MuntsOS Embedded Linux

Application Note #7: Ada LED Flash Example Using Alire

Revision 2
4 September 2021

by Philip Munts
President, Munts Technologies
http://tech.munts.com

Introduction

This application note describes how to create, build, and run an Ada program to flash an LED on a target computer running ***MuntsOS Embedded Linux***, using the ***Alire*** (Ada LIbrary REpository) development environment (https://alire.ada.dev).

Prerequisites

The ***MuntsOS Embedded Linux*** software development environment must be installed on a desktop Linux system (AppNote #1 or AppNote #2).

The **`alr`** command line tool for ***Alire*** must be installed on the desktop Linux system (following the procedure specified at https://alire.ada.dev/docs/#installation, *or* from the Munts Technologies Debian package repository at http://repo.munts.com/).

MuntsOS Embedded Linux must be installed on the target computer (AppNote #3).

Target Platform Hardware

The target platform for the purposes of this application note consists of a Raspberry Pi Zero Wireless mounted in a Zebra Zero Plus Breadboard case. The orange and black jumper wires connect **`+3.3V`** and **`GND`** on the Raspberry Pi expansion header to the breadboard power rails. The yellow jumber connects **`GPIO26`** to a 1K ohm current limiting resistor and an LED.

Test Program Source Code

Available for download at: http://git.munts.com/muntsos/doc/.blinky/blinky.adb

```
WITH Ada.Text_IO; USE Ada.Text_IO;

WITH GPIO.libsimpleio;
WITH RaspberryPi;

PROCEDURE blinky IS

  LED : GPIO.Pin;

BEGIN
  New_Line;
  Put_Line("MuntsOS Ada LED Test");
  New_Line;

  -- Configure a GPIO output to drive an LED

  LED := GPIO.libsimpleio.Create(RaspberryPi.GPIO26, GPIO.Output);

  -- Flash the LED forever (until killed)

  Put_Line("Press CONTROL-C to exit");
  New_Line;

  LOOP
    LED.Put(NOT LED.Get);
    DELAY 0.5;
  END LOOP;
END blinky;
```

Exercise

This example exercise demonstrates how to create an Alire Ada program project for ***MuntsOS Embedded Linux***, compile it, and run it on the test platform hardware.

Step 1: Prepare the **`blinky`** program project directory by running the following commands on the desktop Linux development system:

```
mkdir blinky
alr init --bin blinky
cd blinky
sed -i '/STYLE_CHECKS/ s/enabled/disabled/' blinky.gpr
alr -n with libsimpleio
cp /usr/local/gcc-arm-linux-gnueabihf-muntsos-RaspberryPi1-
crosstool/share/gpr/RaspberryPi1.cgpr default.cgpr
```

The above commands will:

- Create and initialize an Alire project directory.
- Disable GNAT style checks (**`blinky.adb`** does not conform to the GNAT source style).
- Add a reference to the **`libsimpleio`** library crate to the project.
- Copy **`RaspberryPi1.cgpr`** to **`default.cgpr`** to enable cross-compiling.

Step 2: Download the source program file **`blinky.adb`**:

```
wget -O src/blinky.adb
http://git.munts.com/muntsos/doc/.blinky/blinky.adb
```

Step 3: Build the **`blinky`** project:

```
alr build
```

Step 4: Copy **`blinky`** to the test platform:

```
scp bin/blinky root@snoopy:.
```

Step 5: Run the test program on the test platform:

```
ssh root@snoopy
./blinky
```

The LED should begin flashing once a second, until you press **`CONTROL-C`**.