PIC I/O Tutorial

By: Ricardo Arencibia
**Purpose:**

This tutorial will cover the basic Input and Output features available in the PIC18F1320. In this tutorial, we are going to read digital values from a port using a DIP switch and pull up resistors. Then we are going to write the values read into another port to turn on LEDs. We are also going to blink LEDs back and forth. After completing this module, you should be able to read and write digital data from any port in a microprocessor. This module will be tailored to the Microchip PIC18F1320, but includes general hints and procedures applicable to most microprocessors.

**Concepts:**

Before we begin let’s review the basic concept of a microprocessors IO module:

**REGISTER:**

A register stores bits of information. You can write to a register or read from a register. They are especially useful to control the features and configurations of the different modules included in modern microprocessors.

**PORT:**

If you think of a microprocessor as a castle, then a port would be its gate. It allows data in and out of the microprocessor, hence input and output port. Ports can be digital or analogue. Digital ports handle digital data (high ‘1’ or low ‘0’ values). These high and low values are represented by voltage ranges. For the PIC18F1320, a high value is represented by a voltage within a range of 0.8 Vdd to Vdd and a low value is a voltage within the range of 0.2Vdd to Vss (Datasheet Page 254). Analogue ports handle analogue data. These ports are connected to an ADC circuit to sample continuous voltages and convert them into digital values.

**PORT REGISTERS:**

To use a port, as with most modules, in a microprocessor you must access its registers:

**DATA DIRECTION REGISTER:** Configures a port to be an input or output. An input port can read and an output port can write.

**DATA REGISTER:** Stores the digital data that is read or written by a port.
Writing to Port A

Port A is composed of 8 pins RA0 – RA7. Let’s output a high level to all of them.

**Step 1:** Configure all Port A pins as outputs

0b00000000

TRISA

**Step 2:** Write to Port A

0b11111111

PORTA

---

**Pic Microprocessor Naming Convention**

DATA DIRECTION REGISTER = **TRISx**

Input/Read = ‘1’

Output/Write = ‘0’

DATA REGISTER = **PORTx**

---

Reading from Port A

**Step 1:** Configure all Port A pins as inputs

0b11111111

TRISA

**Step 2:** The value read will appear in the PORT A register

0b01010011

PORTA
Reading and Writing from Port B

Step 1: Configure Port A pins

TRISA

Step 2: Write and Read from PORT A register

PORTA

The Build:

Now that we understand the concepts we are going to read digital values from pins RB0 – RB3 in PortB using a DIP switch and pull up resistors. Then we are going to write the values read into pins RA4 - RA7 to turn on LEDs.

Warning: Port A is set as an analogue port by default. RA4 and RA5 are inputs only.

Parts:

- Resistor DIP
- DIP Switch
- LEDs
- PIC18F1320
Code:

//The #include statement lets you include header files with definition and functions
#include "plibcxxx.h"

//The #pragma config statement is used as configuration directive. For more information on
//configuration bits go to MPLAB IDE > HELP > TOPICS > PIC18 Config Settings
#pragma config OSC = INTIO2 // Internal OSC, RA6 and RA7 are I0s
#pragma config WDT = OFF     // WATCH DOG TIMER OFF
#pragma config MCLRE = ON    // MASTER CLEAR PIN OFF
#pragma config LVF = OFF      // LOW VOLTAGE PROGRAMMING

void main(void)
{
    //Config PORTA and PORTB as digital ports. By default
    //PORTA and PORTB are analogue ports. Refer to page 158 in the
    //datasheet for more information.
    ADCON1 = 0xFF;

    //Configure Data Direction Register (Input = '1', Output = '0')
    //Output = RB0 - RB3
    //Input = RA0 - RA3
    //  1 1 1 1 0 0 0 0
    // RA3 RA2 RA1 RA0 RB3 RB2 RB1 RB0
    //
    // To set individual bits you can use TRISAbits.RAx . Refer to header file
    // for more information
    TRISA = 0x0F;
    TRISB = 0x0;

    //Clear PORT B
    PORTB = 0;

    while(1)
    {
        //Transfer the value read in pins RA0 through RA3 to pins RB0 through RB1
        PORTB = (PORTA & 0x0F);
    }
}
Results:
The code below shows you how to blink your LEDs left and right like knight rider. The schematic is the same as the previous example.

**Code 2:** Blink LEDs

```c
#include <p18cxxx.h>
#include <delays.h>

#pragma config OSC = INTIO2   // Internal OSC, RA6 and RA7 are IOs
#pragma config WDT = OFF       // WATCH DOG TIMER OFF
#pragma config MCLRE = ON      // MASTER CLEAR PIN OFF
#pragma config LVP = OFF       // LOW VOLTAGE PROGRAMMING

void main(void)
{
    // Config PORTA and PORTB as digital ports.
    ADCON1 = 0xFF;

    // Configure PORTB as digital output
    TRISB = 0x0;

    // Clear PORT B
    PORTB = 0;

    while(1)
    {
        int i = 1;

        // Blink LED towards right
        for(i ; i <= 16 ; i = i*2)
        {
            PORTB = i;
            Delay10TCYx(400);
        }

        i = 8;

        // Blink LED towards left
        for(i ; i > 1 ; i = i/2)
        {
            PORTB = i;
            Delay10TCYx(400);
        }
    }
}
```