

Contents

Introduction	1
Prerequisites	1
Using the Cortex-M23 IoT Kit Image on MPS2+.....	1
Verify the Pack Installation	1
Copy and Run the Example Application	3
Notes.....	9

Introduction

This document describes a step by step process on how to use the Cortex-M23 based IoT Kit Image for the MPS2+ board with the MDK toolchain. While we've tested these implementations, there will be updates to the tools and FPGA images. Expect differences between these and follow-on implementations.

Prerequisites

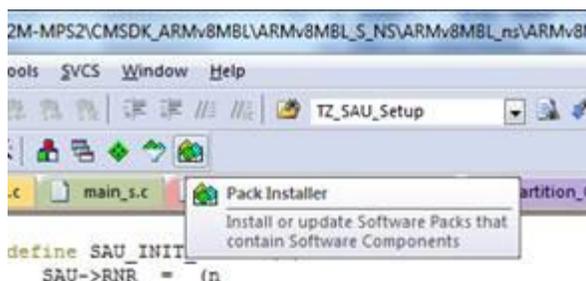
All the ARMv8-M support is now in our standard MDK product as of version 5.23 and later. Just perform the normal MDK installation.

To build and run the examples, you'll need the "CMSIS.5.0.1 (2017-02-03)" pack and "V2M-MPS2_IOTKit_BSP 1.2.0 (2017-02-08) packs available via the pack installer in MDK. These packs have the latest support for the MPS2+ (V2M-MPS2-0318C) board running the Cortex-M23 IoT Kit FPGA image.

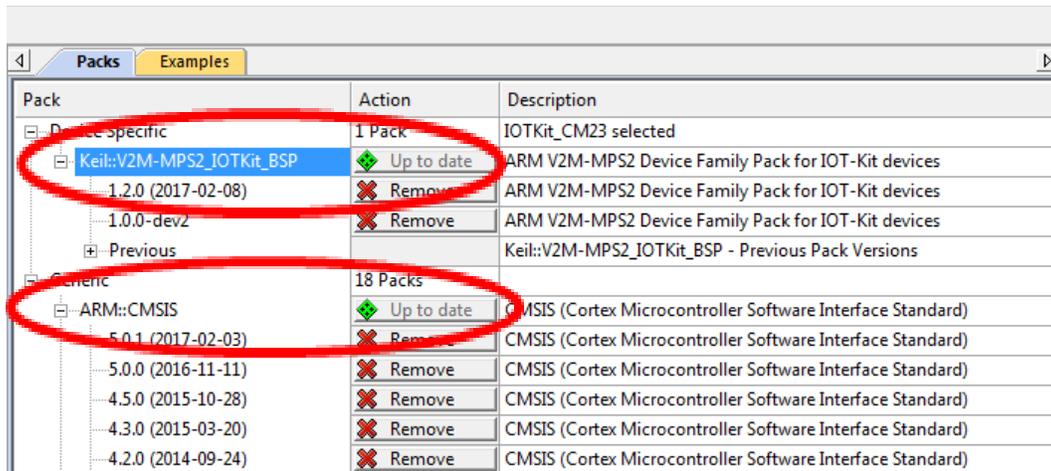
Using the Cortex-M23 IoT Kit Image on MPS2+

Verify the Pack Installation

1. Let's double check that the CMSIS.5.0.1" and "V2M-MPS2_IOTKit_BSP 1.2.0" packs are both installed properly. We'll use the pack installer from MDK. Click on the pack installer icon from the icon bar...



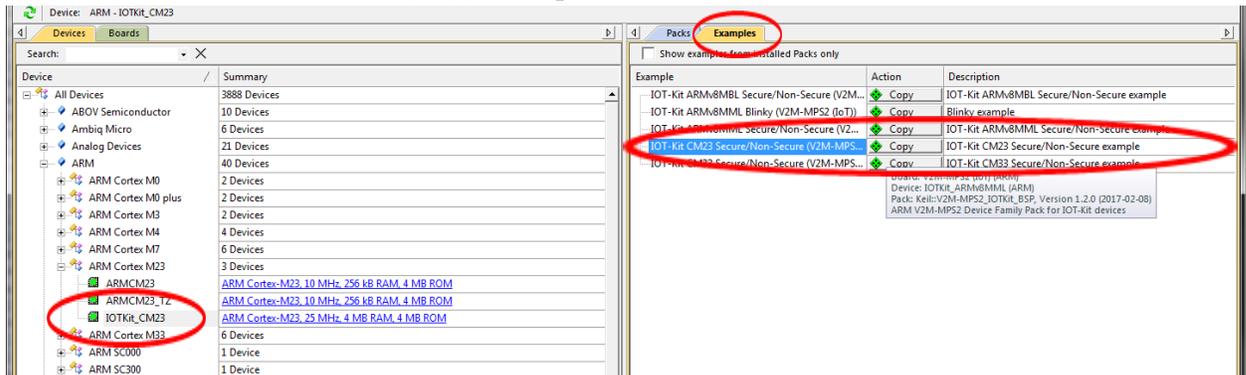
2. Then make sure you have the latest packs loaded, CMSIS.5.0.1-dev5.pack and V2M-MPS2_IOTKit_BSP.1.2.0.pack packs by checking the versions listed and that the “Up to date” button is shown next to each one...



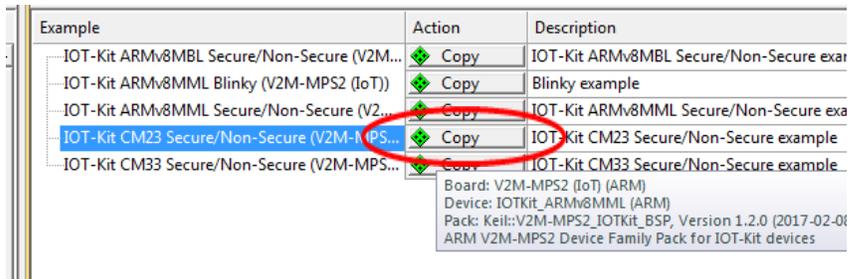
Copy and Run the Example Application

Once you have these packs loaded you can go to the examples tab and export the latest “TrustZone for ARMv8-M RTOS” (µVision Simulator)” example...

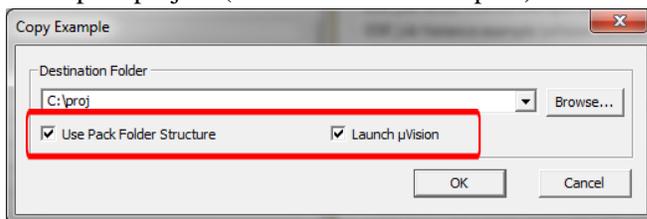
1. On the left side, click the “Devices” tab, then select “ARM” then “ARM Cortex-M23” then “IOTKit_CM23” as the device.
2. On the right side, switch to the “Examples” tab. If needed, scroll down till you see the “IOT-Kit CM23 Secure/Non-Secure (V2M-MPS2 (IoT))” example...



Click the “Copy” button next to the “IOT-Kit CM23 Secure/Non-Secure (V2M-MPS2 (IoT))” example project...



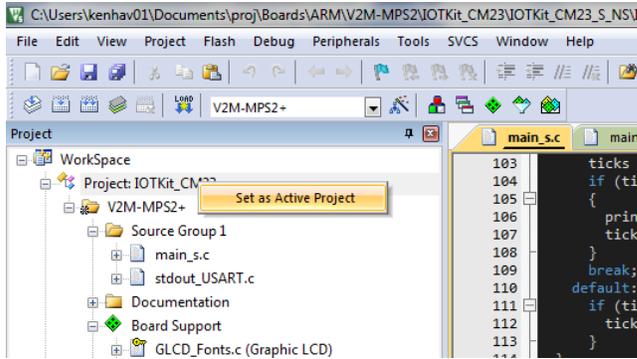
3. ...for running the example application, it is more convenient to check the box to ‘Launch µVision’ with the copied project (Note the destination path)...



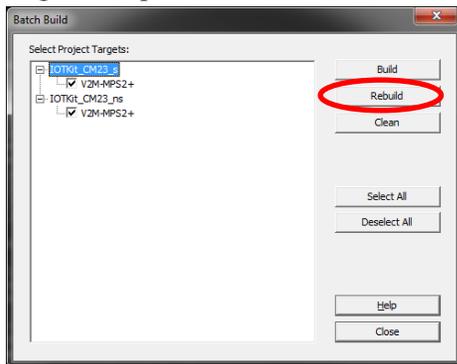
4. Close the Pack Installer and, if you didn’t select the ‘Launch µVision’ check box, open the newly downloaded project in MDK by double clicking the “IOTKit_CM23_s_ns.uvmpw” multi-project file in the folder located in the following directory path noted above...

<load point>\proj\Boards\ARM\V2M-MPS2\IOTKit_CM23\IOTKit_CM23_S_NS\

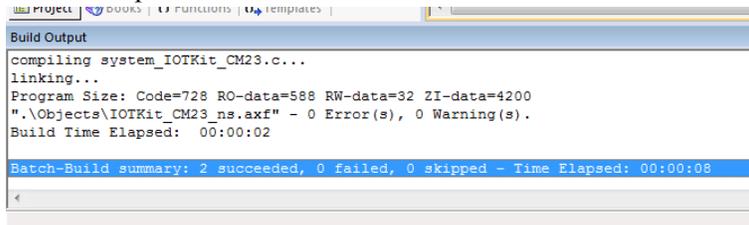
- Make sure that the secure project 'IOTKit_CM23_s' is the active project by right clicking to select "Set as Active Project" and "V2M-MPS2+" is selected from the target drop down menu...



- Click the "Batch Build" button  on the icon bar near the top left of the MDK window, select all targets and press "Rebuild".



Everything should compile with no errors and no warnings. The status of the build will be shown in the "Build Output" window at the bottom...



- If you haven't already done so, make sure you have updated to the latest Cortex-M23 IoT Kit FPGA image for the MPS2+ board. The FPGA image files for the Versatile Express MPS2+ (V2M-MPS2-0318C) development board is available from the web at www.arm.com/mps. About ¾ of the way down the page you will see the .zip archive files. In this example we will be using the "Cortex-M23_IoT_kit_2_0.zip" file...

Download Cortex-M33 example IoT FPGA image for MPS2+

Acceptance of a EULA is required before download

[Cortex-M33_IoT_kit_2_0.zip](#) 13.78 MB

Download Cortex-M23 example IoT FPGA image for MPS2+

Acceptance of a EULA is required before download

[Cortex-M23_IoT_kit_2_0.zip](#) 134.89 MB

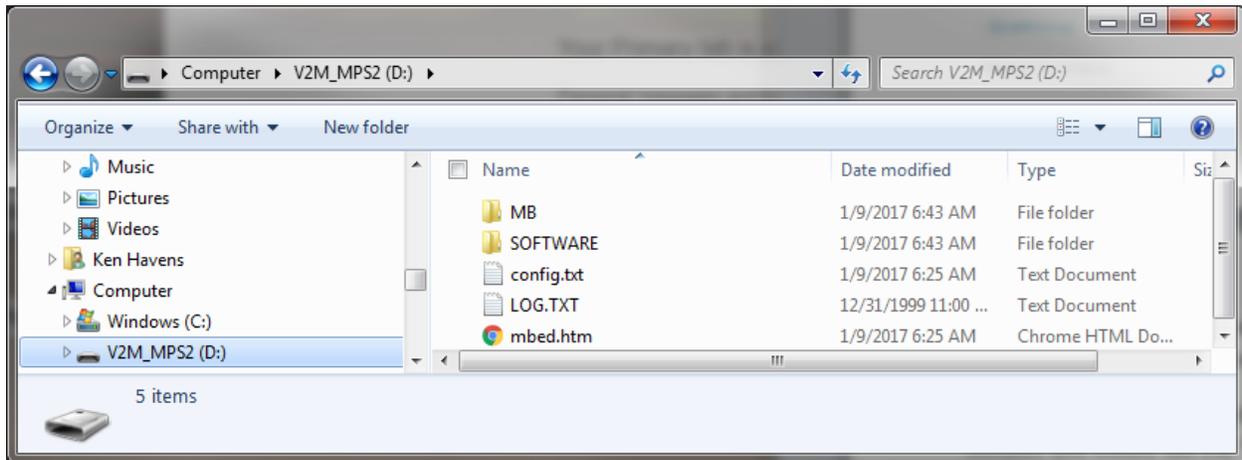
Cortex-M Prototyping System (MPS2) support DVD 3.1

Acceptance of a EULA is required before download

[CMPS-3-1.zip](#) 147.53 MB

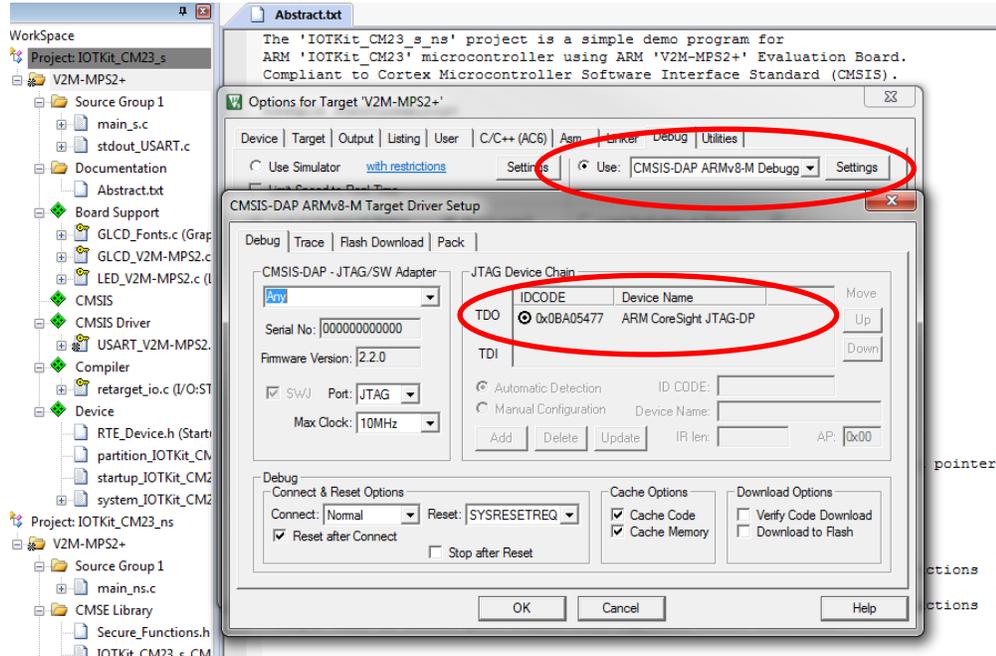
Adapter for Arduino

8. Locate the archived file named “Cortex-M23_IoT_kit_2_0.zip” that you downloaded in the previous step and unpack the archive in a safe place for the next step.
9. Connect the MPS2 board’s power, USB cable and debugger first. Then press the power on switch. After a few minutes you should see the “V2M-MPS2” displayed as an available drive on your host system. It is recommended to copy the entire existing SD Card image to a safe place before erasing. Keep in mind that you can always download the default installation files from the original installer disk that came with the board or from the above website where you downloaded the IoT Kit image. However, if you’ve made any modifications to the files on the boards, it is best to save those away in a safe place. After doing this, erase the entire contents of the SD Card and copy the contents of the recovery directory to the SD Card on the board.
`<loadpoint>\Cortex-M23_IoT_kit_2_0\boards\Recovery*` copied to “V2M_MPS2” as shown on the host.
10. Verify the files were copied to the board by opening the drive on the host and you should see something similar to the below...



11. At this point I recommend powering down the board and powering it back up (Remember to press the power “ON” button) and you should see the splash screen “ARM Versatile Hardware Platforms V2M-MPS2 Motherboard” on the board’s display.

12. Let's check that you have the debug connection configured correctly with in MDK. Press Alt-F7 to open the "Options for Target" dialog box and select the "Debug" tab. On the right side make sure that "CMSIS_DAP ARMv8-M Debugger" is selected in the debugger drop down. To check that it is connected, select the "Settings" button and you should see "ARM CoreSight JTAG-DP" under device name as below...



Click "OK" in both dialog boxes to save any changes and close them out.

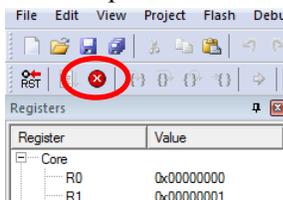
13. Click the Debug button at the top to enter the debug session...



14. Click run and you should see the LCD screen update and the LED lights begin to blink on and off on the board...



15. Press stop...



16. As an example, let's walk through the process of going from non-secure to secure and back, by placing a break point at or near line 89 in the main_ns.c file then press "Run" to stop at that breakpoint...

```

56     }
57   }
58 }
59
60
61 static uint32_t x;
62 /*-----*/
63 Main function
64 *-----*/
65 int main (void)
66 {
67     uint32_t i;
68
69     /* exercise some core register from Non Secure Mode */
70     x = __get_MSP();
71     x = __get_PSP();
72
73     /* register NonSecure callbacks in Secure application */
74     Secure_LED_On_callback(NonSecure_LED_On);
75     Secure_LED_Off_callback(NonSecure_LED_Off);
76
77 #if 0
78     LED_Initialize ();           /* already done in S part */
79 #endif
80
81     SystemCoreClockUpdate();
82     SysTick_Config(SystemCoreClock/100); /* Generate interrupt each 10 ms */
83
84     while (1) {
85         LED_On (5u);
86         for (i = 0; i < 0x100000; i++) __NOP();
87         LED_Off(5u);
88         for (i = 0; i < 0x100000; i++) __NOP();
89         Secure_LED_On(4u);
90         for (i = 0; i < 0x100000; i++) __NOP();
91         Secure_LED_Off(4u);
92         for (i = 0; i < 0x100000; i++) __NOP();
93     }

```

17. From this point, click in to the "Disassembly" window and single step (F11) through the code. After around six steps you'll notice you see the "SG" instruction which is the secure gate instruction. After executing that instruction you should see the debugger update from "Non-Secure Thread" to "Secure Thread" in the register window...

Register	Value
R0	0x00000004
R1	0x00000020
R2	0x00000000
R3	0x002004A8
R4	0x28200004
R5	0x00100000
R6	0x00000000
R7	0x00000000
R8	0x002002D0
R9	0x002002D0
R10	0x00200524
R11	0x00200524
R12	0x002004A8
R13 (SP)	0x28201078
R14 (LR)	0x00200487
R15 (PC)	0x10001BB0
xPSR	0x21000000

```

Disassembly
0x10001BB0 E97FE97F SG
0x10001BB4 F7FFB8CA B Secure_LED_On (0x1000134C)
0x10001BB8 E97FE97F SG
0x10001BBC F7FFBBD4 B Secure_LED_On_callback (0x10001368)
0x10001BC0 E97FE97F SG
0x10001BC4 F7FFBBD4 B Secure_printf (0x1000137C)
0x10001BC8 0000 MOVS r0,r0
0x10001BCA 0000 MOVS r0,r0

```

After the “SG” Instruction...

The screenshot shows a disassembler interface with two main panes. The left pane, titled 'Registers', lists various registers and their values. The right pane, titled 'Disassembly', shows assembly instructions with their addresses and mnemonics.

Registers Pane:

Register	Value
R0	0x00000004
R1	0x00000020
R2	0x00000000
R3	0x002004A8
R4	0x28200004
R5	0x00100000
R6	0x00000000
R7	0x00000000
R8	0x002002D0
R9	0x002002D0
R10	0x00200524
R11	0x00200524
R12	0x002004A8
R13 (SP)	0x38001180
R14 (LR)	0x00200486
R15 (PC)	0x10001BB4
xPSR	0x21000000

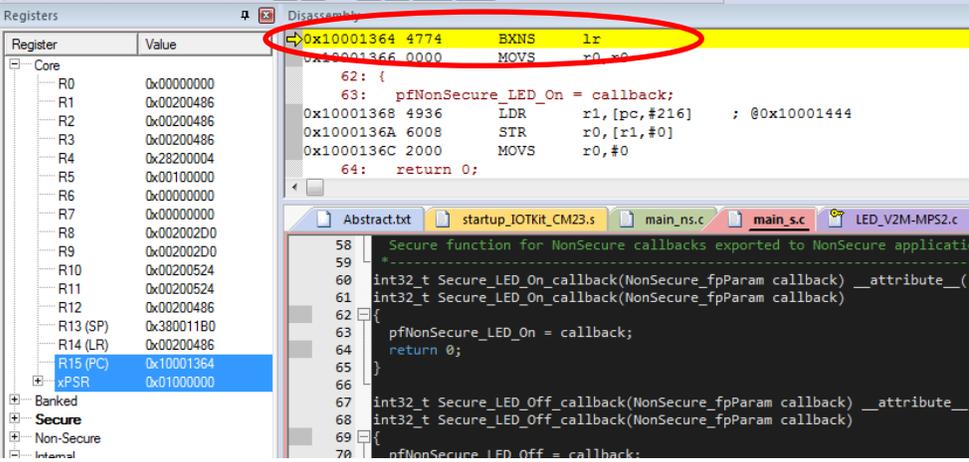
Disassembly Pane:

```
0x10001BB0 F97FF97F SG
0x10001BB4 F7FFBBCA B Secure_LED_On (0x1000134C)
0x10001BB8 E97FF97F SG
0x10001BBC F7FFBD4 B Secure_LED_On_callback (0x10001368)
0x10001BC0 E97FE97F SG
0x10001BC4 F7FFBDDA B Secure_printf (0x1000137C)
0x10001BC8 0000 MOVVS r0,r0
0x10001BCA 0000 MOVVS r0,r0
```

Code Editor Pane:

```
56 }
57 }
58 }
59 }
60 }
61 static uint32_t x;
62 /*-----
63 Main function
64 *-----*/
65 int main (void)
66 {
67     uint32_t i;
68
69     /* exercise some core register from Non Secure Mode */
70     x = __get_MSP();
71     x = __get_PSP();
72
73     /* register NonSecure callbacks in Secure application */
74     Secure_LED_On_callback(NonSecure_LED_On);
```

18. Continuing the single step in the disassembly window for approximately 21 steps and you should see the branch back to non-secure instruction “BXNS” which when executed will take you back to the “Non-Secure Thread” state in the non-secure main program.



19. More information about CMSIS-Core for ARMv8-M can be found in the included documentation in the Manage RTE component “CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M” or by browsing to it in your installation at...

```
<install point> /ARM/PACK/ARM/CMSIS/5.0.1/CMSIS/Documentation/Core/html/index.html
```

20. For more information about the Cortex-M23 IoTKit FPGA image for the MPS2 board look in the following directories of the Cortex-M23_IoT_kit_2_0.zip file...

```
\Cortex-M23_IoT_kit_2_0\app_notes\AN519\docs\ DAI0519B_example_iot_kit_subsystem_for_v2m_mps2.pdf
\Cortex-M23_IoT_kit_2_0\boards\Docs\ARMv8-M_IoT_Kit_UG_M23.pdf
```

Notes

This is a very simple example but I hope it helps you understand Secure and Non-secure operations better.

Some notes:

1. Note that the MDK tools do two incremental loads before starting, one each for the secure and non-secure domains.
2. Since, in most cases, the user will need to load two AXF images, one for secure and one for non-secure, this can be done at boot up of the board via the images.txt file on the SD Card. An example of what I have is below (D:\MB\HBI0263C\AN519\images.txt for the CM23 IoT Kite image)...

```
[IMAGES]
TOTALIMAGES: 2 ;Number of Images (Max: 32)
IMAGE0ADDRESS: 0x10000000 ; Secure address start
IMAGE0FILE: \SOFTWARE\IOT_s.axf ; Secure image
IMAGE1ADDRESS: 0x00200000 ; Non-secure address start
IMAGE1FILE: \SOFTWARE\IOT_ns.axf ; Non-secure image
```

Keep in mind that the file system on the MPS2 board does not support long file names and needs to stick with the 8.3 name format.