AdaCore DIGEST

FEBRUARY 2026

AdaCore

DIGEST

NEWSLETTER 02/10

In This Issue

This issue of the AdaCore Digest covers a mix of different topics. We provide the latest updates on events, public training courses and Ada-related press, but we also provide a deeper background on some of the projects we are working on and what is happening in the community.

Specifically, we cover the following topics:

- AdaCore Release 26.1 announcement
- An update from Olivier Henley on Capstone projects, projects done by final year students using Ada to build a project end-to-end.
- A first-hand experience of using Alire
- A description of our upcoming Polyglot beta
- Recent blog posts
- Upcoming events and training
- AdaCore in the press

We hope you enjoy reading this issue of the AdaCore Digest, and feel free to reach out if you have any suggestions for future content.

26.1 Release

AdaCore is proud to announce the availability of the first stable version of our release branch 26: Release 26.1. If you select this branch for productization builds, it is possible to subscribe to the assurance product on that specific branch and receive critical bug fixes passed this date.

Full release notes for the 26.1 release are available on our documentation website. The new contributions cover many phases in the software development life cycle, from architecture support and language features to compiler enhancements as well as improvements to our testing suite.

Our technical team will provide a deep dive on the top new capabilities in a webinar on March 19th.

Capstone: Open Source Debug Probe in Ada on Bare Metal Hardware

By Olivier Henley

Academia plays a key role in shaping the next generation of engineers. Through the GNAT Academic Program (GAP), AdaCore supports universities working with Ada, SPARK, formal methods, and low-level systems engineering. A central part of this effort is supporting Capstone projects, where small student teams apply their knowledge to concrete, real-world engineering problems.

AdaCore contributes to these projects through regular coaching and mentoring sessions. The focus is practical and technical, covering embedded hardware bring-up, bare-metal Ada programming, and sound system design practices.

Two such Capstone projects are currently underway at Penn State Behrend and Rose-Hulman Institute of Technology. The objective is to design and implement an open-source hardware programmer and debug probe, targeting platforms such as STM32 devices or the Raspberry Pi Pico. Rather than relying on fixed off-the-shelf probes, the programmer itself is implemented as firmware running directly on an MCU, with native USB device support written in Ada.

The projects use Ada's fine-grained runtimes directly on bare metal, without an RTOS, and rely on open-source HAL and middleware components distributed through Alire. The intended outcome is an open-source Alire crate, allowing others to study, reuse, and extend the work.

This approach reflects a broader philosophy shared by GAP. Effective systems engineering requires building complete, end-to-end solutions. By focusing on firmware-driven tooling implemented in Ada, these projects combine hands-on embedded development with sustainable, open-source engineering practices.

The work is ambitious, but deliberately structured to remain manageable. It demonstrates in a concrete way that using Ada instead of C on bare metal is not only viable but advantageous in terms of correctness, maintainability, and long-term control. It also establishes a foundation that can be extended by future Capstone teams and, potentially, adopted by the wider community.

At present, one team is focused on bringing up USB support on an STM32F070xx MCU, while the other is working with the Raspberry Pi Pico (RP2040). This phase is already producing useful design feedback, including evaluation of cost, performance, and available headroom. The near-term milestone is a host application capable of issuing commands to the programmer, enabling reliable read and write access to target flash memory for programming both an initial FPGA bitstream and subsequent softcore firmware running on the FPGA.

The teams have until the end of April to deliver a functional system. If time permits, either later this year or with a future cohort, the next phase will explore debugging support.

Overall, development is progressing well. Once the academic year concludes, I will share further updates and project resources. This is a hands-on effort focused on building real infrastructure under real constraints, using Ada throughout, while giving students direct exposure to the realities of serious systems engineering.

A first-hand experience using Alire (with or without AI)

Alire, the Ada LIbrary REpository is a command-line tool that manages the dependencies for your Ada project for you. It provides an easy-to-use interface to define dependencies, including the compiler and runtime environment, automatically download them, and even upload your own libraries (aka 'crates') back to the Alire ecosystem.

The getting started pages provide easy instructions to download Alire from the provided releases on GitHub, download the needed compilers and libraries, and then build and run a simple HelloWorld example by simply running `alr get –build hello`.

Alire works with a community set of crates, which is a great place to start for your personal or community projects. It includes support for many embedded hardware boards as well; getting something to run on a Raspberry Pi Pico or an ST Micro evaluation board is very straightforward.

At the same time, Alire is powerful enough that you can use it with a more private index of curated crates. This capability allows corporations to define and maintain a set of approved crates for internal use.

Making a small example for a Raspberry Pi Pico, if you happen to have one (or more) lying around, is as simple as creating a new project with `alr init`, answering the questions, stepping into the project directory, adding dependencies to the `pico_bsp` and `embedded_rp2040` with `alr with pico_bsp` and `alr with embedded_rp2040` will automatically download the required libraries and compilers.

Flashing an LED on the pico is then as simple as:

```
        with Pico;
        with RP.GPIO;

        procedure Test is
        begin

           Pico.LED.Configure (RP.GPIO.Output);

           loop
              Pico.LED.toggle;
              delay 0.5;
      end loop
end Test;
```

Then to build: ‘`alr build`’ and you have a binary that you can deploy to your target. There will be a demo of a full fledged GitLab project of this functionality on the AdaCore blog soon, including the use of a Lauterbach **µ**Trace hardware debug tool which provides source level debugging capabilities for Ada, but there is a wonderful community supported hardware Pico Debug Probe and the Capstone project in the previous topic is working to create something similar completely done in Ada.

Alire supports both Ada and SPARK and works with the AdaCore GNAT Pro suite as well as with the FSF compilers, so it is a great location for makers to get started.

Want more assistance? Most Large Language Models are very much aware of Alire and a prompt like the following ‘Can you create a blinky example using Alire and the AdaCore Drivers Library targeting the STM32F469_discovery’ will get your lights blinking easily. More information on some of the embedded boards that will work well is available on ada-lang.io.

Polyglot Beta

There is a resurgence of interest in memory safety driven by regulators in different parts of the world. People are exploring switching from traditional programming languages to memory safe approaches either for their entire software project, or specifically for the layers in their project that matter most. Whether that is security, encryption, scheduling, task management, you name it. Popular memory safe languages are Ada as well as SPARK and of course Rust. Interestingly, as of the writing of this article, Ada is ahead of Rust in the PyPL Popularity of Programming Language index, maintained by Google, Ada is on the 9th spot with tremendous increase in the past 12 months, while Rust dropped a bit to the 10th spot.

Multi-language systems introduce a new challenge, how do you build a project consisting of multiple languages. How do you connect an Ada encryption module to a large existing C++ application? The languages support this binding, but how can you make this easy for the development team?

By 'easy' we mean that:

1. Creating the mapping between API calls and data structures from one language to another needs to be easy and not require tedious manual work
2. The expressiveness of the data structures needs to be maintained; we need to avoid forcing development teams to create elaborate pieces of code to map a C++ data structure to another language
3. If the data structure or API changes, then the binding needs to adapt.

At AdaCore, we have supported customers with multi-language systems for years. The GNAT Pro and SPARK Pro suites of toolchains as well as our static and dynamic analysis solutions were designed with multi-language support in mind.

We have been working in the background to encode our knowledge of multi-language bindings into an automated tool that we dubbed Polyglot. We are looking for a set of beta users that are interested in trying Polyglot with us.

Polyglot works by positioning a proxy between an object (which could be a data structure in memory, or an API) on the callee side as well as the caller side. Development teams do not have to be concerned about the proxies themselves, they are by-products of the compilation and linking process. They can simply call through these proxies to reach functionality in the other language.

Initially, we are looking for projects that call an Ada or SPARK application from C++. So maybe a QT style user interface that calls Ada logic, or a C++ networking app that calls an encryption/decryption module in SPARK.

Let us know if you are interested in trying Polyglot in your environment. Reach out to your AdaCore technical contact and we will set-up a conversation.

Recent Blog Posts

Our blog at adacore.com has a steady stream of interesting content, here is a selection of recent posts:

- Advent of Ada 2025 contest results
- Video: Proving Correctness of AI-Generated Code Using Formal Methods
- Memory management in Rust : The Playroom Analogy
- Suprising places where Ada is used
- MISRA for Memory Safety
- [Video] A Day in the Life of a Software Engineer

Events

Save the date:

- Embedded World 2026: 10-12 March 2026 - Booth 4-116, Nuremberg, Germany
- Army Aviation Warfighting Summit: 15-17 April - Booth 221, Nashville TN
- High Integrity Software Conference: 13 October 2026, Birmingham, UK
- Paris Tech Day - 3 November 2026

Coming soon:

- Polyglot webinar - exact date to be announced
- Boston Tech Day - in September 2026
- Michigan Tech Day - in September 2026

Webinar: AdaCore 26: Accelerating the High-Integrity Software Development Lifecycle
AdaCore is constantly evolving the tools available to build high-integrity software. Join us in this webinar where we provide a high-level overview of the new capabilities in the 26 release of our solutions before we do an engineer-led technical deep dive into the top features. Sign up for the webinar

Training

We offer Public AdaCore Training (for Ada) at fixed intervals throughout the year, making our classroom experience with expert instructors accessible for teams of all sizes. This course is a cost-effective way to access our most popular Ada course. The remaining dates for 2026 are:

- US: March 16 - 20
- EU: June 8 - 12
- US: September 14 - 18

We look forward to seeing you there. See more information here.

If you'd like to take your training a step further, or have a more critical time-based need, we offer Enterprise Training solutions for Ada, Rust and our most popular tool suites (GNAT SAS and GNAT DAS). You can explore all of our options here.

AdaCore in the press

Recently, The New Stack investigated the recent rise in popularity of Ada and asked us for our opinion. Last March, Ada broke into the TIOBE Index top 20 (reaching number 18), and by July, Ada broke the top 10 (reaching number 9 – its highest-ever position on TIOBE). It is now back to number 18.

Moreover, this month Ada also broke into the top 10 in the PopularitY of Programming Language Index (PYPL), landing at number 9.

While programming languages such as Python, C/C++, and Java continue to rank amongst the most popular languages, the resurgence of interest in Ada could partially be explained by the push to use more memory-safe languages.

You can read the full article by Darryl Taft, here: https://thenewstack.io/2025-the-year-of-the-return-of-the-ada-programming-language/

Have you seen the resources section on our website? From here, you can read our Blogs, Case Studies, and Papers. We also have a dedicated newsroom where our most recent press releases and editorials are available.

If you have questions about any of the technologies or services mentioned above, please reach out to your Account Manager or email us at info@adacore.com