Making the Ada Drivers Library

Embedded Programming with Ada

Fabien Chouteau

Embedded Software Engineer at AdaCore

Twitter : @DesChips
GitHub : Fabien-Chouteau
Hackaday.io: Fabien.C
Programming is all about communication
Programming is all about communication

With:

- The compiler
Programming is all about communication

With:

- The compiler
- The other tools (static analyzers, provers, etc.)
Programming is all about communication

With:

- The compiler
- The other tools (static analyzers, provers, etc.)
- Users of your API

Oh, wait. It was me two months ago :(
Programming is all about communication

With:

- The compiler
- The other tools (static analyzers, provers, etc.)
- Users of your API
- Your colleagues
Programming is all about communication

With:

- The compiler
- The other tools (static analyzers, provers, etc.)
- Users of your API
- Your colleagues
- The idiot that wrote this stupid piece of code...
Programming is all about communication

With:

- The compiler
- The other tools (static analyzers, provers, etc.)
- Users of your API
- Your colleagues
- The idiot that wrote this stupid piece of code...
- Oh, wait. It was me two months ago :(

AdaCore
What makes Embedded Programming different?

Every bug costs more:

- More time to investigate
- More time to try a fix
- Potential destruction of hardware
- Updates are difficult

You need more control:

- Low resources (RAM, flash, CPU)
- Interaction with the hardware
- Real-Time constraints
Embedded Programming with Ada
Servo motor example
Servo motor example

2.5 ms

+90 °
Servo motor example

2.5 ms

+90 °
procedure Set_Angle (Angle : Integer);
-- Set desired angle for the servo motor
--
-- @param Angle: Desired rotation angle in degree.
-- Please do not use a value above 90 or below -90!

procedure Set_Angle (Angle : Integer);
type Servo_Angle is range -90 .. 90;
-- Servo rotation angle in degree

procedure Set_Angle (Angle : Servo_Angle);
-- Set desired angle for the servo motor
Set_Angle (100);

warning: value not in range of type "Servo_Angle"
warning: "Constraint_Error" will be raised at run time
procedure Set_Angle_Double (X : Servo_Angle) is
begin
    Set_Angle (X * 2);
end Set_Angle_Double;

Set_Angle_Double (80);

servo_driver.adb:27:4: high: precondition (range check) failure on call to servo_driver.set_angle_double: requires X in -45..45
Phase 1 of 2: generation of Global contracts ...

servo_driver.adb:42:04: error in inlined body at line 23

servo_driver.adb:42:04: value not in range of type

"Servo_Angle" defined at line 7

servo_driver.adb:42:04: "Constraint_Error" would have been raised at run time
(gdb) catch exception
Catchpoint 1: all Ada exceptions
(gdb) run

Catchpoint 1, CONSTRAINT_ERROR
   (servo_driver.adb:23 overflow check failed)
procedure Set_Angle_Catch (X : Servo_Angle) is
begin
    Set_Angle (X * 2);
exception
    when Constraint_Error =>
        Put_Line ("Well, that was close");
end Set_Angle_Catch;
procedure Last_Chance_Handler is
begin

   --  Oops, there's something wrong

   Reset_The_Board;

end Last_Chance_Handler;
*COUGH* COME CLOSER... *SPLUTTER* LET THE WORLD KNOW MY FINAL WORDS...

Yolo

SEE, THIS KIND OF SHIT IS EXACTLY WHY I'M MURDERING YOU.

\[^1\] cyanide and happiness
procedure Set_Angle (Angle : Servo_Angle)
    with Pre => Initialized;
-- Set desired angle for the servo motor

function Initialized return Boolean;
-- Return True if the driver is initialized

procedure Initialize
    with Post => Initialized;
-- Initialize the servo motor driver
procedure Plop (Ptr : not null Some_Pointer);
Hardware mapping

-- High level view of the type

type Servo_Angle is range -90 .. 90

-- Hardware representation of the type

    with Size => 8,
    Alignment => 16;
Memory mapped registers

- Flash: 0x00000000
- RAM
- Peripherals: 0xFFFFFFFF

AdaCore
**Hardware mapping**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Sense</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Sense**: Pin sensing mechanism

- **0**: Disabled
- **2**: Sense for high level
- **3**: Sense for low level
#define SENSE_MASK   (0x30)
#define SENSE_POS    (4)
#define SENSE_DISABLED (0)
#define SENSE_HIGH   (2)
#define SENSE_LOW    (3)

uint8_t *register = 0x80000100;

// Clear Sense field
*register &= ~SENSE_MASK;

// Set sense value
*register |= SENSE_DISABLED << SENSE_POS;
-- High level view of the Sense field

```ada
type Pin_Sense is
  (Disabled,
   High,
   Low)
with Size => 2;
```

-- Hardware representation of the Sense field

```ada
for Pin_Sense use
  (Disabled => 0,
   High    => 2,
   Low     => 3);
```
-- High level view of the register

```ada
type IO_Register is record
  Reserved_A : UInt4;
  SENSE : Pin_Sense;
  Reserved_B : UInt2;
end record;
```

-- Hardware representation of the register

```ada
for IO_Register use record
  Reserved_A at 0 range 0 .. 3;
  SENSE at 0 range 4 .. 5;
  Reserved_B at 0 range 6 .. 7;
end record;
```
Hardware mapping

Register : IO_Register
  with Address => 16#8000_0100#;

Register.SENSE := Disabled;
<field>
  <name>SENSE</name>
  <description>Pin sensing mechanism.</description>
  <lsb>16</lsb> <msb>17</msb>
  <enumeratedValues>
    <enumeratedValue>
      <name>Disabled</name>
      <description>Disabled.</description>
      <value>0x00</value>
    </enumeratedValue>
  </enumeratedValues>
[...]

github.com/AdaCore/svd2ada
Ravenscar Tasking

A.K.A There’s a mini-RTOS in my language

- Tasks (threads)
- Time handling
  - Clock
  - Delays
- Protected Objects:
  - Mutual exclusion
  - Synchronization between tasks
  - Interrupt handling

---

2 blog.adacore.com/theres-a-mini-rtos-in-my-language
task body My_Task is
    Next_Release : Time;
begin
    -- Set Initial release time
    Next_Release := Clock + Milliseconds (100);

    loop
        -- Suspend My_Task
        delay until Next_Release;

        -- Compute the next release time
        Next_Release := Next_Release + Milliseconds (100);

        -- Do something really cool at 10Hz...
        end loop;
end My_Task;
Making the Ada Drivers Library
Ada Drivers Library

- Firmware library
- Hardware and vendor independent
- 100% Ada
- Hosted on GitHub:
  github.com/AdaCore/Ada_Drivers_Library
Components

I2C, SPI, UART, etc.

AdaCore
Supported components

- Audio DAC: SGTL5000, CS43L22, W8994
- Camera: OV2640, OV7725
- IO expander: MCP23XXX, STMPE1600, HT16K33
- Motion: AK8963, BNO055, L3GD20, LIS3DSH, MMA8653, MPU9250
- Range: VL53L0X
- LCD: ILI9341, OTM8009a, ST7735R, SSD1306
- Touch panel: FT5336, FT6X06, STMPE811
- Module:
  - AdaFruit’s trellis
  - AdaFruit’s Thermal printer
Middleware

- Bitmap drawing
- File System: FAT and ARM semi-hosting
- Log utility
Supported platforms

ARM

RISC-V
STM32F405 Discovery (ARM Cortex-M4F)
STM32F469 Discovery (ARM Cortex-M4F)
STM32F769 Discovery (ARM Cortex-M7F)
OpenMV 2 (ARM Cortex-M4F)
Crazyflie 2.0 (ARM Cortex-M4F)
BBC Micro:Bit (ARM Cortex-M0)
HiFive1 (RISC-V)
What’s next?

TODOs:

- New configuration and build system
- More documentation
- Basic out of the box support of all the Cortex-M devices
- Linux GPIO/I2C/SPI support (on the Raspberry Pi for instance)
- AVR platform
- More component drivers
- USB stack and drivers on the STM32
- Bluetooth Low Energy stack on the Micro:Bit
Getting started demo
Download GNAT Community Edition

For free software developers, hobbyists, and students.

## x86-64 GNU Linux (64 bits)

### GNAT GPL Ada

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>gnat-gpl-2017-x86_64-linux-bin.tar.gz</td>
<td>496.34 MB</td>
<td>May 17 2017</td>
</tr>
<tr>
<td>SHA-1: 9c682a5f1f2f23ec03fe21d77bf4c37a0af599b</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SPARK Discovery

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>spark-discovery-gpl-2017-x86_64-linux-bin.tar.gz</td>
<td>104.06 MB</td>
<td>May 29 2017</td>
</tr>
<tr>
<td>SHA-1: a70d7c7f508ed3ab0ecb4a34fccc1df9a9d9039</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### ARM ELF (hosted on linux)

### GNAT GPL Ada

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>gnat-gpl-2017-arm-elf-linux-bin.tar.gz</td>
<td>548.9 MB</td>
<td>May 17 2017</td>
</tr>
<tr>
<td>SHA-1: 71b5830d0242cf9ab294d8895960f0b0b9c241f</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Download Ada Drivers Library

Ada source code and complete sample GNAT projects for selected bare-board platforms supported by GNAT.

Cloning via HTTPS:

Clone with HTTPS

Use Git or checkout with SVN using the web URL:

https://github.com/AdaCore/Ada_Drivers_LIBRARY

Download ZIP
Some projects using the Ada Drivers Library
Crazyflie 2.0 Flight controller

blog.adacore.com/how-to-prevent-drone-crashes-using-spark
blog.adacore.com/make-with-ada-arm-cortex-m-cnc-controller
blog.adacore.com/writing-on-air
DIY instant camera

blog.adacore.com/make-with-ada-diy-instant-camera
github.com/lambourg/Ada_Bare_Metal_Demos
The Make with Ada Competition

- Embedded software project competition
- Open to everyone
- ~8000 euros in prize
- Stay tuned for the next edition (Twitter @adaprogrammers)
2016 Winner project (Stephane Carrez)

github.com/stcarrez/etherscope
2017 Winner project (Jonas Attertun)

What are you going to make?
- GitHub: github.com/AdaCore/Ada_Drivers_Library
- Twitter: @AdaProgrammers