

TUT#1 - TUTORIAL SERIES

HELLO WORLD

This document is copyright and is considered correct at the time of printing / release. The content contained herein is subject to change without notice and no liability is accepted for use of the products contained herein. All product names, trademarks mentioned in this document remain the property of the respective owners.



DUETRONIX

Table of Contents

Revision Information.....	3
Tutorial 1: Hello World.....	4
Configuring the PORT.....	4
Toggle the LED	6
Alternate Code Example	6
Summary	7
Quiz	7
Answer	7
Final Tip.....	8

Revision Information

The following table details important information pertaining to the document. For the latest or current version of this material, go to the following site.

Document Title	Tutorial Series
Date last revised	May, 2008
Revision Detail	1.0.0
Author	Duetronix Information Resources Dept

Tutorial 1: Hello World

Note: The C Compiler in use is the Mikro C compiler Ver 8.1 Demo version.

Now that you have assembled your ProtoPro development tool, let's get started with a project. The basics of ports and registers will be covered very quickly and in the context of the project explanation, but you will catch on as we progress. The subject of ports and how to use them is extensive and ongoing, so no need to bog a beginner down with the details.

Note: Regardless of which ProtoPro you have purchased, this project can be adapted easily to any pic.

PICs from the 16F and 18F series have a "compatible" architecture as they are both 8 BIT microcontroller families. Once you cross over to the DSPICs, PIC 24, and 32 BIT you are dealing with 16 and then 32 bit microcontrollers so they are different.

All PICs have ports and these are an orderly, physical collection of inputs and outputs. A port can have a maximum of 8 IO which does incidentally trace back to 8 bits or $2^8 = 256$.

The Ports are bi directional, meaning simply that depending on how they are configured, they can be used to receive an incoming signal or provide an outgoing signal. Incoming is referred to as an input. Outgoing is an output.

For the hello world project we are going to create a simple project which displays a single LED (Light Emitting Diode) flashing at a regular rate.

Configuring the PORT

Note: There are different types of ports; therefore their characteristics will determine how they can be used, etc.

The port we will use is PORT B. This port is available on most micros, 18 – 40 PIN and the characteristics are similar. PORT B is an 8 BIT port, has some interrupts and some other features beyond the scope of the current so we will worry about the rest later.

There are two registers that control the operation of PORT B, namely TRIS B and PORT B. A TRIS register controls the direction Input or Output and the PORT register holds the value of the data on the register.

Before a micro can do anything it must be initialized. This is usually quite simple as there is no need to configure the entire micro, but only the peripheral that you want to use. In this case it's port b so we do the following.

```

Main()
{
    TRISB=0x0; // PORTB initialized, all outputs.
}
    
```

Sample #1

When configuring simple or digital I/O there are two states, 1 which defines the pin for input and 0 which defines the pin for output. A combination of 1 and 0 can be applied to a single port. So if you want the lower 4 bits to be input and the upper 4 bits output, see below.

PORTB	7	6	5	4	3	2	1	0
OXF0	1	1	1	1	0	0	0	0

For those who have perhaps forgotten the different types of numbers, 0 has value for a PIC. The first address in memory is 0, and the first PIN is 0 and not 1. So a port's labels start at zero or PIN 1 and end on the eighth pin, 7. It's the difference between natural and counting numbers.

The method of writing values to the TRIS register is HEX; you can use decimal if you like but try staying clear. It's not as common as you would hope. As you can see in the table below the PORT B label is a value in HEX, and this is the equivalent value of the binary number in the table above.

In other words 0xF0 (Base 16) = 11110000(Base 2) =240 (Base 10) <- all the same value, just represented differently.

To apply what we have learnt, we have changed the I/O configuration as shown below.

```

Main()
{
    TRISB=0xF0; // PORTB initialized, upper 4 inputs, lower 4 outputs
}
    
```

Sample #2

To get the LED to flash, we do the following. We select one of the pins e.g. PORTB BIT 0 and this is where we connect the LED. We are going to SET (switch on) and CLEAR (switch off) the pin alternately as the code runs.

Toggle the LED

There are a number of ways to get a PIN to toggle on and off. First of all, a pin once written to will hold that state it was set to until a change comes about. So if the PIN is set to 1 it stays that way unless you write a zero to the PIN. We use a timer to control the toggle rate. When the timer elapses, the state gets toggled. You can write more complicated code checking if the timer has expired to either set or clear the pin but that's not necessary. Shown below is a quick way.

```
Main()
{
    TRISB=0xF0; // PORTB initialized, all outputs.

    While(1)
    {
        Delay(20); // Delay of 20ms
        PORTB.F0=PORTB.F0^1; //Toggles status of pin
    }
}
```

Sample #3

- The while (1): statement puts the PIC into an infinite loop, so the code will run indefinitely.
- Delayms(20): sets a delay period of 20ms between on and off states for the LED
- PORTB.F0=PORTB.F0^1: Toggles the pin status using the XOR expression, basically if the PIN is off, it gets switched on and if on, off.

Note: When writing code, especially for an embedded application simple is more. Most PICs have an eventual code limitation of 8K words. Some are as limited as 1 K word. So complicated code is just going to wasted program memory not to mention create long debugging cycles to fix your mistakes.

Alternate Code Example

Just to illustrate another way to accomplish the same as above, the "if then" statement is used to toggle the same PIN. Notice that the code is a line shorter. This type of optimized code construction can be useful when writing larger programs to save space.

```
Main()
{
    TRISB=0xF0; // PORTB initialized, all outputs.

    While(1)
    {
        Delay(20); // Delay of 20ms
        //
        If (PORTB.F0==1) PORTB.F0 =0;
        If (PORTB.F0==0) PORTB.F0 =1;
    }
}
```

Sample #4

And with that, we have created a hello world project. Flash the PIC and watch the LED run.

Summary

- All PICs have ports designated alphabetically e.g. PORT B
- PORTS have up to 8 I/O lines and are configurable using the TRIS command
- Read / Write operations happen using the PORT command
- Write code simply, its best practice – helps debug!

Quiz

Rewrite the code below, to get Pin B1 to toggle alternately to B0 (i.e. When LED on PORTB.F0 is on, the LED on PORTB.F1 is off and vice versa.)?

```
Main()
{
    TRISB=0xF0; // PORTB initialized, all outputs.

    While(1)
    {
        Delay(20); // Delay of 20ms
        PORTB.F0=PORTB.F0^1; //Toggles status of pin
    }
}
```

Sample #5

Answer

```
PORTB.F1=PORTB.F0^1;
```

The XOR function is used such that whatever the value of PORTB.F0 is set to, PORTB.F1 will be opposite.

```
Main()
{
    TRISB=0xF0; // PORTB initialized, all outputs

    While(1)
    {
        Delay(20); // Delay of 20ms
        PORTB.F0=PORTB.F0^1; //Toggles status of pin F0
        PORTB.F1=PORTB.F0^1; //Toggles status of pin F1, alternate to F0
    }
}
```

Sample #6

Final Tip

During the course of the code you will notice the use of `//` this comments out, or isolates text that describes the code. Commenting your code can be very useful as time goes by, especially for large programs or if you are passing on code, upgrading your code in 10 years time. It helps you remember what you were doing and thinking at the time.

Disclaimer and Notices

This work is ©, 2008 Duetronix, South Africa. The document may be used for personal or educational use but may not be used by any corporate or commercial training institution whatsoever without the expressed permission of Duetronix.

All product names and branding mentioned or referred to in this work remain the copyright and intellectual property of their respective owners.

All rights reserved.