Review meeting at embedded world 2020

Tuesday, 25 February 2020

© 2020 Arm Limited

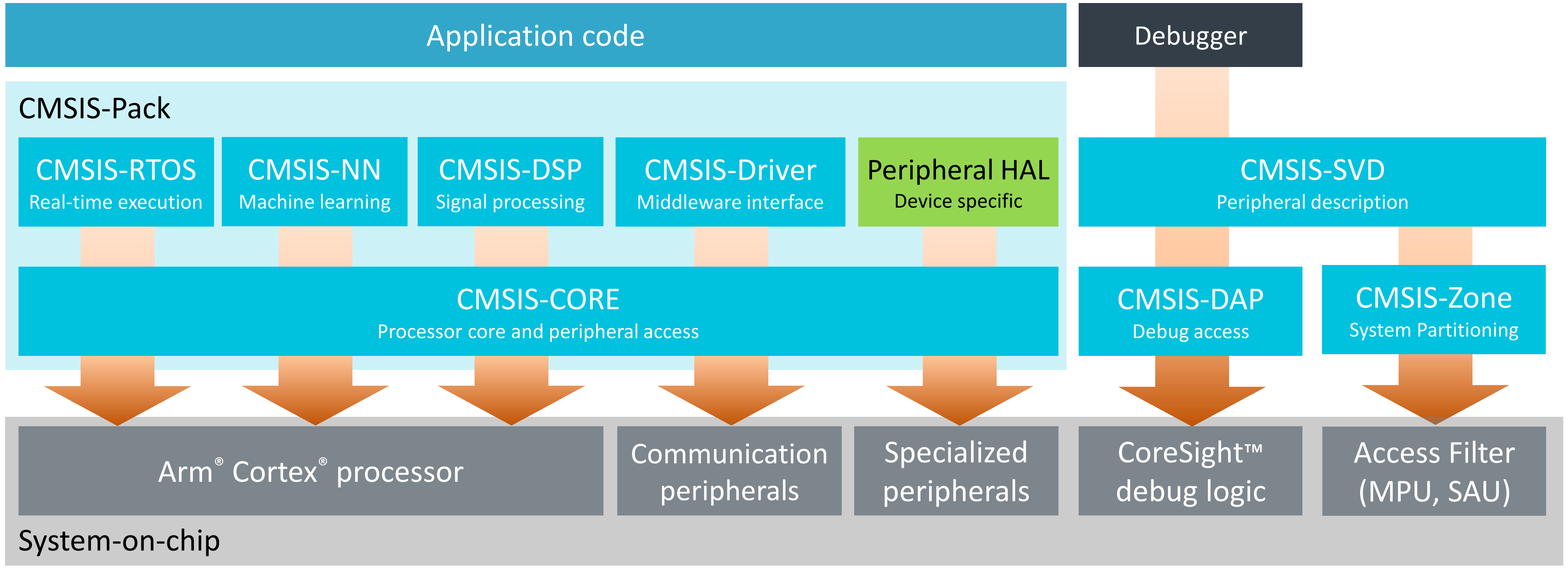
Agenda

- CMSIS Overview
- CMSIS-Zone for system configuration (Multi-core, TrustZone, MPU)
- New IP support for Cortex-M55 and improvements for DSP and Machine Learning
- An open approach for IoT on Cortex-M using software components
- CMSIS-Driver – WiFi and validation tests
- CMSIS and PSA / TF-M - Security Foundation for Cortex-M TrustZone
- CMSIS-Build - Productivity for complex software templates
- Summary, Actions, Roadmap

© 2020 Arm Limited

CMSIS 5

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



© 2020 Arm Limited

CMSIS-Zone

Configure multi-core, TrustZone and MPU

© 2020 Arm Limited

CMSIS-Zone: Device Hardware Configuration

Supports multi-processor systems, TrustZone, and MPU setup

Partition a multi-processor system into single processor views

Setup memory and peripherals for secure/non-secure environment

Generate consistent configuration

- Setup for the Armv8-M TrustZone (SAU, Interrupt assignment to Secure/Non-Secure)
- Setup of device specific Memory Protection Controller (MPC)
- Setup of device specific Peripheral Protection Controller (PPC)
- Generate related linker configuration to ensure consistency

Solves the alignment requirements for MPU on Armv7-M

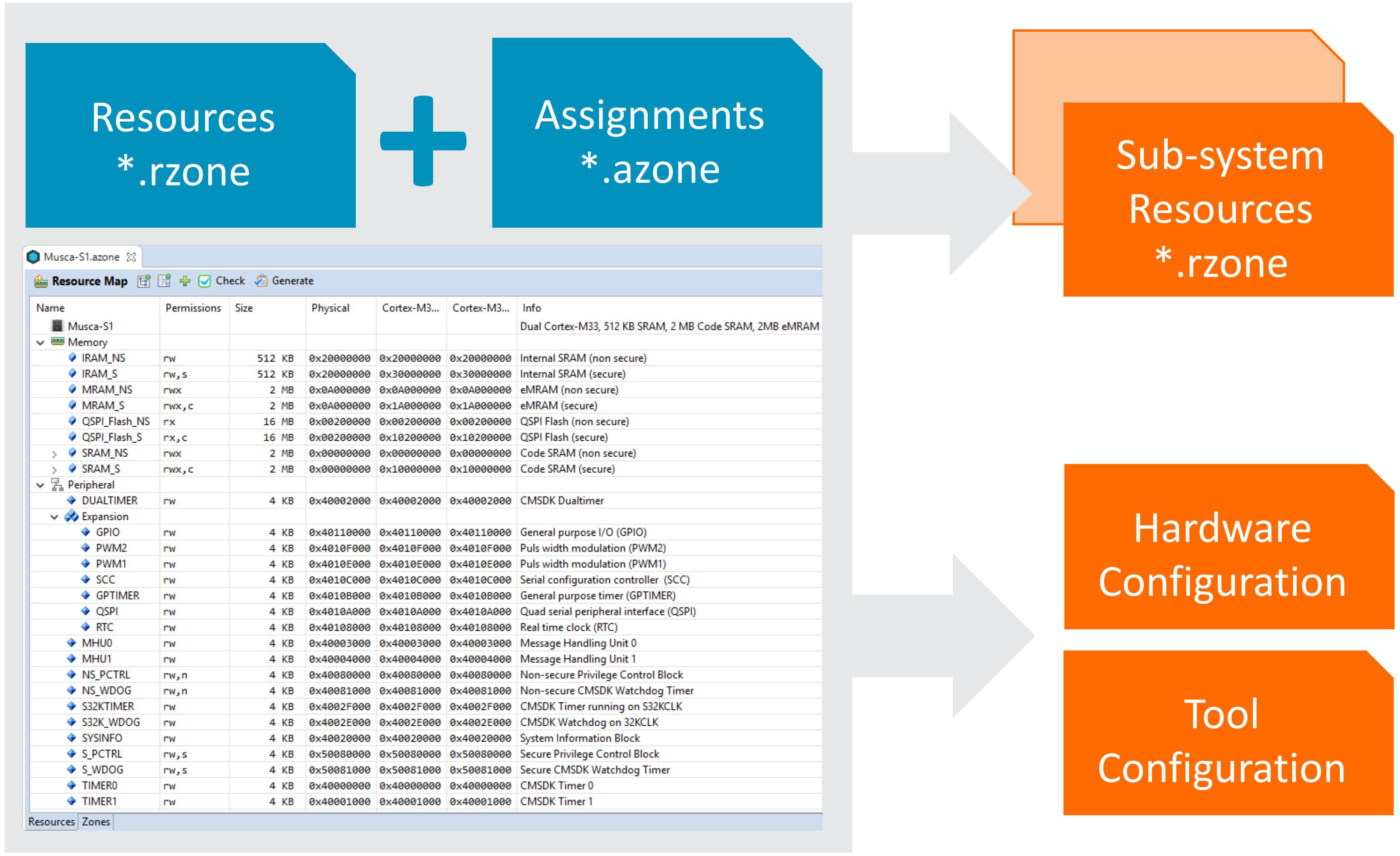
- MPU descriptors are optimized and located with alignment requirements

→ Learn more about device configuration with CMSIS-Zone

© 2020 Arm Limited

CMSIS-Zone – Development Workflow

Configuration and build management for system resources



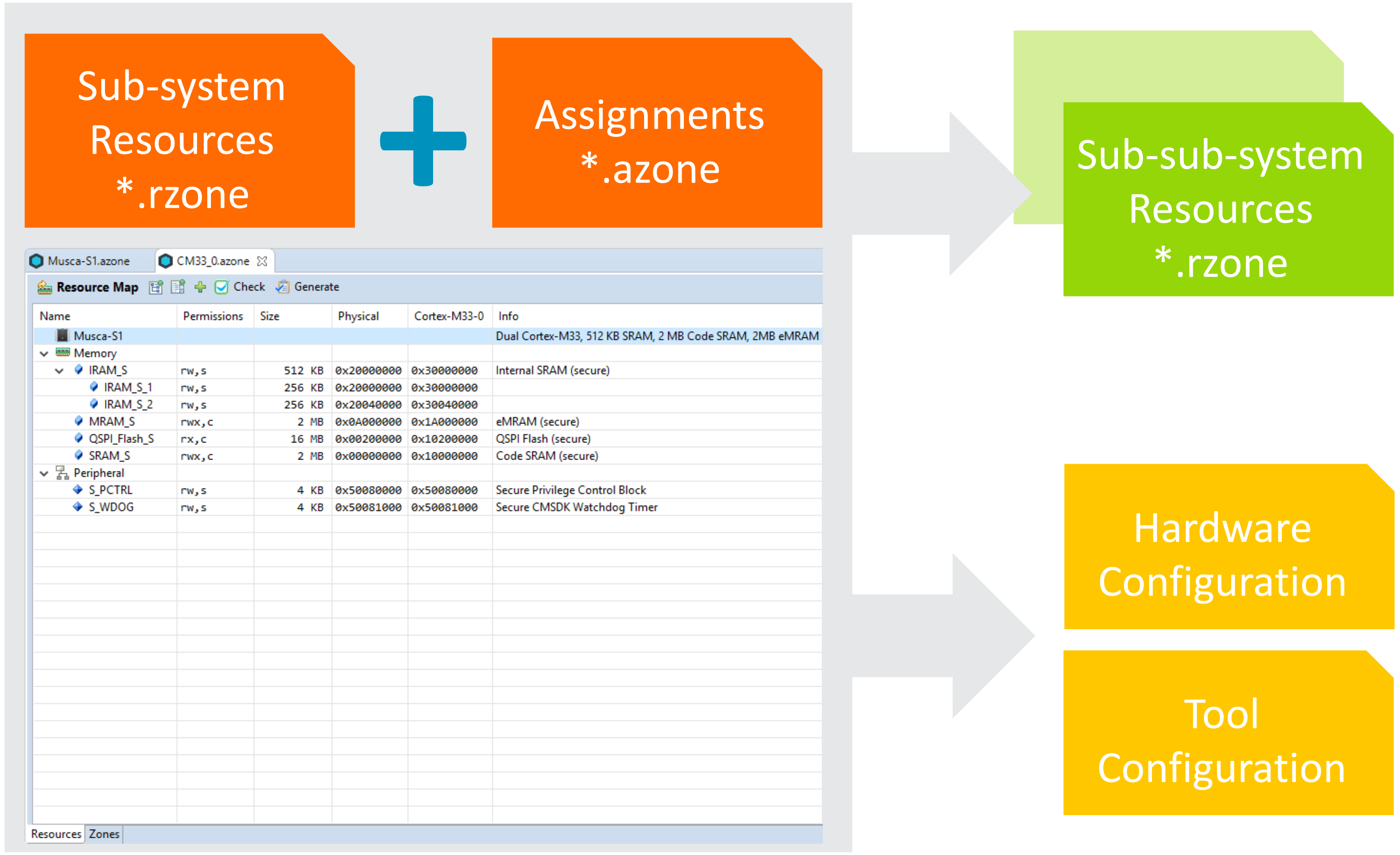
Name	Permissions	Size	Physical	Cortex-M3...	Cortex-M3...	Info
Musca-S1						Dual Cortex-M33, 512 KB SRAM, 2 MB Code SRAM, 2MB eMRAM
Memory						
IRAM_NS	rw	512 KB	0x20000000	0x20000000	0x20000000	Internal SRAM (non secure)
IRAM_S	rw,s	512 KB	0x20000000	0x30000000	0x30000000	Internal SRAM (secure)
MRAM_NS	rwx	2 MB	0x0A000000	0x0A000000	0x0A000000	eMRAM (non secure)
MRAM_S	rwx,c	2 MB	0x0A000000	0x1A000000	0x1A000000	eMRAM (secure)
QSPI_Flash_NS	rx	16 MB	0x00200000	0x00200000	0x00200000	QSPI Flash (non secure)
QSPI_Flash_S	rx,c	16 MB	0x00200000	0x10200000	0x10200000	QSPI Flash (secure)
SRAM_NS	rwx	2 MB	0x00000000	0x00000000	0x00000000	Code SRAM (non secure)
SRAM_S	rwx,c	2 MB	0x00000000	0x10000000	0x10000000	Code SRAM (secure)
Peripheral						
DUALTIMER	rw	4 KB	0x40002000	0x40002000	0x40002000	CMSDK Dualtimer
Expansion						
GPIO	rw	4 KB	0x40110000	0x40110000	0x40110000	General purpose I/O (GPIO)
PWM2	rw	4 KB	0x4010F000	0x4010F000	0x4010F000	Puls width modulation (PWM2)
PWM1	rw	4 KB	0x4010E000	0x4010E000	0x4010E000	Puls width modulation (PWM1)
SCC	rw	4 KB	0x4010C000	0x4010C000	0x4010C000	Serial configuration controller (SCC)
GPTIMER	rw	4 KB	0x4010B000	0x4010B000	0x4010B000	General purpose timer (GPTIMER)
QSPI	rw	4 KB	0x4010A000	0x4010A000	0x4010A000	Quad serial peripheral interface (QSPI)
RTC	rw	4 KB	0x40108000	0x40108000	0x40108000	Real time clock (RTC)
MHU0	rw	4 KB	0x40003000	0x40003000	0x40003000	Message Handling Unit 0
MHU1	rw	4 KB	0x40004000	0x40004000	0x40004000	Message Handling Unit 1
NS_PCTRL	rw,n	4 KB	0x40080000	0x40080000	0x40080000	Non-secure Privilege Control Block
NS_WDOG	rw,n	4 KB	0x40081000	0x40081000	0x40081000	Non-secure CMSDK Watchdog Timer
S32KTIMER	rw	4 KB	0x4002F000	0x4002F000	0x4002F000	CMSDK Timer running on S32KCLK
S32K_WDOG	rw	4 KB	0x4002E000	0x4002E000	0x4002E000	CMSDK Watchdog on 32KCLK
SYSINFO	rw	4 KB	0x40020000	0x40020000	0x40020000	System Information Block
S_PCTRL	rw,s	4 KB	0x50080000	0x50080000	0x50080000	Secure Privilege Control Block
S_WDOG	rw,s	4 KB	0x50081000	0x50081000	0x50081000	Secure CMSDK Watchdog Timer
TIMER0	rw	4 KB	0x40000000	0x40000000	0x40000000	CMSDK Timer 0
TIMER1	rw	4 KB	0x40001000	0x40001000	0x40001000	CMSDK Timer 1

- Resource *.rzone file lists all available systems resources.
- Assignment *.azone file contains partitioning information and is managed by CMSIS-Zone tool
- CMSIS-Zone tool generates sub-system resource files

© 2020 Arm Limited

CMSIS-Zone – Development Workflow

Multi-step approach shows only relevant sub-system



Name	Permissions	Size	Physical	Cortex-M33-0	Info
Musca-S1					Dual Cortex-M33, 512 KB SRAM, 2 MB Code SRAM, 2MB eMRAM
Memory					
IRAM_S	rw,s	512 KB	0x20000000	0x30000000	Internal SRAM (secure)
IRAM_S_1	rw,s	256 KB	0x20000000	0x30000000	
IRAM_S_2	rw,s	256 KB	0x20040000	0x30040000	
MRAM_S	rwx,c	2 MB	0x0A000000	0x1A000000	eMRAM (secure)
QSPI_Flash_S	rx,c	16 MB	0x00200000	0x10200000	QSPI Flash (secure)
SRAM_S	rwx,c	2 MB	0x00000000	0x10000000	Code SRAM (secure)
Peripheral					
S_PCTRL	rw,s	4 KB	0x50080000	0x50080000	Secure Privilege Control Block
S_WDOG	rw,s	4 KB	0x50081000	0x50081000	Secure CMSDK Watchdog Timer

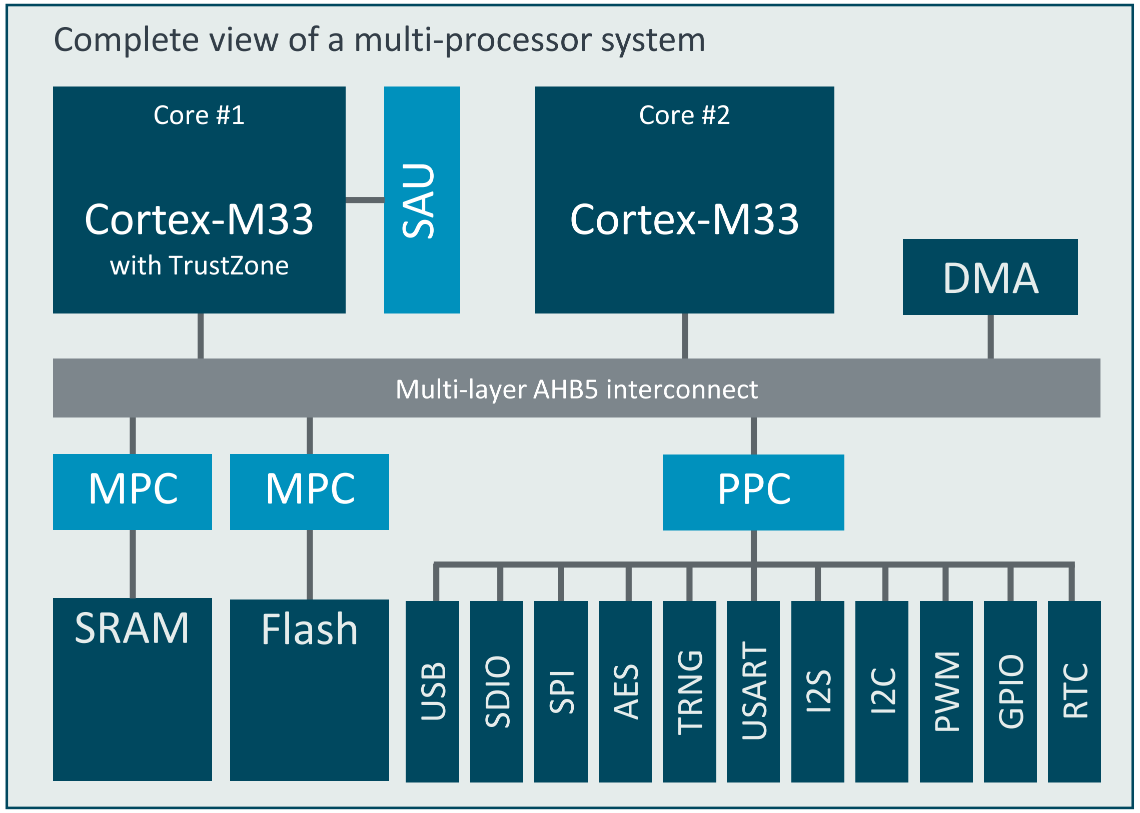
- It is possible to break down complexity of a system in multiple steps.
- Sub-systems expose only the part of the system that is relevant for the user.
- A sub-system user has no visibility to other parts of the system (as it typically configures also the related access protection).

© 2020 Arm Limited

Configuration Steps

Example

Step 1: split the multi-processor system into single processor sub-system

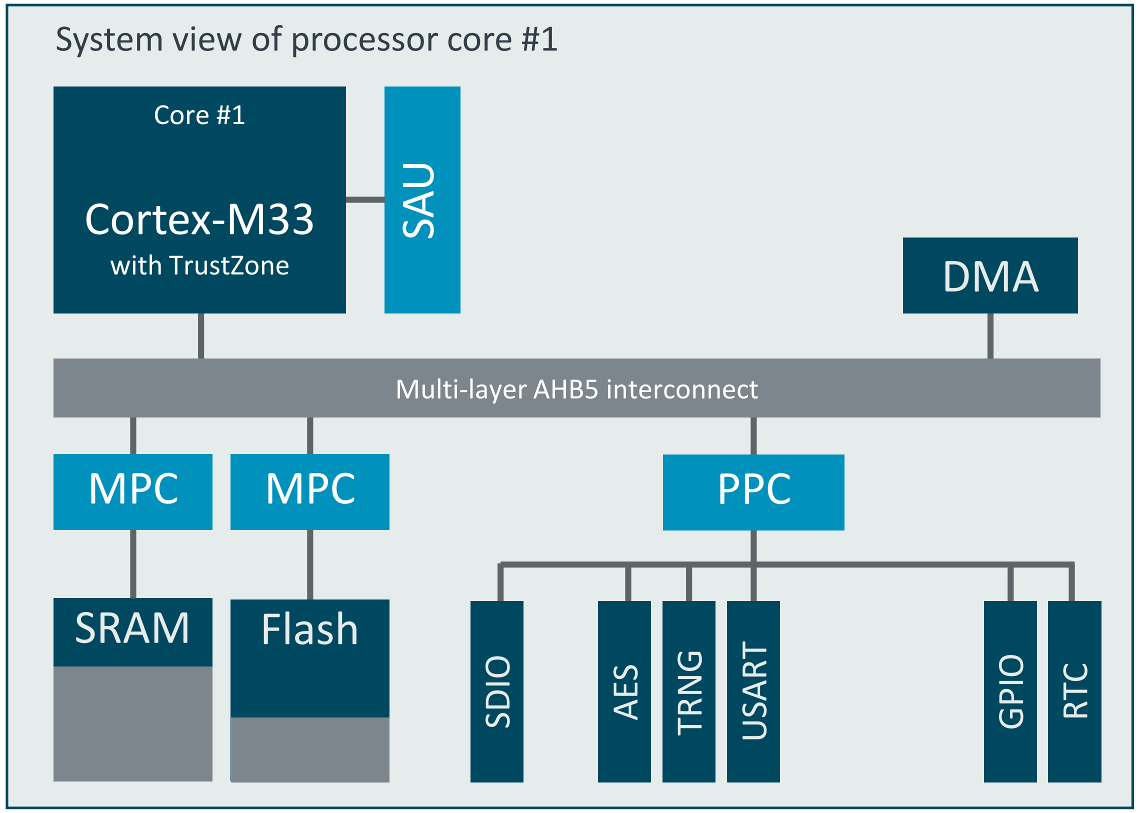


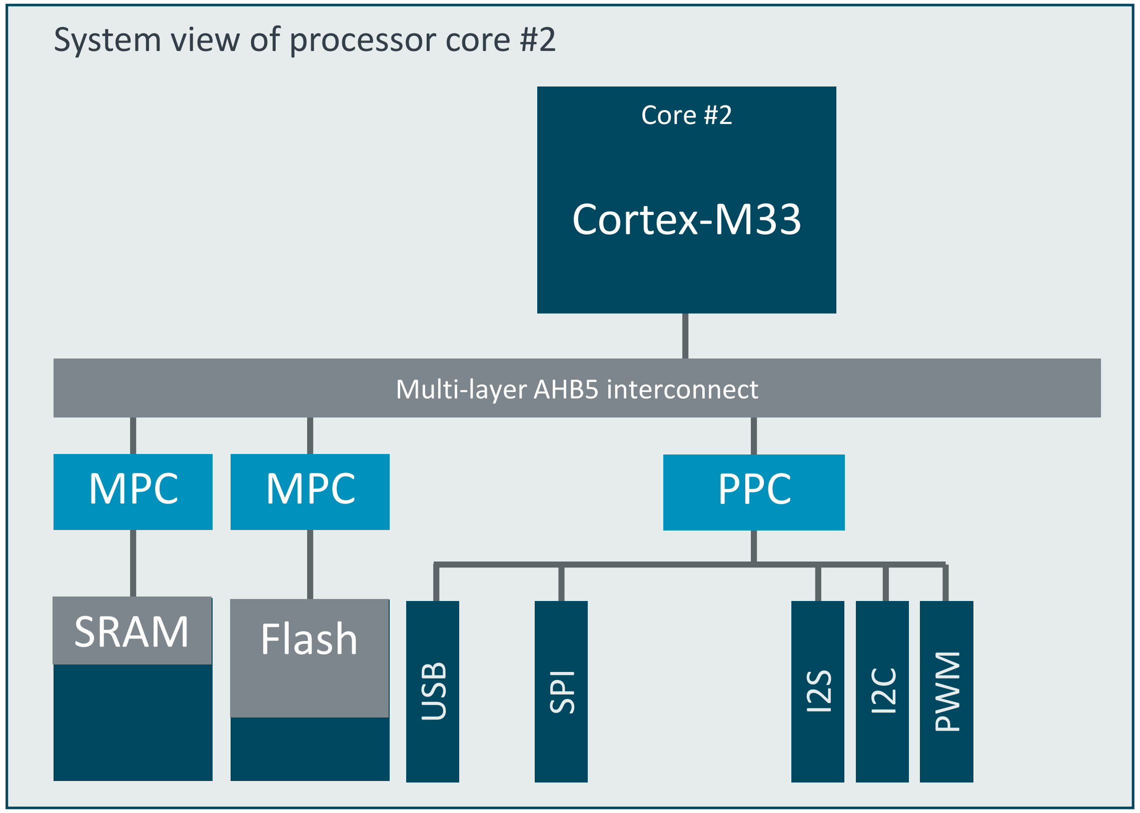
© 2020 Arm Limited

Configuration Steps

Example

Step 1: split the multi-processor system into single processor sub-systems




System view of processor core #2
Core #2
Cortex-M33
Multi-layer AHB5 interconnect
MPC
MPC
PPC
SRAM
Flash
USB
SPI
I2S
I2C
PWM

Step 2: create the partitions for secure and non-secure execution

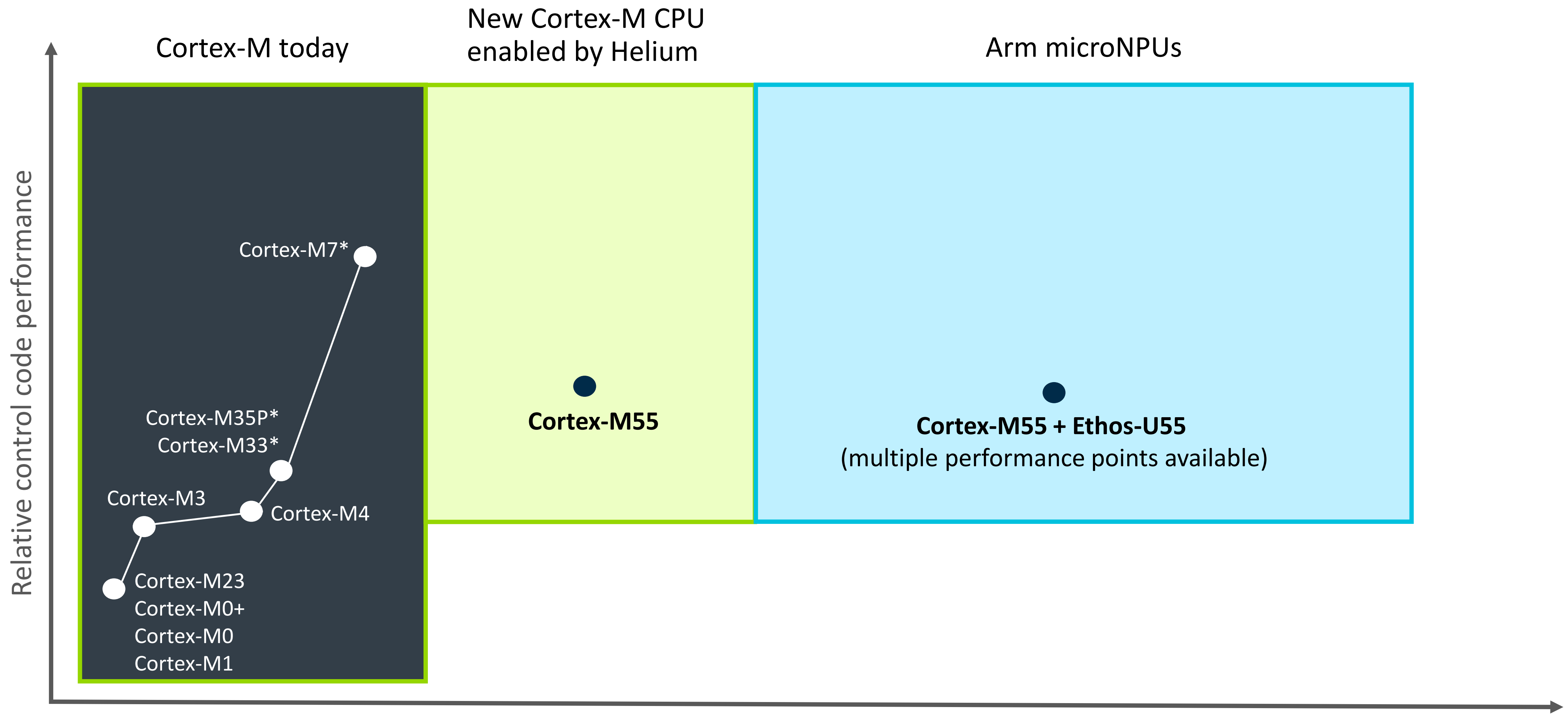
© 2020 Arm Limited

New Arm IP Support Cortex-M55

Software Support for latest Arm IP

© 2020 Arm Limited

Cortex-M processor portfolio covered by CMSIS



Relative ML and signal processing performance

*Existing processors with DSP extensions

**Based on Arm data

Graph not to scale

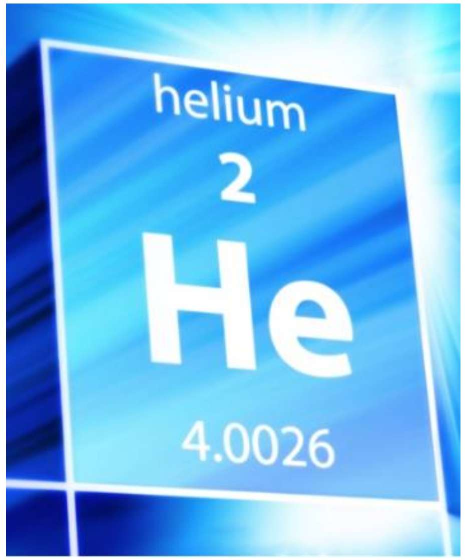
© 2020 Arm Limited

Key Algorithms for Signal Processing and ML

Armv8.1-M and Helium technology

The key algorithms for signal processing and ML are:

- **Complex dot-product** (convolution, correlators).
- **Fast Fourier Transform** (feature extraction, beamformer)
- **Neural-Networks** (matrix multiplications with bytes)
- **Biquad filter** (equalizers, noise filtering)



Armv8.1-M architecture is optimized for those algorithms.
With **performance boost** from 3 to 17 compared to M4, from 2 to 8 compared to M7.

CMSIS-DSP is fully ported to SIMD for Cortex-M family (Armv8.1-M) and Cortex-A with NEON, using the same APIs.

© 2020 Arm Limited

Plans

Preserve your R&D investments

CMSIS DSP/ML kernels :

- Grow the number of DSP/ML kernels, with **software portability** in mind: preserve your software R&D investments along any processor of the Arm portfolio
- Add important DSP kernels (fp64, fp16, math, 2D, logical, sorting, Kalman, interpolation)
- Invite a wider range of developers to contribute thanks to a **data-flow framework**
- **Classical-ML kernels** for constrained embedded markets :

  SVM, Tree, PCA, mean/variance gaussian normalization, clustering by K-Means / LBG ..

© 2020 Arm Limited

An open approach for IoT on Cortex-M

Create IoT Applications with ready-to-use software components

© 2020 Arm Limited

IoT devices are different – how can we deploy IoT software stacks efficiently at scale?

1 Trillion devices by 2035 =

~10 years

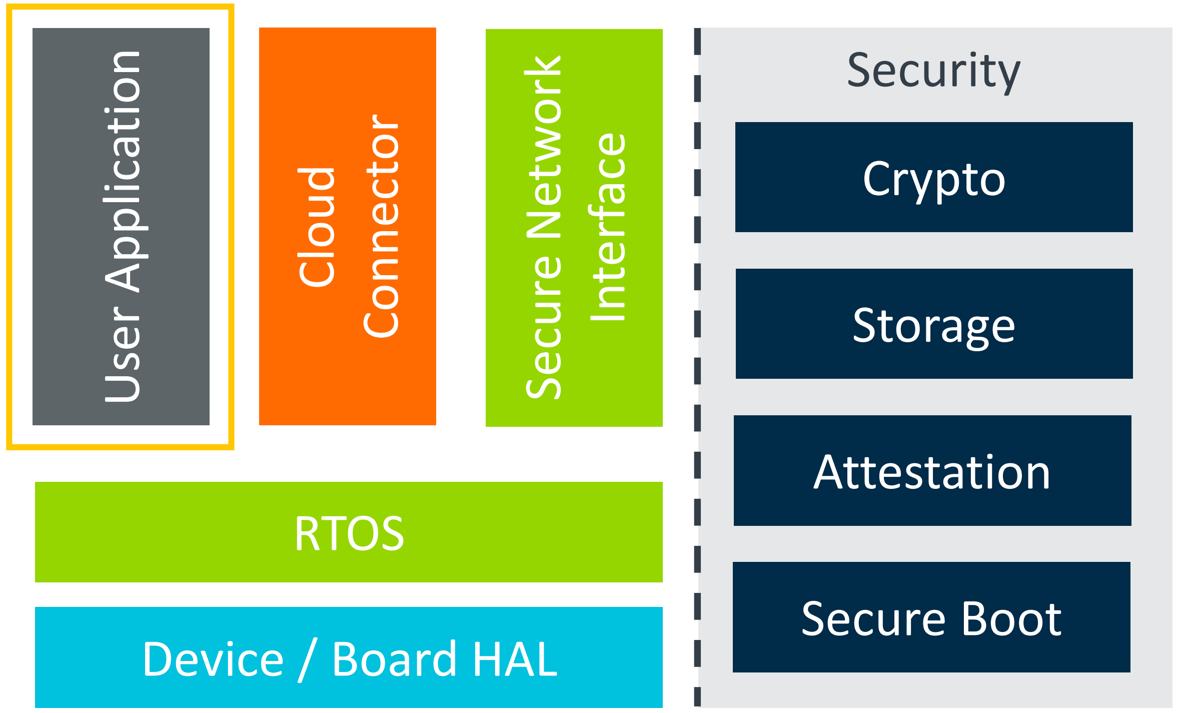
10 million units/year

10.000 bespoke designs

© 2020 Arm Limited

An open approach for IoT on Cortex-M

Simplified view to the software building blocks for IoT endpoints



- **Device / Board HAL:** abstraction of processor and peripherals with hardware specific configuration
- **RTOS**: thread and resource management
- **Secure Network Interface**: encrypted internet connection using different interfaces (Ethernet, WiFi, …)
- **Cloud Connector:** protocol interface to cloud provider
- **User Application:** bespoke functionality of endpoints

Security running on Secure Processing Environment:

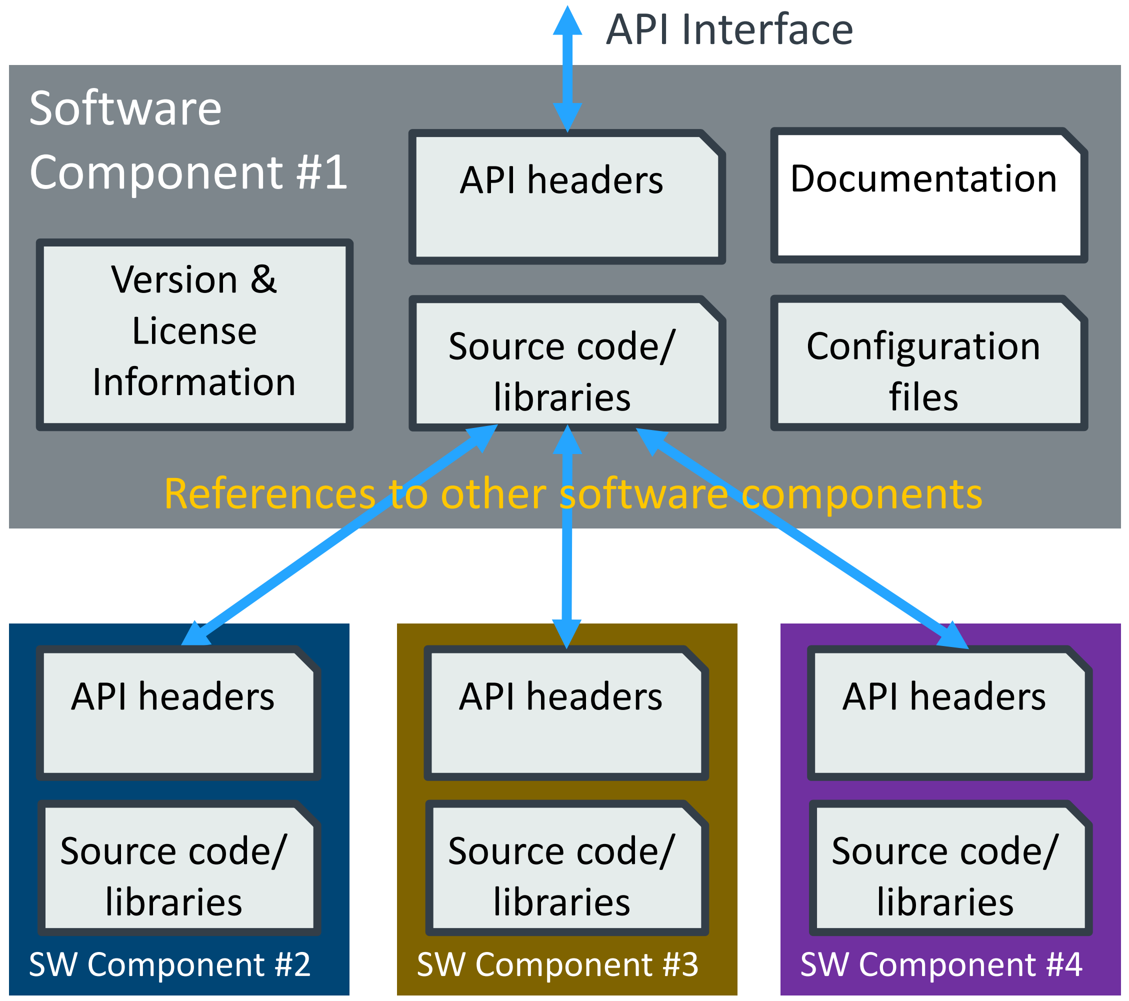
- Crypto services and device identity

Combining various software building blocks effectively is enabled by the CMSIS-Pack system.

© 2020 Arm Limited

CMSIS-Pack: What is a software component?

XML framed information used by project management utilities from various tools



Software components should have:

- Version and history information
- License information
- API interface definition
- Documentation
- Source files
- Configuration files (optional)
- Requirements to other components (optional)

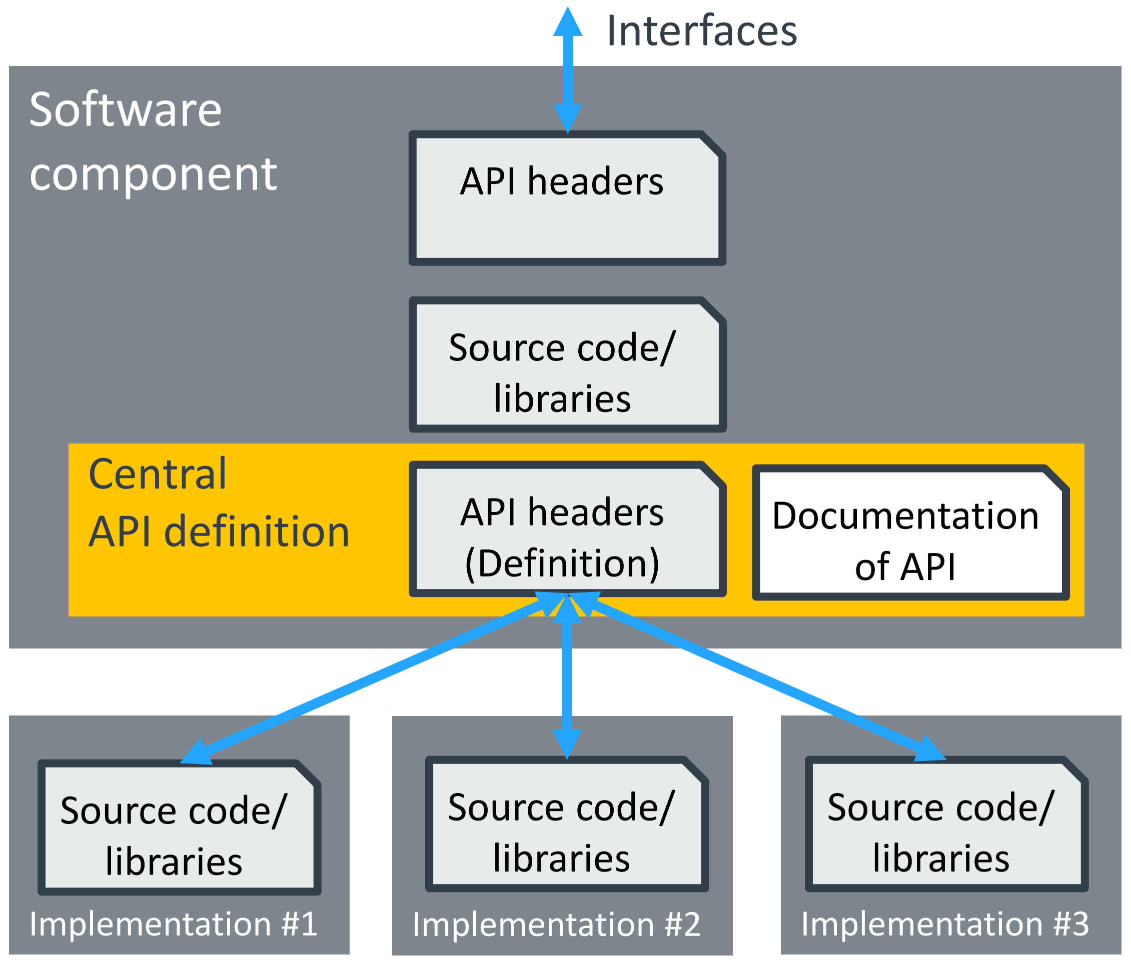
CMSIS-Pack framed software is supported by:

- Mainstream IDEs: Arm DS, Keil MDK, IAR EWARM
- Silicon vendor tools: ADI, OnSemi, STCubeMX
- Several web portals
- Open-source and command-line build tools

© 2020 Arm Limited

CMSIS-Pack: Central API Interface definition

Ensuring consistent interfaces across standard components



A common problem: API headers evolve over time.

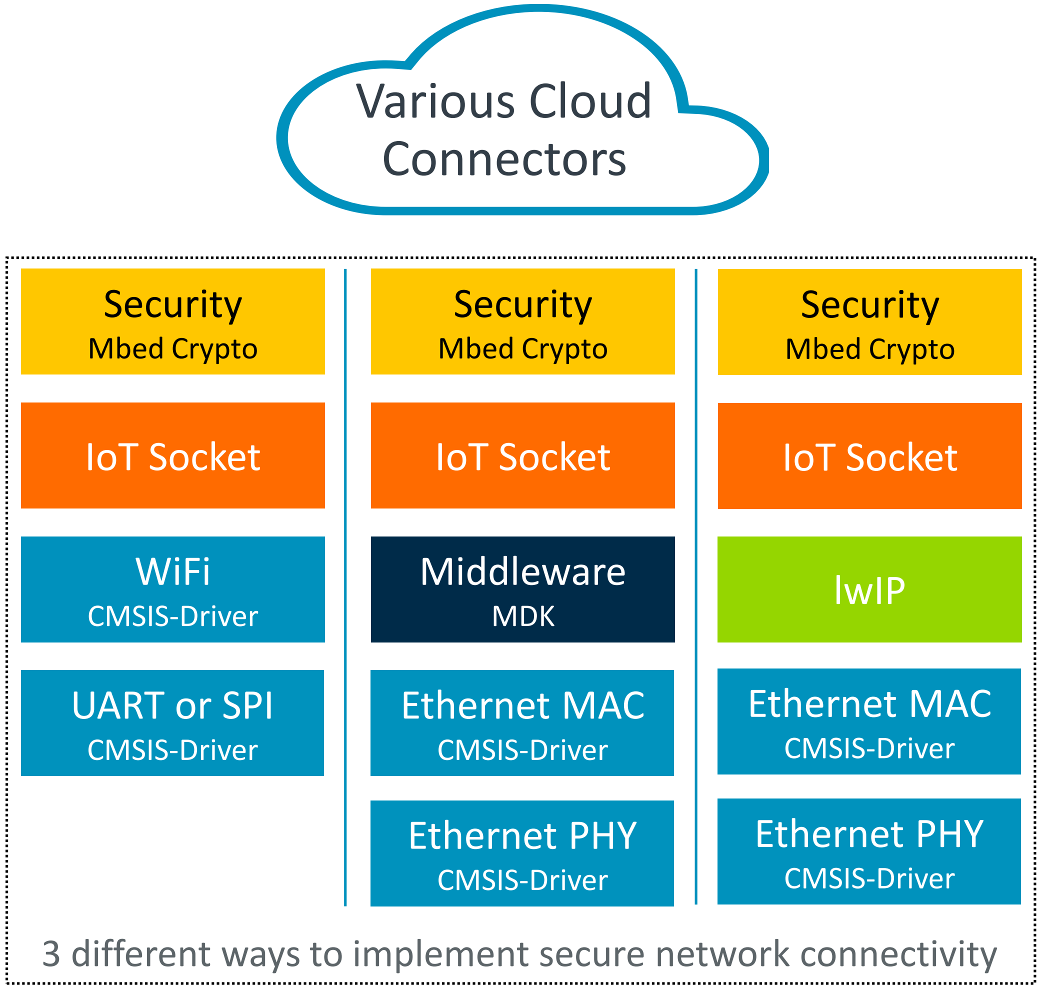
- A central API definition shares header file and documentation of an API interface across multiple other software components to ensure consistency.
- The API interface is distributed separate or as part of the software component that defines this interface. The API header file is therefore consistent.
- An example is the CMSIS-Driver pack that contains various Flash, Ethernet and WiFi drivers – all compatible with the CMSIS-Driver APIs that are published in the CMSIS Pack.

→ Learn how to create scalable software

© 2020 Arm Limited

Secure Network Interface – implementation choices

IoT devices need flexibility for implementing connectivity on Cortex-M



Available Cloud Connectors:

Azure

Google Cloud Platform

arm PELION

IBM

aws

Mbed Crypto implements security

IoT Socket connects to networks with:

- CMSIS-Driver for WiFi implemented with various chipsets
- MDK-Middleware IP networking stack (wired or WiFi)
- LwIP open-source IP communication with wired Ethernet

© 2020 Arm Limited

CMSIS-Driver

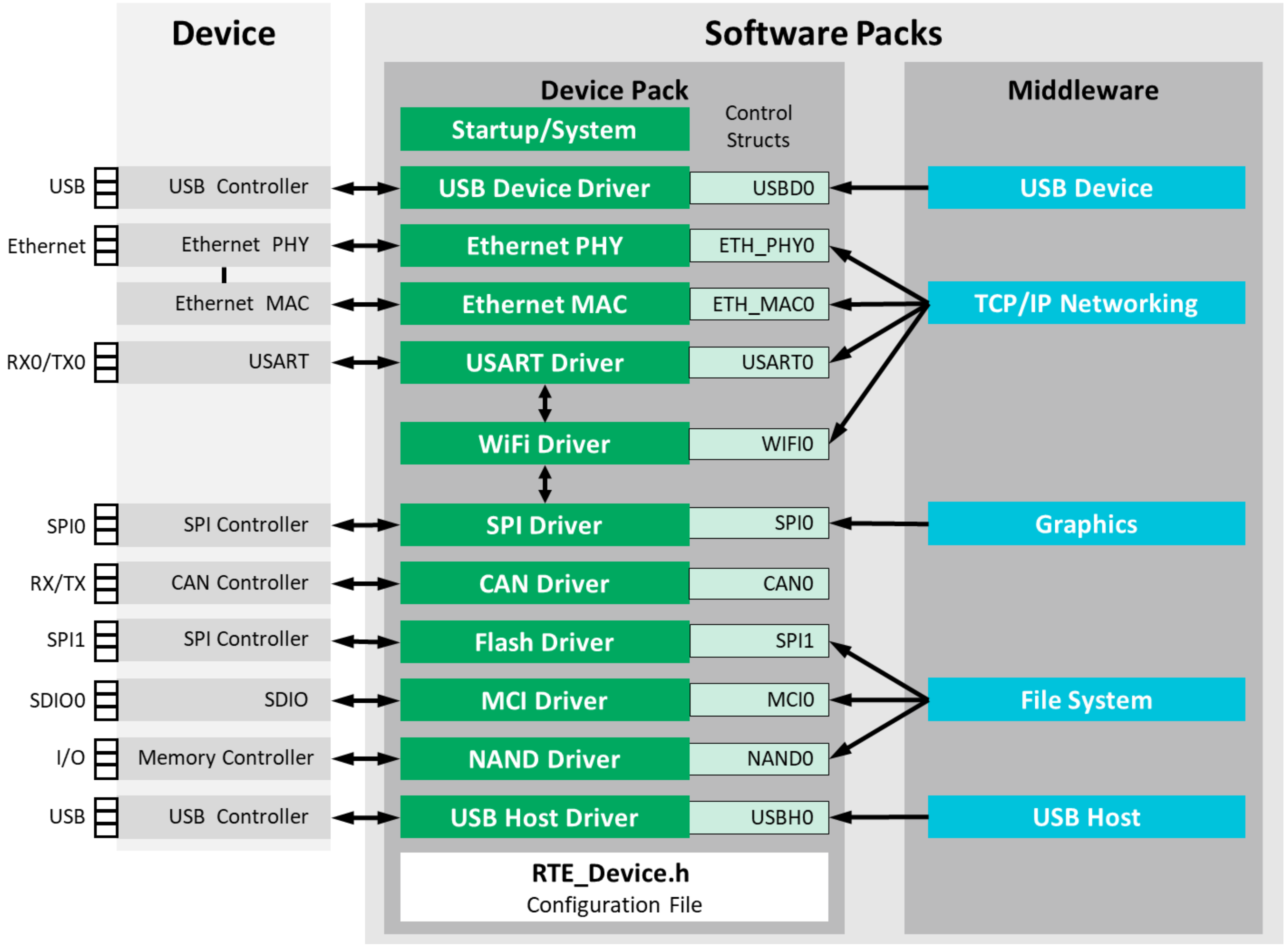
Generic Peripheral Interfaces

© 2020 Arm Limited

CMSIS-Driver

Drivers provide the software interface to device peripherals and CMSIS-Driver define a consistent API.

This allows re-use of various software components across different devices and makes software portable.

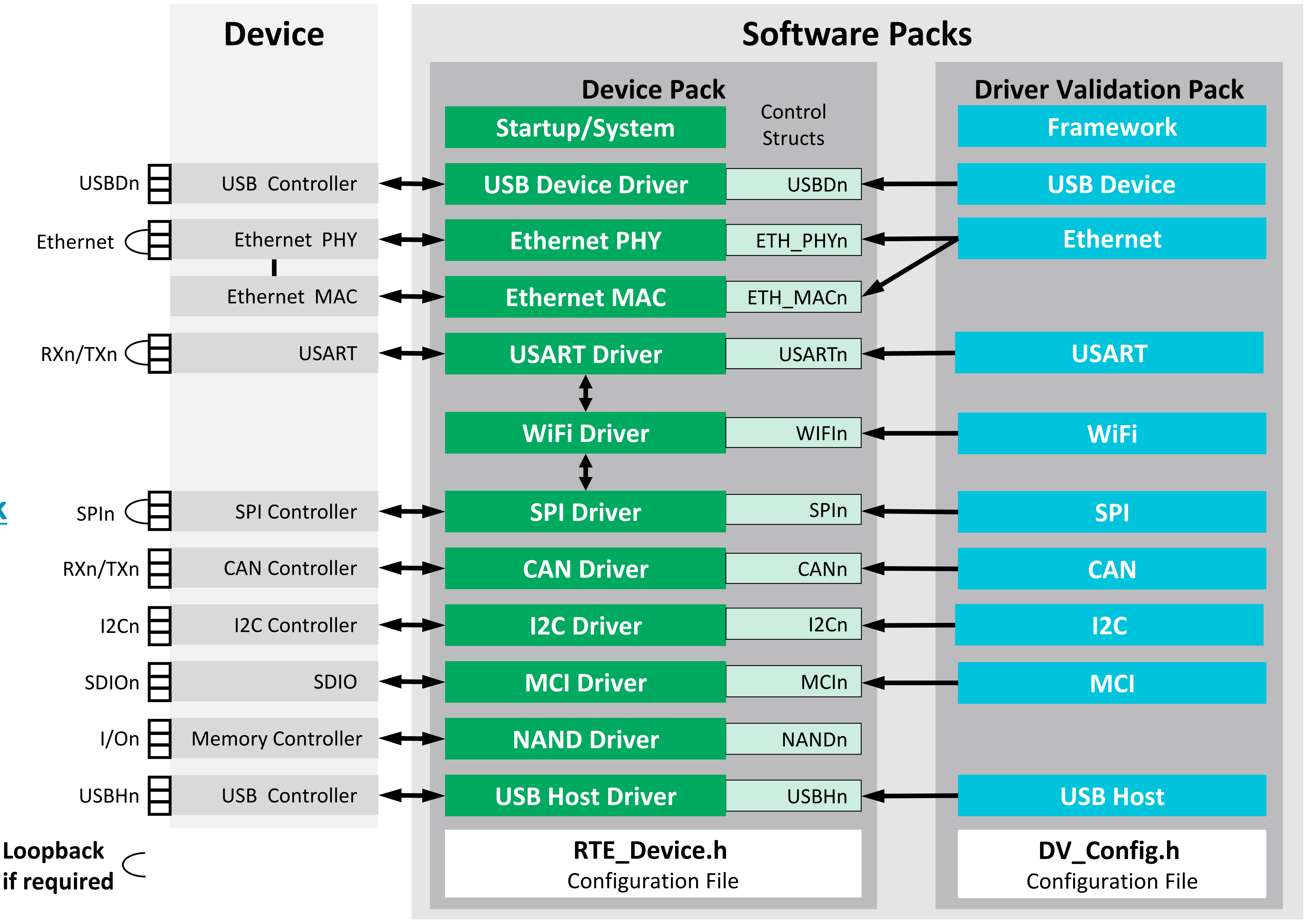


© 2020 Arm Limited

Validation

Drivers can be configured, for example to use specific I/O pins or features like DMA.

To verify the driver setup, CMSIS offers a **Driver Validation Pack** that executes in your hardware. This ensures proper operation before running complex middleware on top.



© 2020 Arm Limited

CMSIS-Driver + Validation

- Documentation:
  - Driver: https://arm-software.github.io/CMSIS_5/Driver/html/index.html
  - Validation: https://arm-software.github.io/CMSIS_5/Driver/html/driverValidation.html

- Packs (https://developer.arm.com/embedded/cmsis/cmsis-packs)
  - Driver: ARM:CMSIS Drivers for external devices (various WiFi, Ethernet PHY, etc.)
  - Driver: MDK-Packs:CMSIS WiFi Driver for Qualcomm QCA4002/4 based WiFi module
  - Driver validation: ARM:CMSIS-Driver_Validation

→ Learn about WiFi drivers and WiFi validation

© 2020 Arm Limited

CMSIS and PSA / TF-M

Security Foundation for Cortex-M TrustZone

© 2020 Arm Limited

Platform Security Architecture

A complete security offering – openly published. Independently tested.

Analyze

Threat models
& security analyses

Methodically
developed

Architect

Hardware & firmware
architect specifications

Open
architecture

Implement

Firmware
source code

TF-M Reference
implementation

Certify

Independently
tested

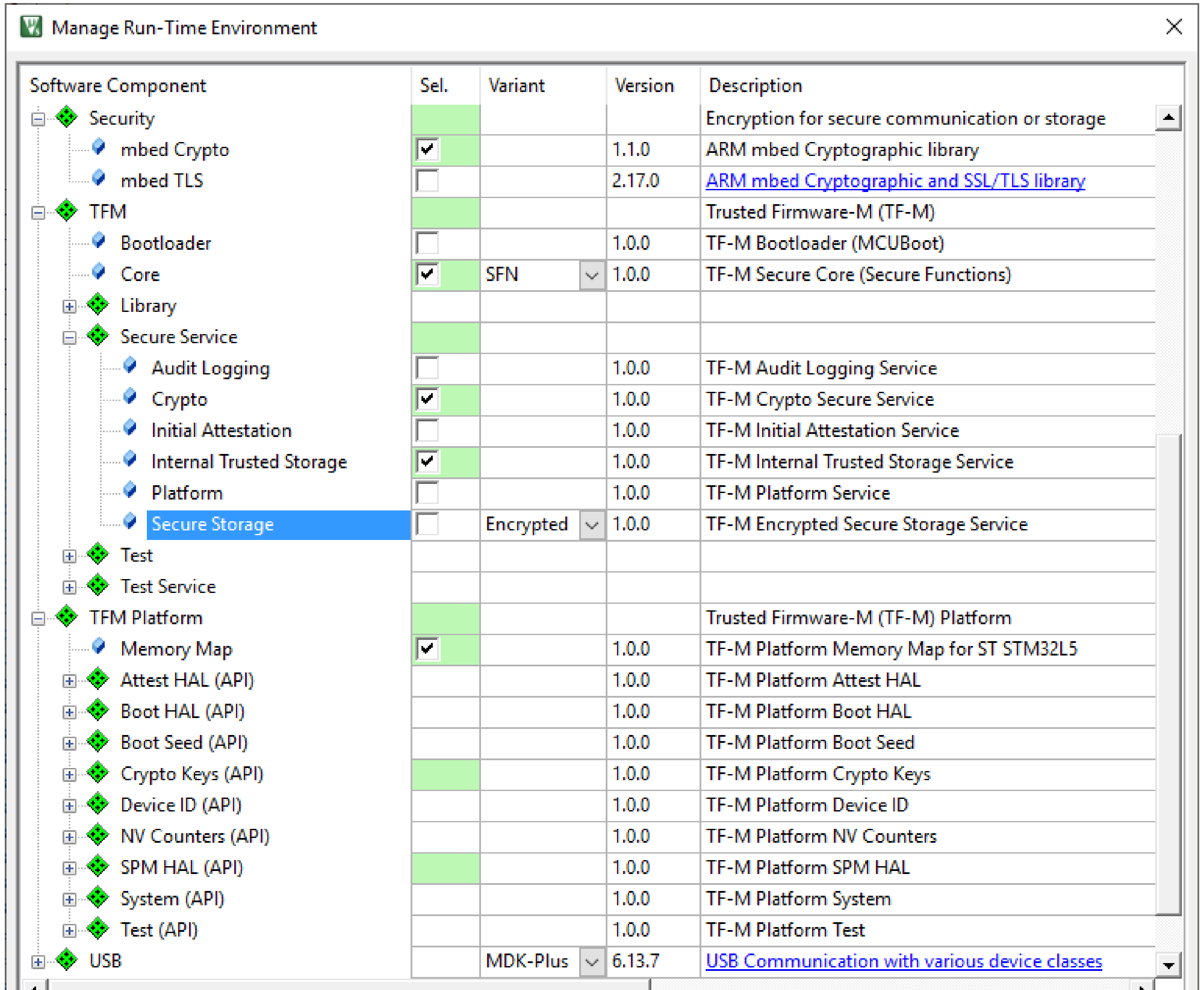
psacertified™

Enabling
trust

psacertified™

TF-M framed as CMSIS-Pack – focus on Armv8-M

Published at http://github.com/arm-software/CMSIS-TFM



Manage Run-Time Environment

Software Component	Sel.	Variant	Version	Description
Security				Encryption for secure communication or storage
mbed Crypto	✓		1.1.0	ARM mbed Cryptographic library
mbed TLS	☐		2.17.0	ARM mbed Cryptographic and SSL/TLS library
TFM				Trusted Firmware-M (TF-M)
Bootloader	☐		1.0.0	TF-M Bootloader (MCUBoot)
Core	✓	SFN	1.0.0	TF-M Secure Core (Secure Functions)
Library				
Secure Service				
Audit Logging	☐		1.0.0	TF-M Audit Logging Service
Crypto	✓		1.0.0	TF-M Crypto Secure Service
Initial Attestation	☐		1.0.0	TF-M Initial Attestation Service
Internal Trusted Storage	✓		1.0.0	TF-M Internal Trusted Storage Service
Platform	☐		1.0.0	TF-M Platform Service
Secure Storage	☐	Encrypted	1.0.0	TF-M Encrypted Secure Storage Service
Test				
Test Service				
TFM Platform				Trusted Firmware-M (TF-M) Platform
Memory Map	✓		1.0.0	TF-M Platform Memory Map for ST STM32L5
Attest HAL (API)			1.0.0	TF-M Platform Attest HAL
Boot HAL (API)			1.0.0	TF-M Platform Boot HAL
Boot Seed (API)			1.0.0	TF-M Platform Boot Seed
Crypto Keys (API)			1.0.0	TF-M Platform Crypto Keys
Device ID (API)			1.0.0	TF-M Platform Device ID
NV Counters (API)			1.0.0	TF-M Platform NV Counters
SPM HAL (API)			1.0.0	TF-M Platform SPM HAL
System (API)			1.0.0	TF-M Platform System
Test (API)			1.0.0	TF-M Platform Test
USB		MDK-Plus	6.13.7	USB Communication with various device classes

- Beta availability: 16. March 2020
- Pack Structure:

  TFM – Trusted Firmware-M reference implementation (contains both: secure + non-secure parts)

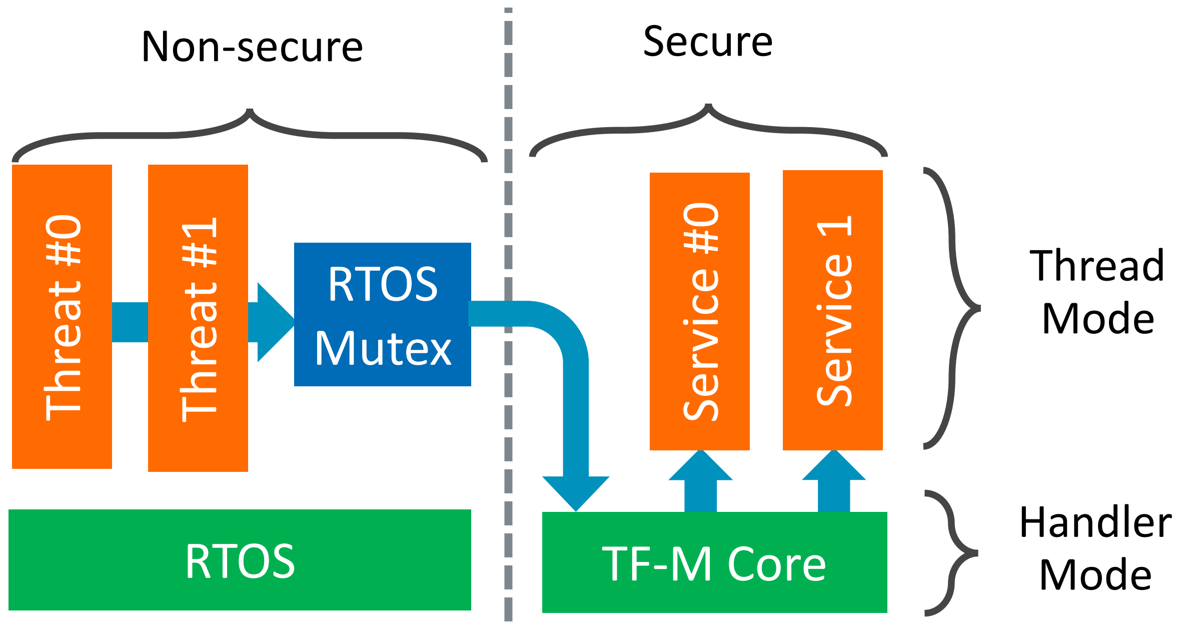
  TFM-Platform – device specific support
  - TFM_Platform_LPC55S6x
  - TFM_Platform_STM32L5
  - .....

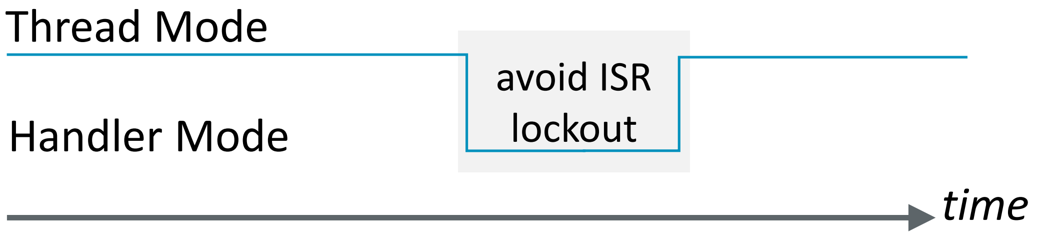
TFM Pack is synchronized with https://trustedfirmware.org/

- Changes will be upstreamed

TFM CMSIS Pack – optimize for efficiency on Armv8-M

Working with **trustedfirmware.org** team on overall simplification



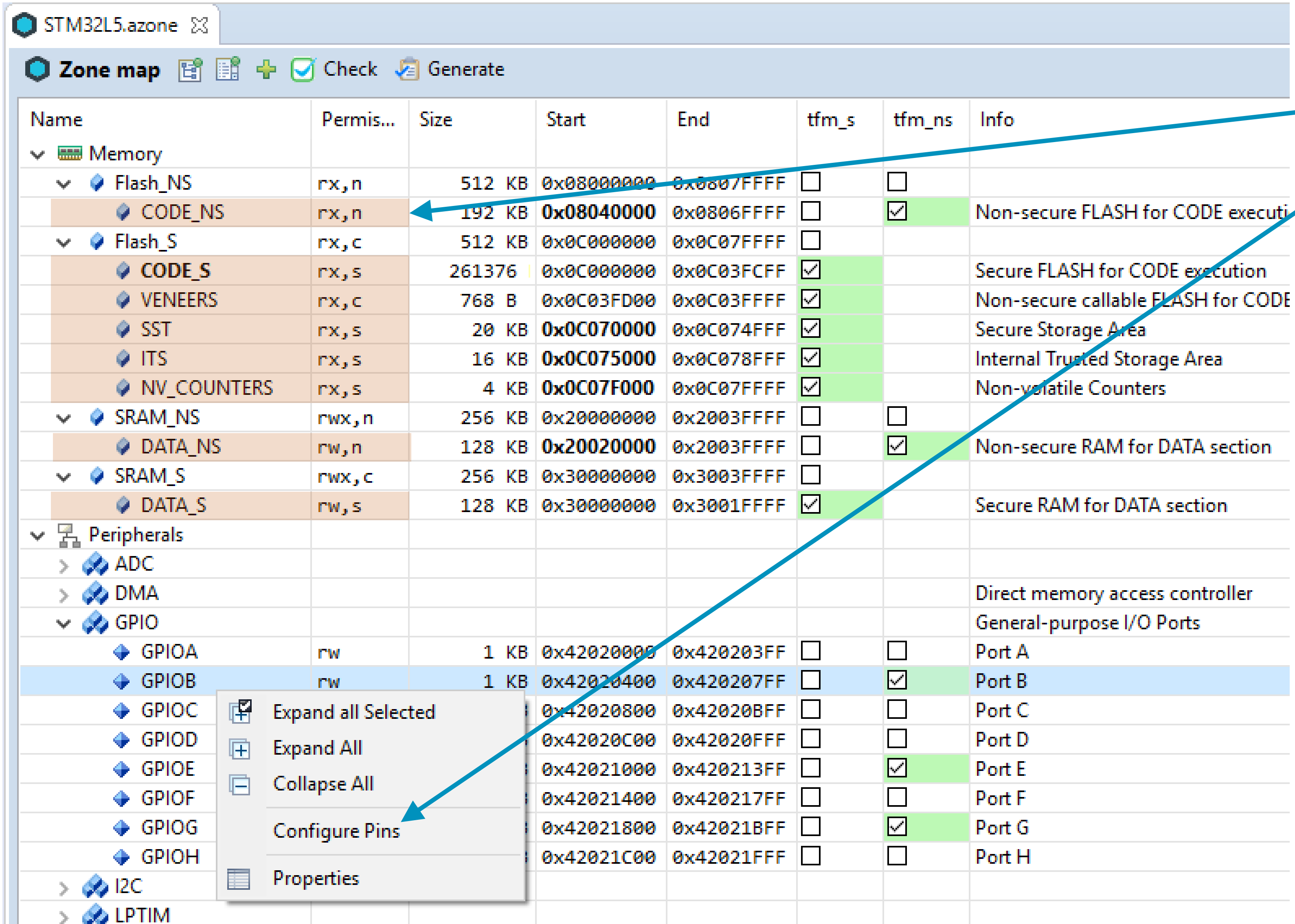


- No special requirements on RTOS (no TZ context management) ☑
- Prefer static configuration vs. dynamic run-time configuration ☑
- Secure Function (SFN) calls instead of IPC ☑
- Simplify MPU handling for isolation level 2 / 3
- Reduce overall memory footprint
- Support for device-specific boot loaders
- Event annotations instead of printf diagnostics
- Interrupt behavior under review (avoid ISR lock-out by carefully choosing SVC handler mode priority)

© 2020 Arm Limited

TFM CMSIS Pack – Security setup with CMSIS-Zone

Define TF-M memory regions and peripheral access rights in three steps:



Name	Permis...	Size	Start	End	tfm_s	tfm_ns	Info
Memory							
Flash_NS	rx,n	512 KB	0x08000000	0x0807FFFF	☐	☐	
CODE_NS	rx,n	192 KB	0x08040000	0x0806FFFF	☐	☑	Non-secure FLASH for CODE executi
Flash_S	rx,c	512 KB	0x0C000000	0x0C07FFFF	☐		
CODE_S	rx,s	261376	0x0C000000	0x0C03FCFF	☑		Secure FLASH for CODE execution
VENEERS	rx,c	768 B	0x0C03FD00	0x0C03FFFF	☑		Non-secure callable FLASH for CODE
SST	rx,s	20 KB	0x0C070000	0x0C074FFF	☑		Secure Storage Area
ITS	rx,s	16 KB	0x0C075000	0x0C078FFF	☑		Internal Trusted Storage Area
NV_COUNTERS	rx,s	4 KB	0x0C07F000	0x0C07FFFF	☑		Non-volatile Counters
SRAM_NS	rwx,n	256 KB	0x20000000	0x2003FFFF	☐	☐	
DATA_NS	rw,n	128 KB	0x20020000	0x2003FFFF	☐	☑	Non-secure RAM for DATA section
SRAM_S	rwx,c	256 KB	0x30000000	0x3003FFFF	☐		
DATA_S	rw,s	128 KB	0x30000000	0x3001FFFF	☑		Secure RAM for DATA section
Peripherals							
ADC							
DMA							Direct memory access controller
GPIO							General-purpose I/O Ports
GPIOA	rw	1 KB	0x42020000	0x420203FF	☐	☐	Port A
GPIOB	rw	1 KB	0x42020400	0x420207FF	☐	☑	Port B
GPIOC			0x42020800	0x42020BFF	☐	☐	Port C
GPIOD			0x42020C00	0x42020FFF	☐	☐	Port D
GPIOE			0x42021000	0x420213FF	☐	☑	Port E
GPIOF			0x42021400	0x420217FF	☐	☐	Port F
GPIOG			0x42021800	0x42021BFF	☐	☑	Port G
GPIOH			0x42021C00	0x42021FFF	☐	☐	Port H
I2C							
LPTIM							

1. Define in CMSIS-Zone:

- Memory regions for TFM
- Peripheral access rights

2. Select resource usage for:

- tfm_s (secure side)
- tfm_ns (non-secure side)

3. Generate setup files:

- partition_gen.h (SAU / ISR assignment)
- mem_layout.h
- SystemIsolation_Config.c (MPC, PPC, security config)

An open approach for IoT on Cortex-M

*** Demo ***

© 2020 Arm Limited

An open approach for IoT on Cortex-M

- Arm takes a holistic view to the development cycle of IoT endpoint devices
- CMSIS provides the software interface standard for interoperability of software components

www.arm.com/psa - Platform Security Architecture to ensure ground up security in devices

www.keil.com/iot - Get started with cloud connectors on microcontrollers

Live demos

hitex EMBEDDED TOOLS & SOLUTIONS

Hall4 - Stand 360

IoT Security with TrustZone and TF-M on STM32L5

© 2020 Arm Limited

CMSIS-Build

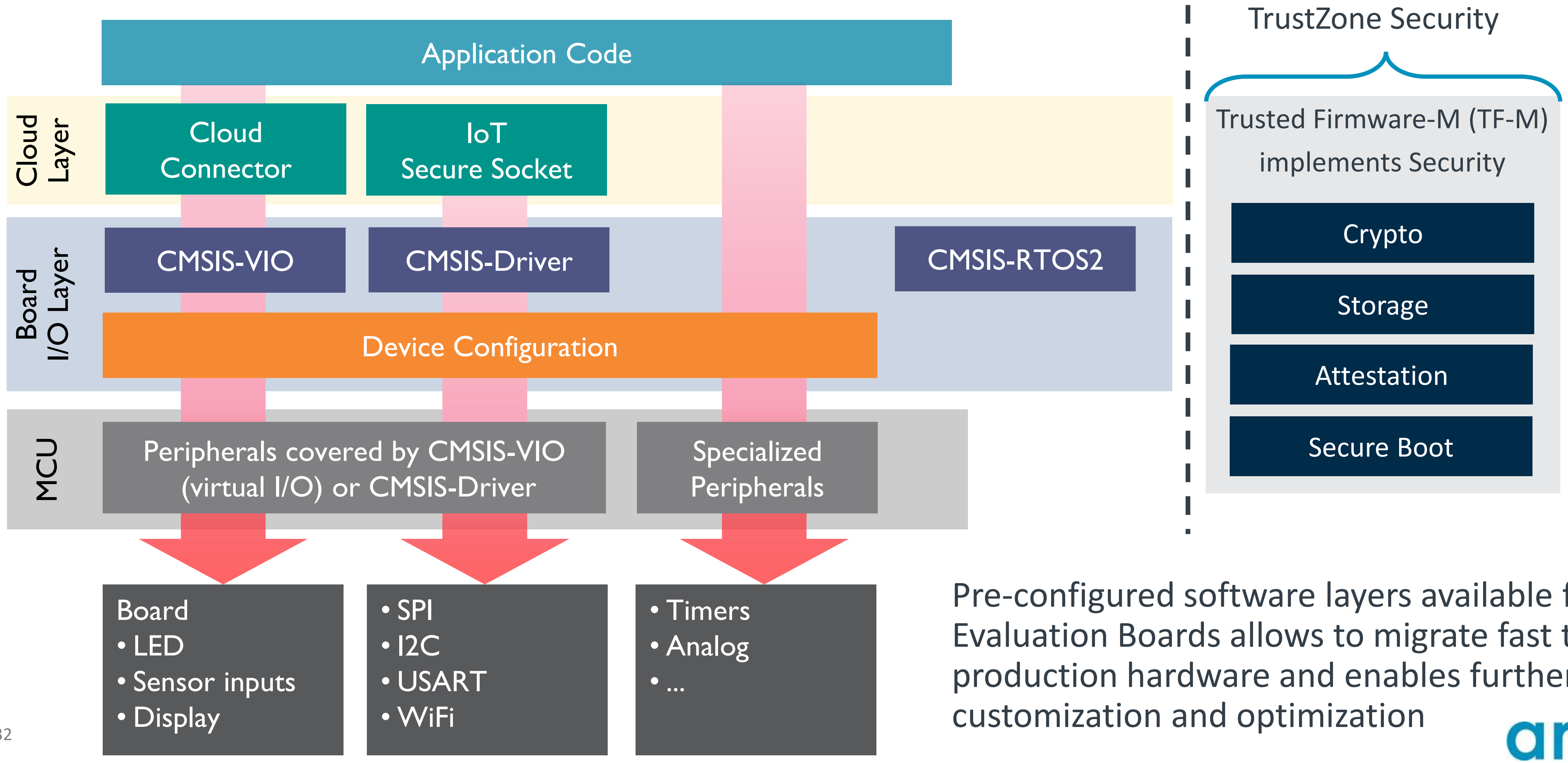
Productivity for complex software templates:

1. Software Layers for efficient code re-use
2. Generic project format
3. Continuous Integration (CI) workflow
4. Virtual I/O for fast migration from eval to production

© 2020 Arm Limited

1. Software layers group pre-configured software components

IoT for Cortex-M – move from eval kit to custom hardware; scale examples to many boards



Pre-configured software layers available for Evaluation Boards allows to migrate fast to production hardware and enables further customization and optimization

2. CMSIS – Generic Project Description (*.cprj) Format

Migrate projects to different IDE; facilitate construction of projects from software layers

Describes everything required for project build:

- CMSIS-Packs used by project for the selected software component
- Compiler toolchain including version (range) and essential command line options
- Hardware target information including device vendor, device name, enabled device features
- Software component selection and configuration file information.
- RTE folder containing preconfigured component configuration files.
- Project specific source code files

Export import to MDK and Eclipse CMSIS-Pack (basis for Arm DS, IAR EW-ARM, etc.)

- MDK release: April 2020, Eclipse CMSIS-Pack release: July 2020

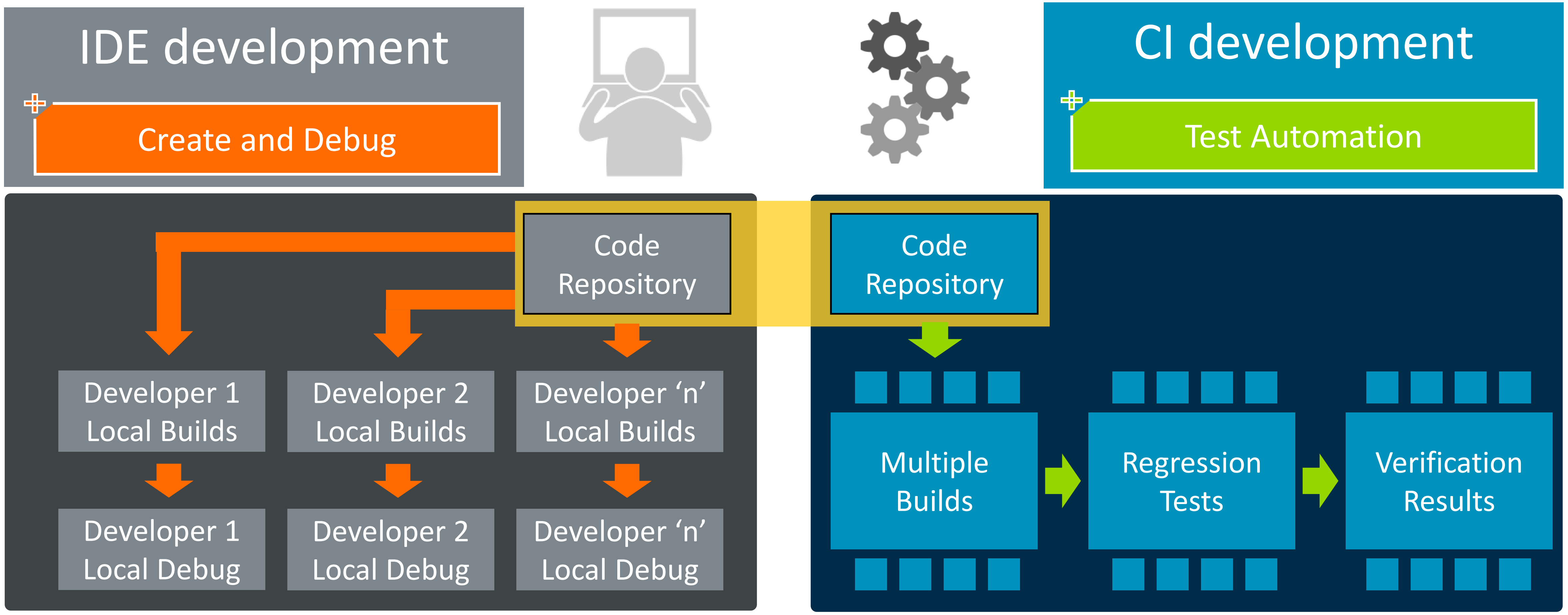
Command-line tools for standard Make builds with *.cprj based projects

- Supports automatic Software Pack upgrades including **ccmerge: Config File Updater**
- Construct projects from different software layers using the **cbuildgen: Build Process Manager**
- Version 1.0.0: Beta: April 2020, Release: July 2020

© 2020 Arm Limited

3. Continuous Integration (CI) workflow

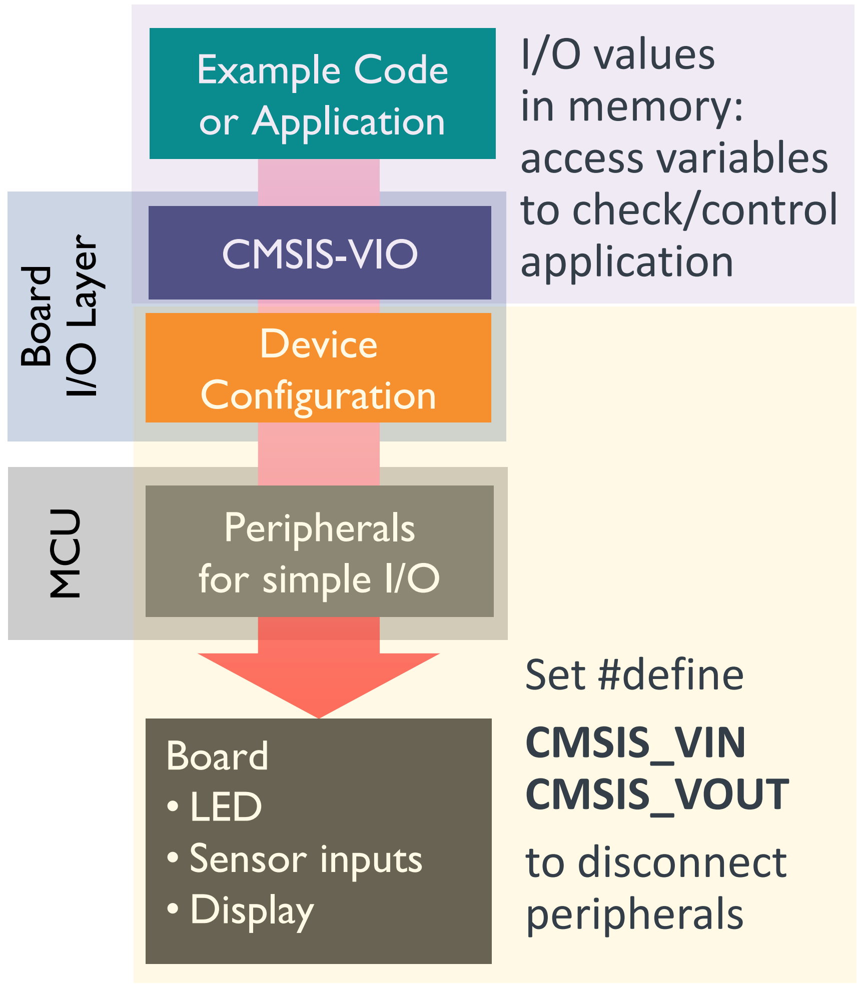
IDE and CI development combined delivers better tested products faster



© 2020 Arm Limited

4. Virtual I/O provides generic API for examples and testing

IoT for Cortex-M – move from eval kit to custom hardware; scale examples to many boards


Example Code or Application
I/O values in memory: access variables to check/control application
Board I/O Layer
CMSIS-VIO
Device Configuration
MCU
Peripherals for simple I/O
Set #define
CMSIS_VIN
CMSIS_VOUT
to disconnect peripherals
Board
• LED
• Sensor inputs
• Display

Program examples help users to understand software faster, but are difficult to scale to many hundred evaluation kits.

CMSIS-VIO solves that problem with:

- Consistent, simple interfaces to demo I/O peripherals
- Software simulation for peripherals that are not available

Program examples are a great start for application programs, but demo I/O is not available in production hardware:

CMSIS-VIO solves that problem with:

- #defines that disconnect the demo I/O

Program examples and application programs needs testing:

CMSIS-VIO solves that problem with:

- Variables accessible by test systems to control application

© 2020 Arm Limited

Summary and Actions

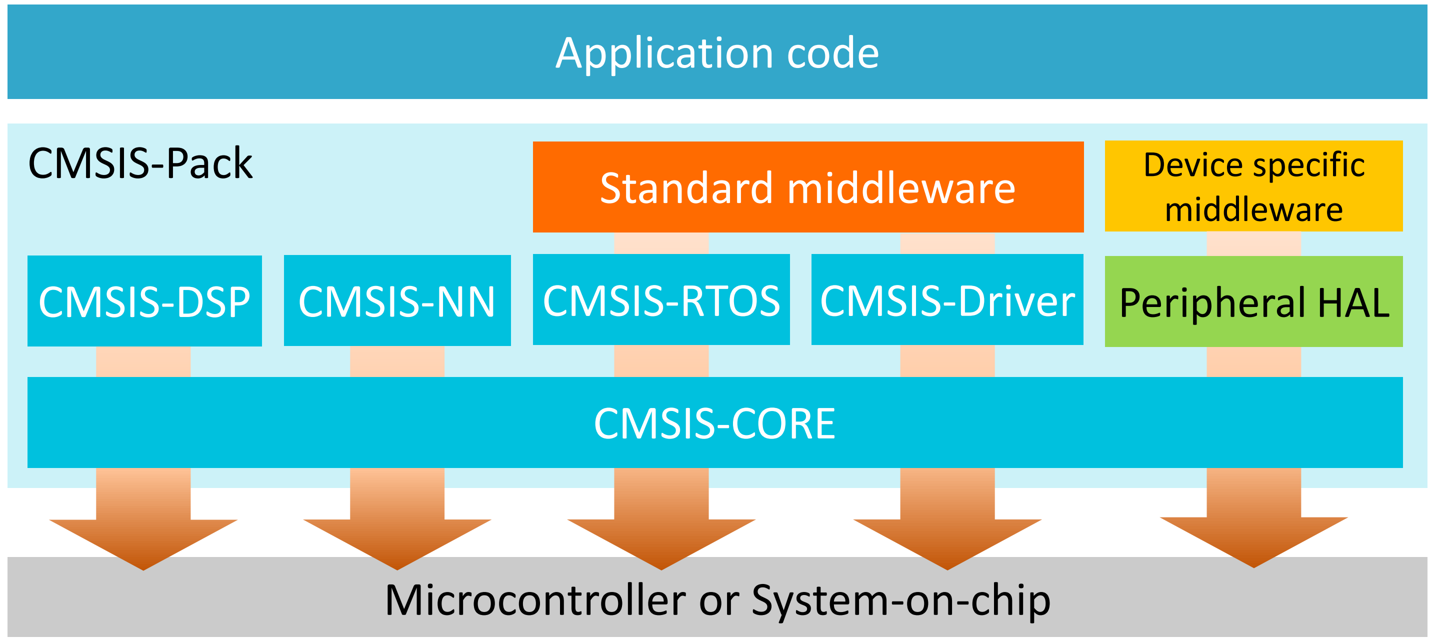
Collaborate with us

© 2020 Arm Limited

CMSIS timeline	Description	How you can contribute
Available already	WiFi driver implementations and IoT Connector implementations	WiFi chip set vendor: add your own driver Compiler vendor: adopt projects to your toolchain
March 2020	TFM beta release framed as CMSIS-Pack with adoption instructions to Cortex-M23/M33 devices (to create a platform)	Review live repository on https://github.com/arm-software/CMSIS-TFM For compiler vendors, help us to make it compatible with your toolchain
April 2020	**CMSIS v5.7.0** release with: - Core(M): Cortex-M55 - enhanced DSP + ML libraries - CMSIS-Build (beta)	Review live repository on https://github.com/arm-software/cmsis_5 Use '**Issues**' to report problems or raise requests
May 2020	**CMSIS-Zone** v1.1 with enhancements and MPU compression for v7M	For chip vendors: add *.rzone files for devices to https://github.com/arm-software/cmsis-zone
May / June 2020	**Tutorials** for IoT connectivity with TrustZone enabled devices	Tbd
Outlook	CMSIS-Build: final release and examples for the software layer concept CMSIS-DAP: Tooling for CMSIS-VIO control and Secure Debug CMSIS-Pack/SVD: better integration of CMSIS-Zone *.rzone files	

CMSIS

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



Benefits for the software developer

- Unified software interfaces
- Reduced learning effort
- RTOS-agnostic middleware
- Project and code templates
- Consistent APIs for device peripherals
- Software deployment and PLM

CMSIS supports the complete development flow (from eval to production) and system optimization

Easy evaluation with program examples for evaluation kits

Fast development with ready-to-use software components and templates

Reliable systems with FuSa certified software components

© 2020 Arm Limited

IoT devices are different – how can we deploy IoT software stacks efficiently at scale?

Using the techniques described in this presentation we can support 10.000 bespoke designs

Thank You

Danke

Merci

谢谢

ありがとう

Gracias

Kiitos

감사합니다

धन्यवाद

شكرًا

תודה

© 2020 Arm Limited

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

© 2020 Arm Limited