BILL WONG | Embedded/Systems/Software Editor
bill.wong@penton.com

Running Ada 2012
On the Cortex-M4

I like to think I write good code, and I've used C and C++ almost since their inception. I admit to incorporating more than one unwanted bug into C applications that were eliminated after sometimes tedious diagnostic sessions. Almost every new microcontroller released has a free C/C++ compiler toolchain associated with it.

Unfortunately, C is very unforgiving, and C++ is only a little better. But they are the mainstay for embedded programmers these days. That's one reason why I have been waiting for AdaCore's delivery of its Ada 2012 toolchain for Arm's Cortex-M platform. It is a free download at libre.adacore.com.

The Cortex-M is the main low-end, 32-bit microcontroller utilized by almost all major microcontroller vendors that have adopted the Arm architecture. Vendors like Green Hills Software, Atego, and Adacore have supported the Cortex platform in the past but with earlier Ada standards. Ada 2012 includes a range of new features including contracts.

ROLLING UP MY SLEEVES

I had already tried out a version of AdaCore's GNAT Programming Studio (GPS). It generated applications that would run on Linux on the BeagleBone based on the Texas Instruments Cortex-A8 platform. The new toolchain targets bare metal, which is needed for many applications.

Setup was easy since GPS was already installed. Setting up the ST-Link debug interface to the STM32 board actually took more time. It was then a matter of running through the flashing LED demo. I included a snippet of code from a flashing LED program to highlight some of the advantages that Ada 2012 provides *(see the listing)*.

Even if you haven't used Ada, you should be able to get an idea of what's going on. In particular, the task body definition highlights the built-in multitasking support. Also note the use of the "modular" (unsigned) data type for Index that limits the addressing of the Pattern array. Unlike C, there is no need to check the result of Next_LED when it is updated. Additionally, I prefer the more verbose if/then/else conditional expression of Ada to the C/C++ ?: combination. I have programmed in APL, and its one-liners were neat but typically indecipherable. C and C++ code can get this way too.

Ada has several benefits over C and C++ . But it has drawbacks as well, such as availability. For ARM Cortex microcontrollers, this is no longer an issue. Developers can take advantage of all the Ada 2012 features, from generics to multitasking.

I would encourage anyone wanting to write bulletproof code for embedded applications to check out the AdaCore/STM32 combination. It is inexpensive and very functional. The ST-Link support also allows it to work with platforms like the STM32F401 Nucleo. I tried it on my Nucleo as well.

The Heartbleed bug is just one of many reminders of how one range check error can wreak havoc. C does not do it, but Ada does check—and not just on array access, by default. Ada will not eliminate all bugs from your code, but it does make it a lot harder to create them. ED

```
with LEDs;        use LEDs;
with Button;      use Button;
with Ada.Real_Time; use Ada.Real_Time;
package body Driver is
  type Index is mod 4;
  Pattern : constant array (Index) of User_LED := (Orange, Red, Blue, Green);
  task body Controller is
    Period     : constant Time_Span := Milliseconds (75);  -- arbitrary
    Next_Start : Time := Clock;
    Next_LED   : Index := 0;
  begin
    loop
      Off (Pattern (Next_LED));
      Next_LED := Next_LED +
       (if Button.Current_Direction = Counterclockwise then -1 else 1);
      On (Pattern (Next_LED));
      Next_Start := Next_Start + Period;
      delay until Next_Start;
    end loop;
  end Controller;
end Driver;
```