8-bit real-time clock/timer. We will ignore this for now.

Value on Power-on Reset:        xxxx xxxx

Value on all other resets:        uuuu uuuu

Address:                                0x01

### 2.2 OPTION\_REG

Description: Used to select options for microcontroller

- ❑ bit 7, RBPU: PORTB Pull-up enable/disable bit, where 0/1 = enable/disable
- ❑ bit 6, INTEDG: Interrupt edge select bit, where 0/1 = interrupt on falling/rising edge of RBO/INT pin
- ❑ bit 5, TOCS: TMR0 clock source select bit. This determines where the instruction clock will come from. If the bit = 0 then the clock will come internally, and CLKOUT will be an output of this clock signal. If bit = 1 then the clock will be from an external source that will be input the RA4/TOCKI pin.
- ❑ bit 4, TOSE:

Value on Power-on Reset:        1111 1111

Value on all other resets:        1111 1111

Address:                                0x81

### 2.3 TRISA

Description: This determines the data direction of PORTA (pins RA4-RA0). Note that TRISA.n (n = 0, 1, ..., 4) corresponds to pin RAn. TRISA.n corresponds to bit n of the TRISA register. If TRISA.n = 0 then PORTA.n is an output, and if TRISA.n = 1 then PORTA.n is an input.

Value on Power-on Reset:     ---1 1111  
Value on all other resets:    ---1 1111  
Address:                     0x85

## 2.4 TRISB

Description: This determines the data direction of PORTB (pins RB7-RB0). Note that TRISB.n (n = 0, 1, ..., 7) corresponds to pin RBn. TRISB.n corresponds to bit n of the TRISB register. If TRISB.n = 0 then PORTB.n is an output, and if TRISB.n = 1 then PORTB.n is an input.

Value on Power-on Reset:     1111 1111  
Value on all other resets:    1111 1111  
Address:                     0x86

## 2.4 PORTA

Description: This is a 5-bit port corresponding to pins RA4-RA0. PORT.n (n = 0, 1, ..., 4) corresponds to RAn.

Value on Power-on Reset:     ---x xxxx  
Value on all other resets:    ---u uuuu  
Address:                     0x05

## 2.5 PORTB

Description: This is an 8-bit port corresponding to pins RB7-RB0. PORT.n (n = 0, 1, ..., 7) corresponds to RBn.

Value on Power-on Reset:     xxxx xxxx  
Value on all other resets:    uuuu uuuu  
Address:                     0x06

## 2.6 INTCON

Description: The user may enable/disable and check the status of interrupts with this register.

- bit 7, GIE: Global interrupt enable bit. 0 = disables all interrupts. 1 = enables all unmasked interrupts.
- bit 6, EEIE: EE write complete interrupt enable bit. 0/1 = disables/enables interrupt
- bit 5, TOIE: TMR0 overflow interrupt enable bit. 0/1 = disables/enables TMR0 interrupt
- bit 4, INTE: RB0/INT interrupt enable bit. 0/1 = disables/enables RB0/INT interrupt
- bit 3, RBIE: RB port change interrupt enable bit. 0/1 = enables the RB port change interrupt
- bit 2, TOIF: TMR0 overflow interrupt flag bit. 0/1 = TMR0 has didn't/did overflow.
- bit 1, INTF: RB0/INT interrupt flag bit. 0/1 = the RB0/INT interrupt didn't/did overflow.
- bit 0, RBIF: RB port change interrupt flag bit. 0 = none of the RB7:RB4 pins have changed state. 1 = at least one of the RB7:RB4 pins changed state.

Value on Power-on Reset:     0000 000x  
Value on all other resets:    0000 000u  
Address:                     0x0b, 0x8b

Note that the flag bits must be cleared in software. Also note that interrupt flags get set regardless of whether an interrupt is enabled or disabled.

## 2.7 Configuration Word

Description: There are 12 bits in this word

- ❑ bits 13:4, Code protection bit: 0/1 = code protection on/off. For our purposes, these should be 1 for now.
- ❑ bit 3, PWRTE: Power-up timer enable bit. 0/1 = power-up timer is enabled/disabled. For our purposes, this should be disabled for now.
- ❑ bit 2, WDT: Watchdog timer enable bit. 0/1 = WDT disabled/enabled. For our purposes, this should be disabled for now.
- ❑ bit 1:0, FOSC1:FOSC0: Oscillator selection bits
  - 11 = RC oscillator
  - 10 = HS oscillator
  - 01 = XT oscillator
  - 00 = LP oscillator

Value on Power-on Reset:        uuuu uuuu

Value on all other resets:        uuuu uuuu

Address:                            0x2007. This is beyond user program memory. The bits of this word are set at the time the chip is being programmed.

## 3. C Programming Using PIC C Lite

The following are some C programming structures for the PICC Lite compiler.

### 3.1 Setting the Configuration Word

```
#include <pic.h>
_CONFIG(x);            // This is a macro. x is the configuration word value.
```

### 3.2 Bit setting

To set and clear bits, the following macros may be used.

```
#define bitset(var,bitno) ((var) |= 1 << (bitno))
#define bitclr(var,bitno) ((var) &= ~(1 << (bitno)))
```

Then

```
bitset(foo,6);
```

will set bit 6 of foo to 1.

### 3.3 TRIS and PORT

The TRIS and PORT registers may be specified using TRISA, TRISB, PORTA, and PORTB. To specify individual bits of TRISA or TRISB, you may use TRISAm or TRISBn, where m = 0, 1, ..., 4 and n = 0, 1, ..., 7. For example, to set bit 3 of TRISA to 1, you can use

```
TRISA3 = 1;
```

To specify individual bits of PORTA or PORTB, you may use RBA.m or RBB.n, where m = 0, 1, 2, 3, 4 and n = 0, 1, ..., 7. For example, to clear bit 5 of PORTB to 0, you can use

```
RB5 = 0;
```

#### 4. C Examples

```
#include <pic1684.h>
// ver 1.0
// Created on 09/03/01 by Ashok B.
// This C program will make a 16F94 PIC microcontroller drive a 7 segment display so
// that the display counts "0", "1", ..., "9", "0", ... The display changes value every second.
// Thus, the program has delay functions that are loops. The number of times a loop is
// executed is based upon the clock rate to get the exact delay.
```

```
void delay_10us(unsigned char t); // t*10us delay
void delay_ms(long t); // t ms delay
void delay1sec(void); // 1 sec delay
unsigned char delay; //global used in the Assembly code
```

```
void main(void)
{
int count = 0; // The value that is to be displayed on the 7 segment display.
TRISA = 0b11111; // PORTA is an input
TRISB = 0b0000000; // PORTB is an output
while(1) {
    switch(count) // increment count and then output the value to PORTB
    // Seven segment LED display is used.
    //RB7 corresponds to 'a' of LED, RB6 corresponds 'b' of LED,..RB1 to 'g' of LED
    {
    case 0:
        PORTB = 0b00000011;
        break;
    case 1:
        PORTB = 0b11110011;
        break;
    case 2:
        PORTB = 0b00100101;
        break;
    case 3:
        PORTB = 0b01100001;
        break;
    case 4:
        PORTB = 0b11010001;
        break;
    case 5:
        PORTB = 0b01001001;
```

```

        break;
    case 6:
        PORTB = 0b00011001;
        break;
    case 7:
        PORTB = 0b11100011;
        break;
    case 8:
        PORTB = 0b00000001;
        break;
    case 9:
        PORTB = 0b01000001;
        break;
    }
    delay1sec();
    count++;
    if (count == 10) count = 0;
}
}

```

// The following are functions to create real-time delay.

```

void delay_10us(unsigned char t)
// provides t * 10 usecs of delay.
// Max of t is 255 which corresponds to 2550 usecs
{
    delay=t; // This is a global variable.
    // Below is an assembly language fragment to create a 10 microsecond delay,
    // Note that in this fragment the C integer variable "delay" is written as "_delay".
    #asm
    DELAY_10US_1:
        CLRWDT
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        DECFSZ _delay,f
        GOTO DELAY_10US_1
    #endasm
}

void delay_ms(long t) // delays t millisecs
{
    do {
        delay_10us(100);
    } while(--t);
}

void delay1sec(void)
{
    delay_ms(1000);
}

```