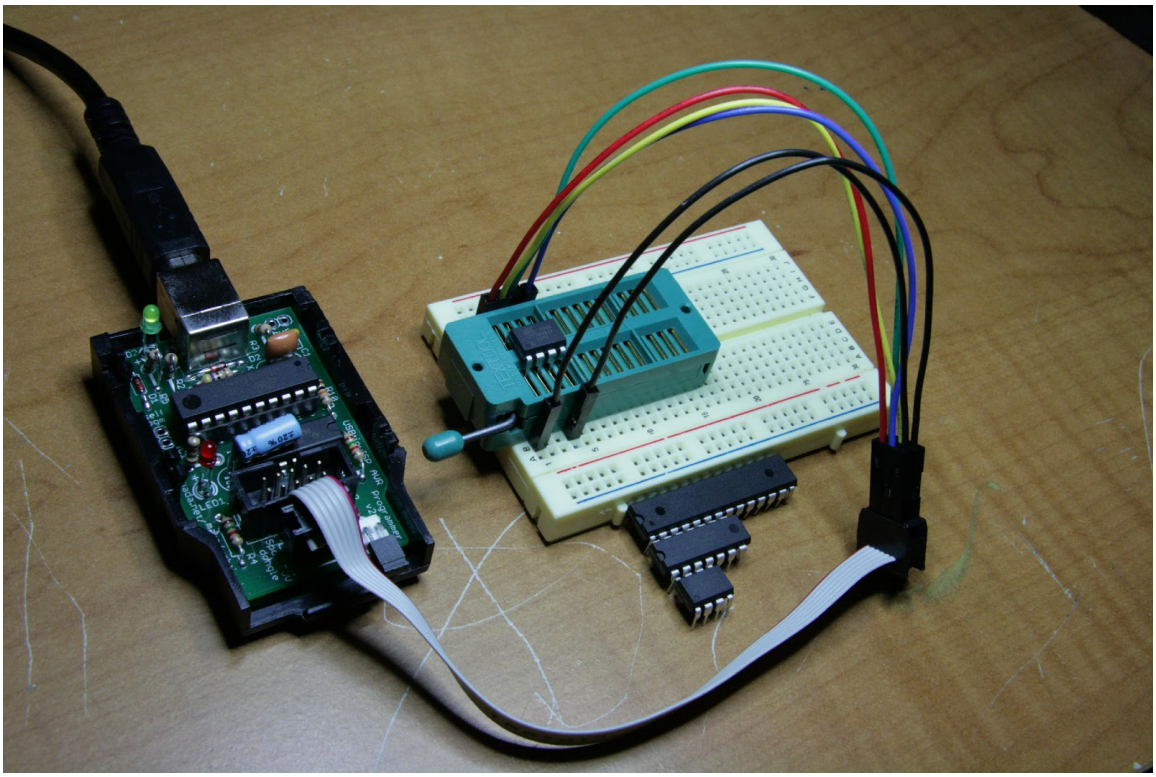
HOW TO PROGRAM A MICROCONTROLLER

An Application Note

By: John Foxworth



INTRODUCTION:

In today's evolving world, technology is not only becoming more and more advanced, but also more and more common in our everyday lives. The invention of "smart" products is revolutionizing the design process for nearly every product imaginable. Shoes containing chips that pair with our phones to keep track of our activity, refrigerators that can track when groceries expire, and now even cars that are capable of driving themselves are all examples of modern inventions that use microcontrollers to make our lives easier.

A microcontroller is a programmable IC, capable of multiple functions depending on how it's programed. Many different kinds of microcontrollers exist that offer a wide range of functionality. The versatility of the microcontroller is what makes it one of the most powerful tools in modern design. This guide will explain the basics of microcontrollers and how they are programmed.

SELECTING A MICROCONTROLLER:

The features and functionality of microcontrollers are unique to each brand/model. Before coding a microcontroller for your project, you must select a model that meets all the requirements of your design. Common features people look for in a microcontroller include I/O pins, clock frequency and storage memory, however a countless number of other specifications exist. Writing code is pointless if the hardware can't interact with your circuit the way you need it to

For beginners, Arduino is a brand of microcontroller commonly used amongst hobbyists and professionals alike. Its software is open source, meaning anyone can contribute to the growing pool of recourses available to its users. A simple Google search will yield countless threads, blogs, and forums with examples, documents, and tutorials on how to write code for a desired application. The development suite used to program the Arduino is available for free on their website, Arduino.cc.

Texas Instruments also has a wide variety of microcontrollers available, some with specific hardware included on board to specialize them for more specific tasks. However, these are more complicated to program and use. Often the user must set data registers manually, often requiring them to read 600+ page data sheets to find the proper line of code or variable to achieve a certain goal. Compared to Arduino, the Texas Instrument microcontrollers have more to offer but require more recourses and time.

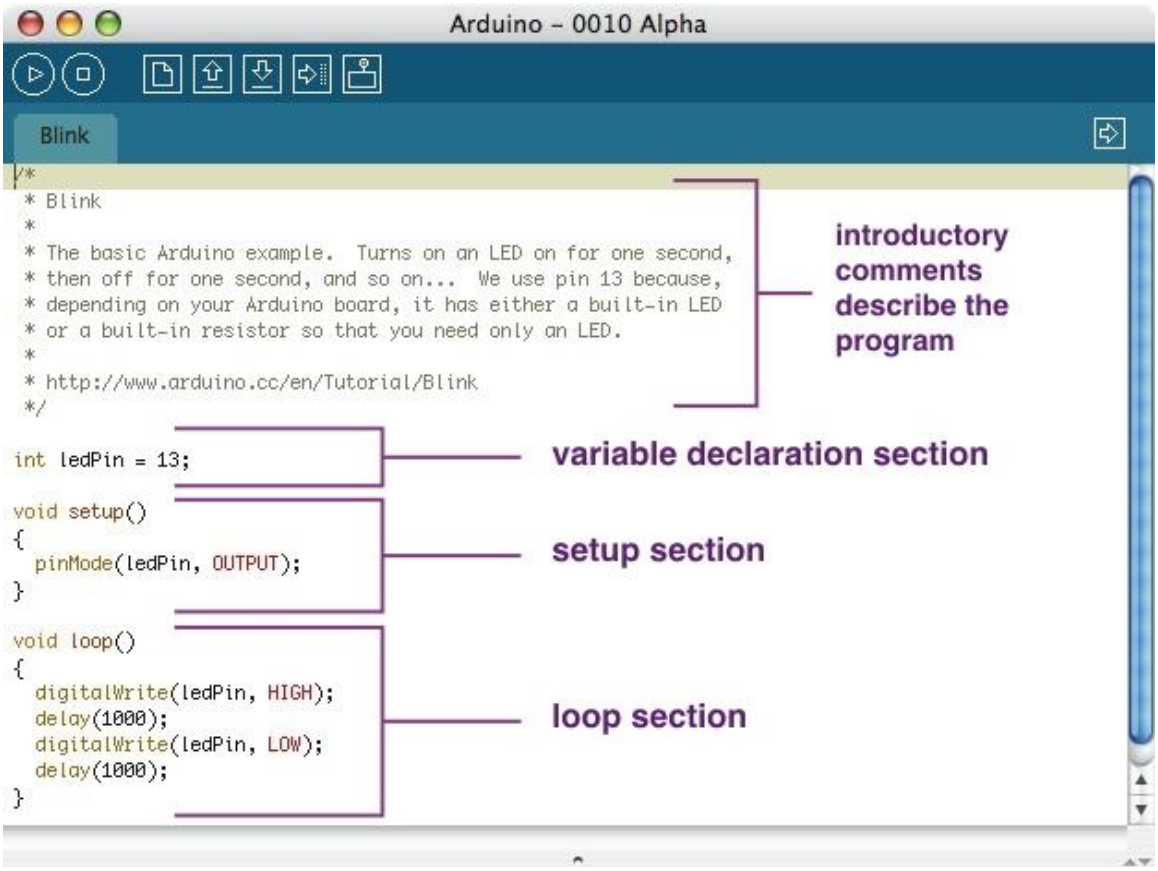
Countless other manufacturers such as Atmel, Intel, Sony and Ubicon all offer microcontrollers with different indented applications. Research and experience will help you select the optimal choice for your design. Many Internet resources such as http://www.instructables.com/id/How-to-choose-a-MicroController/ can help along the way.

PROGRAMMING:

Microcontrollers are typically programmed in higher-level languages such as C++ or Java. One of the essential tools needed to program a microcontroller is an integrated development environment (IDE). This software is usually developed by the creators of the microcontroller, and contains useful tools to help you program

your microcontroller. Common tools found in IDE's include, code editors, compilers, and debuggers. Depending on the application of the microcontrollers, additional features may be added as well.

Once a suitable IDE is obtained, you can begin writing code. For explanatory purposes, this guide will show an example of the Arduino IDE in use. Below is an example of a simple Arduino program that makes an LED blink on and off at a frequency of 1Hz. The code is split into 4 different sections as follows:



It is a generally accepted practice to start any code with a comment section containing a general description of what the code/program does. While this section has no effect on the programs functionality, it's always a good to document it for future reference. User instructions, company and copyright information are also commonly placed here as well.

The second section is the Variable decoration. These variables are global, and can be called in any sections that follow. It is also common to create variables to describe each pins function, and set them equal to the pin number on the board to make coding more intuitive.

Thirdly, comes the "Void Setup()" section. Digital pins on microcontrollers are commonly used as inputs or outputs, but very rarely can they be both. In this section, the user defines which pins are inputs or outputs, as well as any other parameters that must be initialized. While the method of doing so varies for different microcontrollers, almost all of them require a similar step to configure the microcontrollers internal circuitry to fit the needs of your design.

Lastly, the "Void Loop()" section. This section is where the function of your microcontroller is written. Any actions that require reading or writing values from pins, or computing the values of different variables is done here.

COMPILING AND UPLOADING:

This step is almost always handled by the IDE. Once your code is written, it must be uploaded to the microcontrollers. Most have USB interfaces, but some smaller microcontrollers require a special hardware to be programmed. While we typically program microcontrollers in higher level languages, the microcontroller itself runs on assembly. To translate code to a format usable by a microcontroller, a compiler must be used.

A compiler is a software tool that takes higher level code and optimizes it for assembly. Assembly provides specific instructions to the microcontroller on what register operations to perform to match the operation of the original code.

Once the assembly code is created, it can be uploaded to the microcontroller for testing.

DEBUGGING:

Not everything will work perfectly on your first attempt. Debugging is a important part of any design process and programming microcontrollers is no exception. Fortunately, there are several methods to help you locate errors in your code without excessive effort.

The first most basic method is to hook up your microcontroller to the circuit it's made to control. Often enough you can see what's wrong simply by observing the output and how it differs from the intended functionality. This is called black box testing. While this method is simple and doesn't require any additional tools, it is also very limited, as no knowledge of the inner workings of the microcontroller is available.

As mentioned previously, most IDE's contain debuggers that are able to run the code step by step at the users control while keeping track of variable and register values so the user can learn exactly what point the program behaves differently than as intended. This is a form of white box testing, and provides much more information that can be used to deduce the cause of a faulty program.

CONCLUSION:

Microcontrollers are a practical, affordable, and flexible solution to many challenges of circuit design and modern control systems. While details such as the

software used to program the microcontroller can vary, the thought process used to program them is universal. Always keep in mind that online recourses such as the ones listed on the next page are available to you should you require more information.

RESOURCES:

http://www.instructables.com/id/How-to-choose-a-MicroController/

http://arduino.cc/en/Tutorial/HomePage

http://stackoverflow.com/questions/78744/how-to-start-programming-a-microcontroller