Generating CMSIS-Packs for Middleware

How to create scalable software packs to maximize software re-use

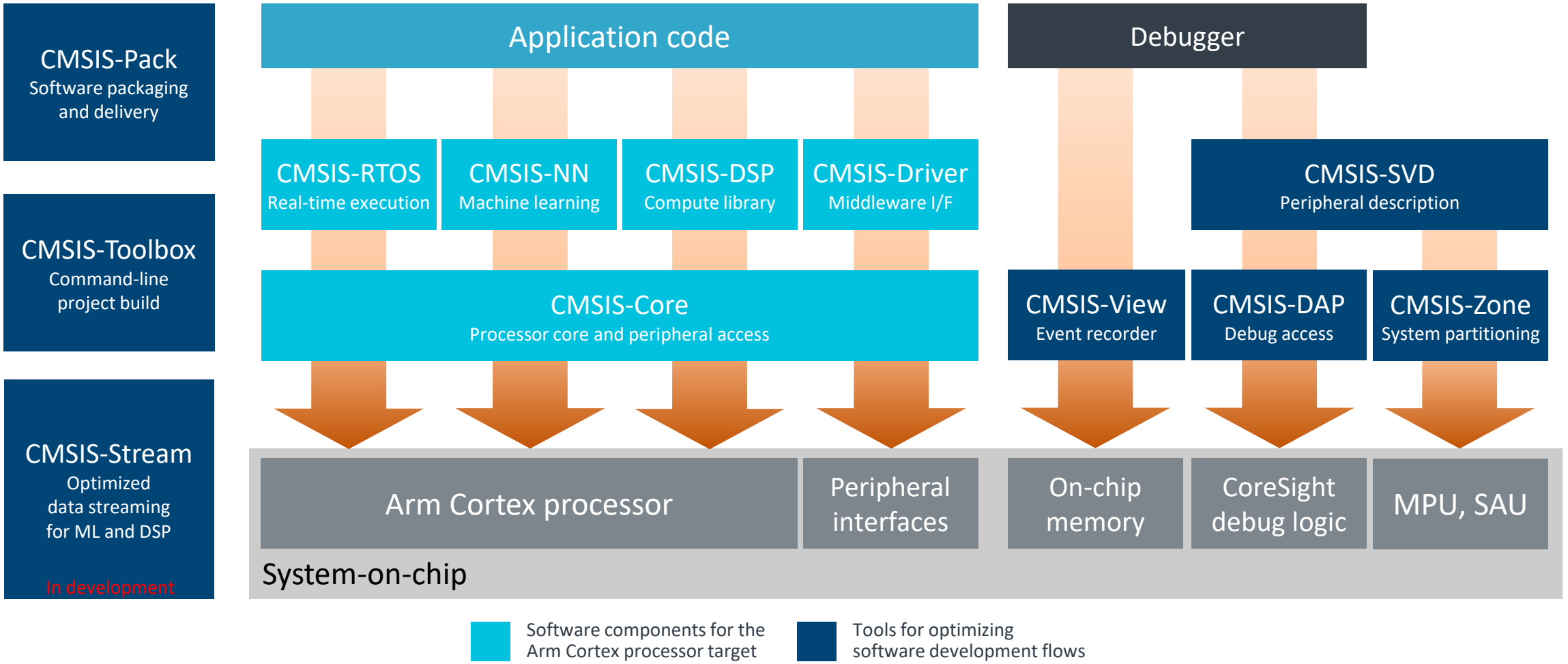
Linaro WG Meeting

Arm MCU Tools Team

18 April 2023

CMSIS Version 6

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



https://community.arm.com/arm-community-blogs/b/tools-software-ides-blog/posts/which-cmsis-components-should-i-care-about

History and CMSIS-Pack Overview

CMSIS is a set of tools, APIs, frameworks, and workflows that simplify software re-use

- 2008: CMSIS-Core has been introduced and CMSIS has been since then continuously extended.
- 2012: CMSIS-Pack has been started to simplify PLM and improve overall user experience.
- 2014: CMSIS-Pack is part of MDK for device support, middleware; resulted in higher adoption and lower support.
- 2017: Eclipse version of CMSIS-Pack system and integration of CMSIS-Pack system in IAR toolchains.
- 2020: Discussions with ST and NXP resulted in Open-CMSIS-Pack project and VS Code integration.

CMSIS-Pack Overview

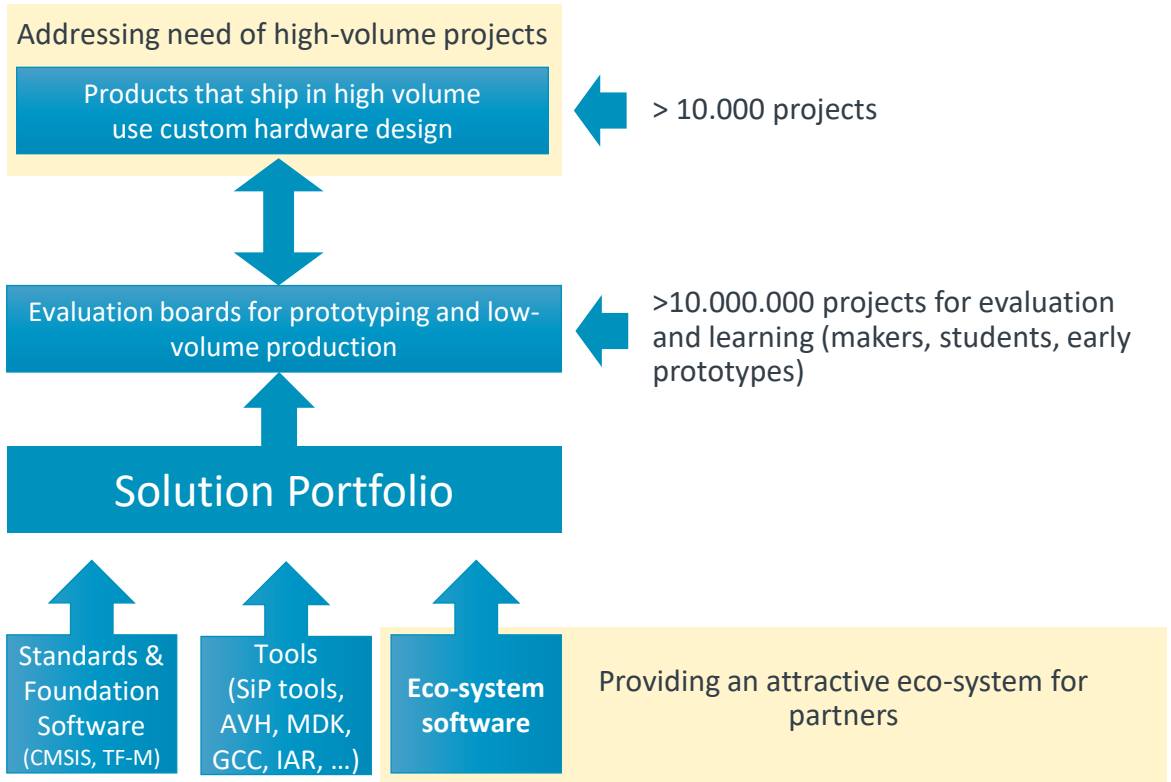
Delivery	Software components, examples, code templates, documentation, device and board support files.
Versioning	Semantic versioning for lifecycle management using an industry standard for reliable production.
Dependency	Specify dependencies upon other packs, software components, toolchains, and APIs.
Retargeting	Select files based on hardware selection or toolchain requirements.

arm

What are the care abouts of our target audience?

MCU designs care about cost; software reuse is key for productivity and quality

It's the software that takes the time.



Re-useable software components with standardized interfaces:

- Allow integration into many different software projects.
- Use established verification and validation development processes that are independent of final target hardware.

Frequently machine learning models are developed and trained in isolation of the final hardware target.

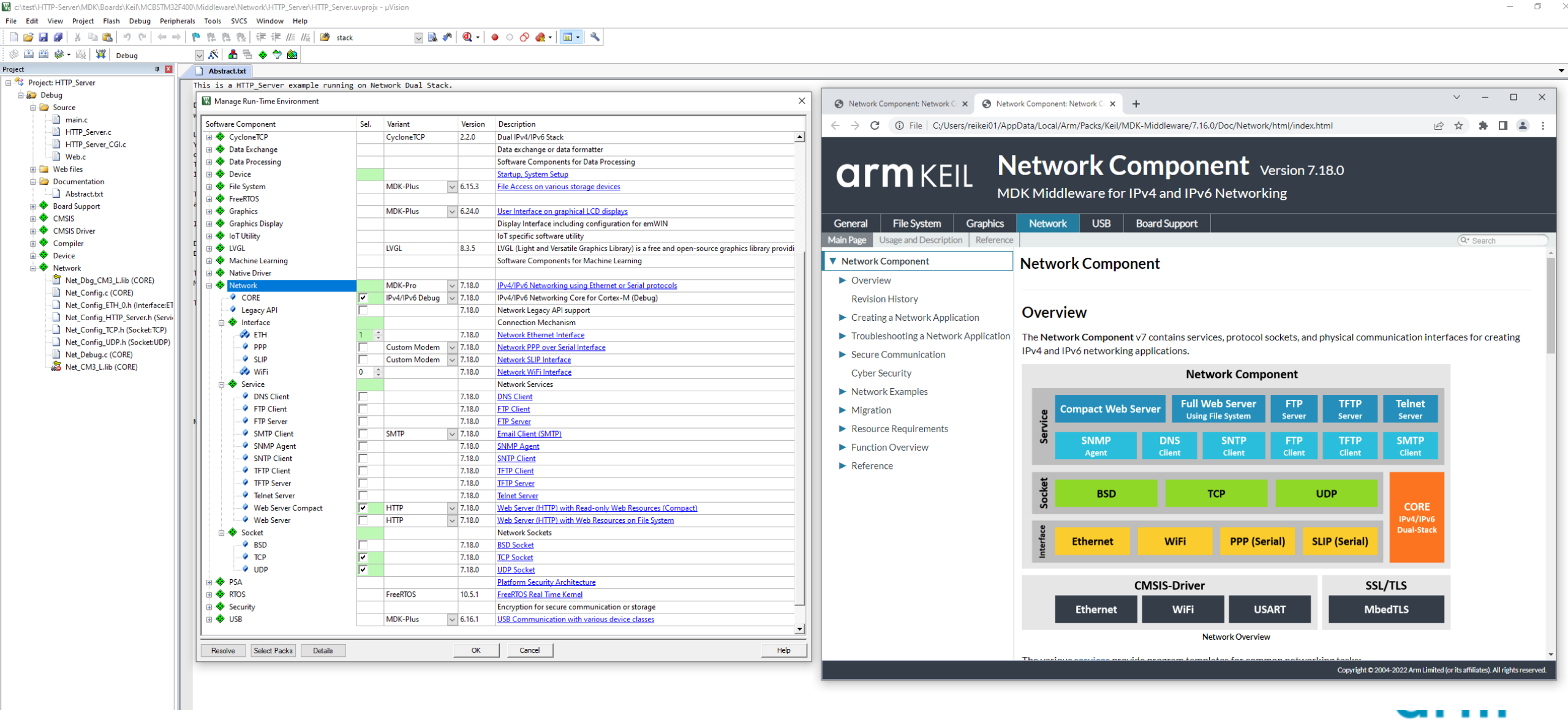
- Use MLOps workflows in the cloud with test and training data.

Big corporations re-use software across multiple projects with diverse development teams or external suppliers.

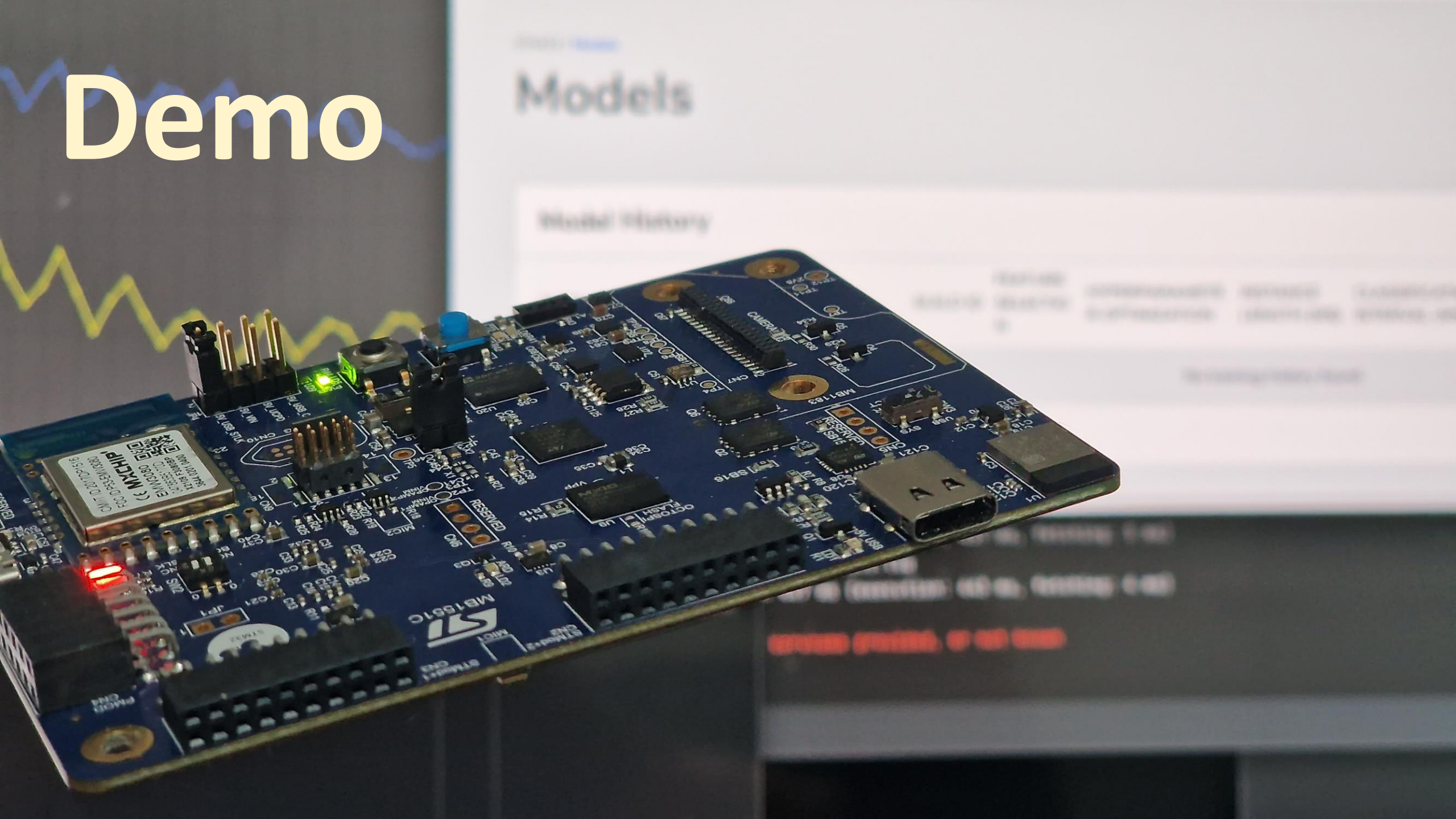
- Tools that enable code reuse are key, but we need to explain the usage.
- Therefore, tools should be complemented by methods and recommendations on how to structure software.

arm

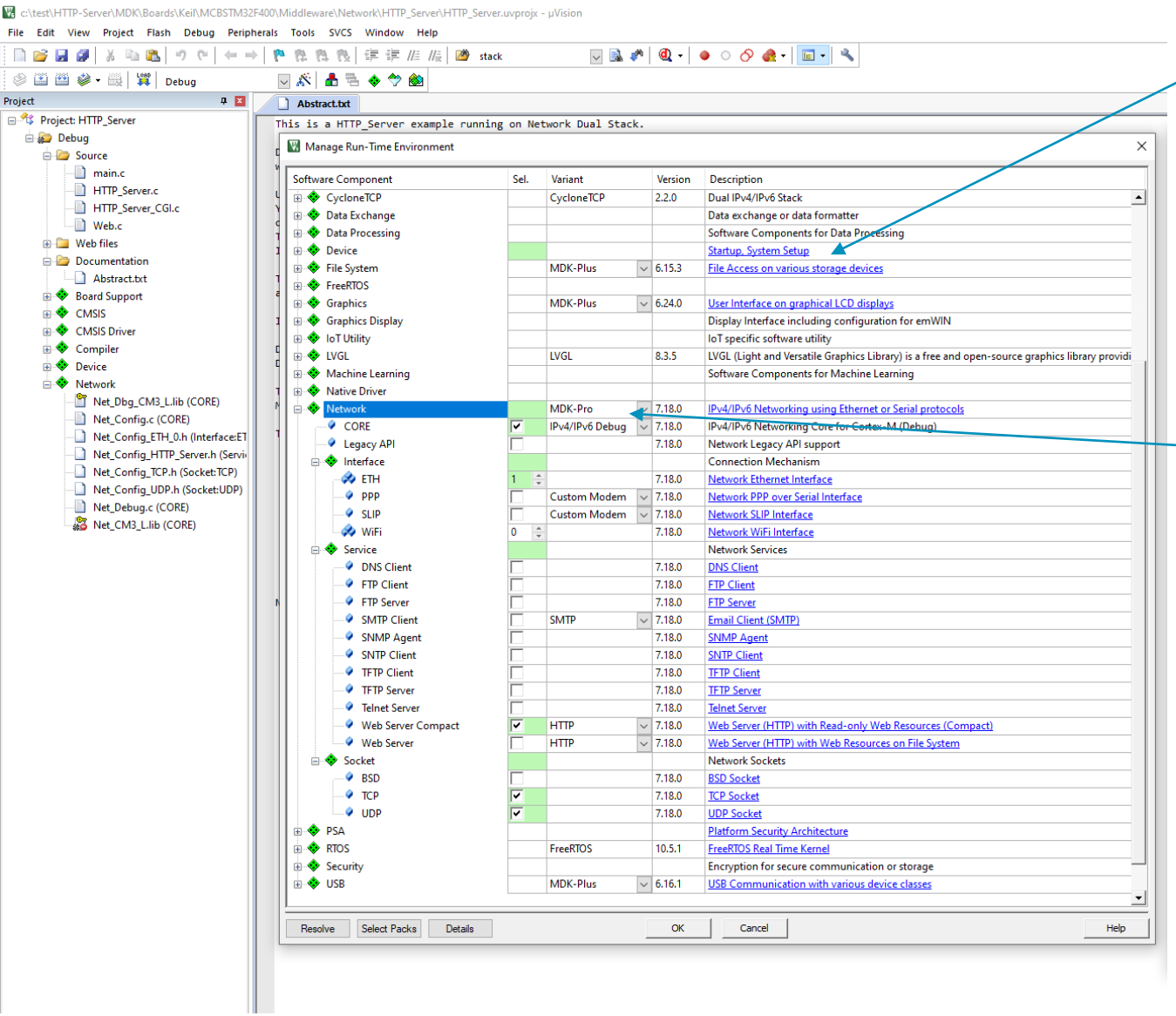
Usage example with Middleware



Demo



Connecting software components to business requirements



<taxonomy> of a `Cclass` can link to local documentation of web pages.

It is OK, to use product names as `Cclass`, i.e. FreeRTOS, but we will discuss in later meetings how to organize top-level `Cclass`.

<Cbundle> of a `Cclass` can link to local documentation or web pages.

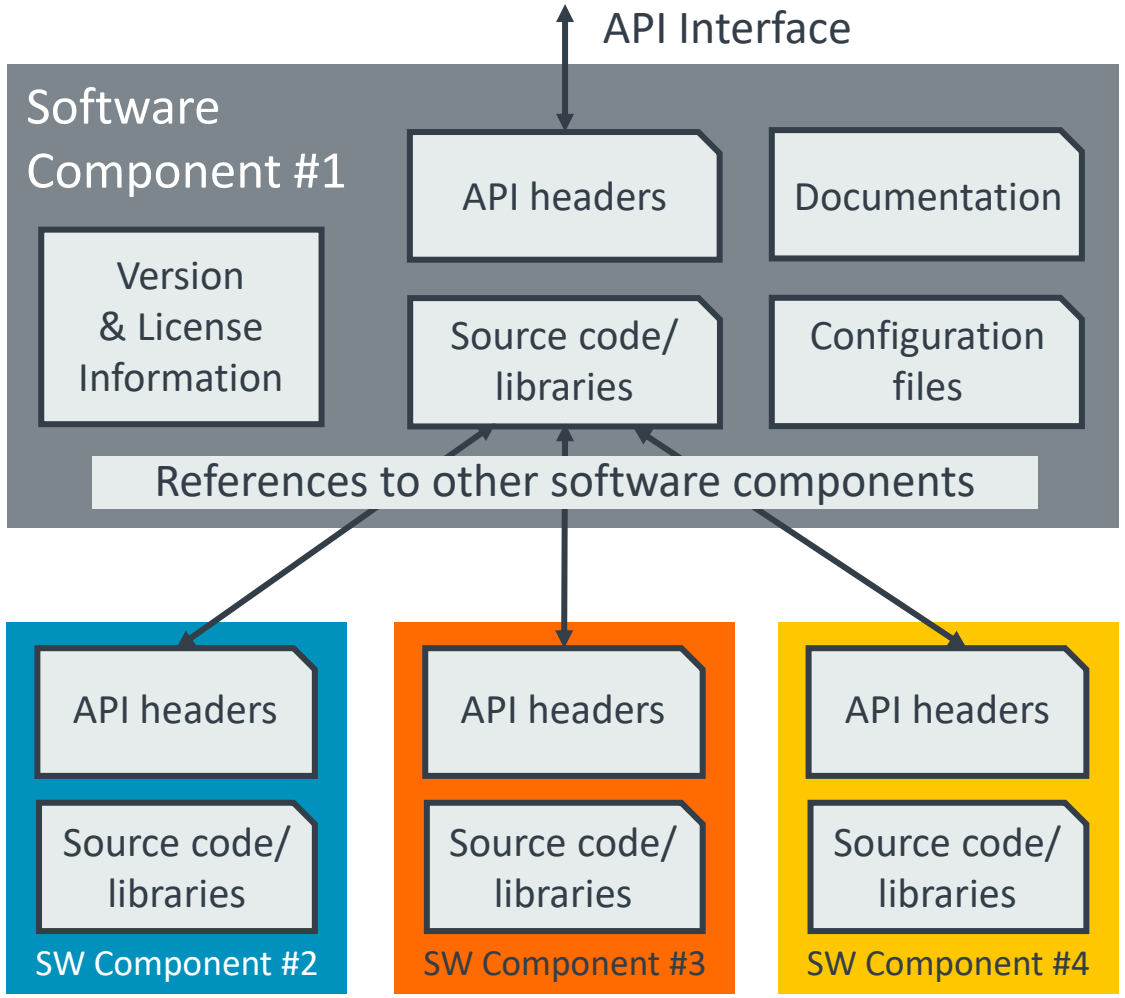
Packs **support your business needs**:

- Pack bundle `Eval` for evaluation.
- Pack bundle `Lite` with restrictions.
- Pack bundle `Full` for full featured version

NOTE: this can come from several packs

What is a Software Component?

XML framed information used by project management utilities from various tools



Software components should have:

- Semantic version, history, and license information
- API interface definition
- Documentation
- Source or library files
- Optional configuration files with sematic version
- Requirements to other components (optional)

CMSIS-Pack framed software is supported by:

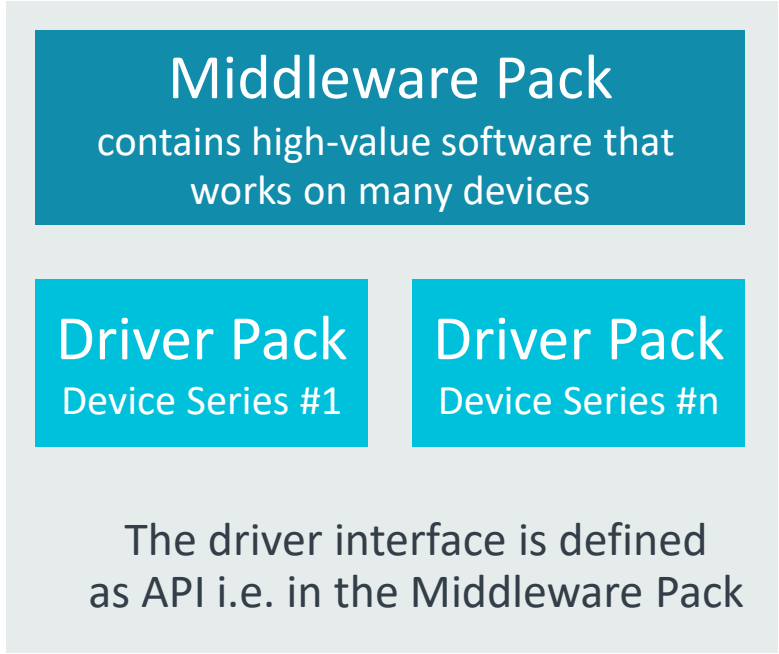
- Mainstream IDEs: Arm DS, Keil MDK, IAR EWARM
- MCU vendor IDEs based on Eclipse: ADI, OnSemi, STCubeMX, MCUXpresso
- Coming soon: VS Code plugins →
- Several web portals
- Open-source and command-line build tools

→ Scalable Software Stack.pdf

Steps to Generate a Pack

1. Structure your software
   - What are the functional blocks, are these usable separately? If so, consider separate components.
   - What are the interfaces to the device and/or hardware?
     - Are there existing interfaces in CMSIS? Take a look to CMSIS-Drivers and consider to provide feedback.
     - If not, consider to create an API to separate device from functional parts.
     - You may provide device-specific interfaces as part of your pack but consider a separate pack as the interface could be overwritten or extended with other packs.
1. Organize and create the file list that will be delivered as Pack
   - Each component can have source code, header and library files, documentation; separate the content logically.
2. Create the XML-based PDSC file using your favorite editor
   - Validate the XML code against the schema to find bugs early in the pack development stage.
3. Use the gen_pack library to create the pack file.
   - It runs many tests automatically and flags errors in the pack structure (missing files etc.).

→ github.com/Open-CMSIS-Pack/SW-Pack-HandsOn



arm

Next steps

Best Practices

- Use an editor with XML linting capabilities (e.g. VS Code with “XML Language Support by Red Hat” extension)
- Keep names short (this also refers to folder names)
- Use semantic versioning on your software component
  - Don’t forget versions for config files as this supports seamless upgrade of your software component for product lifecycle management (PLM)
- Think about your business needs and the scaling of your software to many different targets:
  - Use <api> when possible as it decouples middleware from hardware and helps to create a software stack
  - Use existing drivers APIs from CMSIS help us to improve or help to define new API interfaces
  - Use source code when possible (remove dependency on compiler/device), but you may also ship libraries
  - You may consider to offer compatible packs (eval, lite, full) version
- Watch our meeting Recordings: starting Tue 18. April (15:00 GMT)
  - **Embedded World – CMSIS 6 Replay** – gives a good overall overview of CMSIS
  - 4. April: Reference Example Framework – how we scale one example to many different targets
- When you have questions, reach out to cmsis@arm.com
- We are not perfect, please provide us feedback on what to improve

Pack Generation Examples

How packs are generated in practice

github.com/MDK-Packs/IoT_Socket - Native Pack project, PDSC file manually created

- IoT-Socket interface that is proposed in Open-CMSIS-CDI, during development, the repository can be directly accessed as pack (using cpackget)
- CMSIS utilities are used to validate the creation (XML schema check, PackChk), gen_pack.sh script is used to create the final pack
- Distribution of public packs uses a separate github repository (github.com/MDK-Packs/Pack)
- Pack Index file gives a vendor full control over the pack publishing process

github.com/lvgl/lvgl/tree/master/env_support/cmsis-pack - Graphic Library that uses gen_pack.sh

- PDSC file is created and maintained manually

https://github.com/MDK-Packs/tensorflow-pack - TFLu project + Arm ML components

- Pack generation (PDSC file) is automated with Python scripts and derived from the underlying open-source projects.

https://github.com/FreeRTOS/CMSIS-Packs - AWS FreeRTOS packs (created from CMake based projects)

- Pack generation (PDSC file) is automated PackGen and manifest.yml file

Watch **Embedded World – CMSIS 6 Replay** – contains details and explains tooling

More useful material

- Review online pack tutorials:
  - Pack with software components
  - Device family pack
  - Board support pack
- Pack creation utilities
  - Bash library for pack generation scripts
- PDSC format description
- GitHub action for pack generation

GitHub has a CI environment for testing currently in beta:
https://resources.github.com/arm-github-actions-beta/

Need MDK tool for creating software packs?

Send email to:

cmsis@arm.com

arm

Improve usability and reduce support

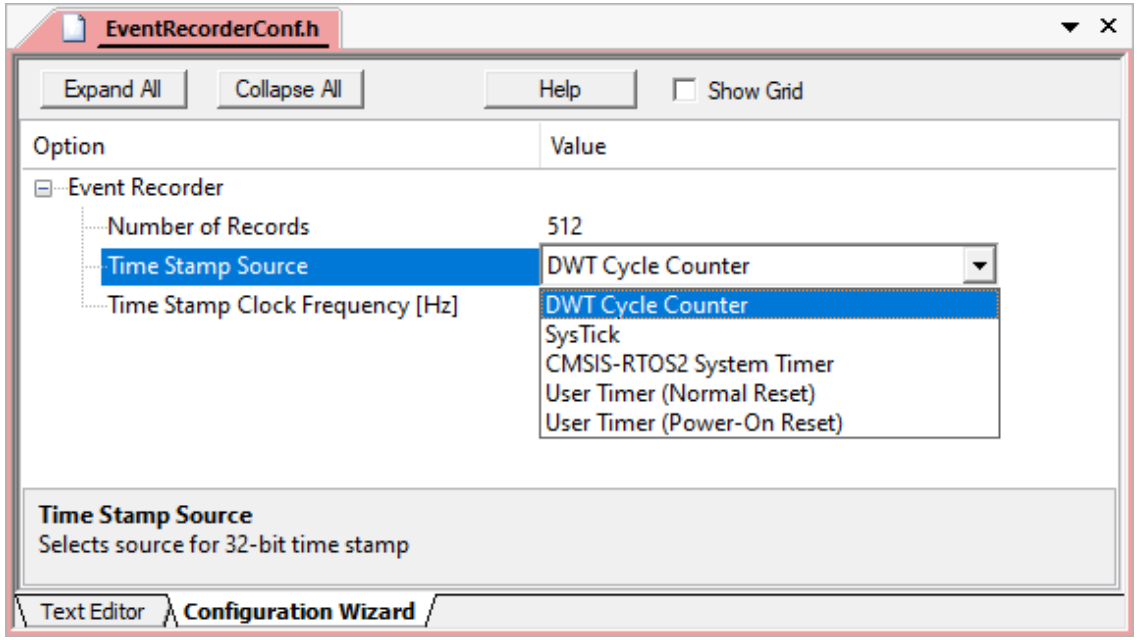
Configuration Wizard Annotations for Config Files

Create GUI-like elements in IDEs for configuration files

Source code

```
//-------- <<< Use Configuration Wizard in Context Menu >>> --------------------
// <h>Event Recorder
//   <o>Number of Records
//     <8=>8 <16=>16 <32=>32 <64=>64 <128=>128 <256=>256 <512=>512 <1024=>1024
//     <2048=>2048 <4096=>4096 <8192=>8192 <16384=>16384 <32768=>32768
//     <65536=>65536
//   <i>Configures size of Event Record Buffer (each record is 16 bytes)
//   <i>Must be 2^n (min=8, max=65536)
#define EVENT_RECORD_COUNT      512U
//   <o>Time Stamp Source
//     <0=> DWT Cycle Counter  <1=> SysTick  <2=> CMSIS-RTOS2 System Timer
//     <3=> User Timer (Normal Reset)  <4=> User Timer (Power-On Reset)
//   <i>Selects source for 32-bit time stamp
#define EVENT_TIMESTAMP_SOURCE  0
//   <o>Time Stamp Clock Frequency [Hz] <0-1000000000>
//   <i>Defines initial time stamp clock frequency (0 when not used)
#define EVENT_TIMESTAMP_FREQ    0U
// </h>
//------------- <<< end of configuration section >>> ---------------------------
```

GUI-like representation



→ Example RTOS configuration

CMSIS-View – Reducing your support burden

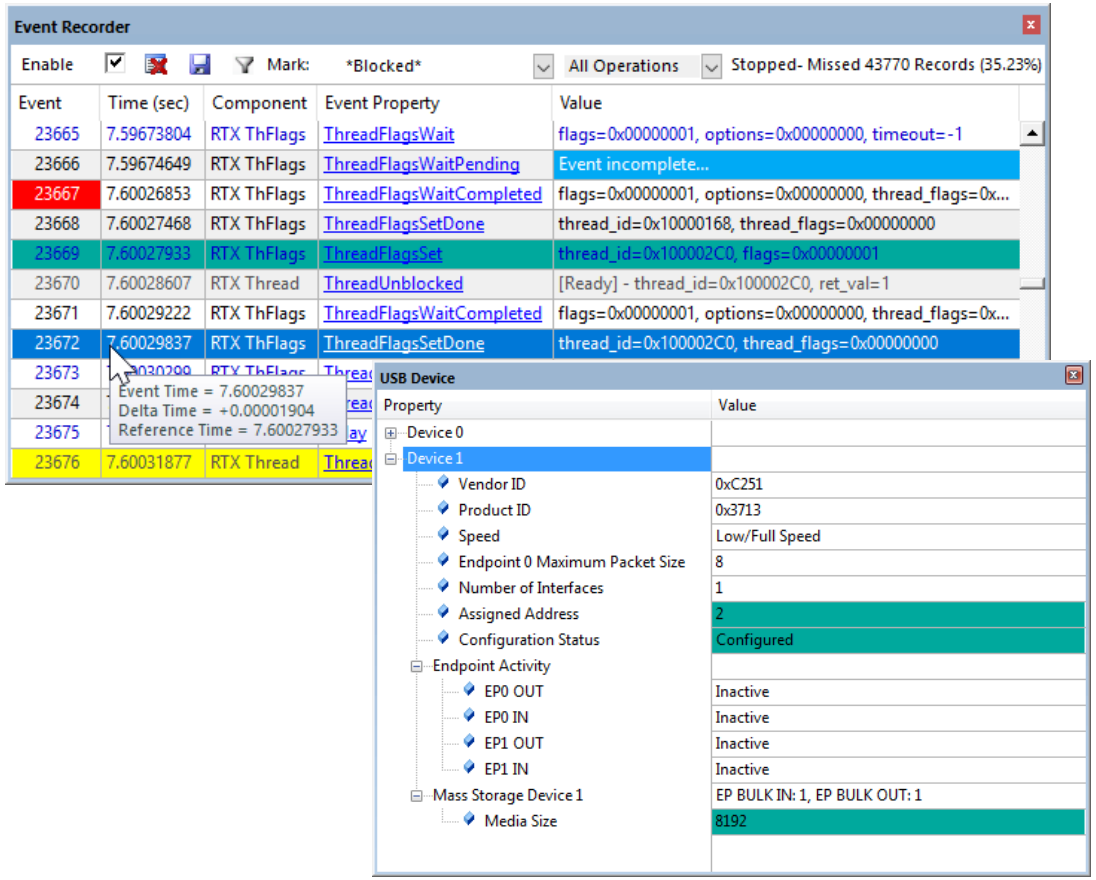
Shows internal operation of embedded applications and software components

Event Recorder/Event Statistics

- API for event annotation functions
- Provides visibility to the dynamic execution of an application at little (memory) cost.
- Collect statistical data about the code execution
- Fast time-deterministic execution with minimal code and timing overhead.
- Event annotations can stay in production code.

Component Viewer

- Reads memory locations of symbols representing variables, arrays, or linked lists.



→ Example for FreeRTOS

What are the benefits …

… when using Packs to deliver middleware?

- **Connect to users:** as software vendor you control distribution which lets you innovate faster.
  - For keil.com/pack updates on your pack repository are scanned once per day.
  - Distribution is independent of other “SDKs”, but base software from other packs can be used.
  - Using the documentation links lets you connect with your customers.

- **Support business goals:** with ways to distribute different variants (eval, lite, full) of a software.
  - Users can seamlessly transition from an eval version to a full version by just installing the pack.
    - The dominate attribute may be applied to a full version of a software.

- **One way to distribute:** for all relevant toolchains, scales to many devices when APIs are applied.
  - CMSIS and the CMSIS-Toolbox supports Arm Compiler, GCC, and IAR.

- **Reduces support efforts:** as it makes it easier for users to integrate in projects
  - Product Lifecycle Management is easier resulting in less support.
    - Updating a pack replaces the software components in the user project.
  - When using versions consequently, users are notified about outdated configuration files.

We are committed to CMSIS…

… and we will make it work for you – but we need your help

- Open-CMSIS Bi-Weekly Workshops: starting Tue 18. April (15:00 GMT)
  - **18. April:** How to create scalable software packs to maximize software re-use
  - **2. May:** Structure of Device Family Packs (DFP) and Board Support Packs (BSP)
  - **16. May:** CI test process for validation of reference applications
  - Review and evolve existing API interfaces – we need to structure taxonomies
  - Any other topics that relate to improving software re-use with packs
- PSA Certified Working Group meetings: 20. April, 18. May (16:00 GMT)
  - Approval of the PSA Certified Firmware Update 1.0 specification
  - Identifying future requirements for firmware update, and evolving the spec
- CMSIS-Stream technical details: Wednesday, 10. May (15:00 GMT)
  - Introduction to infra-structure, tools, and SDS-Framework
  - Discussion of MLOps integration and feedback on potential gaps

To get an invite to these virtual meetings send email to:

cmsis@arm.com

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