MuntsOS Embedded Linux

Application Note #7: Ada LED Flash Example Using Alire

Revision 1
27 July 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 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/debian10).

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 Ada program for ***MuntsOS Embedded Linux*** from an Alire project crate, compile it, and run it on the test platform hardware.

Step 1: Prepare the **`blinky`** project:

```
alr get muntsos_blinky
```

This will download the Alire project crate named **`muntsos_blinky`** and extract it to create a project directory named **`muntsos_blinky_n.n.n_hhhhhhhh`**. The **`n.n.n`** will be replaced with a version number like **`1.0.0`** and the **`hhhhhhhh`** will be replaced with a hexadecimal hash value.

Step 2: Build the **`blinky`** project:

```
cd muntos_blinky_n.n.n_hhhhhhhh
alr build
```

Step 3: Copy **`blinky`** to the test platform:

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

Step 4: 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`**.

Changing the Target Platform

The **`muntsos_blinky`** project stored in the Alire crate repository is preconfigured to build for a Raspberry Pi 1 target platform. The target platform is selected by the symbolic link named **`default.cgpr`** in the project directory. If you wish to build for a different platform, just change **`default.cgpr`** to point to a different ***MuntsOS Embedded Linux*** cross-toolchain. You can find the names of available **`.cgpr`** files using the following command:

```
find /usr/local/*muntsos* -name '*.cgpr'
```

For example, to run **`blinky`** on a Raspberry Pi 3, use the following command to switch to the proper cross-toolchain:

```
ln -s -f
/usr/local/gcc-aarch64-linux-gnu-muntsos-crosstool/share/gpr/Raspberr
yPi3.cgpr default.cgpr
```