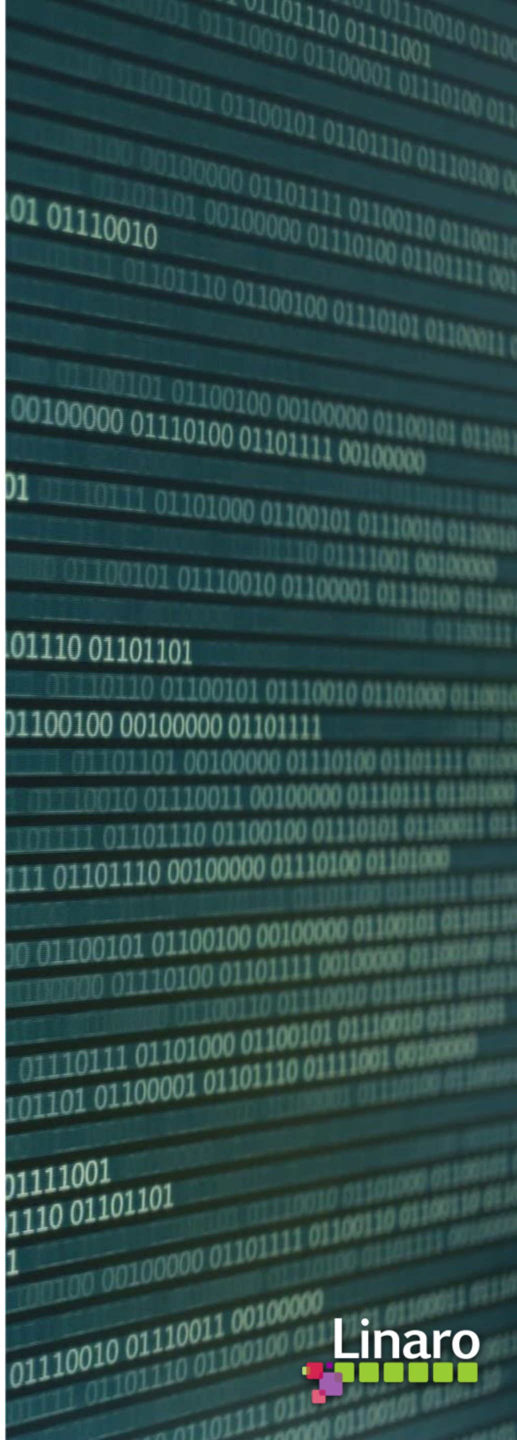
Open-CMSIS-Pack

Technical Project Meeting 2021-10-12

This meeting is recorded !

Agenda

- Top-Level Concept and Requirements [Reinhard]
- ProjManager – demo first PoC implementation [Daniel B.]
- Review of ST concept - any feedback?
- Feedback on Handlebars
- Next steps

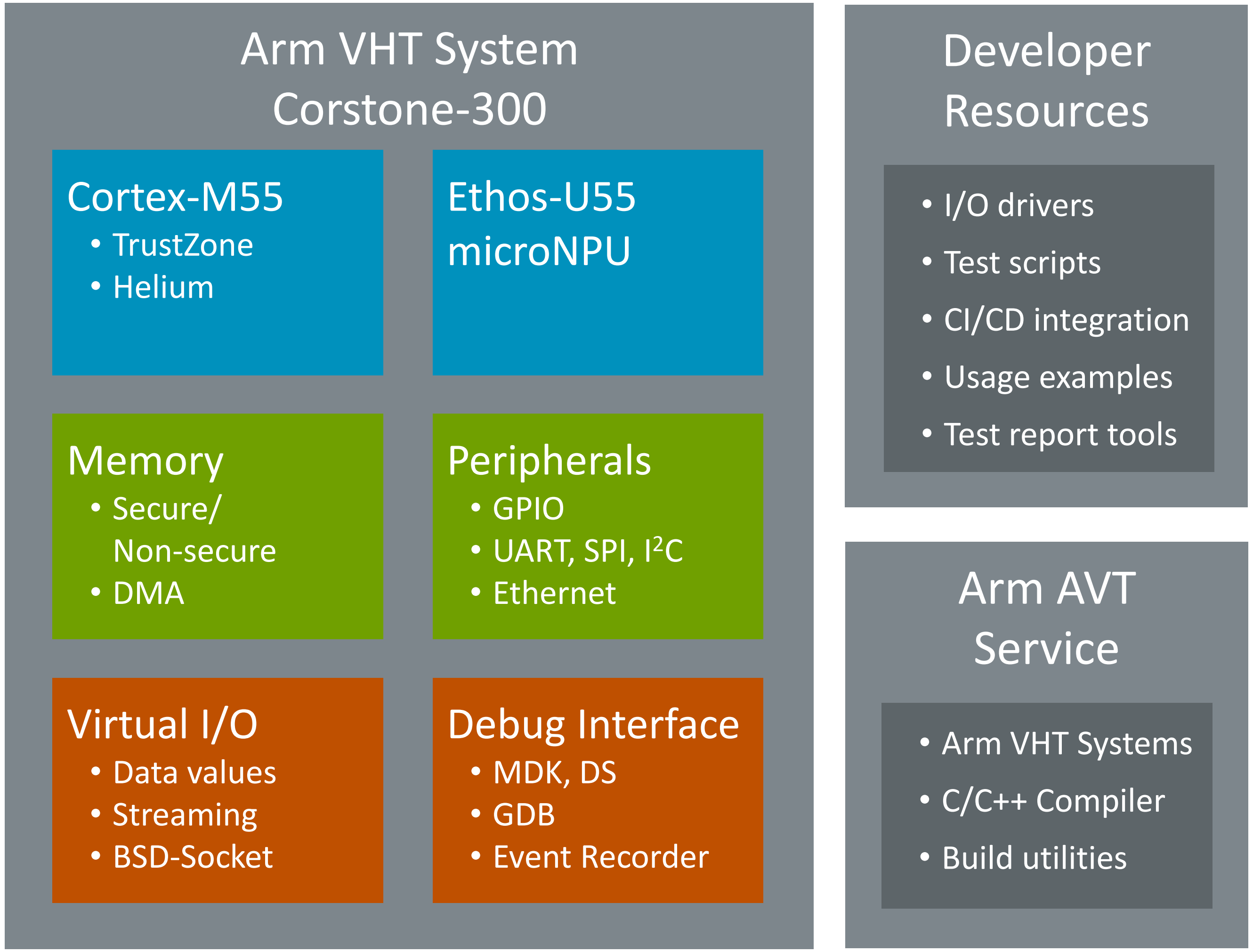


Top-Level Concepts

https://github.com/Open-CMSIS-Pack/Open-CMSIS-Pack/issues/6

Arm Virtual Hardware

- Precise **simulation models** of Cortex-M device sub-systems designed for complex software verification and testing
- Runs any RTOS or bare metal code
- Provides virtual peripheral interfaces for I/O simulation
- Enables test automation of diverse software workloads, including unit, integration tests, and fault injection
- Cloud service that can be integrated in **CI/CD** and **MLOps** development flows



Arm VHT System Corstone-300

Cortex-M55
- TrustZone
- Helium

Ethos-U55 microNPU

Memory
- Secure/ Non-secure
- DMA

Peripherals
- GPIO
- UART, SPI, I^{2}C
- Ethernet

Virtual I/O
- Data values
- Streaming
- BSD-Socket

Debug Interface
- MDK, DS
- GDB
- Event Recorder

Developer Resources
- I/O drivers
- Test scripts
- CI/CD integration
- Usage examples
- Test report tools

Arm AVT Service
- Arm VHT Systems
- C/C++ Compiler
- Build utilities

Workflow for CI: Develop Application Code or Test Cases

Flexible workflows addresses the needs of every developer

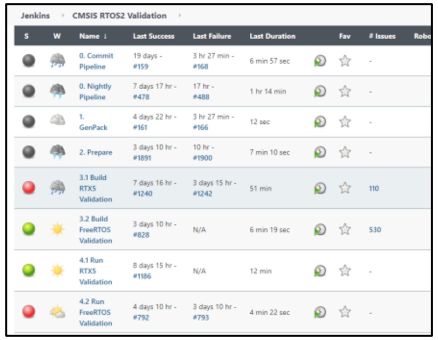
CI hosted in the Cloud

GitHub – Runners

Commit triggers **GitHub Actions** that starts CI using Arm Compiler for build and/or Virtual Hardware for testing.

GitHub Commit

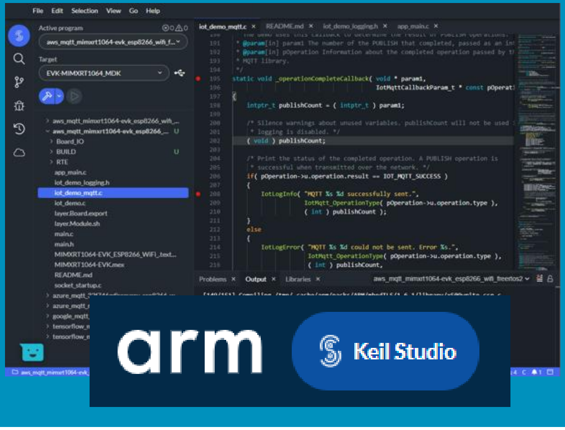
Test Results



Jenkins › CMSIS RTOS2 Validation ›

S	W	Name ↓	Last Success	Last Failure	Last Duration		Fav	# Issues	Robo
●	⛈	0. Commit Pipeline	19 days - #159	3 hr 27 min - #168	6 min 57 sec		☆	-	
●	⛈	0. Nightly Pipeline	7 days 17 hr - #478	17 hr - #488	1 hr 14 min		☆	-	
●	☁	1. GenPack	4 days 22 hr - #161	3 hr 27 min - #166	12 sec		☆	-	
●	⛈	2. Prepare	3 days 10 hr - #1891	10 hr - #1900	7 min 10 sec		☆	-	
●	⛈	3.1 Build RTX5 Validation	7 days 16 hr - #1240	3 days 15 hr - #1242	51 min		☆	110	
●	☀	3.2 Build FreeRTOS Validation	3 days 10 hr - #828	N/A	6 min 19 sec		☆	530	
●	☀	4.1 Run RTX5 Validation	8 days 15 hr - #1186	N/A	12 min		☆	-	
●	⛅	4.2 Run FreeRTOS Validation	4 days 10 hr - #792	3 days 10 hr - #793	4 min 22 sec		☆	-	

Cloud flow with IDE in Browser



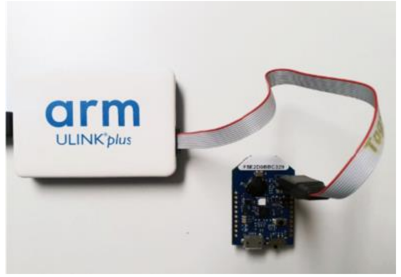
```
 * @param[in] param1 The number of the PUBLISH that completed, passed as an in
 * @param[in] pOperation Information about the completed operation passed by th
 * MQTT library.
 */
static void _operationCompleteCallback( void * param1,
                                        IotMqttCallbackParam_t * const pOperat
{
    intptr_t publishCount = ( intptr_t ) param1;

    /* Silence warnings about unused variables. publishCount will not be used i
     * logging is disabled. */
    ( void ) publishCount;

    /* Print the status of the completed operation. A PUBLISH operation is
     * successful when transmitted over the network. */
    if( pOperation->u.operation.result == IOT_MQTT_SUCCESS )
    {
        IotLogInfo( "MQTT %s %d successfully sent.",
                    IotMqtt_OperationType( pOperation->u.operation.type ),
                    ( int ) publishCount );
    }
    else
    {
        IotLogError( "MQTT %s %d could not be sent. Error %s.",
                     IotMqtt_OperationType( pOperation->u.operation.type ),
                     ( int ) publishCount,
```

Classic Tools on Desktop (MDK, DS)

Hardware Boards on your Desk

Develop Test cases

Deploy to bespoke hardware



All environments generate
Event Log files for off-line analysis

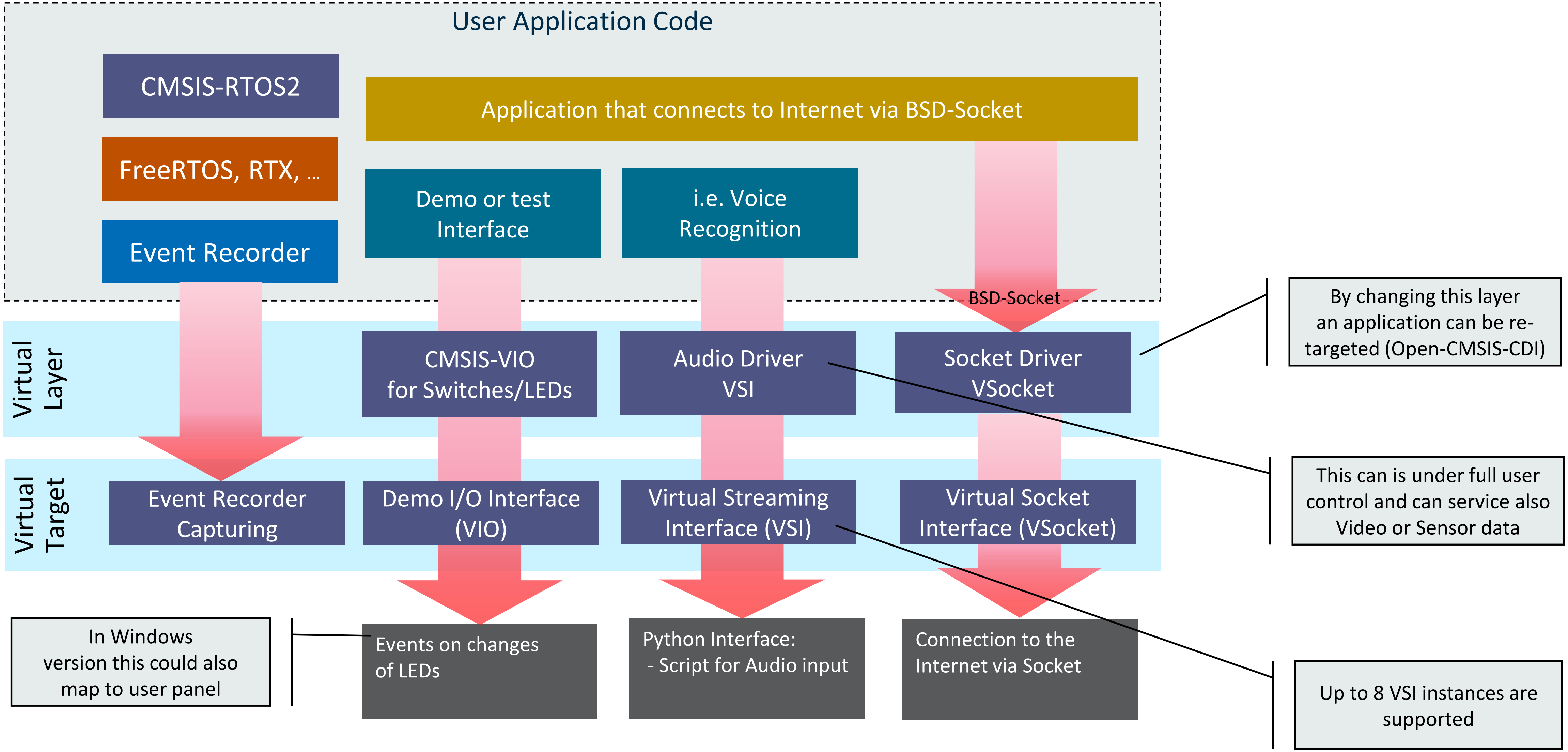
Evaluation Board

MPS3 with
FPGA image

Simulation

Target Hardware

FVP Platform for IoT/DSP/ML Software Development



By changing this layer an application can be re-targeted (Open-CMSIS-CDI)

This can is under full user control and can service also Video or Sensor data

Up to 8 VSI instances are supported

In Windows version this could also map to user panel

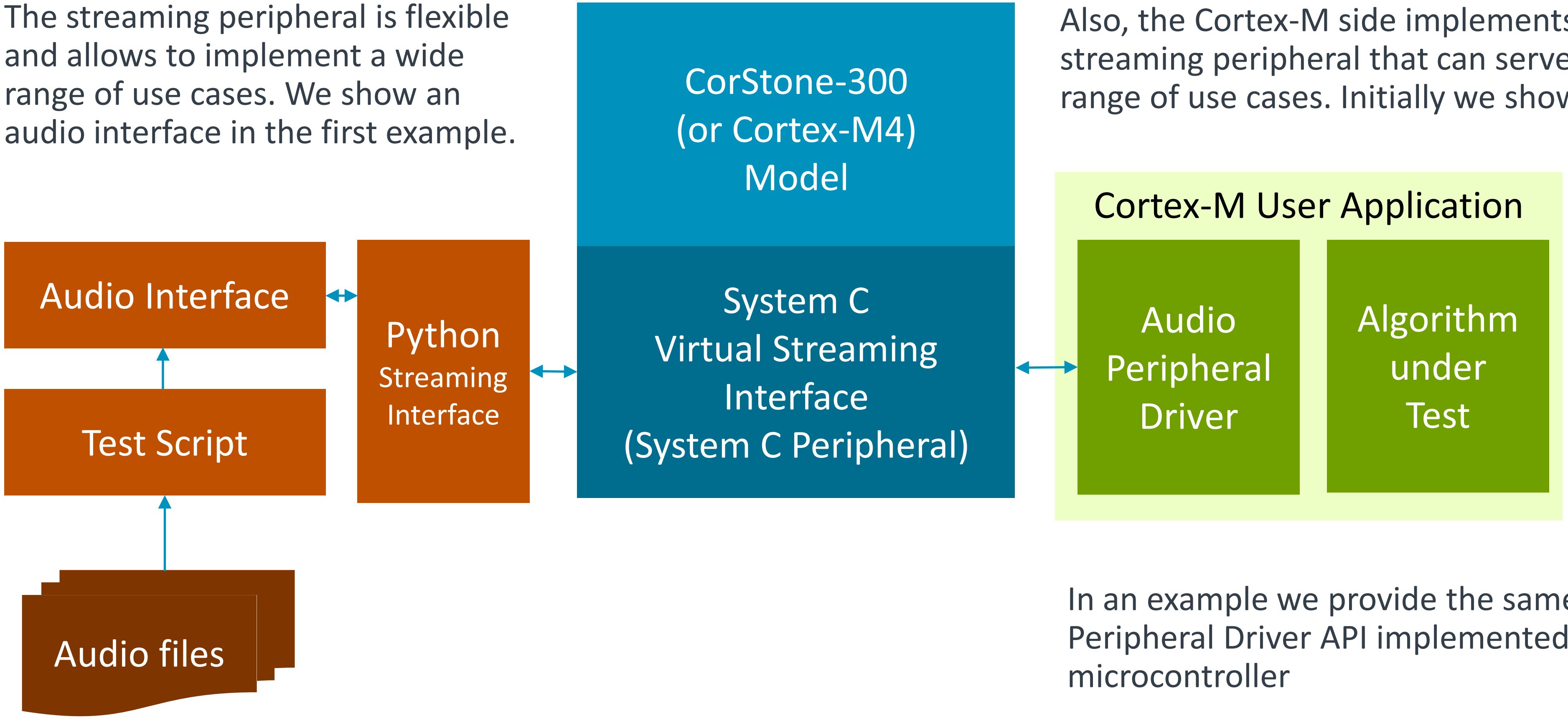
FVP/FM Streaming Peripheral Extension

First PoC implementation of Streaming Interface

FVP Implementation for Linux and Windows

The streaming peripheral is flexible and allows to implement a wide range of use cases. We show an audio interface in the first example.

Also, the Cortex-M side implements a flexible streaming peripheral that can serve a wide range of use cases. Initially we show audio.

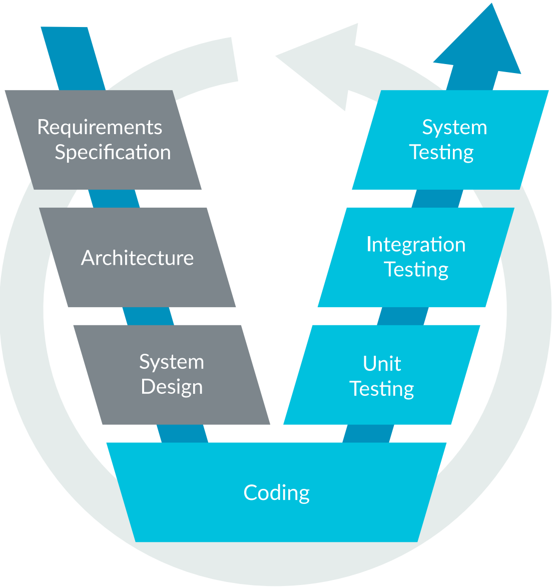


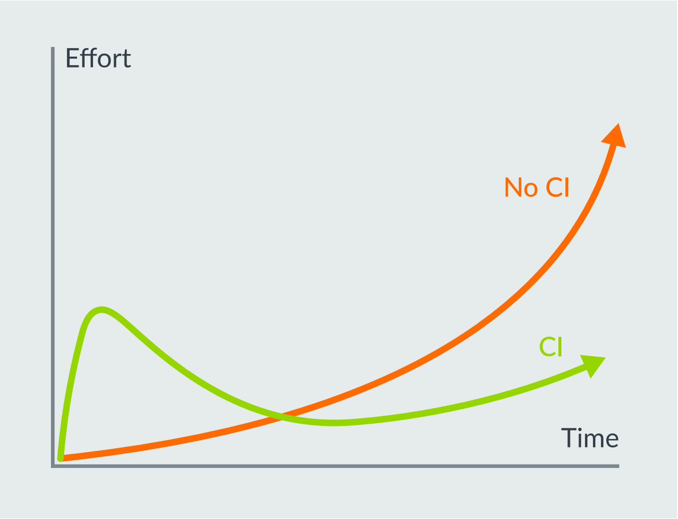
In an example we provide the same Audio Peripheral Driver API implemented on a real microcontroller

Types of Software Testing

Better quality faster, conforming to safety standards

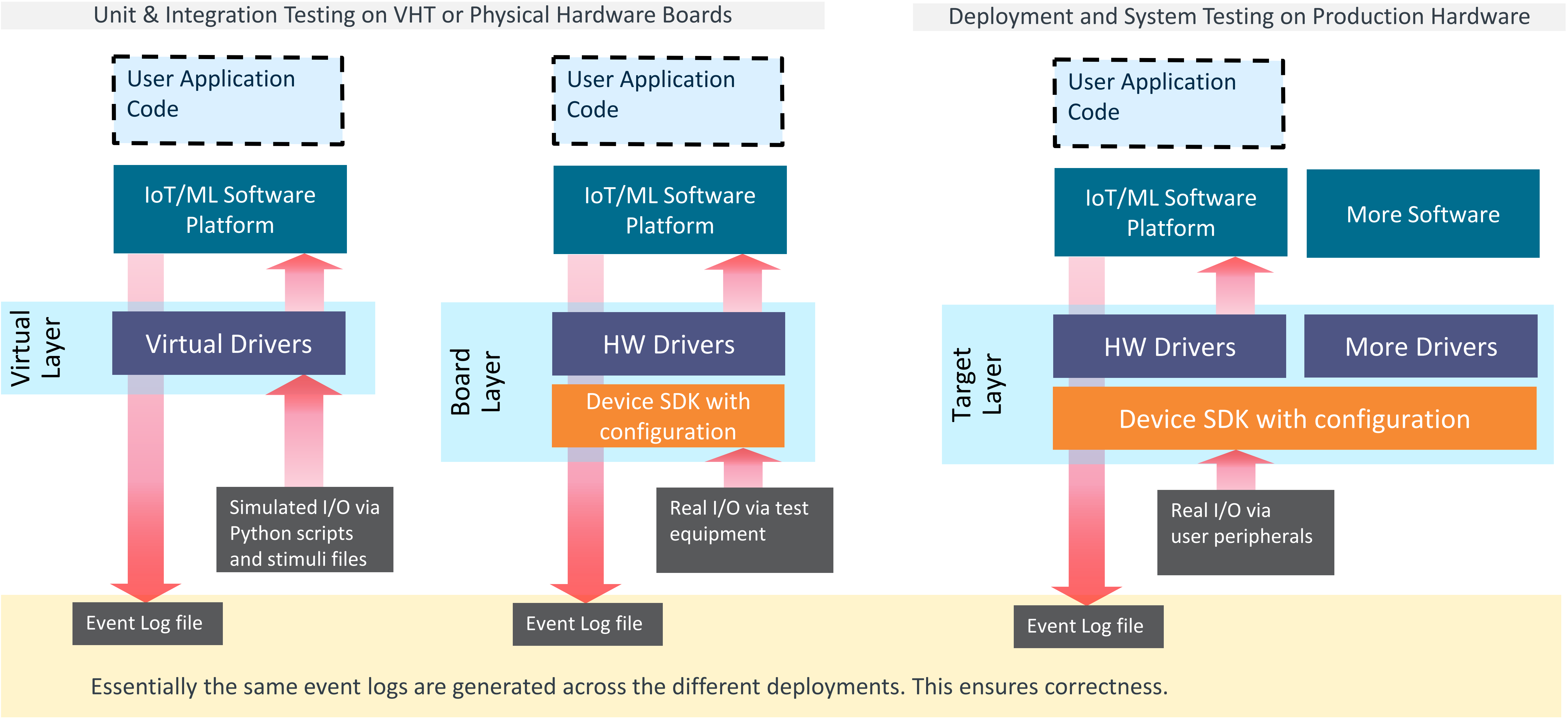
- Unit Testing
  - Test little chunks of code at a time
  - Tested against your ‘test’ build
- Integration Testing
  - Test whether two components work together when they are combined. Verifies that the interface between them works properly
  - Tested against your ‘test’ build
- System (Black-box) Testing
  - Test that final system works as expected. Control external controls & stimuli to system and measure response
  - Tested against your ‘release’ build
- **Regression Testing**
  - **Suite of tests (unit & integration tests) & run continuously upon version control updates.**
  - **Used in Continuous Integration (CI)**





Application Software – from Virtual to Physical Hardware

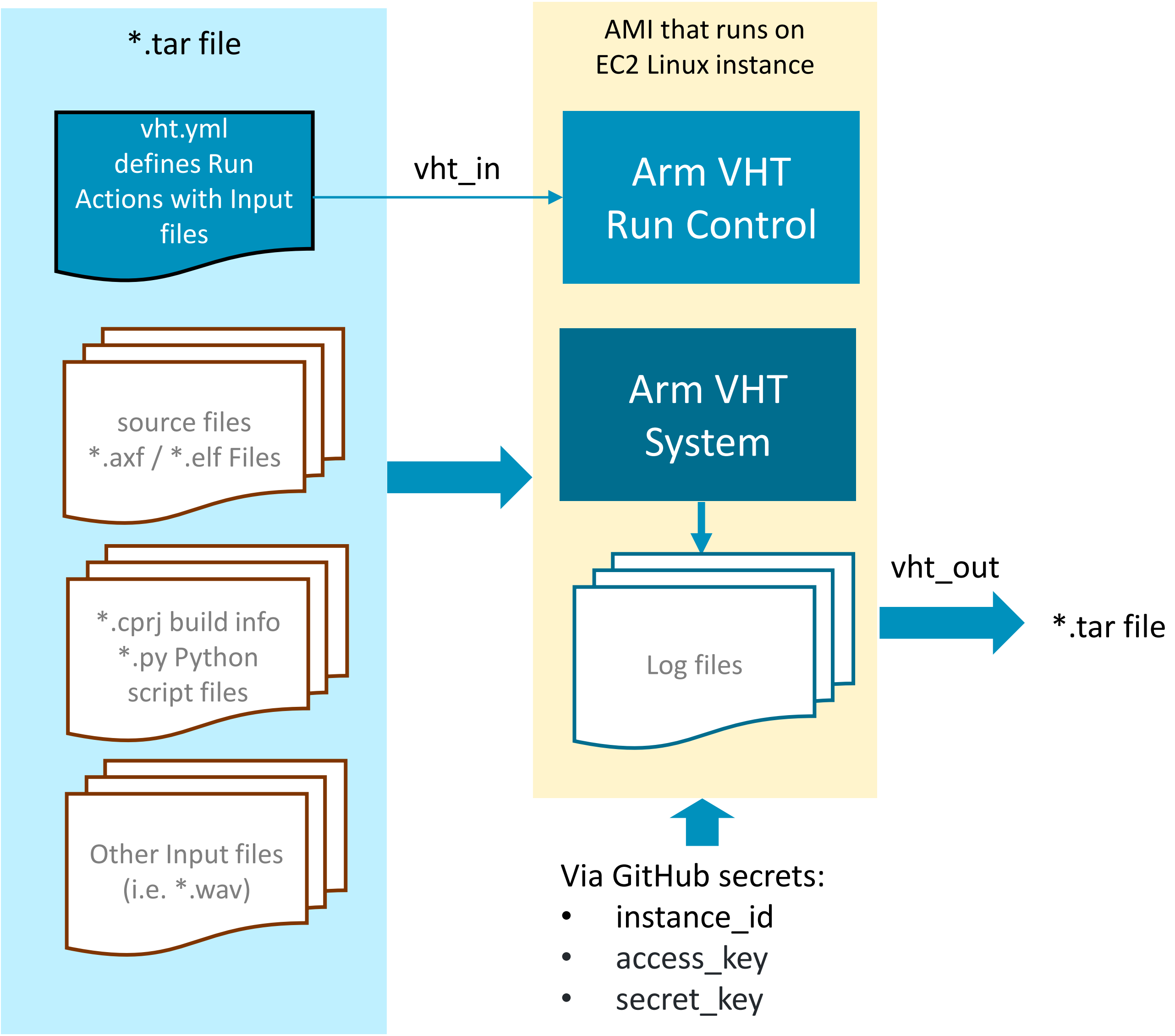
Provide evidence of correctness on Arm Virtual Hardware Target and Physical Hardware



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

Essentially the same event logs are generated across the different deployments. This ensures correctness.

Run VHT on AMI – from GitHub actions



*.tar file

vht.yml defines Run Actions with Input files

source files *.axf / *.elf Files

*.cprj build info *.py Python script files

Other Input files (i.e. *.wav)

vht_in

AMI that runs on EC2 Linux instance

Arm VHT Run Control

Arm VHT System

Log files

vht_out

*.tar file

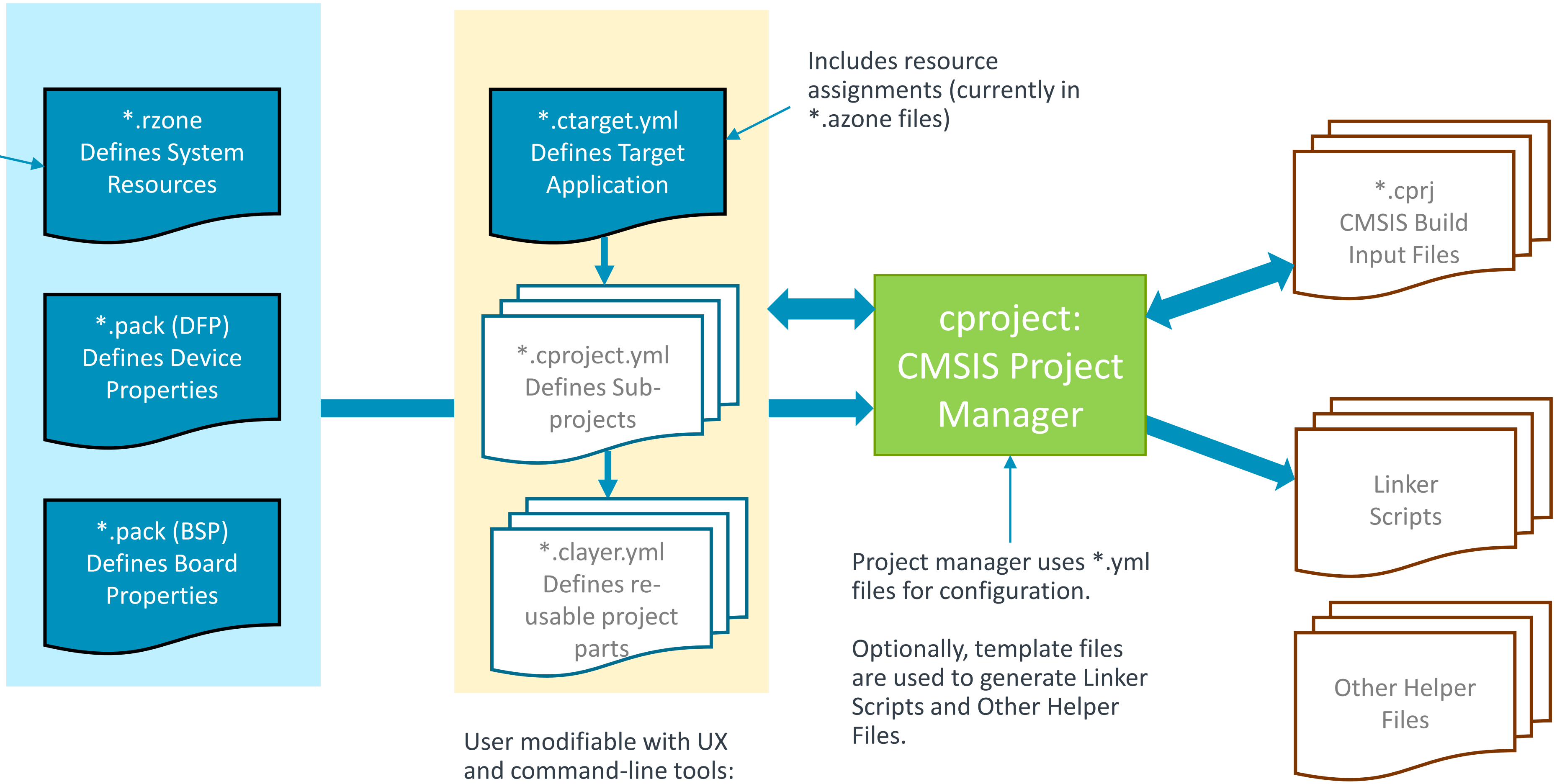
Via GitHub secrets:

- instance_id
- access_key
- secret_key

```
name: 'Run Arm VHT on AMI'
description: 'Run one or more executable files on Arm Virtual Hardware Targets'
inputs:
  vht_in:
    description: 'input tar file with vht.yml commands, executable images, and input scripts'
    required: true
  instance_id:
    description: 'instance id for connection'
    required: true
    default: ''
  secret_key:
    description: 'secret key for connection'
    required: true
    default: ''
  access_key:
    description: 'access key for connection'
    required: true
    default: ''

outputs:
  vht_out:
    description: 'output tar file with log files from Arm VHT execution'
```

cproject: Tool Flow for Project Generation

https://github.com/Open-CMSIS-Pack/Open-CMSIS-Pack/issues/12



If *.rzone does not exist: memory resource information is taken from DFP

Includes resource assignments (currently in *.azone files)

Project manager uses *.yml files for configuration.

Optionally, template files are used to generate Linker Scripts and Other Helper Files.

User modifiable with UX and command-line tools: *.yaml format with schema

Potential *.yml structure (simplified!)

*.ctarget.yml

[board:]   (optional)
device:
vendor:

tool-chain: compiler

options:
 - global CPU options
 - global tool-chain options

projects:
  - references *.cproject.yml files

*.cproject.yml

optional: board, device, vendor,
tool-chain (taken from *.ctarget)

options:
 - CPU i.e. secure/non-secure
 - global tool-chain options

groups:
   files:
     - references to source files

layers:
     - references *.clayer.yml files

components:
     - list of components

*.clayer.yml

groups:
   files:
     - references to source files

components:
     - list of components

Feedback

- Handlebars: https://github.com/Open-CMSIS-Pack/Open-CMSIS-Pack/issues/32
  - Java Run-Time Environment (JRE) seems no longer an issue
  - Gap to FreeMarker is missing evaluation language

Next Steps:

- Add “Component – Tags/Classification” (request from NXP)

- Define Component Naming syntax for referring a component in script files, some early idea
  - Vendor::Cclass:Csub:Cname
  - Cclass:Csub:Cname@version
  - Cclass&bundle:Csub&variant:Cname&variant

- Next Open-CMSIS-Pack meeting: 19. Oct. 2021 @ 16:00

Thank you