

Introduction:

The mbed is an embedded development system based on the 32-bit ARM Cortex-M3 microcontroller. It is designed to help facilitate the rapid development of embedded electronics applications through powerful computing capabilities combined with an easy-to-use software environment that's readily accessible on any computer. Very few (if any) tutorials on the mbed device exist at the time of this writing, however, so we here at NBitWonder have taken the liberty of writing one of the first official mbed tutorials in the hopes that you will find it useful and get out there and start making awesome things with it! In this tutorial, we will interface the mbed to the computer, download a simple program, and then learn how to use the online compiler and write a simple program for the mbed.

Disclaimer:

The contents of this tutorial are made freely available for public use under a Creative Commons Attribution 3.0 License. For more information on this license, please refer to <http://creativecommons.org/licenses/by/3.0>. The purpose of these tutorials is to be informative and, hopefully, fun. These tutorials are written and tested to ensure that they can be performed in a safe, non-destructive manner. To that end, NBitWonder and the author will not be held accountable for any incident of injury, death, or property destruction resulting from improper use of the instructions contained within this tutorial. Be as safe as you are smart, have fun, and we will make it through the tutorial with no unfortunate mishaps.

Tools and Materials:

For this tutorial, you will need the following tools and materials:

- mbed microcontroller(1x) : The mbed microcontroller is available from a number of distributors, including [Arrow Electronics](#), [Sparkfun](#), [DigiKey](#), [Mouser](#), and [Farnell](#).
- USB to miniUSB Cable(1x) (Included with the mbed system)
- Solderless Breadboard (x1) (Optional)

And that's it! No special device programmers, no special software, just your system and a cable.

Unboxing the mbed:

The mbed package contains the mbed, 2 pinout cards, the mbed device module, some simple instructions, and a USB to miniUSB cable. Please make sure your box has all of these components.



Fig 1. mbed Kit Components

Connecting to the Computer:



To interface the mbed to the computer, insert the USB cable into the miniUSB socket. Upon connecting the mbed, it should show up as a flash drive on your computer. If the mbed programmer does not appear on your computer, make sure the USB cable is plugged all the way in and try again. Additionally, Windows occasionally has trouble seeing new drives right away, so hitting F5 in an explorer window may be a useful fix. For any other problems, contact mbed support at support@mbed.org.

Going Online:

Once the mbed is connected to the computer, navigate to the mbed drive and open the MBED.htm file located in the drive.

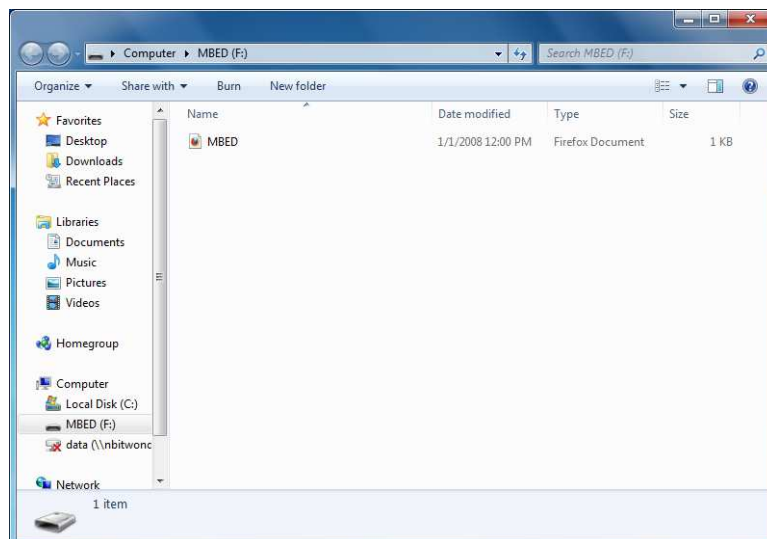


Fig 2. mbed Drive Contents

Opening this file brings up the webpage shown below:

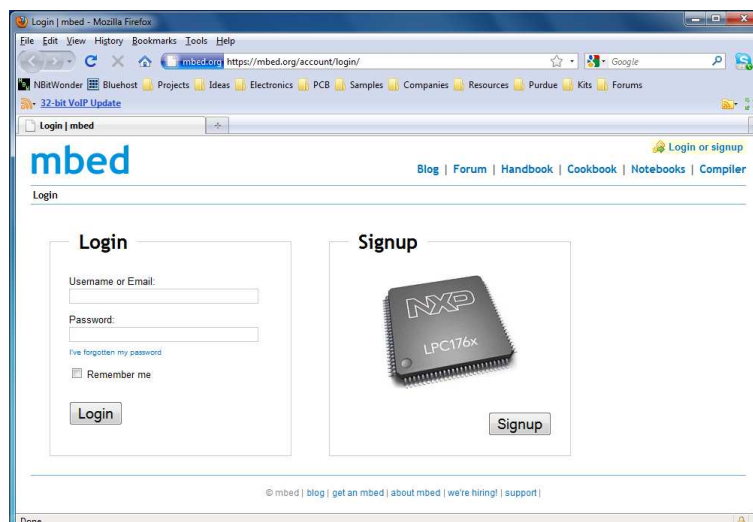


Fig 3. mbed Online Portal

If you haven't already, go ahead and create an account. Once you have created an account or logged in to an existing account, you will be taken to the webpage shown below:

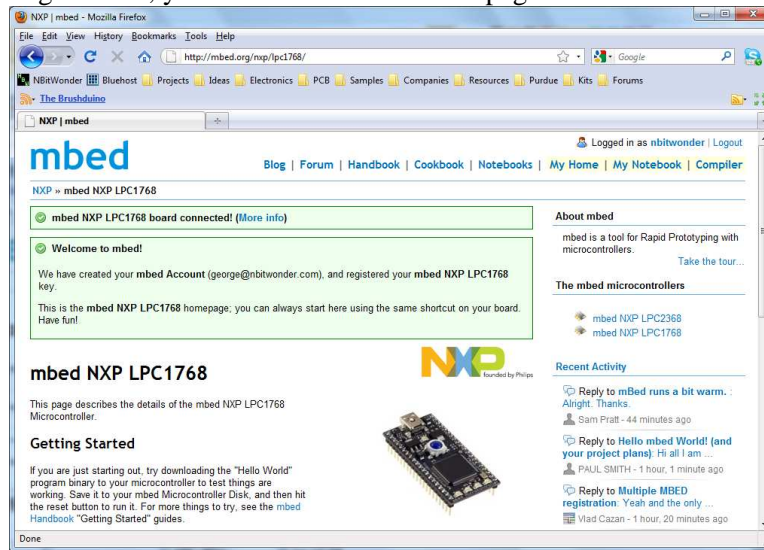


Fig 4. mbed Homepage

The First Program: HelloWorld:

From this page you can access numerous resources related to the mbed microcontroller, which we will explore in more detail in future tutorials. For now, however, you should simply navigate down the main page until you come to the icon shown in figure 5.



Fig 5. HelloWorld icon

Click on this icon to download the HelloWorld binary file to your computer. Then, if it isn't already there, copy or move the file to your mbed drive. The blue led indicator light near the USB port should flash while the file is being copied. With the file saved to your mbed, hit the reset button on the mbed board. Upon doing so, a single blue led should begin to flash. Congratulations, you have successfully run your first program on the mbed system!

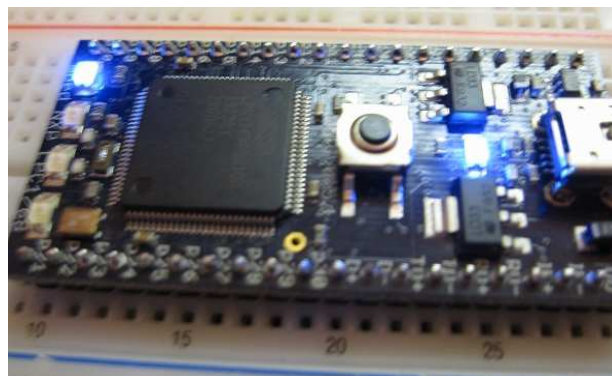


Fig 6. mbed Running HelloWorld Program

Moving Beyond HelloWorld: Using the mbed Compiler:

While great, the HelloWorld program doesn't do a whole lot, and is mostly meant to develop a basic understanding of the architecture being used. If you wish, you may remove the HelloWorld program from your mbed drive. Now that HelloWorld has been done, it's time to move further, developing a better understanding of the mbed compiler and writing programs for the mbed embedded development system. Return to the mbed homepage (<http://mbed.org>) and click on the "Compiler" tab in the upper right corner. After initializing, your screen should look something like figure 7.

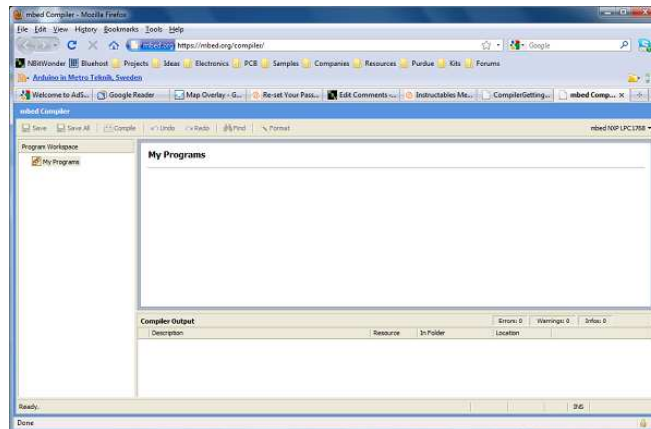


Fig 7. mbed Compiler Window

Now, right click the "My Programs" icon and select "New Program". A window will pop up asking you to enter project information. I named my project "Counter", but feel free to name yours whatever you want. Upon creating the project, a new named project will appear, containing the main.cpp and mbed.h files. The created main.cpp file should contain all of the code necessary to implement the HelloWorld program and is shown in figure 8.

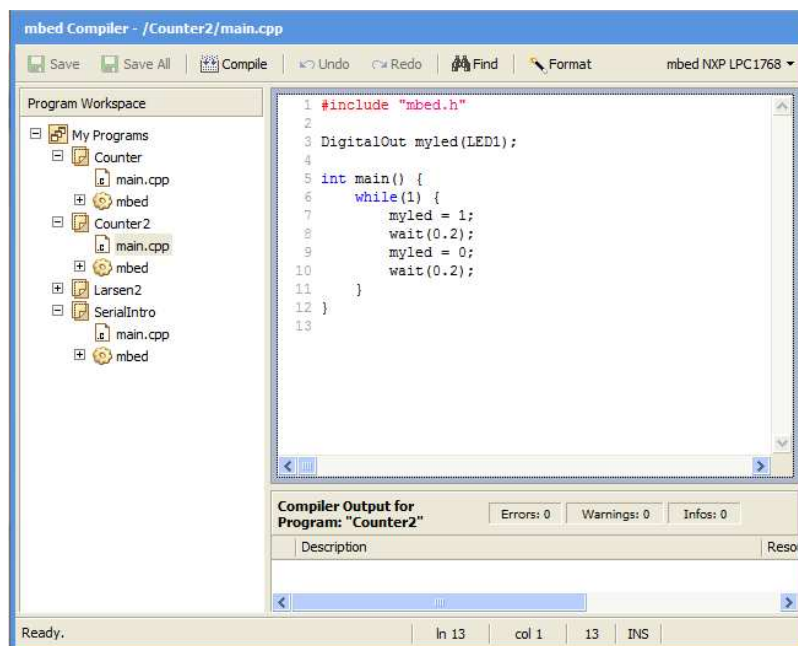


Fig 8. main.cpp Initial Program Setup

The main.cpp begins with an include statement that includes the mbed header file. After this is a pin assignment created using the DigitalOut function, which is a function from a library of the same name used to create a digital output pin. For more information about the mbed libraries and their functions, select the mbed documentation file (located just below main.cpp in this example) and select an mbed library to learn more about the functions available in that library. After the pin assignment statement is the main loop of the program, written in C++. The mbed compiler supports C++ syntax and constructs within any of its code and also allows standard C++ libraries to be included and used in addition to the specialized mbed libraries.

Writing a Program: Binary Counter Example:

Program Description

For the first written program, we are going to design and code a 4-bit binary adder display. The 4 leds on the mbed module will represent the bits, with a 1 represented by a lit led, and a 0 represented by an unlit led. For the example code, the display will increment once a second.

Variables and Initializations

For this program, we will require four digital output pins to drive the four onboard mbed leds. Additionally, we will need a 4-bit value to hold the current value to be output to the display. The four led outputs are assigned to a 4-bit bus called Bits using the mbed BusOut function of the BusOut library. (For additional details on the BusOut function, consult <http://mbed.org/handbook/BusOut>) For this example the scope of the variable containing the binary value to display is not particularly critical, so it is made global and initialized just after the led pin assignments outside the main function. For this program, the variable is called bin_val and is initialized as type char.

Implementing the main() Function

The main function for this program is implemented inside of an infinite while(1) loop. Every cycle through the program, the value inside bin_val is incremented. In C++, char values are represented as 8 bits, but for the purposes of our binary counter, only 4 bits are needed. Therefore, we need to include a check to ensure that a 4-bit overflow condition does not occur (i.e. the value in bin_val needs to be reassigned to 0 if it exceeds decimal 15). Once this is done, the value of bin_val must be assigned to Bits to display on the output. Finally, **do not** forget to include the wait(1) statement at the end of the loop. Failure to include this statement will result in the leds updating much too fast for the human eye to see, and creating the appearance that all leds are on simultaneously. When you are finished writing your main function for the binary counter, it should look similar to that shown in figure 9.

```
#include "mbed.h"

//LED Output initializations
BusOut Bits(LED1,LED2,LED3,LED4); //LED outputs for counter

char bin_val = 0x00;
int main() {
    while(1) {
        //increment bin_val, resetting if 4-bit overflow condition occurs
        bin_val++;
        if(bin_val > 0x0F)
        {
            bin_val = 0;
        }
        //Assign bin_val to led output
        Bits = bin_val;
        //wait 1 second
        wait(1);
    }
}
```

Fig 9. Binary Counter main() Function

Additional resources, links, and code related to this tutorial can be found at <http://nbitwonder.com/tutorials/microcontrollers/mbed/introduction>.

Conclusion:

In this tutorial, we were introduced to the mbed embedded development system and downloaded our first program. Then, we became more familiar with the mbed compiler, writing a simpler binary counter program. We here at NBitWonder hope you have enjoyed this first foray into the world of 32-bit microcontroller development and that you will join us for subsequent tutorials as we delve deeper into the mbed and other microcontroller architectures. Code for the mbed published by NBitWonder can be found at <http://mbed.org/users/nbitwonder/> and <http://mbed.org/users/benbitwonder/>, and you can always receive the latest information and news on NBitWonder at our website, <http://nbitwonder.com>.

Sources Cited:

- “mbed Setup Guide”, November 2009. [Online]. Available: <http://mbed.org/handbook/Setup>. [Accessed February 16th, 2010].
- “Downloading a Program”, November 2009. [Online]. Available: <http://mbed.org/handbook/MicrocontrollerGettingStarted>. [Accessed February 16th, 2010]
- “Getting Started – Compiling a Program”, November 2009. [Online]. Available: <http://mbed.org/handbook/CompilerGettingStarted>. [Accessed February 16th, 2010].