	SSE-300 MPS3 BSP Pack User Guide
Version:	1.5.0
Date of Issue:	February 20, 2024

Contents

1 Concept 2
- 1.1 Introduction 2
- 1.2 Prerequisites 2
- 1.3 Documents 3

2 Installing the Pack 4
- 2.1 CMSIS-Toolbox 4
- 2.2 IAR Embedded Workbench 4

3 Blinky example: CMSIS-Toolbox 5
- 3.1 Import 5
- 3.2 Build 5
- 3.3 Run - Terminal 6
- 3.4 Run - FPGA 6

4 Blinky example: Keil Studio 7
- 4.1 Import 7
- 4.2 Build 8
- 4.3 Run - Terminal 8
- 4.4 Run - FPGA 9

5 Blinky example: IAR Embedded Workbench 10
- 5.1 Import 10
- 5.2 Build 11
- 5.3 Run and Debug 11
- 5.4 Run - Terminal 12
- 5.5 Run - FPGA 12

6 Vio example: CMSIS-Toolbox 13
- 6.1 Import 13
- 6.2 Build 13
- 6.3 Run - Terminal 14

7 Vio example: Keil Studio 15
- 7.1 Import 15
- 7.2 Build 16
- 7.3 Run - Terminal 17

8 Vio example: IAR Embedded Workbench 18
- 8.1 Import 18
- 8.2 Build 19
- 8.3 Run and Debug 19
- 8.4 Run - Terminal 20

9 Attachments 21
9.1 Blinky example tree 21
9.2 Vio example tree 21

1 Concept

1.1 Introduction

This document is a general guide to use the SSE-300 MPS3 BSP pack. The CMSIS pack is to be used with the Corstone®-300 platform MPS3 FVP model or AN552 FPGA (AN552: Arm Corstone™ SSE-300 with Cortex®-M55 and Ethos™-U55: Example Subsystem for MPS3). The pack contains necessary source files, a linker script file, and a specification document to kick-start development for the Corstone-300 MPS3 platform, and a reference secure side Blinky example to enable a user to understand uVision and IAR project configuration. The pack also provides a System View Description (SVD) file for the platform to be used with the uVision and IAR debugger. This document specifies system prerequisites and explains how to build and run the reference Blinky example on the SSE-300 MPS3 FVP model and on the AN552 FPGA.

Terms and Abbreviations:

Terms	Meaning
BSP	Board Support Pack
FVP	Fixed Virtual Platform
FPGA	Field Programmable Gate Array
VIO	Virtual I/O

1.2 Prerequisites

- **Developement Environments:**
  CMSIS-Toolbox v2.2.1 or IAR Embedded Workbench v9.50,
- **Compiler:**
  Arm Compiler for Embedded (Version 6.20 or newer),
  GNU Arm Embedded Toolchain Arm GCC (Version 11.3 or newer),
  LLVM Embedded Toolchain for Arm (Version 17.0.1 or newer),
  IAR C/C++ Compiler for Arm (Version 9.50.1 or newer),
- **FVP model:**
  Corstone SSE-300 MPS3 FVP model,
- **MPS3 FPGA:**
  AN552: Arm Corstone™ SSE-300 with Cortex®-M55 and Ethos™-U55 Example Subsystem for MPS3 FPGA,
- **Package manager:**
  vcpkg.
- **Python version 3.9 for VIO examples**

Download **Python3.9** like the following on **Ubuntu** if the version at **apt** is not working:

```
$ sudo apt-get update && sudo apt-get upgrade
$ sudo apt-get install -y make build-essential libssl-dev zlib1g-dev libbz2-dev
libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev libncursesw5-dev
xz-utils tk-dev liblzma-dev tk-dev libffi-dev

$ wget https://www.python.org/ftp/python/3.9.18/Python-3.9.18.tgz
$ tar xzf Python-3.9.18.tgz && cd Python-3.9.18

$ ./configure --prefix=/opt/python/3.9.18/ --enable-optimizations
$ make -j "$(nproc)"
$ sudo make altinstall

$ export PYTHONHOME=/opt/python/3.9.18
```

1.3 Documents

1. Application Note AN552: AN552 is a Single Cortex-M55 FPGA implementation of the Arm Corstone SSE-300 with Cortex-M55 and Ethos™-U55 Example Subsystem that uses SIE-300, SIE-200 components. The application note states which components are included, the memory map of AN552, details about the FPGA peripherals.
2. Arm Corstone-300 Foundation IP Technical Overview: contains overview of the IP and its features.
3. Arm Corstone SSE-300 Subsystem Technical Reference Manual: contains the specification of the architecture of the subsystem, description of several interfaces (address, data width, clock/power/reset domain), functional description of the components.

SSE-300 MPS3 BSP Pack contains additional documentation in the "Documents" folder.

---

2 Installing the Pack

2.1 CMSIS-Toolbox

To install the `ARM::V2M_MPS3_SSE_300_BSP` pack in command-line, use:

```
$ cpackget add ARM.V2M_MPS3_SSE_300_BSP
```

This will install the prerequired `ARM.CMSIS.6.0.0` and `ARM.CMSIS-Compiler.2.0.0` packages as well.

2.2 IAR Embedded Workbench

Install `ARM::V2M_MPS3_SSE_300_BSP` using the CMSIS-Pack Manager. `(Find: Project > CMSIS-Pack Manager)` The pack can be browsed by selecting SSE-300-MPS3 device under ARM Cortex M55 Devices.

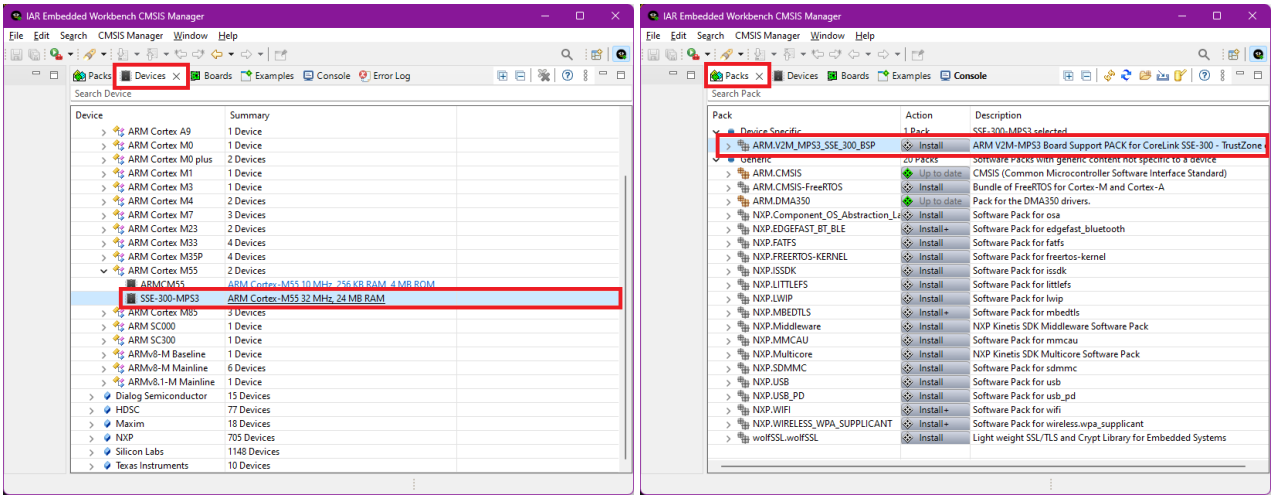

Figure 1: Selecting `ARM::V2M_MPS3_SSE_300_BSP` using IAR CMSIS-Pack Manager

3 Blinky example: CMSIS-Toolbox

You can see the example's folder structure under Blinky example tree chapter. The following example will assume the usage of a linux machine, therefore some commands might differ on Windows and Mac.
As the example contains a `Makefile`, you can use that as well, but we will go through in every configuration without it as well.

3.1 Import

Copy the containing folder from the CMSIS Pack to your working directory, and provide it with the necessary permissions:

```
$ cp -r ${CMSIS_PACK_ROOT}/ARM/V2M_MPS3_SSE_300_BSP/1.5.0/Examples/Blinky blinky-example
$ chmod -R +w blinky-example
$ cd blinky-example
```

Then install the project dependencies:

```
$ vcpkg activate
```

Read more about vcpkg

3.2 Build

The examples support the `[AC6, GCC, IAR, CLANG]` compilers, which are represented as `[armclang, gcc, iar, clang]` in the `Makefile`. Let's make a build with `AC6` toolchain:

```
$ cbuild Blinky.csolution.yml --packs --update-rte --jobs $(nproc) --context .Debug --toolchain AC6
```

or

```
$ make armclang
```

If you just want to generate the `.cprj` project files, without building the whole project, you can use `csolution`.

```
$ csolution convert Blinky.csolution.yml --context .[Debug, Release] --toolchain
[AC6, GCC, IAR, CLANG]
```

Used tags while building:

Tag	Mandatory	Meaning
—packs	No	Downloading packs required for the pack.
—update-rte	No	Overwrite RTE folder content from pack.
—jobs [n]	No	Build on `[n]` threads.
—toolchain	Yes	Compiler Toolchain. If missing builds for all 4 compilers.
—context	Yes	Selecting the build type or board. Select `.Debug` or `.Release`

See the help and documentation of `cbuild` for further help.

Binary:

Find the binary files in the following location:

```
build/{platform}/{toolchain}/{type}/{example}/outdir/Blinky.[axf,elf,out]
```

as

```
build/MPS3/AC6/Debug/Blinky/outdir/Blinky.axf
```

3.3 Run - Terminal

After building the target, you can launch the FVP, using the command:

```
$ <path_to_fvp>/FVP_Corstone_SSE-300_Ethos-U55 -a build/MPS3/AC6/Debug/Blinky/outdir/Blinky.axf
```

or, after setting the model path `<path_to_fvp>` in the `Makefile`:

```
$ make run-armclang
```

3.4 Run - FPGA

If you translated with **`make`**, then you should already have binary files in the **`bins/`** folder. To translate the output elf file to binary, run the following command for the specific compiler:

Compiler	Command	Arguments
armclang	`fromelf`	`--bin bins/Blinky.axf -o bins/Blinky_armclang.bin`
gcc	`arm-none-eabi-objcopy`	`-O binary bins/Blinky.elf bins/Blinky_gcc.bin`
clang	`llvm-objcopy`	`-O binary bins/Blinky.clang.elf bins/Blinky_clang.bin`
iar	`ielftool`	`--bin bins/Blinky.out bins/Blinky_iar.bin`

Copy the binary to the FPGA's SD card `(x:/SOFTWARE)` and set the address in images.txt `(x:/MB/HBI0309C/AN552/images.txt)` to `0x00000000`, then restart the FPGA.

```
IMAGE0ADDRESS: 0x00000000 ;
IMAGE0UPDATE: AUTO ;
IMAGE0FILE: /SOFTWARE/Blinky.bin ;
```

After uploading the example code, restart the FPGA and see the output on the LEDs and the serial output.

4 Blinky example: Keil Studio

You can see the example's folder structure under Blinky example tree chapter.

To Use Keil Studio, install the Keil Studio Pack in **Visual Studio Code**

4.1 Import

Copy the containing folder from the CMSIS Pack:

```
$ cp -r ${CMSIS_PACK_ROOT}/ARM/V2M_MPS3_SSE_300_BSP/1.5.0/Examples/Blinky blinky-example
$ chmod -R +w blinky-example
$ cd blinky-example
```

Then open your working directory in **Visual Studio Code**.

Or open from Visual Studio Code:

Go to CMSIS, then create a solution with `SSE-300-MPS3` selected as the following:

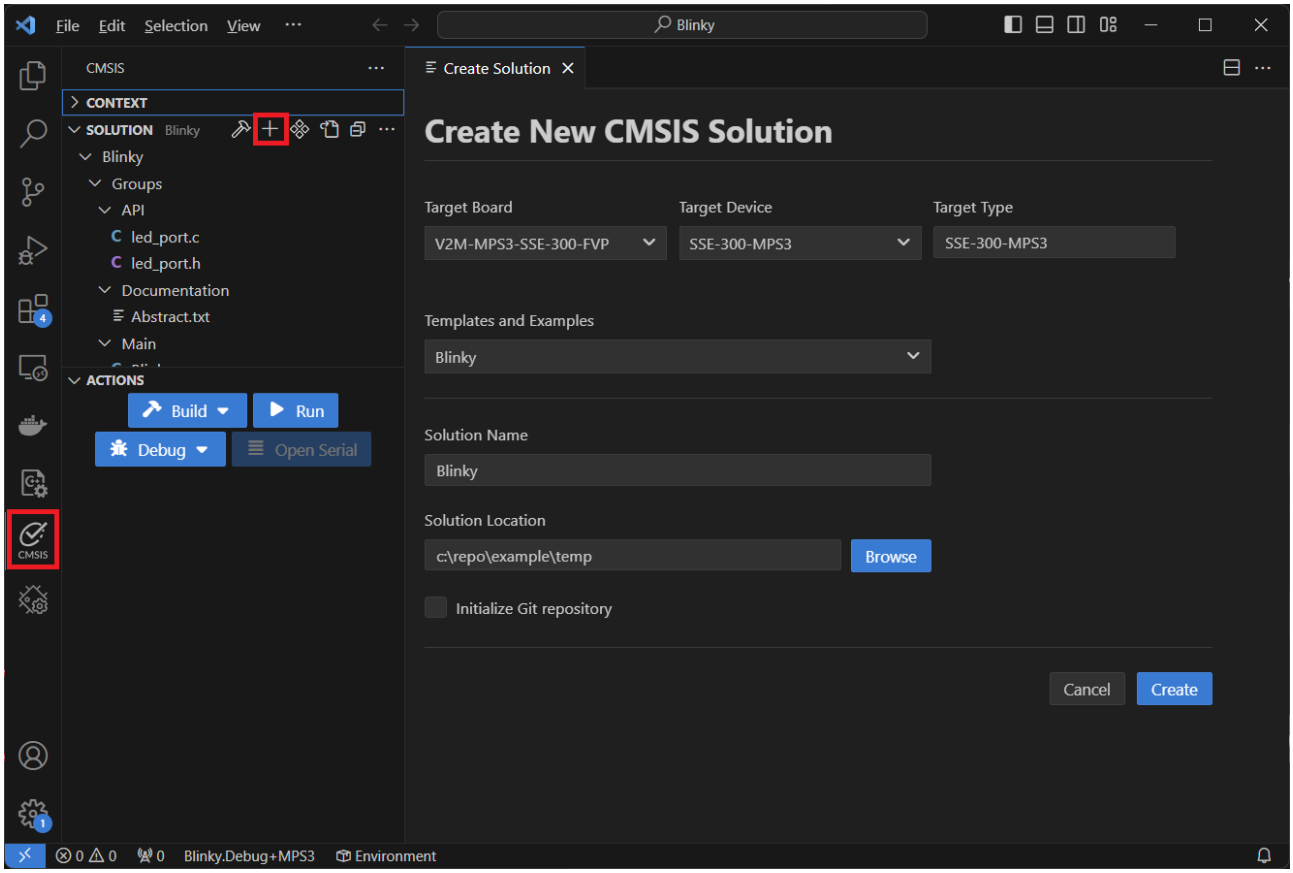

Figure 2: Keil Studio: Importing Blinky Project

4.2 Build

To build the example, open the **CMSIS** tab on the left side of **Visual Studio Code**, then click on **Build**.

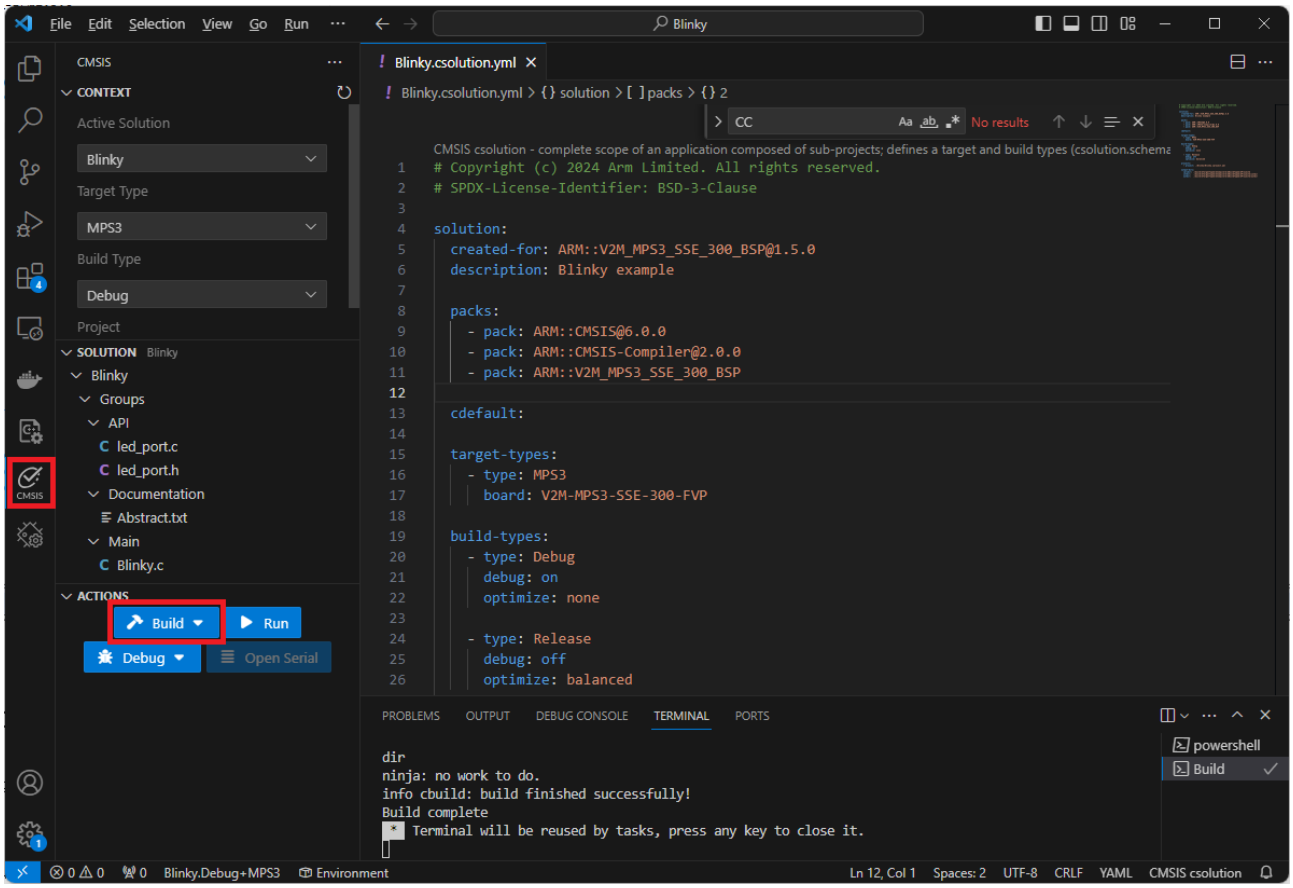

Figure 3: Keil Studio: Building Project

Set compiler:
By default the project is built by `GCC` toolchain.
To build with other toolchain, add `'compiler: [AC6, GCC, IAR, CLANG]'` to `Blinky.csolution.yml` under `solution:` tag, like:

```
12.
13.  compiler:  AC6
14.  cdefault:
15.
```

Binary:
Find the binary files in the following location:

`build/{platform}/{toolchain}/{type}/{example}/outdir/Blinky.[axf,elf,out]`
as
`build/MPS3/AC6/Debug/Blinky/outdir/Blinky.axf`

4.3 Run - Terminal

After building the target, you can launch the FVP, using the command:

```
$ <path_to_fvp>/FVP_Corstone_SSE-300_Ethos-U55 -a build/MPS3/AC6/Debug/Blinky/outdir/Blinky.axf
```

4.4 Run - FPGA

To translate the output elf file to binary, run the following command for the specific compiler:

Compiler	Command	Arguments
armclang	`fromelf`	`--bin bins/Blinky.axf -o bins/Blinky_armclang.bin`
gcc	`arm-none-eabi-objcopy`	`-O binary bins/Blinky.elf bins/Blinky_gcc.bin`
clang	`llvm-objcopy`	`-O binary bins/Blinky.clang.elf bins/Blinky_clang.bin`
iar	`ielftool`	`--bin bins/Blinky.out bins/Blinky_iar.bin`

Copy the binary to the FPGA's SD card `(x:/SOFTWARE)` and set the address in images.txt `(x:/MB/HBI0309C/AN552/images.txt)` to `0x00000000`, then restart the FPGA.

```
IMAGE0ADDRESS: 0x00000000 ;
IMAGE0UPDATE: AUTO ;
IMAGE0FILE: /SOFTWARE/Blinky.bin ;
```

After uploading the example code, restart the FPGA and see the output on the LEDs and the serial output.

5 Blinky example: IAR Embedded Workbench

You can see the example's folder structure under Blinky example tree chapter.

5.1 Import

To import the example, first you need to copy the project files, as shown in section 3. After copying the project, import it the following way:

1. Create a new workspace, `(File > New Workspace)`
2. Open Create a new project window for importing `csolution` projects. `(Project > Create New Project...)`
3. Select tool chain: `CMake for Arm`
4. Select project template: `Import csolution.yml`
5. Click `Next`, navigate to the project folder and select the `Blinky.csolution.yml` file.

Before building

1. add `'compiler: IAR'` to `Blinky.csolution.yml` under `solution:` tag.

```
12.
13.  compiler: IAR
14.  cdefault:
15.
```

2. set CMake and CMSIS-Toolbox binary path.
   The minimum version that supports IAR is 2.2.1.
   Go to `(Tools > Options... > CMake/CMSIS-Toolbox)`.

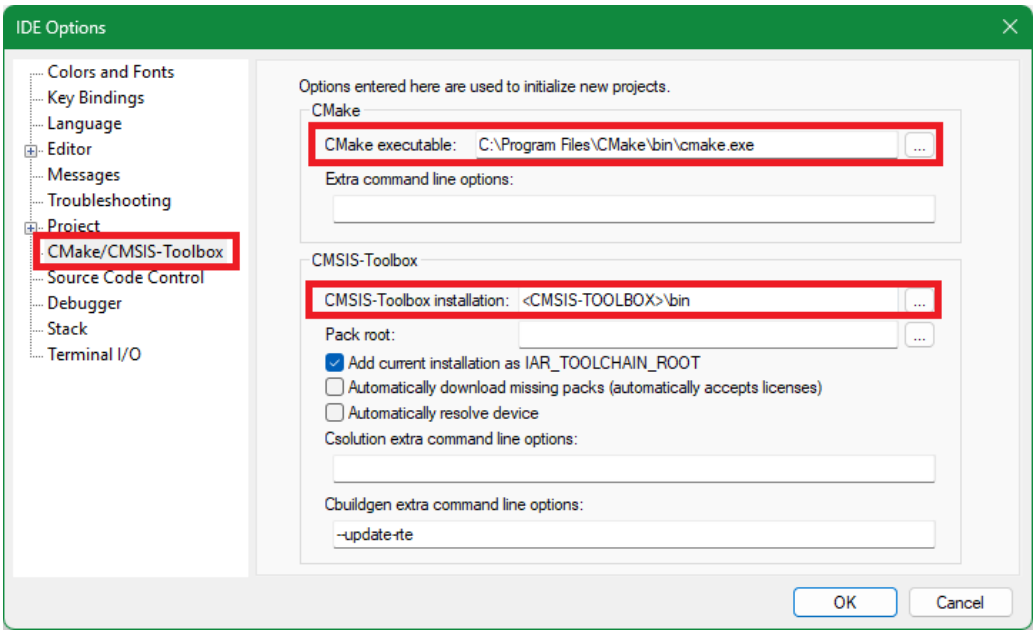

Figure 4: IAR EW: Set CMSIS-Toolbox path

If you encounter issues while generating the project files, first make sure that you are using the correct CMSIS-Toolbox version! Minimum **2.2.1**.

Please refer to the CMSIS-Toolbox Installation Guide or install it with `$ vcpkg activate`.

5.2 Build

To build the example click on the "Make" button, or press "F7".

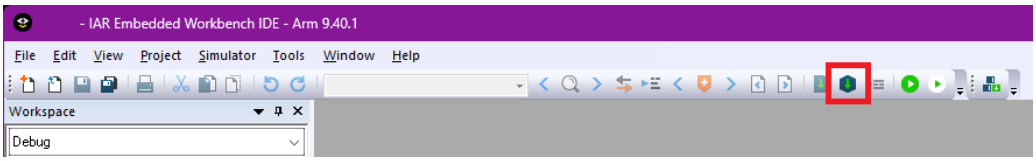

Figure 5: IAR EW: Building the Blinky example

5.3 Run and Debug

This section explains how to run the Blinky example on the Corstone SSE-300 FVP model. First, download and install the SSE-300 FVP from the link provided in the Prerequisites section.
To debug the example inside the IAR Embedded Workbench software, follow the steps below.

Start up the FVP with a CADI server:

```
<path_to_fvp>/FVP_Corstone_SSE-300_Ethos-U55.exe -S
```

Set up the debugger inside IAR:

Open the project Options, by clicking on the root file in the Workspace and then selecting it from the Project menu item. `(Project > Options)`
Select the Debugger in the Category list and select CADI as Driver on the Setup tab.

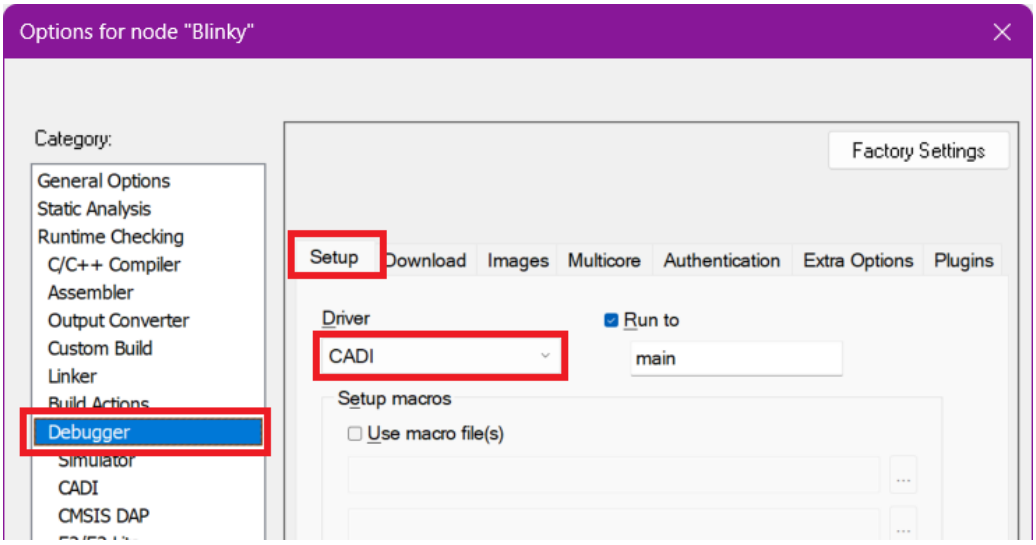

Figure 6: IAR EW: Selecting Debugger

You can select the desired CADI server in the CADI Category if there are more than one you wish to use. Otherwise, it can be left empty.

To run the debugger click on the "Download and Debug" button. The application will be automatically loaded to the model by the debugger.

5.4 Run - Terminal

After building the target, you can launch the FVP from the terminal, using the command:
`<path_to_fvp>/FVP_Corstone_SSE-300_Ethos-U55.exe <path_to_bin>/Blinky.out`

5.5 Run - FPGA

To translate the output elf file to binary, run the following command for the specific compiler:

Compiler	Command	Arguments
armclang	`fromelf`	`--bin bins/Blinky.axf -o bins/Blinky_armclang.bin`
gcc	`arm-none-eabi-objcopy`	`-O binary bins/Blinky.elf bins/Blinky_gcc.bin`
clang	`llvm-objcopy`	`-O binary bins/Blinky.clang.elf bins/Blinky_clang.bin`
iar	`ielftool`	`--bin bins/Blinky.out bins/Blinky_iar.bin`

Copy the binary to the FPGA's SD card `(x:/SOFTWARE)` and set the address in images.txt `(x:/MB/HBI0309C/AN552/images.txt)` to `0x00000000`, then restart the FPGA.

```
IMAGE0ADDRESS: 0x00000000 ;
IMAGE0UPDATE: AUTO ;
IMAGE0FILE: /SOFTWARE/Blinky.bin ;
```

To debug the project:

1. Select the "CMSIS DAP" Driver, and connect to the running example afterwards.
   `(Project > Options > Debugger > Setup > Driver)`
2. Click on "Run and Debug".

Caution

If you have are using a revision **B** board, use the **HBI0309B** directory. The examples are only verified on the revision **C** boards.

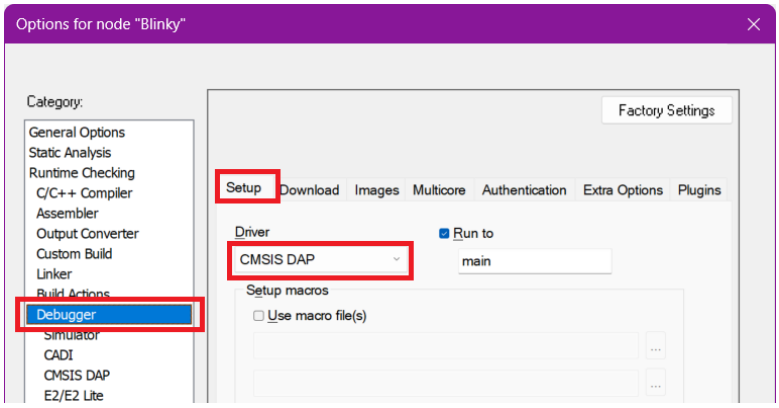

Figure 7: IAR EW: Select "CMSIS DAP" Debugger for MPS3 FPGA

6 Vio example: CMSIS-Toolbox

You can see the example's folder structure under Vio example tree chapter. The following example will assume the usage of a linux machine, therefore some commands might differ on Windows and Mac.
As the example contains a `Makefile`, you can use that as well, but we will go through in every configuration without it as well.

6.1 Import

Copy the containing folder from the CMSIS Pack to your working directory, and provide it with the necessary permissions:

```
$ cp -r ${CMSIS_PACK_ROOT}/ARM/V2M_MPS3_SSE_300_BSP/1.5.0/Examples/Vio vio-example
$ chmod -R +w vio-example
$ cd vio-example
```

Then install the project dependencies:
`$ vcpkg activate`
Read more about vcpkg

6.2 Build

The examples support the `[AC6, GCC, IAR, CLANG]` compilers, which are represented as `[armclang, gcc, iar, clang]` in the `Makefile`. Let's make a build with `AC6` toolchain:

```
$ cbuild Vio.csolution.yml --packs --update-rte --jobs $(nproc) --context .Debug --toolchain AC6
```
or
```
$ make armclang
```

After building, make sure to provide a minimum of **`0x600`** memory to stack allocation.

In `Vio/RTE/Device/SSE_300_MPS3/regions_V2M_MPS3_SSE_300_FVP.h` modify
`#define __STACK_SIZE 0x00000200` to
`#define __STACK_SIZE 0x00000600`.
Then rebuild without `--update-rte`.

Used tags while building:

Tag	Mandatory	Meaning
—packs	No	Downloading packs required for the pack.
—update-rte	No	Overwrite RTE folder content from pack.
—jobs [n]	No	Build on `[n]` threads.
—toolchain	Yes	Compiler Toolchain. If missing builds for all 4 compilers.
—context	Yes	Selecting the build type or board. Select `.Debug` or `.Release`

See the help and documentation of `cbuild` for further help.

If you just want to generate the `.cprj` project files, without building the whole project, you can use `csolution`.
`$ csolution convert Blinky.csolution.yml --context .[Debug, Release] --toolchain [AC6, GCC, IAR, CLANG]`

Binary:
Find the binary files in the following location:

`build/{platform}/{toolchain}/{type}/{example}/outdir/Vio.[axf,elf,out]`
as
`build/MPS3/AC6/Debug/Vio/outdir/Vio.axf`

6.3 Run - Terminal

Download **arm_vio.py**, like the following:
On Windows:
```
PS > wget https://github.com/ARM-software/AVH/raw/main/interface/python/arm_vio.py -OutFile arm_vio.py
```
On Linux:
```
$ wget https://github.com/ARM-software/AVH/raw/main/interface/python/arm_vio.py -o arm_vio.py
```

Then set `verbosity = logging.DEBUG` in it.

```
15
16  ## Set verbosity level
17  verbosity = logging.DEBUG
18  #verbosity = logging.ERROR
19
```

Figure 8: Setting debug mode in `arm_vio.py`

Set environmental variables:
On Windows:
```
PS > $env:PYTHONHOME = 'C:/Users/<user>/AppData/Local/Programs/Python/Python39'
```
On Linux:
```
$ export PYTHONHOME=/opt/python/3.9.18
```

After building the target, you can launch the FVP, using the command:
```
$ <path_to_fvp>/FVP_Corstone_SSE-300_Ethos-U55 -a build/MPS3/AC6/Debug/Vio/outdir/Vio.axf
  -C mps3_board.v_path=<path_to_vio_py_dir>
```

or, after setting the model path `<path_to_fvp>` and `<path_to_vio_py_dir>` in the `Makefile`:
```
$ make run-armclang
```

Remember to use the path of the containing folder of `arm_vio.py` when specifying path `<path_to_vio_py_dir>`.

7 Vio example: Keil Studio

You can see the example's folder structure under Vio example tree chapter.

To Use Keil Studio, install the Keil Studio Pack in **Visual Studio Code**

7.1 Import

Copy the containing folder from the CMSIS Pack:

```
$ cp -r ${CMSIS_PACK_ROOT}/ARM/V2M_MPS3_SSE_300_BSP/1.5.0/Examples/Vio vio-example
$ chmod -R +w vio-example
$ cd vio-example
```

Then open your working directory in **Visual Studio Code**.

Or open from Visual Studio Code:

Go to CMSIS, then create a solution with `SSE-300-MPS3` selected as the following:

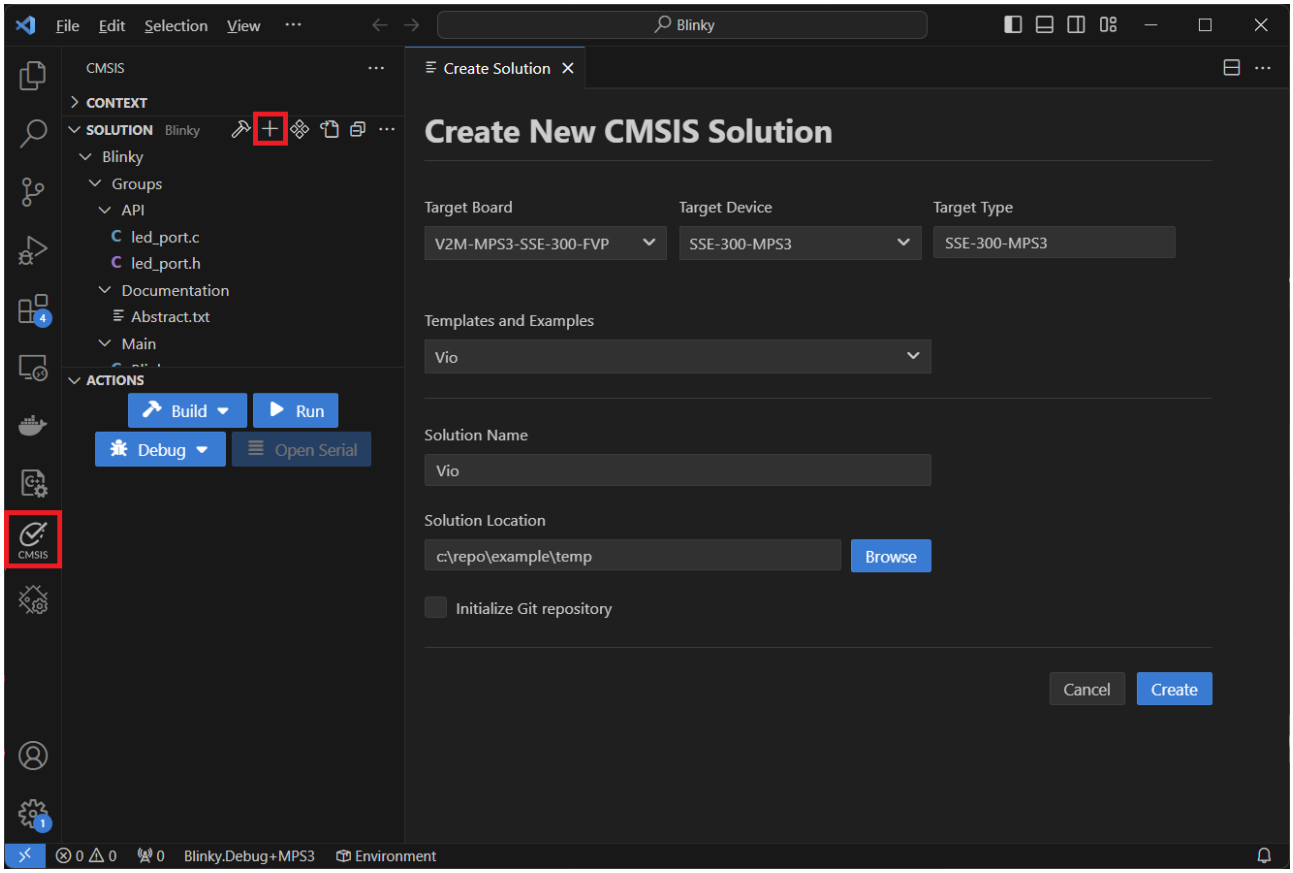

Figure 9: Keil Studio: Importing Vio Project

7.2 Build

To build the example, open the **CMSIS** tab on the left side of **Visual Studio Code**, then click on **Build**.

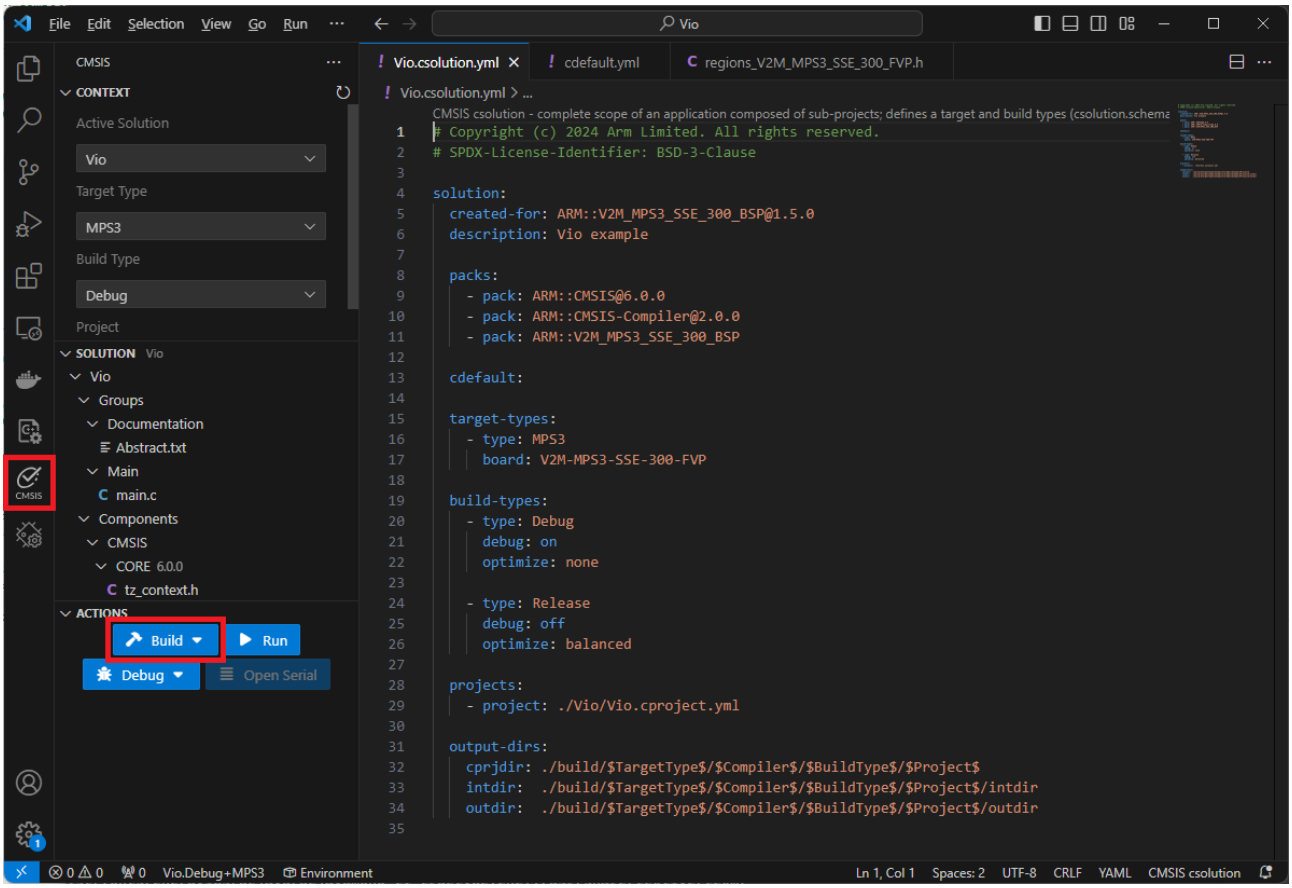

Figure 10: Keil Studio: Building Project

Caution

After building, make sure to provide a minimum of **0x600** memory to stack allocation.

In `Vio/RTE/Device/SSE_300_MPS3/regions_V2M_MPS3_SSE_300_FVP.h` modify
`#define __STACK_SIZE 0x00000200` to
`#define __STACK_SIZE 0x00000600`.
Then rebuild.

Set compiler:
By default the project is built by `GCC` toolchain.
To build with other toolchain, add `'compiler: [AC6, GCC, IAR, CLANG]'` to `Vio.csolution.yml` under `solution:` tag, like:

```
12.
13.  compiler:  AC6
14.  cdefault:
15.
```

Binary:
Find the binary files in the following location:

`build/{platform}/{toolchain}/{type}/{example}/outdir/Vio.[axf,elf,out]`
as
`build/MPS3/AC6/Debug/Blinky/outdir/Vio.axf`

7.3 Run - Terminal

Download **arm_vio.py**, like the following:

On Windows:

```
PS > wget https://github.com/ARM-software/AVH/raw/main/interface/python/arm_vio.py -OutFile arm_vio.py
```

On Linux:

```
$ wget https://github.com/ARM-software/AVH/raw/main/interface/python/arm_vio.py -o arm_vio.py
```

Then set `verbosity = logging.DEBUG` in it.

```
15
16    ## Set verbosity level
17    verbosity = logging.DEBUG
18    #verbosity = logging.ERROR
19
```

Figure 11: Setting debug mode in `arm_vio.py`

Set environmental variables:

On Windows:

```
PS > $env:PYTHONHOME = 'C:/Users/<user>/AppData/Local/Programs/Python/Python39'
```

On Linux:

```
$ export PYTHONHOME=/opt/python/3.9.18
```

After building the target, you can launch the FVP, using the command:

```
$ <path_to_fvp>/FVP_Corstone_SSE-300_Ethos-U55 -a build/MPS3/AC6/Debug/Vio/outdir/Vio.axf
  -C mps3_board.v_path=<path_to_vio_py_dir>
```

or, after setting the model path `<path_to_fvp>` and `<path_to_vio_py_dir>` in the `Makefile`:

```
$ make run-armclang
```

Remember to use the path of the containing folder of `arm_vio.py` when specifying path `<path_to_vio_py_dir>`.

8 Vio example: IAR Embedded Workbench

You can see the example's folder structure under Vio example tree chapter.

8.1 Import

To import the example, first you need to copy the project files, as shown in section 3. After copying the project, import it the following way:

1. Create a new workspace, `(File > New Workspace)`
2. Open Create a new project window for importing `csolution` projects. `(Project > Create New Project...)`
3. Select tool chain: `CMake for Arm`
4. Select project template: `Import csolution.yml`
5. Click `Next`, navigate to the project folder and select the `Vio.csolution.yml` file.

Before building

1. add `'compiler: IAR'` to `Vio.csolution.yml` under `solution:` tag.

```
12.
13.  compiler: IAR
14.  cdefault:
15.
```

2. set CMake and CMSIS-Toolbox binary path.
   The minimum version that supports IAR is 2.2.1.
   Go to `(Tools > Options... > CMake/CMSIS-Toolbox)`.

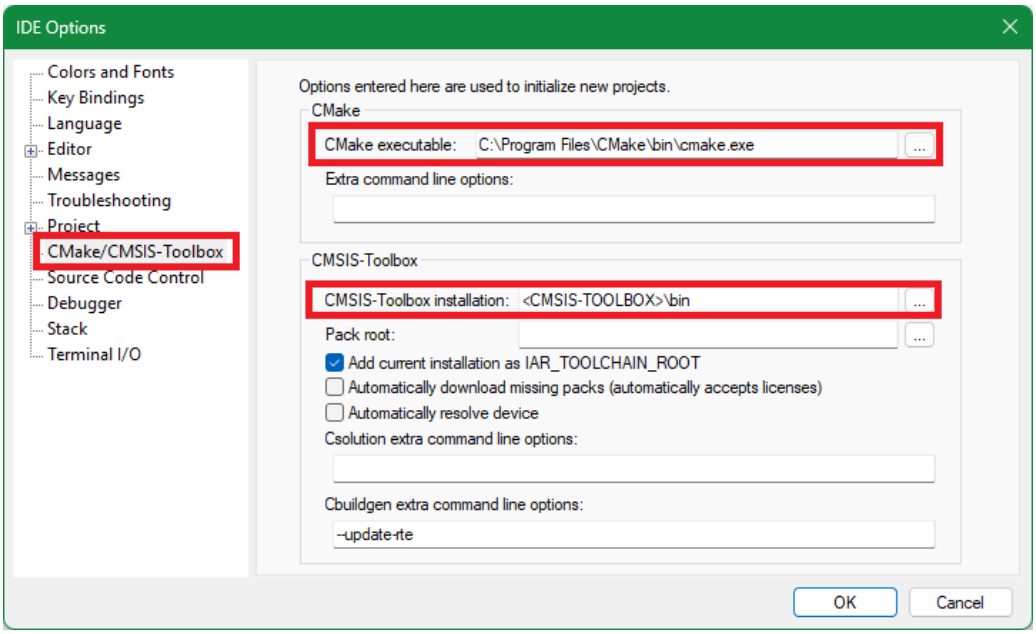

Figure 12: IAR EW: Set CMSIS-Toolbox path

If you encounter issues while generating the project files, first make sure that you are using the correct CMSIS-Toolbox version! Minimum **2.2.1**.

Please refer to the CMSIS-Toolbox Installation Guide or install it with `$ vcpkg activate`.

8.2 Build

To build the example click on the "Make" button, or press "F7".

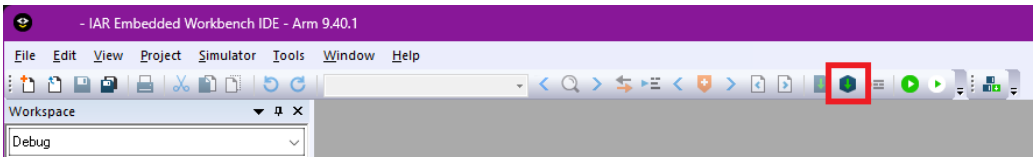

Figure 13: IAR EW: Building the Vio example

After building, make sure to provide a minimum of **0x600** memory to stack allocation.

In `Vio/RTE/Device/SSE_300_MPS3/regions_V2M_MPS3_SSE_300_FVP.h` modify
`#define __STACK_SIZE 0x00000200` to
`#define __STACK_SIZE 0x00000600`.
Then rebuild.

8.3 Run and Debug

This section explains how to run the Vio example on the Corstone SSE-300 FVP model. First, download and install the SSE-300 FVP from the link provided in the Prerequisites section.

The VIO example might not work with newer Python versions. To run the example use version Python3.9.x.

To debug the example inside the IAR Embedded Workbench software, follow the steps below.
Download **arm_vio.py**, like the following:

```
PS > wget https://github.com/ARM-software/AVH/raw/main/interface/python/arm_vio.py -OutFile arm_vio.py
```

Then uncomment the `#verbosity = logging.DEBUG` line in it.

Set environmental variables:

```
PS > $env:PYTHONHOME = 'C:/Users/<user>/AppData/Local/Programs/Python/Python39'
```

Start up the FVP with a CADI server:

```
<path_to_fvp>/FVP_Corstone_SSE-300_Ethos-U55.exe -S -C mps3_board.v_path=<path_to_arm_vio.py_folder>
```

Configure the debugger inside IAR:

Open the project Options, by clicking on the root file in the Workspace and then selecting it from the Project menu item. `(Project > Options)`

Select the Debugger in the Category list and select CADI as Driver on the Setup tab.

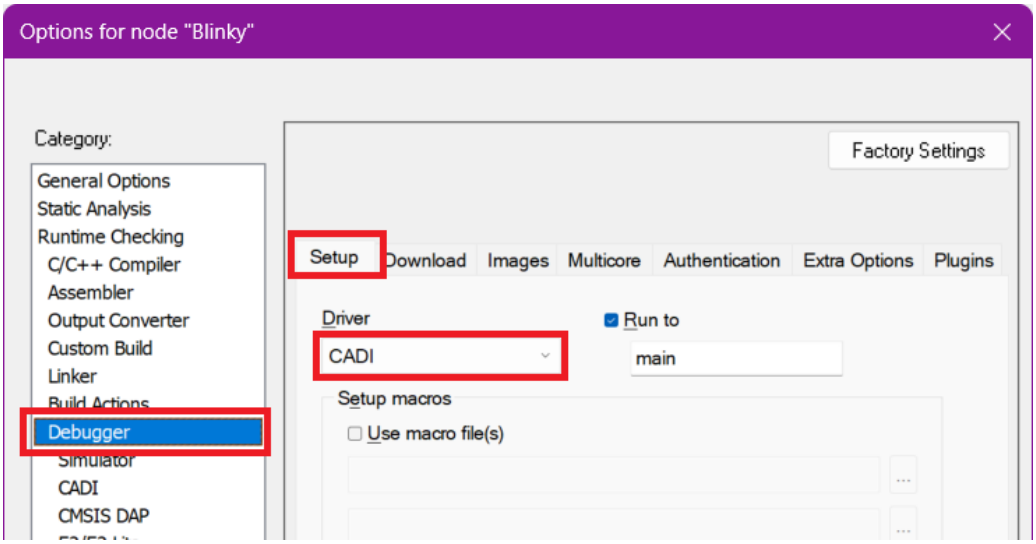

Figure 14: IAR EW: Selecting Debugger

You can select the desired CADI server in the CADI Category if there are more than one you wish to use. Otherwise, it can be left empty.

To run the debugger click on the "Download and Debug" button. The application will be automatically loaded to the model by the debugger.

8.4 Run - Terminal

1. Download **arm_vio.py**, like the following:

```
PS > wget https://github.com/ARM-software/AVH/raw/main/interface/python/arm_vio.py -OutFile arm_vio.py
```

Then set `verbosity = logging.DEBUG` in it.

2. Set environmental variables to run the example:

```
PS > $env:PYTHONHOME = 'C:/Users/<user>/AppData/Local/Programs/Python/Python39'
```

3. Execute the Example with the following command:

```
PS > <path_to__fvp>/FVP_Corstone_SSE-300_Ethos-U55.exe <path_to_out>/Vio.out
 -C mps3_board.v_path=<path_to_arm_vio.py_folder>
```

The VIO example might not work with newer Python versions. To run the example use version Python3.9.x.

9 Attachments

9.1 Blinky example tree

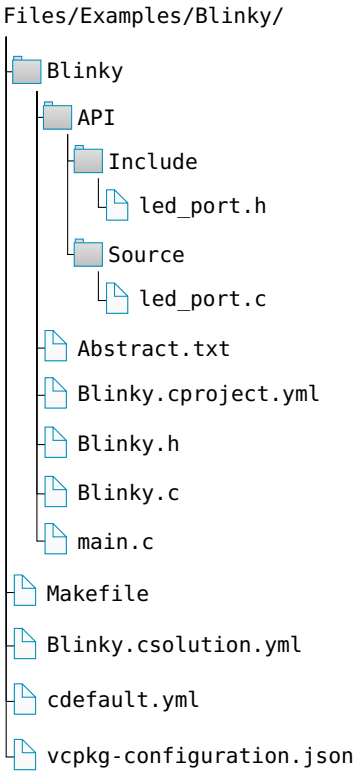

Figure 15: Blinky examples folder structure

9.2 Vio example tree

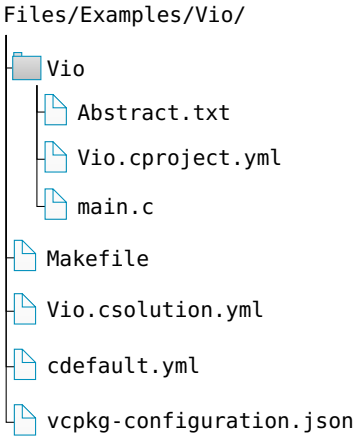

Figure 16: VIO examples folder structure