



arm



Generating CMSIS-Packs for Middleware

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

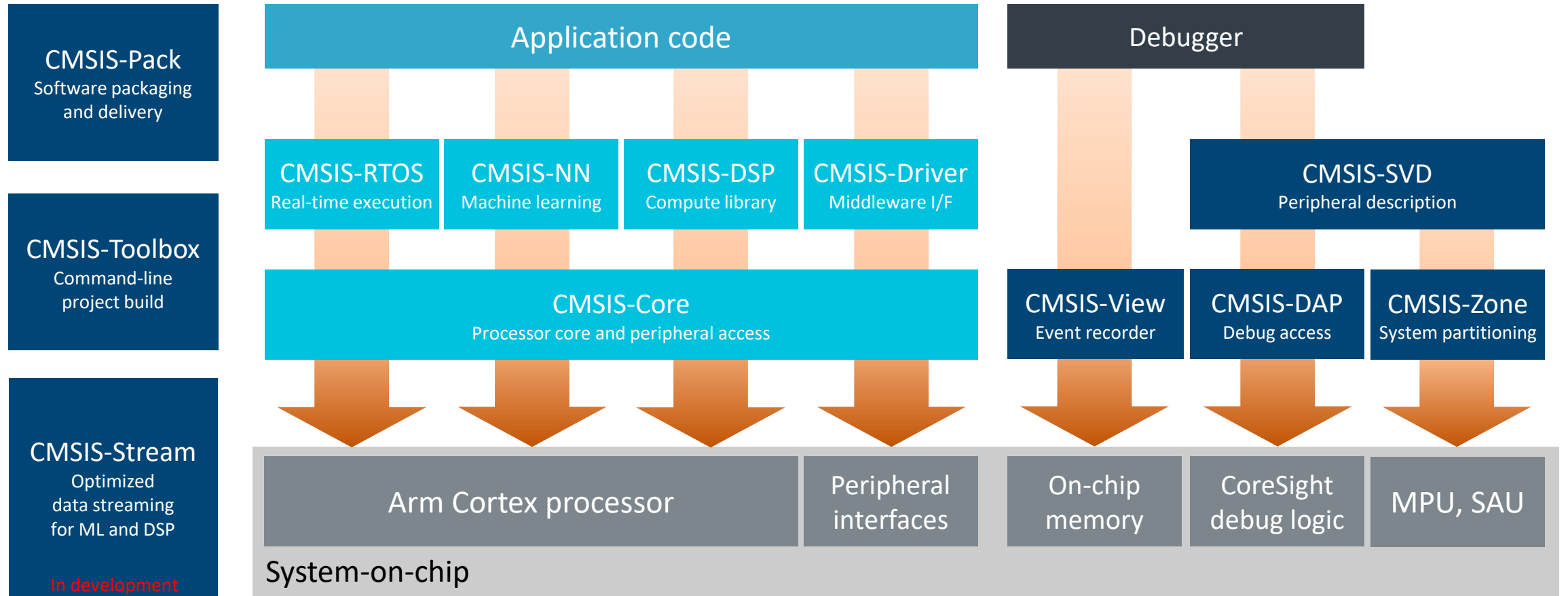
Linaro WG Meeting

Arm MCU Tools Team
18 April 2023



CMSIS Version 6

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



Software components for the Arm Cortex processor target
 Tools for optimizing software development flows



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.

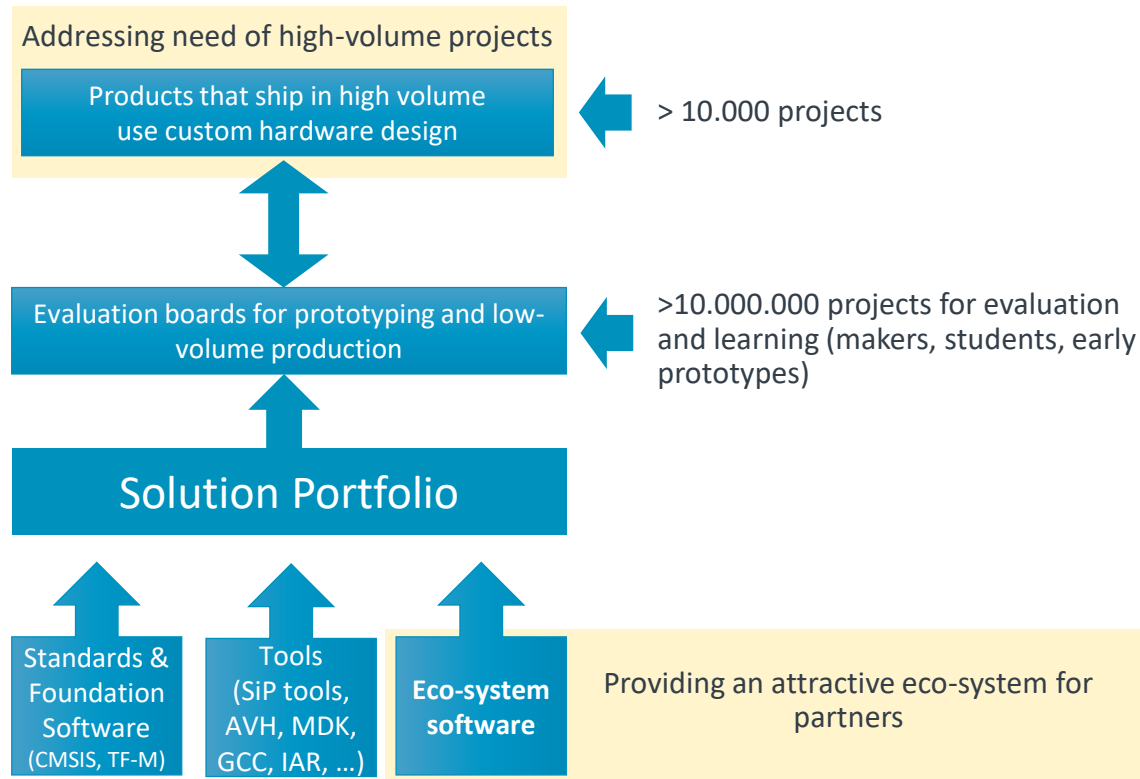
Retargeting

Select files based on hardware selection or toolchain requirements.

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.

Usage example with Middleware

c:\test\HTTP_Server\MDK\Boards\Keil\MCBSTM32F400\Middleware\Network\HTTP_Server\HTTP_Server.uvprojx - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

stack

Debug

Project: HTTP_Server

Abstract.txt

This is a HTTP_Server example running on Network Dual Stack.

Manage Run-Time Environment

Software Component	Sel.	Variants	Version	Description
CycloneTCP	<input checked="" type="checkbox"/>	CycloneTCP	2.2.0	Dual IPv4/IPv6 Stack
Data Exchange	<input checked="" type="checkbox"/>			Data exchange or data formatter
Data Processing	<input checked="" type="checkbox"/>			Software Components for Data Processing
Device	<input checked="" type="checkbox"/>			Startup System Setup
File System	<input checked="" type="checkbox"/>	MDK-Plus	6.15.3	File Access on various storage devices
FreeRTOS	<input checked="" type="checkbox"/>			
Graphics	<input checked="" type="checkbox"/>	MDK-Plus	6.24.0	User Interface on graphical LCD displays
Graphics Display	<input checked="" type="checkbox"/>			Display Interface including configuration for emWIN
IoT Utility	<input checked="" type="checkbox"/>			IoT specific software utility
LVGL	<input checked="" type="checkbox"/>	LVGL	8.3.5	LVGL (Light and Versatile Graphics Library) is a free and open-source graphics library provided by the Espressif System
Machine Learning	<input checked="" type="checkbox"/>			Software Components for Machine Learning
Native Driver	<input checked="" type="checkbox"/>			
Network	<input checked="" type="checkbox"/>	MDK-Pro	7.18.0	IPv4/IPv6 Networking using Ethernet or Serial protocols
CORE	<input checked="" type="checkbox"/>	IPv4/IPv6 Debug	7.18.0	IPv4/IPv6 Networking Core for Cortex-M (Debug)
Legacy API	<input type="checkbox"/>		7.18.0	Network Legacy API support
Interface	<input checked="" type="checkbox"/>			Connection Mechanism
ETH	<input checked="" type="checkbox"/>		7.18.0	Network Ethernet Interface
PPP	<input type="checkbox"/>	Custom Modem	7.18.0	Network PPP over Serial Interface
SLIP	<input type="checkbox"/>	Custom Modem	7.18.0	Network SLIP Interface
WiFi	<input type="checkbox"/>		7.18.0	Network WiFi Interface
Service	<input checked="" type="checkbox"/>			Network Services
DNS Client	<input type="checkbox"/>		7.18.0	DNS Client
FTP Client	<input type="checkbox"/>		7.18.0	FTP Client
FTP Server	<input type="checkbox"/>		7.18.0	FTP Server
SMTP Client	<input type="checkbox"/>	SMTP	7.18.0	Email Client (SMTP)
SNMP Agent	<input type="checkbox"/>		7.18.0	SNMP Agent
SNTP Client	<input type="checkbox"/>		7.18.0	SNTP Client
TFTP Client	<input type="checkbox"/>		7.18.0	TFTP Client
TFTP Server	<input type="checkbox"/>		7.18.0	TFTP Server
Telnet Server	<input type="checkbox"/>		7.18.0	Telnet Server
Web Server Compact	<input checked="" type="checkbox"/>	HTTP	7.18.0	Web Server (HTTP) with Read-only Web Resources (Compact)
Web Server	<input type="checkbox"/>	HTTP	7.18.0	Web Server (HTTP) with Web Resources on File System
Socket	<input checked="" type="checkbox"/>			Network Sockets
BSD	<input type="checkbox"/>		7.18.0	BSD Socket
TCP	<input checked="" type="checkbox"/>		7.18.0	TCP Socket
UDP	<input checked="" type="checkbox"/>		7.18.0	UDP Socket
PSA	<input checked="" type="checkbox"/>			Platform Security Architecture
RTOS	<input checked="" type="checkbox"/>	FreeRTOS	10.5.1	FreeRTOS Real Time Kernel
Security	<input checked="" type="checkbox"/>			Encryption for secure communication or storage
USB	<input checked="" type="checkbox"/>	MDK-Plus	6.16.1	USB Communication with various device classes

Resolve Select Packs Details OK Cancel Help

Network Component: Network C x Network Component: Network C x

File | C:/Users/reikei01/AppData/Local/Arm/Packs/Keil/MDK-Middleware/7.18.0/Doc/Network/html/index.html

arm KEIL Network Component Version 7.18.0

MDK Middleware for IPv4 and IPv6 Networking

General File System Graphics **Network** USB Board Support

Main Page Usage and Description Reference

Network Component

- Overview
- Revision History
- Creating a Network Application
- Troubleshooting a Network Application
- Secure Communication
- Cyber Security
- Network Examples
- Migration
- Resource Requirements
- Function Overview
- Reference

Network Component Overview

The Network Component v7 contains services, protocol sockets, and physical communication interfaces for creating IPv4 and IPv6 networking applications.

Service	Compact Web Server	Full Web Server Using File System	FTP Server	TFTP Server	Telnet Server	
	SNMP Agent	DNS Client	SNTP Client	FTP Client	TFTP Client	SMTP Client

Socket	BSD	TCP	UDP	CORE IPv4/IPv6 Dual-Stack
Interface	Ethernet	WiFi	PPP (Serial)	SLIP (Serial)

CMSIS-Driver	SSL/TLS		
Ethernet	WiFi	USART	MbedTLS

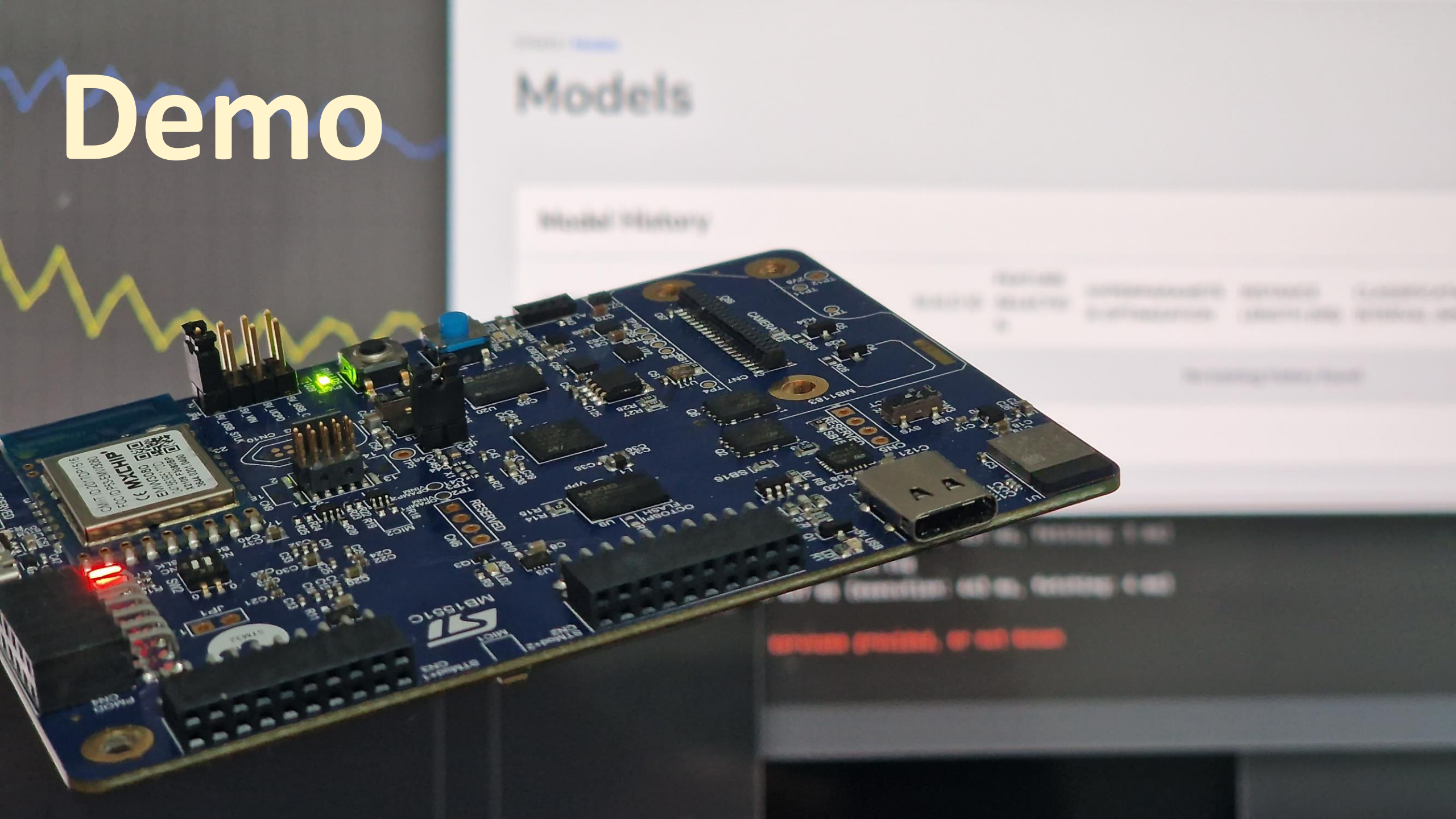
Network Overview

The various services provide program templates for common networking tasks.

Copyright © 2004-2022 Arm Limited (or its affiliates). All rights reserved.



Demo



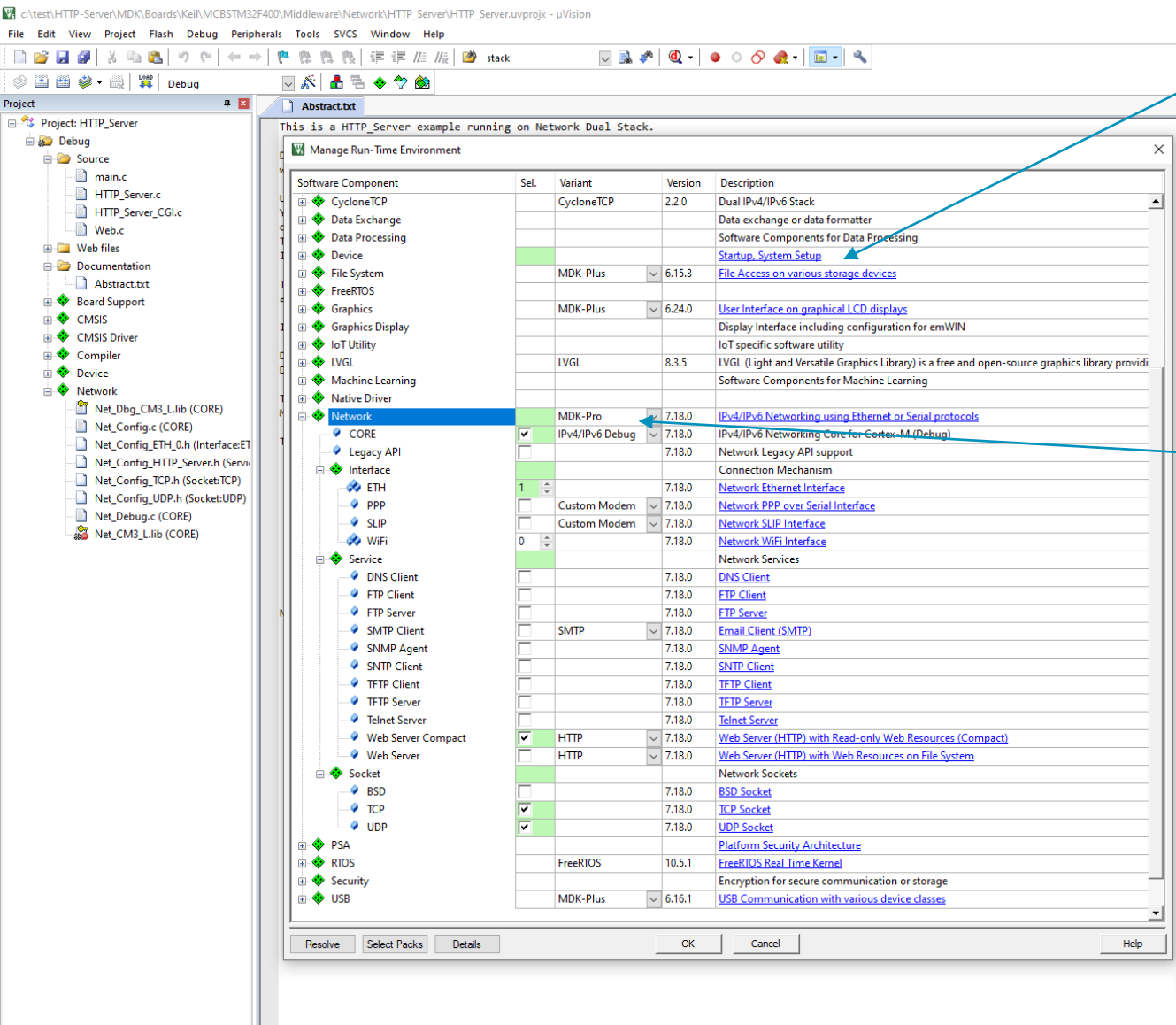
Models

Model History

Model Name	Created At	Updated At	Version
Model A	2023-10-26	2023-10-26	1.0
Model B	2023-10-26	2023-10-26	1.0
Model C	2023-10-26	2023-10-26	1.0
Model D	2023-10-26	2023-10-26	1.0

STATUS: SUCCESS

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.

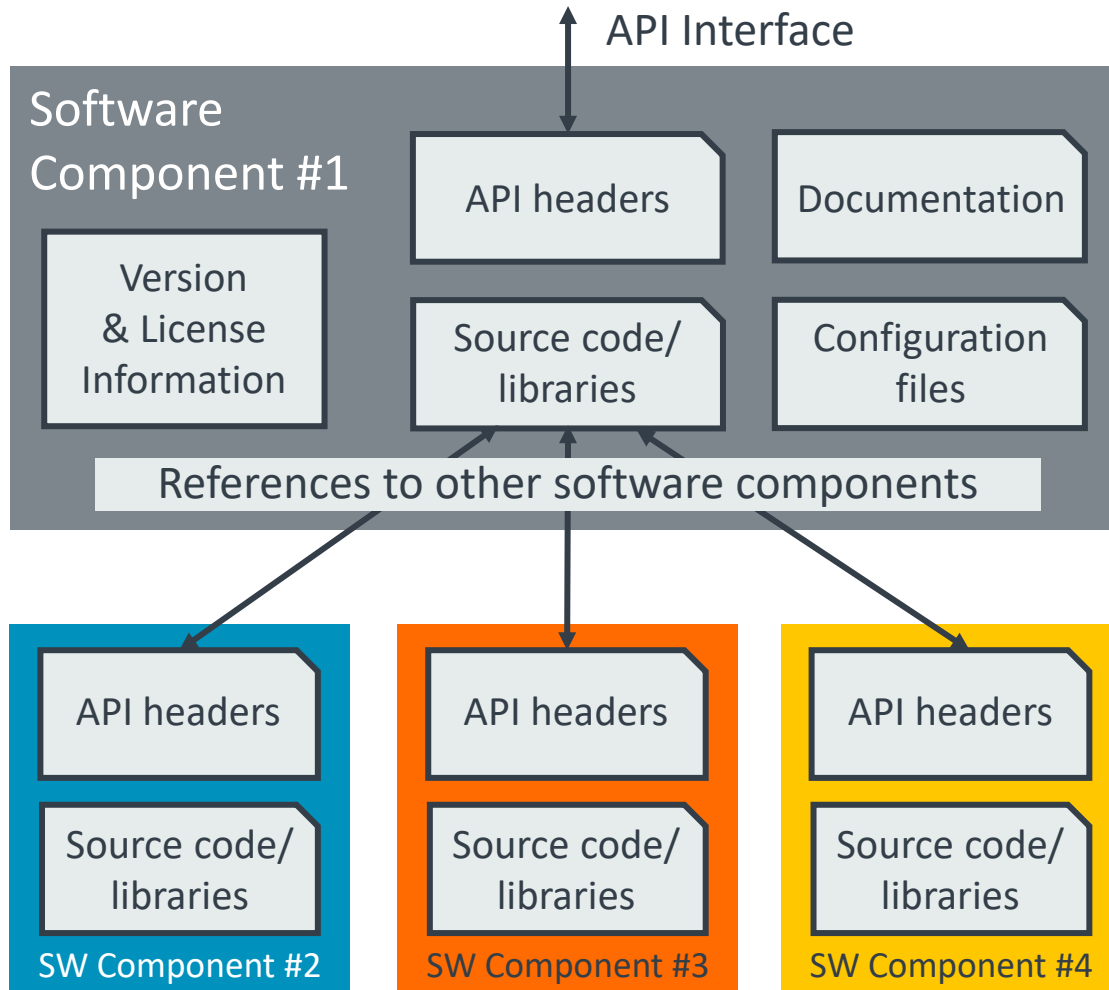
Packs support your business needs:

- 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 semantic 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

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

Middleware Pack

contains high-value software that works on many devices

Driver Pack

Device Series #1

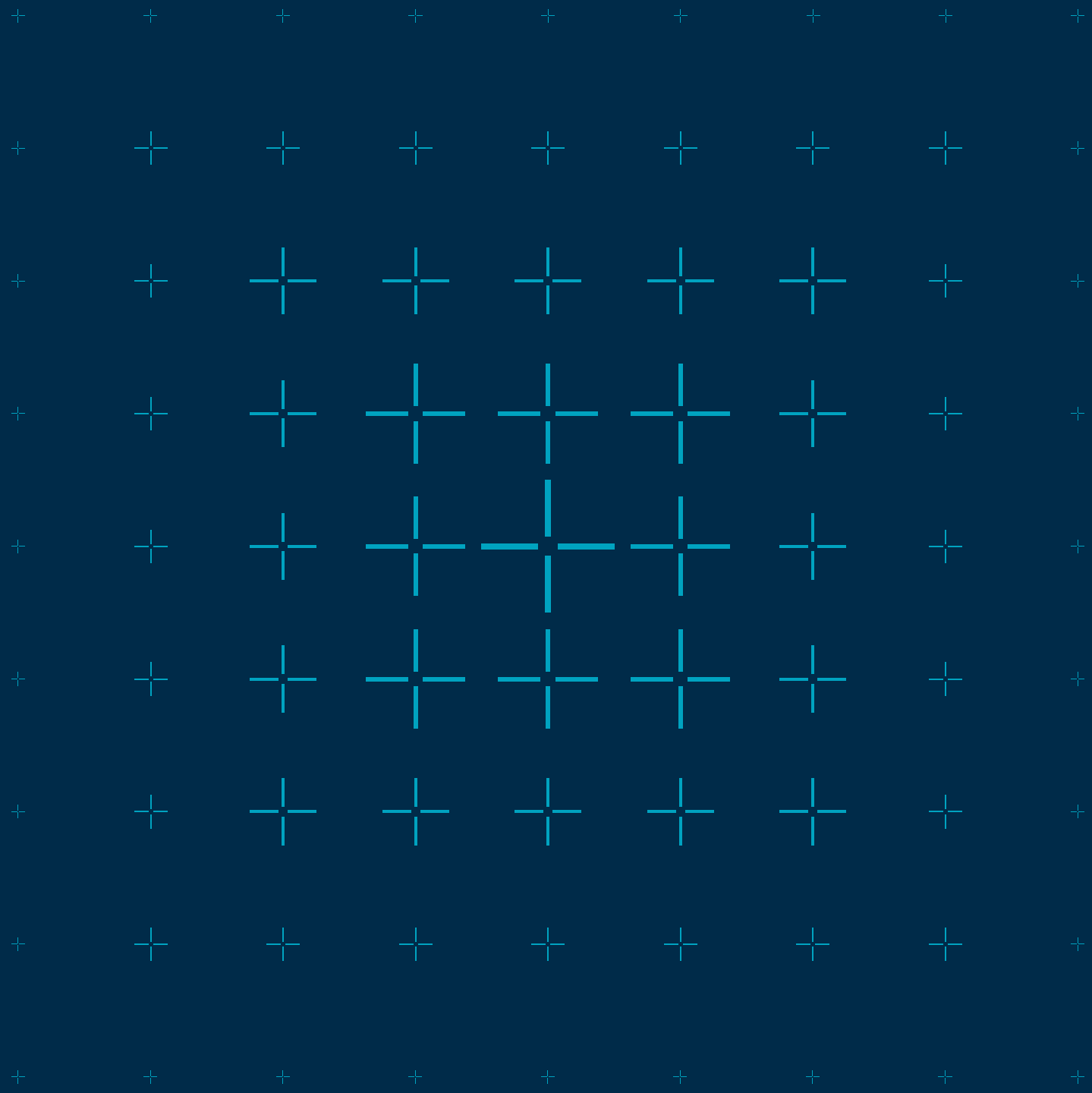
Driver Pack

Device Series #n

The driver interface is defined as API i.e. in the Middleware Pack

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

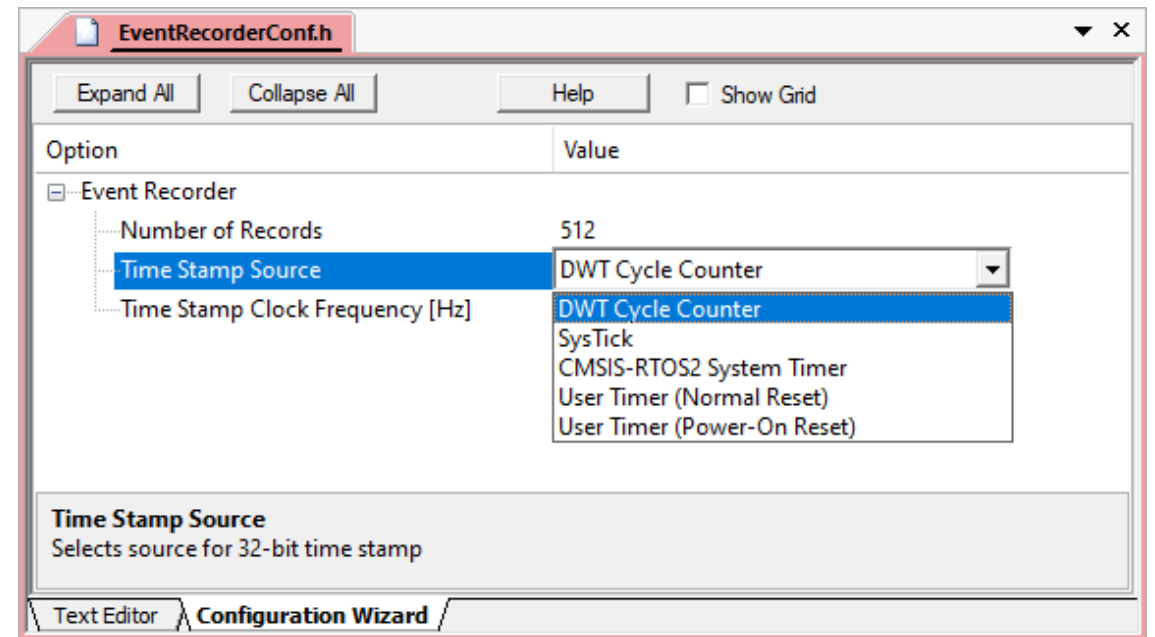
Configuration Wizard Annotations for Config Files

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] <0-100000000>  
// <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.

The screenshot shows two windows from the CMSIS-View tool. The top window is the 'Event Recorder' window, which displays a table of recorded events. The bottom window is the 'USB Device' window, which shows the properties of a connected USB device.

Event	Time (sec)	Component	Event Property	Value
23665	7.59673804	RTX ThFlags	ThreadFlagsWait	flags=0x00000001, options=0x00000000, timeout=-1
23666	7.59674649	RTX ThFlags	ThreadFlagsWaitPending	Event incomplete...
23667	7.60026853	RTX ThFlags	ThreadFlagsWaitCompleted	flags=0x00000001, options=0x00000000, thread_flags=0x...
23668	7.60027468	RTX ThFlags	ThreadFlagsSetDone	thread_id=0x10000168, thread_flags=0x00000000
23669	7.60027933	RTX ThFlags	ThreadFlagsSet	thread_id=0x100002C0, flags=0x00000001
23670	7.60028607	RTX Thread	ThreadUnblocked	[Ready] - thread_id=0x100002C0, ret_val=1
23671	7.60029222	RTX ThFlags	ThreadFlagsWaitCompleted	flags=0x00000001, options=0x00000000, thread_flags=0x...
23672	7.60029837	RTX ThFlags	ThreadFlagsSetDone	thread_id=0x100002C0, thread_flags=0x00000000
23673	7.60030200	RTX ThFlags	ThreadFlagsWaitPending	Event Time = 7.60029837 Delta Time = +0.00001904 Reference Time = 7.60027933
23674	7.60030200	RTX ThFlags	ThreadFlagsWaitCompleted	
23675	7.60030200	RTX ThFlags	ThreadFlagsSetDone	
23676	7.60031877	RTX Thread	ThreadUnblocked	

Property	Value
Device 0	
Device 1	
Vendor ID	0xC251
Product ID	0x3713
Speed	Low/Full Speed
Endpoint 0 Maximum Packet Size	8
Number of Interfaces	1
Assigned Address	2
Configuration Status	Configured
Endpoint Activity	
EP0 OUT	Inactive
EP0 IN	Inactive
EP1 OUT	Inactive
EP1 IN	Inactive
Mass Storage Device 1	
Media Size	EP BULK IN: 1, EP BULK OUT: 1 8192

→ [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

arm

Thank You

Danke

Gracias

Grazie

谢谢

ありがとう

Asante

Merci

감사합니다

धन्यवाद

Kiitos

شكرًا

ধন্যবাদ

תודה



The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks