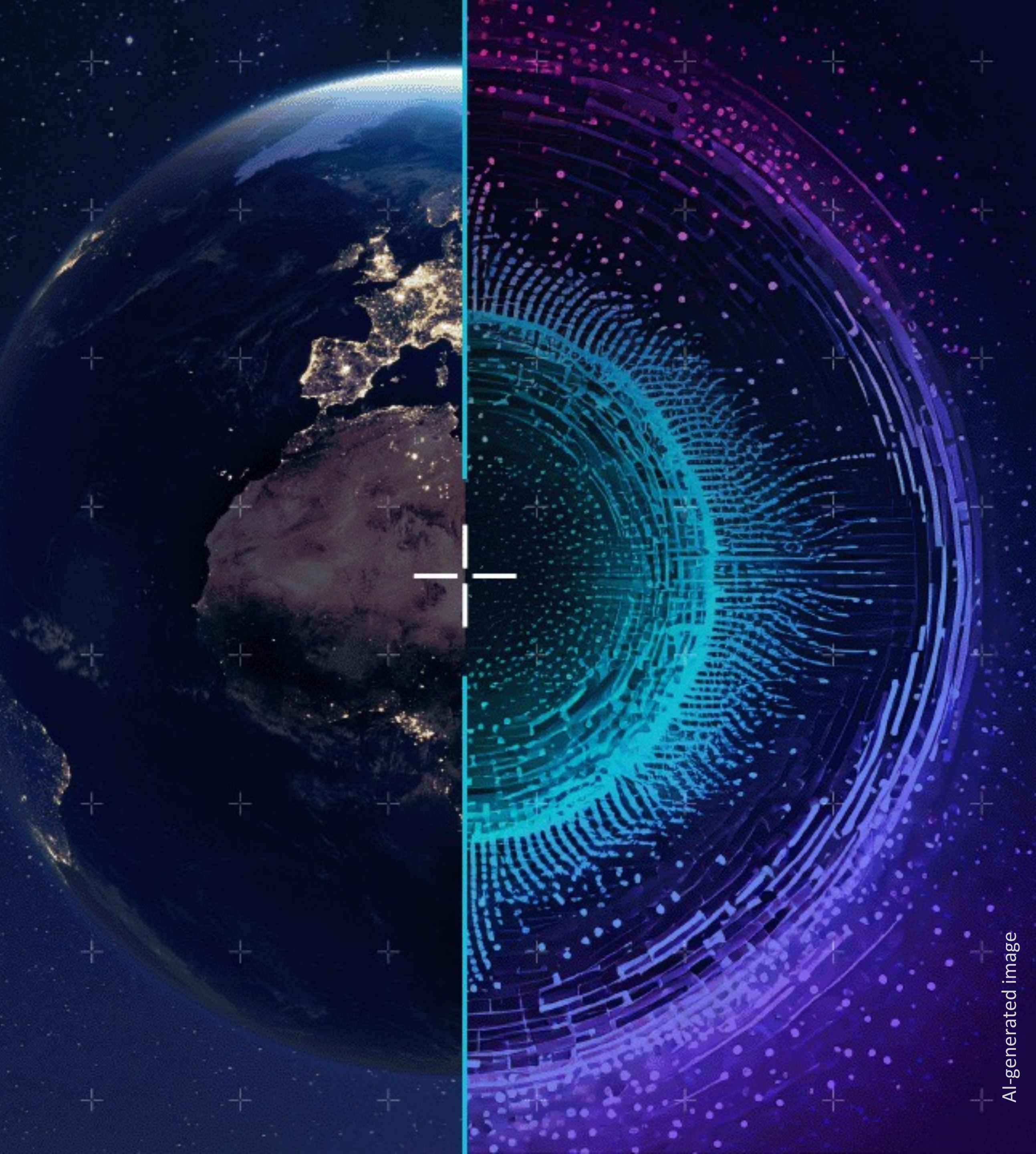
arm

CMSIS + ML

Welcome to the 16th Partner Meeting

embedded world 2024

Arm MCU Tools Team
09 April 2023

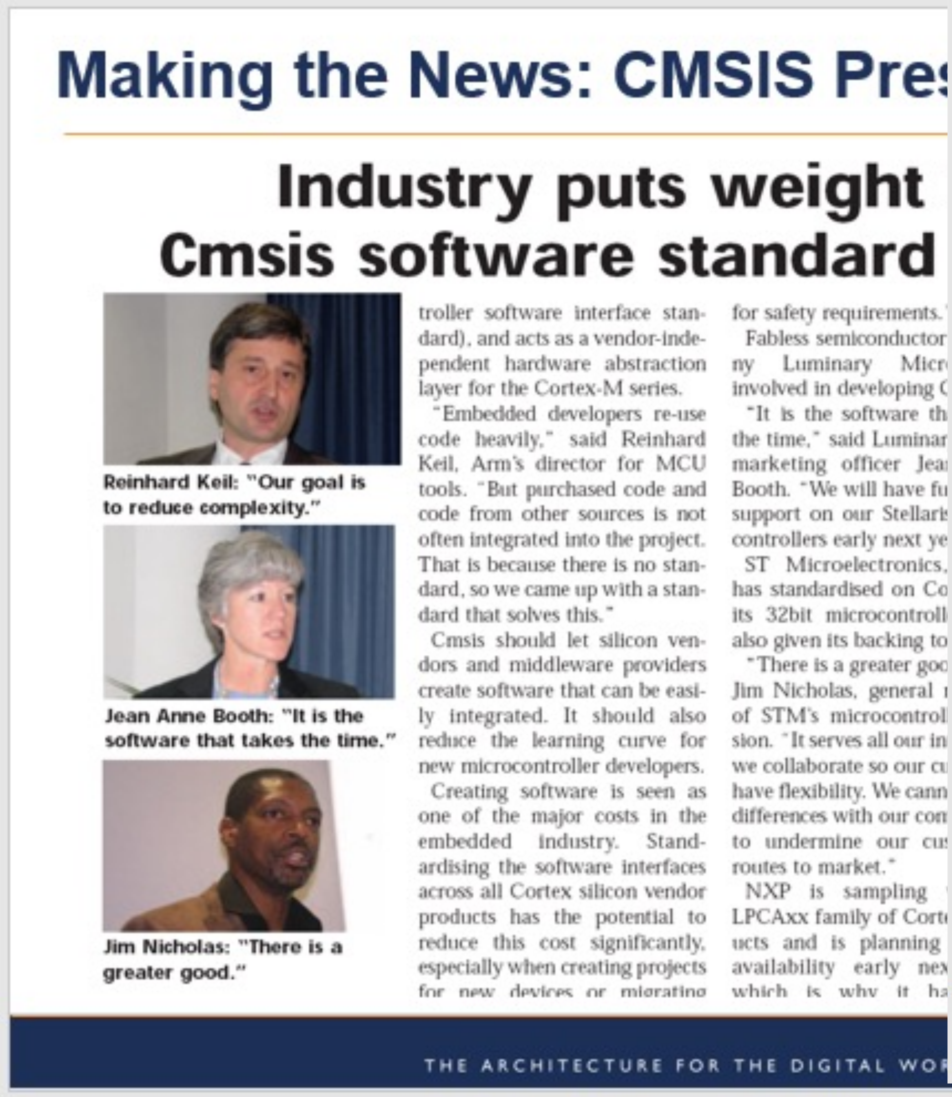


AI-generated image

Agenda

- History of CMSIS and future challenges
- What is CMSIS today?
- Foundation Tools for Developing ML on Edge Devices
- Tool Components for VS Code
  - Open-CMSIS-Pack/CMSIS-Toolbox
  - Debug for Cortex-A/M including Multicore
  - LLVM Embedded Toolchain
  - Arm Tools Artifactory
- Demo of Tool components
- Create Reusable Software Stacks
  - How we promote software from eco-system partners
- More CMSIS Innovations
  - C++ Matrix and Vector Compute Algorithms
  - CMSIS-SVD Improvements
  - CMSIS-Pack Download Authorization
  - New Standardized APIs for middleware
- Questions and Feedback

CMSIS Partner Meetings – how it began!



Making the News: CMSIS Pres

Industry puts weight
Cmsis software standard

Reinhard Keil: "Our goal is to reduce complexity."

Jean Anne Booth: "It is the software that takes the time."

Jim Nicholas: "There is a greater good."

troller software interface standard), and acts as a vendor-independent hardware abstraction layer for the Cortex-M series.

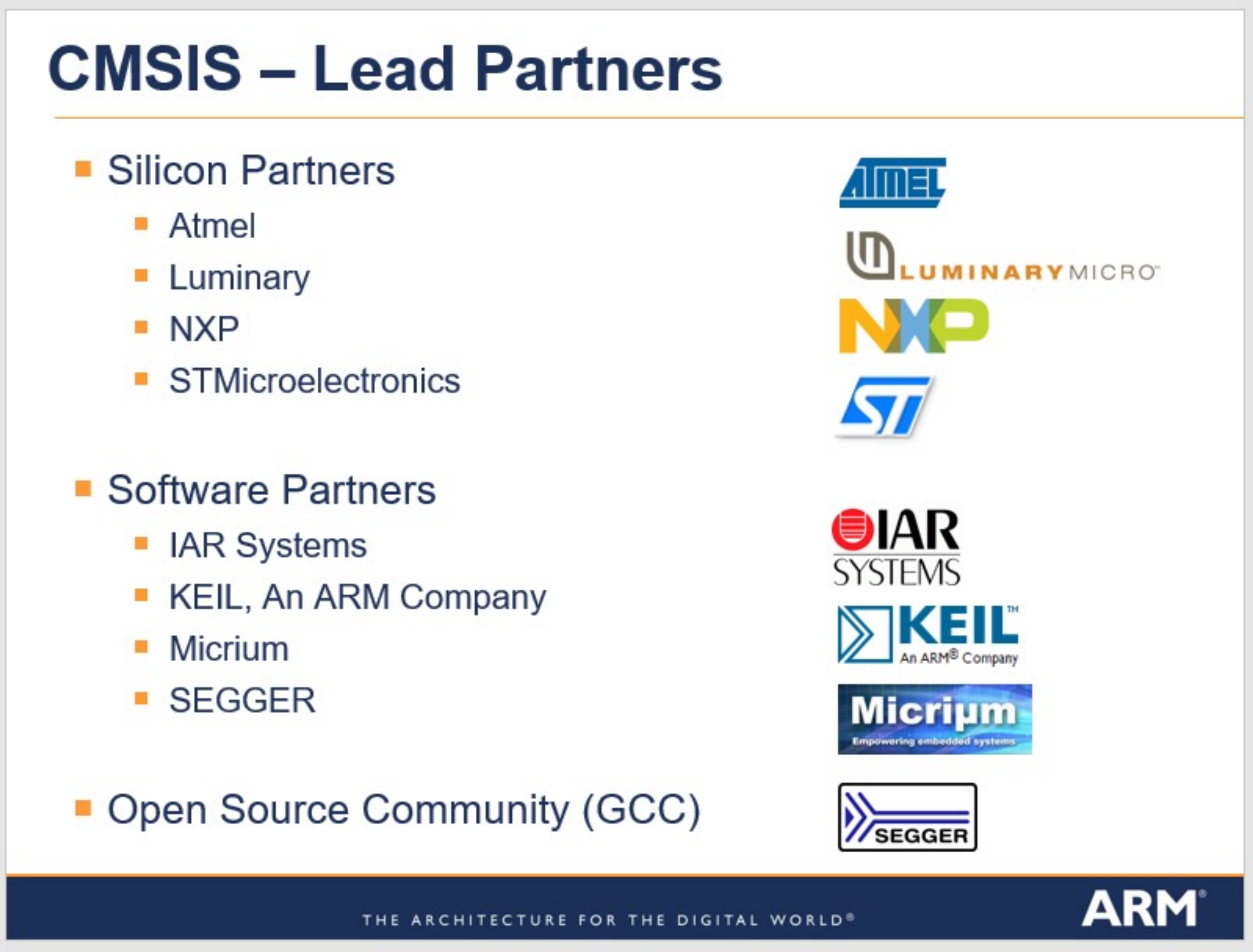
"Embedded developers re-use code heavily," said Reinhard Keil, Arm's director for MCU tools. "But purchased code and code from other sources is not often integrated into the project. That is because there is no standard, so we came up with a standard that solves this."

Cmsis should let silicon vendors and middleware providers create software that can be easily integrated. It should also reduce the learning curve for new microcontroller developers.

Creating software is seen as one of the major costs in the embedded industry. Standardising the software interfaces across all Cortex silicon vendor products has the potential to reduce this cost significantly, especially when creating projects for new devices or migrating

for safety requirements.'
Fabless semiconductor
ny Luminary Micr
involved in developing C
"It is the software th
the time," said Luminar
marketing officer Jea
Booth. "We will have fu
support on our Stellaris
controllers early next ye
ST Microelectronics,
has standardised on Co
its 32bit microcontroll
also given its backing to
"There is a greater goo
Jim Nicholas, general
of STM's microcontrol
sion. "It serves all our in
we collaborate so our cu
have flexibility. We cann
differences with our com
to undermine our cus
routes to market."
NXP is sampling
LPCAxx family of Corte
ucts and is planning
availability early nex
which is why it ha

THE ARCHITECTURE FOR THE DIGITAL WOR



CMSIS – Lead Partners

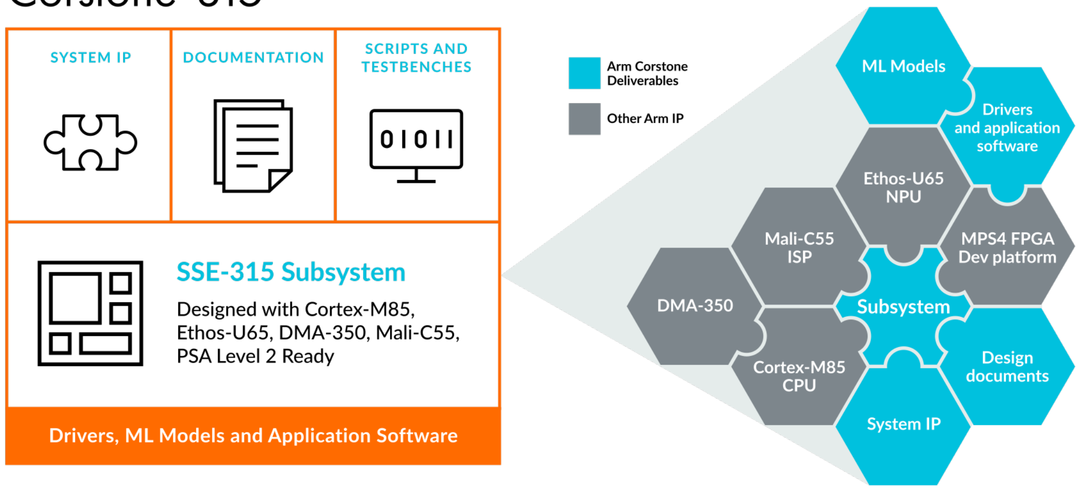
- Silicon Partners
  - Atmel
  - Luminary
  - NXP
  - STMicroelectronics
- Software Partners
  - IAR Systems
  - KEIL, An ARM Company
  - Micrium
  - SEGGER
- Open Source Community (GCC)

ATMEL
LUMINARY MICRO
NXP
ST
IAR SYSTEMS
KEIL An ARM® Company
Micrium Empowering embedded systems
SEGGER

THE ARCHITECTURE FOR THE DIGITAL WORLD®

ARM

Corstone-315



Open-CMSIS-Pack

Simplifying IoT Workflows and Lifecycle Management

arm Linaro NXP ST life.augmented

Foundation tool components

What is CMSIS today?

Christopher Seidl

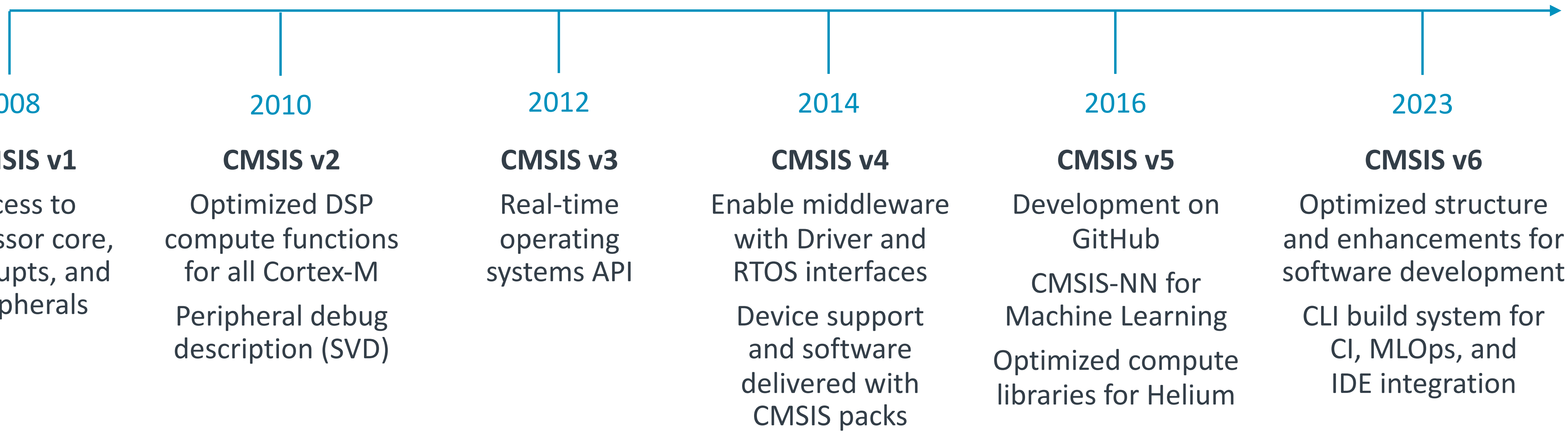
CMSIS - Fifteen Years of Software Evolution

10,000 supported devices

1000+ packs from 60 vendors

6 million GitHub projects

Multiple toolchains



2008

CMSIS v1

Access to processor core, interrupts, and peripherals

2010

CMSIS v2

Optimized DSP compute functions for all Cortex-M

Peripheral debug description (SVD)

2012

CMSIS v3

Real-time operating systems API

2014

CMSIS v4

Enable middleware with Driver and RTOS interfaces

Device support and software delivered with CMSIS packs

2016

CMSIS v5

Development on GitHub

CMSIS-NN for Machine Learning

Optimized compute libraries for Helium

2023

CMSIS v6

Optimized structure and enhancements for software development

CLI build system for CI, MLOps, and IDE integration

Two Decades of Microcontroller Innovation

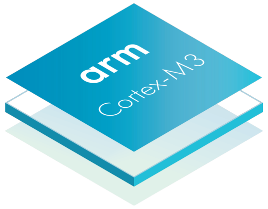
32-bit Compute

TrustZone Security

Helium ML/DSP

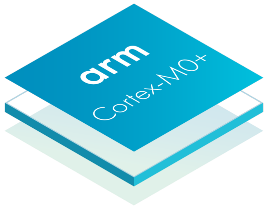
2004

Birth of the Low Cost, Low Power 32-bit Microcontroller



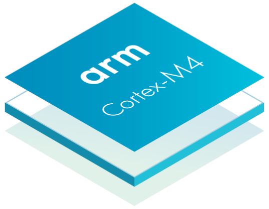
2009

Focus on ultra-low cost / power / area



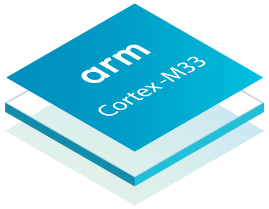
2010

Delivering DSP Extensions



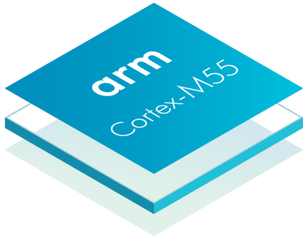
2016

Introducing TrustZone advanced security for Cortex-M



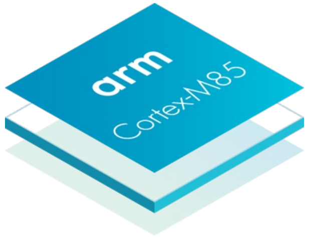
2020

Delivering Helium architecture technology for ML & DSP



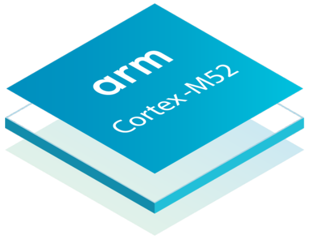
2022

Addressing higher performance ML & DSP applications



2023

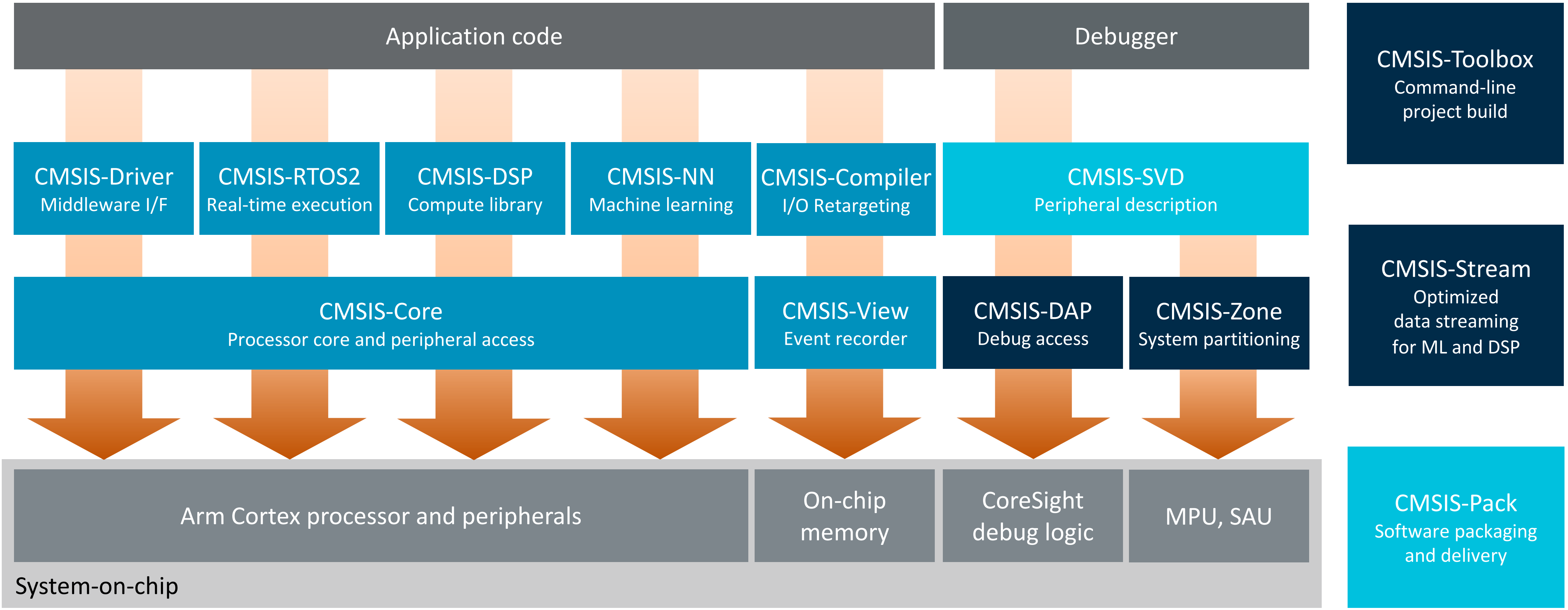
Today: Delivering high-efficiency ML & DSP



CMSIS Version 6

Consistent software framework for billions of devices

github.com/ARM-software/CMSIS_6



Application code

Debugger

CMSIS-Toolbox
Command-line project build

CMSIS-Driver
Middleware I/F

CMSIS-RTOS2
Real-time execution

CMSIS-DSP
Compute library

CMSIS-NN
Machine learning

CMSIS-Compiler
I/O Retargeting

CMSIS-SVD
Peripheral description

CMSIS-Stream
Optimized data streaming for ML and DSP

CMSIS-Core
Processor core and peripheral access

CMSIS-View
Event recorder

CMSIS-DAP
Debug access

CMSIS-Zone
System partitioning

Arm Cortex processor and peripherals

On-chip memory

CoreSight debug logic

MPU, SAU

System-on-chip

CMSIS-Pack
Software packaging and delivery

🟦 Software components for the Arm Cortex processor target

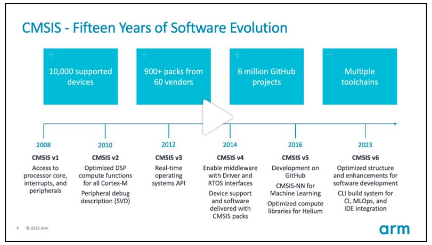
⬛ Tools for optimizing software development flows

🟦 Specifications

More information

CMSIS is not only for Cortex-M

[CMSIS v6 Overview]



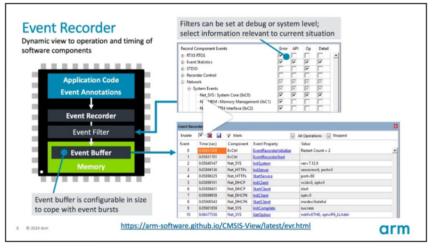
CMSIS - Fifteen Years of Software Evolution

10,000 supported devices | 900+ packs from 60 vendors | 6 million GitHub projects | Multiple toolchains

2008	2010	2012	2014	2016	2023
CMSIS v1	**CMSIS v2**	**CMSIS v3**	**CMSIS v4**	**CMSIS v5**	**CMSIS v6**
Access to processor core, interrupts, and peripherals	Optimized DSP compute functions for all Cortex-M Peripheral debug description (SVD)	Real-time operating systems API	Enable middleware with Driver and RTOS interfaces Device support and software delivered with CMSIS packs	Development on GitHub CMSIS-NN for Machine Learning Optimized compute libraries for Helium	Optimized structure and enhancements for software development CLI build system for CI, MLOps, and IDE integration

4 © 2023 Arm

[CMSIS-View/Compiler]



Event Recorder

Dynamic view to operation and timing of software components

Filters can be set at debug or system level; select information relevant to current situation

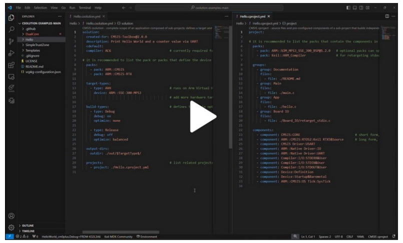
Event buffer is configurable in size to cope with event bursts

Event	Time (sec)	Component	Event Property	Value
0	0.05831509	EvCtrl	EventRecorderInitialize	Restart Count = 2
1	0.05831701	EvCtrl	EventRecorderStart	
2	0.05840147	Net_SYS	InitSystem	ver=7.12.0
3	0.05894136	Net_HTTPs	InitServer	sessions=6, port=0
4	0.05896325	Net_HTTPs	StartService	port=80
5	0.05899101	Net_DHCP	InitClient	vcid=0, opt=0
6	0.05899451	Net_DHCP	StartClient	start
7	0.05899959	Net_DHCP6	InitClient	opt=0
8	0.05900543	Net_DHCP6	StartClient	mode=Stateful
9	0.05901859	Net_SYS	InitComplete	success
10	0.06477536	Net_SYS	GetOption	netif=ETH0, opt=IP6_LLAddr

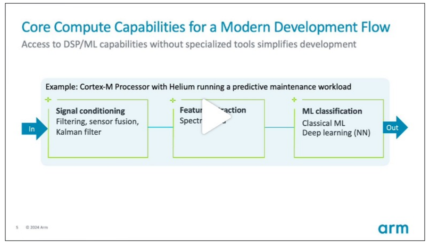
https://arm-software.github.io/CMSIS-View/latest/evr.html

6 © 2024 Arm

[CMSIS-Toolbox]



[CMSIS-Stream/SDS]



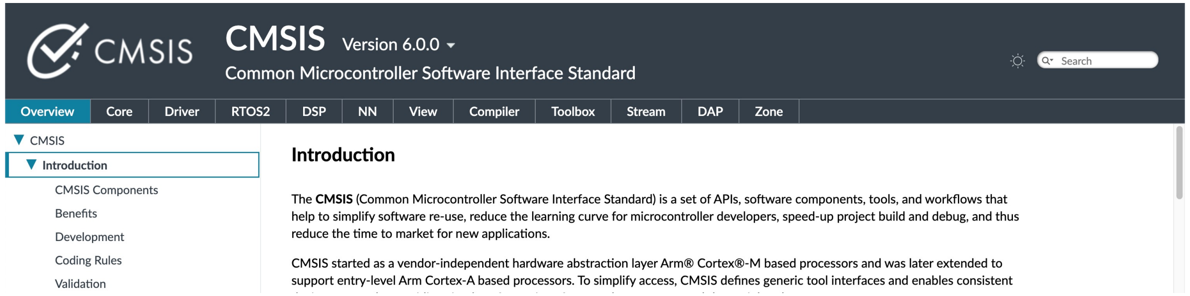
Core Compute Capabilities for a Modern Development Flow

Access to DSP/ML capabilities without specialized tools simplifies development

Example: Cortex-M Processor with Helium running a predictive maintenance workload

5 © 2024 Arm

[CMSIS v6 Documentation]



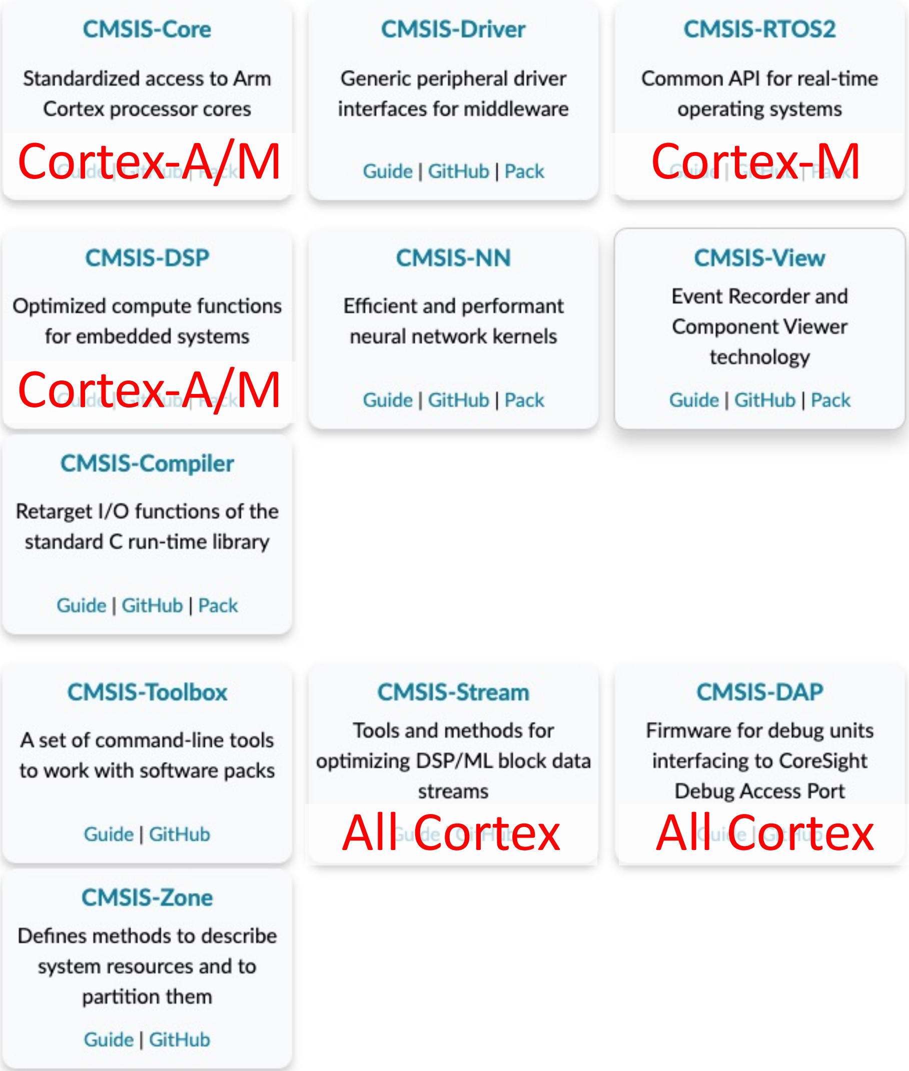
CMSIS **CMSIS** Version 6.0.0

Common Microcontroller Software Interface Standard

Introduction

The **CMSIS** (Common Microcontroller Software Interface Standard) is a set of APIs, software components, tools, and workflows that help to simplify software re-use, reduce the learning curve for microcontroller developers, speed-up project build and debug, and thus reduce the time to market for new applications.

CMSIS started as a vendor-independent hardware abstraction layer Arm® Cortex®-M based processors and was later extended to support entry-level Arm Cortex-A based processors. To simplify access, CMSIS defines generic tool interfaces and enables consistent



CMSIS-Core
Standardized access to Arm Cortex processor cores
Cortex-A/M

CMSIS-Driver
Generic peripheral driver interfaces for middleware
Guide | GitHub | Pack

CMSIS-RTOS2
Common API for real-time operating systems
Cortex-M

CMSIS-DSP
Optimized compute functions for embedded systems
Cortex-A/M

CMSIS-NN
Efficient and performant neural network kernels
Guide | GitHub | Pack

CMSIS-View
Event Recorder and Component Viewer technology
Guide | GitHub | Pack

CMSIS-Compiler
Retarget I/O functions of the standard C run-time library
Guide | GitHub | Pack

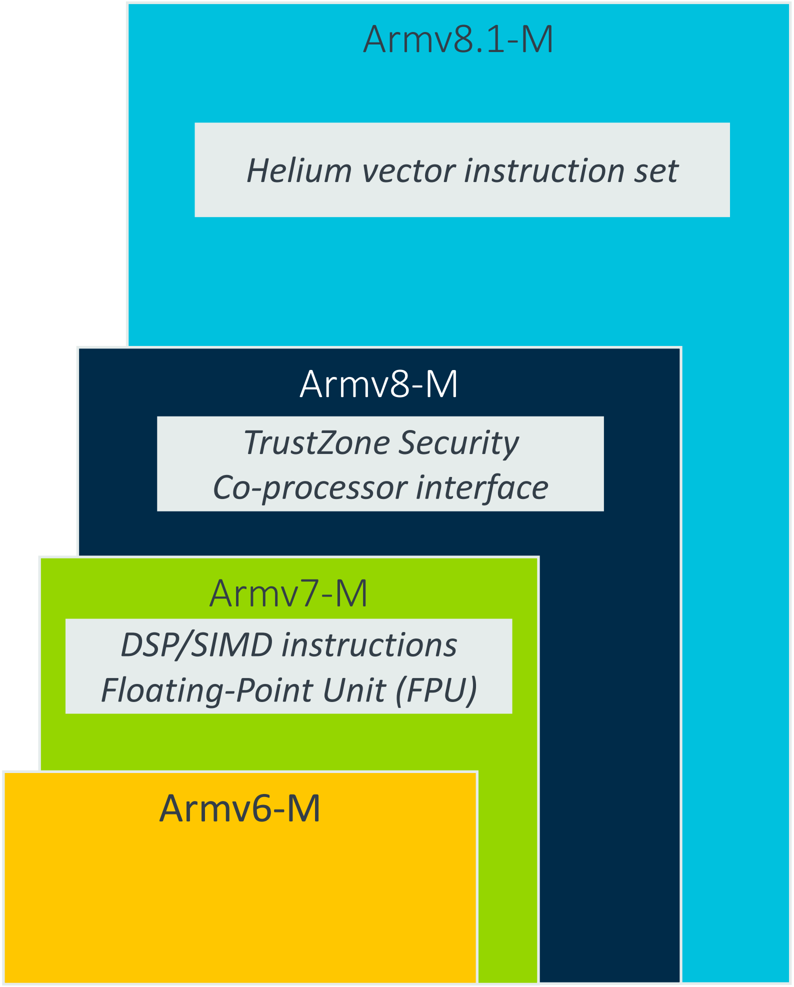
CMSIS-Toolbox
A set of command-line tools to work with software packs
Guide | GitHub

CMSIS-Stream
Tools and methods for optimizing DSP/ML block data streams
All Cortex

CMSIS-DAP
Firmware for debug units interfacing to CoreSight Debug Access Port
All Cortex

CMSIS-Zone
Defines methods to describe system resources and to partition them
Guide | GitHub

Cortex-M Processor Portfolio – Instruction Set Evolution



Cortex-M85
Cortex-M55
Cortex-M52

Armv8.1-M

Helium vector instruction set

Cortex-M33
Cortex-M23*
Cortex-M35P

Armv8-M

TrustZone Security
Co-processor interface

Cortex-M7
Cortex-M4
Cortex-M3**

Armv7-M

DSP/SIMD instructions
Floating-Point Unit (FPU)

Cortex-M0+
Cortex-M0

Armv6-M

>150 new scalar and vector instruction
Low overhead loops
Predication
Arithmetic support for 8-bit fixed and 16-bit float
Gather load, scatter store
Complex math

→ **Ease of development**
→ **Energy-efficient compute**
→ **Increased throughput**
→ **Smarter devices**

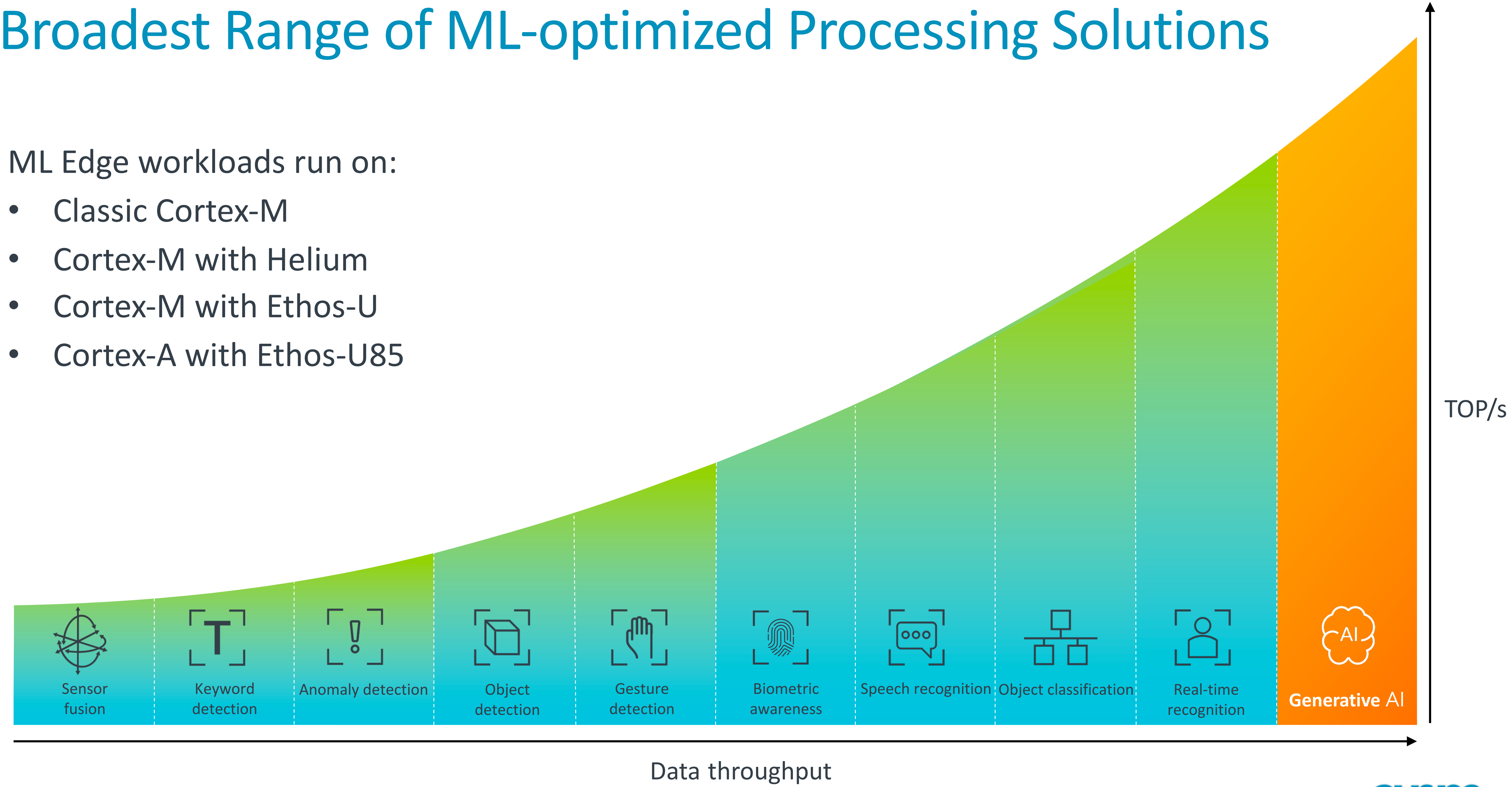
* Cortex-M23 does not include co-processor interface, DSP/SIMD and FPU

** Cortex-M3 does not include DSP/SIMD and FPU

Broadest Range of ML-optimized Processing Solutions

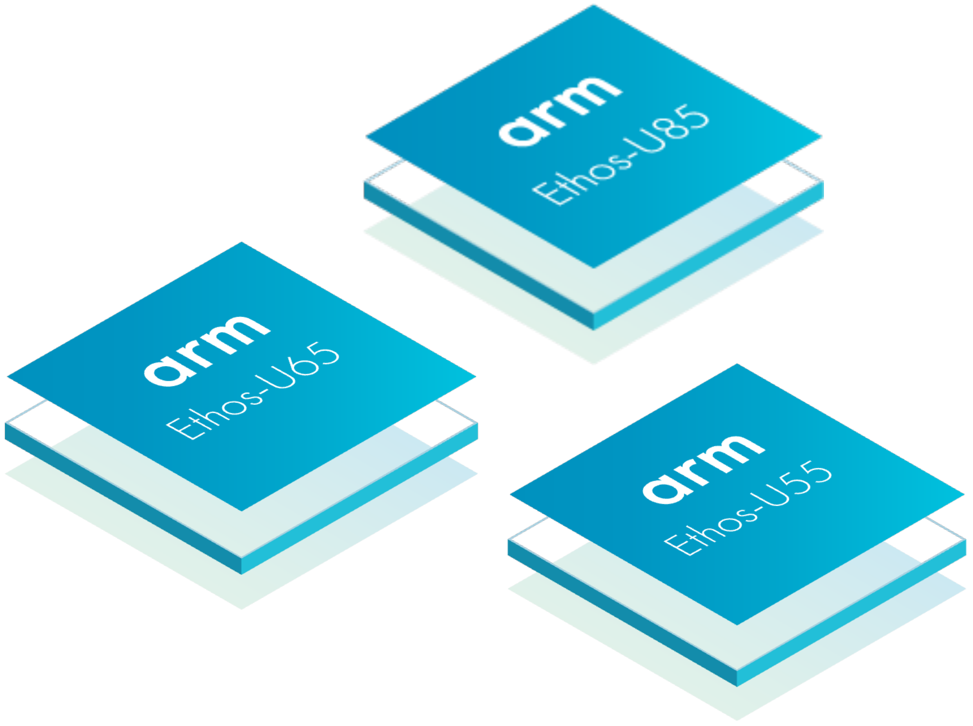
ML Edge workloads run on:

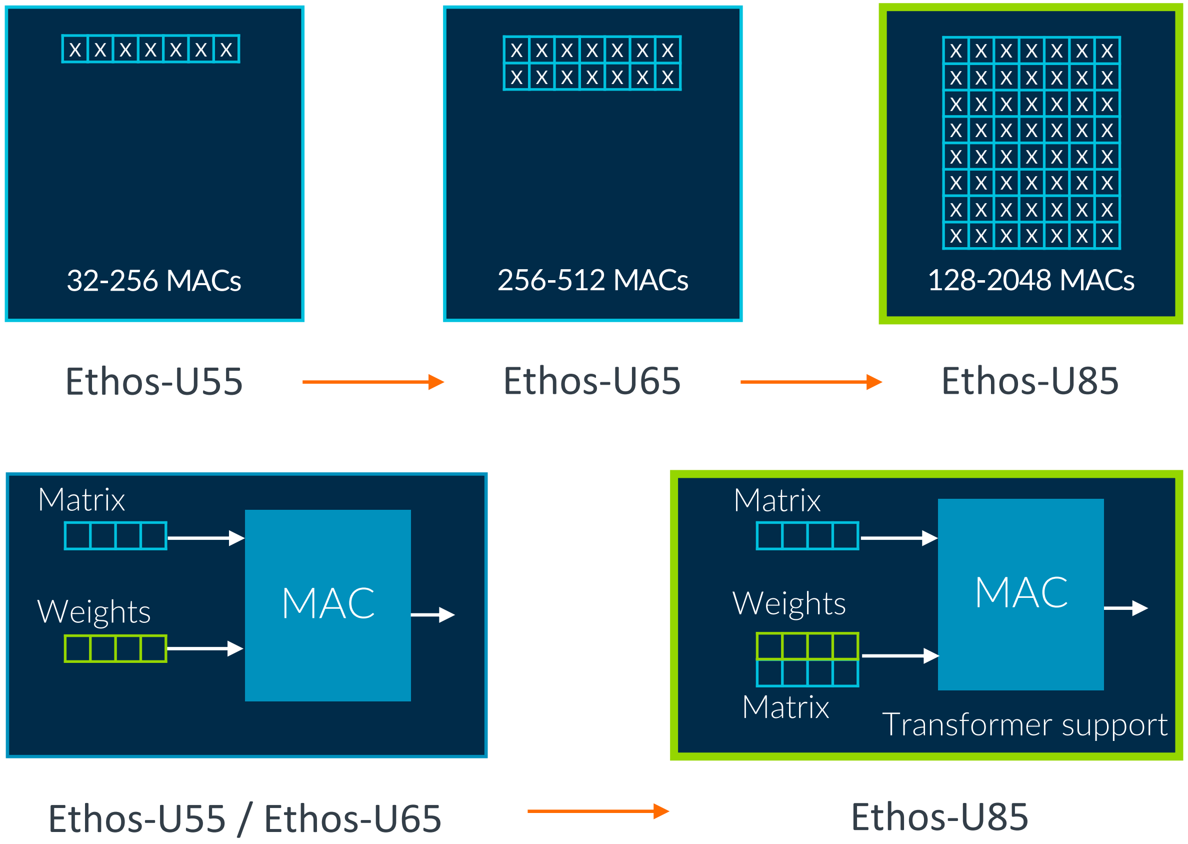
- Classic Cortex-M
- Cortex-M with Helium
- Cortex-M with Ethos-U
- Cortex-A with Ethos-U85



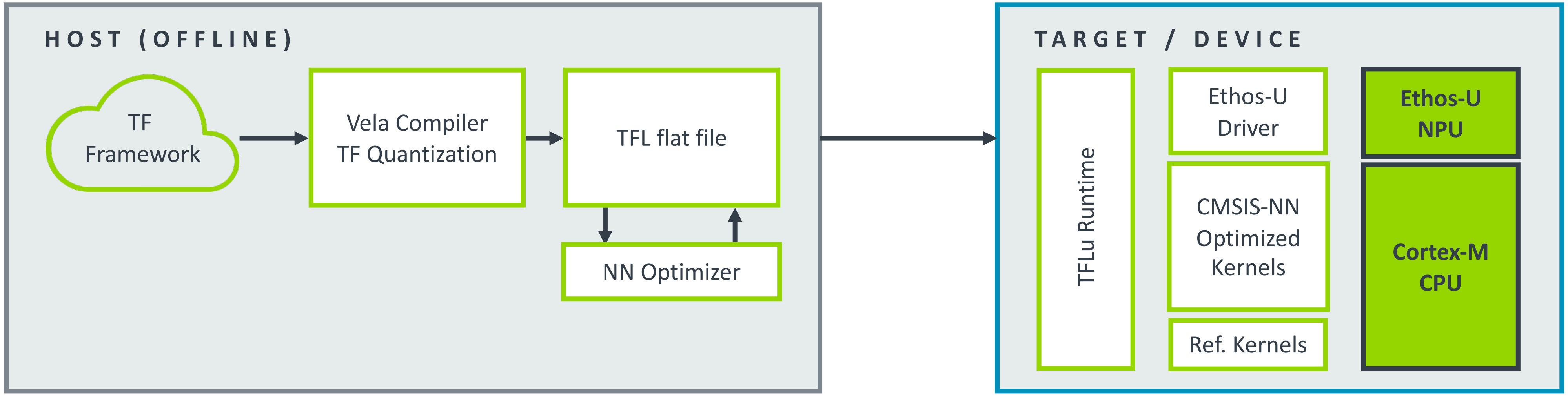
Ethos-U: Unlocking the Full Potential of Neural Networks

Accelerating implementation of higher performance AI enabled systems





Ethos-U software flow on Cortex-M systems



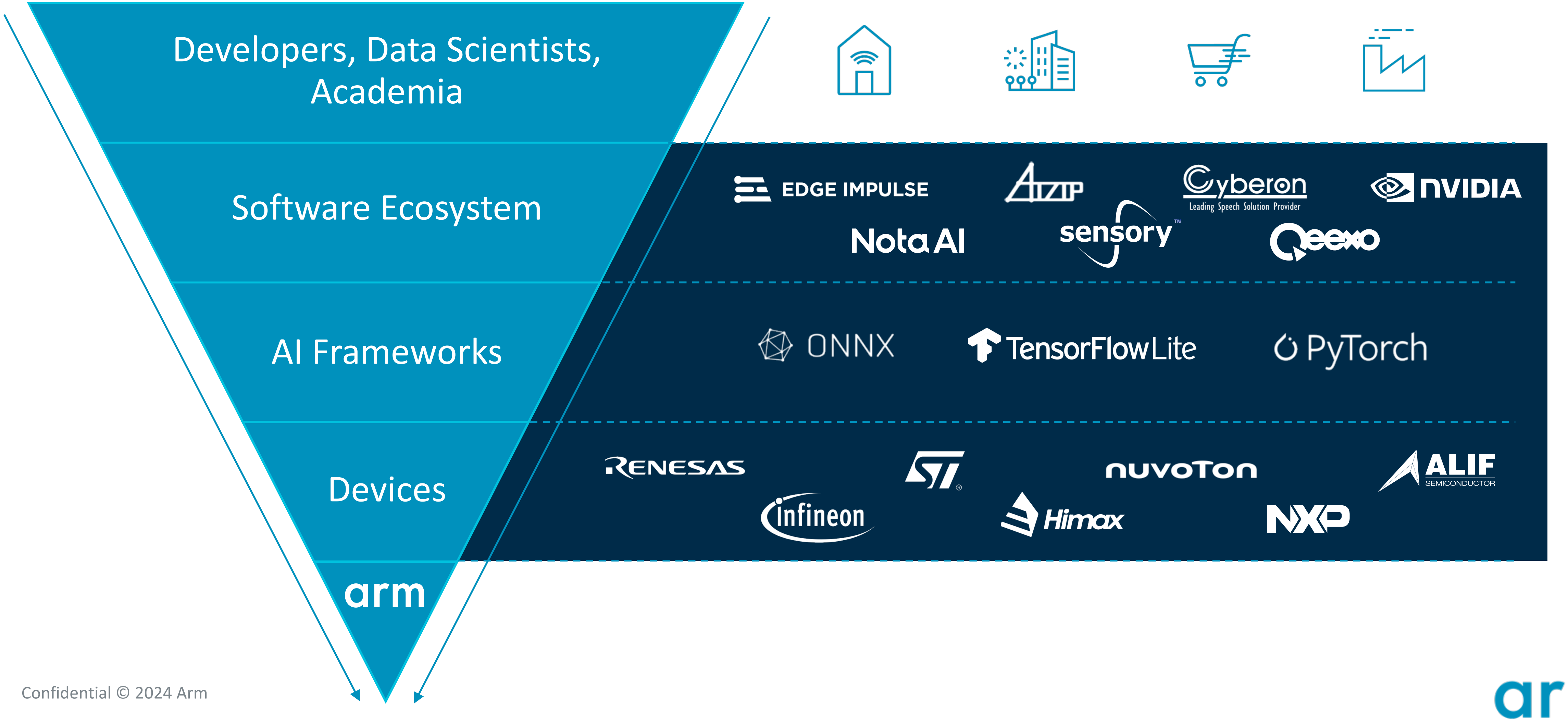
Linked platform runtime

- Train network in TensorFlow
- Model conditioning techniques: Collaborative clustering, pruning and QAT to improve model performance on Ethos-U while preserving its accuracy.
- Quantize it to Int8 TFL flatbuffer file (.tflite file)
- NN Optimizer identifies graphs to run on Ethos-U
  - Optimizes, schedules and allocates these graphs
  - Lossless compression, reducing size of tflite file

- Runtime executable file on device
- Accelerates kernels on Ethos-U
  - Driver handles the communication
- The remaining layers are executed on Cortex-M
  - CMSIS-NN optimized kernels if available
  - Fallback on the TFLµ reference kernels

The AI Software Ecosystem is Converging on Arm

Accelerating software development and navigating emerging frameworks



Developers, Data Scientists, Academia

Software Ecosystem

AI Frameworks

Devices

arm

arm

Foundation Tools & Software for Developing ML on Edge Devices

+ Ecosystem Partnerships

Reinhard Keil

AI on Edge Devices – Grow Opportunity

Energy Efficient AI Technology

www.52audio.com/archives/194825.html

Applied to New Emerging Markets

- Medical diagnoses
- Natural voice-controlled devices
- Vision based automation
- Sustainable agriculture
- etc.

ML on Edge Devices = compute + libraries + tools

Arm support for DSP/ML

Helium and Ethos-U Benefits: more computations

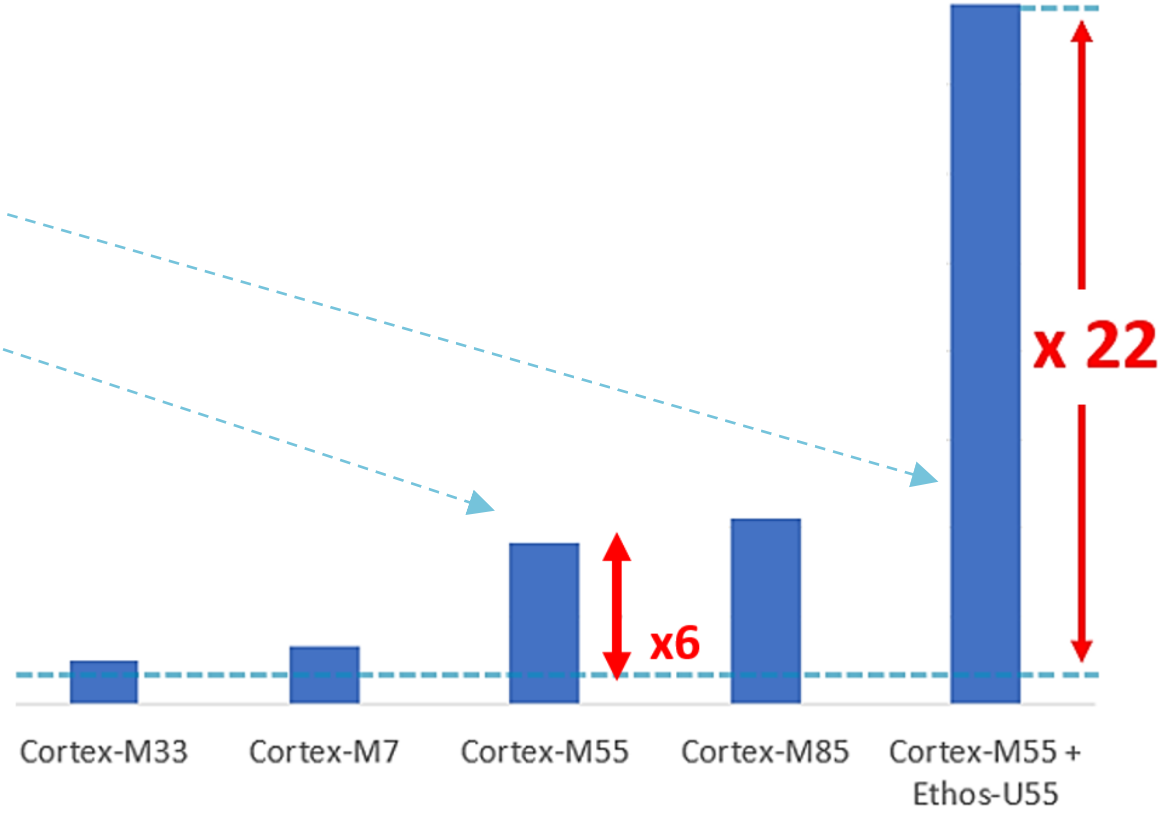
vector processing (MVE) and micro NPU
+ new DSP/ML kernels
+ stream-based PoC

Foundation Tools & Software: simplify development

- CMSIS-DSP/NN libraries, Python wrapper, Proof of concepts
- stream-based processing techniques
- Open software and tools platform

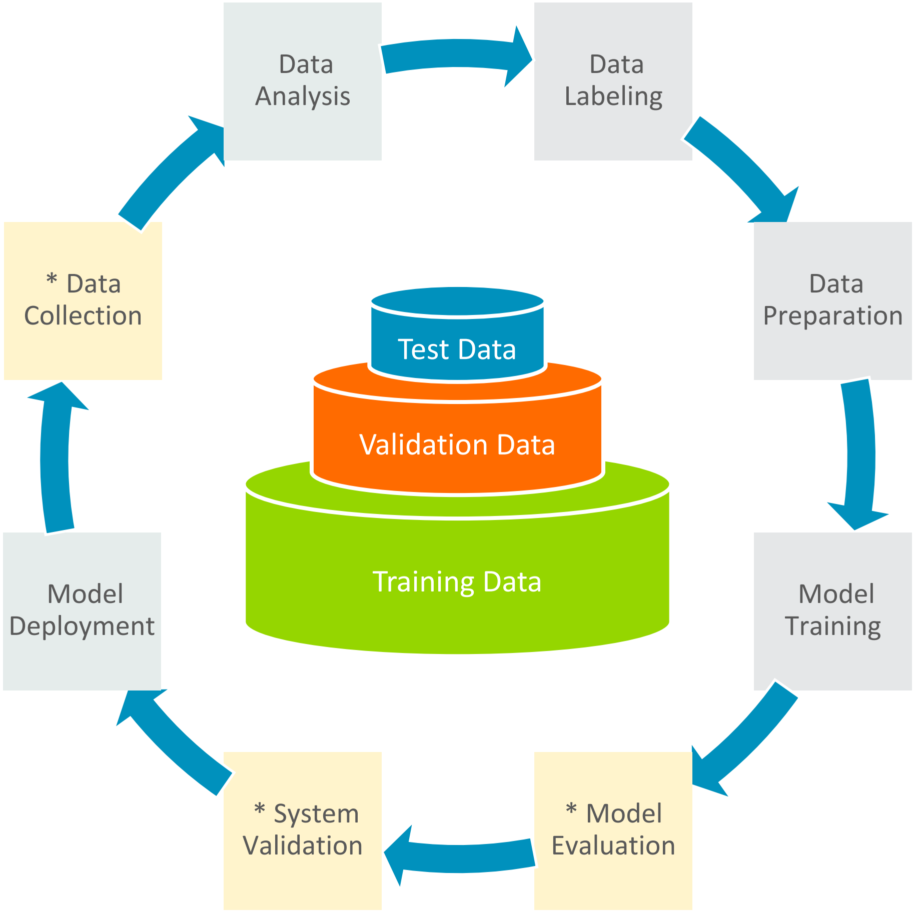
EEMBC AudioMark = DSP + NN Workload

(baseline Cortex-M4)



MLOps: deploy and maintain Machine Learning (ML) models

Combines machine learning data analytics with continuous development (DevOps)

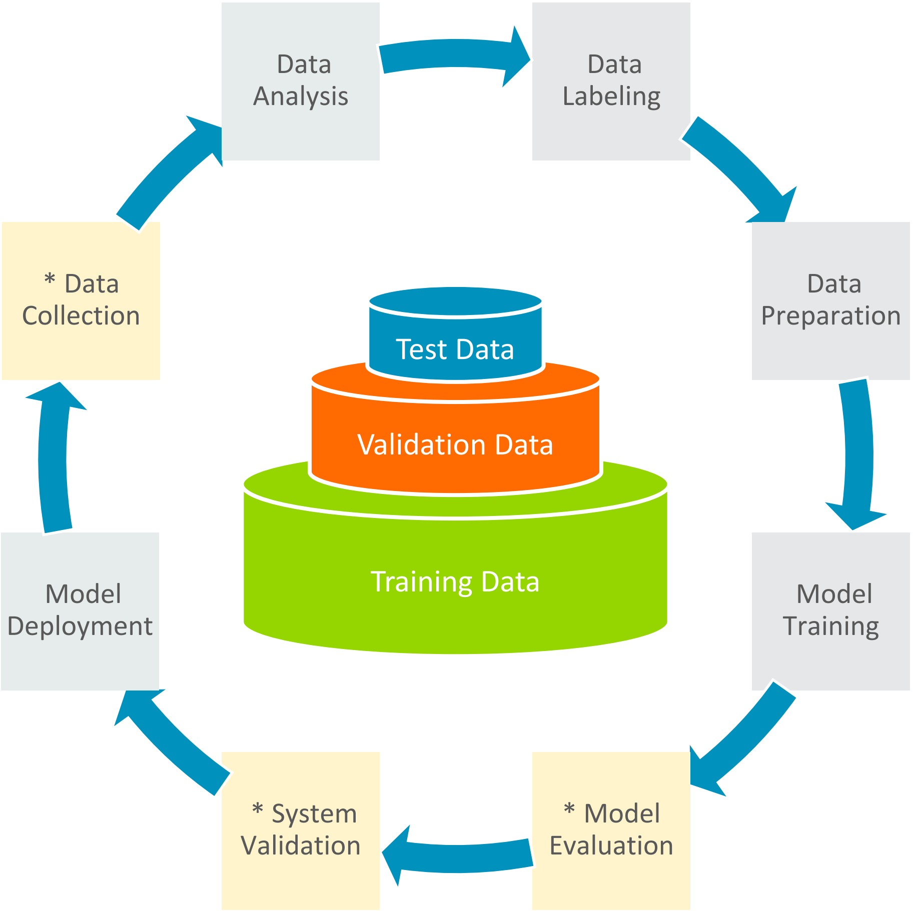


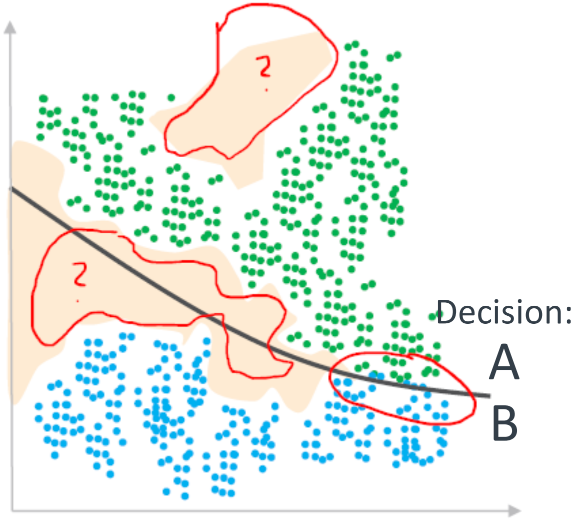
* Supported by Arm Virtual Hardware and SDS Framework

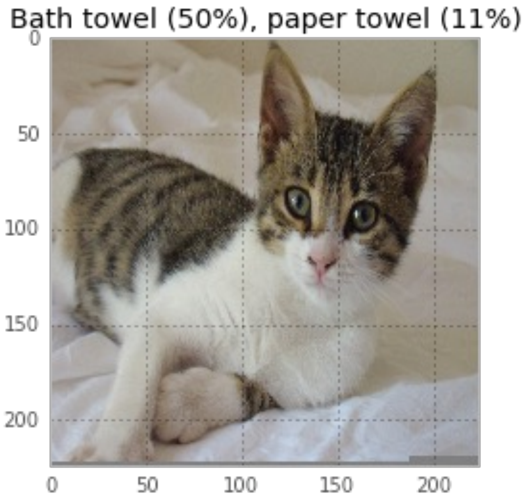
- ML models are tested and developed in isolated systems.
- MLOps is an iterative process to transition the ML model to production systems.
- Adding ML is an evolutionary process.
- Evaluation and validation require the model to run on target hardware.

Machine Learning (ML) Requires Real-World Data

Data collection requires frequently inputs of the final target system







Source:
https://codewords.recurse.com/issues/five/why-do-neural-networks-think-a-panda-is-a-vulture

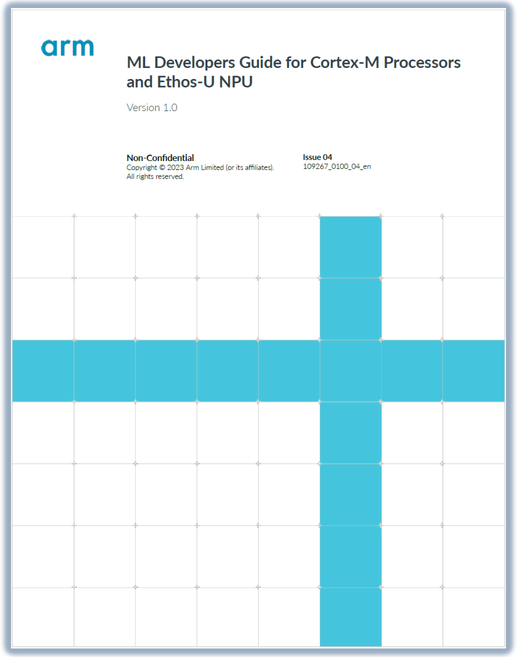
* Supported by Arm Virtual Hardware and SDS Framework

Material for ML Developers

- [ML Developers Guide for Cortex-M Processors and Ethos-U]
  - Embedded Developers that use microcontrollers and/or Ethos-U
  - MLOps system architects that integrate the various development tools
  - Data scientists that develop new ML models and need performance information

- Foundation Tool Components for MLOps Systems
  - Setup of a **Docker container** for MLOps systems
  - Create trained ML models with different compilers for target processors
  - Compare performance (inference time) using Arm Virtual Hardware

- Synchronous Data Stream (SDS) Framework
  - Flexible data stream management for sensor and audio data interfaces
  - Provides methods to **record real-world data** for analysis and development
  - **Playback real-world data** for algorithm validation using Arm Virtual Hardware



ML Developers Guide for Cortex-M Processors and Ethos-U NPU

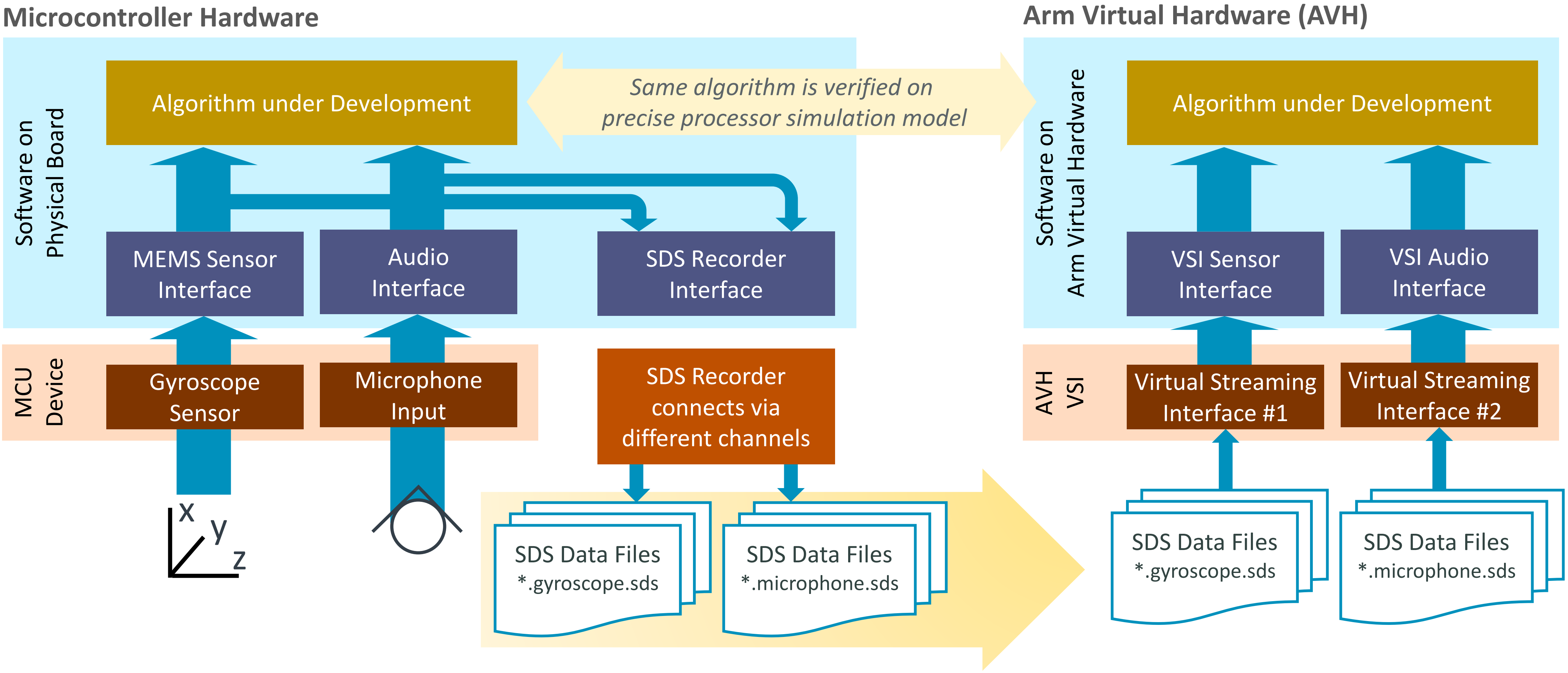
Version 1.0

Non-Confidential
Copyright © 2023 Arm Limited (or its affiliates).
All rights reserved.

Issue 04
109267_0100_04_en

SDS: recording & playback of real-world data for testing

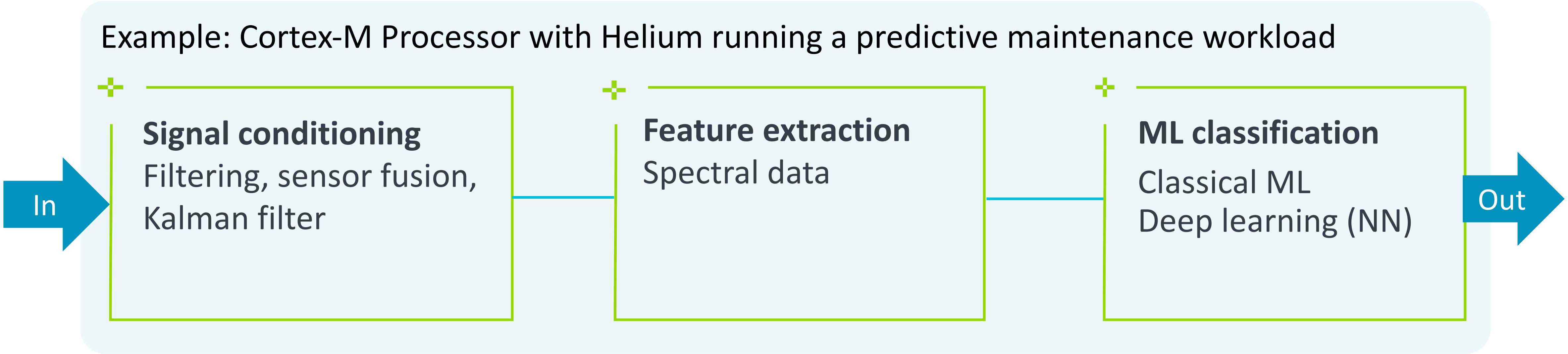
Combined with AVH it enables repeatable test automation in CI systems and MLOps cloud services



Microcontroller Hardware
Arm Virtual Hardware (AVH)
Software on Physical Board
Algorithm under Development
Same algorithm is verified on precise processor simulation model
MEMS Sensor Interface
Audio Interface
SDS Recorder Interface
MCU Device
Gyroscope Sensor
Microphone Input
SDS Recorder connects via different channels
x
y
z
SDS Data Files *.gyroscope.sds
SDS Data Files *.microphone.sds
Software on Arm Virtual Hardware
Algorithm under Development
VSI Sensor Interface
VSI Audio Interface
AVH VSI
Virtual Streaming Interface #1
Virtual Streaming Interface #2
SDS Data Files *.gyroscope.sds
SDS Data Files *.microphone.sds

Core Compute Capabilities for a Modern Development Flow

Access to DSP/ML capabilities without specialized tools simplifies development



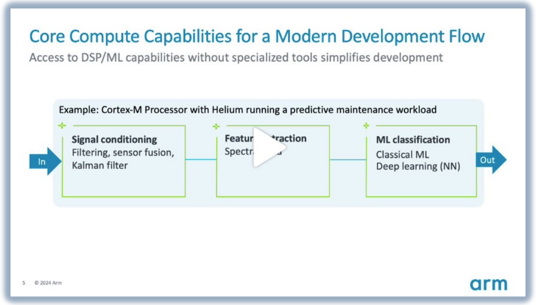
Example: Cortex-M Processor with Helium running a predictive maintenance workload

Signal conditioning
Filtering, sensor fusion,
Kalman filter

Feature extraction
Spectral data

ML classification
Classical ML
Deep learning (NN)

- [CMSIS-DSP](#) and [CMSIS-NN](#)
  Optimized compute libraries
- [CMSIS-Stream](#)
  Optimized data streaming between compute nodes



Core Compute Capabilities for a Modern Development Flow
Access to DSP/ML capabilities without specialized tools simplifies development

Example: Cortex-M Processor with Helium running a predictive maintenance workload

Signal conditioning
Filtering, sensor fusion,
Kalman filter

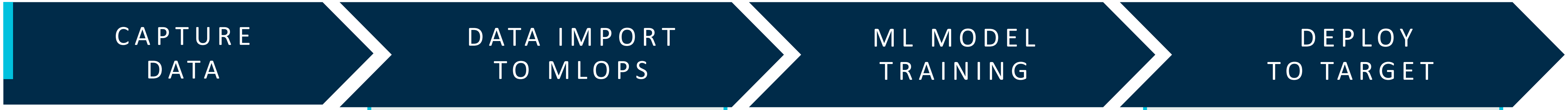
Featur traction
Spectr

ML classification
Classical ML
Deep learning (NN)

In
Out

5 © 2024 Arm

MLOps workflow exemplified with TDK Qeexo AutoML


CAPTURE DATA

DATA IMPORT TO MLOPS

ML MODEL TRAINING

DEPLOY TO TARGET

CAPTURE DATA

- Add SDS framework to target application
- Define sensors channels and capturing frequency
- Create metadata files to describe sensor data
- Capture SDS data files
- Verify SDS data files using a viewer

DATA IMPORT TO MLOPS

- Convert SDS data files and label data
- Upload data files to MLOps system, for example Qeexo AutoML

SEGMENT SET VERSION: 1

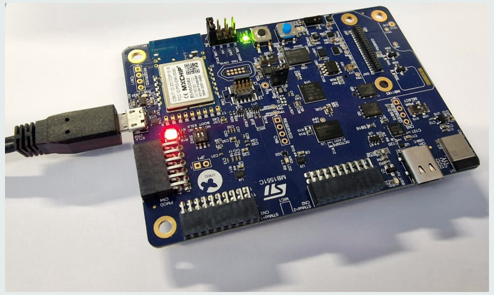
PLOT TYPE: Time-domain

ML MODEL TRAINING

- Data cleaning and preprocessing
- Feature extraction
- ML model selection
- Parameter optimization
- Model Validation
- Model conversion and download

DEPLOY TO TARGET

- Integrate ML model library in target project
- Validate ML model using Arm Virtual Hardware
- Final system test in target hardware

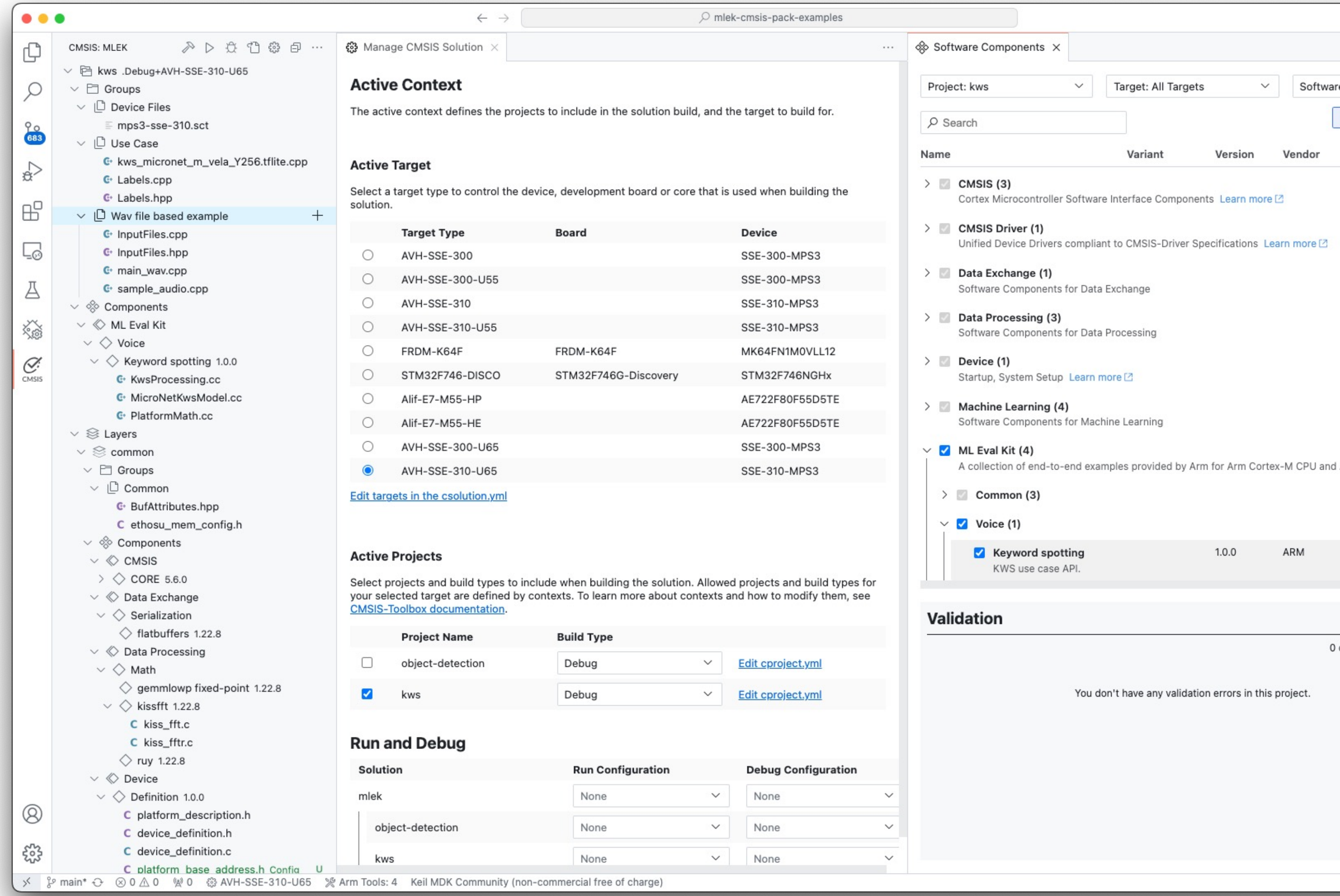


Tool Components for Microsoft Visual Studio Code

Joe Alderson

Flexible extensions for embedded and IoT

- A composable set of Microsoft Visual Studio Code extensions
- Use together as part of MDK v6 or separately
- Plug and play device support for debug probes and development boards
- Access to the CMSIS Pack ecosystem
- Integration of Open-CMSIS-Pack



```
kws .Debug+AVH-SSE-310-U65
  Groups
    Device Files
      mps3-sse-310.sct
    Use Case
      kws_micronet_m_vela_Y256.tflite.cpp
      Labels.cpp
      Labels.hpp
    Wav file based example                +
      InputFiles.cpp
      InputFiles.hpp
      main_wav.cpp
      sample_audio.cpp
  Components
    ML Eval Kit
      Voice
        Keyword spotting 1.0.0
          KwsProcessing.cc
          MicroNetKwsModel.cc
          PlatformMath.cc
  Layers
    common
      Groups
        Common
          BufAttributes.hpp
          ethosu_mem_config.h
      Components
        CMSIS
          CORE 5.6.0
        Data Exchange
          Serialization
            flatbuffers 1.22.8
        Data Processing
          Math
            gemmlowp fixed-point 1.22.8
            kissfft 1.22.8
              kiss_fft.c
              kiss_fftr.c
            ruy 1.22.8
        Device
          Definition 1.0.0
            platform_description.h
            device_definition.h
            device_definition.c
            platform_base_address.h Config   U
```

Manage CMSIS Solution

Active Context

The active context defines the projects to include in the solution build, and the target to build for.

Active Target

Select a target type to control the device, development board or core that is used when building the solution.

	Target Type	Board	Device
○	AVH-SSE-300		SSE-300-MPS3
○	AVH-SSE-300-U55		SSE-300-MPS3
○	AVH-SSE-310		SSE-310-MPS3
○	AVH-SSE-310-U55		SSE-310-MPS3
○	FRDM-K64F	FRDM-K64F	MK64FN1M0VLL12
○	STM32F746-DISCO	STM32F746G-Discovery	STM32F746NGHx
○	Alif-E7-M55-HP		AE722F80F55D5TE
○	Alif-E7-M55-HE		AE722F80F55D5TE
○	AVH-SSE-300-U65		SSE-300-MPS3
●	AVH-SSE-310-U65		SSE-310-MPS3

Edit targets in the csolution.yml

Active Projects

Select projects and build types to include when building the solution. Allowed projects and build types for your selected target are defined by contexts. To learn more about contexts and how to modify them, see CMSIS-Toolbox documentation.

	Project Name	Build Type	
□	object-detection	Debug	Edit cproject.yml
☑	kws	Debug	Edit cproject.yml

Run and Debug

Solution	Run Configuration	Debug Configuration
mlek	None	None
object-detection	None	None
kws	None	None

Software Components

Project: kws | Target: All Targets | Softwar

Search

Name	Variant	Version	Vendor
CMSIS (3) Cortex Microcontroller Software Interface Components Learn more			
CMSIS Driver (1) Unified Device Drivers compliant to CMSIS-Driver Specifications Learn more			
Data Exchange (1) Software Components for Data Exchange			
Data Processing (3) Software Components for Data Processing			
Device (1) Startup, System Setup Learn more			
Machine Learning (4) Software Components for Machine Learning			
☑ ML Eval Kit (4) A collection of end-to-end examples provided by Arm for Arm Cortex-M CPU and			
Common (3)			
☑ Voice (1)			
☑ Keyword spotting KWS use case API.		1.0.0	ARM

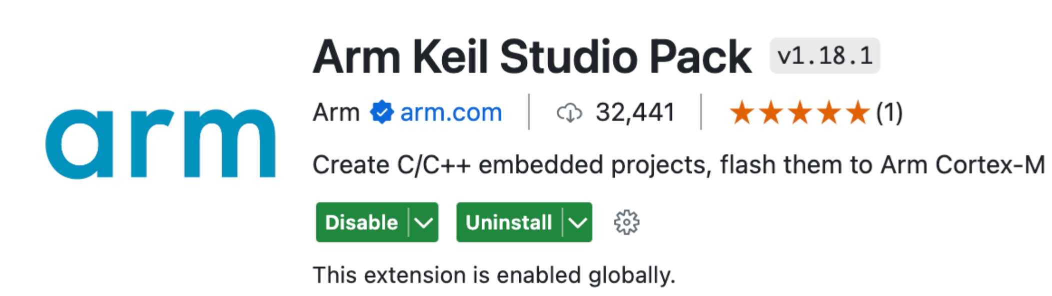
Validation

0

You don't have any validation errors in this project.

main* ⊗ 0 ⚠ 0 0 AVH-SSE-310-U65 Arm Tools: 4 Keil MDK Community (non-commercial free of charge)

Extend your tools with extensions and APIs

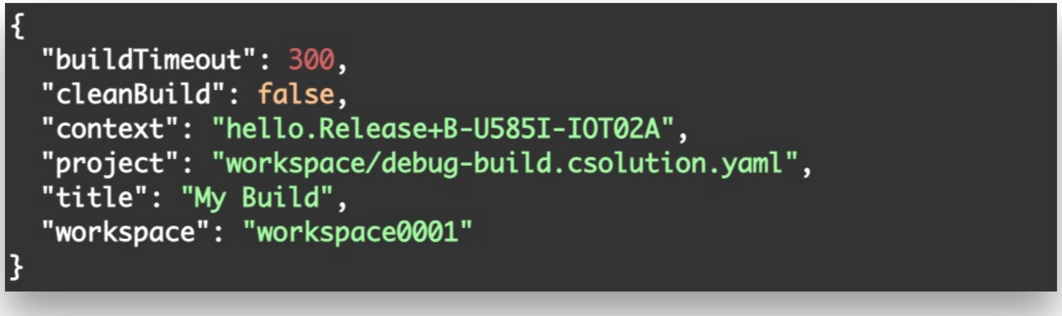


Arm Keil Studio Pack v1.18.1

Arm arm.com | 32,441 | ★★★★★ (1)

Create C/C++ embedded projects, flash them to Arm Cortex-M

This extension is enabled globally.

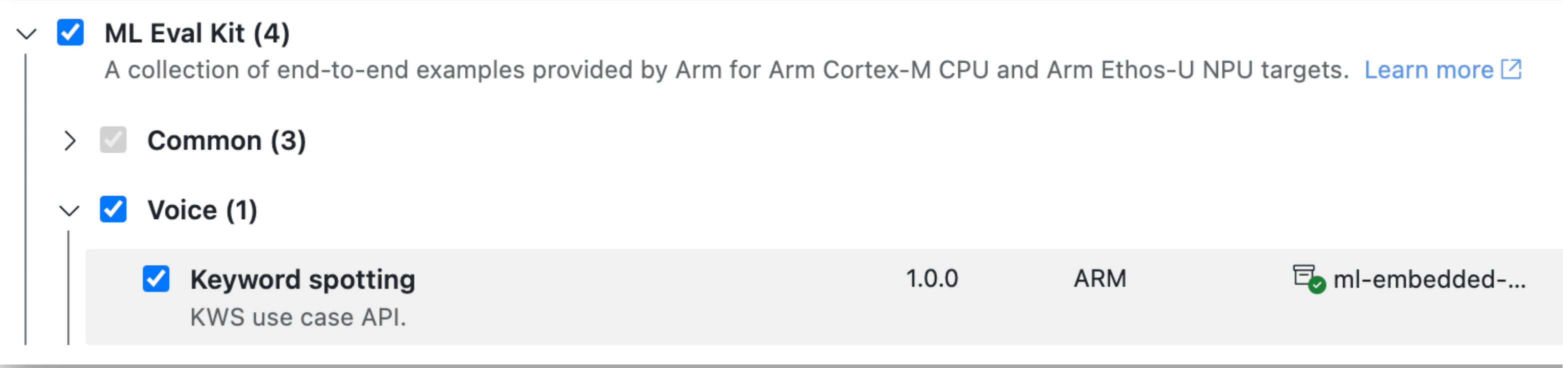


```
{
  "buildTimeout": 300,
  "cleanBuild": false,
  "context": "hello.Release+B-U585I-IOT02A",
  "project": "workspace/debug-build.csolution.yaml",
  "title": "My Build",
  "workspace": "workspace0001"
}
```

- Integrate Arm’s extensions into your tools products and workflows, creating the best overall experience for your developers
- **Use the VS Code dependency system or access the extension API directly**
- Access device information and software examples from the Open CMSIS Pack ecosystem through APIs to enhance your websites or tools products
- **Contact Arm about API access**

Validate middleware dependencies

- **Choose from professional middleware in thousands of CMSIS-Packs**
- Resolve dependencies across your stack automatically
- Download and install required CMSIS-Packs with a single click



☑ ML Eval Kit (4)
A collection of end-to-end examples provided by Arm for Arm Cortex-M CPU and Arm Ethos-U NPU targets. Learn more

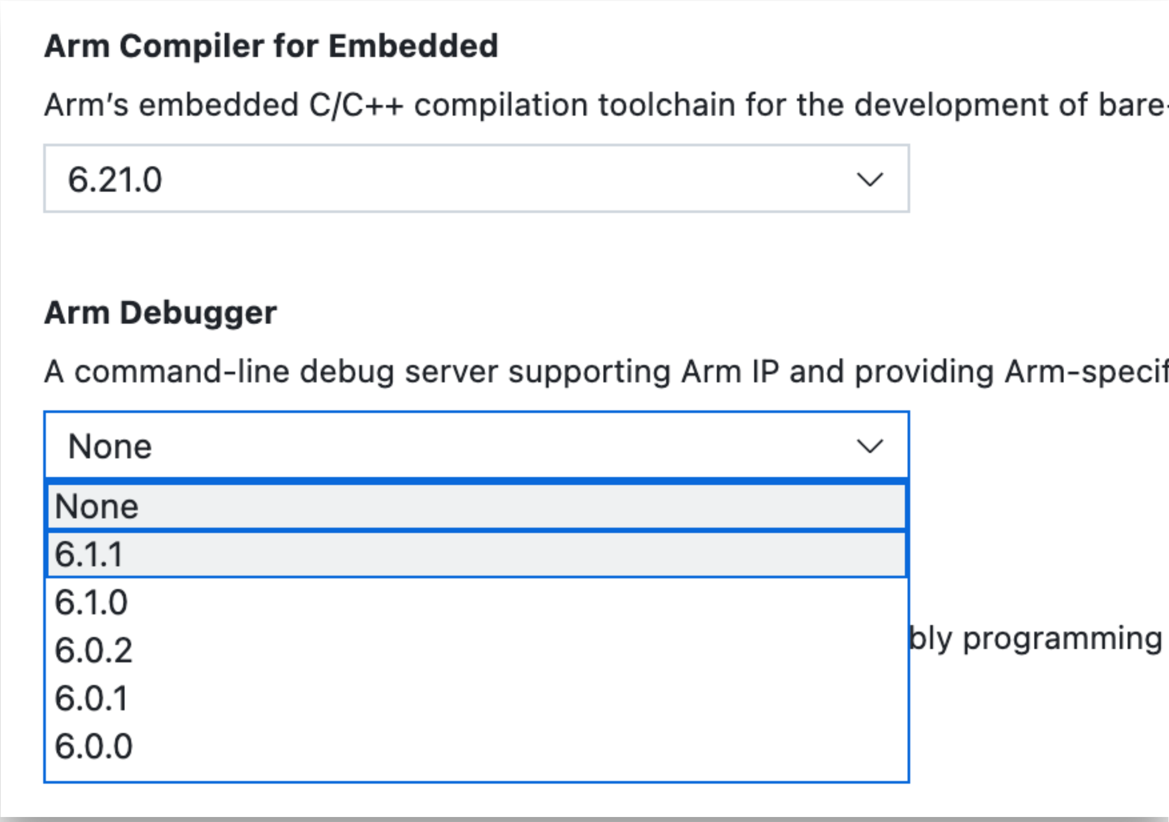
☑ Common (3)

☑ Voice (1)

☑ **Keyword spotting** 1.0.0 ARM ml-embedded-...
KWS use case API.

Pin tools versions

- **Keep your engineering team in sync across tools, source code and project settings**
- Create reproducible builds that pin your compiler, debugger, CMSIS toolbox and third party build tools like ninja
- Share your configuration with your team through source control



Arm Compiler for Embedded

Arm's embedded C/C++ compilation toolchain for the development of bare

6.21.0

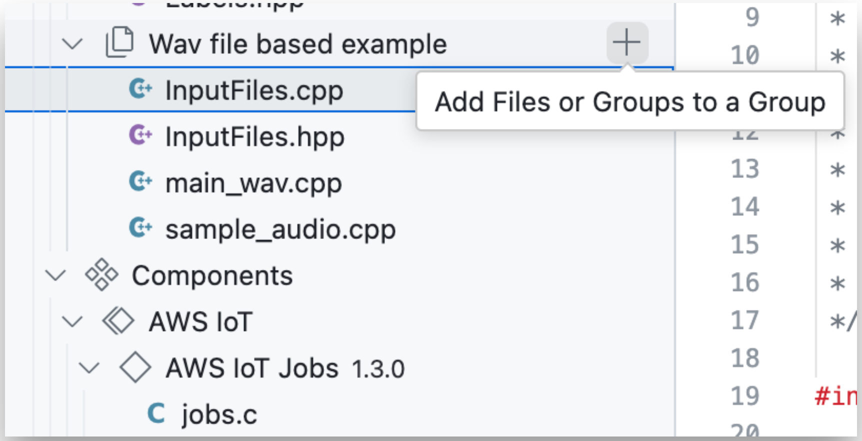
Arm Debugger

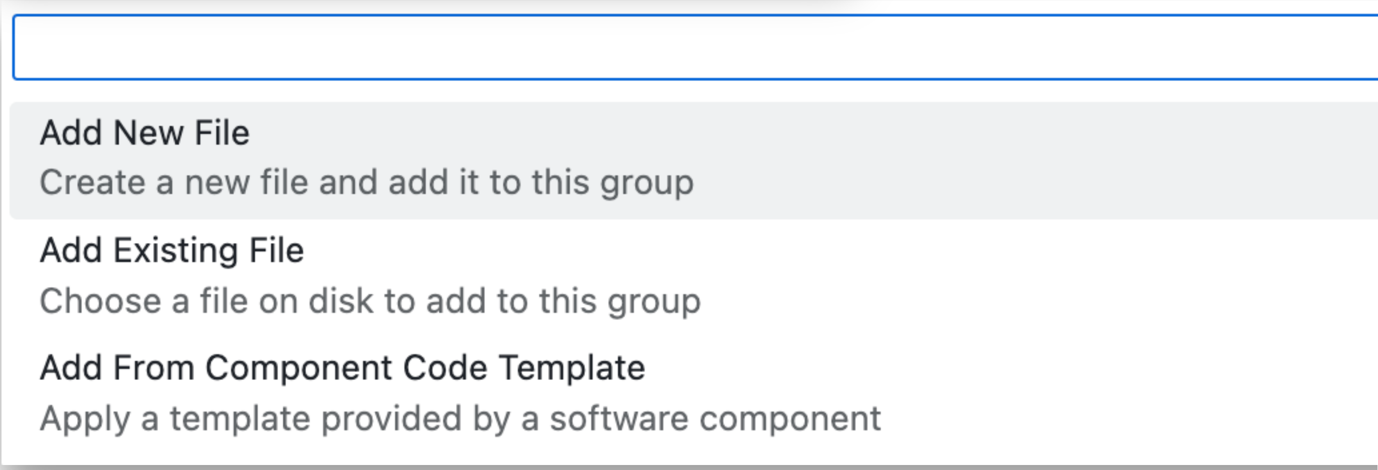
A command-line debug server supporting Arm IP and providing Arm-specif

None

Rapid prototyping with code templates

- **Copy code template files directly into your application to build up your solution quickly**
- Software components are shipped with example templates
- Once a component has been added, you can select the template easily from the solution outline view



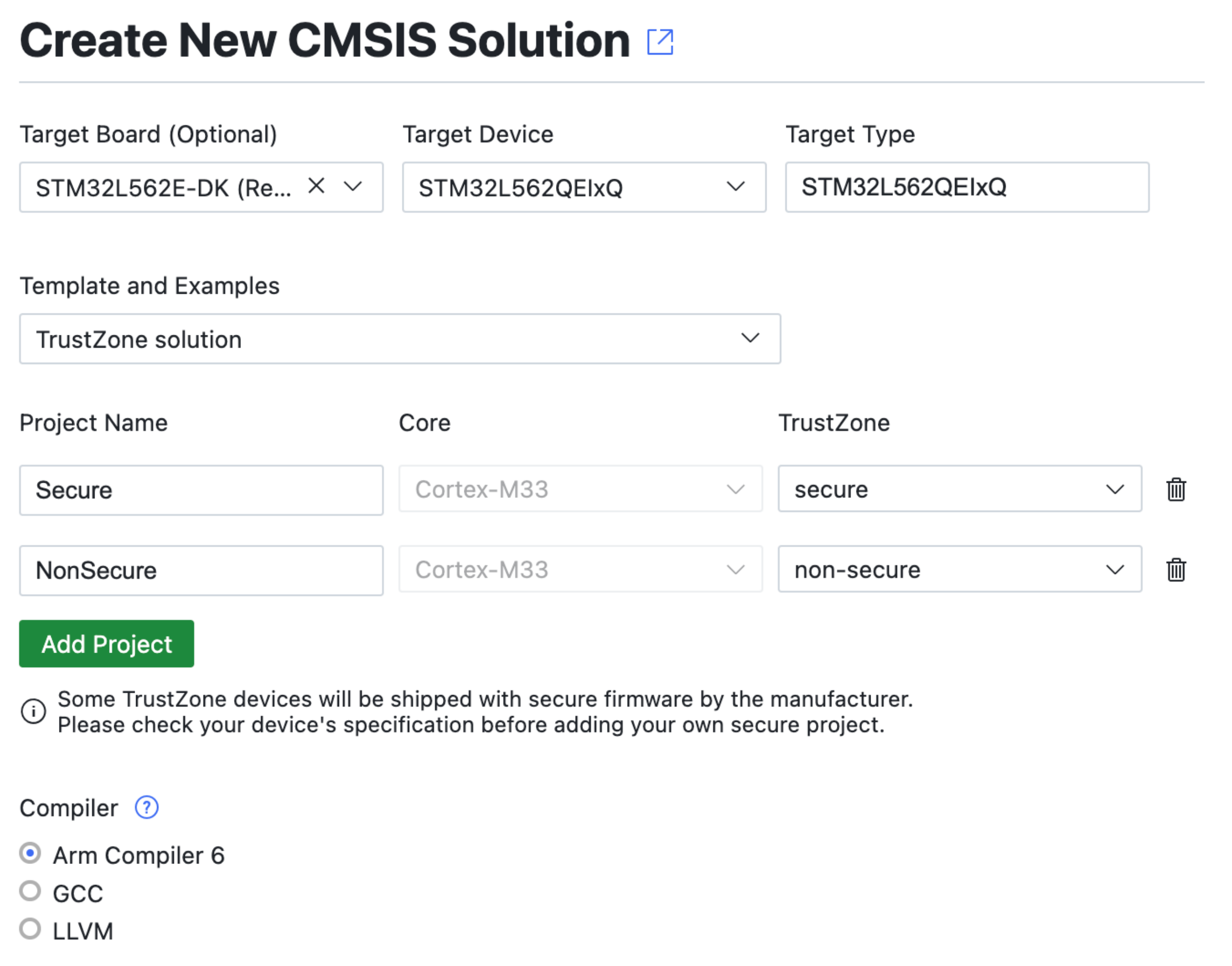

Add New File
Create a new file and add it to this group

Add Existing File
Choose a file on disk to add to this group

Add From Component Code Template
Apply a template provided by a software component

New solutions

- **Create your next solution from a basic template, or start from one of thousands of examples**
- Pre-set device and core information for 10,000+ MCUs and hundreds of development boards
- Pick from Arm Compiler 6, GCC or LLVM to get started
- Handle complex edits and flags directly in the Open CMSIS Pack yml files



Create New CMSIS Solution

Target Board (Optional)	Target Device	Target Type
STM32L562E-DK (Re...	STM32L562QEIxQ	STM32L562QEIxQ

Template and Examples: TrustZone solution

Project Name	Core	TrustZone
Secure	Cortex-M33	secure
NonSecure	Cortex-M33	non-secure

Add Project

ⓘ Some TrustZone devices will be shipped with secure firmware by the manufacturer. Please check your device's specification before adding your own secure project.

Compiler ?

● Arm Compiler 6
○ GCC
○ LLVM

Arm Keil Studio Pack – Essential VS Code Extensions

Project & Build	Description	Used Services
Arm CMSIS csolution (*)	Create and Manage CMSIS based projects	CMSIS-Toolbox (CMake, Ninja), Compiler (AC6, GCC, LLVM) Arm License Manager – for activation of Arm Compiler
Arm Environment Manager	Arm Tools installation and activation	MSFT vcpkg Arm License Manager – for activation of Arm Compiler
clangd (LLVM)	Intellisense	
YAML (RedHat)	YAML Language Support	

Debug	Description	Used Services
Arm Debugger	Debug for Cortex-M/A processors	Arm CLI Debugger, MSDAP
Arm Device Manager	Manages device connections and configuration for Arm Cortex-M	ULINK series, CMSIS-DAP, ST-Link, Arm Fixed Virtual Platforms
Eclipse CDT Cloud		MSDAP
Memory Inspector	Memory Window	
Peripheral Inspector	SVD supported access to peripherals	
Web Socket		

Tools Roadmap

Visual Studio Code - CMSIS	🟦 **MDK v6** - AC6.22 support - Keil Studio Desktop	- Enhanced local CMSIS Pack support - 3rd party code generator - µVision v5.40	◆ **MDK v6.1** - AC6.23 support - Reference Apps with Layer discovery		⬢ **MDK v6.x** - Cortex-A/M Support	
MDK-Middleware		🟦 **Middleware 8.0.0** - Free-to-use - For all compilers: AC6, GCC, IAR, LLVM		● **Middleware 8.1.0** - Maintenance		
CMSIS	🟦 **CMSIS-Toolbox 2.3.0** - Improved Cmake Backend Pre/Post build	⬢ **CMSIS 6.1.0** - Maintenance release - Cortex-M52 - C++ Vector Operations	◆ **CMSIS-Toolbox 2.4.0** - Maintenance - Collecting Requirements	● **CMSIS 6.2.0** - Collecting Requirements		
Visual Studio Code - Debug	🟦 **Arm Debugger 6.1.1** - Core register view - Memory inspector - Run on remote AVH - Debug connection config	◆ **Feature enhancement** - Off-chip memory support via scripting - Strategy for Cortex-A/M debug configuration	● **Cortex-A/M** - Initial multicore support - RTOS aware processes and threads stack view	● **Arm Debugger 6.x** - Enhanced disassembly view - Define future trace architecture	● **Security** - Secure debug authentication - Segger JLink	● **Trace and events** - Trace visualisation - Component viewer - Event recorder
	2024-CQ1	2024-CQ2	2024-CQ3	2024-CQ4	2025-CQ1	Future

🟦 Done ◆ Committed ● Tentative ⬢ Requested

arm

LLVM Embedded Toolchain (LLVM ET) for Arm - Status

- LLVM-embedded-toolchain-for-Arm in Github
- Components
  - clang
  - lld
  - LLVM binutils
  - picolibc C standard library
    - Experimental newlib is available – feedback is needed
    - Considering LLVM libc in future (currently incomplete)
  - LLVM libc++ C++ standard library
- Release follows the upstream LLVM schedule
  - Current 17.0.1
  - April 2024: 18.0.0
  - October 2024: 19.0.0
- How to get involved
  - Github project – report issues, create PRs
  - Working Group sync up – every 4 weeks

- LLVM ET vs. GCC

  Performance
  - GCC is a bit better on synthetic benchmarks on smaller cores
  - LLVM is much better on application benchmarks (like CMSIS DSP) on bigger cores, especially ones with MVE

  Code size
  - Density of code generation is similar
  - picolibc is comparable to newlib-nano

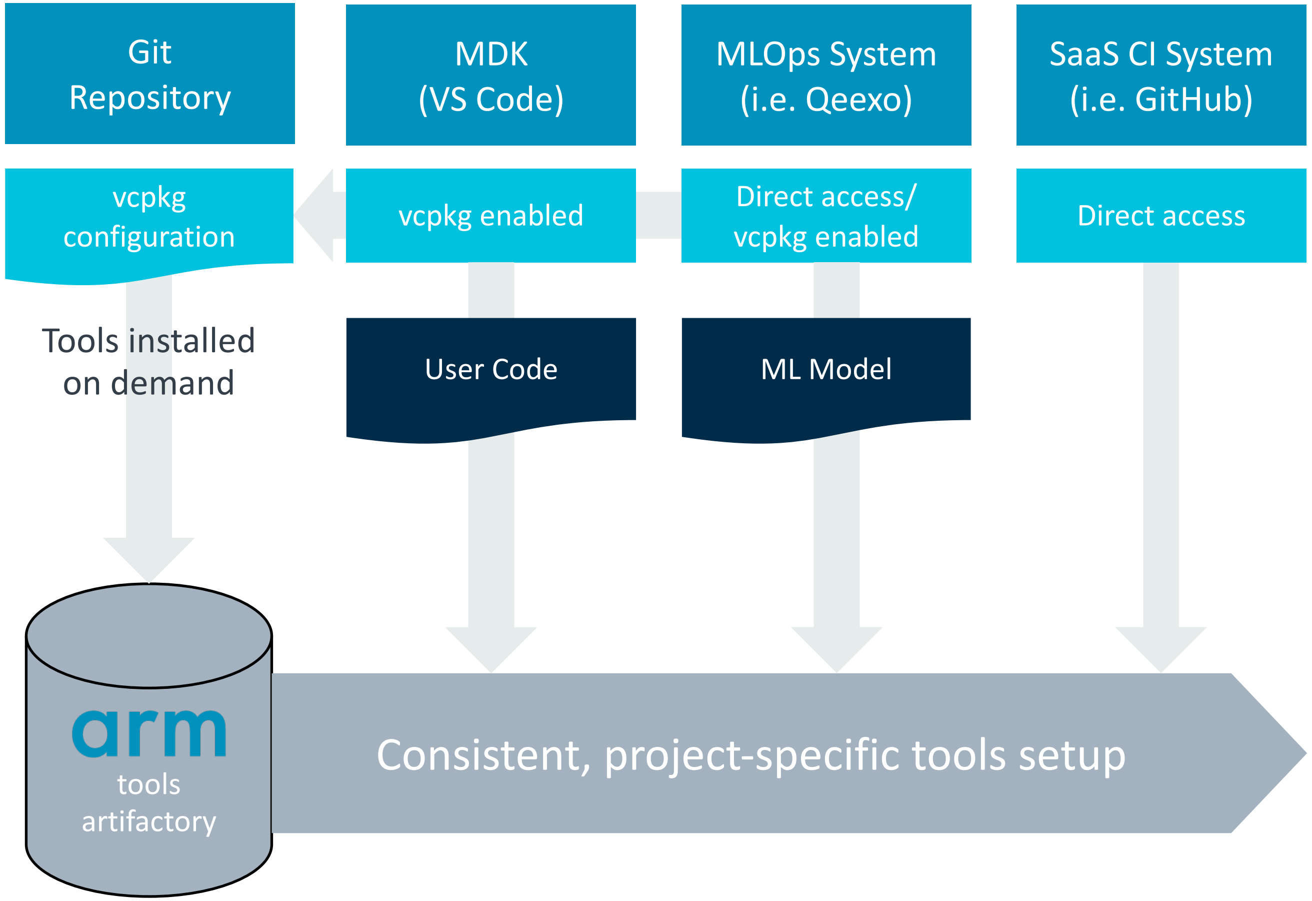
- LLVM ET vs. Arm Compiler 6
  - Most of optimization work is upstream'ed

  With optimize settings reaches ~95% on MVE
  - Jump threading (-mllvm -enable-dfa-jump-thread)
  - Inlining threshold (-mllvm -inline-threshold=500)
  - Loop unrolling threshold (-mllvm -unroll-threshold=450)

  Going forward provide an optimization tuning guide or predefined config files

Automated delivery of Arm tools

Tool deployment to MDK (VS Code), CI SaaS, and MLOps


Git Repository
vcpkg configuration
Tools installed on demand
MDK (VS Code)
vcpkg enabled
User Code
MLOps System (i.e. Qeexo)
Direct access/ vcpkg enabled
ML Model
SaaS CI System (i.e. GitHub)
Direct access
arm tools artifactory
Consistent, project-specific tools setup

- artifacts.tools.arm.com provides access to all tools for installation in different environments.
- Microsoft vcpkg simplifies the tool installation across various host systems.
  - The *vcpkg_configuration.json* file specifies the required tools.
  - Adding *vcpkg_configuration.json* to the project ensures consistent setup.
- MLOps and CI systems may access tools directly.
- Example for Docker setup: github.com/ARM-software/AVH-MLOps

It’s demo time

Joachim Krech

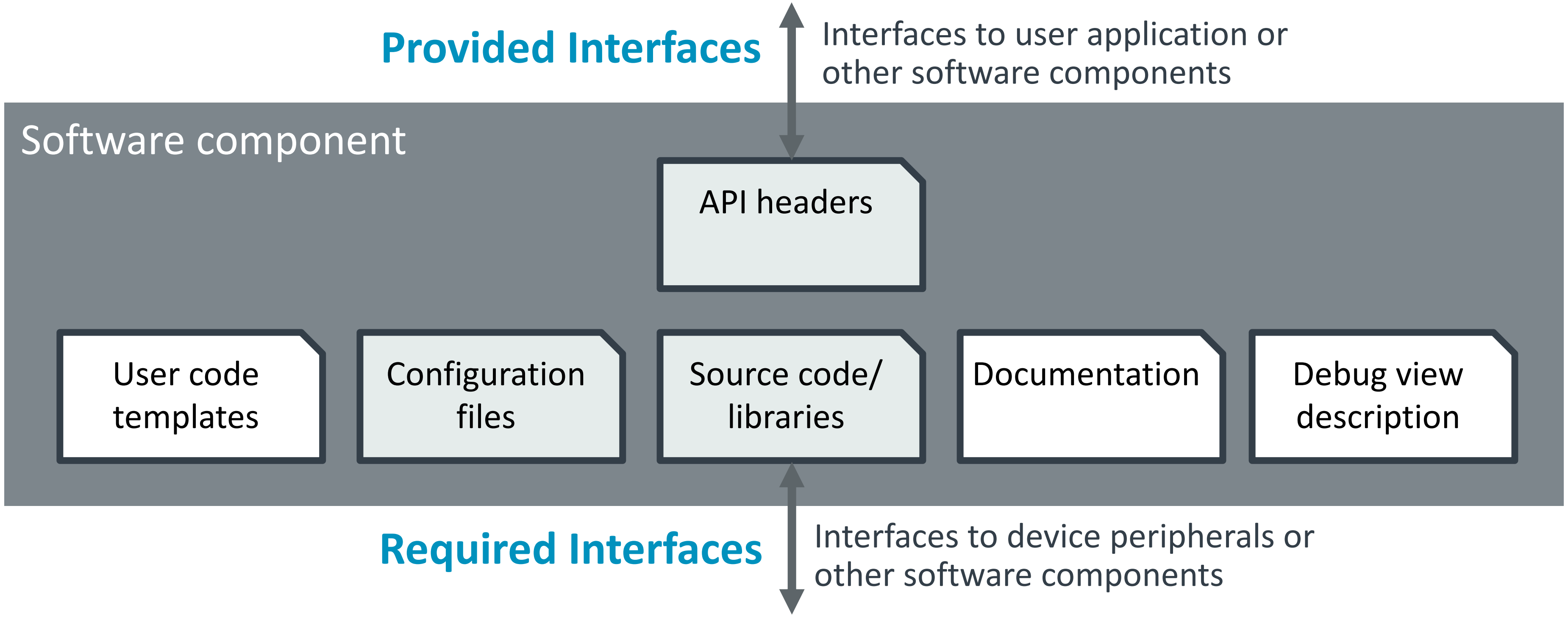
arm

Create Reusable Software Stacks

on arm

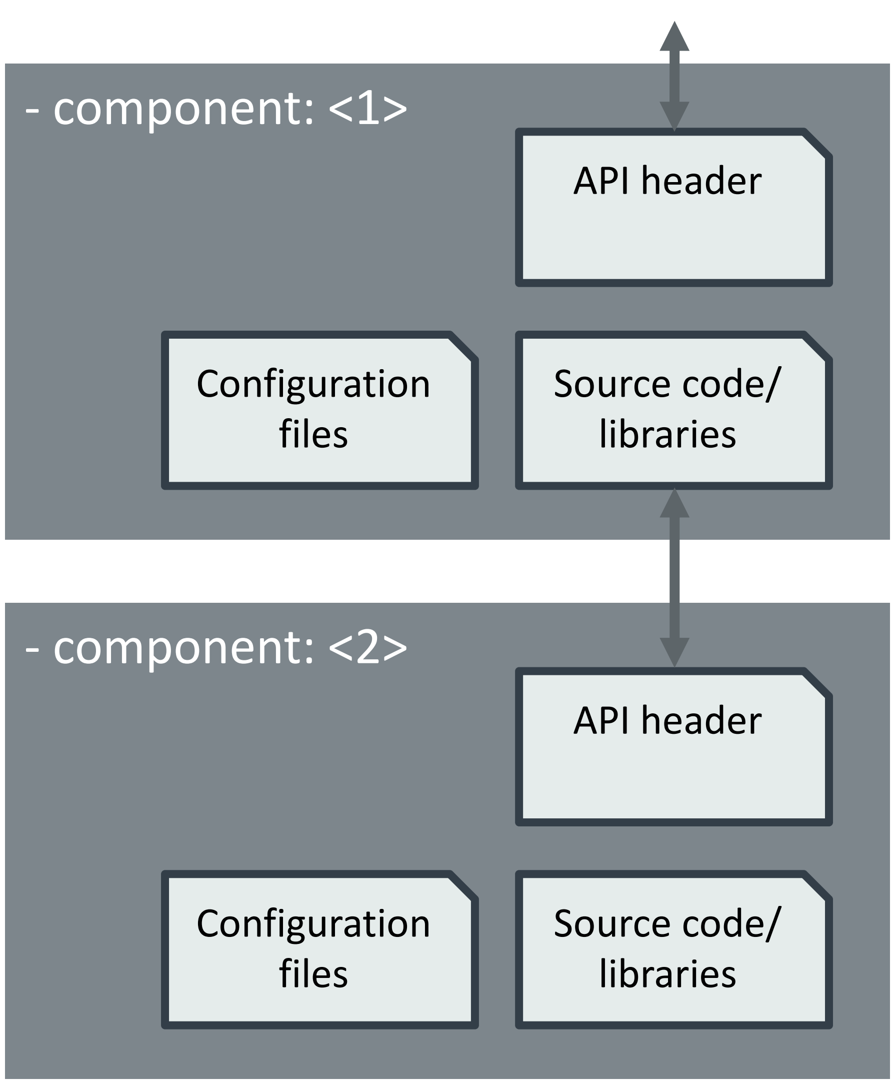
Reinhard Keil

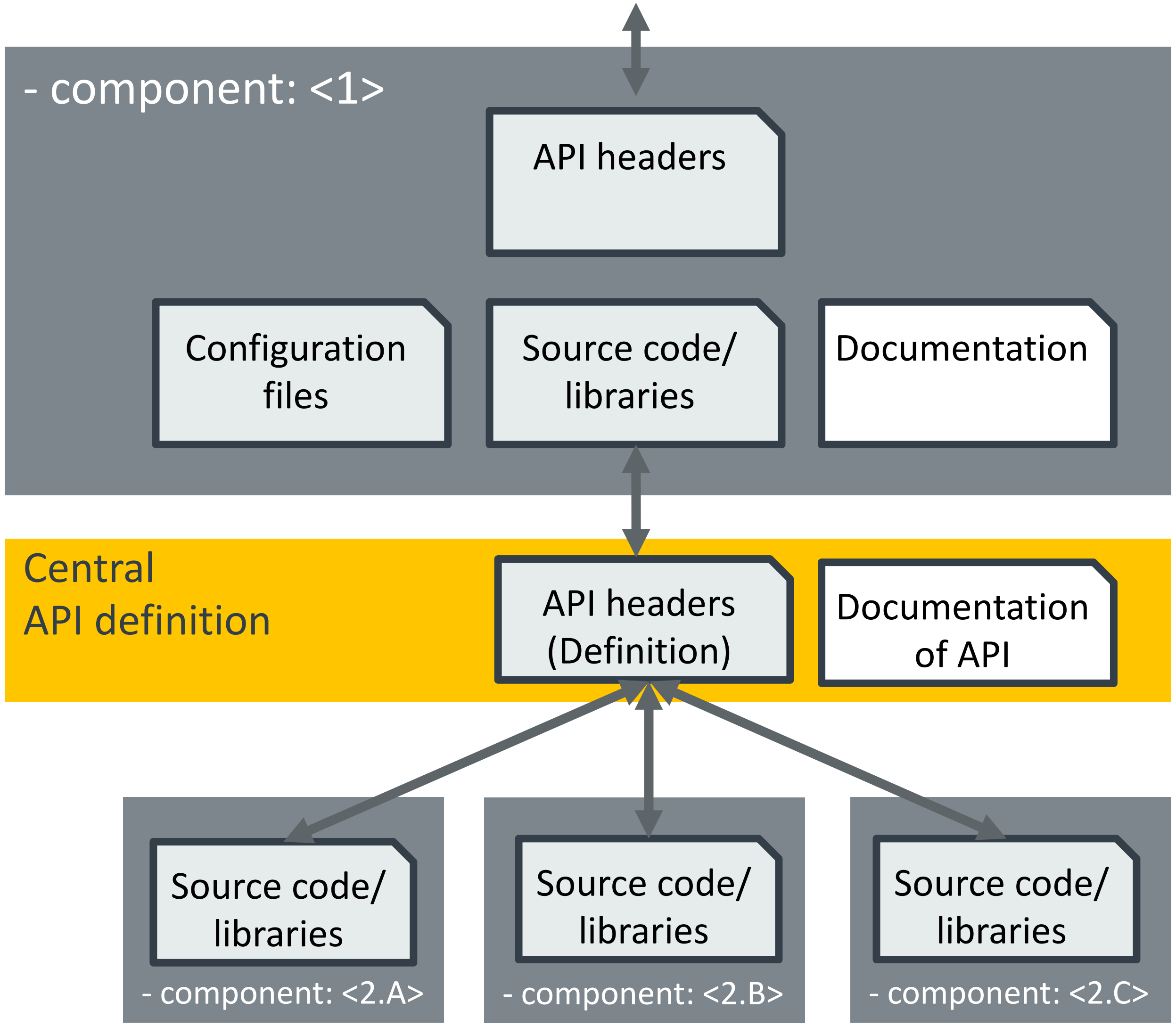
What is a software component?



- A software component encapsulates a set of related functions.
- Components should be substitutable by other components at design time.
- Components can have dependencies on other components.

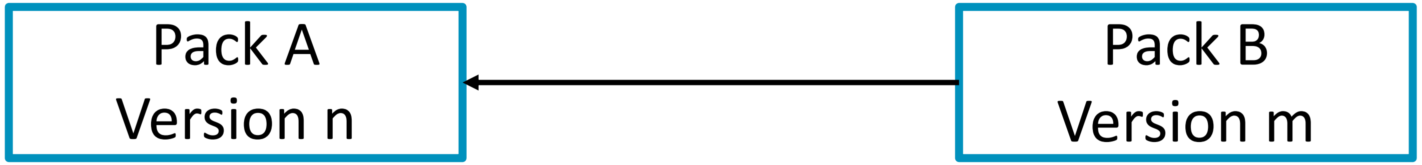
Connecting software components



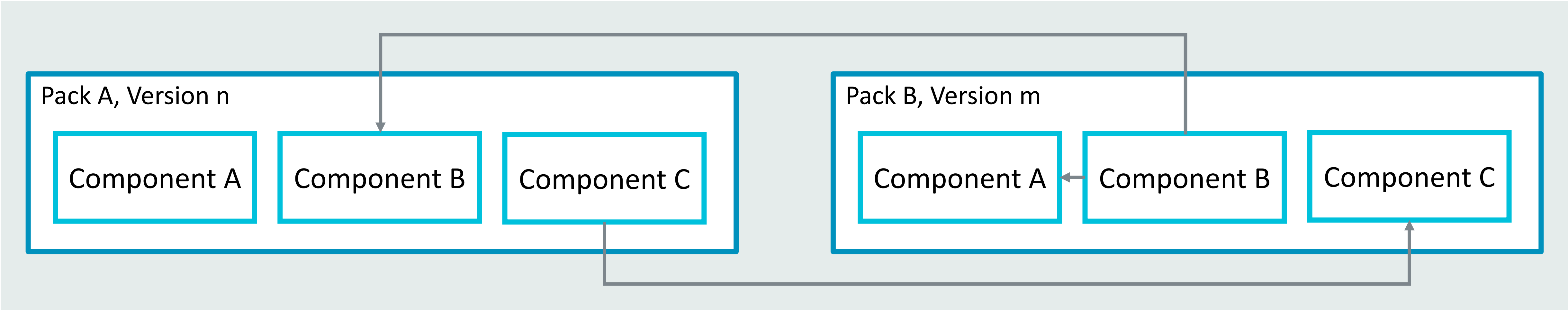


Relationships of packs and software components

- **Packs** can require other packs to be available:

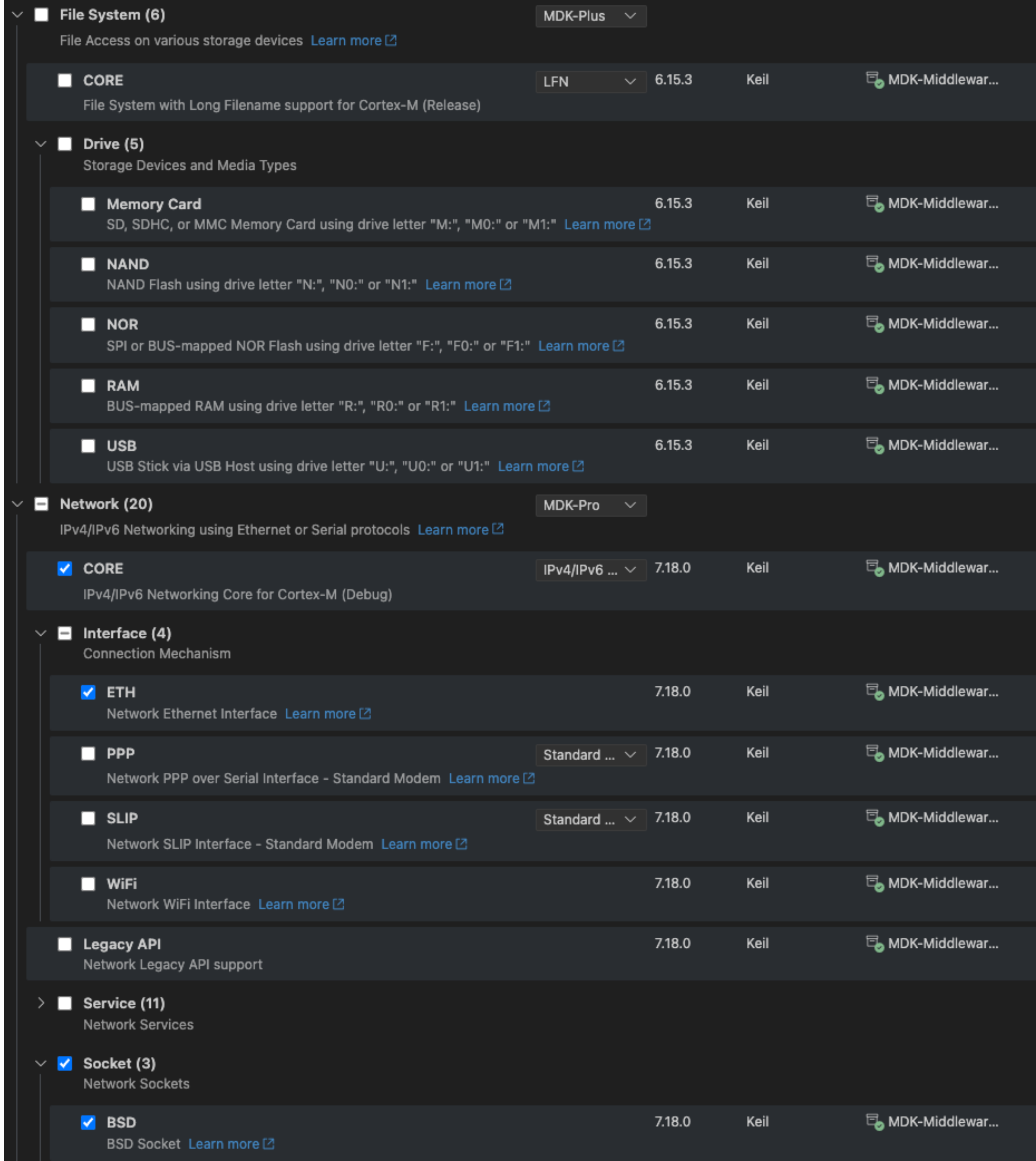


- **Components** can have dependencies on other components; either from the same or from other packs:



Example: MDK Middleware

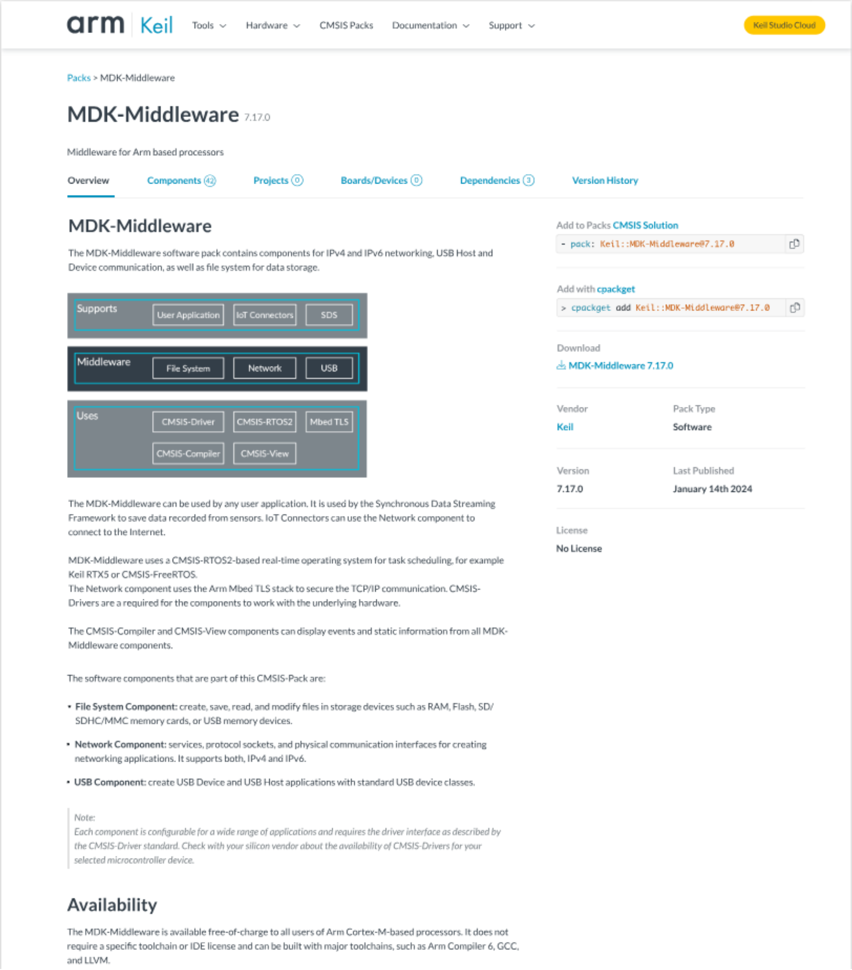
- Network
  - IPv4/IPv6 TCP/IP connectivity via Ethernet or serial connection
- USB
  - USB Host and USB Device support
  - High-performance, small footprint
  - No need for Windows/Linux drivers
- File System
  - ROM, RAM, Flash, SD/MMC/SDHC
  - FAT32 support
  - Simultaneous device access
- mbedTLS



☐ **File System (6)** File Access on various storage devices Learn more	MDK-Plus			
☐ **CORE** File System with Long Filename support for Cortex-M (Release)	LFN	6.15.3	Keil	MDK-Middlewar...
☐ **Drive (5)** Storage Devices and Media Types				
☐ **Memory Card** SD, SDHC, or MMC Memory Card using drive letter "M:", "M0:" or "M1:" Learn more		6.15.3	Keil	MDK-Middlewar...
☐ **NAND** NAND Flash using drive letter "N:", "N0:" or "N1:" Learn more		6.15.3	Keil	MDK-Middlewar...
☐ **NOR** SPI or BUS-mapped NOR Flash using drive letter "F:", "F0:" or "F1:" Learn more		6.15.3	Keil	MDK-Middlewar...
☐ **RAM** BUS-mapped RAM using drive letter "R:", "R0:" or "R1:" Learn more		6.15.3	Keil	MDK-Middlewar...
☐ **USB** USB Stick via USB Host using drive letter "U:", "U0:" or "U1:" Learn more		6.15.3	Keil	MDK-Middlewar...
⊟ **Network (20)** IPv4/IPv6 Networking using Ethernet or Serial protocols Learn more	MDK-Pro			
☑ **CORE** IPv4/IPv6 Networking Core for Cortex-M (Debug)	IPv4/IPv6 ...	7.18.0	Keil	MDK-Middlewar...
⊟ **Interface (4)** Connection Mechanism				
☑ **ETH** Network Ethernet Interface Learn more		7.18.0	Keil	MDK-Middlewar...
☐ **PPP** Network PPP over Serial Interface - Standard Modem Learn more	Standard ...	7.18.0	Keil	MDK-Middlewar...
☐ **SLIP** Network SLIP Interface - Standard Modem Learn more	Standard ...	7.18.0	Keil	MDK-Middlewar...
☐ **WiFi** Network WiFi Interface Learn more		7.18.0	Keil	MDK-Middlewar...
☐ **Legacy API** Network Legacy API support		7.18.0	Keil	MDK-Middlewar...
☐ **Service (11)** Network Services				
☑ **Socket (3)** Network Sockets				
☑ **BSD** BSD Socket Learn more		7.18.0	Keil	MDK-Middlewar...

Pack Datasheet

Overview Text (Readme)



MDK-Middleware 7.17.0

Middleware for Arm based processors

MDK-Middleware

The MDK-Middleware software pack contains components for IPv4 and IPv6 networking, USB Host and Device communication, as well as file system for data storage.

The MDK-Middleware can be used by any user application. It is used by the Synchronous Data Streaming Framework to save data recorded from sensors. IoT Connectors can use the Network component to connect to the Internet.

MDK-Middleware uses a CMSIS-RTOS2-based real-time operating system for task scheduling, for example Keil RTX5 or CMSIS-FreeRTOS.
The Network component uses the Arm Mbed TLS stack to secure the TCP/IP communication. CMSIS-Drivers are a required for the components to work with the underlying hardware.

The CMSIS-Compiler and CMSIS-View components can display events and static information from all MDK-Middleware components.

The software components that are part of this CMSIS-Pack are:

- **File System Component:** create, save, read, and modify files in storage devices such as RAM, Flash, SD/SDHC/MMC memory cards, or USB memory devices.
- **Network Component:** services, protocol sockets, and physical communication interfaces for creating networking applications. It supports both, IPv4 and IPv6.
- **USB Component:** create USB Device and USB Host applications with standard USB device classes.

> *Note:*
> *Each component is configurable for a wide range of applications and requires the driver interface as described by the CMSIS-Driver standard. Check with your silicon vendor about the availability of CMSIS-Drivers for your selected microcontroller device.*

Availability

The MDK-Middleware is available free-of-charge to all users of Arm Cortex-M-based processors. It does not require a specific toolchain or IDE license and can be built with major toolchains, such as Arm Compiler 6, GCC, and LLVM.

Add to Packs CMSIS Solution

```
- pack: Keil::MDK-Middleware@7.17.0
```

Add with cpackget

```
> cpackget add Keil::MDK-Middleware@7.17.0
```

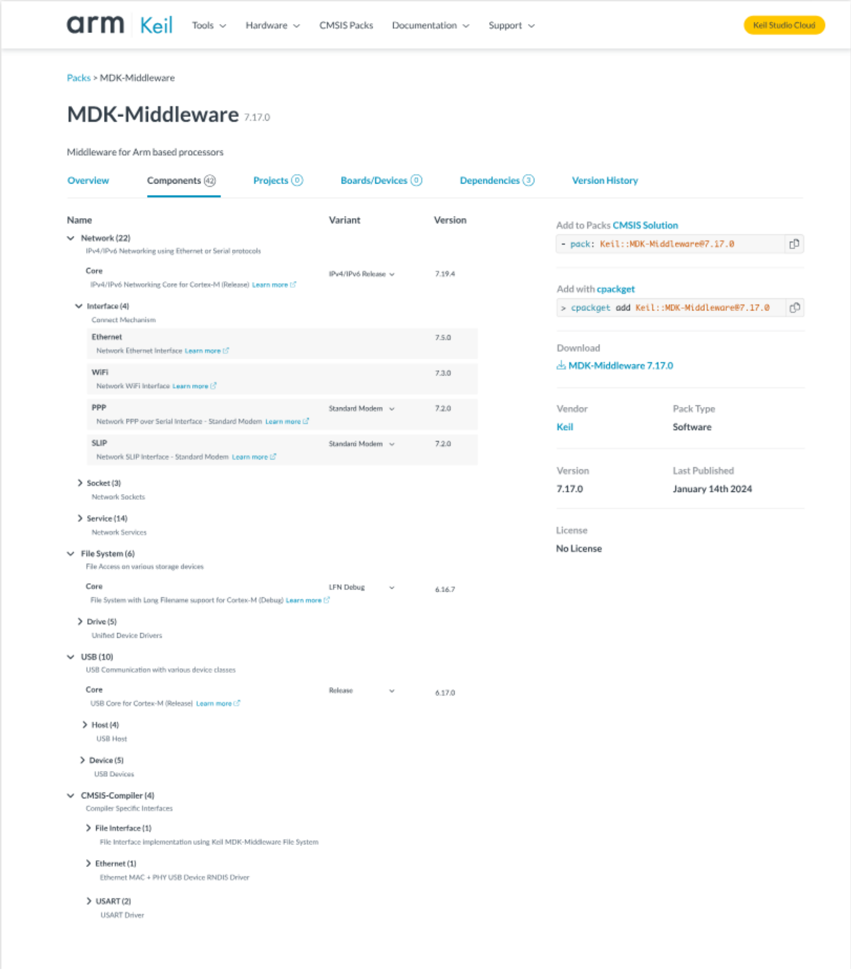
Download
MDK-Middleware 7.17.0

Vendor	Pack Type
Keil	Software

Version	Last Published
7.17.0	January 14th 2024

License
No License

List of all components



MDK-Middleware 7.17.0

Middleware for Arm based processors

Name	Variant	Version
Network (22) IPv4/IPv6 Networking using Ethernet or Serial protocols		
Core IPv4/IPv6 Networking Core for Cortex-M (Release) Learn more	IPv4/IPv6 Release	7.19.4
Interface (4) Connect Mechanism		
Ethernet Network Ethernet Interface Learn more		7.5.0
WiFi Network WiFi Interface Learn more		7.3.0
PPP Network PPP over Serial Interface - Standard Modem Learn more	Standard Modem	7.2.0
SLIP Network SLIP Interface - Standard Modem Learn more	Standard Modem	7.2.0
Socket (3) Network Sockets		
Service (14) Network Services		
File System (6) File Access on various storage devices		
Core File System with Long Filename support for Cortex-M (Debug) Learn more	LFN Debug	6.16.7
Drive (5) Unified Device Drivers		
USB (10) USB Communication with various device classes		
Core USB Core for Cortex-M (Release) Learn more	Release	6.17.0
Host (4) USB Host		
Device (5) USB Devices		
CMSIS-Compiler (4) Compiler Specific Interfaces		
File Interface (1) File Interface implementation using Keil MDK-Middleware File System		
Ethernet (1) Ethernet MAC + PHY USB Device RNDIS Driver		
USART (2) USART Driver		

Add to Packs CMSIS Solution

```
- pack: Keil::MDK-Middleware@7.17.0
```

Add with cpackget

```
> cpackget add Keil::MDK-Middleware@7.17.0
```

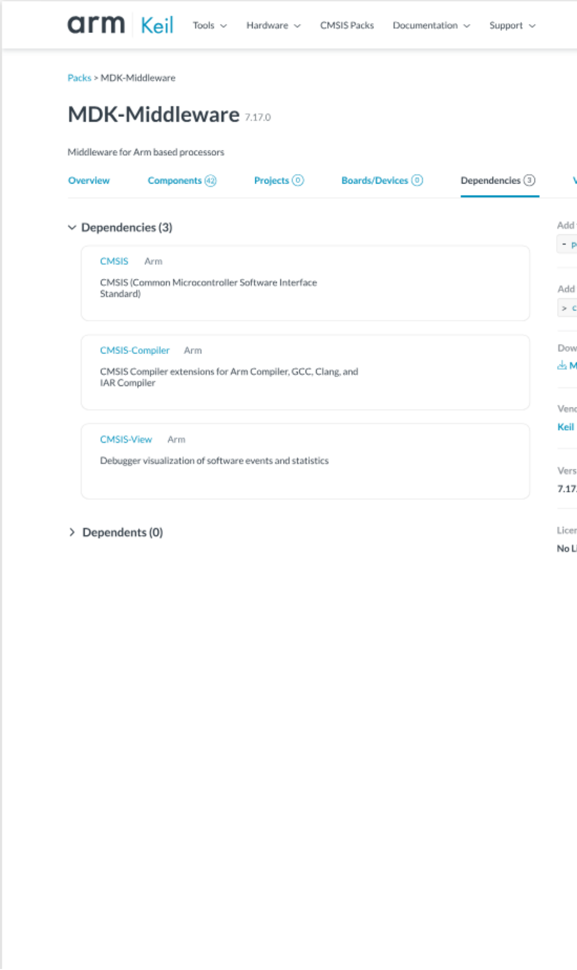
Download
MDK-Middleware 7.17.0

Vendor	Pack Type
Keil	Software

Version	Last Published
7.17.0	January 14th 2024

License
No License

Other packs required



MDK-Middleware 7.17.0

Middleware for Arm based processors

Dependencies (3)

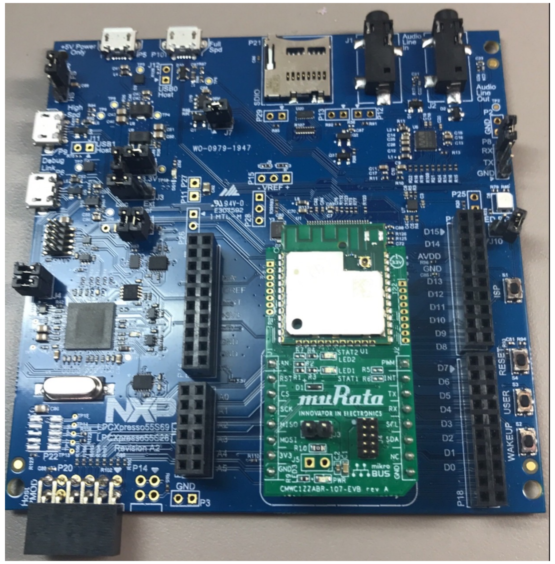
CMSIS Arm
CMSIS (Common Microcontroller Software Interface Standard)

CMSIS-Compiler Arm
CMSIS Compiler extensions for Arm Compiler, GCC, Clang, and IAR Compiler

CMSIS-View Arm
Debugger visualization of software events and statistics

Dependents (0)

Distribution of Reference Application Examples

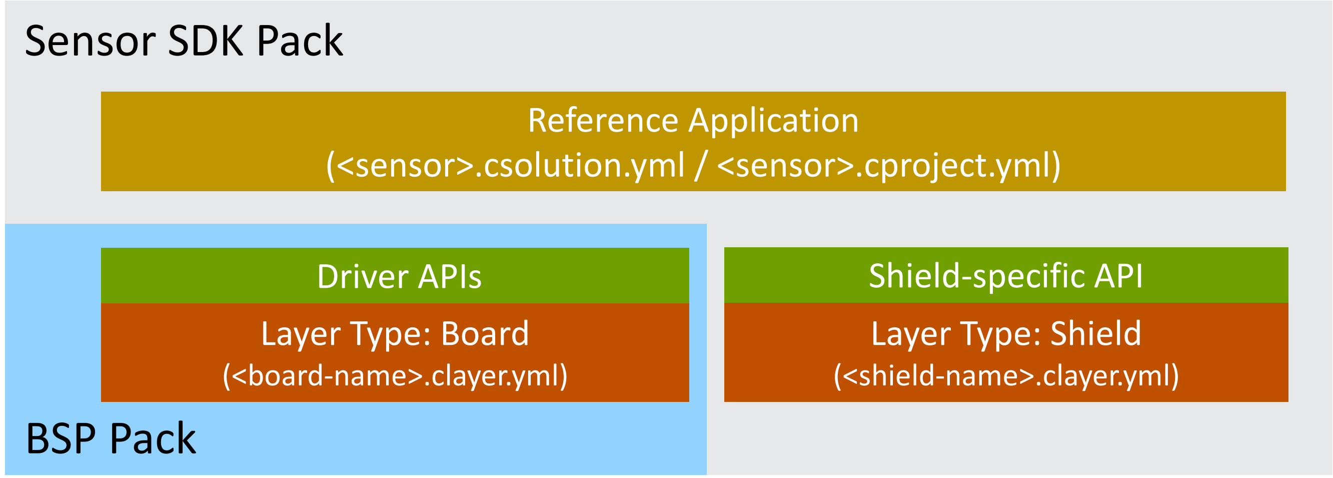


Example: Sensor SDK Pack (NXP_Sensor_SDK) that contains:

- Agnostic middleware for a sensor that is configurable (part of the Reference Application)
- Board/Device agnostic examples uses this middleware (part of the Reference Application)
- One or more Shield layers that provides configuration settings for the agnostic middleware

Board Layers are provided by a Board Support BSP Pack that is board specific

- Connections describe the compatibility of the different layers



Sensor SDK Pack

Reference Application
(<sensor>.csolution.yml / <sensor>.cproject.yml)

Driver APIs

Layer Type: Board
(<board-name>.clayer.yml)

Shield-specific API

Layer Type: Shield
(<shield-name>.clayer.yml)

BSP Pack

Sensor SDK Pack PDSC:

<example> describes Reference Application

<clayer> describes <shield-name>.clayer.yml

BSP Pack PDSC:

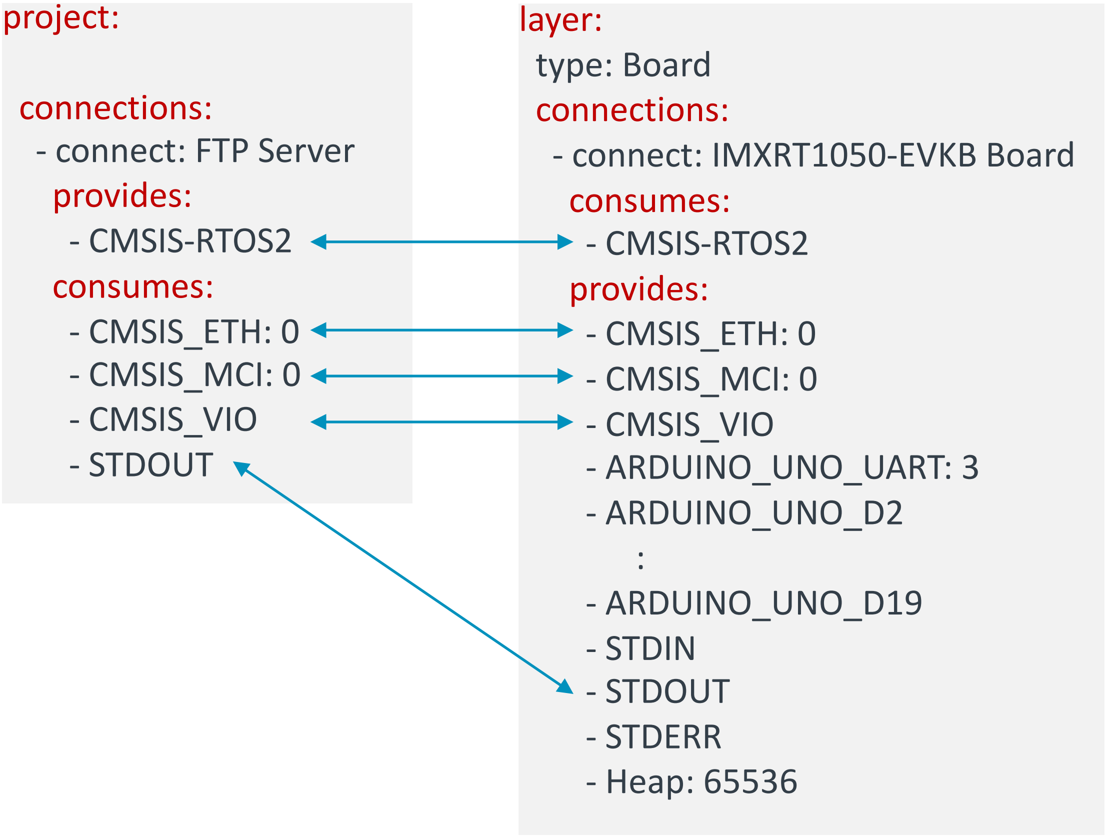
<clayer> describes <board-name>.clayer.yml

Reference Application Examples: MDK Middleware

Connections exemplified on MDK Middleware

Example cproject.yml

Board clayer.yml



```
project:

  connections:
    - connect: FTP Server
      provides:
        - CMSIS-RTOS2
      consumes:
        - CMSIS_ETH: 0
        - CMSIS_MCI: 0
        - CMSIS_VIO
        - STDOUT
```

```
layer:
  type: Board
  connections:
    - connect: IMXRT1050-EVKB Board
      consumes:
        - CMSIS-RTOS2
      provides:
        - CMSIS_ETH: 0
        - CMSIS_MCI: 0
        - CMSIS_VIO
        - ARDUINO_UNO_UART: 3
        - ARDUINO_UNO_D2
             :
        - ARDUINO_UNO_D19
        - STDIN
        - STDOUT
        - STDERR
        - Heap: 65536
```

Command-Line workflow:

```
solution:
  cdefault:

  target-types:
# Step 1: Specify your board, for example with:
#   - type: IMXRT1050-EVKB Board
#     board: NXP::IMXRT1050-EVKB
# Step 2: Run `cbuild setup` and use cbuild-idx.yml to identify variables
#     variables:
#       - Board-Layer:  ./layer/board/imxrt1050-iot.clayer.yml
```

IDE workflow:

1. User selects a reference example and specifies a board

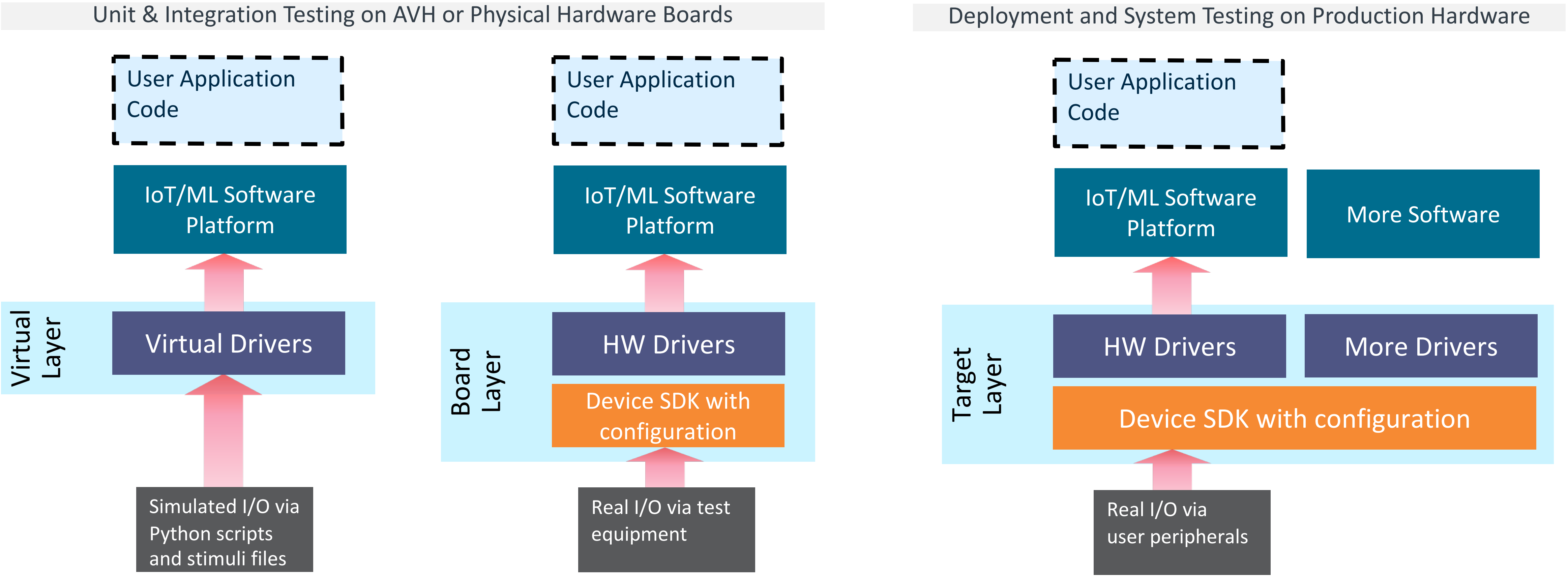
- IDE runs `cbuild setup` command, this generates `cbuild-idx.yml` with variable settings
  - This command installs a potential missing BSP and DFP pack
  - IDE shows one or more potential configurations

2. User selects a configuration

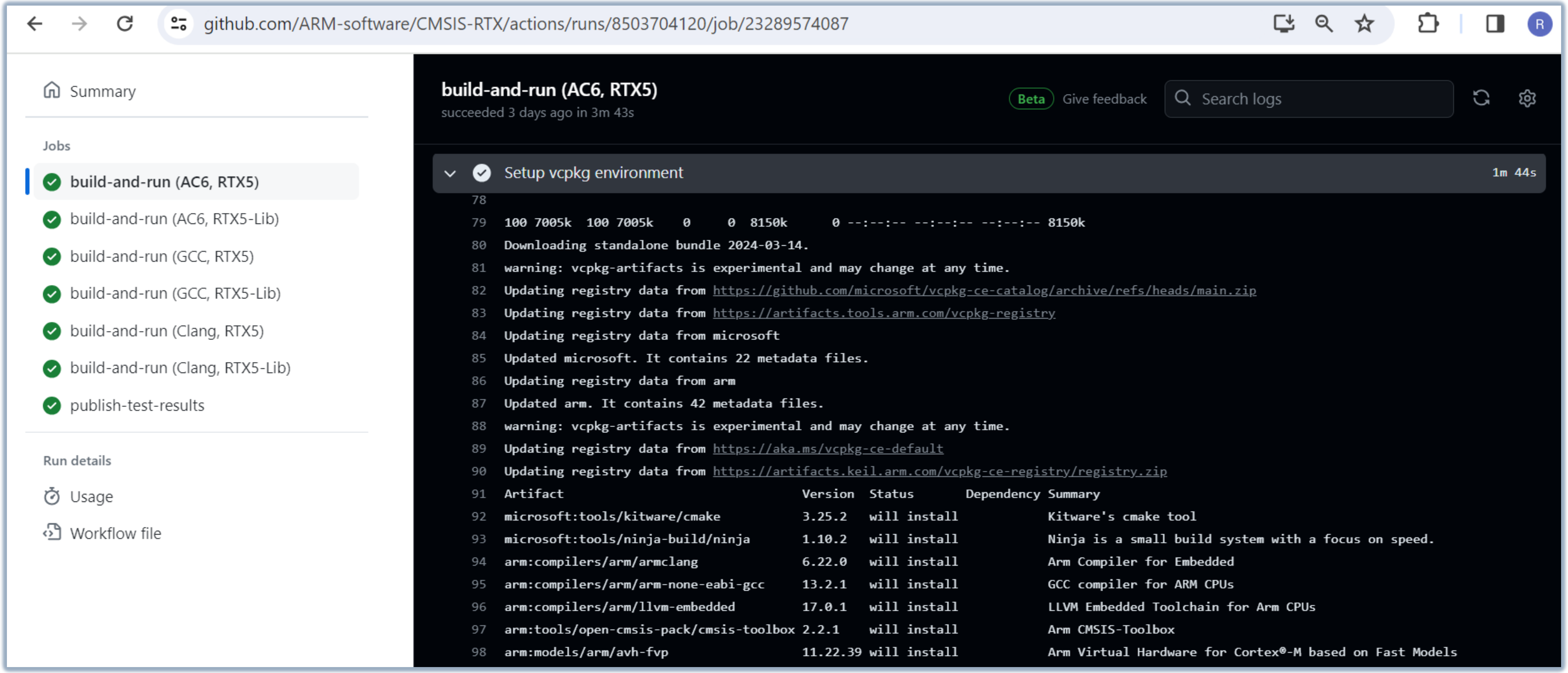
- IDE copies variable settings from `cbuild-idx.yml` to `csolution.yml` which adds the layers
  - Note: layers may be copied to workspace

Software Stack Validation – from Virtual to Physical Hardware

Validation on Arm Virtual Hardware (AVH) in CI systems; Deployment to physical devices


Unit & Integration Testing on AVH or Physical Hardware Boards
User Application Code
IoT/ML Software Platform
Virtual Layer
Virtual Drivers
Simulated I/O via Python scripts and stimuli files
User Application Code
IoT/ML Software Platform
Board Layer
HW Drivers
Device SDK with configuration
Real I/O via test equipment
Deployment and System Testing on Production Hardware
User Application Code
IoT/ML Software Platform
More Software
Target Layer
HW Drivers
More Drivers
Device SDK with configuration
Real I/O via user peripherals

Validate software stacks using different Compilers & AVH


github.com/ARM-software/CMSIS-RTX/actions/runs/8503704120/job/23289574087
Summary
Jobs
build-and-run (AC6, RTX5)
build-and-run (AC6, RTX5-Lib)
build-and-run (GCC, RTX5)
build-and-run (GCC, RTX5-Lib)
build-and-run (Clang, RTX5)
build-and-run (Clang, RTX5-Lib)
publish-test-results
Run details
Usage
Workflow file
Beta
Give feedback
Search logs
Setup vcpkg environment
1m 44s

build-and-run (AC6, RTX5)

succeeded 3 days ago in 3m 43s

```

100 7005k  100 7005k    0     0  8150k      0 --:--:-- --:--:-- --:--:-- 8150k
Downloading standalone bundle 2024-03-14.
warning: vcpkg-artifacts is experimental and may change at any time.
Updating registry data from https://github.com/microsoft/vcpkg-ce-catalog/archive/refs/heads/main.zip
Updating registry data from https://artifacts.tools.arm.com/vcpkg-registry
Updating registry data from microsoft
Updated microsoft. It contains 22 metadata files.
Updating registry data from arm
Updated arm. It contains 42 metadata files.
warning: vcpkg-artifacts is experimental and may change at any time.
Updating registry data from https://aka.ms/vcpkg-ce-default
Updating registry data from https://artifacts.keil.arm.com/vcpkg-ce-registry/registry.zip
Artifact                               Version  Status      Dependency Summary
microsoft:tools/kitware/cmake          3.25.2   will install           Kitware's cmake tool
microsoft:tools/ninja-build/ninja      1.10.2   will install           Ninja is a small build system with a focus on speed.
arm:compilers/arm/armclang             6.22.0   will install           Arm Compiler for Embedded
arm:compilers/arm/arm-none-eabi-gcc    13.2.1   will install           GCC compiler for ARM CPUs
arm:compilers/arm/llvm-embedded        17.0.1   will install           LLVM Embedded Toolchain for Arm CPUs
arm:tools/open-cmsis-pack/cmsis-toolbox 2.2.1   will install           Arm CMSIS-Toolbox
arm:models/arm/avh-fvp                 11.22.39 will install           Arm Virtual Hardware for Cortex®-M based on Fast Models
```

Virtual Workshop: Create Scalable Software Packs

11. June 2024, 15:00 – 17:00 GMT

Topics:

- Overall structure of a scalable software pack
- Tools for creating software packs
- Taxonomy of software components
- API interfaces
- Reference application examples
- Testing and validation

Register via: cmsis@arm.com

More CMSIS Innovations

- C++ Matrix and Vector Compute Algorithms
- CMSIS-SVD Improvements
- CMSIS-Pack Download Authorization

Reinhard Keil

C++ templates for matrix and vector operations (experimental)

- Documentation: https://arm-software.github.io/CMSIS-DSP/main/dsppp_main.html

Example:

```
constexpr int NB = 32;

Vector<float32_t,NB> a;
Vector<float32_t,NB> b;
Vector<float32_t,NB> c;
Vector<float32_t,NB> d = a + b * c;
```

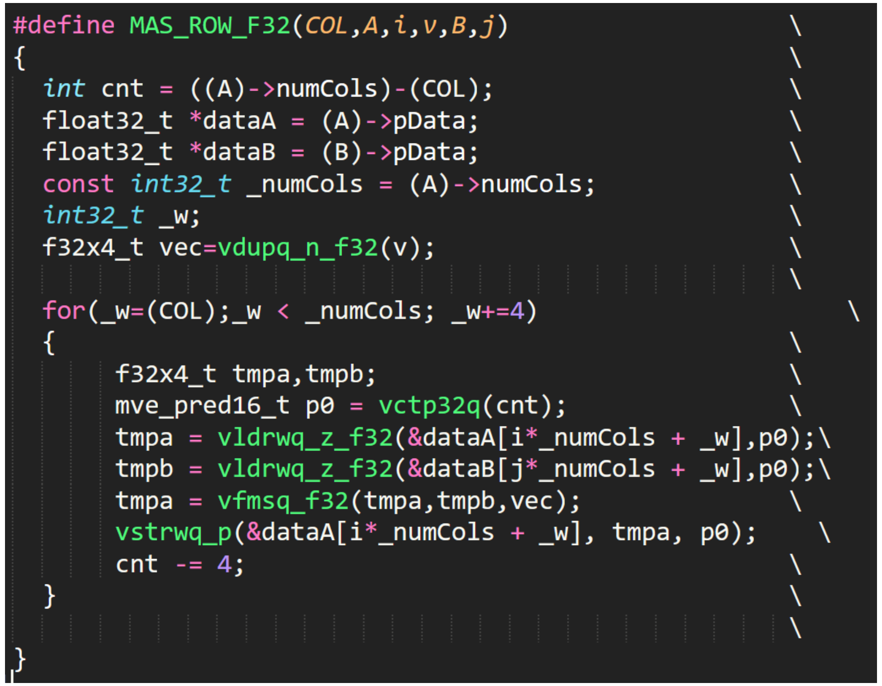
All vector operations (+,*) are done in one pass with one loop.
There is no more any temporary buffer.

- Matrix operations:
  - operators: +, -, *
  - dot for vector / matrix products
  - diagonal to create a diagonal matrix from a vector.
  - identity to create an identity matrix
  - transpose to create the transposed matrix
  - outer for the outer product of two vectors
  - VectorView (slice of a matrix)
  - MatrixView (subset of a matrix)

Please give feedback, i.e. via github.com/ARM-software/CMSIS-DSP/issues

Example: Matrix inverse (Gauss-Jordan with pivot algorithm)

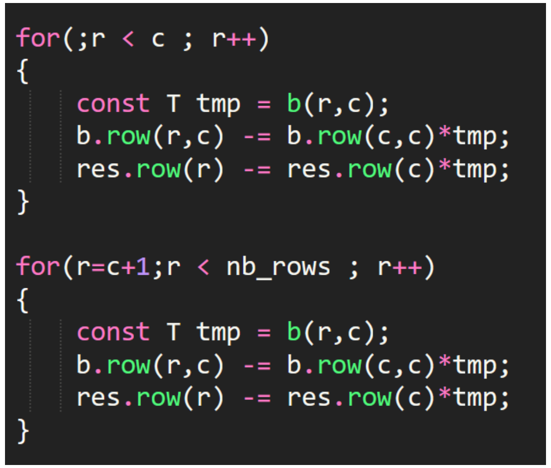
C Implementation (like in CMSIS-DSP)



```
#define MAS_ROW_F32(COL,A,i,v,B,j)                     \
{                                                      \
  int cnt = ((A)->numCols)-(COL);                      \
  float32_t *dataA = (A)->pData;                       \
  float32_t *dataB = (B)->pData;                       \
  const int32_t _numCols = (A)->numCols;               \
  int32_t _w;                                          \
  f32x4_t vec=vdupq_n_f32(v);                          \
                                                       \
  for(_w=(COL);_w < _numCols; _w+=4)                       \
  {                                                    \
       f32x4_t tmpa,tmpb;                              \
       mve_pred16_t p0 = vctp32q(cnt);                 \
       tmpa = vldrwq_z_f32(&dataA[i*_numCols + _w],p0);\
       tmpb = vldrwq_z_f32(&dataB[j*_numCols + _w],p0);\
       tmpa = vfmsq_f32(tmpa,tmpb,vec);                \
       vstrwq_p(&dataA[i*_numCols + _w], tmpa, p0);      \
       cnt -= 4;                                       \
  }                                                    \
                                                       \
}
```

Total = 130 lines (without comments) - only for f32 Helium

C++ Implementation using templates



```
for(;r < c ; r++)
{
    const T tmp = b(r,c);
    b.row(r,c) -= b.row(c,c)*tmp;
    res.row(r) -= res.row(c)*tmp;
}

for(r=c+1;r < nb_rows ; r++)
{
    const T tmp = b(r,c);
    b.row(r,c) -= b.row(c,c)*tmp;
    res.row(r) -= res.row(c)*tmp;
}
```

Total = 70 lines - All datatypes and architectures

Matrix inverse f32 (cycles) Arm Compiler 6.21

dims	C++	C	C++ Improvement
4 x 4	776	1298	40.22%
8 x 8	4008	5285	24.16%
16 x 16	19314	24914	22.48%

Open-CMSIS-Pack Project

Request for feedback on these proposed enhancements

CMSIS-SVD Improvements

- SVD files describe the peripherals of one device
- Using conditions and includes device variants could be described
  - github.com/Open-CMSIS-Pack/svd-spec/issues/5
  - github.com/Open-CMSIS-Pack/svd-spec/issues/6
- This results in CMSIS-SVD 2.0
  - For backward compatibility a converter to CMSIS-SVD 1.0 format is required
- Aspects of 64-bit architectures should be considered in CMSIS-SVD 2.0
- **What other features are missing?**

Pack Download Authentication

- Packs can be behind an access protection, but authentication in CMSIS-Pack tools is missing
- Proposal is to work on authentication for pack access; this could be potentially used for commercial software.

Additional Standardize APIs?

- IoT Socket 2.0 multiple sockets and multicast
  - Supports Matter
- PWM standardization for motor control
  - Enable shields with motor control hardware

We are committed to CMSIS and requirements for ML …

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

- Open-CMSIS Technical Meeting every Tuesday, 15:00 GMT

- Virtual Workshop: Create Scalable Software Packs
  - Audience: software vendors
  - 11. June 15:00 – 17:00 GMT

- Feedback via github issues on the various projects
  - github.com/arm-software/cmsis_6 – project overview
  - github.com/ARM-software/CMSIS_6/issues – for CMSIS core components
  - github.com/ARM-software/AVH-MLOps/issues – for ML components
  - https://github.com/ARM-software/arm-2d – Graphics utilizing Helium

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

cmsis@arm.com

Questions?

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