

MuntsOS Embedded Linux

Application Note #21: Ada Extension Programs

**Revision 1
16 July 2025**

**by Philip Munts
dba Munts Technologies
<http://tech.munts.com>**

Introduction

This Application Note describes, with a worked example, how to write, deploy and install a self-contained **MuntsOS Embedded Linux** extension program written in the Ada programming language.

Extension Programs are one of three ways of extending a MuntsOS Embedded Linux installation (the other ways two are extension packages installed from `/boot/extensions` and tar balls unpacked from `/boot/tarballs`).

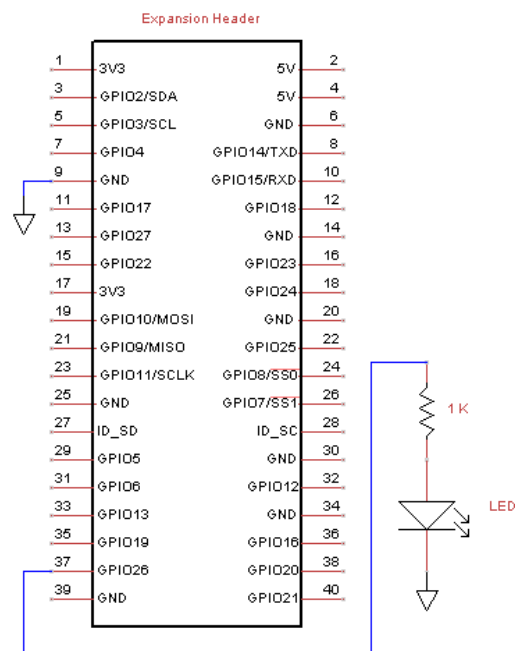
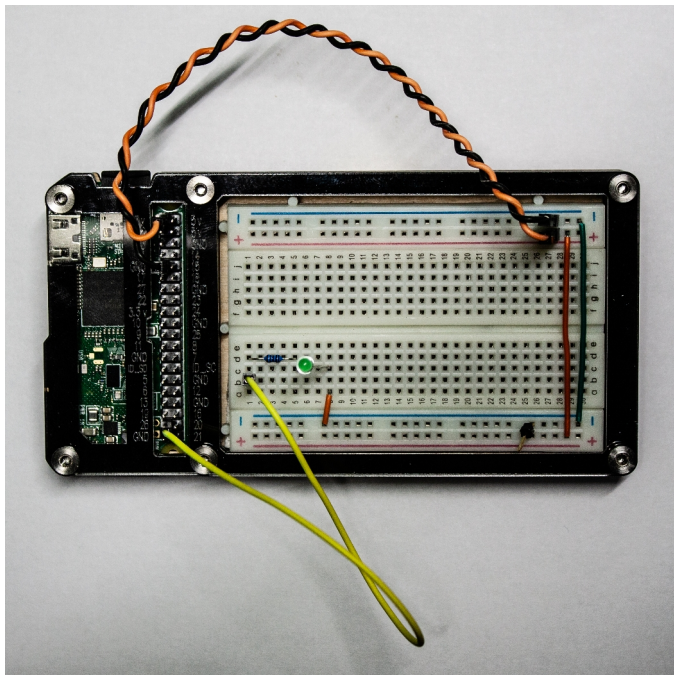
Prerequisites

The **MuntsOS Embedded Linux** software development environment must be installed on your Linux development computer ([AppNote #1](#) or [AppNote #2](#)).

The `alr` command line tool for [Alire](#) must be installed on your Linux development computer (either by following the procedure specified at <https://alire.ada.dev/docs/#installation> or by installing from the **Munts Technologies Debian Package Repository** at <https://repo.munts.com/debian12>).

MuntsOS Embedded Linux must be installed on your target computer ([AppNote #3](#)).

Target Computer Hardware



The target computer for the purposes of this application note consists of a [Raspberry Pi Zero 2 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 jumper connects `GPIO26` to a 1K ohm current limiting resistor and an LED.

Ada Extension Program Template

https://git.munts.com/muntsos/examples/ada/programs/extension_program_template.adb.

```
1  WITH Ada.Text_IO; USE Ada.Text_IO;
2  WITH Debug;
3  WITH libLinux;
4  WITH Watchdog.libsimpleio;
5
6  PROCEDURE extension_program_template IS
7
8      err : Integer;
9      wd  : Watchdog.Timer;
10
11      -- Add your application specific data declarations here
12
13  BEGIN
14      IF Debug.Enabled THEN
15          New_Line;
16          Put_Line("Starting Ada Extension Program");
17          New_Line;
18      ELSE
19          Put("Starting Ada Extension Program");
20
21          -- Run as background process
22
23          libLinux.Detach(err);
24
25          -- Create a watchdog timer device object
26
27          wd := Watchdog.libsimpleio.Create;
28          wd.SetTimeout(5.0);
29      END IF;
30
31      -- Add your application specific initialization code here
32
33      LOOP
34          -- Add your application specific event handling code here
35
36          IF NOT Debug.Enabled THEN
37              wd.Kick;
38          END IF;
39      END LOOP;
40  END extension_program_template;
```

Source Code Annotations

Line 1: Pulls in package `Ada.Text_IO`.

Line 2: Pulls in package `Debug`, which provides output subprograms that are qualified by the `DEBUGLEVEL` environment variable. If `DEBUGLEVEL` is set to an integer value greater than zero, output to `stderr` is enabled.

Line 3: Pulls in package `libLinux`, from which the procedure `Detach` will be used. `Detach` disconnects from the program's from its controlling terminal and switches to the program to running as a background procoess.

Line 4: Pulls in package `Watchdog.libsimpleio`, which provides hardware watchdog timer services.

Lines 8-9: Declares an error return variable for `libLinux.Detach` and a watchdog timer object.

Line 11: Your application specific data declarations can go here, including one or more Ada tasks that will do all the real work.

Lines 15-17: Startup banner for debug mode.

Line 19: Startup banner for normal mode.

Line 23: `libLinux.Detach` detaches the program from its controlling terminal and continues execution as a background process.

Lines 27-28: Starts the watchdog timer and sets its period to 5 seconds. All **MuntsOS Embedded Linux** kernels configure the watchdog timer for unstopable mode. If the program hangs for any reason or terminates for any reason (including by an unhandled exeption or signal), the watchdog timer will issue a hardware reset after 5 seconds and restart the system.

Line 31: Your application initialization code can go here. This may include starting one or more Ada tasks that do all the real work.

Line 33: Beginning of the event loop, which is supposed to run forever.

Line 34: Your application specific event handling code can go here.

Line 37: Resets the watchdog timer for another 5 seconds..

Line 39: End of the event loop.

Create the Ada Extension Program Project

Use the following procedure to create your Ada extension program project using [Alire](#):

```
alr -n init --bin myextension
cd myextension
alr -n with muntsos_aarch64
cp
/usr/local/share/muntsos/examples/ada/programs/extension_program_template.adb src/myextension.adb
```

Now edit `src/myextension.adb` with your favorite Linux text editor:

1. Change the program name from `extension_program_template` to `myextension` with a "find and replace all" operation.
2. Add the following package reference after line 2:

```
WITH GPIO.libsimpleio;
```

3. Add the following package reference after line 3:

```
WITH RaspberryPi;
```

4. Add a declaration for an LED object by adding the following at line 11:

```
LED : GPIO.Pin;
```

5. Add code to initialize the LED object by adding the following at line 31:

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

6. Add code to flash the LED inside the event loop by adding the following at line 34:

```
LED.Put(NOT LED.Get);
DELAY 0.5;
```

Test the Ada Extension Program

1. Build, install temporarily, and run `myextension`:

```
alr build
scp bin/myextension root@snoopy:.
ssh root@snoopy
./myextension
```

The LED should begin flashing once a second.

2. After a while, kill `myextension` on the target computer:

```
killall myextension
```

After 5 seconds, the target computer should reboot.

Permanent Installation

You can install your extension program permanently to the target computer by copying the extension program file to `/boot/autoexec.d`, using commands like the following:

```
alr build
scp bin/myextension root@snoopy:.
ssh root@snoopy
mount -orw /boot
mkdir -p /boot/autoexec.d
mv myextension /boot/autoexec.d
umount /boot/autoexec.d
reboot
```

The **MuntsOS Embedded Linux** startup script `/etc/rc` will load and execute extension programs from either `/boot/autoexec.d` or `/boot/extensions`. The "best practices" recommendation is to install custom extension programs to `/boot/autoexec.d` and only install curated extension programs (*i.e.* using `sysconfig`) to `/boot/extensions`.