



Being Productive with Open Source Eclipse Development Tools

Frédéric Desbiens, Eclipse Foundation
Alexander Fedorov, ArSysOp

September 15, 2020



Agenda

- > The Case for Open Software at OpenHW and Beyond
- > CORE-V IDE and its building blocks

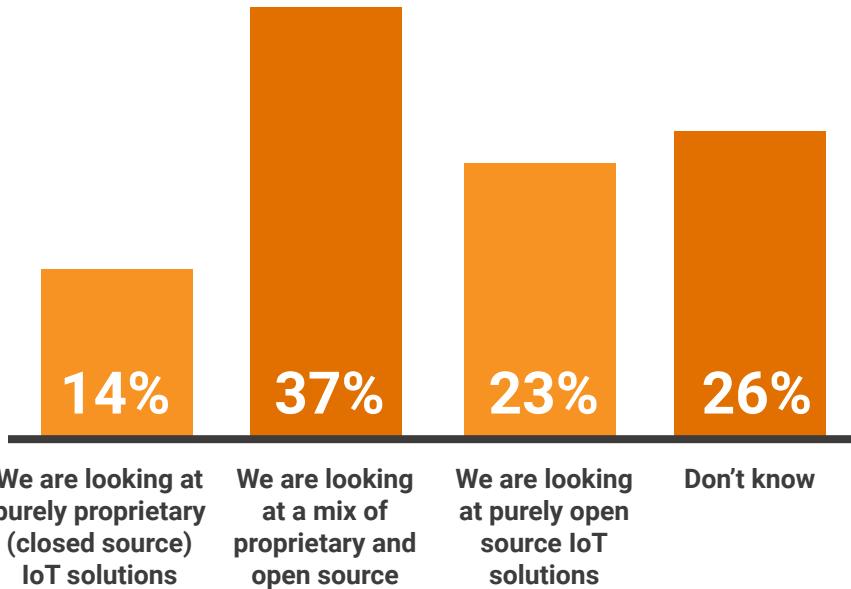


Openness Creates Ecosystems

ECLIPSE®
FOUNDATION

KEY TAKEAWAY: 2019 IoT Commercial Adoption Survey

What are your plans for implementing IoT solutions based on open source technology?



60% of companies are factoring open source into their IoT deployment plans. This clearly means the dominant IoT platforms in the market will either be open source or based on an open source core.

2019 IoT Commercial Adoption Survey

Top 3 advantages of using open source technologies

(% of question respondents)

Flexibility

55%

Cost

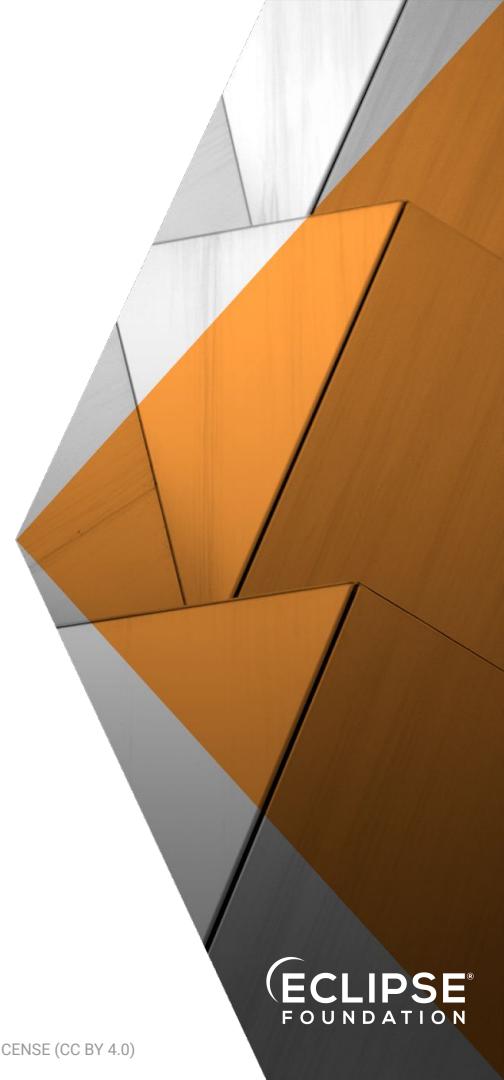
49%

More Control

41%

OpenHW Group Software Task Group

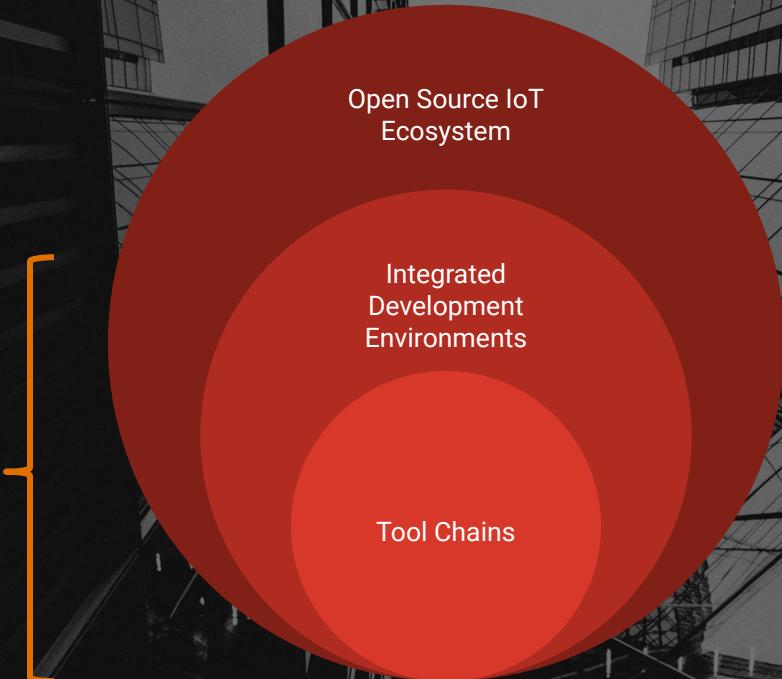
- > Compiler tool chains (GNU, LLVM, proprietary) and Operating systems (RTOS, *nix, *bsd)
 - Tool chain Includes assembler, linker, debugger and libraries
- > Processor and platform models (ISS, cycle accurate, QEMU, OVPSim)
- > IDEs (Eclipse family, PlatformIO, proprietary)
- > Benchmarking (specifically Embench)
- > Demonstration applications





OpenHW Group Software Task Group: Goals

- Create a thriving commercial ecosystem for CORE-V software tools, models and operating systems
- To see those tools, models and operating systems which are open source maintained as part of official upstream distributions



ECLIPSE
FOUNDATION

eclipse
iot

ECLIPSE CLOUD
DEV TOOLS

ECLIPSE
FOUNDATION



Powering the world's leading commercial IoT solutions



Things



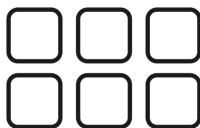
Edge



Cloud



Eclipse IoT Community (as of 9/2020)



8M+

lines of code

45

projects

350+

contributors

46

member
organizations



IoT Working Group Member Organizations

Strategic members



BOSCH

Invented for life



EUROTECH

Imagine. Build. Succeed.



Red Hat



CANONICAL



cedalo



cx COMPLEX



fortiss



itemis



NOKIA



SIEMENS

Sterwen
Technology



Who am I?

Alexander Fedorov @ ArSysOp

 Eclipse Platform Committer

 Eclipse CDT Committer

 Eclipse Passage Project Lead

Leading IDE effort
at Software Task Group



What do we expect from IDE?

- > Configurability
- > Extensibility
- > Scalability
- > Usability
- > Agility



Copyright (c) 2020 OpenHW Group and others

CORE-V IDE already aggregates:



-  Eclipse Modeling Framework
-  Eclipse Platform
-  Eclipse CDT
-  Eclipse Embedded CDT
(GNU MCU/ARM
Eclipse Plug-ins)



Eclipse Modeling Framework

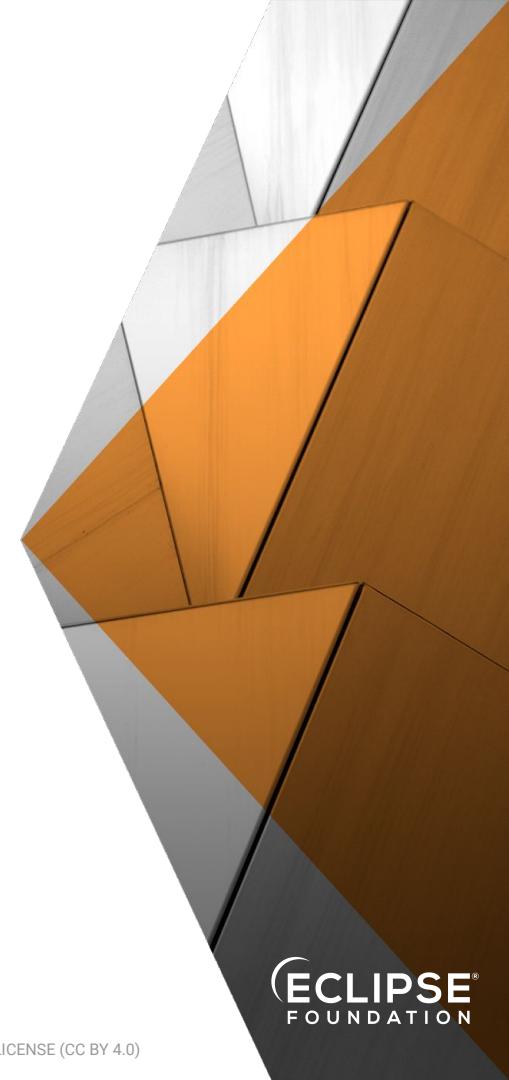
The EMF project is a modeling framework and code generation facility for building tools and other applications based on a structured data model.

Learn more:
<https://www.eclipse.org/modeling/>



Eclipse Platform

- > OSGi-based extensible runtime
- > Workspace to manage project metadata
- > “natures” and “builders” to process content
- > SWT and JFace to create reusable UI
- > Workbench model to organize perspectives
- > p2 to resolve dependencies during update
- > User assistance capabilities





Eclipse CDT: the right choice for IDE

Nios II IDE 	MCUXpresso 	Code Warrior 	TrueStudio 	Sloebier (for Arduino) jantje 	Kalray 	iDev
VX Software 	Artik IDE 	e2 studio 	Momentics 	Code Composer 	Sourcery CodeBench 	Simplicity Studio
DAVE 	Xtensa Xplorer 	DS-5 	CrossCore (CCES) 	eGui 	Ascet Developer 	Cdevelop
XSDK 	Luminosity 	SoftConsole 	Snapdragon Debugger 	Wind River Workbench 	System Workbench 	COSIDE®

Eclipse CDT: project configuration

The screenshot illustrates the Eclipse CDT (C/C++ Development Tools) interface for project configuration, specifically for a RISC-V target.

Left Sidebar: A vertical sidebar on the left contains a tree view of configuration categories:

- Resource
- Builders
- C/C++ Build
- Build Variables
- Environment
- Logging
- Settings** (selected)
- Tool Chain Editor
- C/C++ General
- Git
- Linux Tools Path
- MCU
- Project Natures
- Project References
- Run/Debug Settings
- Task Repository
- Task Tags
- Validation
- WikiText

Properties for rtc_func_ri5cy_rv32m1_vega

Settings tab (selected):

- Configuration: debug [Active]
- Tool Settings, Toolchains, Devices, Container Settings, Build Steps tabs
- Name: GNU MCU RISC-V GCC (riscv-none-embed-gcc)
- Architecture: RISC-V
- Prefix: riscv32-unknown-elf-
- Suffix:
- C compiler: gcc
- C++ compiler: g++
- Archiver: ar
- Hex/Bin converter: objcopy
- Listing generator: objdump
- Size command: size

Preprocessor Include Paths, Macros etc.

Configuration: debug [Active]

Entries tab (selected):

- Setting Entries:
 - CDT User Setting Entries
 - Exported Entries from Referenced Projects [Shared]
 - CDT Managed Build Setting Entries [Shared]
 - CDT RISC-V Cross GCC Built-in Compiler Settings** (selected)
- Language: GNU C
- Entries list:
 - /home/jonah/riscv/vega/boards/rv32m1_vega/demo_apps
 - /home/jonah/riscv/vega/boards
 - /home/jonah/riscv/vega/boards/rv32m1_vega/demo_apps/rtc_func
 - /home/jonah/riscv/toolchain/riscv32-unknown-elf-gcc/lib/gcc/riscv32
 - /home/jonah/riscv/toolchain/riscv32-unknown-elf-gcc/riscv32-unkno
 - /home/jonah/riscv/toolchain/riscv32-unknown-elf-gcc/riscv32-unkno
 - # CPU_RV32M1_ri5cy=1
- Add..., Edit..., Clear Entries, Export, Move Up, Move Down buttons

Preferences

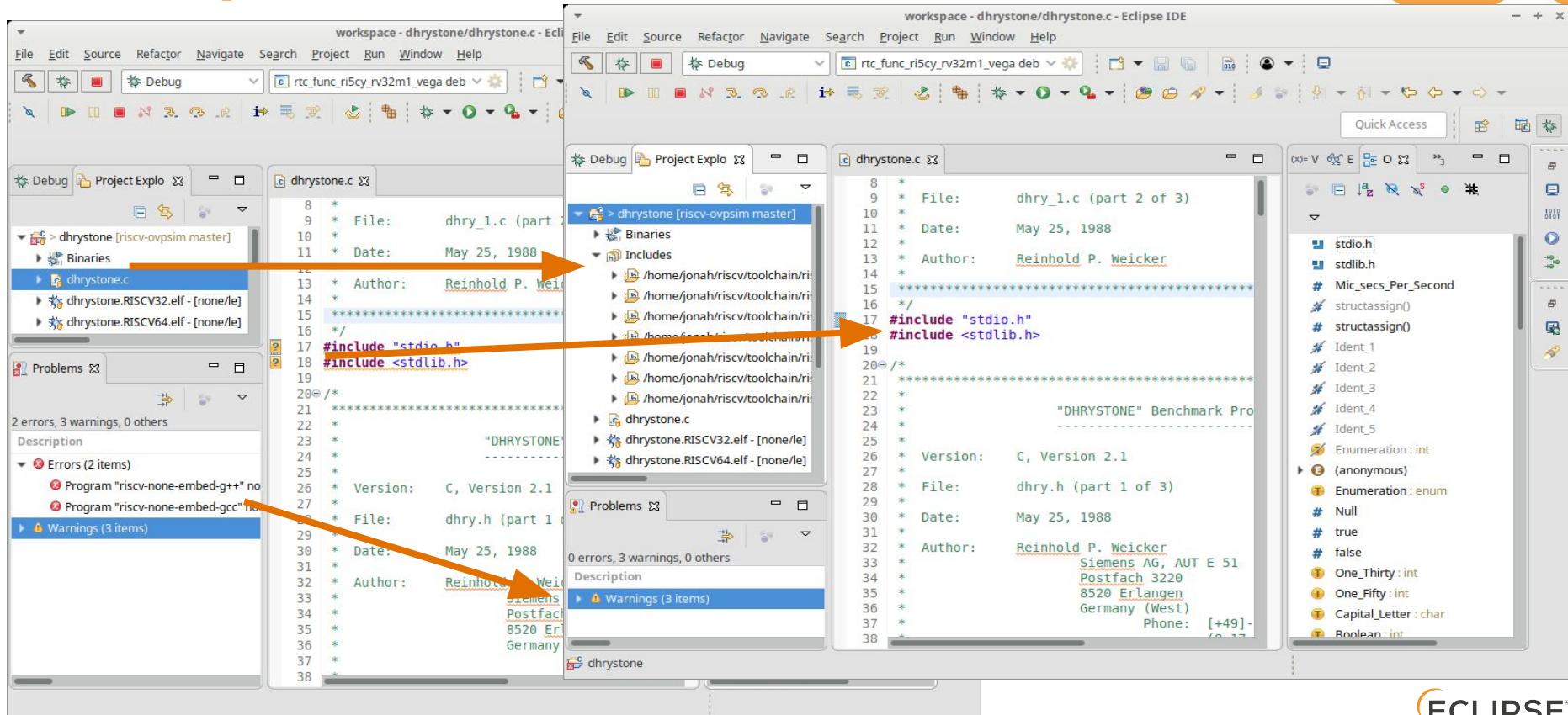
Global RISC-V Toolchains Paths

Configure the locations where various GNU RISC-V toolchains are installed. The values are stored within Eclipse. Unless redefined more specifically, they are used for all projects in all workspaces.

Default toolchain: **GNU MCU RISC-V GCC**

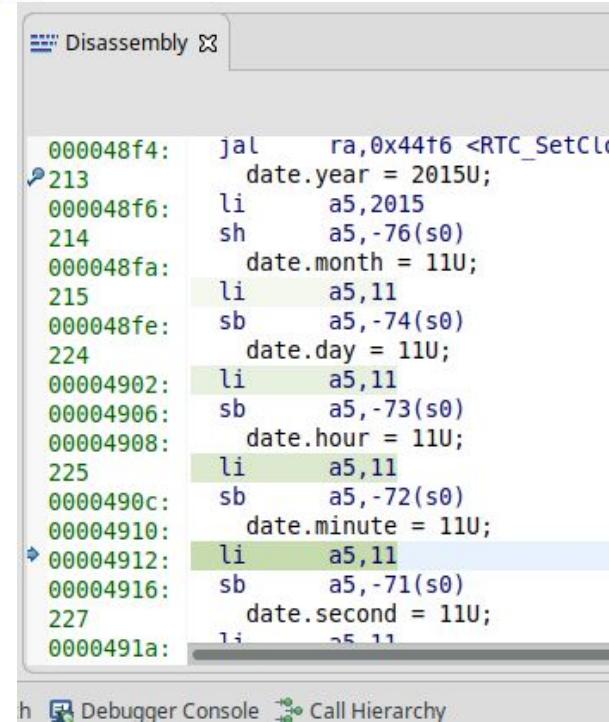
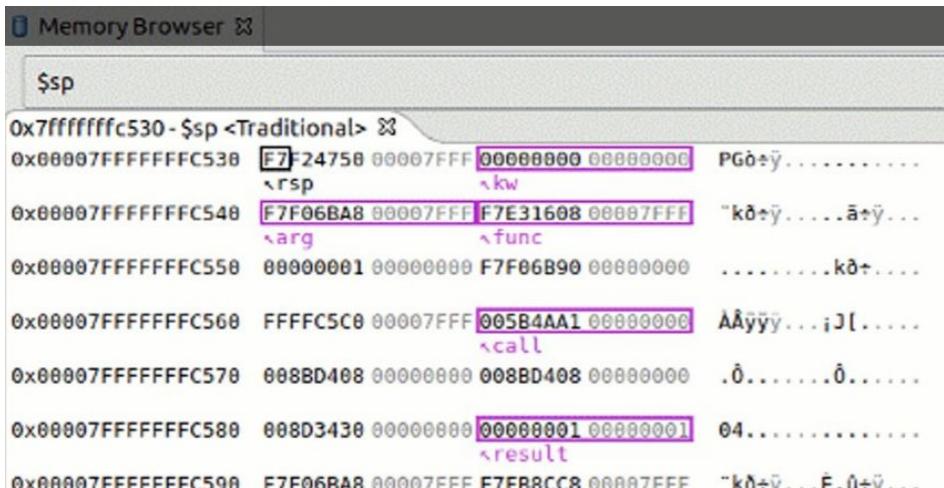


Eclipse CDT: before & after



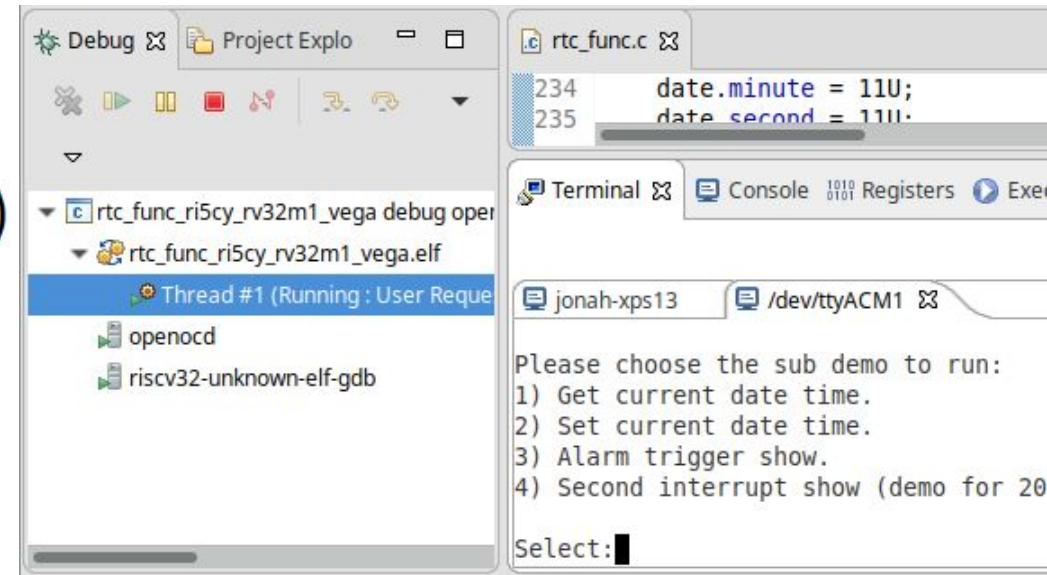
Eclipse CDT: rich debug information

- Variables, Breakpoints, Expressions & Hovers
- Disassembly
 - Instruction Stepping
 - Gradients to easily visualize stepping
- Memory Browsing with Annotations



Eclipse CDT: CLI tools integration

- Integrated Terminals and Consoles
- Debugger Console View
 - Full GDB command line experience with all the niceties of an IDE
- Terminal View
 - Serial Connection target
 - Telnet/SSH (e.g. to openocd)
 - Local terminal (e.g. bash)





Eclipse Embedded CDT

- create/build/manage embedded ARM/AArch64/RISC-V applications
- ready to run templates for some ARM Cortex-M processors
- debugging support via JTAG/SWD
- examine and modify peripheral registers during debug sessions
- supports a wide range of 32 and 64-bit toolchains

Learn more <https://projects.eclipse.org/projects/iot.embed-cdt>

berry:1 (ilg) - TigerVNC

eclipse-workspace - f4b/src/main.cpp - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Debug Project

f4b-qemu [GDB QEMU Debugging]

f4b.elf

Thread #1 (CPU#0 [runn...])

main() at main.cpp:205

qemu-system-gnueclips

arm-none-eabi-gdb

main.cpp

```
205     blinkLeds[i].turnOn ();
206 }
207
208 // First second is long
209 timer.sleep (Timer::FREQUENCY_HZ);
210
211 for (size_t i = 0; i < (sizeof(blinkLeds) / sizeof(blinkLeds[0])); ++i)
212 {
213     blinkLeds[i].turnOff ();
214 }
215
216 timer.sleep (BLINK_OFF_DURATION);
217
218 ++seconds;
219 trace_printf ("Second %d\n", seconds);
220
221 if ((sizeof(blinkLeds) / sizeof(blinkLeds[0])) >= loops)
```

V B E

Name	Type	Value
argc	int	1
argv	char**	0x20
seconds	uint32_t	0
loops	int	16

Console

```
f4b-qemu [GDB QEMU Debugging]
Hello ARM World!
Standard output message.
Standard error message.
System clock: 168000000 Hz
[led:green on]
[led:orange on]
[led:red on]
[led:blue on]
```

Writable Smart Insert 20%

ST Discovery kit for STM32F407/417 lines

Eclipse Embedded CDT on Raspberry Pi 4 (based on Eclipse Platform for Aarch64)

More to consider



Eclipse LPS4J
Eclipse LPS4E

LSP & DAP



PlatformIO

Embedded
development



Eclipse Passage

License checks
Usage constraints



Eclipse LSP4J

Java implementation of VSCode's language server protocol (LSP) and debug adapter protocol (DAP)

Learn more about LSP:

<https://microsoft.github.io/language-server-protocol/>

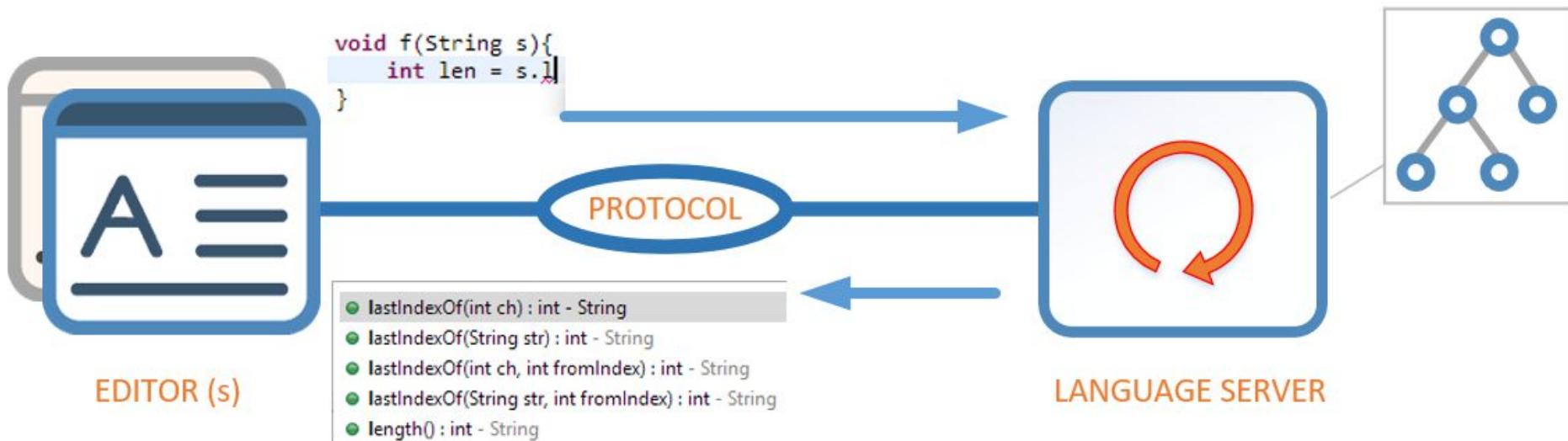
Learn more about DAP:

<https://microsoft.github.io/debug-adapter-protocol/>



Eclipse LSP4E

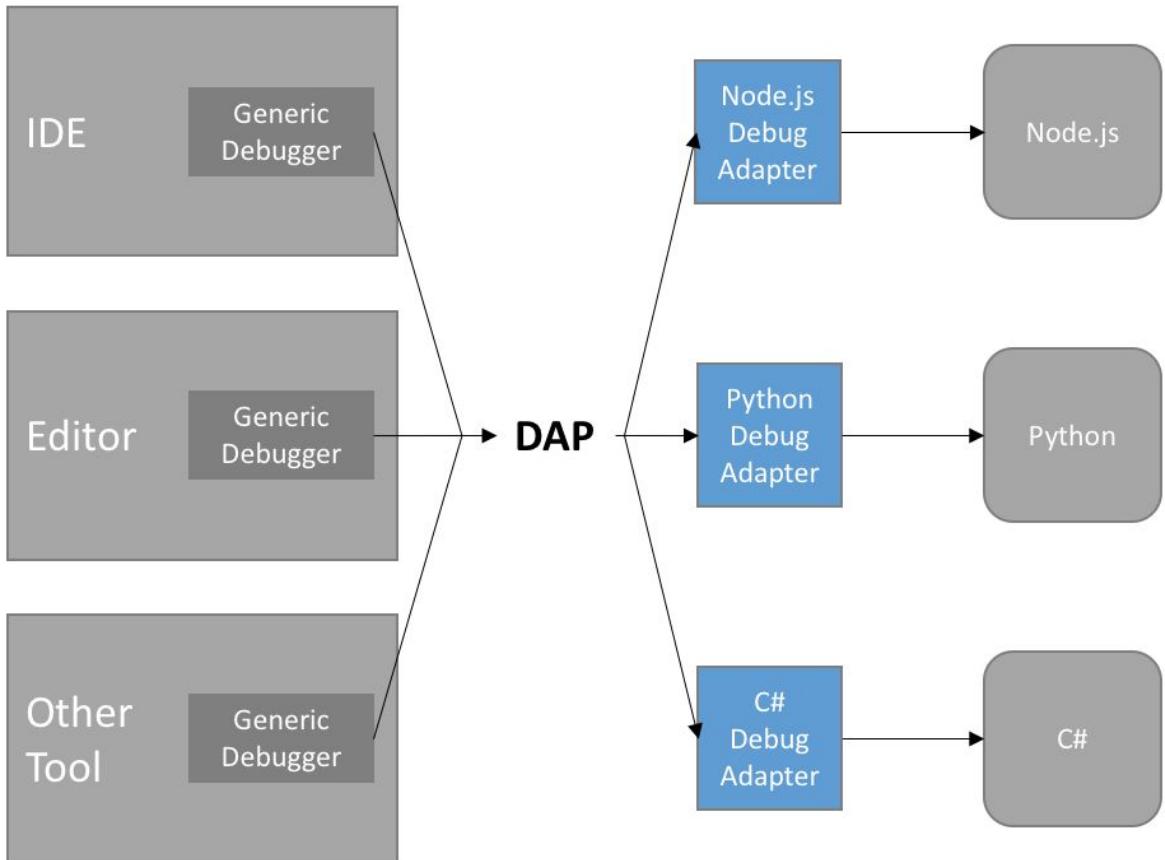
Move heavy tasks like AST building and traversing to a “language server” (separate process)



Debug Adapter Protocol

Development Tools

Debuggers





Embedded development

PlatformIO is a professional collaborative platform for embedded development that support multiple IDE including Eclipse

- **800+ target boards** (development kits)

- **20+ software frameworks**

(*Arduino, ARM mbed, CMSIS, ESP-IDF, FreeRTOS, STM32Cube, Zephyr RTOS, and others*)

- **30+ semiconductor architectures and development platforms**

(*ARM, AVR, Espressif 8266/32, MCS-51, MSP430, PIC32, STM8, RISC-V, and others*)

- Over **10,000 libraries**

- **All famous operating systems**

(*Windows, macOS, Linux, FreeBSD, Linux ARMv6+, card-sized PCs*)



PlatformIO

Eclipse integration

- Multi-board and Multi-architecture programming experience
- Debugging, Unit Testing, Static Analysis, Firmware Inspection, and Remote Development out-of-the-box
- Developers can work simultaneously on the same embedded project using different development environments and the favourite operating system
- Code for any supported framework can be compiled and uploaded to a target platform in minutes
- Developers no longer have to manually find and assemble an environment of toolchains

The screenshot illustrates the Eclipse IDE interface with the PlatformIO debugger plugin. The left side shows the Project Explorer with various build targets and source files. The right side features several views: the Debug view with a context menu open; the Expressions view displaying memory values; the Disassembly view showing assembly code; and the Registers view showing the state of CPU registers. The code editor at the bottom shows C++ code for initializing a timer.

```
57 // Set SysTick to 1ms interval, common to all cores
58 void init( void )
59 {
60     // Capture error
61     while ( 1 );
62     NVIC_SetPriority (SysTick_IRQn, ( 1 << 8 ) - 1 );
63     // Clock PORT for Digital I/O
64     PM->APBMASK.reg |= PM_APBMASK_PORT ;
65     // Clock EIC for I/O interrupts
66     PM->APBMASK.reg |= PM_APBMASK_EIC ;
67     // Clock SERCOM for Serial
68     PM->APBMASK.reg |= PM_APBMASK_SERCOM0 ;
69     // Clock TCC for Pulse and Analog
70     PM->APBMASK.reg |= PM_APBMASK_TCC0 | PM_APBMASK_TCC1 ;
71     // Clock ADC/DAC for Analog
72     PM->APBMASK.reg |= PM_APBMASK_ADC | PM_APBMASK_DAC ;
73 }
```



Eclipse Passage

Define and control functionality constraints

The screenshot shows the Eclipse IDE interface with the following details:

- Licensing Status Dialog:** A modal window titled "Licensing" displays the "Licensing status". It shows a single entry in the table:

Provider	Name	Version	Identifier	Level
Rcp	Rcp	0.0.0	some.licensed.rcp.product	warn

Below the table, it says "Please contact your Licensing Operator for details" and provides the "Eclipse Passage" URL: <https://www.eclipse.org/passage>.

Buttons at the bottom of the dialog include "Configuration...", "Import...", "Hardware...", and "Close".

- MANIFEST.MF Editor:** The main workspace shows the MANIFEST.MF file with the following content:

```
org.ecl.lic.equinox;bundle-version="0.0.0",
org.ecl.lic.e4.ui;bundle-version="0.0.0"
ExecutionEnvironment: JavaSE-12
Name: some.licensed.rcp
*Provide=Capability: licensing.feature;licensing.feature="some.licensed.rcp.product"
Bundle-ActivationPolicy: lazy
22
```
- Outline View:** On the right, the Outline view lists various manifest entries:
 - Manifest-Version
 - Bundle-ManifestVersion
 - Bundle-Name
 - Bundle-SymbolicName
 - Bundle-Version
 - Require-Bundle
 - javax.inject (0.0.0)
 - org.eclipse.core.runtime (0.0.0)
 - org.eclipse.swt (0.0.0)
 - org.eclipse.jface (0.0.0)
 - org.eclipse.e4.core.di (0.0.0)
 - org.eclipse.e4.ui.di (0.0.0)
 - org.eclipse.e4.ui.services (0.0.0)
 - org.eclipse.e4.ui.model.workbench (0.0.0)
 - org.eclipse.e4.ui.workbench (0.0.0)
 - org.eclipse.core.contexts (0.0.0)
 - org.eclipse.passage.lic.equinox (0.0.0)
 - org.eclipse.passage.lic.e4.ui (0.0.0)
 - Bundle-RequiredExecutionEnvironment
 - JavaSE-12
 - Automatic-Module-Name
 - Provide-Capability
 - Bundle-ActivationPolicy

What do we have in the nearest plans?

- > Integrate toolchain from Embecosm
- > Add “Hello World” sample project
- > Provide project templates
- > Publish binaries to be a foundation for downstream solutions





You are welcome to participate!

1. Create account at <https://www.eclipse.org/>
2. Sign Eclipse Contributor Agreement (electronically)
3. Specify your GitHub id in your Eclipse profile
4. Fork <https://github.com/openhwgroup/core-v-ide-cdt>
5. Don't forget to add “signed-off-by” to commit message

Example:

```
Signed-off-by: Alexander Fedorov <alexander.fedorov@arsysop.ru>
```

Thank you!

Questions?

Frédéric Desbiens
@BlueberryCoder

Alexander Fedorov
alexander.fedorov@arsysop.ru

