Morse Code Tutor -
from the ground up

Part 1: Introduction

Bruce E. Hall, W8BH

Would you like to build your own Morse Code tutor for about $20? More importantly, would you like to be able to program it yourself using the Arduino IDE? If so, this article may be for you.

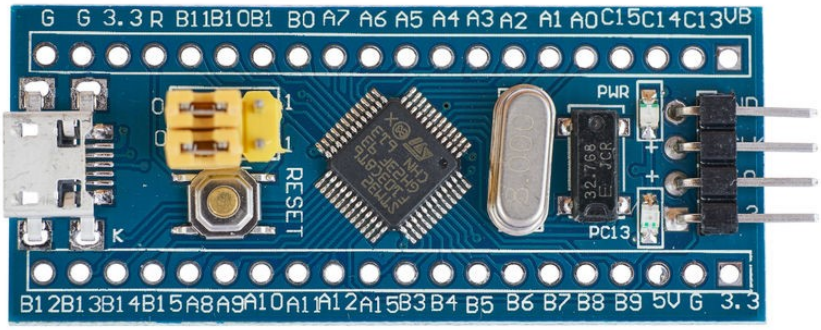
Every year I attend the ARCI Four Days in May conference. It is a wonderful gathering of ham radio enthusiasts who like to build. One of this year's talks was by Jack Purdum, W8TEE. He used a very inexpensive microcontroller to build a device that can send and receive Morse code. Being the ~~cheap~~ frugal ham that I am, I was hooked. I had to try it at home.

The following day I ordered the parts, and soon enough I had all of the components in my hands. I breadboarded the circuit, downloaded Jack's software code, and crossed my fingers as I pressed the 'Build' button in the IDE. Will the software work? Will I see the screen light up and hear Morse code?

No, I did not. It didn't work. After an hour of poking and prodding I gave up. How could I fix it if I didn't understand it? Sure, I understood the schematic and I knew how to compile and upload code, but that wasn't enough to get this project working.

The quickest solution would have been to ask for help. But since I am a man who never asks for directions, that wouldn't do. It was time to figure this out on my own.

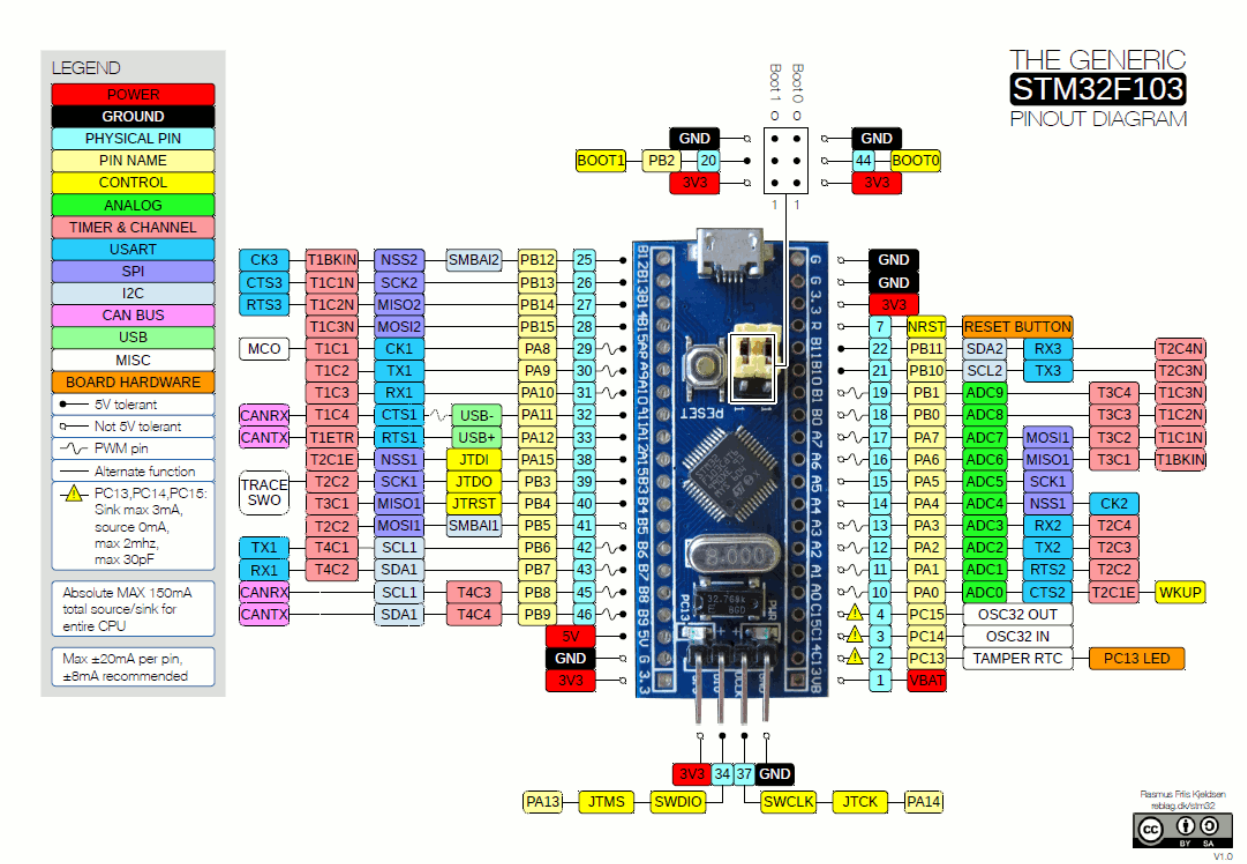
The Blue Pill.



The microcontroller used in this project is the STM32F103C from ST Microelectronics. It is mounted on a very inexpensive board that is curiously named the "Blue Pill" (No, not that blue pill. This blue pill). Search eBay for suppliers and pick up a

few. Current prices are $2 to $4 depending on shipping.

The STM32 is a very capable device. It contains an ARM Cortex-M3 core running at 72 MHz. This is six times faster than the standard Arduino! In addition, there is 128K of onboard flash memory (4x Arduino) and 20K of RAM (10x Arduino), giving you plenty of room for your code and variables.



The Blue Pill has a full complement of digital and analog I/O pins; more than enough for this project.

So, with all of these great features, what's the catch? There isn't one, really, but there are a few details to consider. First, this is not an Arduino, so if you want to use it like one, you'll need to configure the IDE to accept it. There are YouTube videos on how to use the Blue Pill in an Arduino environment; I am using the stm32duino package at dan.drown.org. Second, the board quality suffers: the USB connectors are not soldered well, and the on-board voltage regulator is weak. And finally, there are 3 different ways to upload code to the Blue Pill; I chose the STLINK dongle since it doesn't require moving the bootloader jumpers.

For this tutorial I am assuming that you are familiar with the Arduino IDE and how to connect a programming cable to the Blue Pill. I assume that you can breadboard and know how to power the device. This is not a tutorial on how to "build" it -- The connections between the components are straightforward. I will explain them as we add each new piece of hardware.

Top-Down vs. Bottom-up

Do you like to solve problems by beginning in abstract terms, and then fleshing out the details? This top-down approach works for some people but not for me. With programming (and building radios), I like starting with the smallest of details and solving the tiny problems before attempting anything complex. I gradually acquire the building-blocks and working tools that I can confidently use later. So how in the world should we start?

Hello World.

The first step is to confirm that the microcontroller is working and that you are able to upload code to it. The usual standard is a simple program that displays "Hello World". The microcontroller version of Hello World is a blink sketch which flashes an onboard LED. Here is the complete sketch:

```
#define LED   PC13

void setup() {
  pinMode(LED, OUTPUT);
}

void loop() {
  digitalWrite(LED, LOW);            // turn the LED on
  delay(500);                        // wait half a second
  digitalWrite(LED, HIGH);           // turn the LED off
  delay(500);                        // wait half a second
}
```

Upload the code. If the LED blinks, congratulations! Continue to Part 2 and do some Morse.

Overview.

I am presenting material in the same order as I built the device myself. We start simple and add complexity bit by bit.

1. Introduction (this text)
2. Simple Morse
3. More Morse with paddles
4. Adding an LCD Display
5. The Rotary Encoder
6. The Menu System
7. A Simple Matter of Programming
8. Add an SD card

Each topic builds on the preceding steps, and includes a discussion of the hardware and software involved. My github account houses the source code for each topic, including this one.

By the end, you will have an inexpensive but useful device that will help you learn to send and receive Morse code. You will understand how the software works and be able to customize it. Who knows, you might inspire others to learn the Code, too!

73, Bruce.