MPS2(+) Getting Started Guide

Version 1.0

Non-Confidential

Issue 01

110479_0100_01_en

MPS2(+) Getting Started Guide

Release information

Document history

Issue	Date	Confidentiality	Change
0100-01	25 April 2025	Non-Confidential	Initial release

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. MPS2(+) FPGA prototyping board

The MPS2 platform offers a relatively large FPGA for prototyping Cortex-M based designs with a range of debug options.

MPS2

MPS2 is an FPGA-based motherboard targeting Cortex-M systems.

It provides:

- Preconfigured implementations of M-class cores for demonstration and software validation
- An FPGA prototyping environment for hardware development

DesignStart

Arm DesignStart is an Arm IP online access portal, enabling users fast and easy access to trial a selection of Arm products without charge.

Products available include one of the most licensed Arm processor, the Arm Cortex-M0 processor and an array of Arm Artisan Physical IP solutions optimized for a wide range of foundry processes. A new design option for the performance efficient Arm Cortex-M3 processor is also available.

There are two ways to access industry-leading processor IP through DesignStart:

- DesignStart Eval enables anybody to click and get instant access to Cortex-M0, Cortex-M3 and their subsystems. Designers can configure or modify the subsystem, add their own IP and peripherals, and then prototype on an FPGA. See here for more information.
- DesignStart Pro is for companies looking to develop their own chip. Companies simply register on the DesignStart website, sign and return a simplified contract, then start building their chip. No upfront fees, just a simple success-based royalty once they are in production. See here for more information.

DesignStart has its own Arm Connected Community. Visit here for DesignStart information, resources and technical information.

Versions

There are 2 versions of MPS2. MPS2 (V2M-MPS2-0318A) and MPS2+ (V2M-MPS2-0318C) which provides double the FPGA capacity of MPS2. Only MPS2+ is currently available for purchase.

M0 DesignStart works with both revisions. M3 DesignStart is designed for the MPS2+.

Application Notes

The preconfigured cores for MPS2 are provided as Application Notes.

See here for more information regarding MPS2. This page provides a table of information about the Application Notes available for the board, including links to download the documentation for each one. This page also gives links to downloads for the latest CMPS deliverables for MPS2(+), which contains the images for the Armv6-M and Armv7-M Application Notes. There are also links to the new Armv8-M example IoT FPGA Images for the MPS2+ board.

The image downloads are subject to an EULA.

Mbed Target

The MPS2 board is a recognised Mbed Target board. There are 3 target options:

- Arm Cortex-M Prototyping System - This is for use with the standard preconfigured cores. More details here.

  NOTE: When used in Mbed online compiler you can chose which core you are working with: M0, M0+, M3 (default), M4 or M7. The selection is made from a drop down box in the top right of the online compiler page.
- MPS2 with Arm Cortex-M3 DesignStart. More details here.
- Example Subsystem for Embedded SSE-100 (IOT Subsystem for Cortex-M) - More details here.

In the github repo for mbed, the targets MPS2, CM3DS_MPS2 and IOTSS can be found in the TARGET_ARM_SSG folder.

There is new firmware for MPS2+ available which adds compatibility improvements for Mbed - See here

2. Keil Board Support Packs for MPS2(+)

There are a number of Keil board support packs available for the MPS2 board. Here we give a brief description and link for each pack. Follow the links for further information about these packs.

Arm MPS2(+) Board Support Pack for Cortex-M System Design Kit

This pack provides Keil support for the ARMv7-M processor images available for MPS2(+)

The pack is available here.

Arm MPS2+ Board Support Pack for DesignStart Devices

This pack provides Keil support for the DesignStart images available for MPS2+

The pack is available here.

Arm MPS2+ Device Family pack for IoT-Kit devices

This pack provides Keil support for the ARMv8-M based IoT kits images available for MPS2+

The pack is available here.

Arm MPS2+ Board Support pack for Example SSE-200 (AN521)

For more information visit the Community page here.

3. Keil pack for SSE-200 IoT System for MPS2+

A Keil board support pack is available for the SSE-200 IoT Subsystem for Cortex-M image for MPS2+. There is also a Trusted Firmware for M-Class Middleware pack available.

Note

Please ensure that you are running the latest version of the FPGA image for SSE-200 IoT Subsystem for Cortex-M (AN521) on your MPS2+ board. This is available here.

V2M-MPS2_SSE_200 Keil Support Pack

Version 2.0.0

- Adds support for the TF-M BSP requirements
- New drivers added:
  - SPI
  - GP Timer
  - QSPI Flash controller
- Removed support for previous examples
- Requires installation of Arm TF-M Middleware pack version 1.1.0 (see below)

Version 1.0.3

The features provided by this pack include:

- Timer driver
- Dualtimer driver
- GPIO driver
- I2C driver
- MPS2 IO driver
- UART driver
- SPI driver
- Watchdog driver
- Driver for security control
- Non TF-M examples

The pack is available here.

Trusted Firmware-M Middleware Pack

The Trusted Firmware-M (TF-M) Middleware Keil Pack is a derived work, based on the publicly available, open source project: https://www.trustedfirmware.org/

Keil Middleware Packs generally require a suitable Board Support Pack for the relevant platform.

Version: 1.1.0

This pack includes:

- TF-M Core, Audit Log and Secure Storage Service as secure services
- Non-Secure API.
- Out-of-the box examples based on TF-M configurations are also attached.
- Provides support for SSE-200-MPS2 BSP v2.0.0 (see above).

The TF-M Middleware Pack is available here.

4. SSE-200 for MPS2+ Software Examples

Three example software projects are provided with the Keil Trusted Firmware-M Pack v1.1.0 for use with the V2M-MPS2_SSE_200 Board Support Pack versions 2.0.0.

The examples are all based on Trusted Firmware-M projects:

- TF-M Demo – This example is based on Trusted Firmware-M project's 'Default Configuration'.
- TF-M Core Test - This example is based on TFM project's 'Core Tests'.
- TF-M Regression Test - This example is based on Trusted Firmware-M project's 'Regression Configuration'.

For instructions on how to build, run and debug these examples, see the TF-M Examples User Guide

Please note that Keil Debug of these examples will only work with FPGA image for SSE-200 IoT Subsystem for Cortex-M (AN521) Version 3.0 or later available here.

Three example software projects are provided with the Keil Arm MPS2 SSE-200 Board Support Pack v1.0.3.

Note

These examples do not build when Keil Arm MPS2 SSE-200 BSP v2.0.0 or later is installed.

Further information about these examples can be found on the following pages:

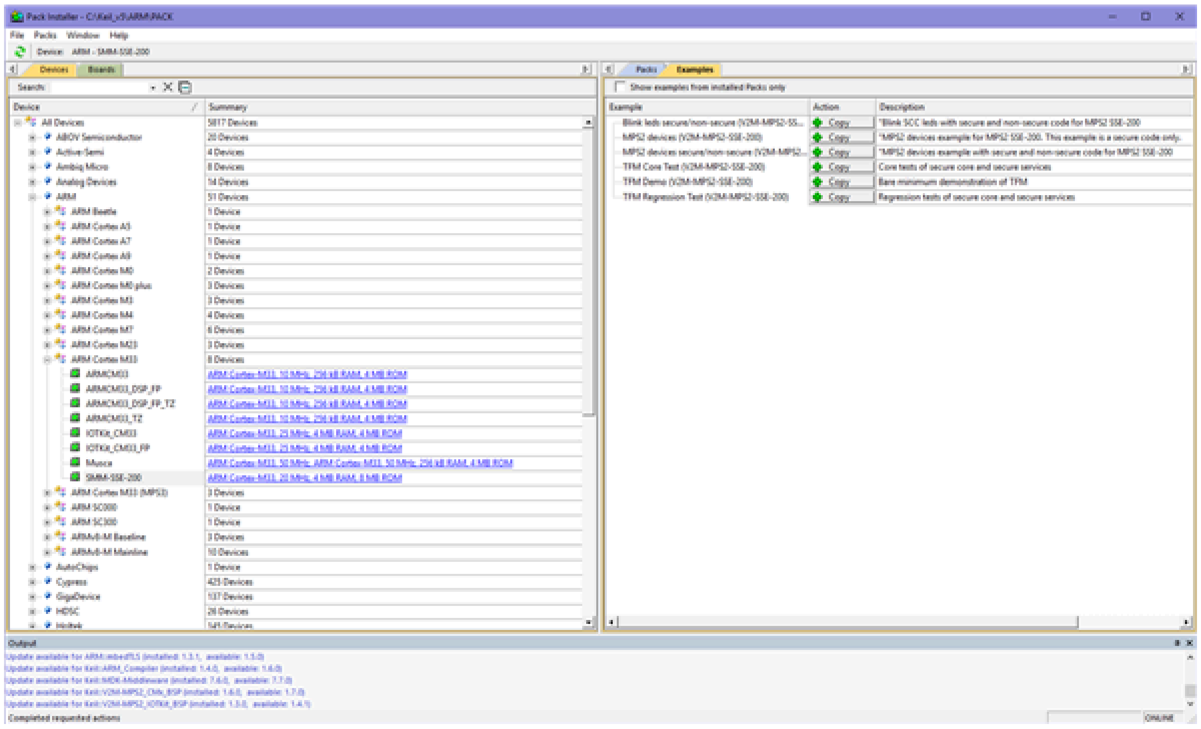
- SSE-200 Devices Example
- SSE-200 Secure/Non-secure Devices Example
- SSE-200 Secure/Non-secure Blinky Example

Accessing the Examples

When you have the pack installed in Keil MDK, you can make copies of these projects using MDK Pack Installer. From the Pack Installer under the Devices tab in the left hand pane select ARM -> ARM Cortex-M33 -> SMM-SSE-200.

In the right hand pane of the Pack Installer select the Examples tab. There you will be presented with the the examples for SSE-200. You can press the "Copy" button for each of the examples to create a copy of the example in a folder of your choosing.

Figure 4-1: Accessing the Examples



Pack Installer - C:\Keil_v5\ARM\PACK

Device: ARM - SMM-SSE-200

Device	Summary
All Devices	5817 Devices
ABOV Semiconductor	20 Devices
Active-Semi	4 Devices
Ambiq Micro	8 Devices
Analog Devices	14 Devices
ARM	51 Devices
ARM Beetle	1 Device
ARM Cortex A5	1 Device
ARM Cortex A7	1 Device
ARM Cortex A9	1 Device
ARM Cortex M0	2 Devices
ARM Cortex M0 plus	3 Devices
ARM Cortex M3	3 Devices
ARM Cortex M4	4 Devices
ARM Cortex M7	6 Devices
ARM Cortex M23	3 Devices
ARM Cortex M33	8 Devices
ARMCM33	ARM Cortex-M33, 10 MHz, 256 kB RAM, 4 MB ROM
ARMCM33_DSP_FP	ARM Cortex-M33, 10 MHz, 256 kB RAM, 4 MB ROM
ARMCM33_DSP_FP_TZ	ARM Cortex-M33, 10 MHz, 256 kB RAM, 4 MB ROM
ARMCM33_TZ	ARM Cortex-M33, 10 MHz, 256 kB RAM, 4 MB ROM
IOTKit_CM33	ARM Cortex-M33, 25 MHz, 4 MB RAM, 4 MB ROM
IOTKit_CM33_FP	ARM Cortex-M33, 25 MHz, 4 MB RAM, 4 MB ROM
Musca	ARM Cortex-M33, 50 MHz, ARM Cortex-M33, 50 MHz, 256 kB RAM, 4 MB ROM
SMM-SSE-200	ARM Cortex-M33, 20 MHz, 4 MB RAM, 8 MB ROM
ARM Cortex M33 (MPS3)	3 Devices
ARM SC000	1 Device
ARM SC300	1 Device
ARMv8-M Baseline	3 Devices
ARMv8-M Mainline	10 Devices
AutoChips	1 Device
Cypress	423 Devices
GigaDevice	137 Devices
HDSC	28 Devices
Holtek	141 Devices

Show examples from installed Packs only

Example	Action	Description
Blink leds secure/non-secure (V2M-MPS2-SS...	Copy	"Blink SCC leds with secure and non-secure code for MPS2 SSE-200
MPS2 devices (V2M-MPS2-SSE-200)	Copy	"MPS2 devices example for MPS2 SSE-200. This example is a secure code only.
MPS2 devices secure/non-secure (V2M-MPS2...	Copy	"MPS2 devices example with secure and non-secure code for MPS2 SSE-200
TFM Core Test (V2M-MPS2-SSE-200)	Copy	Core tests of secure core and secure services
TFM Demo (V2M-MPS2-SSE-200)	Copy	Bare minimum demonstration of TFM
TFM Regression Test (V2M-MPS2-SSE-200)	Copy	Regression tests of secure core and secure services

Output

```
Update available for ARM::mbedTLS (installed: 1.3.1,  available: 1.5.0)
Update available for Keil::ARM_Compiler (installed: 1.4.0,  available: 1.6.0)
Update available for Keil::MDK-Middleware (installed: 7.6.0,  available: 7.7.0)
Update available for Keil::V2M-MPS2_CMx_BSP (installed: 1.6.0,  available: 1.7.0)
Update available for Keil::V2M-MPS2_IOTKit_BSP (installed: 1.3.0,  available: 1.4.1)
```

Completed requested actions

ONLINE

4.1 MPS2+ SSE-200 TF-M Examples User Guide

Each of the examples consist of 4 projects. Two projects provide the secure and the non-secure images to be run on the board. Two other projects provide secure services libraries (Audit Log, Secure Storage) utilised by the Secure image.

Build

In order to build or load a project it needs to be set as the active project in Keil.

To compile this example correctly the secure services needs to be compiled first as libraries, then the secure project as a binary, finally the non-secure.

As a result of the secure build an object file with the veneers is generated which is then linked to the non-secure image.

Run the Example

To run the example on the V2M-MPS2 board you need to load both the secure and non-secure images onto the board.

Output from the examples are printed to the serial port:

```
UART0 serial port at 115200 baudrate (8N1).
```

Programming the board

To program the MPS2 board, follow these steps:

- Copy the `<secure>.axf` and `<non_secure>.axf` files generated during compilation in `<MPS2 device name>/SOFTWARE/`
- Go to `<MPS2 device name>/MB/HBI0263<board revision letter>/AN521/ folder`
- Open the `<MPS2 device name>/MB/HBI0263<board revision letter>/AN521/images.txt` file
- Update the “TOTALIMAGES:” with “TOTALIMAGES: 2”
- Add/update the “IMAGExADDRESS:” and “IMAGExFILE:” settings with the location and the binary names. You should have similar to this:

```
TITLE: Versatile Express Images Configuration File

[IMAGES]
TOTALIMAGES: 2                   ;Number of Images (Max: 32)

IMAGE0ADDRESS: 0x00000000
IMAGE0FILE: \Software\tfm_s.axf  ; TF-M example application secure binary
IMAGE1ADDRESS: 0x00000000
IMAGE1FILE: \Software\tfm_ns.axf ; TF-M example application non-secure binary
```

- Close the images.txt file
- Eject the usb device
- Reset the board

Note

axf file names need to fit in the 8.3 file name format required by MPS2, and filenames generated must be lower case

Debugging

Please note that Keil Debug of these examples will only work with FPGA image for SSE-200 IoT Subsystem for Cortex-M (AN521) Version 3.0 or later.

Debugging has only been tested with ULINK Pro Debugger. To enable it, ensure the following debug settings:

- Right-click on the project name you want to debug and “Set as Active Project”.
- Go to menu “Project” -> `"Options for <project_name>"`.
- Go to “Debug” tab and select ULINK Pro Debugger.
- Go to “Settings” and under the “Debug” tab un-tick both “Cache Options”, “Cache Code” and “Cache Memory”.
- Under “Flash Download” tab, select “Do not Erase” in “Download Function” and un-tick “Program”, “Verify” and “Reset and Run”.
- On the main project Window start debugging in “Debug” -> “Start/Stop Debug Session”.

See the ReadMe file provided with the example for further information.

4.2 MPS2+ SSE-200 Devices Example

PLEASE NOTE: These examples do not build when Keil Arm MPS2 SSE-200 BSP v2.0.0 or later is installed.

This example shows some of the functionality of the MPS2 devices (SCC leds, switches, user buttons, user leds, touchscreen and graphical LCD).

The application shows the following functionalities:

- Shows a value via SCC leds (8 leds).
- The user button 0 increments the value.
- The user button 1 decrements the value.
- When the button 0 is pressed, the used LED 0 is switched on and used LED 1 is switched off.
- When the button 1 is pressed, the used LED 1 is switched on and used LED 0is siwtched off.
- When the user touchs the center of the touchscreen, the SCC LEDs blinks and the graphical LCD changes the image.

The example runs in secure mode only.

See the ReadMe file provided with the example for more information.

4.3 MPS2+ SSE-200 Devices Secure/Non-secure Example

PLEASE NOTE: These examples do not build when Keil Arm MPS2 SSE-200 BSP v2.0.0 or later is installed.

This example shows the same functionality described in the MPS2 device example. However, the implementation is different. The implementation is divided into secure and non-secure code.

The secure code performs the following functionality:

- Sets secure and non-secure system configuration.
- Gets and provides button states.
- Gets and provides touchscreen information.
- Sleep function.
- Switches on/off user LEDs based on user button states.

The non-secure code performs the following functionality:

- Requests to the secure code:

    - Button states.
    - If the touchscreen is pressed in the central area.
    - Sleep for X amount of ms.
- Switches on/off the SCC LEDs.

To compile the example correctly it is necessary to first compile the secure code, and then the non-secure code.

Compiling the secure code generates the veneers object file required by the non-secure code to compile correctly.

See the ReadMe file provided with the example for further information.

4.4 MPS2+ SSE-200 Secure/Non-secure Blinky Example

PLEASE NOTE: These examples do not build when Keil Arm MPS2 SSE-200 BSP v2.0.0 or later is installed.

This example swicthes ON and OFF the MPS2 SCC LEDs based on the MPS2 user button pressed.

The implementation is divided in secure and non-secure code.

The secure code performs the follow functionality:

- Sets secure and non-secure system configuration.
- Gets and provides button states.
- Sleep function.
- Switches on/off user LEDs based on user button states.

The non-secure code performs the follow functionality:

- Requests to the secure code:
    - Button states.
    - Sleep for X amount of ms.
- Switches on/off the SCC LEDs based on:
    - If User button 0 is pressed, the SCC LEDs blink state is enabled.
    - If User button 1 is pressed, the SCC LEDs blink state is disabled.

To compile the example correctly it is necessary to first compile the secure code, and then the non-secure code.

Compiling the secure code generates the veneers object file required by the non-secure code to compile correctly.

See the ReadMe file provided with the example for further information.

5. Running uCLinux on the Arm MPS2(+) platform

uClinux is Linux built for microcontroller platforms without an MMU. It relies on a specific version of libc not available in the standard Linaro toolchain binary releases. These instructions use Buildroot to automate compiling the toolchain and creating a simple busybox filesystem.

MPS2 is an Arm supplied FPGA based hardware development platform that can be configured to use a variety of Arm Cortex-M family cores. The platform is also available as a FVP (Fixed Virtual Platform) software model - included with Arm's DS-5 toolchain.

The instructions and configuration files supplied describe how to build and run uClinux on

- MPS2 hardware with Cortex-M7 (AN500) or CM3DS (Cortex-M3 Design Start) FPGA images
- MPS2 Cortex-M7 FVP

It should be possible to port the example to other Cortex-M variants. However changes may be required to the following configuration files:

- The buildroot defconfig will need modifying, to reflect the processor variant and relevant dts file
- The device tree configuration (The kernel sources include a device tree for Cortex-M3/an385)
- The boot wrapper makefile (which has specific assumptions about where RAM is located)

MPS2 specific configuration files

Download and extract the `MPS2_config.zip` archive associated with this document

Build the kernel and busybox with buildroot

To build the kernel and busybox with buildroot:

1. Download and extract Buildroot. These instructions were tested with 2017.05, but should work with any recent version of Buildroot.
2. Copy the contents of `MPS2_config/configs` to `<buildroot>/configs`.
3. In `<buildroot>/board/arm` create the folder `mps2`
4. Copy the contents of `MPS2_config/mps2` into `<buildroot>/board/arm/mps2`
5. Apply the defconfig and build buildroot:

```
$ make arm_mps2_<platform>_defconfig
$ make -j16
```

This may take a while, it is rebuilding gcc + linux + busybox and will produce 3 things:

- Linux kernel Image + busybox in initramfs in `output/build/linux-4.11.3/arch/arm/boot/Image`
- Device tree in `output/images/<platform>.dtb`

- Toolchain in `output/host/usr/bin/`

Note: The FVP target uses a non-upstreamed variant of mps2-an399.dts, because the FVP requires a different ethernet device node than used by the in-tree dts.

Build the bootwrapper

This bootwrapper is a simple Linux bootloader intended for use with Cortex-M devices on Arm's MPS development platform.

1. Download the boot wrapper

   ```
   git clone https://github.com/ARM-software/bootwrapper.git -b cortex-m-linux
   ```

2. Copy the kernel Image and dtb produced by Buildroot into the root of the bootwrapper folder.
3. Edit the bootwrapper Makefile to use the correct PHYS OFFSET & .dtb for your platform

   ```
   #For MPS2 CM7  PHYS_OFFSET = 0x60000000
   #For CM3DS  PHYS_OFFSET = 0x21000000
   #For MPS2 CM7  DTB = ./mps2-an399.dtb
   #For FVP CM7  DTB = ./mps2-an399-fvp.dtb
   #For CM3DS  DTB = ./cm3ds.dtb
   ```

4. For CM3DS only edit `boot.s` to change the address of the UART to `0x40005000` (this will allow you to see early kernel boot messages)
5. Start the build

   ```
   export CROSS_COMPILE=<workspace>/buildroot-2017.05.01/output/host/usr/bin/arm-
   buildroot-uclinux-uclibcgnueabi-
   cd ./bootwrapper
   make
   ```

This will produce `boot.axf` (A small assembler shim which jumps from flash to RAM. Loads at `0x0`, jumps to PHYS_OFFSET). `linux.axf` (Includes the bootwrapper + Image at PHYS_OFFSET)

Run on MPS2 FVP model

Launch the FVP from the command line using:

```
$ FVP_MPS2_Cortex-M7 -a linux.axf
```

Run on MPS2 Development Board

The MPS2 board allows images to be programmed into flash via configuration files on the board's MMC card.

1. Ensure the boards MMC card is mounted on your host PC.
2. Copy `boot.axf` and `linux.axf` into the `SOFTWARE/` folder
3. Edit `MB/HBI0xyz/ANxyz/images.txt` (The exact path/file to modify is dictated by the FPGA image you have selected in `MB/board.txt`) to contain:

   ```
   [IMAGES]
   ```

```
  TOTALIMAGES: 2
  IMAGE0ADDRESS: 0x00000000
  IMAGE0FILE: \SOFTWARE\boot.axf
  IMAGE1ADDRESS: PHYS_OFFSET**
  IMAGE1FILE: \SOFTWARE\linux.axf
**PHYS_OFFSET is the appropriate RAM address for your platform, detailed above
```

4. Set baudrate to 9600bps on the UART console
5. Reset and boot

An example Linux boot log is included in the supplied `MPS2_config.zip` archive

6. MPS2(+) Mbed OS automated testing

The aim of this guide is to walk you through using the Mbed OS greentea testing tools to run automated tests on the Arm MPS2.

For more information, the main greentea page is here: Arm Mbed greentea on GitHub

Preparation

Before starting, ensure the following:

- The MPS2 has been configured to run Cortex-M3 DesignStart (Evaluation kit available for download from here)
- The MPS2 has been configured to load binaries called 'mbed.bin' in the root directory of the MPS2 drive
- Both the USB coming from the MPS2 and the USB from the UART port are connected to your host machine (and all drivers have finished installing) (don't connect terminal to uart)
- The MPS2 has been properly mounted as a drive. On Windows this should be automatic. On Linux you may need to configure your automounter.
- Note what serial port is being used by the MPS2. In Windows, this can be found in the Device Manager and should show up as 'COMx' where 'x' is a number.
    - On Linux, you may need to look in your `/dev` directory. The final serial port may look like `/dev/ttyACM0`, but this will ultimately depend on your OS.

Environment setup

These instructions use the virtual environment 'virtualenv' which is optional. System installations under virtualenv are non-persistent, thus allowing you to avoid lasting modifications to your computers system.

If you wish to install and run the virtual environment, run the following commands in the directory you wish to set up the tests in:

```
> pip install virtualenv
> virtualenv venv
```

Activate the virtual environment.

- On Windows: `> .\venv\Scripts\activate`
- On Linux/Mac: `> source venv/bin/activate`

Install tools

From the same directory, run the following commands, in the order given, to install the necessary tools and Mbed OS

```
> pip install mbed-cli mbed-ls
> pip install -U mbed-greentea
> pip install -U mbed-host-tests
> git clone https://github.com/ARMmbed/mbed-os
> cd mbed-os
```

Identify target board

Now that the tools are installed, we need to find your MPS2's unique 'target id'. Run the following:

```
> mbedls
```

This should produce output similar to this:

```
+---------------+---------------------+------------+------------
+-------------------------+-----------------+

| platform_name | platform_name_unique | mount_point | serial_port |
 target_id                | daplink_version |

+---------------+---------------------+------------+------------
+-------------------------+-----------------+

| ARM_CM3DS_MPS2| ARM_CM3DS_MPS2[0]    | E:          | unknown     |
 50040200074D652F3828F333 | 0001            |

+---------------+---------------------+------------+------------
+-------------------------+-----------------+
```

Notice how the 'serial_port' is showing up as 'unknown'. We need to tell mbedls which serial port to use.

Create a 'mbedls.json' file within the 'mbed-os' directory you are currently in. Add the following:

```
{
 "50040200074D652F3828F333": {
 "serial_port": "COM11"
 }
}
```

Where the 24 figure number should match the 'target_id' that was listed when running 'mbedls'.

You'll also need to change 'COM11' to the correct serial port. You should have found this earlier, please see the "Preparation" steps above for more information.

After saving the file, if the platform name is unknown in the output from mbedls, then run the following

```
> mbedls --mock 5004:ARM_CM3DS_MPS2
```

where the 4 figure number is the first four digits from the 'target id' that we see when we run 'mbedls'

Finally, run ‘mbedls’ again and you should see that the serial port and platform are correctly set (example below):

```
> mbedls
+---------------+---------------------+------------+------------
+-------------------------+----------------+

| platform_name | platform_name_unique | mount_point | serial_port |
 target_id                | daplink_version |

+---------------+---------------------+------------+------------
+-------------------------+----------------+

| ARM_CM3DS_MPS2| ARM_CM3DS_MPS2[0]    | E:          | COM11       |
50040200074D652F3828F333  | 0001            |

+---------------+---------------------+------------+------------
+-------------------------+----------------+
```

Building and running tests

From the mbed-os directory run the following:

```
> mbed test -m ARM_CM3DS_MPS2 -t ARM
```

You should see the tests build (if not already done) then start seeing tests run.

It will take approximately 30 minutes to complete.

7. FPGA Prescale register address incorrect in MPS2(+) software header file

The FPGA prescaler register address is incorrectly defined in the header file `SMM_MPS2.h`.

This file is used in all Arm test software for MPS2, such as selftest, shieldtest, demo etc. The same header file is also used in MBED base code for MPS2.

The incorrect file defines the following register offsets:

```
RESERVED; // Offset: 0x01C
PRESCALE; // Offset: 0x020
PSCNTR;   // Offset: 0x024
```

In order to use the prescaler feature on MPS2, SMM_MPS2.h will need to be edited locally such that these offsets are correctly defined as:

```
PRESCALE; // Offset: 0x01C
PSCNTR;   // Offset: 0x020
RESERVED; // Offset: 0x024
```

8. MPS2+ Firmware update for reboot.txt method

It is possible to control the MPS2+ Motherboard Configuration Controller (MCC) via the USB2.0 Configuration port. This functionality is described in the MPS2+ Technical Reference Manual (TRM) section 3.2 “Remote USB Operation”.

The reboot.txt method does not work correctly with the firmware version delivered with MPS2+.

A new version of firmware (mbb_v300.ebf) is now available that fixes this issue. A download of the new firmware is provided below.

The fix also requires a change to the contents of reboot, reset and shutdown .txt files. It is necessary to make the following changes:

A string must now be written at the start of the text file.

- reboot.txt must contain - “hsyxhj”
- reset.txt must contain - “jkmcgx”
- shutdown.txt must contain - “bmqjfe”

Examples of these files are also provided below. The rest of the method for performing the reboot, reset and shutdown action has not changed.

mbb_v300.ebf

reboot.txt

reset.txt

shutdown.txt