arm Tech Symposia

Practical steps for developing secure IoT endpoints

Senior Marketing Manager
Arm
Zhang Zheng

#Arm Tech Symposia

Agenda

Components that enable secure IoT endpoints

- Trusted Firmware-M (TF-M) implements the PSA software services
- CMSIS components for device software foundation
- Hardware security of Armv8-M based devices

Challenges of IoT endpoint development

- Resource partitioning and security setup of the system hardware
- Using standard software components on a diverse range of devices
- Validation and verification of secure embedded systems

Development tools and software components that help

- CMSIS-Zone manages system resources and tool configuration
- Software packs that manage software components for diverse devices
- Compliance tests and tool features for validation and verification

arm TechCon

Components of a secure IoT endpoint

Software and hardware landscape

Platform Security Architecture

A complete security offering. Independently tested.

Analyze

Threat models
& security analyses

Architect

Hardware & firmware
architect specifications

Implement

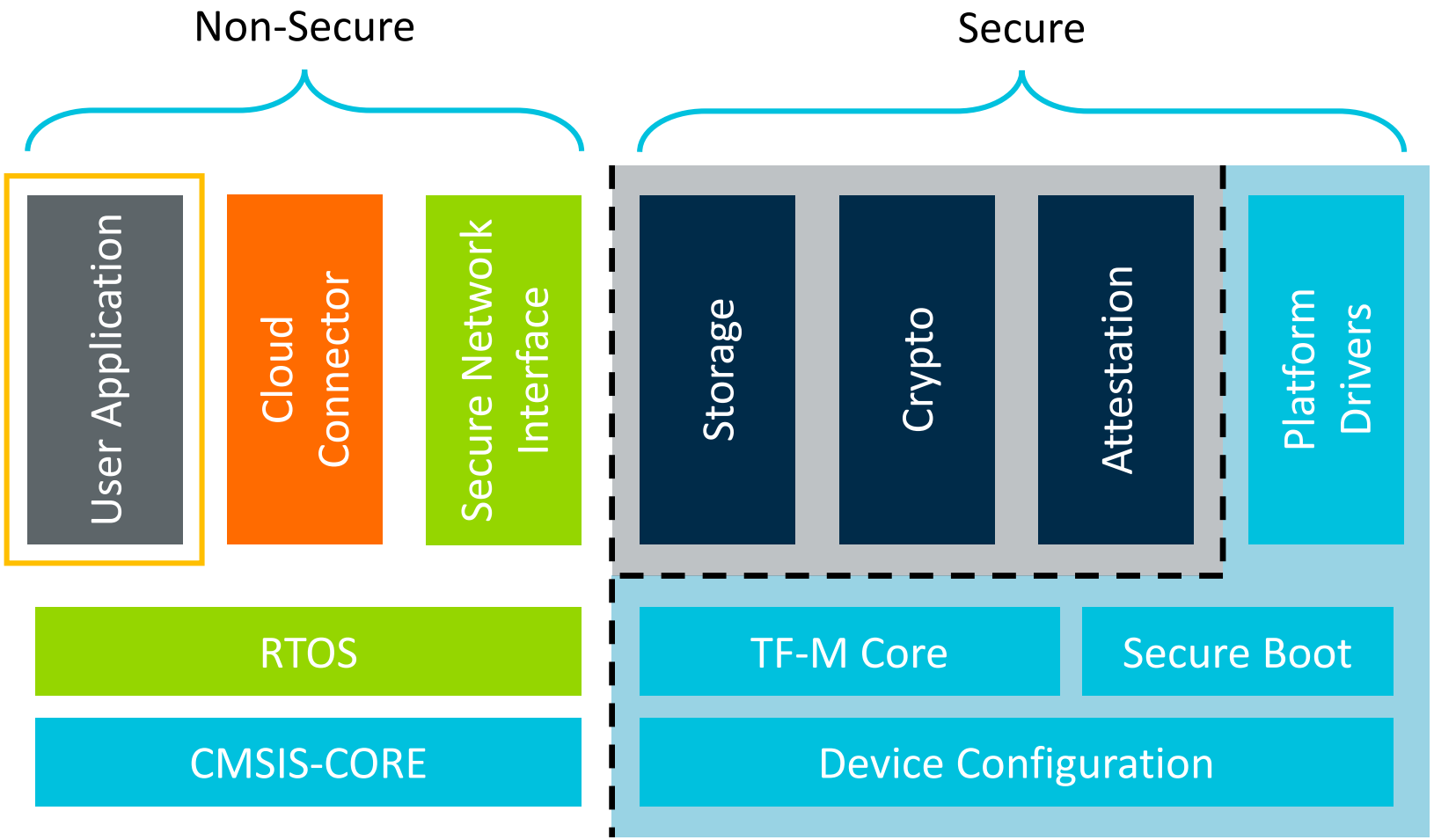
Firmware
source code

Certify

psacertified™

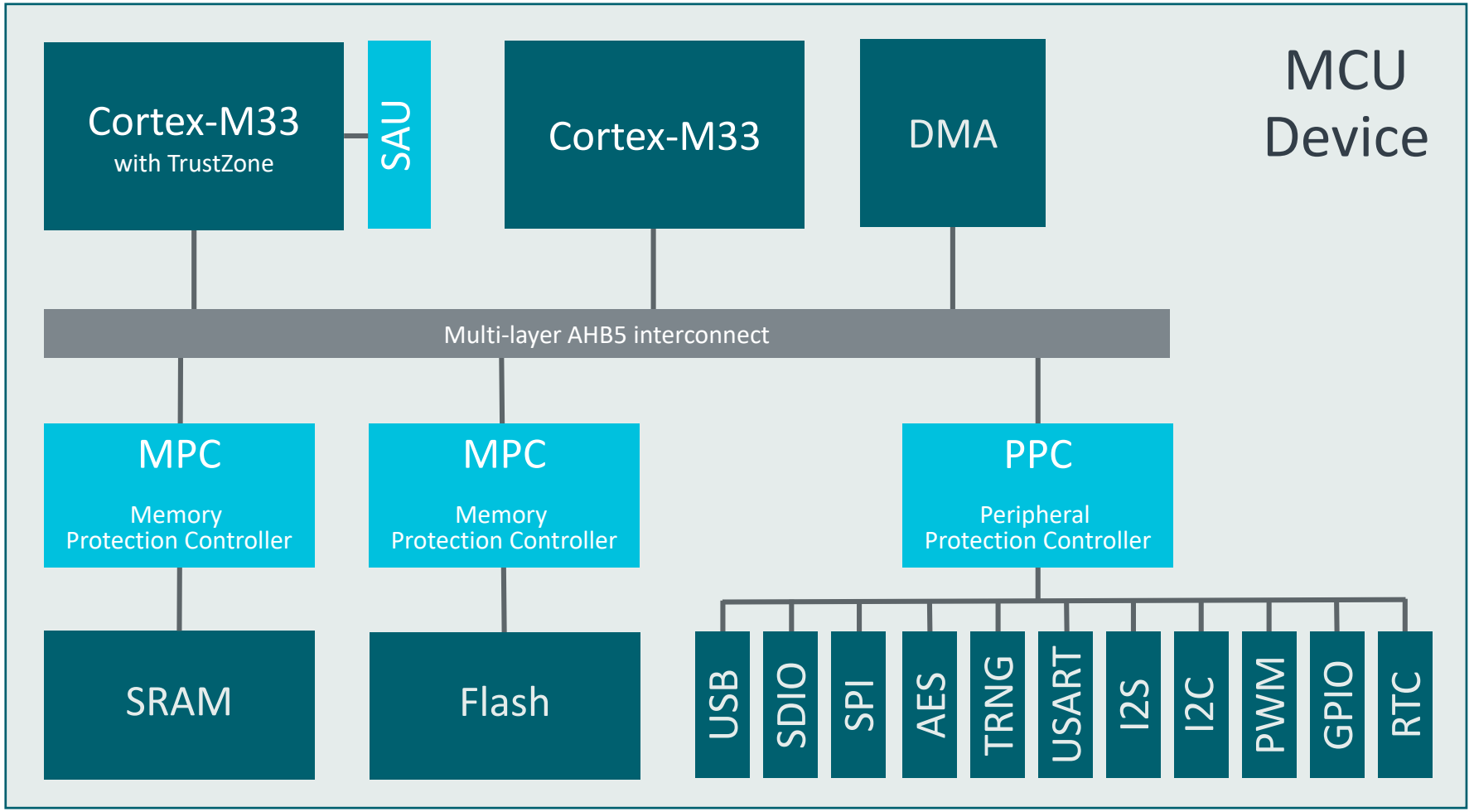
Trusted Firmware-M – Framework for Cortex-M23/M33

Cloud software stack for Armv8-M utilizing TrustZone



Devices for Secure IoT Endpoints

Rich feature sets and security mechanisms of today's microcontrollers



The Challenge

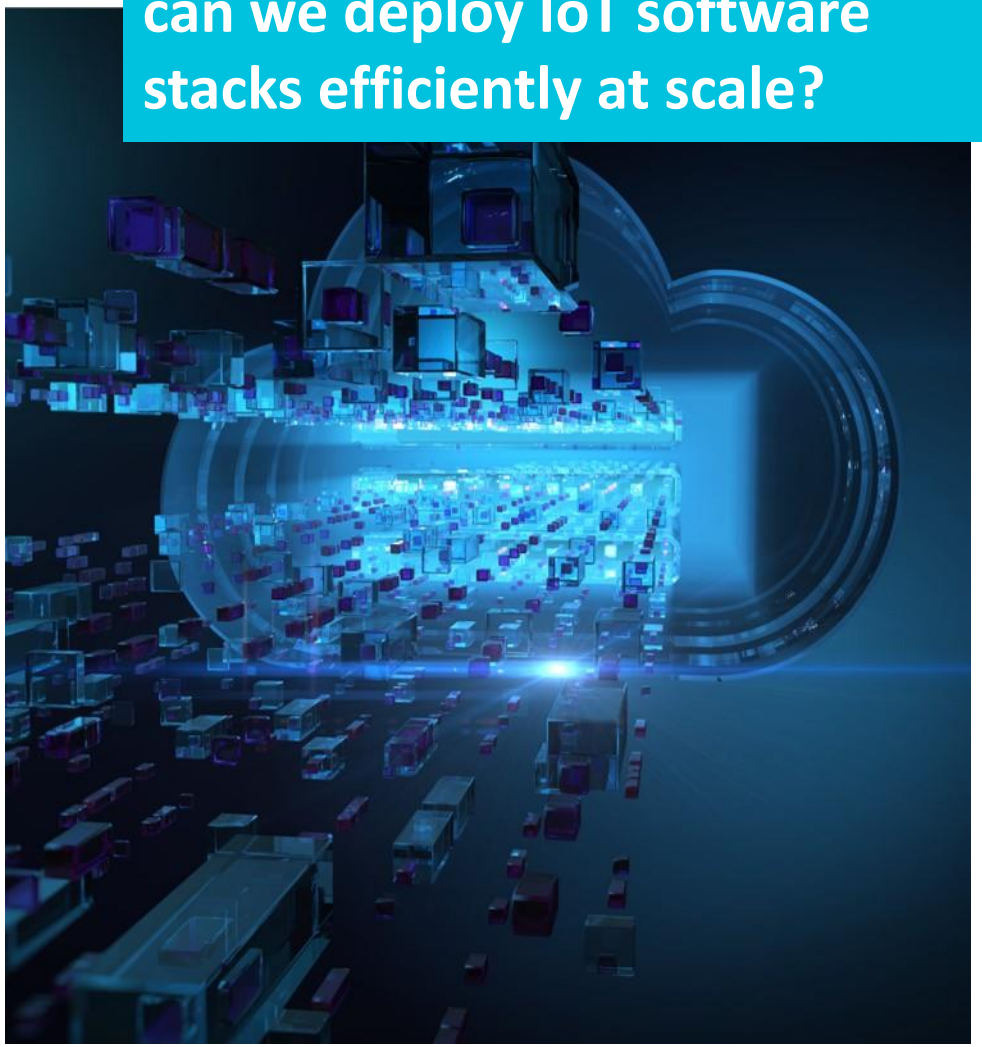
The security settings of SAU, MPC, and PPC must be consistent.

arm TechCon

Challenges of IoT Endpoint Development

Click to add subtitle

IoT devices are different, so 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

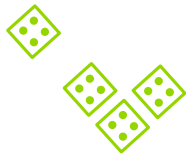
Challenges of IoT Endpoint Development

Using standard software components on diverse devices

Device hardware configuration

- Multi-processor devices share resources that need partitioning
- Programmers want simple views to available resources
- Various hardware security filters require consistent setup
- Configuration of the software development tools must match the overall hardware setup

Manage software components



- Complex IoT software stacks should rely on standard software components
- Devices use foundation software, hardware drivers, and need application specific setup
- Memory footprint is important
- Managing 3rd party software over the product lifecycle

System Validation

- Verification of security is a new challenge in the embedded industry
- How to ensure the time deterministic operation that is required in many embedded systems
- IoT endpoints should be power efficient, but hardware configuration is complicated

arm TechCon

Development tools and software components that help

Challenge: Device Hardware Configuration

Solution: CMSIS-Zone simplifies Armv8-M system configuration

Partition a multi-processor system into single processor views

Setup memory and peripherals for secure/non-secure environment

Generate consistent configuration

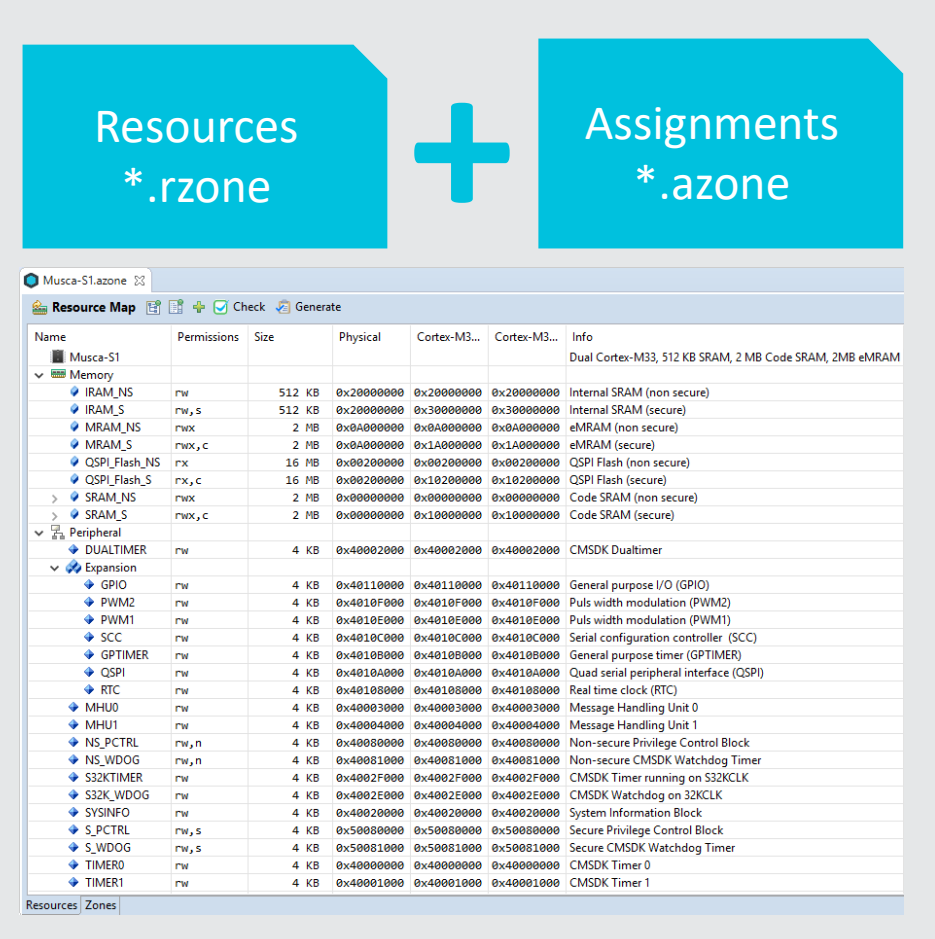
- Setup for the Armv8-M TrustZone (SAU, Interrupt assignment to Secure/Non-Secure)
- Setup of device specific Memory Protection Controller (MPC)
- Setup of device specific Peripheral Protection Controller (PPC)

Generate related linker configuration to ensure consistency

→ Learn more about device configuration with CMSIS-Zone

CMSIS-Zone – Development Workflow

Configuration and build management for system resources



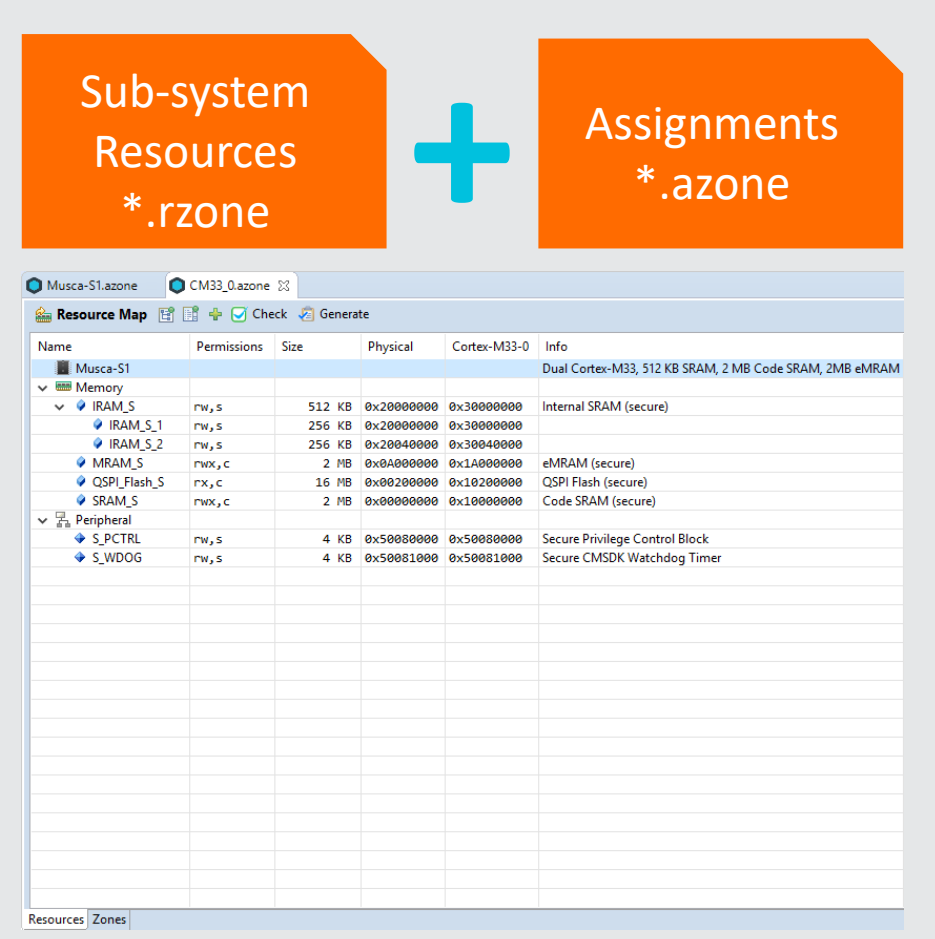
Resource *.rzone file lists all available systems resources.

Assignment *.azone file contains partitioning information and is managed by CMSIS-Zone tool

CMSIS-Zone tool generates sub-system resource files

CMSIS-Zone – Development Workflow

Multi-step approach shows only relevant sub-system



Sub-sub-system Resources *.rzone

Hardware Configuration

Tool Configuration

It is possible to break down complexity of a system in multiple steps.

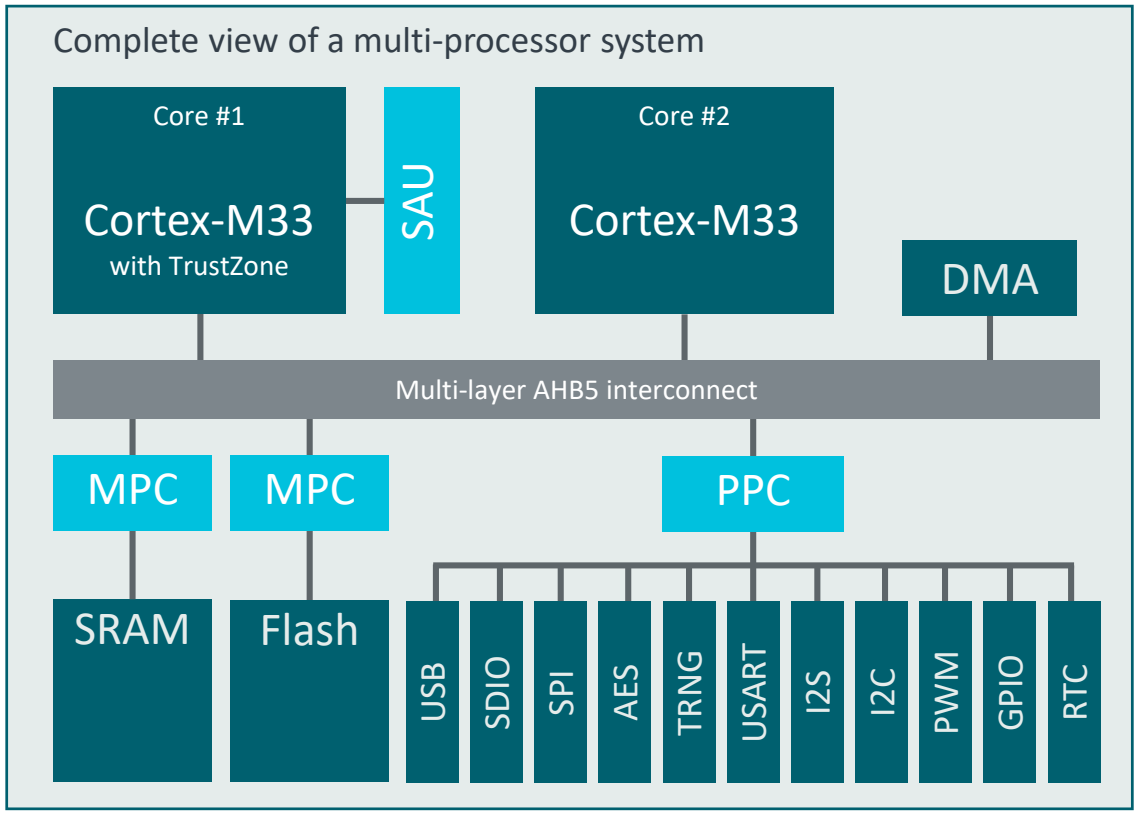
Sub-systems expose only the part of the system that is relevant for the user.

A sub-system user has no visibility to other parts of the system (as it typically configures also the related access protection).

Configuration Steps

Example

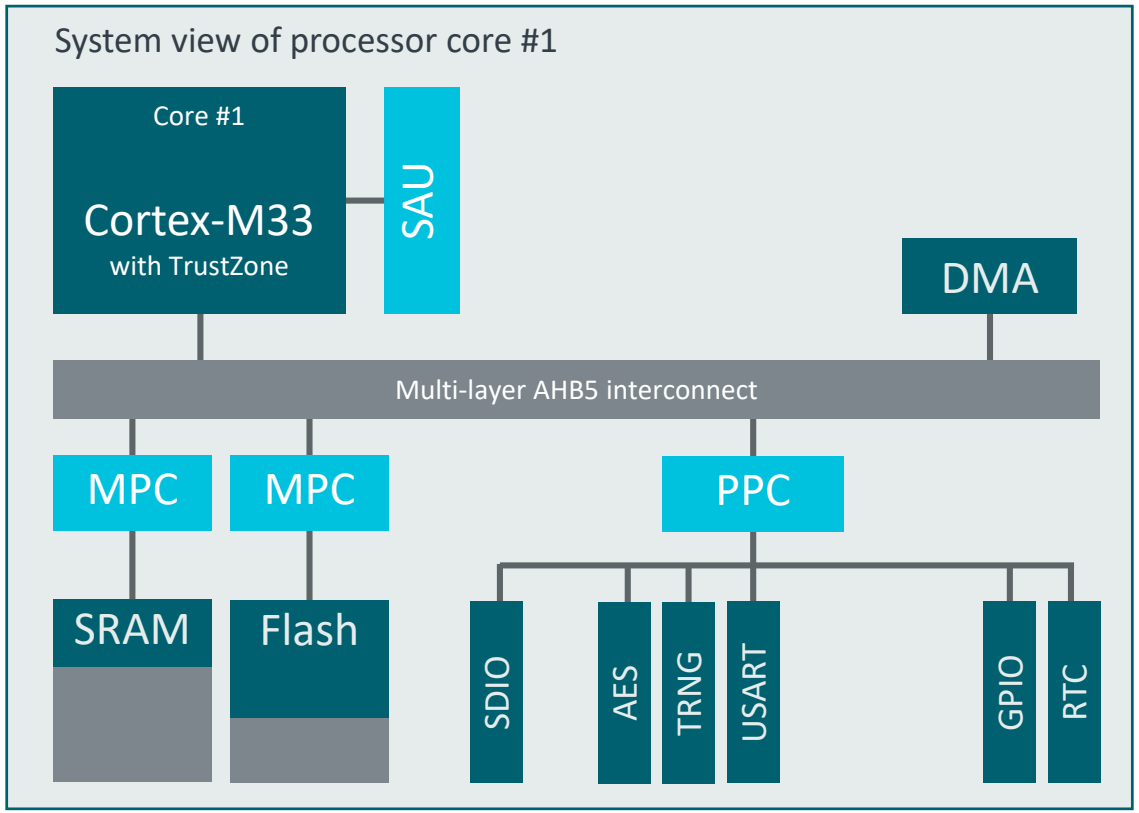
Step 1: split the multi-processor system into single processor sub-system

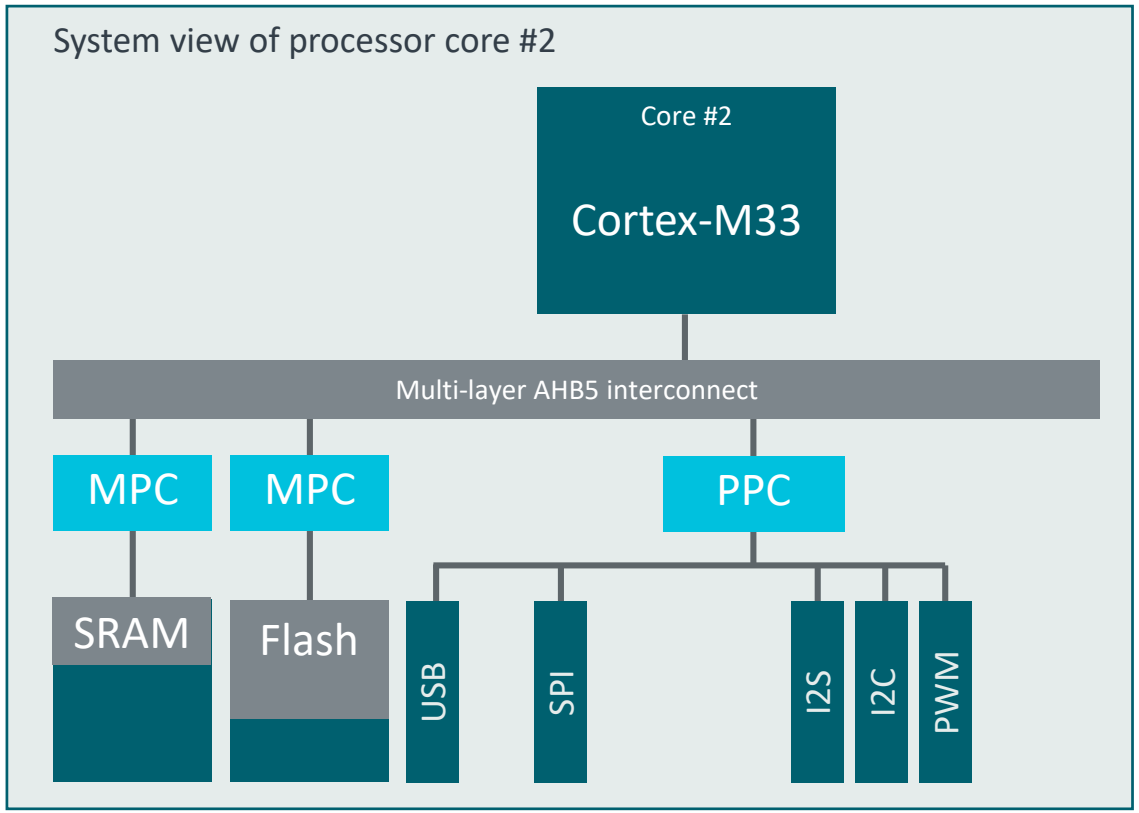


Configuration Steps

Example

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





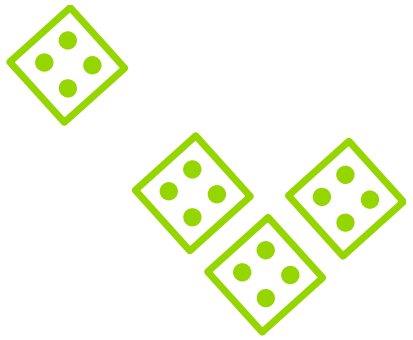
Step 2: create the partitions for secure and non-secure execution

arm TechCon

CMSIS-Zone Demo

Challenge: Manage Software Components

Solution: CMSIS-Pack seamlessly combines software components



CMSIS-Pack defines a software delivery mechanism

- Software components, code templates, documentation, device and board support
- Software versioning for product lifecycle management (PLM)
- Manage dependencies to other software components
- Ensure consistent software APIs across components
- Simplify retargeting as software is selected based on hardware and toolchain

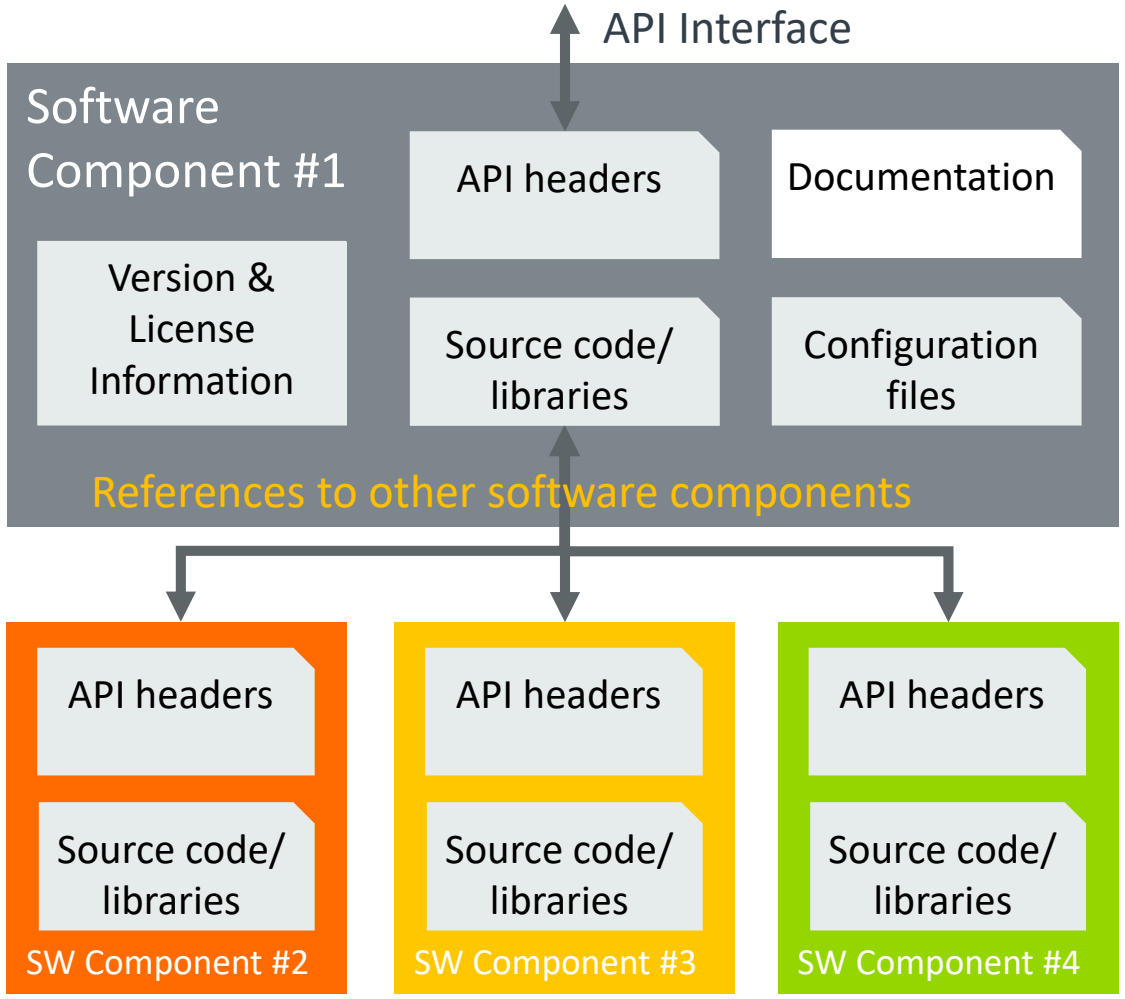
CMSIS-Pack system is today available in several mainstream toolchains

- Arm DS Development Studio and Arm Keil MDK
- IAR Embedded Workbench for Arm
- CMSIS Eclipse Plug-In (open-source) that is basis for several other implementations

→ Learn more about CMSIS software components

CMSIS-Pack

What is a software component?



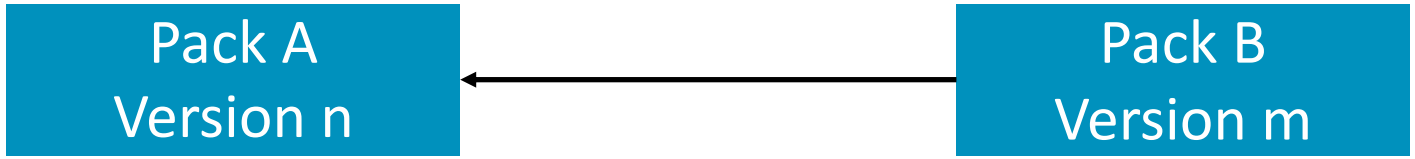
Software components should have:

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

CMSIS-Pack defines an XML format that frames this information and is used by project management utilities from various tools.

Relationships of Packs and Software Components

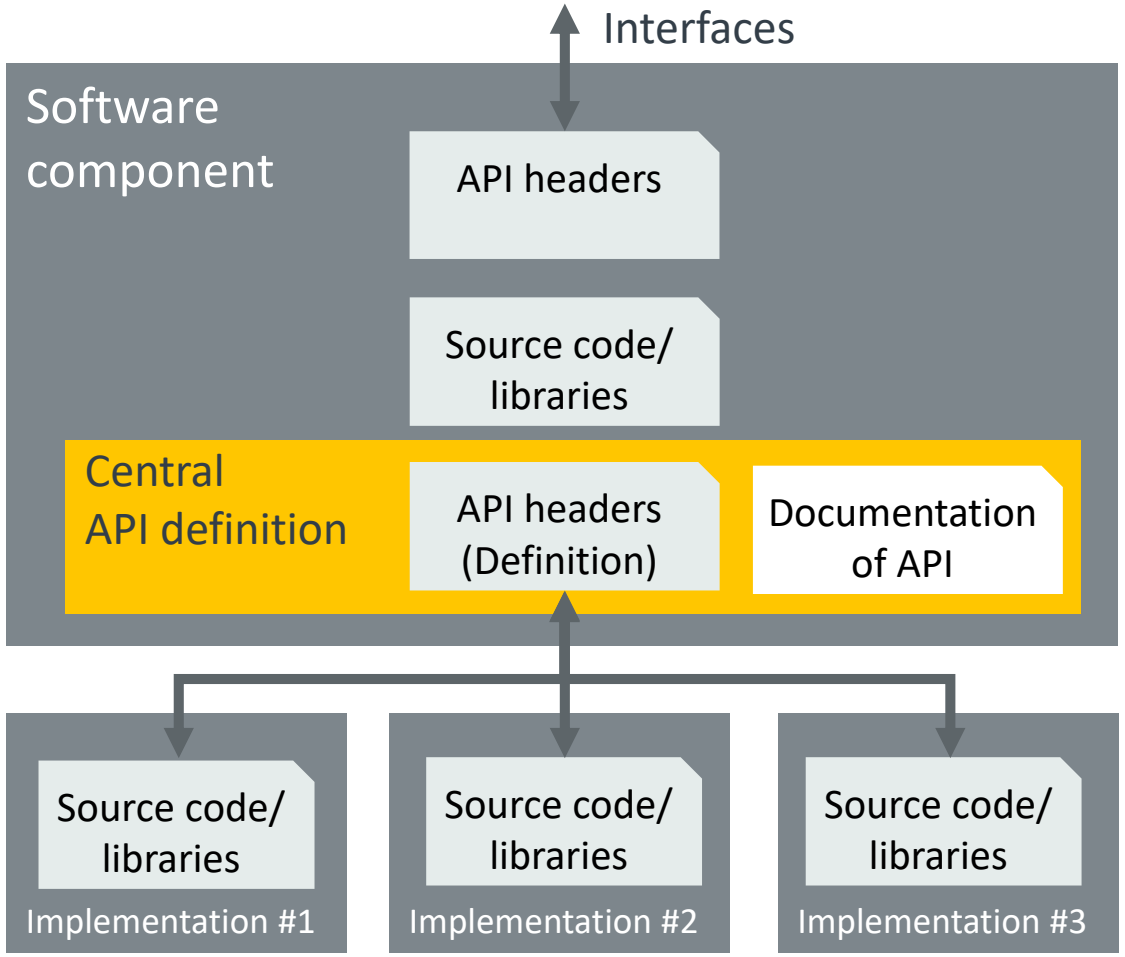
Packs can require other packs to be installed:



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

Central API Interface definition for software components

API headers evolve over time

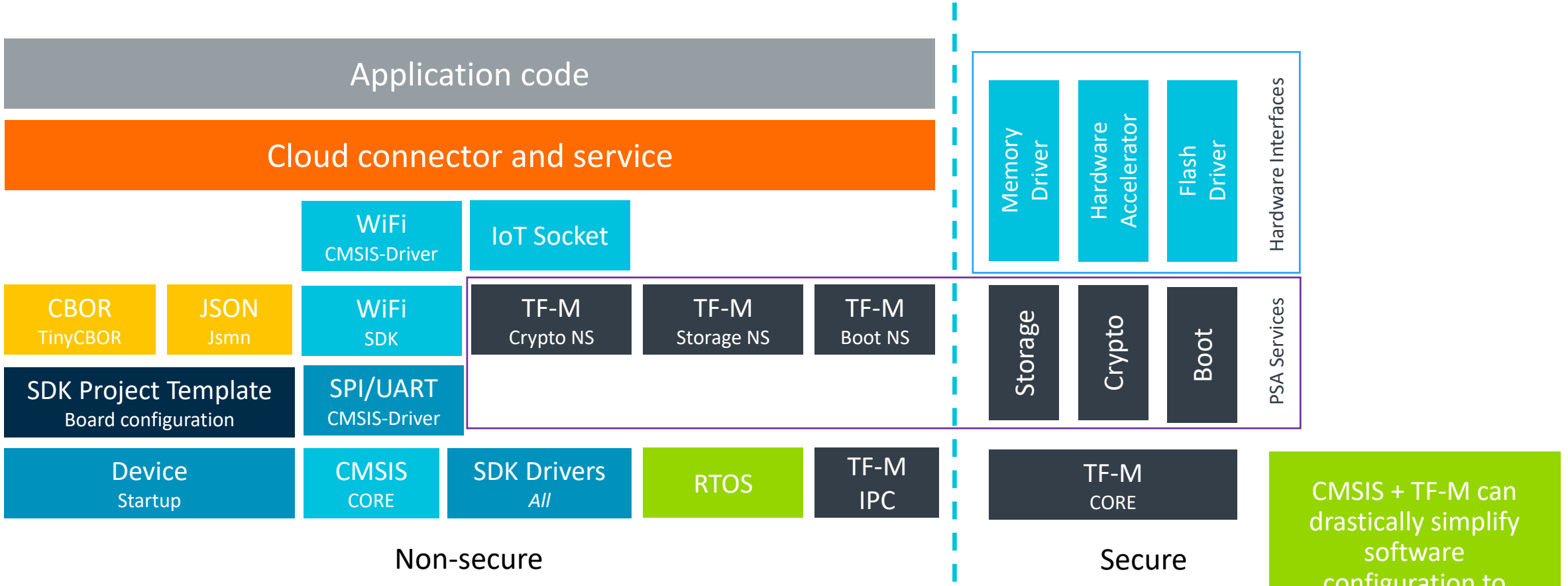


A central API definition shares header file and documentation of an API interface across multiple other software components to ensure consistency.

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

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

Cloud software stack on Armv8-M with TrustZone



Alignment with CMSIS and TF-M can simplify security, crypto, and firmware update

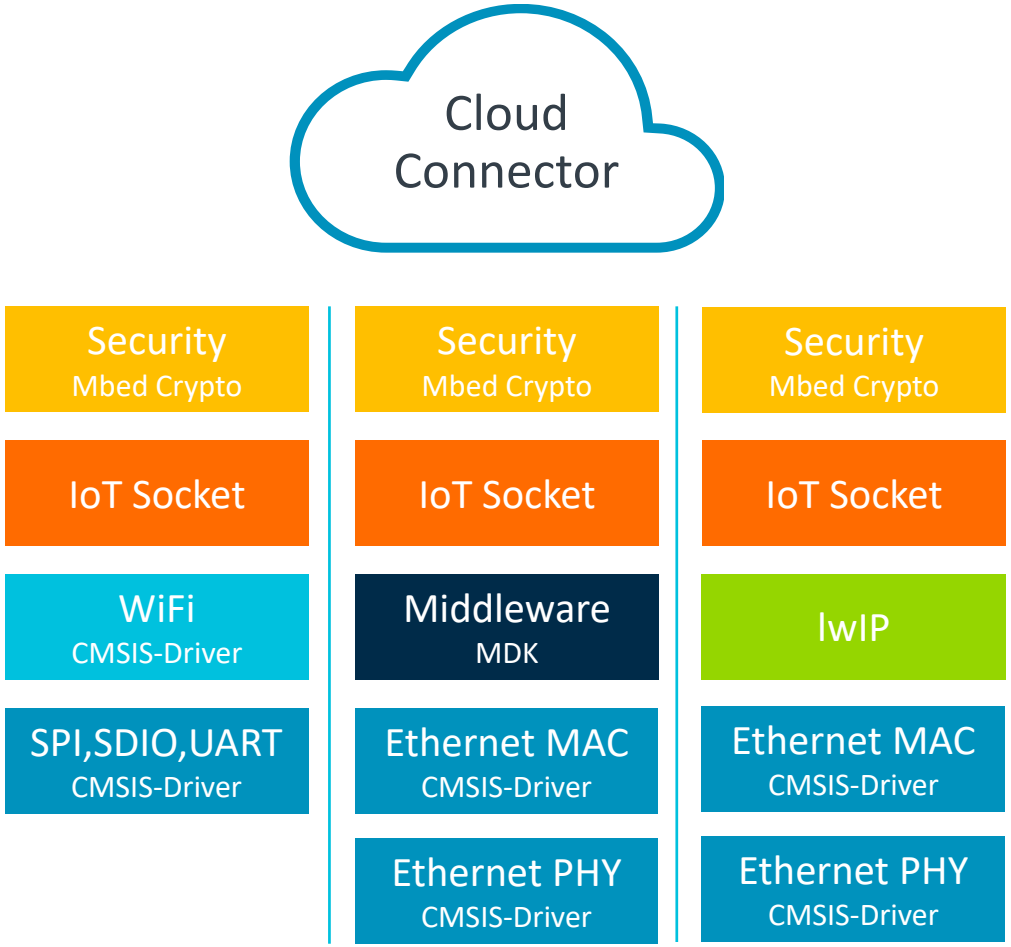
CMSIS pack | RTOS pack | Cloud pack | BSP | DFP | 3rd party pack | Mbed TLS pack | Com pack

arm TechCon

Component System Demo

WiFi driver and IoT socket component combined

Generic communication foundation for cloud connectors on Cortex-M



Azure

IBM

Google Cloud Platform

aws

arm PELION

Security is provided by Mbed Crypto

IoT Socket can interface with:

- WiFi CMSIS-Driver to connect to various wireless chipsets
- MDK-Middleware network stack
- LwIP (optional, WiP – contributions welcome)

Challenge: System Validation

Solution: Verification with PSA test and MDK debug features

PSA Functional API Test Suite for testing security functions of TF-M

- API compliance test suite validates Crypto, Secure Storage and Attestation
- Test results are fundament for the PSA Certified program
- These test results give confidence on the system security implementation

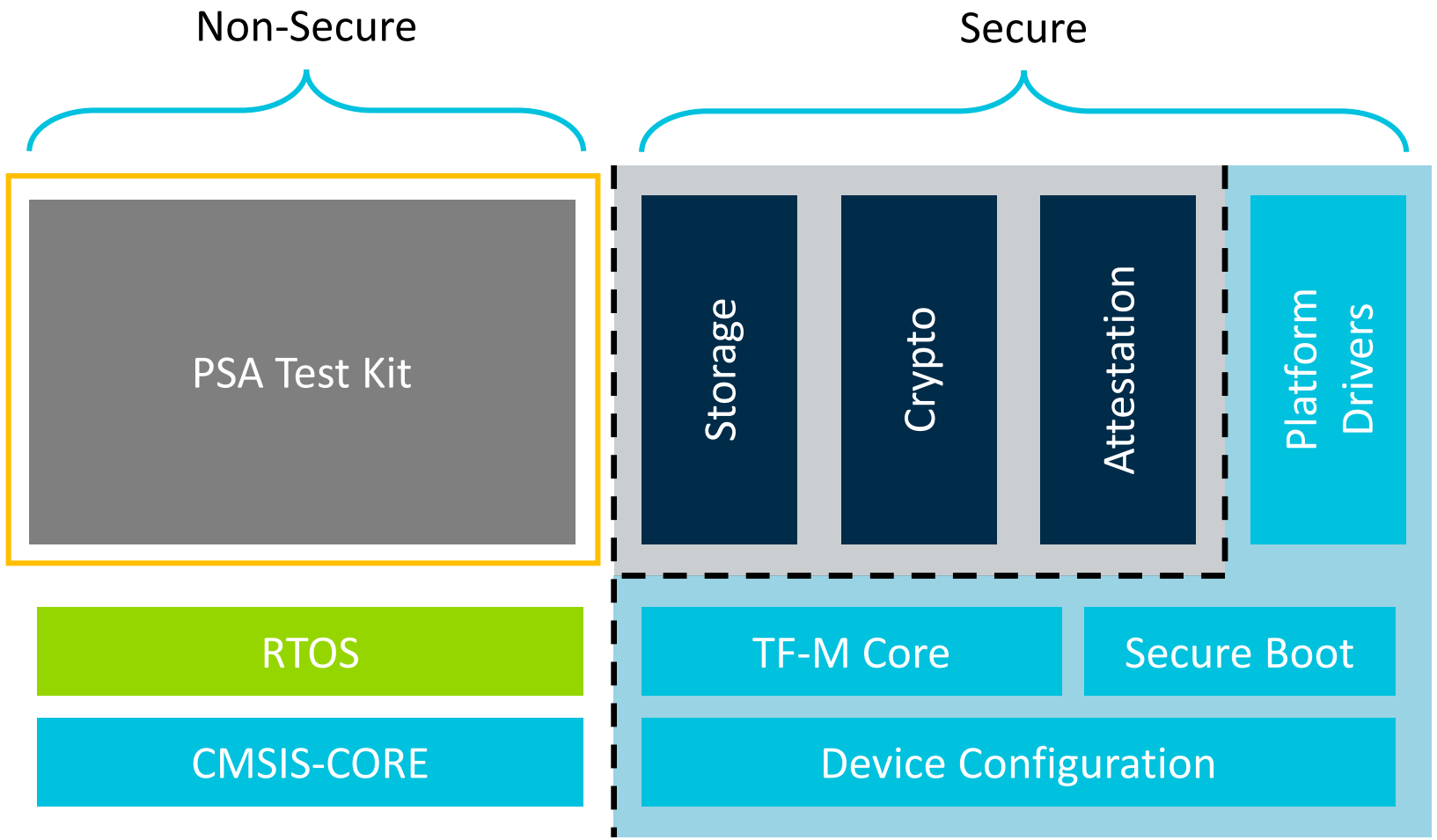
Arm Keil MDK includes features for IoT endnote validation

- System Analyzer shows the overall system timing (ISR, threads, user events)
- Component Viewer shows the status of software components
- Event Recorder records timing of events
- Power Measurement with ULINKplus helps to optimize power consumption

→ Learn more about IoT System Validation

Validate Security Functions of TF-M Implementation

Running the PSA Functional API Test Suite on your implementation



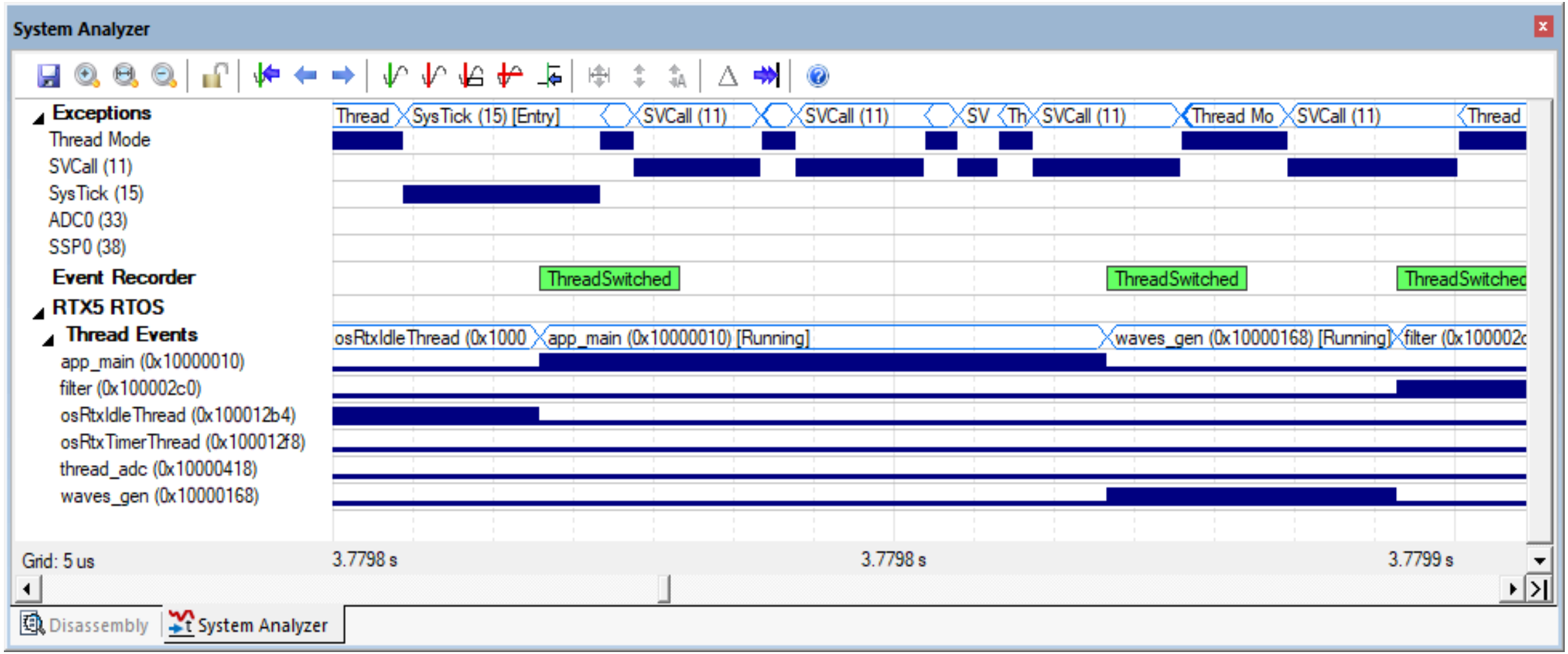
PSA Test Kit

Gives confidence on system security implementation

Verifies Crypto, Storage and Attestation

Timing Analysis of an Embedded System

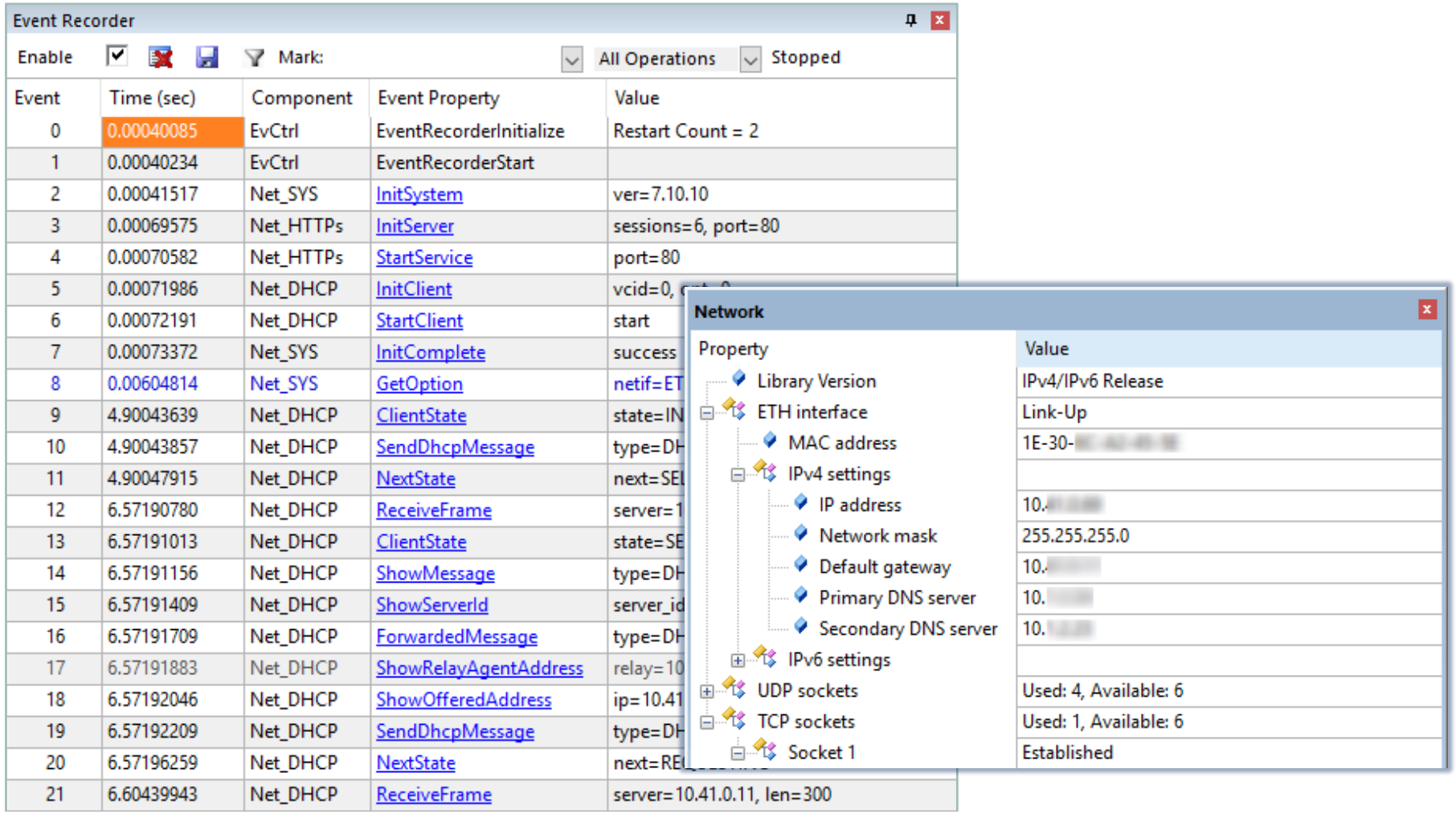
Check run-time status and dynamic software behavior



Analysis for Software Components

Event Recorder showing status and dynamic software behavior

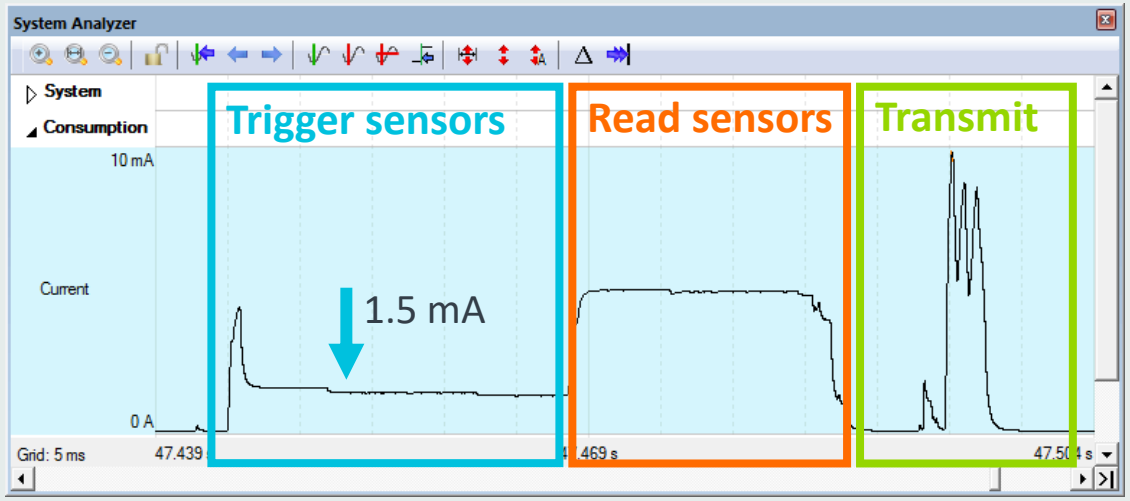
- Dynamic operation with timing
- Status information
- Preconfigured for MDK-Middleware, RTX5, and FreeRTOS
- Easy to add to custom components



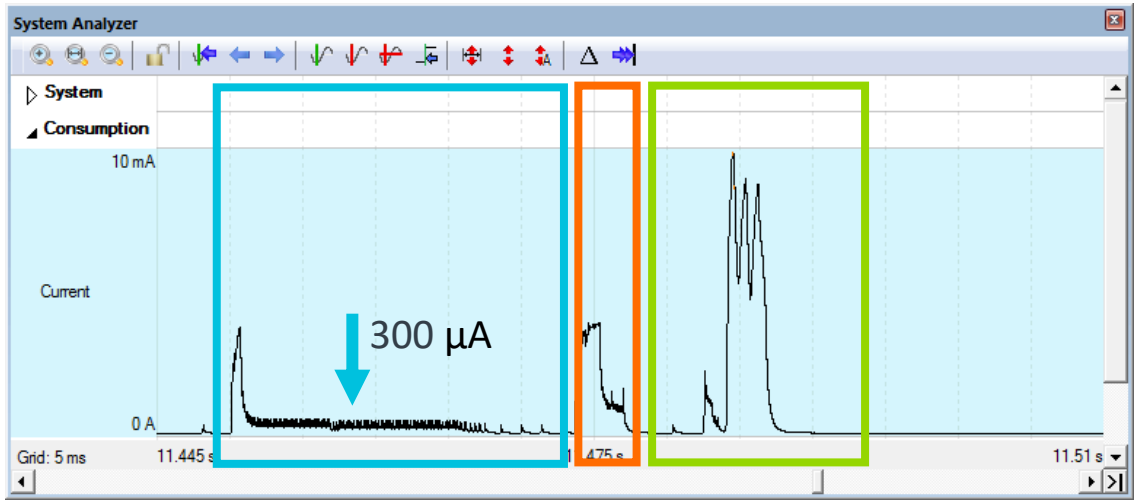
Analyzing and Optimizing Energy Consumption

Example: IoT Sensor Interface

original



optimized



ULINKplus

- Power measurement
- 50 Mbps SWO trace
- 1 kV isolation
- I/O pins for test automation

Optimizations identified

- Triggering sensors with lower current
- Read sensors faster with less MCU activity
- Identify hardware issues

Overall 75% less energy consumed

Summary

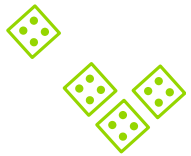
Using standard software components

Device hardware configuration

✓CMSIS-Zone simplifies resource partitioning and clearly shows available resources.

✓Generate consistent setup for access control and software tools.

Manage software components



- ✓ Software packs provide ready-to-use software components (CMSIS, RTOS, TF-M, IoT Socket).
- ✓ Integration of the CMSIS-Pack system into modern IDEs helps to setup working software on the selected device.

System validation

- ✓ PSA test suite gives confidence in system security configuration.
- ✓ Keil MDK analyzes complex systems and helps to identify timing or power problems.

TF-M and CMSIS let you focus on application development for secure IoT endpoints.
Thus, it becomes as easy as classic embedded systems programming.

https://community.arm.com/cn/

Trademark and copyright statement
The trademarks featured in this presentation are registered and/or unregistered 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.

Copyright © 2019

Thank You!