Programming micro-controllers with godafoss

Wouter van Ooijen

February 9, 2020

1 Introduction

- target system
- target micro-controller
- download
- compile/build versus run
- C++ versus godafoss
- electronics 'on the fly'
- format, whitespace

2 First steps

2.1 Blink

The first thing to do on micro-controller system is to blink a LED. It proves that your system is working and that you are a genius. So let's look at the code for a blink-a-led application in godafoss. The line numbers are for reference, they are not part of the real source code.

```
[ 1] #include "godafoss.hpp"
[ 2]
[ 3] int main(){
[ 4]    godafoss::blink<
[ 5]       godafoss::target<>::led,
[ 6]       godafoss::target<>::timing::ms< 500 >
[ 7]    >();
[ 8] }
```

The first line makes the godafoss features available for use in the code. It must be on a line by itself, and it is the first line of every godafoss application.

The int main() on line 3 upto the matching on line 8 are required by the C++ language to identify what the application does when it is run. You will find such a 'main function' in every C++ application. The int before main is required by the C++ language because it is useful in C++ applications that run on a PC. For a micro-controller application it has no meaning, but it must still be present. Just pretend that it is not there.

The lines 4 .. 7 are the interesting part. The godafoss::blink phrase means 'the blink within the godafoss namespace'. A namsepace is a bit like a family. There are many persons named Martin in the world, so to identify a specific Martin you say something like Martin King: the person named Martin within the King family. (Assuming there is only one King family, and only one Martin in that family.) In C++ you would write this as King::Martin. The godafoss namespace has a blink function, which is used here to do the blinking.

We want to blink the LED that is on the target board. This LED is connected to a specific pin of the micro-controller on which the application will run. To have the desired effect, the blink function must be configured to do the blinking on the correct micro-controller pin. The godafoss namespace has a target¡¿ in it that contains the features available on the target system. For each target that has an LED, godafoss provides this pin as godafoss::target¡¿::led, so this is the first item used to configure the blink function. The angle brackets ¡¿ after target are required because a target can be configured. Specifying ¡¿ means that the default(s) are used.

The next thing that must be configured is how long a blink period (the sum of the on and off times) must be. This is specified via the timing service that is provided by each godafoss target. In this example 500 ms (half a second) is specified (to be divided evenly among the on and the off period), so the blinking will be at a rate of 2 per second (2 Hertz).

The fully configured godafoss::blink doesn't need any run-time information to do its work, so the () on line 7 specifies that it is called, which means that its code is executed. As configured godafoss::blink will blink forever, so that is all there is to blinking the LED on a target board.

The next step is to blink an LED that is external to the board. This requires the LED to be connected properly, and the blink function must be configured with the micro-controller pin that it is connected to. A suitable series resistor is needed to limit the current, otherwise the LED or the micro-controller might be damaged. In nearly all cases, a 1 kOhm (1000 Ohm) resistor will do. Such a resistor has a color code of brown-black-red or brown-black-black-brown, in both cases followed by other rings that are not important to us.

image

A resistor is symmetrical (apart from aesthetics you can swap its two leads). An LED however is not: it must be connected the right way round, or it will not work.

image

Use a solderless breadboard and plug-in wires to connect the pin marked 2 or D2 to the resistor, the other side of the resistor to the LED pin at the round side of the LED, and the other (flat) side of the LED to the ground (marked GND on most boards).

image image

Provided that the correct Arduino target board is selected in the development environment, the godafoss::target namsepace will contain the pins as d0, d1, etc and a0, a1, etc. The LED is connected to the d7 pin, so that pin must be used to configure the blink function.

```
[ 1] #include "godafoss.hpp"
[ 2]
[ 3] int main(){
[ 4]    godafoss::blink<
[ 5]       godafoss::target<>::d2,
[ 6]       godafoss::target<>::timing::ms< 500 >
[ 7]    >();
[ 8] }
```

All this godafoss and godafoss::target is a bit longwieldy and repetitive. Aliases can be created via 'using' statements to make the application code itself (the main) shorter and easier to read, at the cost of a few extra lines at the top.

```
[ 1] #include "godafoss.hpp"
[ 2]
[ 3] using gf     = godafoss;
[ 4] using target = gf::target<>;
[ 5] using timing = gf::timing;
[ 6]
[ 7] int main(){
[ 8]    gf::blink<
[ 9]       target::d2,
[10]       timing::ms< 500 >
[11]    >();
[12] }
```

One step further, it is often considered good practice to separate the 'flexible' (easily changeable) part of the application from the details of how how it works. In the case of the blink application, the pin that is used and the blinking period can be lifted out of the blink call.

```
[ 1] #include "godafoss.hpp"
```

```
[ 2]
[ 3] using gf     = godafoss;
[ 4] using target = gf::target<>;
[ 5] using timing = gf::timing;
[ 6]
[ 7] using led    = target::d2;
[ 8] using period = timing::ms< 500 >;
[ 9]
[10] int main(){
[11]    gf::blink< led, period >();
[12] }
```

This separation might seem unnecessary for such a small and simple program, but it pays off for a larger application, where the led and the period are likely to be used at multiple locations in the source.

2.2 Morse

Morse is a way to encode letters and digits in a sequence of short and long beeps (or flashes of light). A famous example is the sequence that was used as a distress (call for help) signal: three short beeps, three longer beeps, three short beeps. Interpreted as morse code this can be read as SOS, which was in popular culture read as Save Our Ship or (later) Save Our Souls. (Note: the real morse code for the letters SOS would be three short beeps, *a pause*, three long beeps, *a pause*, three short beeps. The pauses are needed to separate the letters.)

Beeping is much like blinking an LED, only faster. The so-called middle A note, commonly used for tuning, has a frequency of 440 Hz, which means that the full period is 1/440 s = 2.27 ms. Hence the configuration parameter for blink to produce a middle A is half that value: 2270 us.

To physically produce a beep, a speaker must be connected to the target. As for the LED, a series resistor must be used to limit the current. Like a resistor, a speaker is symmetrical (swapping the connections has no effect). A 1k resistor can be used to get a soft sound, 100 Ohm (brown-black-brown or brown-black-black-black) will produce a louder sound.

image

Changing the 500 ms used in the blink applications to 2270 us produces a continuous 440 Hz tone.

```
[ 1] #include "godafoss.hpp"
[ 2]
[ 3] using gf     = godafoss;
[ 4] using target = gf::target;
[ 5] using timing = gf::timing;
```

```
[ 6]
[ 7] using led    = target::d3;
[ 8] using period = timing::us< 1120 >;
[ 9]
[10] int main(){
[11]    gf::blink< led, period >();
[12] }
```

image frequency meter 440 Hz

The blink function that was used so far blinks forever. A second version of this function exists, that takes a second duration parameter, to specify how long the blinking (or in this case, the beeping) must be done.

- beep - blink an amount of time - different on and off periods

3 Scale factors

4 Resistor color codes

5 Arduino pin names

6 Using a solderless breadboard