

Alternative languages for safe and secure RISC-V programming

Fabien Chouteau

Embedded Software Engineer at AdaCore

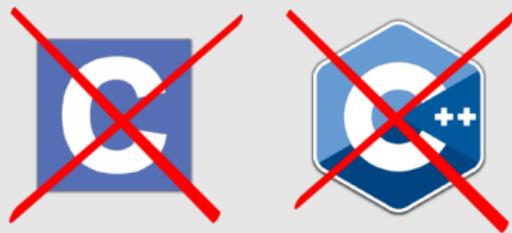
 Twitter : @DesChips

 GitHub : Fabien-Chouteau

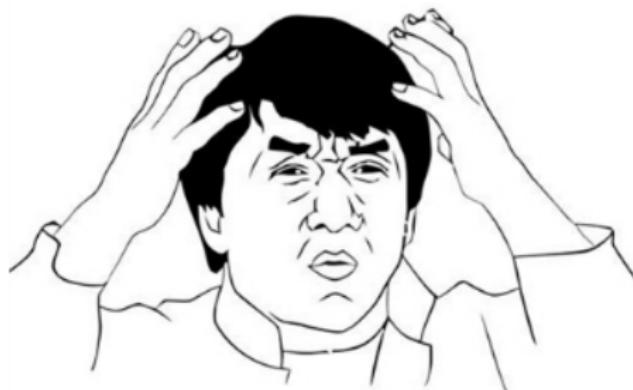
 Hackaday.io: Fabien.C

What do I mean by “alternative”?

What do I mean by “alternative”?



BUT WHY?





Ada
2012



SPARK





Ada
2012



SPARK



Ada and SPARK

- Designed for Safety and Security
- Powerful means of specification
- Strong type checking
- Object Oriented Programming
- Concurrent programming features
- Generic templates
- Encapsulation
- Hierarchical program composition / programming-in-the-large

Programming is all about communication

With:

- The compiler
- The other tools (static analyzers, provers, etc.)
- Users of your API
- Your team
- Yourself in a couple weeks...

Example



Specifications

```
type Servo_Angle is new Float range -90.0 .. 90.0
--  Servo rotation angle in degree

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

Contracts

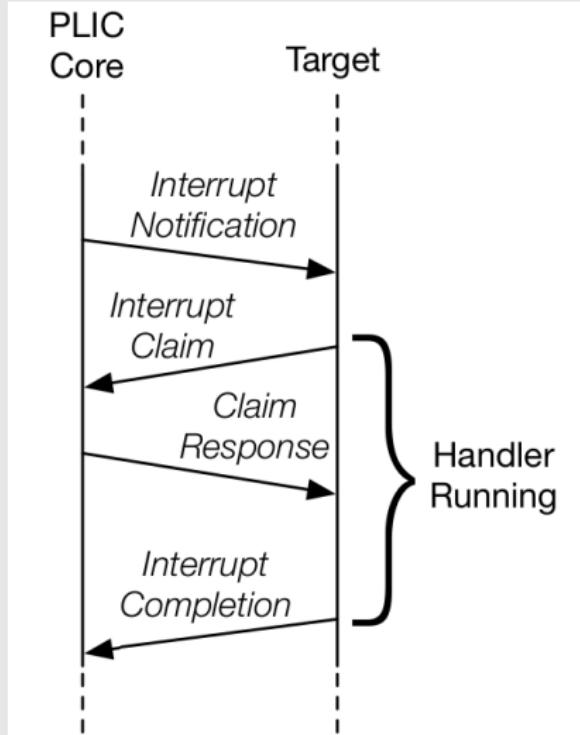
```
type Stack is tagged private;

function Empty (S : Stack) return Boolean;
function Full (S : Stack) return Boolean;

procedure Push (S : in out Stack; Val : Integer)
  with Pre => not S.Full,
       Post => not S.Empty;

procedure Pop (S : in out Stack; Val : out Integer)
  with Pre => not S.Empty,
       Post => not S.Full;
```

Example: Writing an API for the RISC-V PLIC



Example: Writing an API for the RISC-V PLIC

```
Max_Interrupt : constant := 15;

type Any_Interrupt_ID is
  range 0 .. Max_Interrupt + 1;

No_Interrupt : constant Any_Interrupt_ID := Any_Interrupt_ID'Last;
--  Special value indicating no interrupt

subtype Interrupt_ID is
  Any_Interrupt_ID range 0 .. Max_Interrupt;
```

Example: Writing an API for the RISC-V PLIC

```
function Claimed return Any_Interrupt_ID  
  
with Ghost;  
  
--  Return the last claimed interrupt
```

Example: Writing an API for the RISC-V PLIC

```
function Claim return Any_Interrupt_ID

with Pre => Claimed = No_Interrupt,

      Post => Claimed = Claim'Result;

-- Claim an interrupt on the PLIC for the current hart and
-- return its ID. Return No_Interrupt if there was no pending
-- interrupts for the hart when the claim was serviced.
```

Example: Writing an API for the RISC-V PLIC

```
procedure Complete (Interrupt : Interrupt_ID)

  with Pre  => Claimed /= No_Interrupt
                and then
      Interrupt = Claimed,

  Post => Claimed = No_Interrupt;

--  Signal an interrupt completion for the current hart
```

Checks

- At run-time
 - Checks inserted in the code
 - For debug or testing
- At Compile time
 - Compiler
 - Static analyzer
 - Formal verification (SPARK)

Functional Safety

Hardware mapping

```
-- High level view of the type
type Servo_Angle is new Float range -90.0 .. 90.0

-- Hardware representation of the type
with Size      => 32,
     Alignment => 16;
```

Hardware mapping

```
-- High level view of the Sense field
type Pin_Sense is
  (Disabled,
   High,
   Low)
with Size => 2;

-- Hardware representation of the Sense field
for Pin_Sense use
  (Disabled => 0,
   High      => 2,
   Low       => 3);
```

Hardware mapping

```
-- High level view of the register
type IO_Register is record
    Reserved_A : UInt4;
    SENSE      : Pin_Sense;
    Reserved_B : UInt2;
end record with Size => 32;

-- Hardware representation of the register
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

```
#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;
```

Hardware mapping

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

```
Register.SENSE := Disabled;
```

SVD -> Ada

```
<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>
  [...]
```

github.com/AdaCore/svd2ada

Interfacing with C / Assembly

```
with Interfaces.C; use Interfaces.C;

[...]

function My_C_Function (A : int) return int
  with Pre => A /= 0;

pragma Import (C, My_C_Function, "my_c_function");

function My_Ada_Function (A : int) return int;
pragma Export (C, My_Ada_Function, "my_ada_function");
```

Where Ada/SPARK is used?



Avionics



Defense



Rail



Space

Emerging domains



Automotive



Security

- Software development eco-system:
 - GNAT: Ada compiler part of GCC
 - IDEs
 - Code Coverage
 - Static analysis
 - Formal verification
 - Simulink code generator
- Open-source
- Ada, SPARK, C, C++
- Dozens of platforms (including RISC-V :)
- Front line support
- DO-178, EN 50128, ISO 26262

AdaCore joins the RISC-V foundation

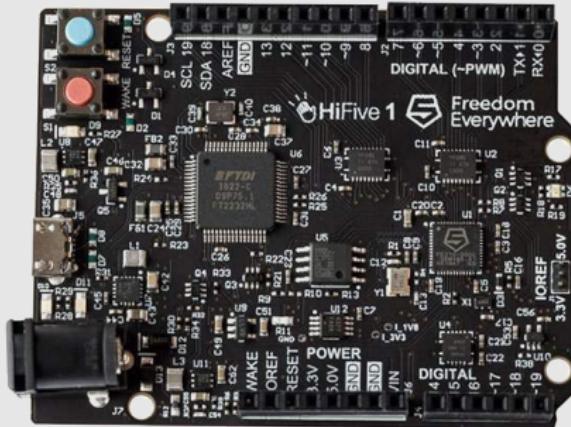
AdaCore



AdaCore

Getting started

Hardware



Download and install the tools: adacore.com/community

Download GNAT Community Edition

For free software developers, hobbyists, and students.

RISC-V ELF (hosted on linux64)

GNAT Community

[README.txt](#) 2.1 KiB Jul 10 2018
SHA-1: f48a0f0edd581f60fd35023ee358e7d3be4f0d8c

[gnat-community-2018-20180524-riscv32-elf-linux64-bin](#) 137.2 MiB May 28 2018
SHA-1: e2dfca056b3d427f26c1e4337cbe25185c49ebf5

x86-64 GNU Linux (64 bits)

GNAT Community

[README.txt](#) 2.1 KiB Jul 10 2018
SHA-1: f48a0f0edd581f60fd35023ee358e7d3be4f0d8c

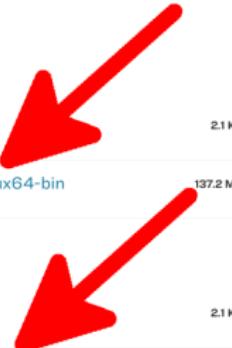
[gnat-community-2018-20180528-x86_64-linux-bin](#) 430.8 MiB May 29 2018
SHA-1: 10360eb85955d40f340f67244fe8415cb0877ffcc

ARM ELF (hosted on linux64)

GNAT Community

[README.txt](#) 2.1 KiB Jul 10 2018
SHA-1: f48a0f0edd581f60fd35023ee358e7d3be4f0d8c

[gnat-community-2018-20180524-arm-elf-linux64-bin](#) 199.6 MiB May 28 2018



Download Ada Drivers Library

This screenshot shows the GitHub repository page for AdaCore/Ada_Drivers_Library. The page includes a header with navigation links like 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the header, there's a search bar and a main content area for the repository. The repository details show 1,179 commits, 22 branches, 0 releases, and 15 contributors. A green progress bar indicates the repository is 100% complete. The main list of files and commits is visible, with a red arrow pointing to the 'Clone with HTTPS' section on the right.

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

1,179 commits 22 branches 0 releases 15 contributors BSD-3-Clause

Branch: master New pull request Create new file Upload files Find file Clone or download

File / Commit	Description	Date
.gitignore	Use new GPRbuild attribute: Create_Missing_Dirs	7 months ago
boards	robust, safer version of GPIO_PIO	5 months ago
components	SGTL5000: Fix some typos	3 months ago
docs	docs/filesystem.md: Add documentation for directory handling	3 months ago
examples	Examples: Bring back the blinky and serial examples for the STM32F4 d...	2 months ago
hal	HAL.SDMMC: Add single and multiple block write cmd definition	5 months ago
middleware	File_IO: Improve error handling	2 months ago
scripts	Examples: Bring back the blinky and serial examples for the STM32F4 d...	2 months ago
testsuite	Monitor.Block_Drivers: Show data size	5 months ago
arch	add convenience macro to conditionally enable a pin	3 months ago
pat-rogers Merge pull request #129 from AdaCore/add_gpio_driver	...	3 months ago

Clone with HTTPS Use SSH
Get Git or checkout with SVN using the web URL.
https://github.com/AdaCore/Ada_Drivers_L

Download ZIP

Keep the door open

What has already been done

- Open specs and documentation
- RISC-V support in open-source tools:
 - Compilers (GCC, LLVM)
 - Debuggers (Gdb, openocd)
 - Simulators (QEMU)

Challenges

- Complexity of extension combinations
RV(32|64|128) | M A C B [F|D|Q] ...
- Deviation from the standard
- Custom/proprietary extensions
- Reference implementations in C

Hardware description

Do we need to go beyond SVD?

- Registers ✓
- Interrupts ✓
- CPU specs ?
- RAM and ROM banks ?
- Modular representation ?
- Tools that generate SVD from custom design ?

Resources

- learn.adacore.com : interactive learning website
- Twitter : @AdaProgrammers
- Reddit : r/ada