arm

CMSIS

Review Meeting

Simplify IoT and ML for microcontrollers

CMSIS Team

15. March 2022

Agenda

- CMSIS – Overview and market adoption (www.arm.com/cmsis)
- Open-CMSIS-Pack – Infrastructure to manage software components and improve code reuse (www.open-cmsis-pack.org)
- Open-CMSIS-CDI – Common device interface for IoT and ML applications
- CMSIS-DAP v2.1 – Update on firmware for CoreSight debug access protocol
- CMSIS-DSP/NN – Update on software library for DSP and neuronal networks

arm

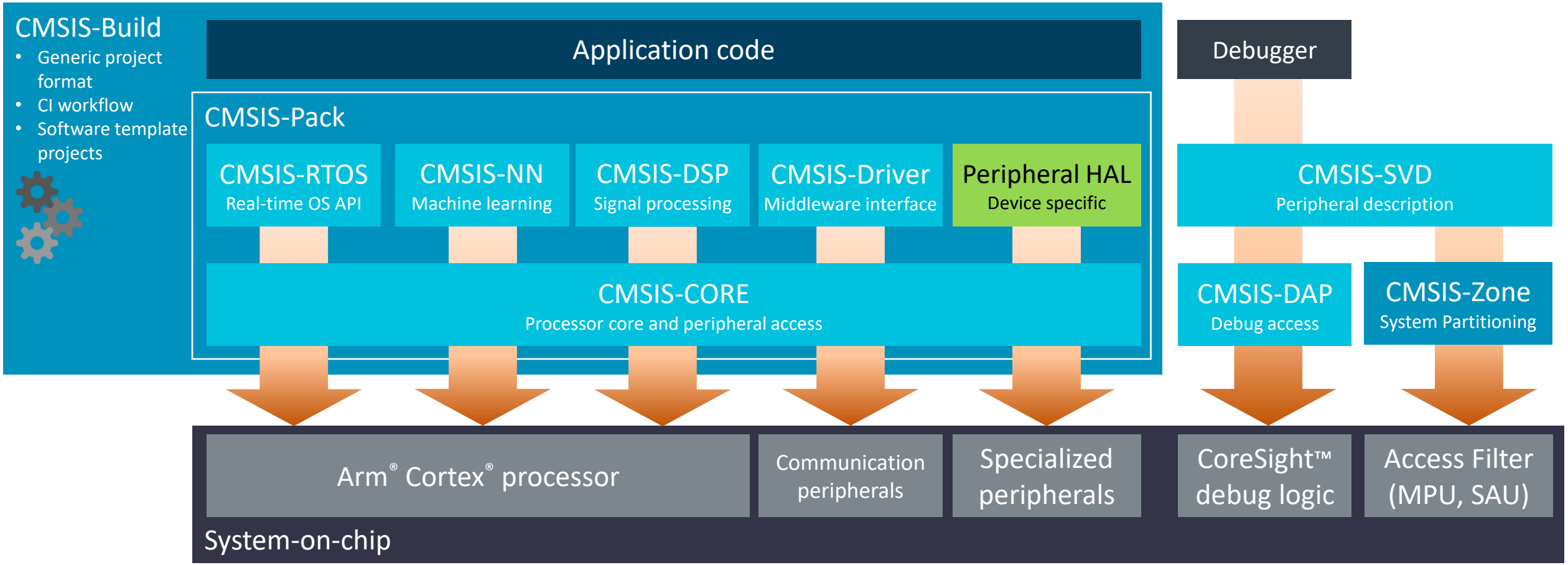
CMSIS

Overview

Christopher Seidl, xxx

CMSIS Overview of today's components

Consistent software framework for Arm Cortex-M and Cortex-A5/A7/A9 based systems



Open-CMSIS-Pack | CMSIS

CMSIS evolution to version 6

Overview of current initiatives

Open-CMSIS-Pack

CMSIS-Build
Framework and tools for productivity

CMSIS-Pack
Delivery mechanism

CMSIS-Zone
System partitioning management

Open-CMSIS-CDI

CMSIS-RTOS
Real-time operating system API

CMSIS-Driver*
Standard peripherals API

*partial

CMSIS Classic

CMSIS-SVD
Peripheral description

CMSIS-DSP
Digital signal processing functions

CMSIS-DAP
Corsight Debug Access Protocol firmware

CMSIS-NN
Machine learning functions

CMSIS-Core
Core and peripheral access

Adoption of CMSIS components

- 67 pack vendors (including pure SW vendors such as Alibaba, AWS, and Tencent)
- More than 800 different packs available publicly
- Close to 9,500 devices supported by 45 different silicon vendors
- Close to 800 unique CMSIS-Driver components covering more than 25% of all devices
- More than 450 development boards supported

The Open-CMSIS-Pack Project in Linaro

Reinhard Keil
Sr. Director Embedded Technology, Arm

Linaro

www.linaro.org

arm

Open-CMSIS-Pack

www.open-cmsis-pack.org

github.com/open-cmsis-pack

Roadmap

- Create command-line tools for project build based on software packs
- Create workflows and utilities for the verification of software packs
- Extend the pack description format for better usability across the complete workflow
- Define processes that simplify the creation of software packs from other sources, such as CMake based projects
- Develop the concept of a software layer that defines a collection of pre-configured software components
- Organize the taxonomies of standard APIs that are essential for re-useable software stack

Founding Members

arm Linaro NXP ST life.augmented

Technical Project Meetings

- Tuesdays 16:00 (CET)
- Mailing list: https://op-lists.linaro.org/mailman/listinfo/open-cmsis-pack-dev

arm

What Requirements did we consider?

Use cases driven by Application Developer

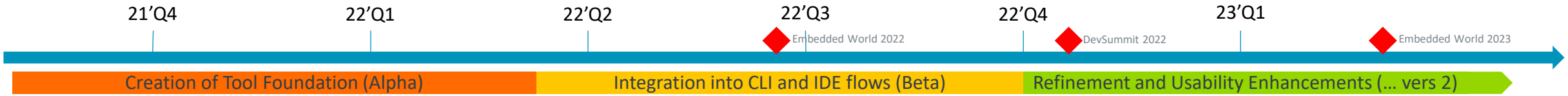
- Holistic view on software projects considering:
  - Structure
    - many dependent/related projects
    - reuse of partial projects
  - Code Generation:
    - build order dependencies
    - multiple build configurations
    - HW resource allocation partitioning and dependencies
    - generated/assisted software configuration
  - Deployment and Download:
    - flash programming setup and configuration
    - Firmware update processes including OTA programming
  - Debugging:
    - debug setup and configuration
- Simplify testing and porting of applications across devices and boards

CMSIS-Toolbox

github.com/Open-CMSIS-Pack/devtools/releases

Command-line tools – tool foundation for CLI and IDE software development flows

- Package creation and validation
  - **packgen** - create a software pack from a Cmake based software repository
  - **packchk** - semantic validation of a software pack description and the archive content
- Package management including discovery of components, devices, boards and examples
  - **cpackget** - download, add and remove packs and local repositories to CMSIS_PACK_ROOT
- Project management for constructing projects from local files and software components
  - **csolution -** manage complex applications with *.yaml user input files and content from CMSIS-Packs and output cbuild files for project build (XML: cprj format)
- Project build management
  - **cbuild (aka CMSIS-Build)** - convert a single target, single configuration project (XML: cprj format) to a CMake build
- Package index utilities
  - **vidx2pidx** - create a flat index file from a vendor index file; a public index is maintained here: www.keil.com/pack/index.pidx; vendor index: www.keil.com/pack/keil.vidx



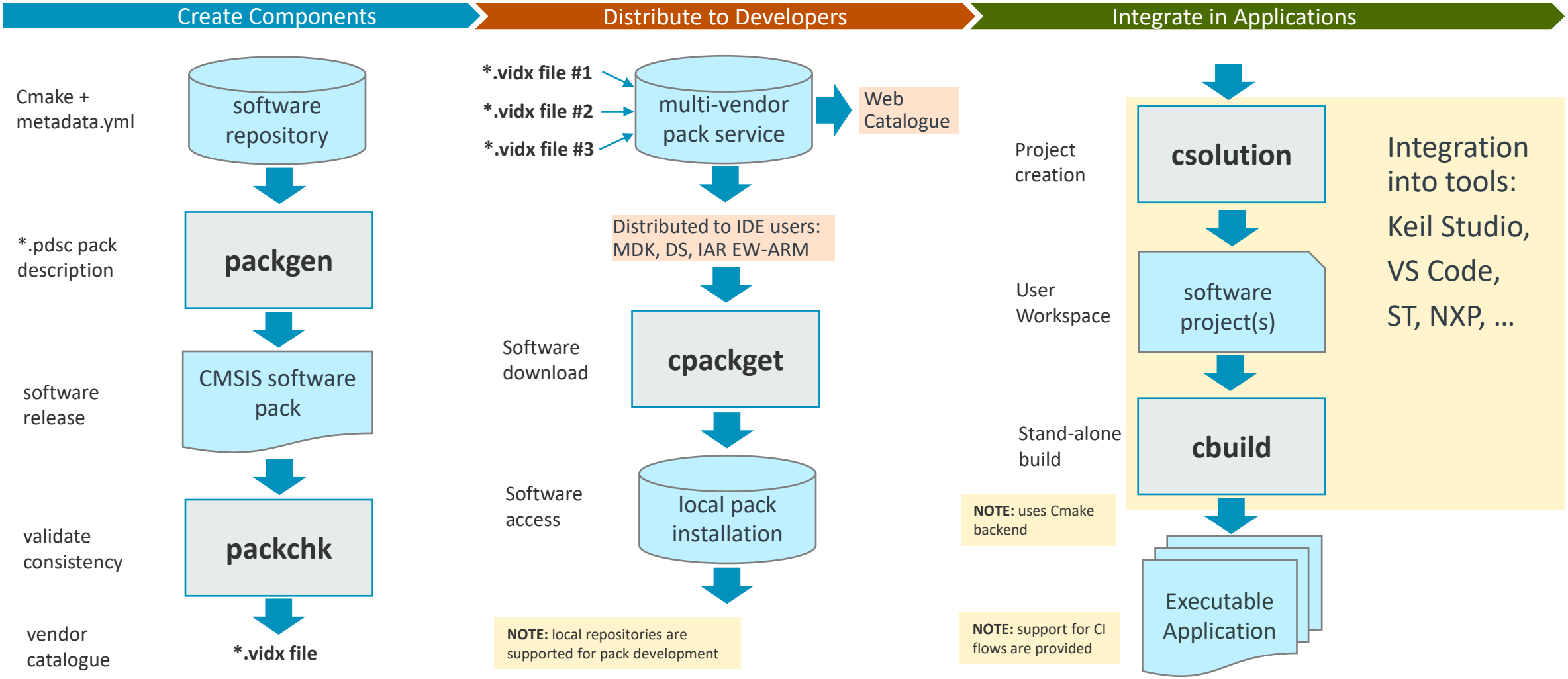
- Open-CMSIS-Pack workgroup defines product scope
- Interfaces to Device Configuration Tools
- Implementation by engineering teams

- Foundation tools are integrated into Keil Studio and VS Code
- Open-CMSIS-Pack and foundation tools get extended (CMSIS-Zone features)

- Improve compatibility with: Keil MDK, Arm DS, IAR EW-ARM
- Widen adoption by collaboration with eco-system (of Arm, NXP, and ST)

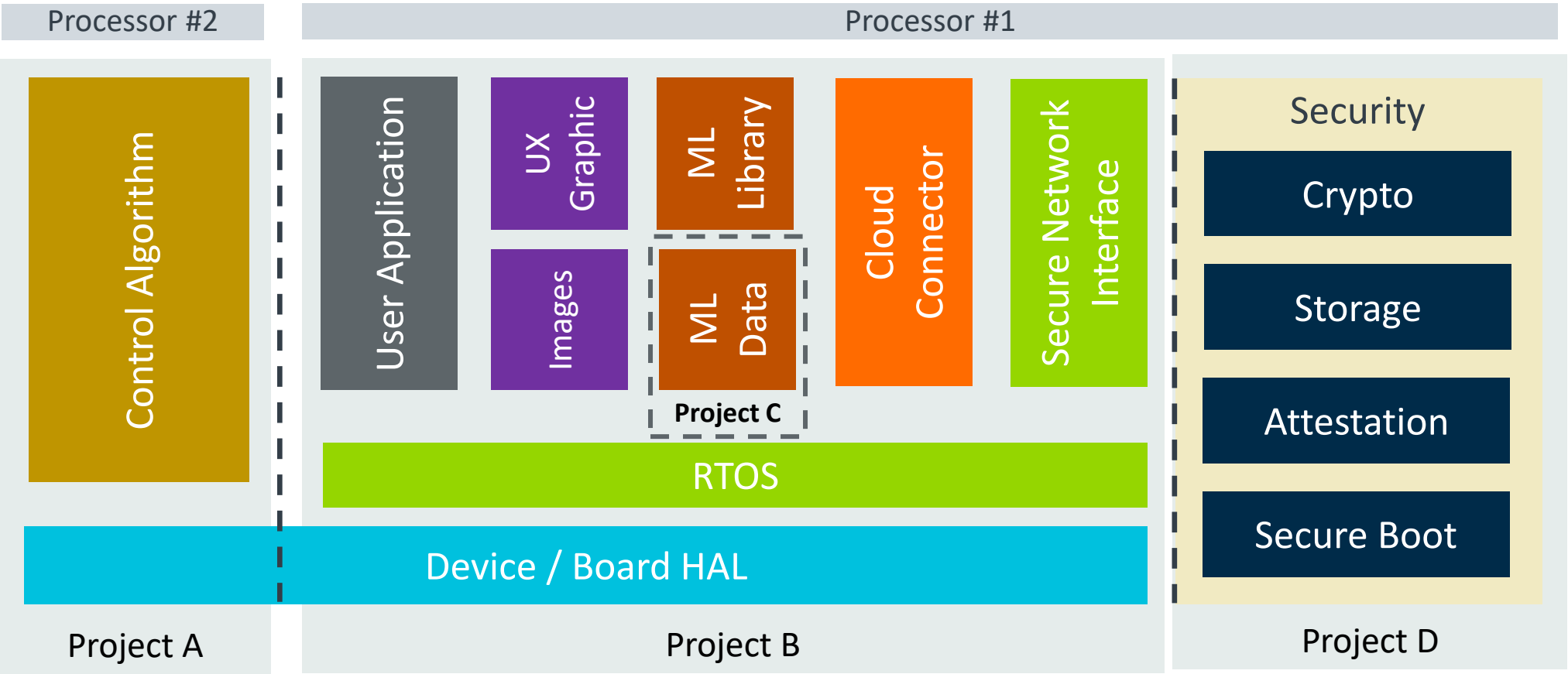
Ecosystem and Tools for effective software reuse

Relationship of CMSIS-Toolbox to development flows and software tooling



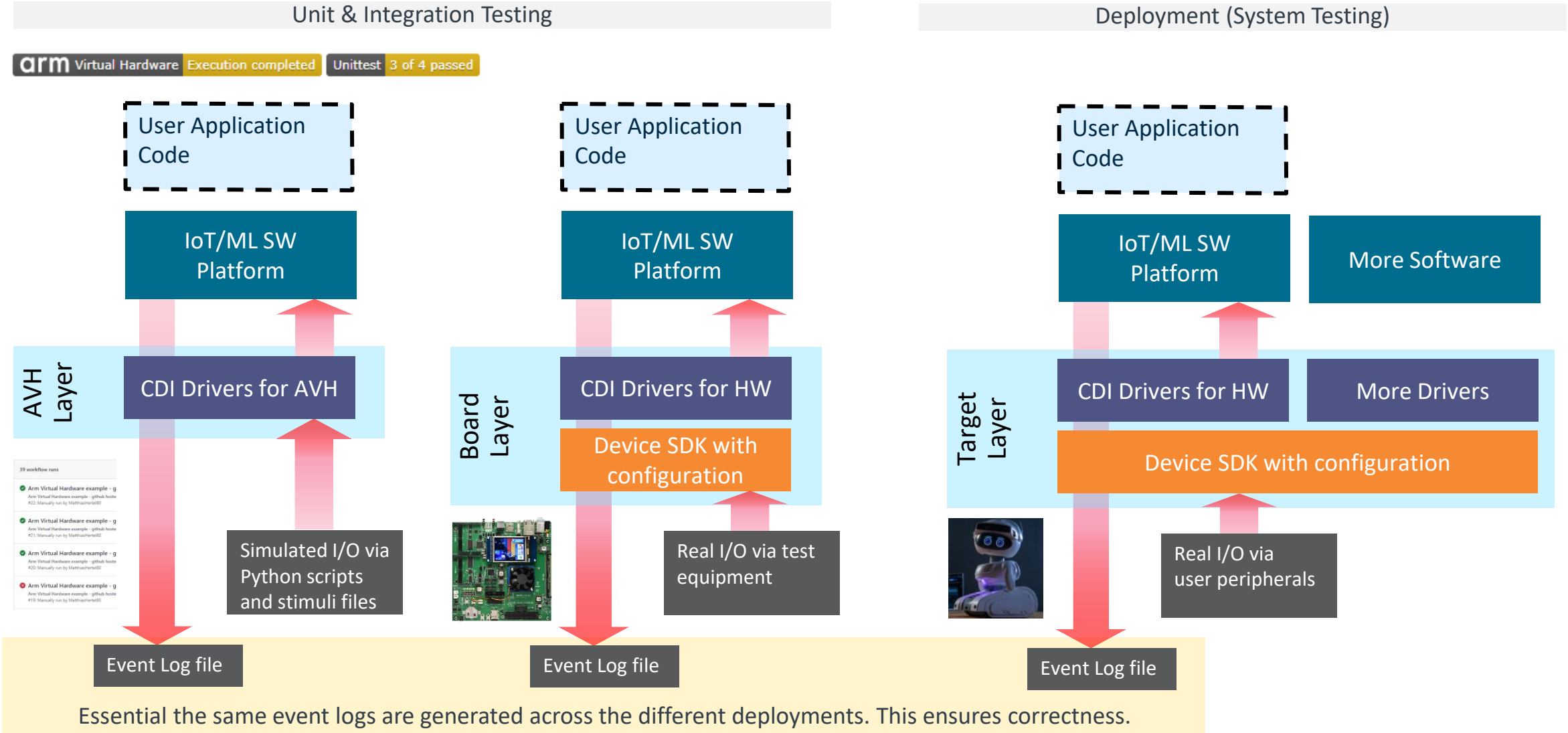
csolution: Multi-Project Management

Separate projects independently developed; combined into a “solution” workspace



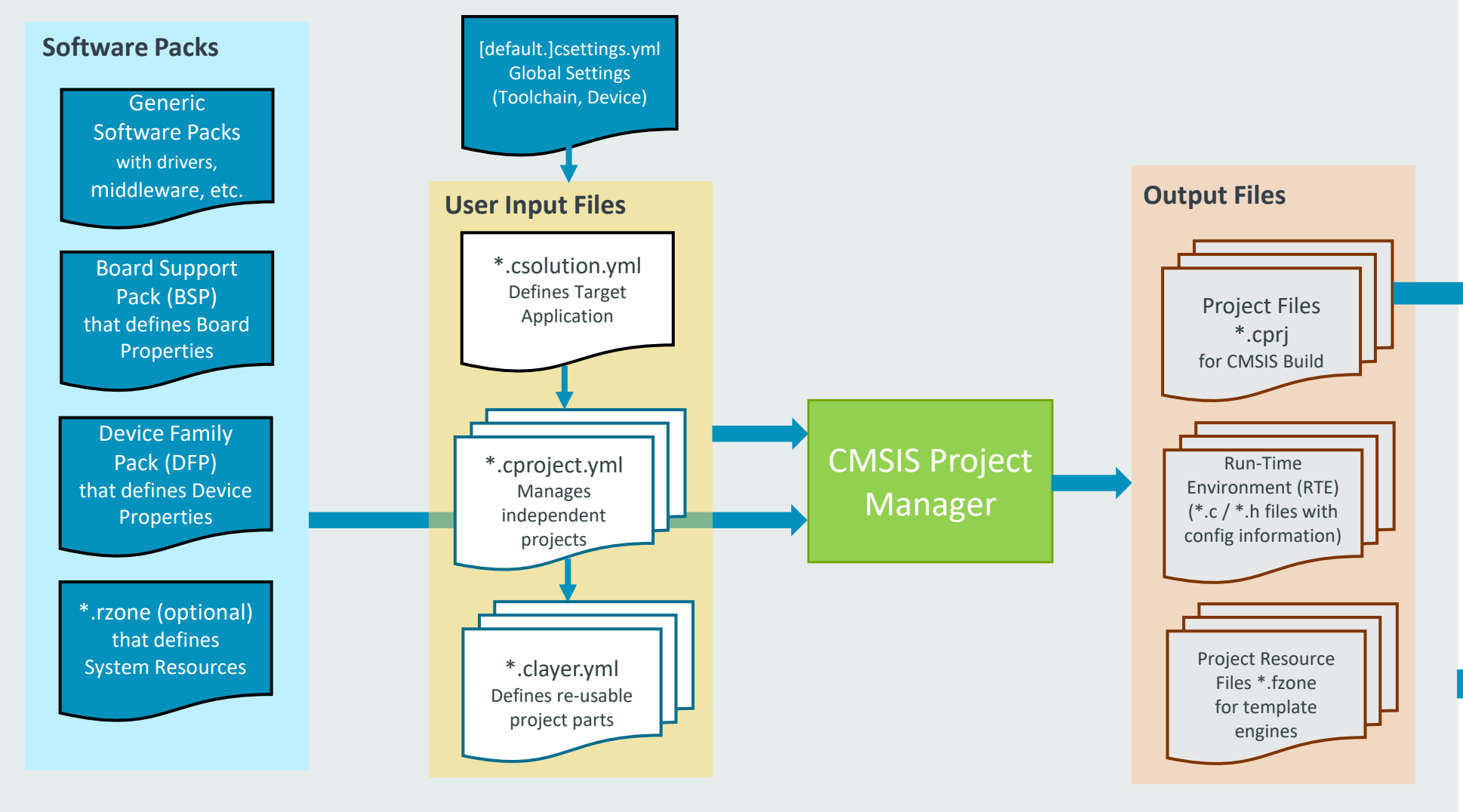
csolution: deployment to different targets for test automation

CI/CD environment for test automation – scale from Simulation to Hardware to Deployment



Essential the same event logs are generated across the different deployments. This ensures correctness.

csolution: CMSIS Project Manager



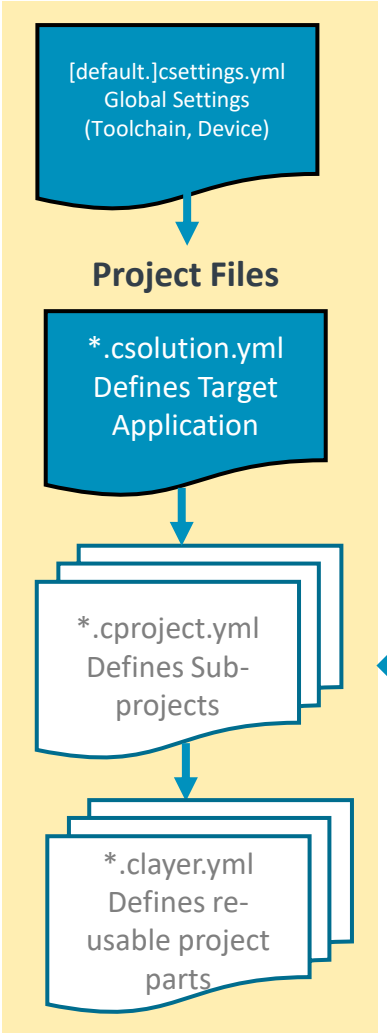
CMSIS-Build

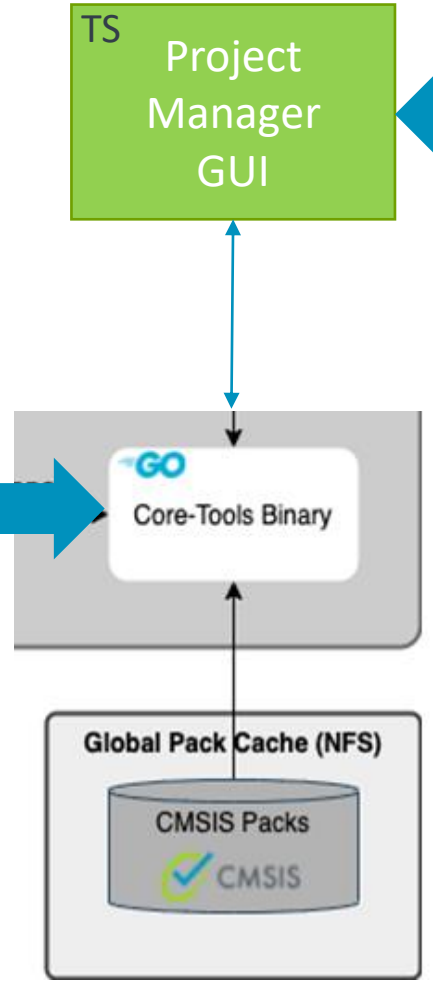
Linker Scripts

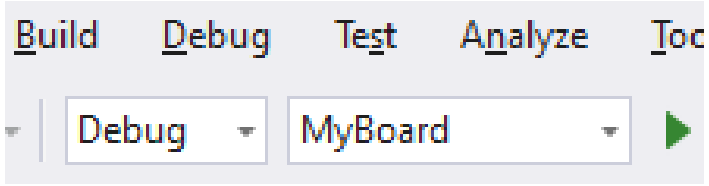
Via template engine

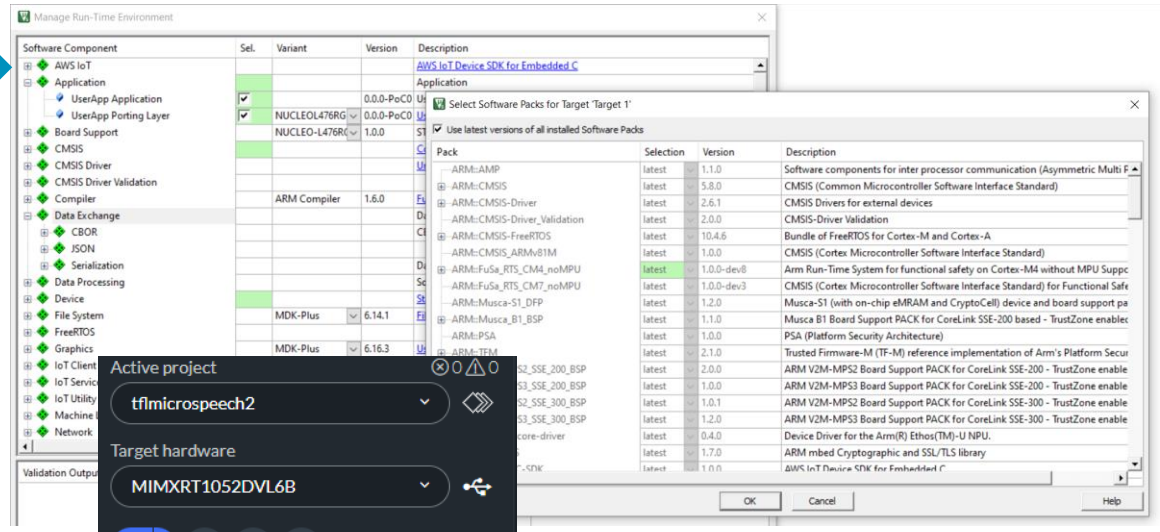
Initial solution uses pre-defined linker scripts

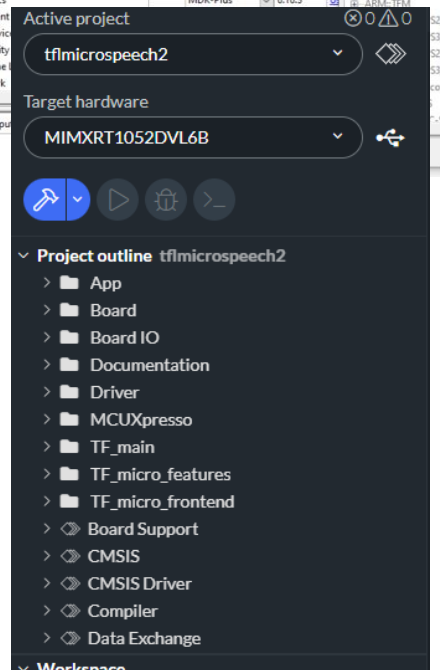
csolution: IDE integration











csolution – Project Example

my.cproject.yml

```yaml
project:
  compiler: AC6@6.16            # compiler (version optional)
  processor:                    # processor settings
    fpu: on                     # floating-point unit

groups:                         # file groups
  - group: My files
    files:
      - file: main.c
  - group: HAL
    files:
      - file: .\hal\driver1.c

components:                     # CMSIS software components
  - component: Device:Startup
  - component: CMSIS:Core

layers:
  - layer: Board-Interfaces.clayer.yml
```

MyApplication.csolution.yml

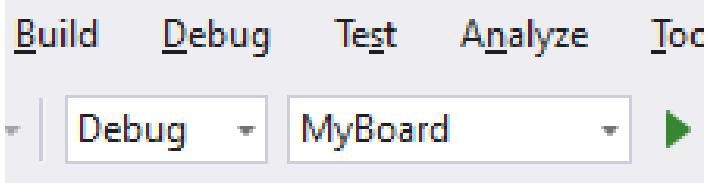
```yaml
target-types:
  - type: MyBoard               # user-selected name
    board: NUCLEO-U575ZI-Q      # board selects also device

  - type: MyDevice
    device: NXP::LPC55S69
  - type: Virtual Target
    device: Cortex-M7-VHT


build-types:
  - type: Debug                 # user-selected name
     . . .                      # tool chain options

  - type: Test                  # user-selected name
     . . .                      # or defines

solution:
  - project: my.cproject.yml  # multiple projects
  - project: trustzone.cproject.yml
```



Demo

OPEN-CMSIS-PACK PROJECT

Kyle Dando, NXP

SECURE CONNECTIONS
FOR A SMARTER WORLD

NXP OPEN-CMSIS-PACK PROJECT VISION

Need for a Standard Solution

- Efficient and consistent middleware delivery
- Extending capabilities with partner solutions

Strong Modularity

- Quickly add support for devices & boards
- Uniform model for Config tools, IDE & CLI

NXP’s Participation

- Integrating middleware Software Packs
- Collaboration on Tool development


```
<examples>
  <example name="CMSIS-RTOS2 FreeRTOS Blinky" doc="
    <description>CMSIS-RTOS2 Blinky example using F
    <board name="uVision Simulator" vendor="Keil"/>
    <project>
      <environment name="uv" load="Blinky.uvprojx"/
    </project>
    <attributes>
      <component Cclass="CMSIS" Cgroup="CORE"/>
      <component Cclass="CMSIS" Cgroup="RTOS2"/>
      <component Cclass="Device" Cgroup="Startup"/>
      <category>Getting Started</category>
    </attributes>
  </example>
```

```
project:
  components:
    - component: ARM::CMSIS:CORE
    - component: NXP::Device:CMSIS:LPC55S69_system
    - component: NXP::Device:SDK Drivers:clock
    - component: NXP::Device:SDK Drivers:common
    - component: NXP::Device:SDK Drivers:flexcomm
    - component: NXP::Device:SDK Drivers:gpio
    - component: NXP::Device:SDK Drivers:iocon
    - component: NXP::Device:SDK Drivers:lists
    - component: NXP::Device:SDK Drivers:power
    - component: NXP::Device:SDK Drivers:reset
    - component: NXP::Device:SDK Drivers:usart
    - component: NXP::Device:SDK Drivers:usart_adapter
    - component: NXP::Device:SDK Utilities:assert_lite
    - component: NXP::Device:SDK Utilities:debug_console_li
    - component: NXP::Device:Startup

  groups:
    - group: source
      files:
        - file: ./source/hello_world_s.c
        - file: ./source/LPC55S69_cm33_core0_flash_s.scf
```

life.augmented

ST contribution to Open-CMSIS-Pack

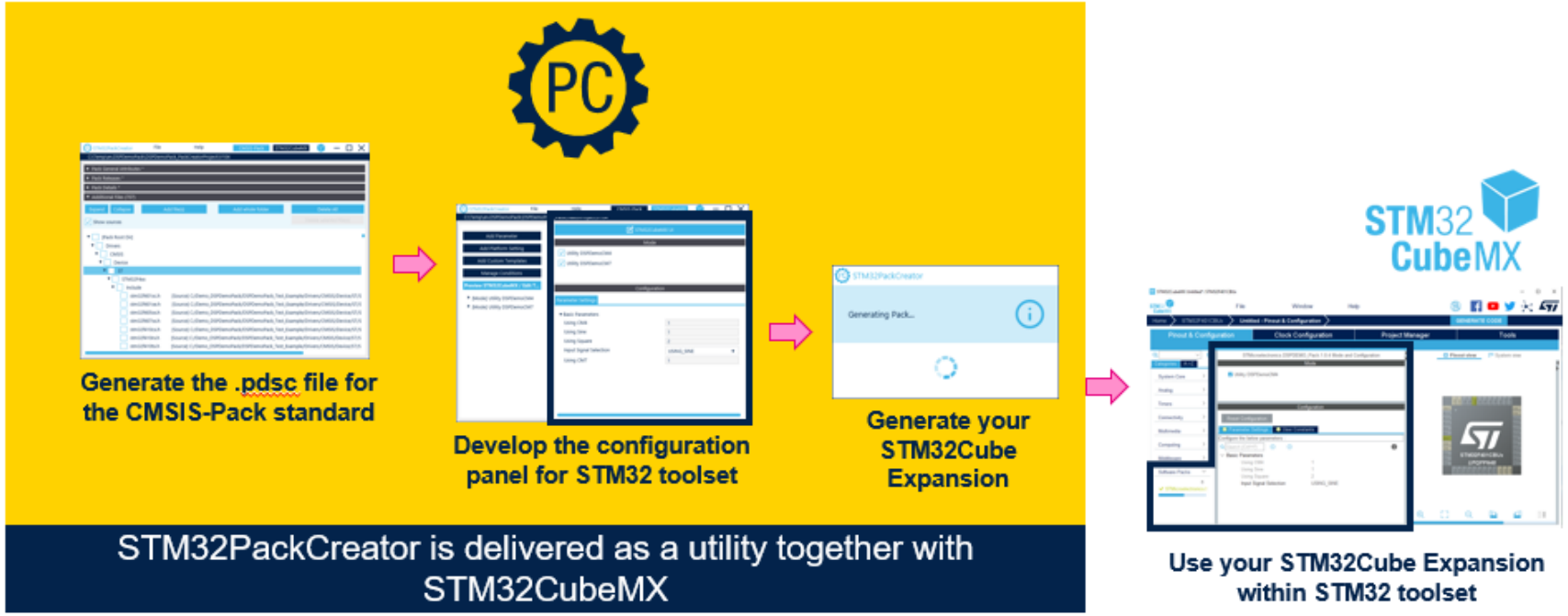
ST/MCD

Eric Finco, ST

STM32 and Open-CMSIS-Pack

- **CMSIS-Pack had been recognized as industry standard for software packaging**
  - Adopted in STM32Cube Ecosystem since 2017
  - Foundation of STM32Cube Expansions architecture
  - STM32PackCreator tool easing creation CMSIS-Pack based expansions

STM32 CubeExpansion



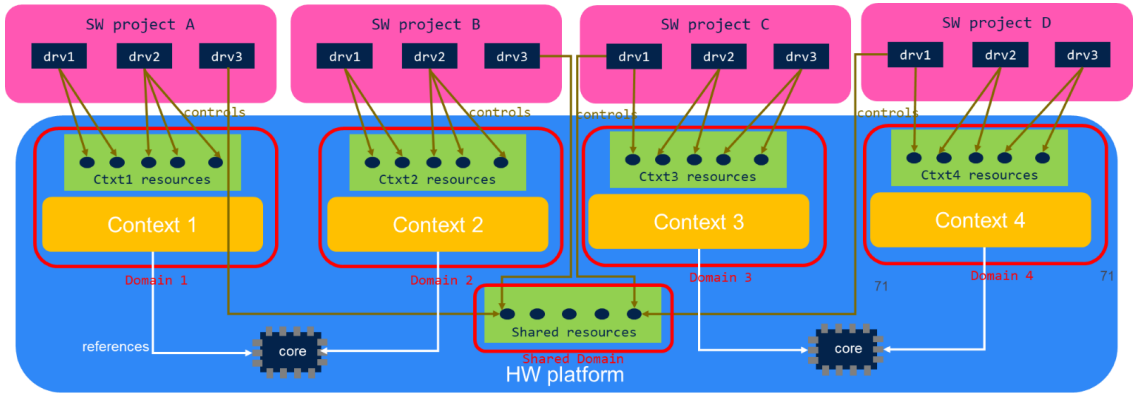
life.augmented

STM32 and Open-CMSIS-Pack

- **CMSIS-Pack had been recognized as industry standard for software packaging**
  - Adopted in STM32Cube Ecosystem since 2017
  - Foundation of STM32Cube Expansions architecture
  - STM32PackCreator tool easing creation CMSIS-Pack based expansions

- **Open-CMSIS-Pack is the evolution of CMSIS-Pack, we aim to:**
  - Enlarge the Open-CMSIS-Pack adoption
  - Extend the standard to support more advanced configurations
  - Extend the standard to add custom features

Open-CMSIS Pack



life.augmented

Open-CMSIS-Pack – Eco-system integration benefits

Compiler Support in CMSIS Toolbox

Add *.cmake files to reflect supported compiler (versions)

```
Directory of C:\ctools\etc

03/10/2022  01:44 PM            3,622 AC5.5.6.7.cmake
03/10/2022  01:43 PM           13,615 AC6.6.16.0.cmake
03/14/2022  01:38 PM            6,339 GCC.10.2.1.cmake
03/14/2022  01:38 PM            4,325 IAR.8.50.6.cmake
```

Pack with Board Support

Available for tools

Contains examples

Debug configuration

EVK-MIMXRT1064_MDK NXP

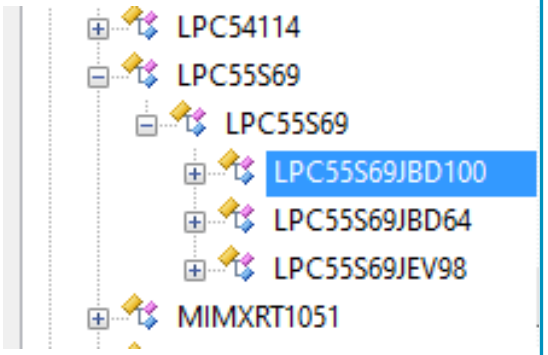
EVKB-IMXRT1050_MDK NXP

Device Family Packs

Defines device parameters

Easy to access by tools

Controlled by silicon vendor



Pack with Software Components

Can be derived from repositories
> github.com/arm-software/cmsis-freertos

One pack supports multiple IDEs and CLI environments

When correctly structured, can work with any device

arm

John Thompson

Challenges in IoT deployment

Highly fragmented MCU software ecosystem

Portability

How to port SW across different HW platforms

Cloud Connectivity

How to connect with ease to any cloud

Firmware Update

How to securely boot and update devices in the field

Outreach

How to deploy software to developers at scale

Security

How to achieve all of this securely

Project Centauri

Secure firmware update, for any IoT software stack running on Cortex-M devices

Driving the Cortex-M software ecosystem

- Firmware update and security primitives on every device
- Allowing IoT applications to runs on a broad range of physical devices as well as virtual hardware
- Free choice of which IoT software stack or RTOS works best for your application
- With reference software delivered to developers with rich tools based on Keil Studio and Open-CMSIS-Pack

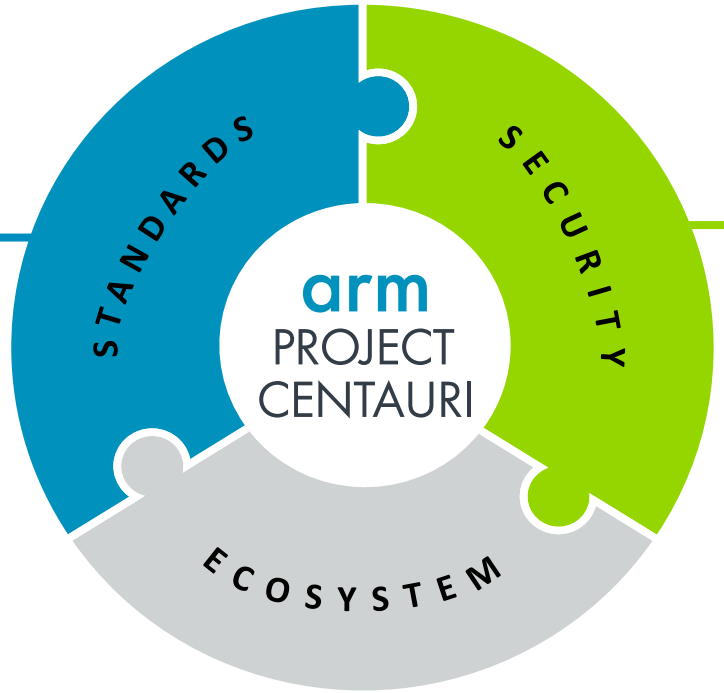
Project Centauri

Open-CMSIS CDI

Open-CMSIS Pack

Foundational Standards

- Cloud service-to-device specification
- Boot and firmware update
- Packaging and delivery standard



psacertified™

Device Security

- Defining how to implement security
- PSA Functional APIs
- TF-M, Mbed TLS, MCU Boot

Ecosystem Engagement

- Deployable Reference Implementations (IoT-SDK)
- Rich catalog of third-party software packs
- Certification program

arm

Use cases

What problems does Centauri address?

Firmware update	Secure boot	Initial version
	Secure update for whole of device firmware	Initial version
	Partial/delta secure update	Future
	Secure update of individual components (ML assets, individual components)	Future
Hardware independence	Application portability across different physical devices	Initial version
	Application portability between virtual and physical targets	Initial version
Software Portability	Application portability between RTOSes (FreeRTOS, AzureRTOS, etc)	Initial version
	Application portability between IP-based Connectivity technologies (WiFi, Ethernet, Matter, NB-IoT)	Initial version
	Application portability between non-IP Connectivity technologies	Future
	Application portability between IoT services (AWS, Azure, etc)	Initial version
Developer Experience	Access to ecosystem of software components delivered in a consistent way	Initial version

Core Deliverables

Standard APIs

Open-CMSIS-CDI is a standard set of interfaces for essential device capabilities and security

- Supporting secure firmware update
- Enabling major IoT stacks to run on different hardware
- Based on existing, adopted standards
- Curated by Arm, evolved in the open based on feedback from major industry partners
- Allow meaningful platform differentiation

Reference Implementation

A reference implementations of these standards known as the **Open IoT-SDK**

- Delivered under permissive, open-source licenses
- Pre-integrated with major IoT stacks
- Proven through deployments in Arm Total Solutions
- Providing a foundation for further innovation by partners

Certification Program

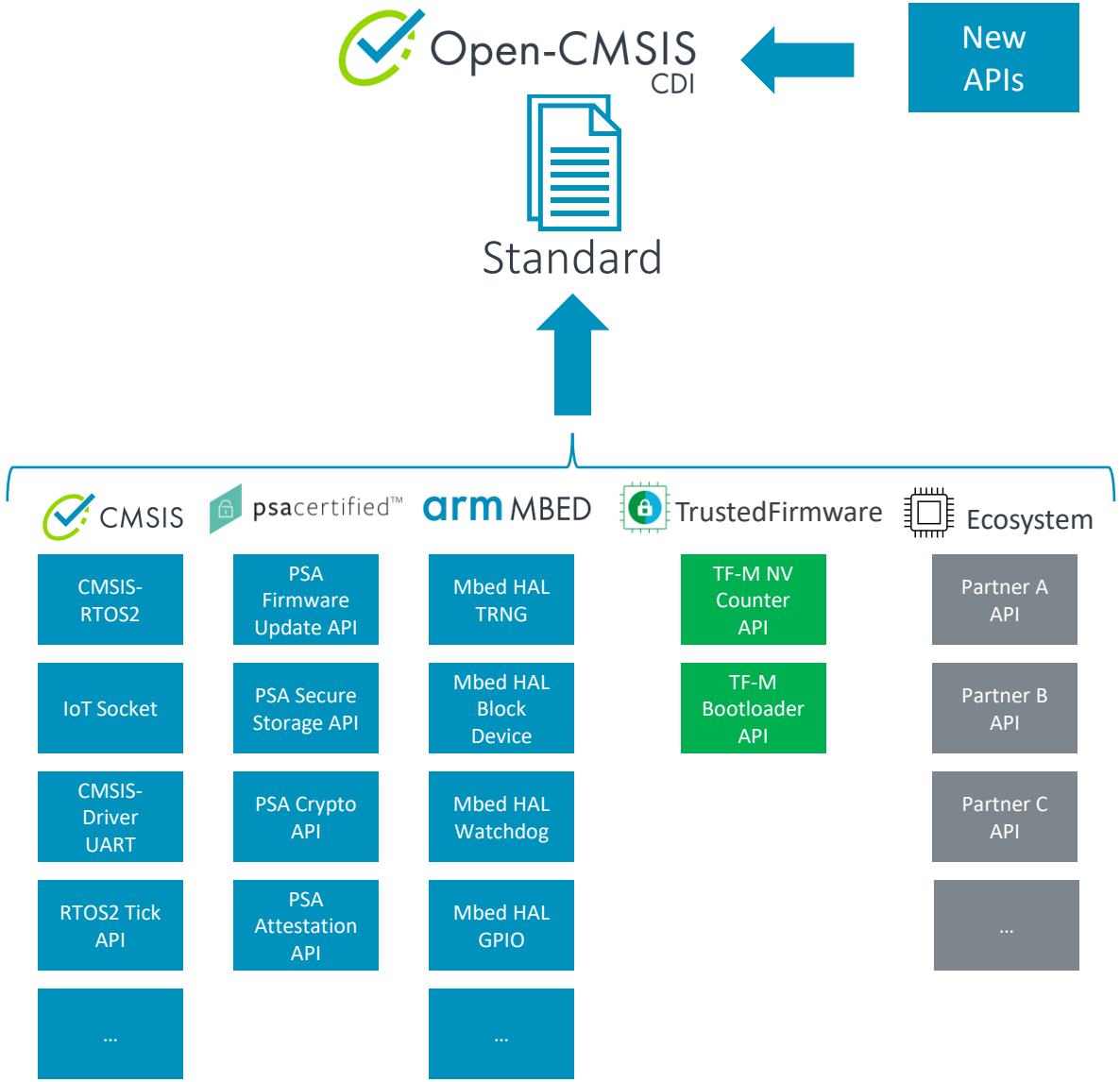
A compatibility and/or certification program run with partners to

- Promote platforms which offer secure device management through full lifecycle
- Provide developers with confidence about the software they rely on
- Based on freely available testing tools

Open-CMSIS-CDI

The Common Device Interface (CDI)

- Cloud service to device software specification
- Secure firmware update for any IoT software stack
- Delivered with a full reference implementation
- Starting with established APIs and standards from Arm and our partners
- Complements the Open-CMSIS-Pack project with common APIs
- Partner collaboration will help evolve the standard to support more use cases

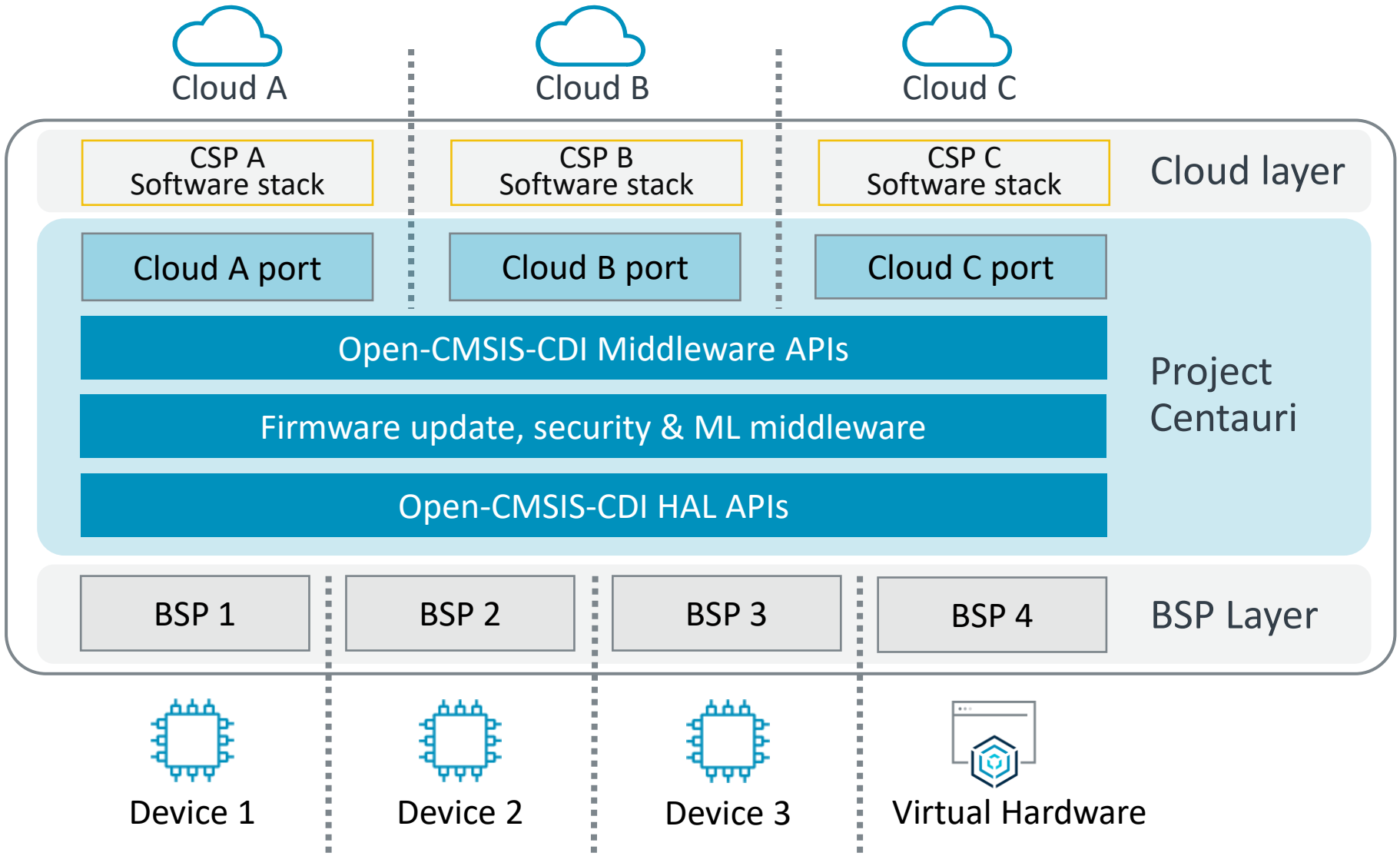


Open-CMSIS-CDI – An open standard

Developed by an open community

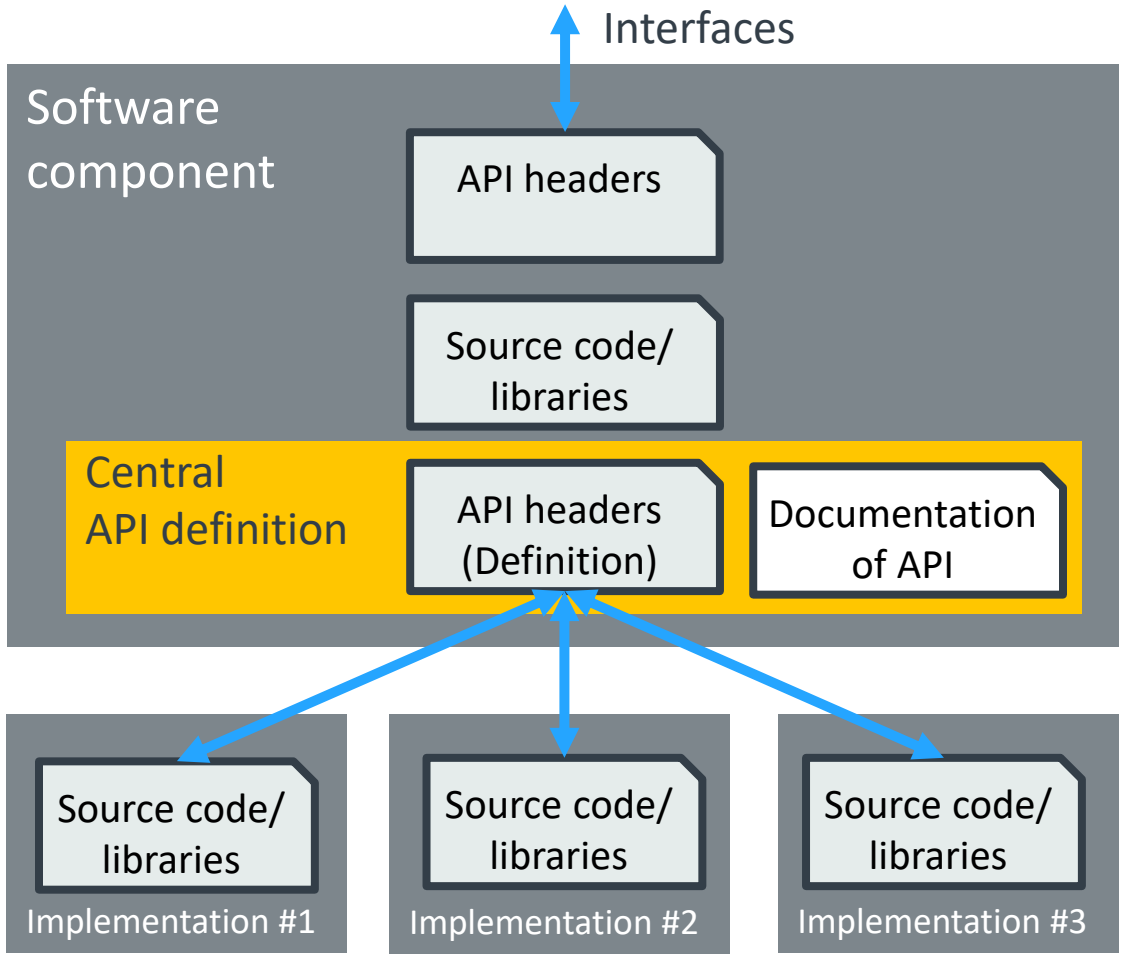
- Open-CMSIS-CDI will be developed as an open standard, managed by a community project
  - Arm's initial proposal is a starting point from which to build
  - Partner inputs are essential to ensuring we solve the right problems and address the key use-cases
- Facilitated by Linaro, Open-CMSIS-CDI will be run along the same lines as Open-CMSIS-Pack
  - Technical working groups on all the relevant topics
  - Backlogs maintained in public GitHub repos
  - Public meetings, free for anyone to attend, with minutes published in the open
  - Development in the open: with standards, documentation and tests available under permissive open-source licenses
- We expect to launch the first technical meetings in April
  - For more information, e-mail cmsis@arm.com
  - If you're interested in participating, please contact Arm or Linaro and we will inform you once the calls start

High-Level Architecture



CDI-Pack: Central API Interface definition in CMSIS-Pack format

Ensuring consistent interfaces and naming taxonomy across the industry



- Organizes the taxonomies of standard APIs that are essential for re-useable software stacks
- Solves a common problem: API headers evolve over time.

A central API definition shares header file and documentation of an API interface across multiple other software components to ensure consistency.

The API interface is distributed separate or as part of the software component that defines this interface. The API header file is therefore consistent.

An example is the CMSIS-Driver pack that contains various Ethernet and Flash drivers – all compatible with the CMSIS-Driver APIs that are published in the CMSIS Pack.

Timeline

April 2022	July 2022	October 2022	Q1 2023
Working Group Kick-off	Draft CDI Specification	CDI Specification Public Availability	General certification program ready for Embedded World'23
• Working groups on standards with lead partners • First meeting scheduled for April 21st • Alignment on working structure and goals	• Candidate APIs identified • Taxonomy for CMSIS-Pack system proposed • Reference implementation preview available	• Public launch of the specification with public endorsement of partners • Software running on virtual HW and physical boards	• General certification program ready to begin in early 2023 • Deliberately staggered behind launch of initial platforms with lead partners

Call to Action

Help set the CDI direction

- Contact Arm to join working group
  - cmsis@arm.com
- Join working group on April 21st to:
  - Review Arm's initial proposal
  - Propose your own API requirements
  - Submit "wish list" to Arm to help guide initial direction

CMSIS-DAP v2.x

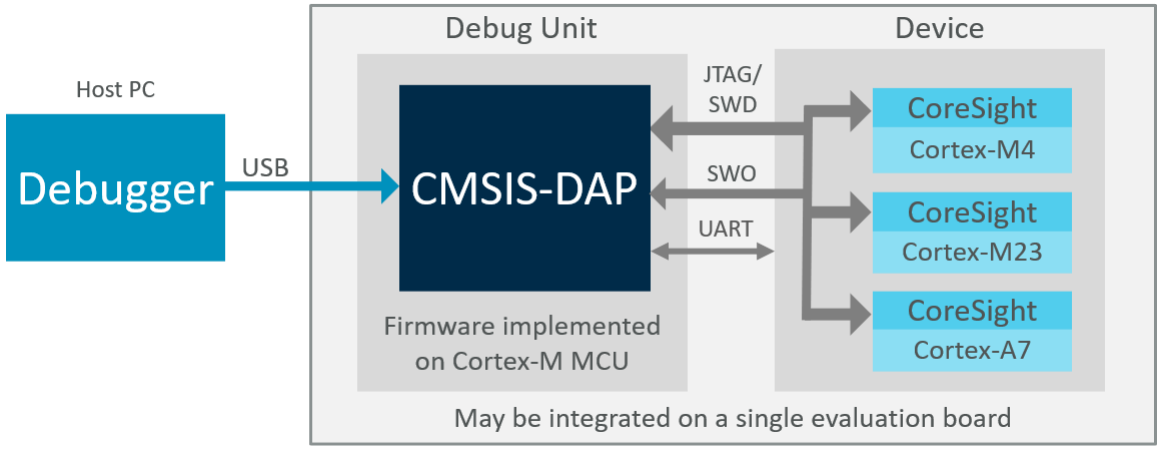
Speed improvements and integration of Event Recorder for CI and MLOps workflows

Christopher Seidl

CMSIS-DAP: Standardized firmware for debug adapters

Specification and implementation of firmware to access CoreSight DAP

- CMSIS-DAP provides a standardized interface for debuggers and supports multi-core debugging.
- Provides easy and low-cost integration of a debug unit may on an evaluation board.



- CMSIS-DAP v2.1:
  - Debug UART (printf) via USB COM or CMSIS-DAP command interface
  - Board Identification in firmware
  - Example firmware implementations
- Future: Support for CoreSight ADIv6 debug protocol and first-stage capture for Event Recorder

Board identification for generic CMSIS-DAP firmware

Assigns a Board ID to existing firmware – no need to re-compile

- Create generic DAP FW for the circuit you are using on dev kits
- Info in Board Pack - `NXP.LPCXpresso55S69_BSP.pdsc`

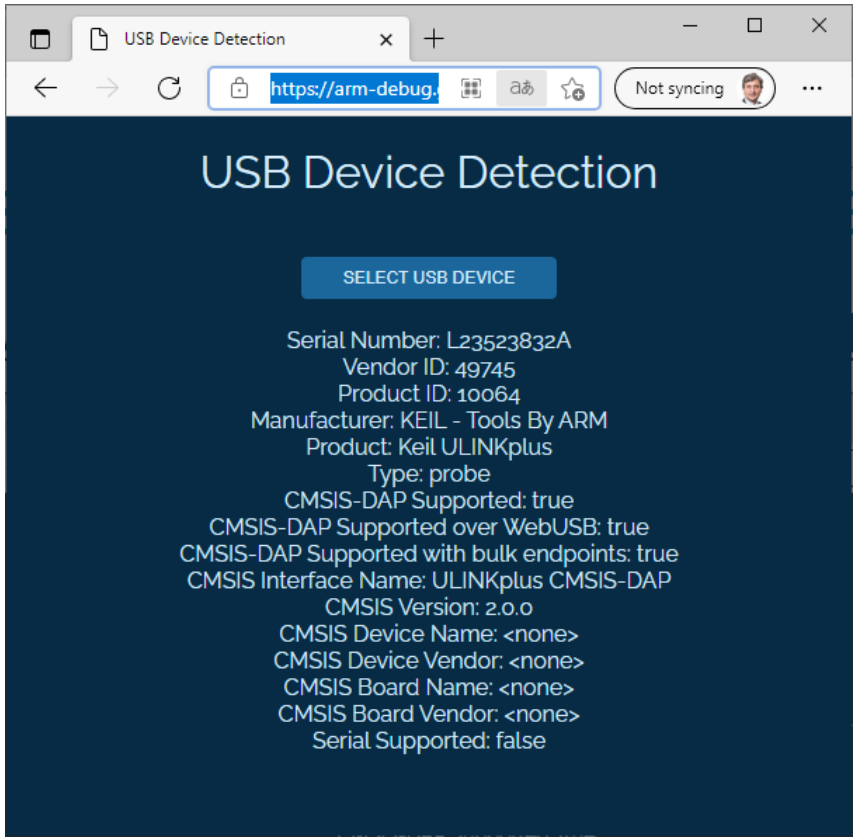
```
<board vendor="NXP" name="LPCXpresso55S69">
 …
    <mountedDevice Dname="LPC55S69" Dvendor="NXP:11"/>
 …
</board>
```

- Create patch file – `LPCXpresso55S69.patch`:

```
# symbol            : string/values
TargetBoardVendor   : "NXP"
TargetBoardName     : "LPCXpresso55S69"
TargetDeviceVendor  : "NXP"
TargetDeviceName    : "LPC55S69"
```

- Patch firmware:
  `patchELF CMSIS_DAP.axf LPCXpresso55S69.patch`
- Program patched CMSIS_DAP firmware on specific development board

- Validate deployment to board
  https://arm-debug.github.io/device-detection/



CMSIS-DSP/NN update

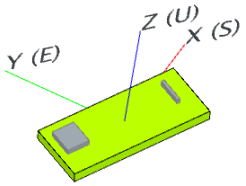
Laurent Le Faucheur, Senior Principal Engineer

CMSIS for DSP + ML in IoT and automotive markets

- Endpoints are implementing AI using a **combination of DSP and ML**
- Implement **Classical-ML + DSP** in *CMSIS-DSP*, and **Neural Networks** in *CMSIS-NN*

Typical processing flow for endpoint AI in vibration, voice and vision domains

- **Key next steps**: extend current kernels with all data types (including fp16, fp64) and add **computer vision kernels**
- There is a separate repository to verify and share new concepts :



Status and plans (1Q22)

Key outcomes on DSP and ML kernels from last year :

Enhanced PythonWrapper to ease transition from Python NumPy/SciPy to a CMSIS-DSP C implementation
A graph scheduler generator "Synchronous data-flow"
CMSIS-DSP extended for F16 and F64 formats
Arm-2D pixel processing (draw, fill, rotate, alpha-blending) is integrated in LVGL
Optimizations of FFT, complex convolution and filtering for MVE, int16 x int8 matrix multiply, beamformer and LC3-specific kernels
CMSIS-NN +30% performance on Cortex-M0/M3 on convolution operators
Integration with Apache TVM, add SVDF operator support

CMSIS-DSP outreach activities :

Cortex-M55 Programmer's Guide: migrating from Neon
System Modeling with Arm Virtual Hardware and OpenModelica
DSP online conference presentation of CMSIS. Member of the EEMBC for AudioMark definition with CMSIS kernels to come.

Next CMSIS DSP & ML computing kernels :

Computer vision kernels (always-on detector, filters, statistics, lens correction, contour detection, image alignment, etc ..)
New linear algebra kernels. MVE-optimized release of LibSpeex.
Stream-based processing scheduler of software components
Develop new use-cases/system modeling using OpenModelica and AVH
CMSIS-NN : Library size reduction when used with TensorFlow Lite Micro.

Summary and Questions

Reinhard Keil

Project Centauri

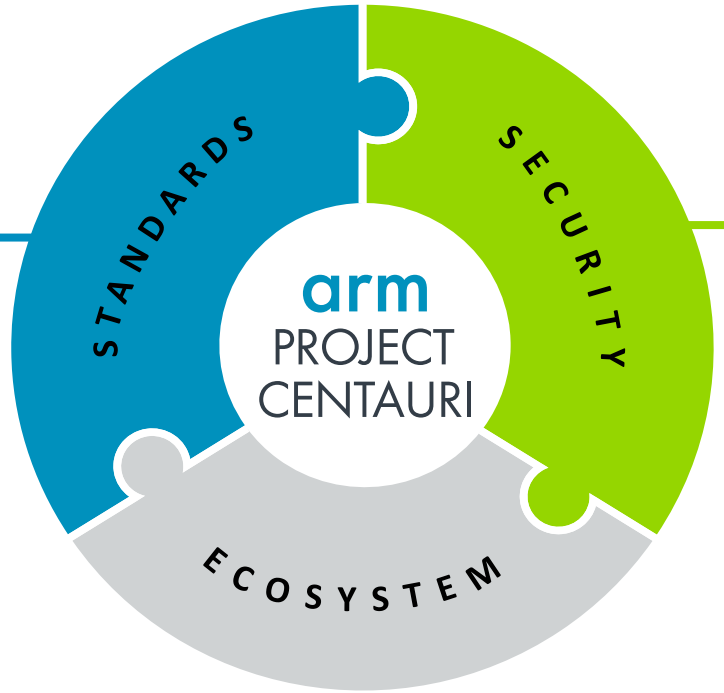
Our goal - Simplify IoT and ML for microcontrollers

- Open-CMSIS-CDI implements the interface requirements for Cloud and ML stacks
- Arm Virtual Hardware (AVH) simulates all CDI interfaces; validate IoT/ML software stacks with CI/CD
- CDI drivers are provided by Arm for Cortex-M AVH test platforms; adopted by silicon vendors for devices
- Tools (with IDE and CLI flows) make it easy to switch from Arm Virtual Hardware to physical hardware

Open-CMSIS CDI Open-CMSIS Pack

Foundational Standards

- Cloud service-to-device specification
- Boot and firmware update
- Packaging and delivery standard



psacertified™

Device Security

- Defining how to implement security
- PSA Functional APIs
- TF-M, Mbed TLS, MCU Boot

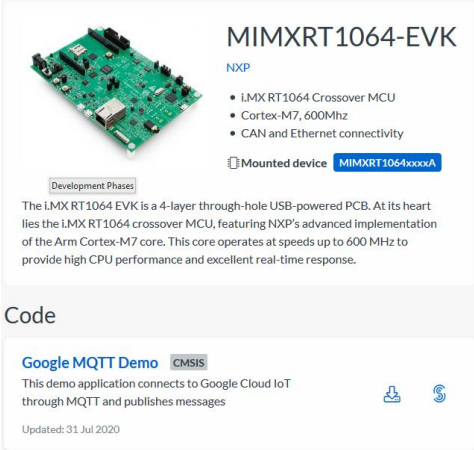
Ecosystem Engagement

- Deployable Reference Implementations
- Rich catalog of third-party software packs
- Certification program

arm

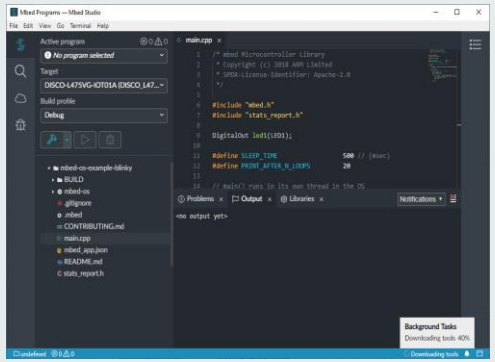
The CMSIS eco-system – flexibility for developers

DISCOVER POSSIBILITIES → EXPLORE REFERENCE DESIGNS → DEVELOP APPLICATION → COMMERICAL TOOL SUPPORT



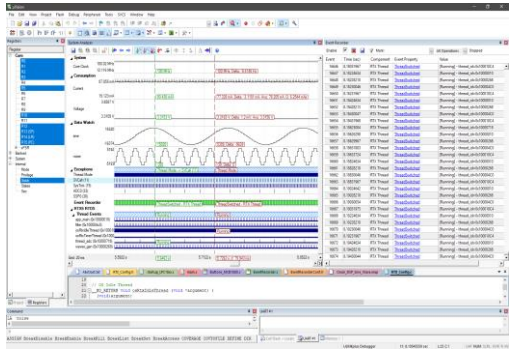
Enter parameters of your application

- Compare devices
- Evaluation boards
- Reference code examples



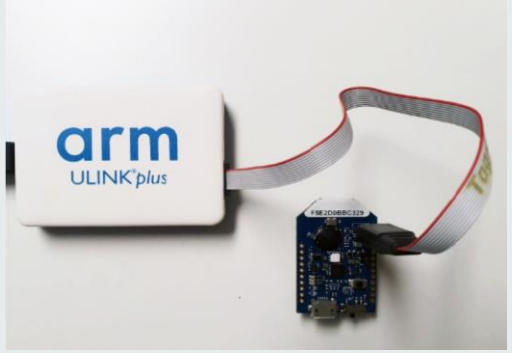
Use online tools for evaluation

- Explore code
- Zero installation hassle
- Always up to date



Wide range of tooling

- Supported by many IDEs
- Command-Line workflow with CMake backend
- Desktop and cloud workflows



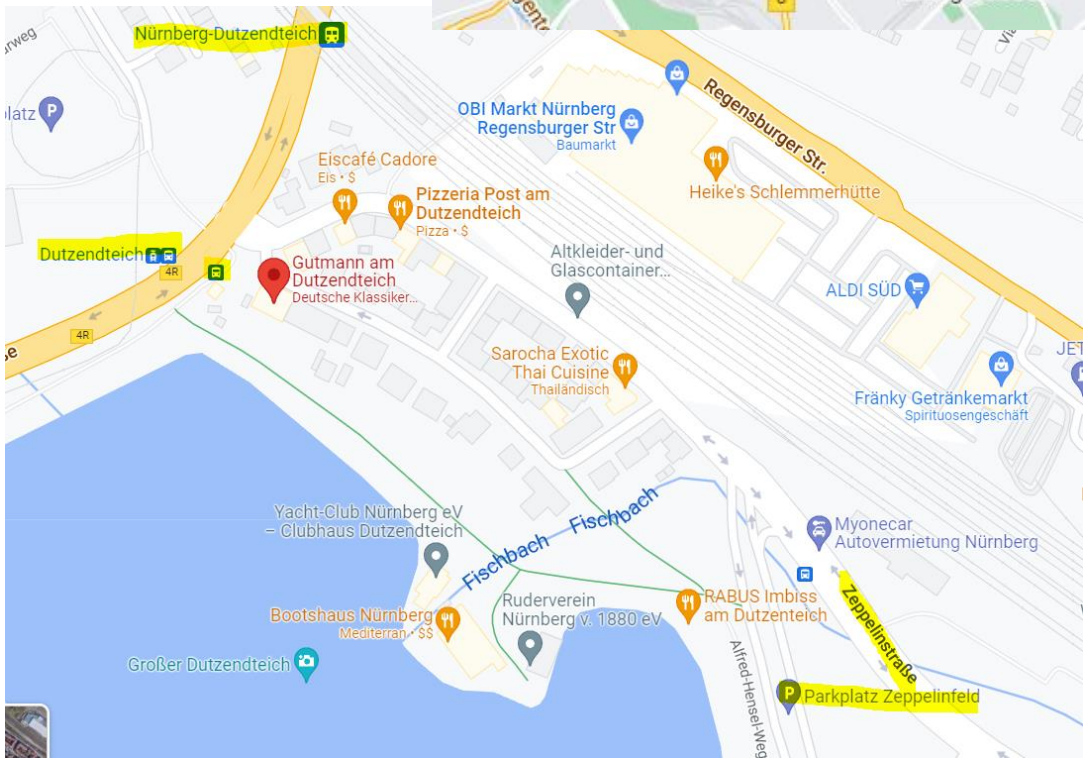
Mass production

- Supported by commercial development tools
- Compatible with Arm Virtual Hardware for CI

CMSIS Meeting at Embedded World

Monday, June 20, 16:30 • In-person meeting

- Progress review of CMSIS
- Collaborate and discuss common problems
- Initiate a creator program



Where: Gutmann am Dutzendteich,
Bayernstr. 150, 90478 Nürnberg
www.gutmann-am-dutzendteich.de/anfahrt.html

By Car: please use the parking area at Zeppelinfeld

Public transport:
Train station „ Nünberg-Dutzendteich“ or
Bus station „Dutzendteich“

- **To sign up (deadline June 6th), send email to cmsis@arm.com**

Get Involved

- Sign-up as lead partner by email to cmsis@arm.com for Open-CMSIS-CDI

- Participate on Open-CMSIS-Pack and Open-CMSIS-CDI Technical Meetings
  - open-cmsis-pack.org
  - open-cmsis-cdi.org

- Review progress on classic CMSIS github.com/arm-software/cmsis_5

- Meet with us at Embedded World, register by email: cmsis@arm.com

arm

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה