arm	**SSE-315** BSP Pack User Guide
Version:	1.0.0
Date of Issue:	March 1, 2024

Contents

1 Concept

1.1 Introduction

This document is a general guide to use the SSE-315 BSP pack. The CMSIS pack is to be used with the Corstone®-315 platform FVP model (Arm Corstone™ SSE-315 with Cortex®-M85 Example Subsystem). The pack contains necessary source files, a linker script file, and a specification document to kick-start development for the Corstone-315 platform. Provide reference secure side Blinky and Vio examples, to enable a user to understand csolution 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 and Vio examples on the SSE-315 FVP model.

Terms and Abbreviations:

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

1.2 Arm Mali®-C55 Versatile Image Signal Processor

The Corstone-315 Example Subsystem contains the Arm Mali®-C55 Versatile Image Signal Processor for Computer Vision and Smart Display Systems. The pack contains the necessary source files for HDLCD, to enable displaying the results of Image Signal Processor.
Example application and Driver can be found in Arm™ Mali®-C55 bare-metal ISP driver gitlab repository.

1.3 Prerequisites

- **Developement Environments:**
  CMSIS-Toolbox v2.3.0, Keil Studio for Visual Studio Code or IAR Embedded Workbench v9.50.2,

- **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-315 FVP model,

- **Package manager:**
  vcpkg.

- **Python version 3.9 for VIO examples**

Note

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
```

1.4 Documents

1. Corstone-315 FVP Technical Overview: contains overview of the FVP and its features.
2. Arm Corstone™ SSE-315 with Cortex-M85 and Ethos™-U65: Example Subsystem: contains overview of the FVP and its features.
3. Arm Corstone SSE-315 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.

Note

SSE-315 BSP Pack contains additional documentation in the "Documents" folder.

2 Installation

2.1 CMSIS-Toolbox

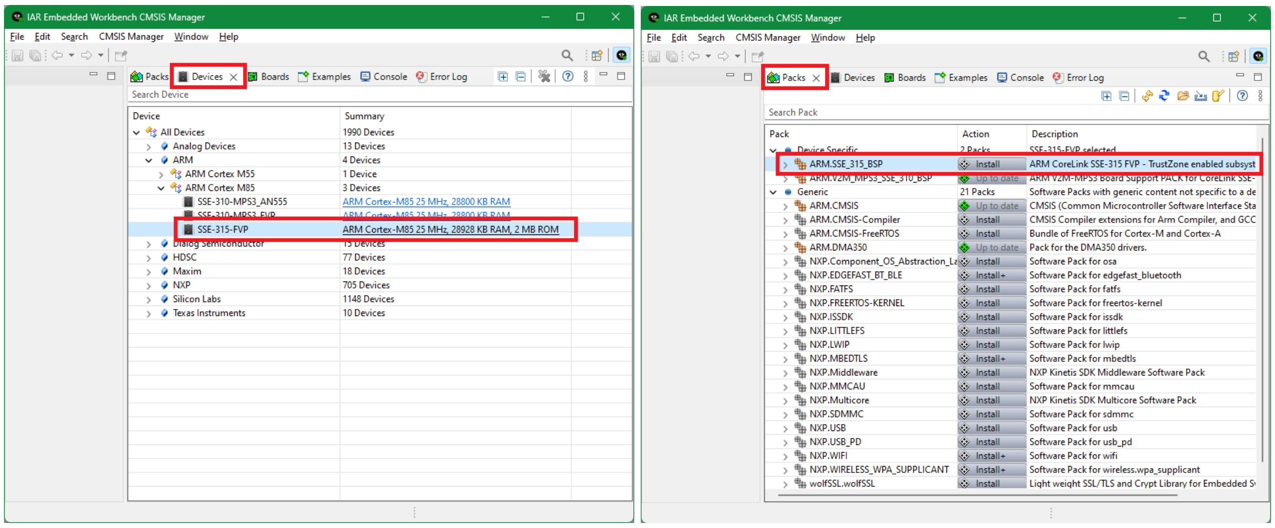
To install the `ARM::SSE_315_BSP` pack in command-line, use:

```
$ cpackget add ARM.SSE_315_BSP
```

This will install the prerequired `ARM.CMSIS.6.0.0`, `ARM.CMSIS-Compiler.2.0.0` and `ARM.DMA350.1.0.0` packages as well.

2.2 IAR Embedded Workbench

Install `ARM::SSE_315_BSP` using the CMSIS-Pack Manager. `(Find: Project > CMSIS-Pack Manager)` The pack can be browsed by selecting SSE-315 device under ARM Cortex M85 Devices.



Device	Summary
All Devices	1990 Devices
Analog Devices	13 Devices
ARM	4 Devices
ARM Cortex M55	1 Device
ARM Cortex M85	3 Devices
SSE-310-MPS3_AN555	ARM Cortex-M85 25 MHz, 28800 KB RAM
SSE-310-MPS3_FVP	ARM Cortex-M85 25 MHz, 28800 KB RAM
SSE-315-FVP	ARM Cortex-M85 25 MHz, 28928 KB RAM, 2 MB ROM
Dialog Semiconductor	15 Devices
HDSC	77 Devices
Maxim	18 Devices
NXP	705 Devices
Silicon Labs	1148 Devices
Texas Instruments	10 Devices

Pack	Action	Description
Device Specific	2 Packs	SSE-315-FVP selected
ARM.SSE_315_BSP	Install	ARM CoreLink SSE-315 FVP - TrustZone enabled subsyst
ARM.V2M_MPS3_SSE_310_BSP	Up to date	ARM V2M-MPS3 Board Support PACK for CoreLink SSE-
Generic	21 Packs	Software Packs with generic content not specific to a de
ARM.CMSIS	Up to date	CMSIS (Common Microcontroller Software Interface Sta
ARM.CMSIS-Compiler	Install	CMSIS Compiler extensions for Arm Compiler, and GCC
ARM.CMSIS-FreeRTOS	Install	Bundle of FreeRTOS for Cortex-M and Cortex-A
ARM.DMA350	Up to date	Pack for the DMA350 drivers.
NXP.Component_OS_Abstraction_La	Install	Software Pack for osa
NXP.EDGEFAST_BT_BLE	Install+	Software Pack for edgefast_bluetooth
NXP.FATFS	Install	Software Pack for fatfs
NXP.FREERTOS-KERNEL	Install	Software Pack for freertos-kernel
NXP.ISSDK	Install	Software Pack for issdk
NXP.LITTLEFS	Install	Software Pack for littlefs
NXP.LWIP	Install	Software Pack for lwip
NXP.MBEDTLS	Install+	Software Pack for mbedtls
NXP.Middleware	Install	NXP Kinetis SDK Middleware Software Pack
NXP.MMCAU	Install	Software Pack for mmcau
NXP.Multicore	Install	NXP Kinetis SDK Multicore Software Pack
NXP.SDMMC	Install	Software Pack for sdmmc
NXP.USB	Install	Software Pack for usb
NXP.USB_PD	Install	Software Pack for usb_pd
NXP.WIFI	Install+	Software Pack for wifi
NXP.WIRELESS_WPA_SUPPLICANT	Install+	Software Pack for wireless.wpa_supplicant
wolfSSL.wolfSSL	Install	Light weight SSL/TLS and Crypt Library for Embedded S

Figure 1: Selecting `ARM::SSE_315_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/SSE_315_BSP/1.0.0/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
```

Note

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

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/SSE-315-FVP/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-315 -a build/SSE-315-FVP/AC6/Debug/Blinky/outdir/Blinky.axf
```

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

```
$ make run-armclang
```

4 Blinky example: Keil Studio

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

Note

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

4.1 Import

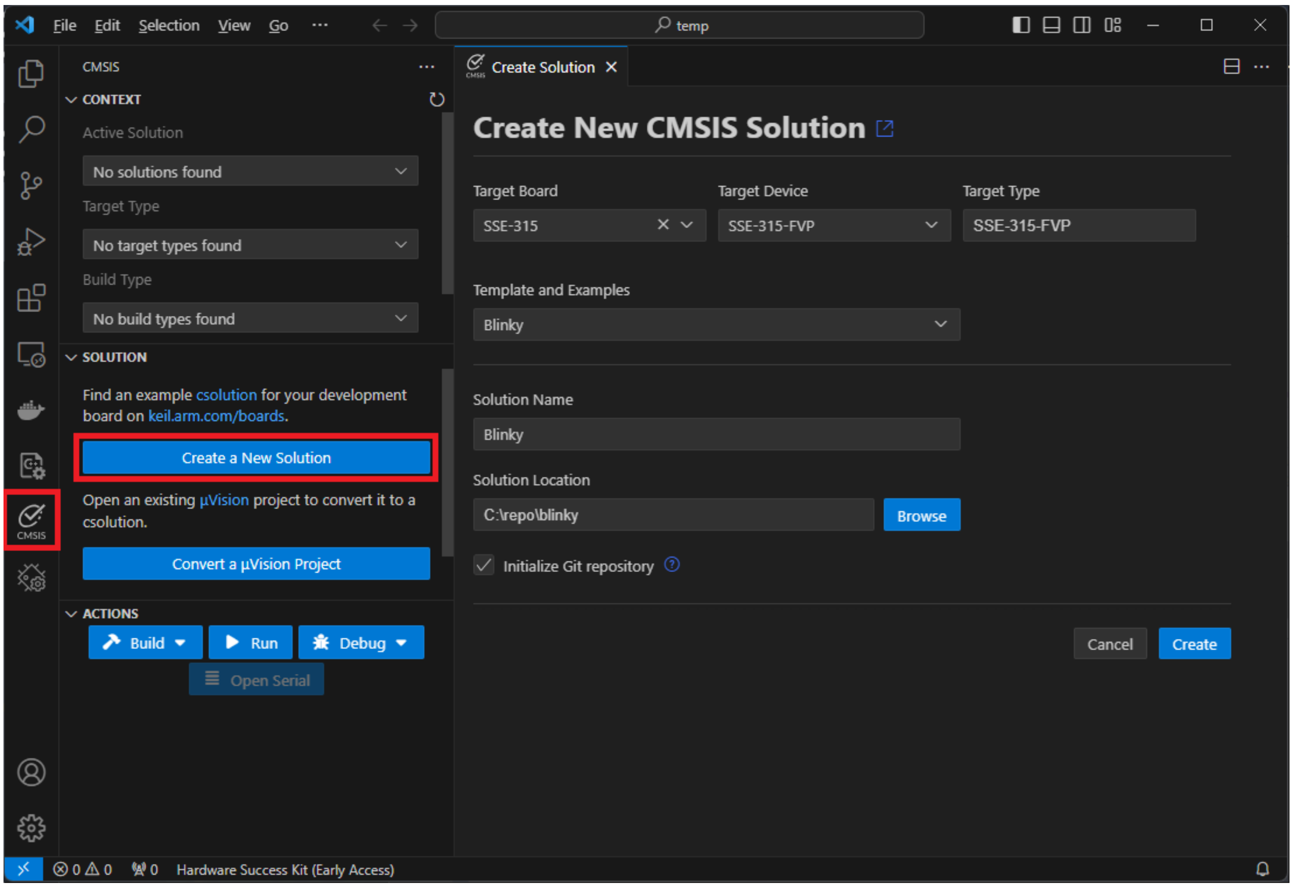
Copy the containing folder from the CMSIS Pack:

```
$ cp -r ${CMSIS_PACK_ROOT}/ARM/SSE_315_BSP/1.0.0/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-315` selected as the following:



Active Solution: No solutions found
Target Type: No target types found
Build Type: No build types found

Find an example csolution for your development board on keil.arm.com/boards.

Open an existing µVision project to convert it to a csolution.

Create New CMSIS Solution

Target Board	Target Device	Target Type
SSE-315	SSE-315-FVP	SSE-315-FVP

Template and Examples: Blinky

Solution Name: Blinky

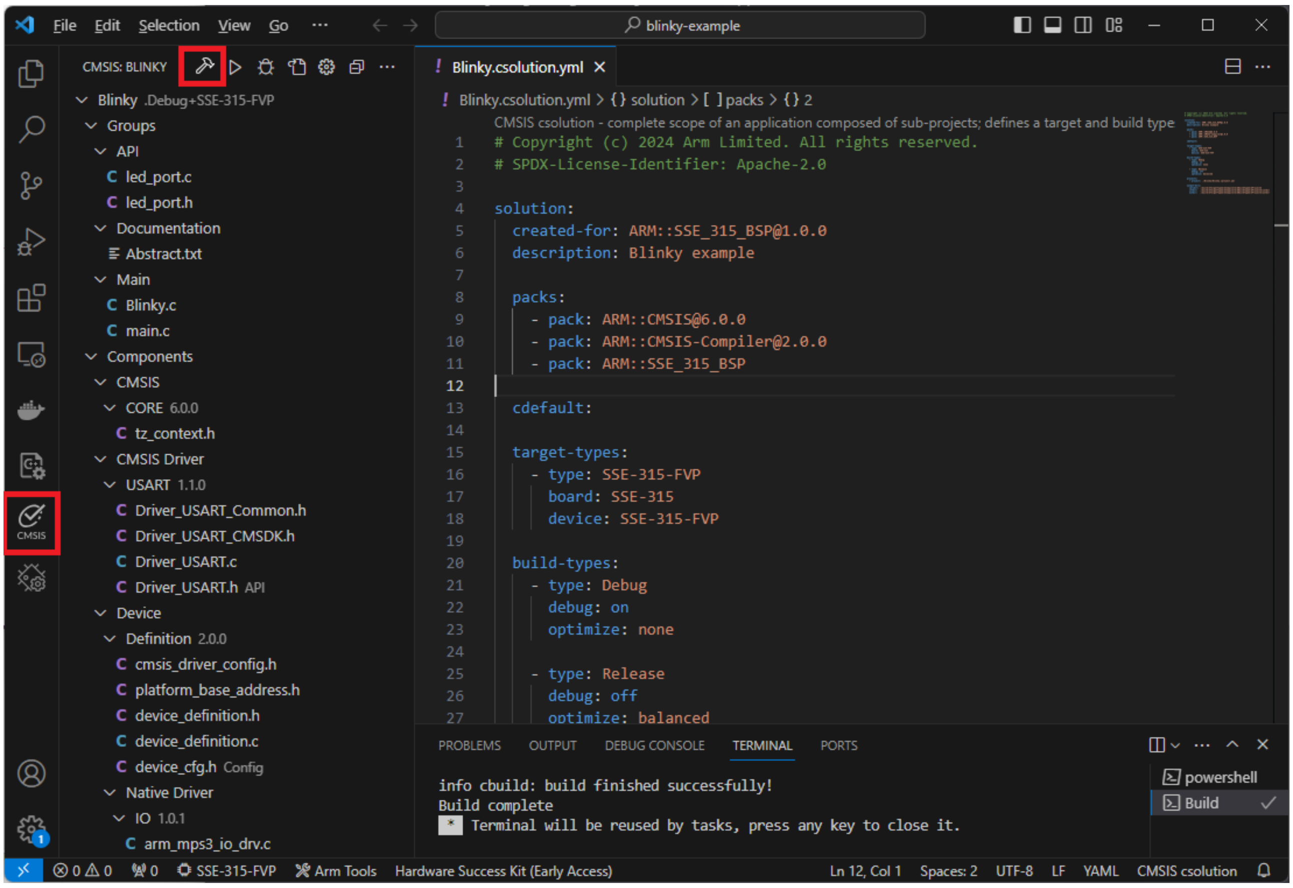
Solution Location: C:\repo\blinky

✓ Initialize Git repository

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**.



```yaml
# Copyright (c) 2024 Arm Limited. All rights reserved.
# SPDX-License-Identifier: Apache-2.0

solution:
  created-for: ARM::SSE_315_BSP@1.0.0
  description: Blinky example

  packs:
    - pack: ARM::CMSIS@6.0.0
    - pack: ARM::CMSIS-Compiler@2.0.0
    - pack: ARM::SSE_315_BSP

  cdefault:

  target-types:
    - type: SSE-315-FVP
      board: SSE-315
      device: SSE-315-FVP

  build-types:
    - type: Debug
      debug: on
      optimize: none

    - type: Release
      debug: off
      optimize: balanced
```

```
info cbuild: build finished successfully!
Build complete
 *  Terminal will be reused by tasks, press any key to close it.
```

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/SSE-315-FVP/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-315 -a build/SSE-315-FVP/AC6/Debug/Blinky/outdir/Blinky.axf
```

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

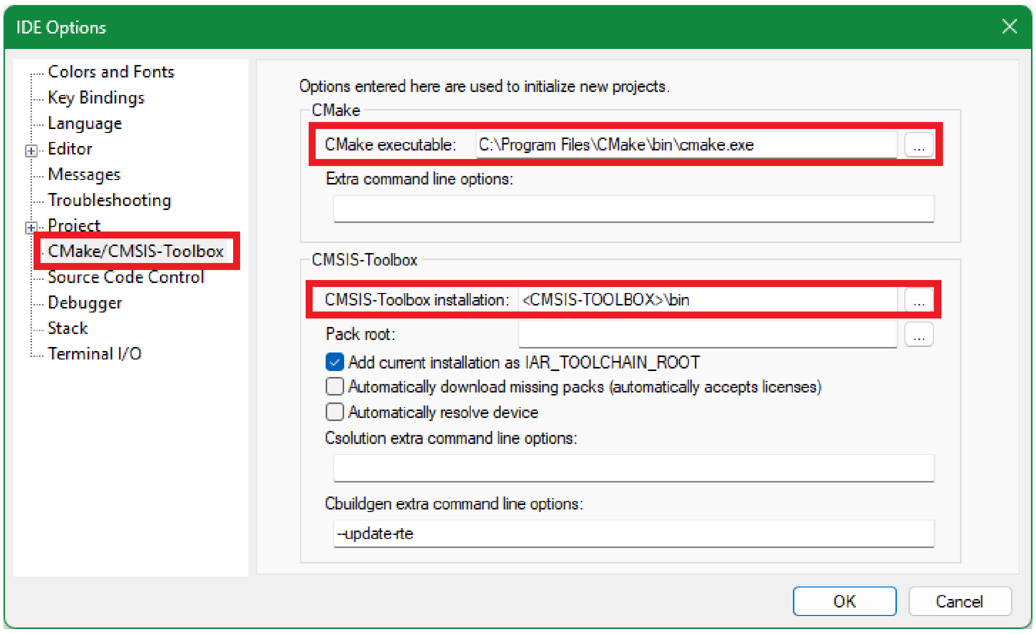
1. add `'compiler: IAR'` to `Blinky.csolution.yml` under `solution:` tag.

```

  compiler: IAR
  cdefault:

```

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)`.



Options entered here are used to initialize new projects.

CMake

CMake executable: C:\Program Files\CMake\bin\cmake.exe

Extra command line options:

CMSIS-Toolbox

CMSIS-Toolbox installation: <CMSIS-TOOLBOX>\bin

Pack root:

☑ Add current installation as IAR_TOOLCHAIN_ROOT

☐ Automatically download missing packs (automatically accepts licenses)

☐ Automatically resolve device

Csolution extra command line options:

Cbuildgen extra command line options: --update-rte

Figure 4: IAR EW: Set CMSIS-Toolbox path

Caution

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".

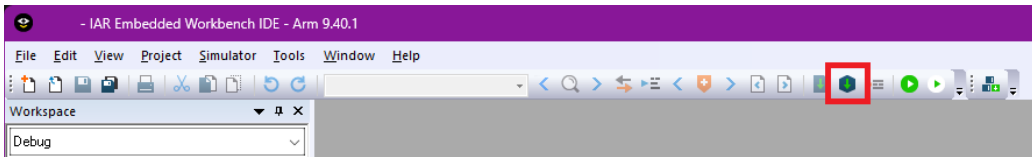

- IAR Embedded Workbench IDE - Arm 9.40.1
File Edit View Project Simulator Tools Window Help
Workspace
Debug

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-315 FVP model. First, download and install the SSE-315 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-315.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.

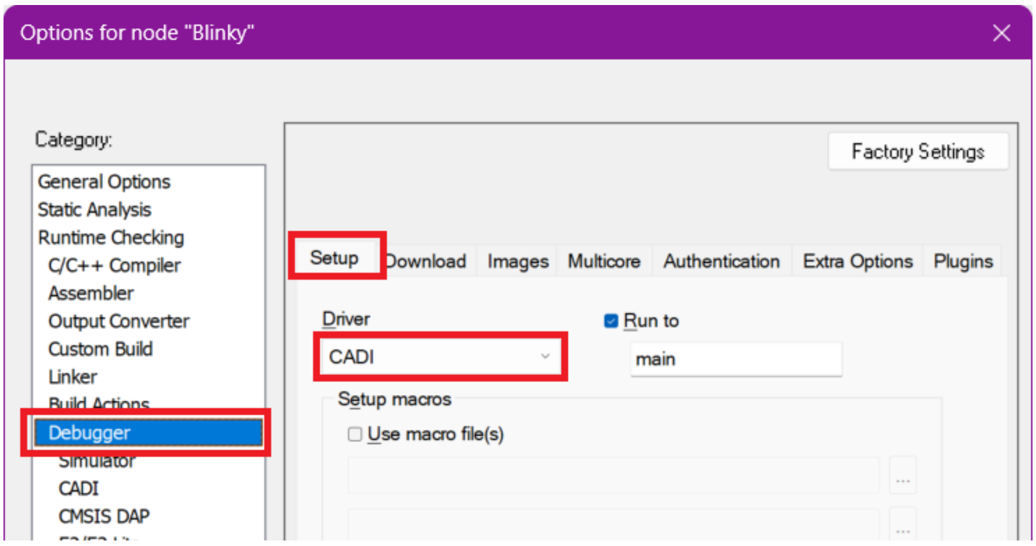


Figure 6: IAR EW: Selecting Debugger

Note

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-315.exe <path_to_bin>/Blinky.out`

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/SSE_315_BSP/1.0.0/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
```

Caution

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

In `Vio/RTE/Device/SSE_315_FVP/regions_SSE_315.h` modify
`#define __STACK_SIZE 0x00000200` to
`#define __STACK_SIZE 0x00000600`.
Then rebuild with `--rebuild`, and 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.

Note

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]
```

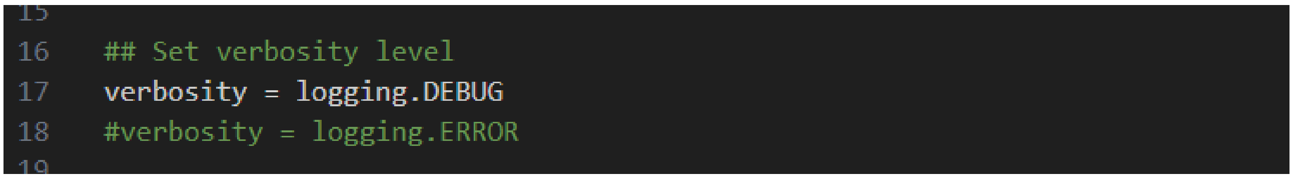
Binary:
Find the binary files in the following location:

```
build/{platform}/{toolchain}/{type}/{example}/outdir/Vio.[axf,elf,out]
```
as
```
build/SSE-315-FVP/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.



```
## Set verbosity level
verbosity = logging.DEBUG
#verbosity = logging.ERROR
```

Figure 7: 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-315 -a build/SSE-315-FVP/AC6/Debug/Vio/outdir/Vio.axf
  -C mps4_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
```

Caution

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.

Note

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

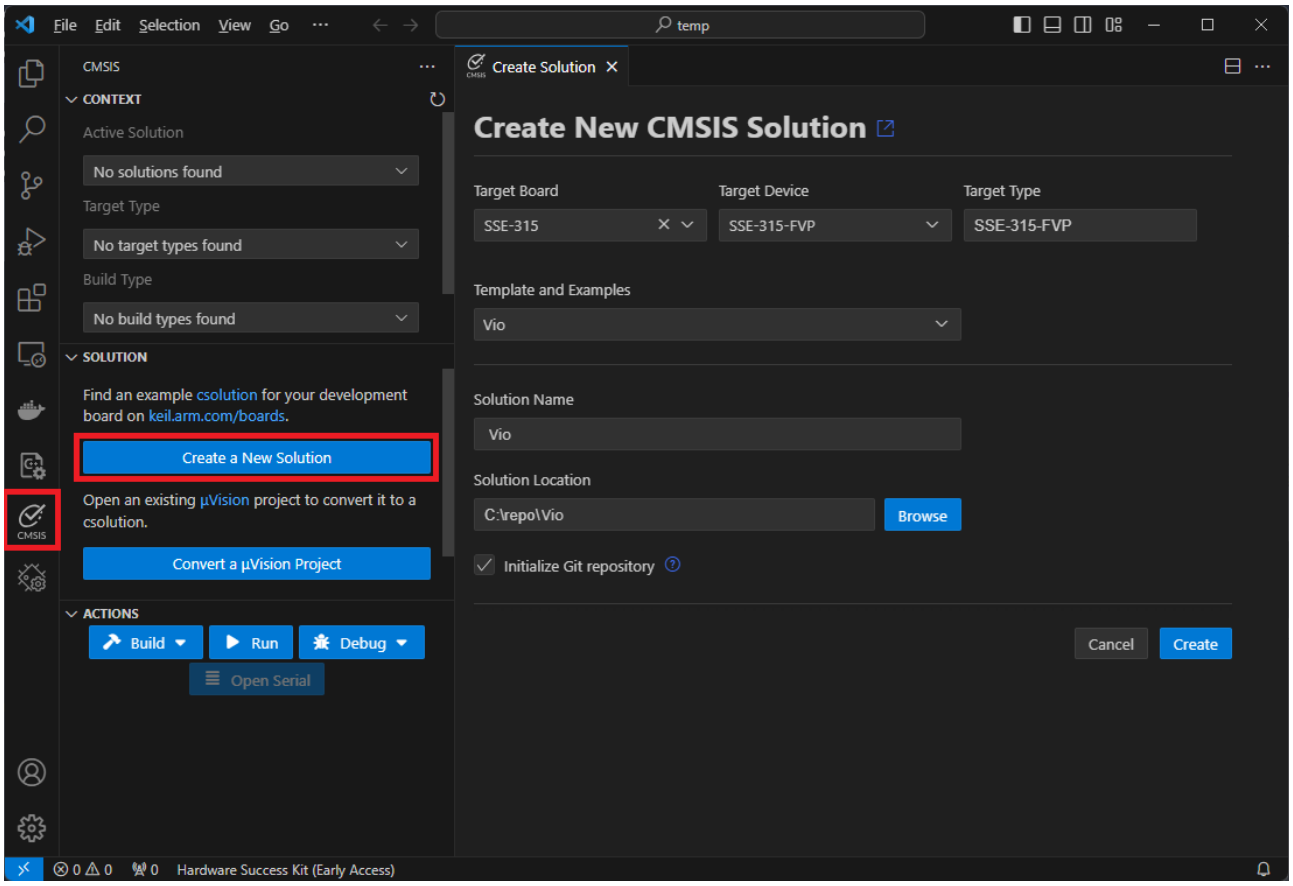
7.1 Import

Copy the containing folder from the CMSIS Pack:

```
$ cp -r ${CMSIS_PACK_ROOT}/ARM/SSE_315_BSP/1.0.0/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-315` selected as the following:



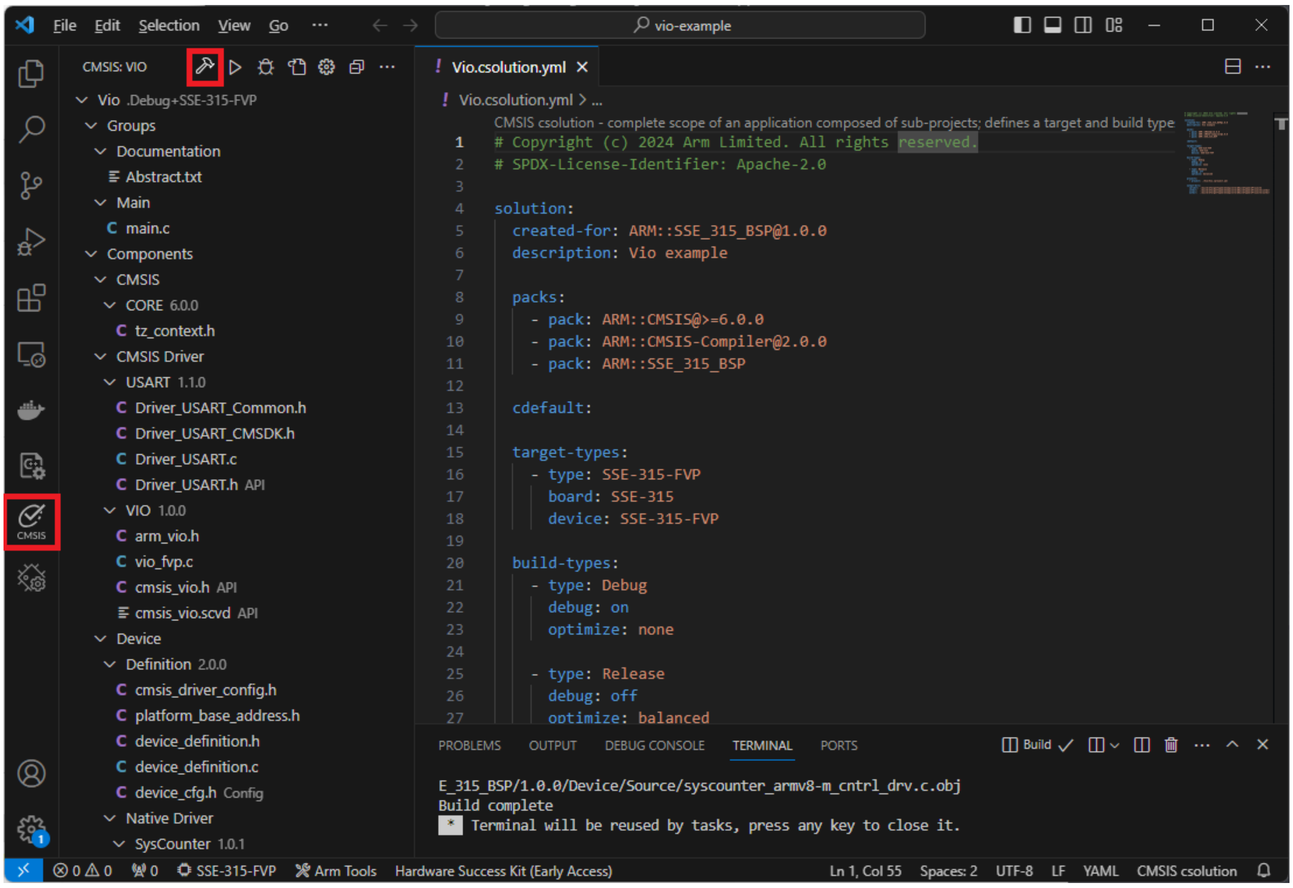
Create New CMSIS Solution

Target Board: SSE-315
Target Device: SSE-315-FVP
Target Type: SSE-315-FVP
Template and Examples: Vio
Solution Name: Vio
Solution Location: C:\repo\Vio
Initialize Git repository: ✓

Figure 8: 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**.



```
# Copyright (c) 2024 Arm Limited. All rights reserved.
# SPDX-License-Identifier: Apache-2.0

solution:
  created-for: ARM::SSE_315_BSP@1.0.0
  description: Vio example

  packs:
    - pack: ARM::CMSIS@>=6.0.0
    - pack: ARM::CMSIS-Compiler@2.0.0
    - pack: ARM::SSE_315_BSP

  cdefault:

  target-types:
    - type: SSE-315-FVP
      board: SSE-315
      device: SSE-315-FVP

  build-types:
    - type: Debug
      debug: on
      optimize: none

    - type: Release
      debug: off
      optimize: balanced
```

```
E_315_BSP/1.0.0/Device/Source/syscounter_armv8-m_cntrl_drv.c.obj
Build complete
 *  Terminal will be reused by tasks, press any key to close it.
```

Figure 9: Keil Studio: Building Project

Caution

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

In `Vio/RTE/Device/SSE_315_FVP/regions_SSE_315.h` modify
`#define __STACK_SIZE 0x00000200` to
`#define __STACK_SIZE 0x00000600`.
Then rebuild. (Next to build: `More Actions(···) > 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:

```

  compiler: AC6
  cdefault:

```

Binary:
Find the binary files in the following location:

`build/{platform}/{toolchain}/{type}/{example}/outdir/Vio.[axf,elf,out]`
as
`build/SSE-315-FVP/AC6/Debug/Blinky/outdir/Vio.axf`

7.3 Run - Terminal

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

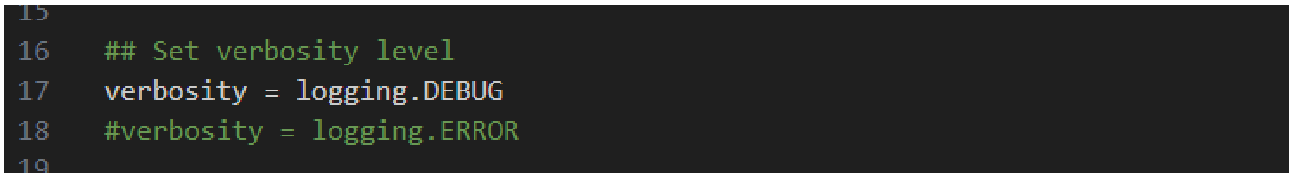
On Windows:

```
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.



```
## Set verbosity level
verbosity = logging.DEBUG
#verbosity = logging.ERROR
```

Figure 10: 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-315 -a build/SSE-315-FVP/AC6/Debug/Vio/outdir/Vio.axf
  -C mps4_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
```

Caution

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

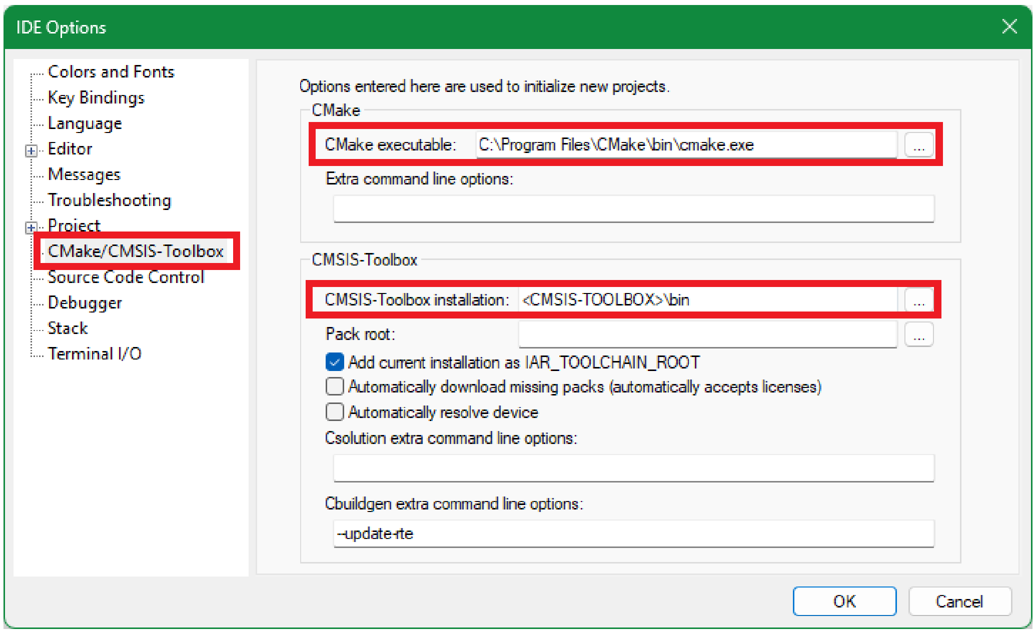
1. add `'compiler: IAR'` to `Vio.csolution.yml` under `solution:` tag.

```

  compiler:  IAR
  cdefault:

```

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)`.



Options entered here are used to initialize new projects.

CMake

CMake executable: C:\Program Files\CMake\bin\cmake.exe

Extra command line options:

CMSIS-Toolbox

CMSIS-Toolbox installation: <CMSIS-TOOLBOX>\bin

Pack root:

[✓] Add current installation as IAR_TOOLCHAIN_ROOT

[ ] Automatically download missing packs (automatically accepts licenses)

[ ] Automatically resolve device

Csolution extra command line options:

Cbuildgen extra command line options: --update-rte

Figure 11: IAR EW: Set CMSIS-Toolbox path

Caution

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".

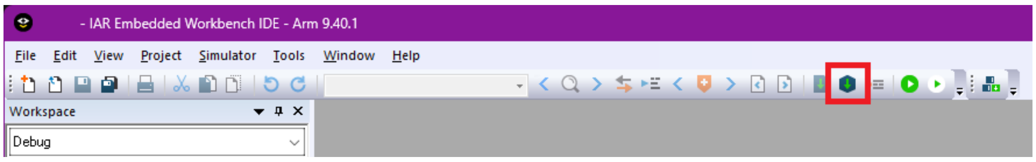


Figure 12: IAR EW: Building the Vio example

Caution

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

In `Vio/RTE/Device/SSE_315_FVP/regions_SSE_315.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-315 FVP model. First, download and install the SSE-315 FVP from the link provided in the Prerequisites section.

Caution

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'
```

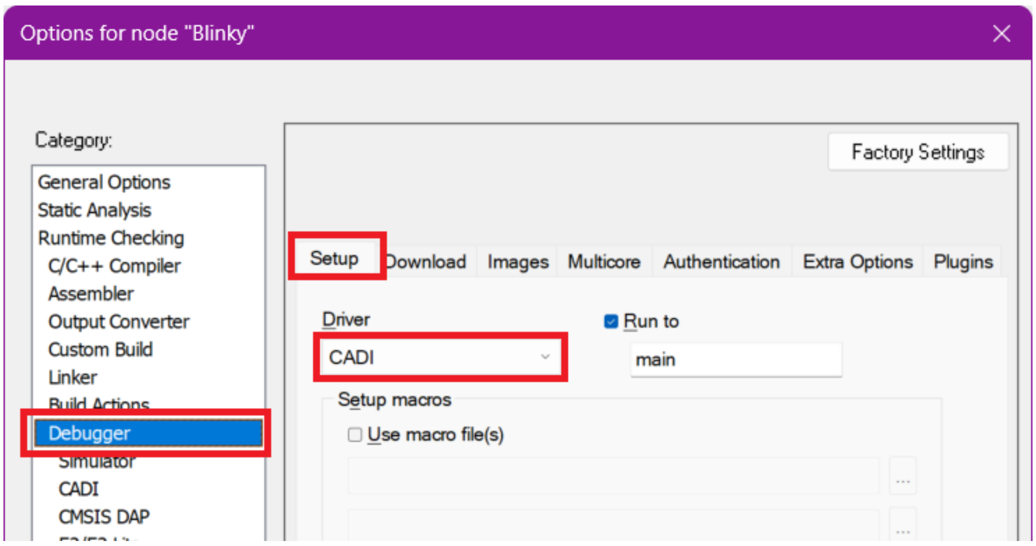
Start up the FVP with a CADI server:

```
<path_to_fvp>/FVP_Corstone_SSE-315.exe -S -C mps4_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.



Driver: CADI

☑ Run to: main

Figure 13: IAR EW: Selecting Debugger

Note

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-315.exe <path_to_out>/Vio.out
 -C mps4_board.v_path=<path_to_arm_vio.py_folder>
```

Caution

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

```
Files/Examples/Blinky/
├── Blinky
│   ├── API
│   │   ├── Include
│   │   │   └── led_port.h
│   │   └── Source
│   │       └── led_port.c
│   ├── Abstract.txt
│   ├── Blinky.cproject.yml
│   ├── Blinky.h
│   ├── Blinky.c
│   └── main.c
├── Makefile
├── Blinky.csolution.yml
├── cdefault.yml
└── vcpkg-configuration.json
```

Figure 14: Blinky examples folder structure

9.2 Vio example tree

```
Files/Examples/Vio/
├── Vio
│   ├── Abstract.txt
│   ├── Vio.cproject.yml
│   └── main.c
├── Makefile
├── Vio.csolution.yml
├── cdefault.yml
└── vcpkg-configuration.json
```

Figure 15: VIO examples folder structure