Hands-on_2_LPC55S69 Secure Lab Instructions

Table of Contents

Objectives

In this lab, you will learn:

- What is the secure Trusted Execution Environment on the LPC55S69
- How to use the CM33 Trust Zone
- Non-Secure ↔ Secure environment transitions

Hardware

NXP LPCxpresso55S69 – LPC55S69-EVK
Micro-USB cable
Jumper

Trusted Execution Environment TEE high level description

In this lab you will gain a basic understanding of the benefits of using the secure functions in the LPC55S69 MCU. Specifically, the use of the Trusted Execution Environment TEE.

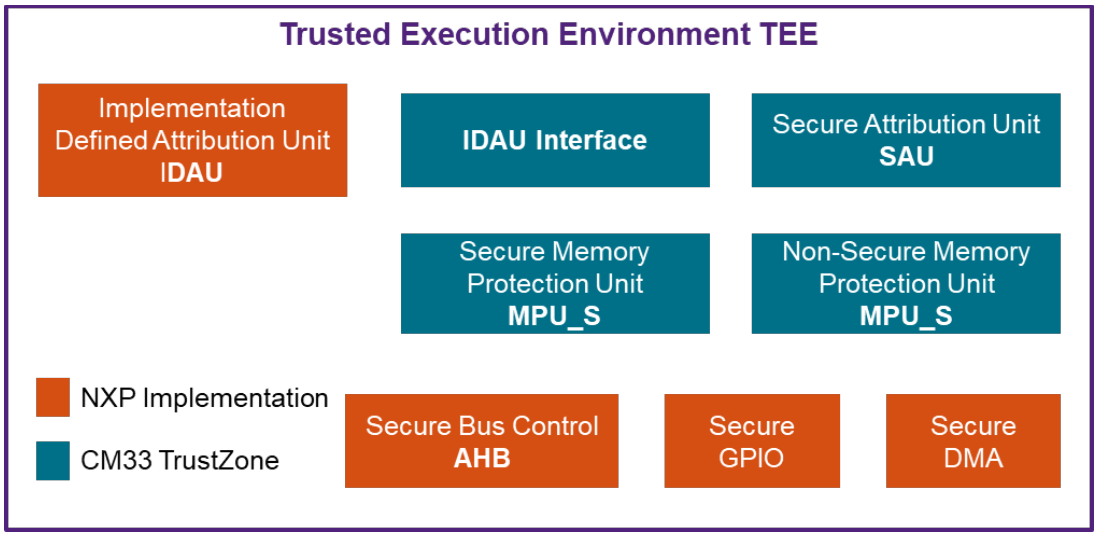


Figure 1. *TEE Block Diagram*

TrustZone

TrustZone is an optional CM33 security extension available in LPC55S69. This feature was designed to provide a foundation for improved system security. The MCU has Secure **S** and Non-Secure **NS** states.

Secure **S**

Memory and peripherals that are only accessible by S-software or S-masters. Illegitimate accesses that are made by NS software to S memory are blocked and raise an exception. A CPU in S-state only executes S-code, S-data is only accessible through S-code.

Non-Secure Callable **NSC**

Is a particular type of S location that is permitted to hold an Secure Gateway **SG** instruction to enable software to transition from NS to S state

A portion of S-memory can be marked as Non-Secure Callable NSC memory which the main functionality is to provide cross-domain calls. NSC memory regions contain tables of small branch veneers or entry points.

Having veneer tables like entry points inside the NSC memory means that the S binary is never exposing to the NS world.

Non-Secure **NS**

Addresses used for memory and peripherals accessible by all software running on the device. NS-data is accessible through S-code or NS-code.

Secure Attribution Unit **SAU**

Controls the security state of a memory region, the LPC55S69 provides 8 configurable SAU regions. The memory partition into Secure and Non-Secure regions is independent of the security state that the processor executes in.

NXP Implementation

Implementation Defined Attribution Unit **IDAU**

The software can define up to eight SAU regions. If these regions are not enough, the application can use a device-specific controller logic like IDAU.

NXP IDAU implementation of ARM TrustZone for core0 involves using address bit 28 to divide the address space into potential S and NS regions. Address bit 28 is not decoded in memory access hardware. Each physical has two potential addresses a S-address with bit 28 set and a NS-address with b28 clear. For example, RAM0 start address is 0x3000_0000 in the S-world (b28=1) or 0x2000_0000 in the NS world (b28=0)

Figure 2 shows how the LPC55S6x IDAU divides the memory into S / NS using the address b28. There are two alias locations for every memory or peripheral address.

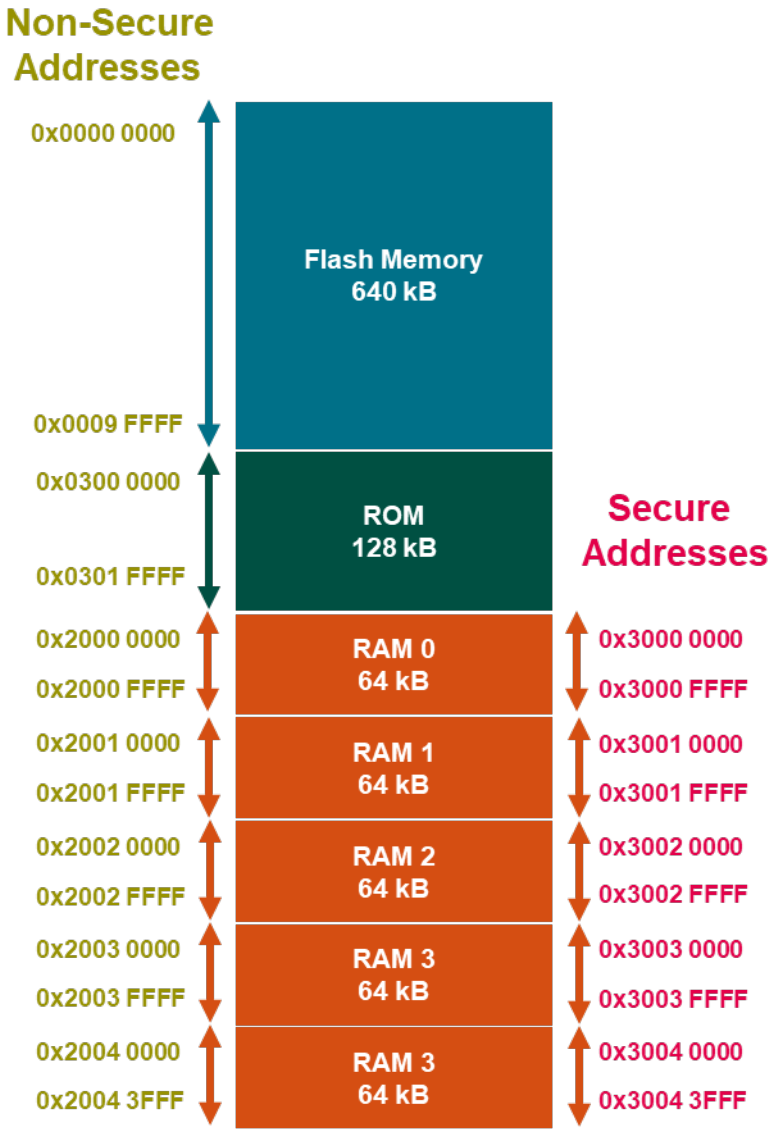


Figure 2. *LPC55S69 Memory Map*

The memory partition into Secure and Non-Secure regions is independent of the security state that the processor executes in.

Secure Bus Controller

The LPC55S69 use a matrix of secure bus controller modules to manage the data flow in the MCU. A combination of a Peripheral Protection Checker **PPC**, the Memory Protection Checker **MPC**, the Master Security Wrapper **MSW.**

The **secure AHB** controller is a module on LPC55S69 that allows programming security attributes for all PPCs, MPCs, or MSWs. Secure AHB controller provides a second protection layer for safe, trusted execution at system-level. With secure AHB each peripheral is capable of configuring individual access rules.

Secure DMA

For TrustZone a DMA can be configured in S-mode for secure thread.

Secure GPIO

Secure GPIO has the same functions as normal GPIO. However, the access rules to this Secure GPIO for different security levels are configured through the Secure AHB controller which can only be accessed in Secure state.

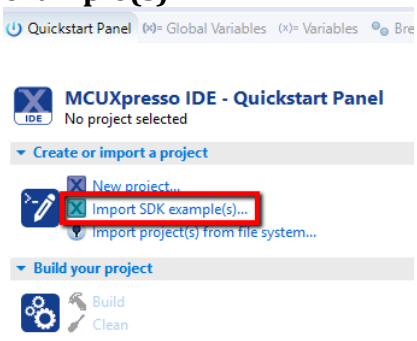
Running the lab

Import the MCUXpresso hello_world lab projects

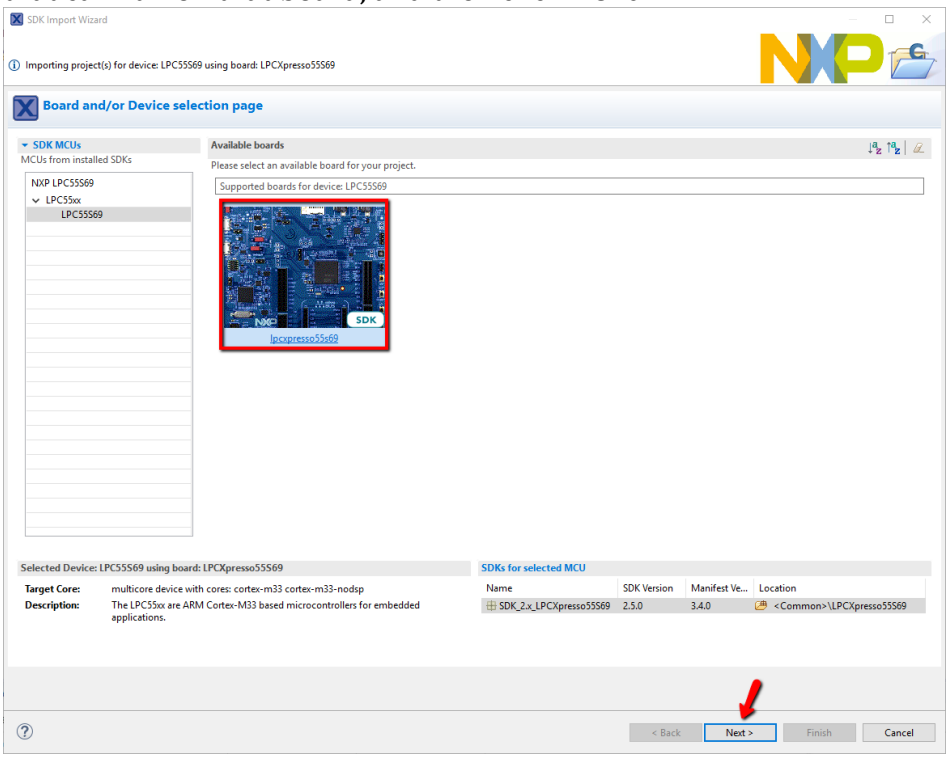
This lab uses two projects: hello_world_ns and hello_world_s. Both projects use the LPC55SX6 CM33 Core0, **hello_world_ns** is the non-secure application, and **hello_world_s** contains all the TEE secure software and configuration.

1. Open MCUXpresso IDE 11.0.0 or later.

2. Find the **Quickstart Panel** in the lower left-hand corner, select **Import SDK example(s)...**

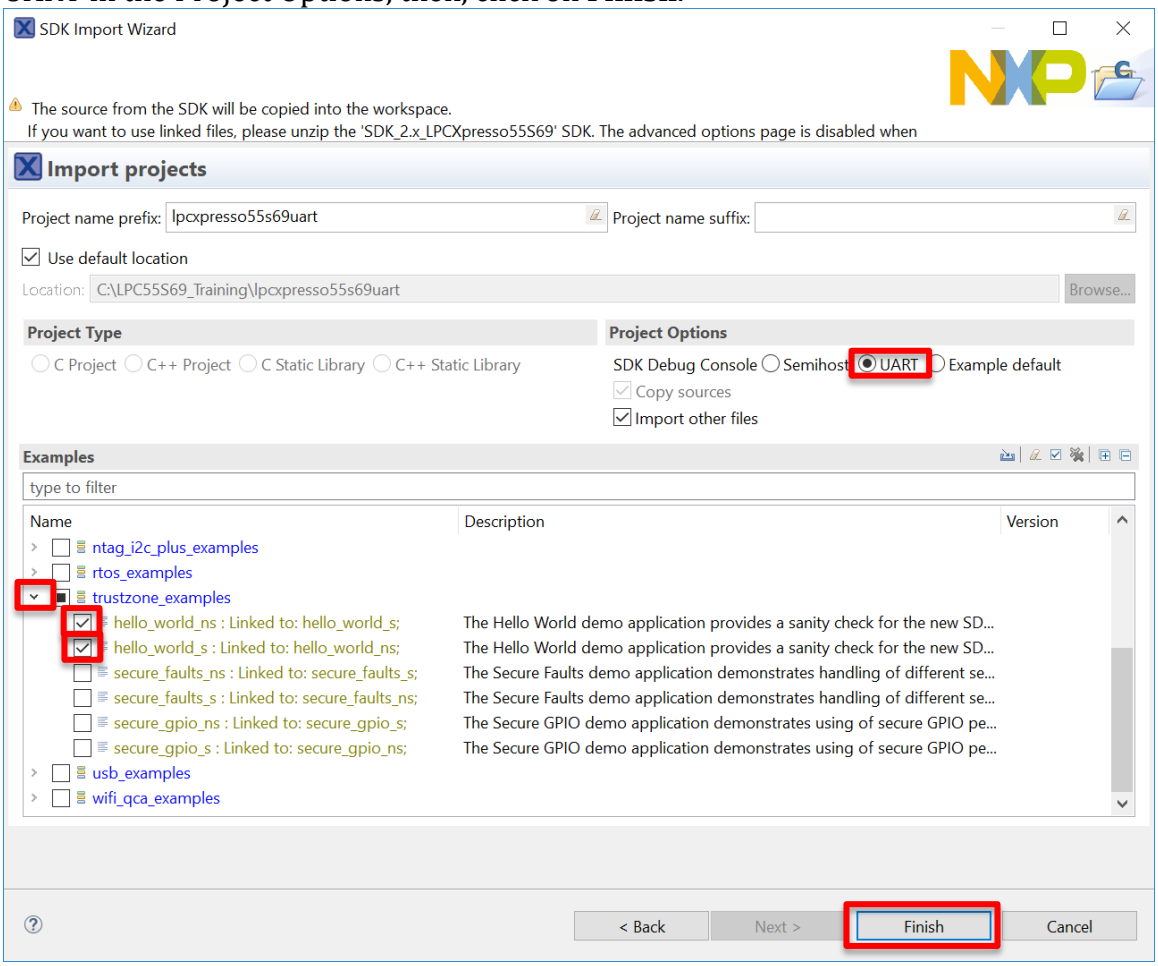


3. Click on the **LPCXpresso55S69** board to select that you want to import an example that can run on that board, and then click **Next**

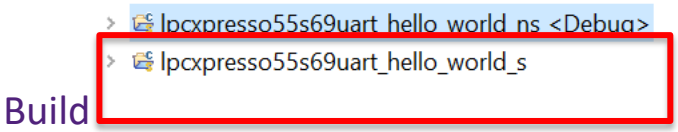


4. Use the arrow button to expand the **trustzone_examples** category, and then click the checkbox next to **hello_world_ns** & **hello_world_s** to select both projects. Select

UART in the Project Options, then, click on **Finish**.



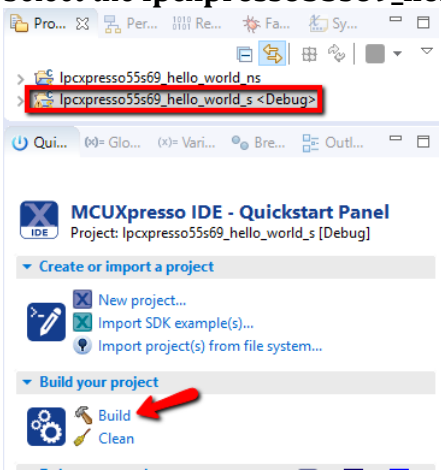
5. You should now see two new projects loaded into the workspace as shown below



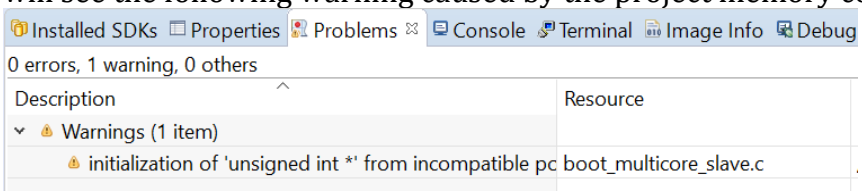
Build

6. Now it's time to compile.

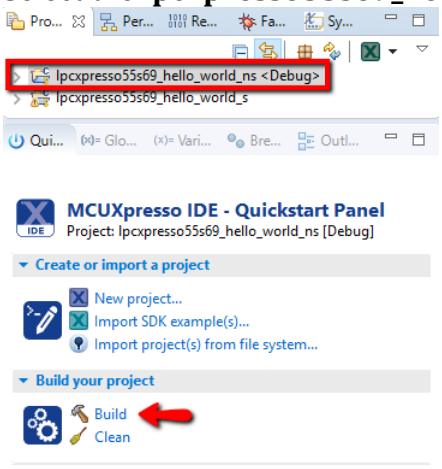
Select the **lpcxpresso55s69_hello_world_s** project and click **Build**



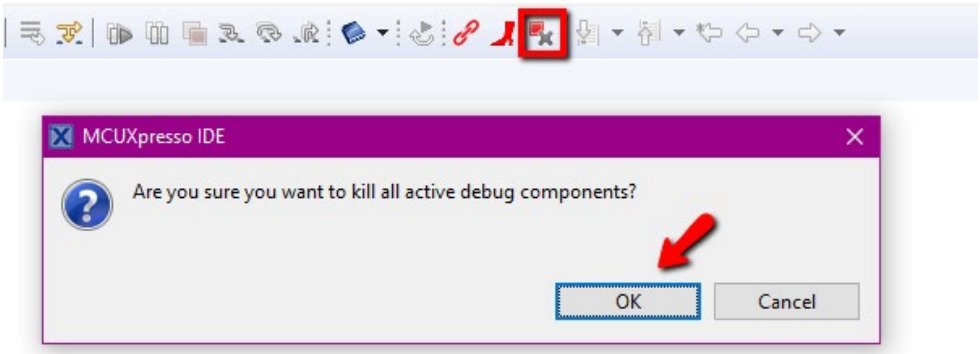
Wait until lpcxpresso55s69_hello_world_s finish building without any errors, you will see the following warning caused by the project memory configuration.



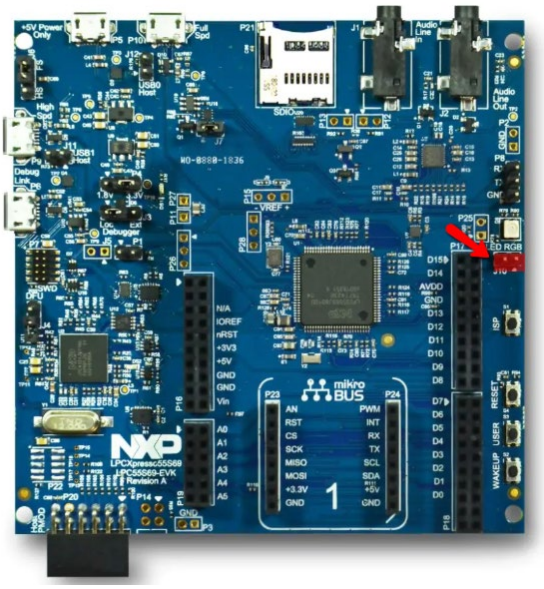
Select the **lpcxpresso55s69_hello_world_ns** project and click **Build**



7. To close the current debug session click **Cleanup Debug** → select **OK** when the window pop-up asks to kill active debug components

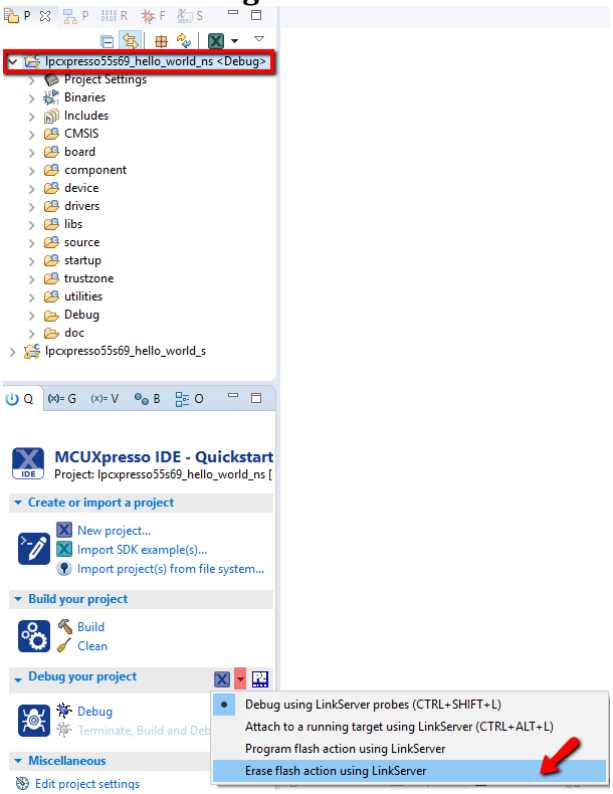


8. Close jumper J10 to use the In-System-Programming **ISP** over UART.

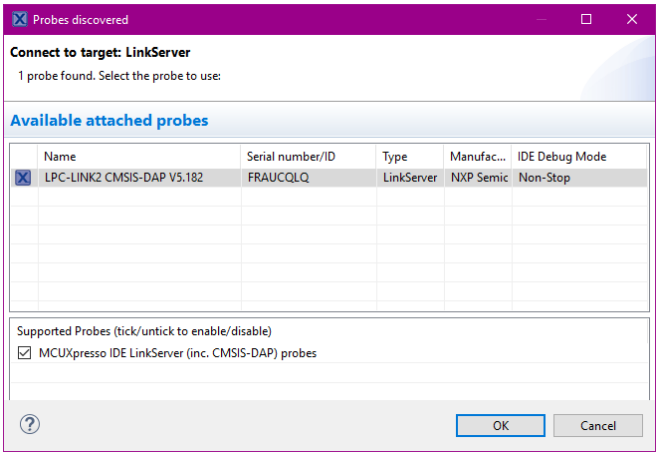


9. Connect the Debug Link port from your LPCXpresso55S69 to your PC using a micro-USB cable.

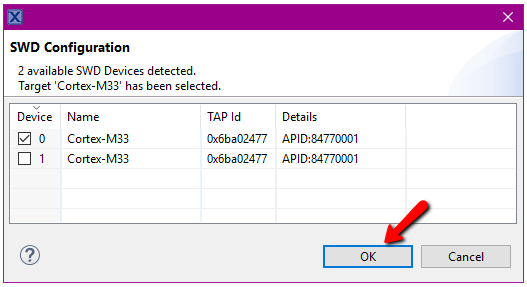
10. To mass erase your device, select the **lpcxpresso55s69_hello_world_ns** at the project explorer, then using the Quick Start Panel look for the Debug your project section, expand the **Debug with Link Server drop-down menu** then choose **Erase flash action using LinkServer**



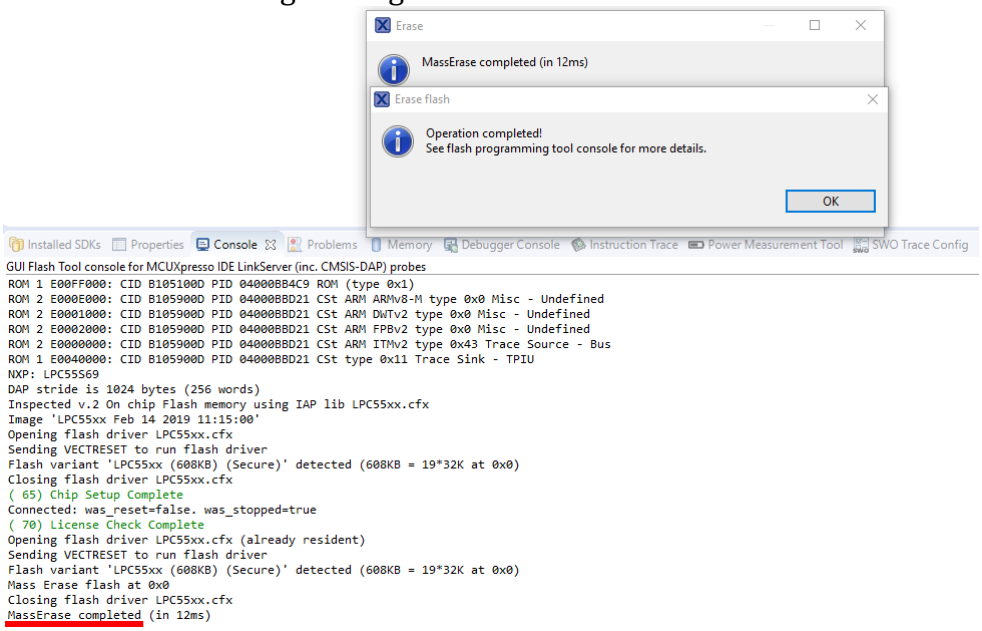
11. Select your probe and click **OK** to start programming your board.



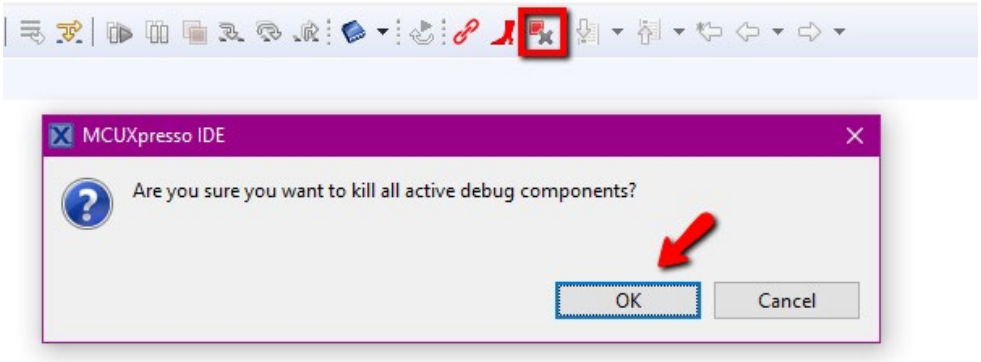
12. Select the **Cortex-M33 core 0** and click **OK**



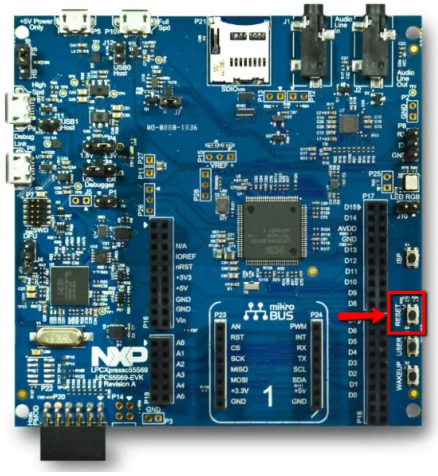
13. Wait until the program is successfully erased, to verify that the MassErase worked look for the following message at the console window



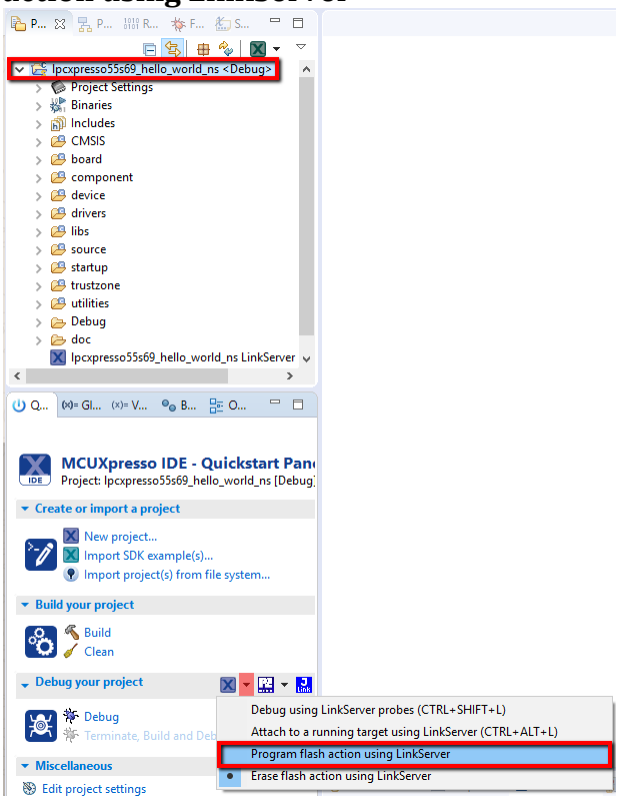
14. Click **OK** at the Operation completed! Window
15. To close the current debug session click **Cleanup Debug** → select **OK** when the window pop-up asks to kill active debug components



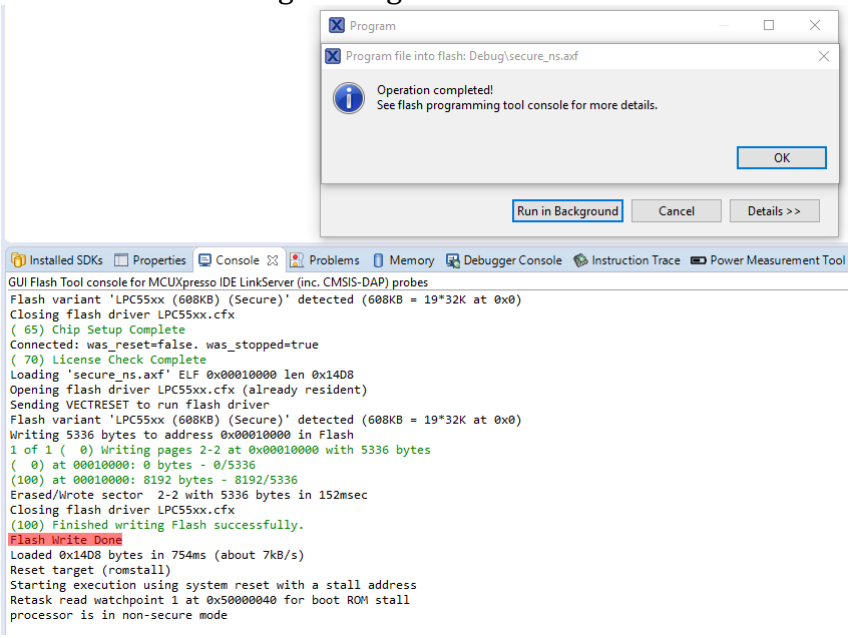
16. Remove the J10 jumper, then do a pin **RESET** using **S4** push-button



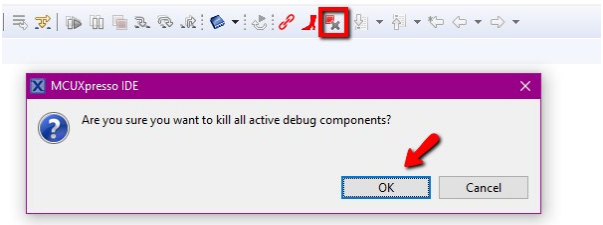
17. To flash the NS project, select the **lpcxpresso55s69_hello_world_ns** at the project explorer, then using the Quick Start Panel look for the Debug your project section, expand the **Debug with Link Server drop-down menu** then choose **Program flash action using LinkServer**



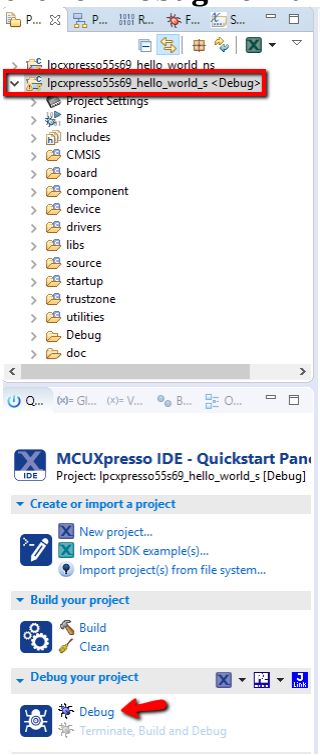
18. Wait until the program is successfully flashed, to verify that the flashing worked look for the following message at the console window



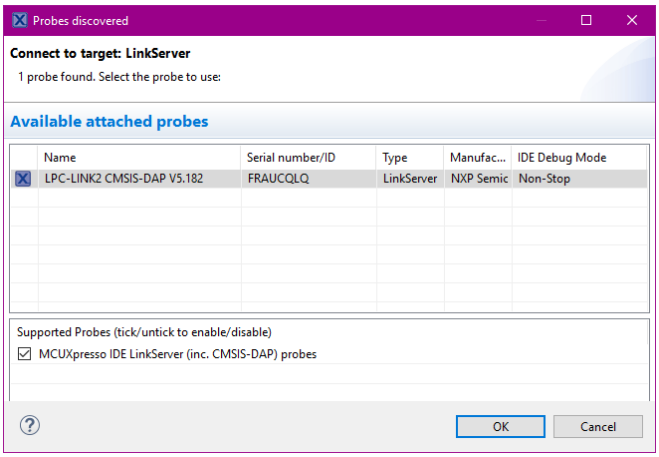
19. Click **OK** at the Operation completed! Window
20. To close the current debug session click **Cleanup Debug** → select **OK** when the window pop-up asks to kill active debug components



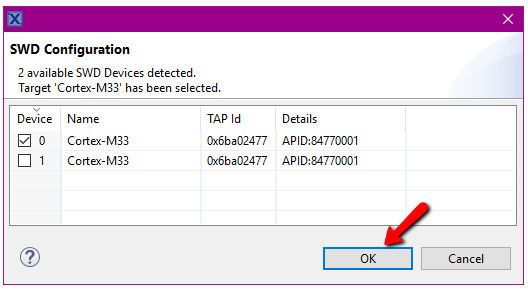
21. To program the secure project. Select the **lpcxpresso55s69_hello_world_s** project, click on **Debug** from the Quick Start Panel



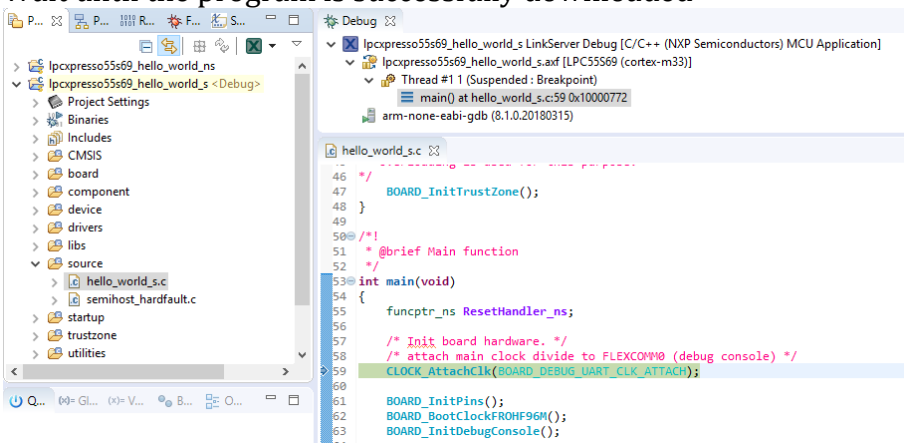
22. Select your probe and click **OK** to start programming your board.



23. Choose the **Cortex-M33 core 0** and click **OK**

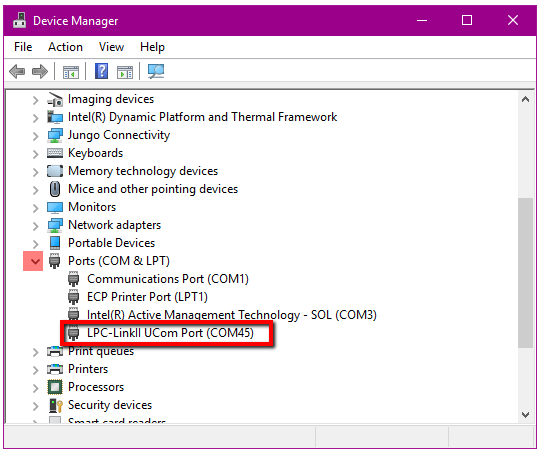


24. Wait until the program is successfully downloaded



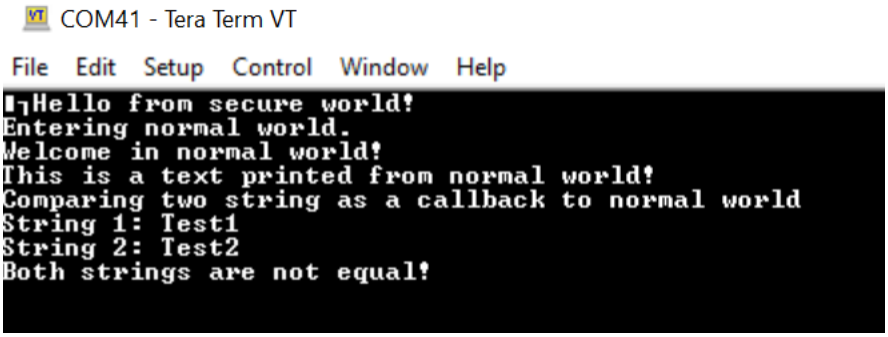
At this moment the LPCXpresso55S69 has both projects in flash, but the IDE isn't able to debug inside the NS project. Wait until the program is successfully downloaded

25. Open the **Device Manager** on your Windows PC and identify the COM port of your board



Open a Terminal emulator software (TeraTerm, PuTTY) and connect the COM port using the following settings `115200 baudrate, 8 data bitsm, No parity, One stop bit, No flow control`

26. After pressing F8 (Resume)  the Comm port will display the following.



TrustZone configuration

To create a TrustZone application is necessary to divide the software into two projects one for Secure called lpcxpresso55s69_hello_world_s and a second one for Non-Secure called lpcxpresso55s69_hello_world_ns. One of the main differences between these two projects resides in the memory configuration.

Figure 3 and Figure 4 show how both projects use their corresponding IDAU address without overlapping the physical locations. The veneer table needs to use a SAU region to be defined as a NSC.

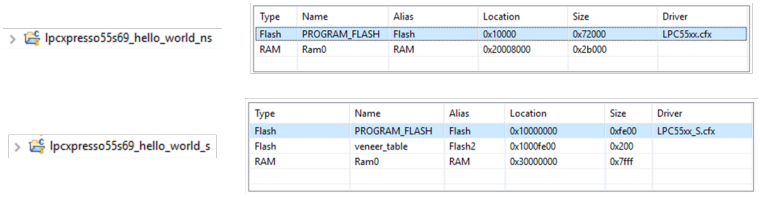


Figure 3. *lpcxpresso55s69_hello_world_ns & lpcxpresso55s69_hello_world_s memory configuration*

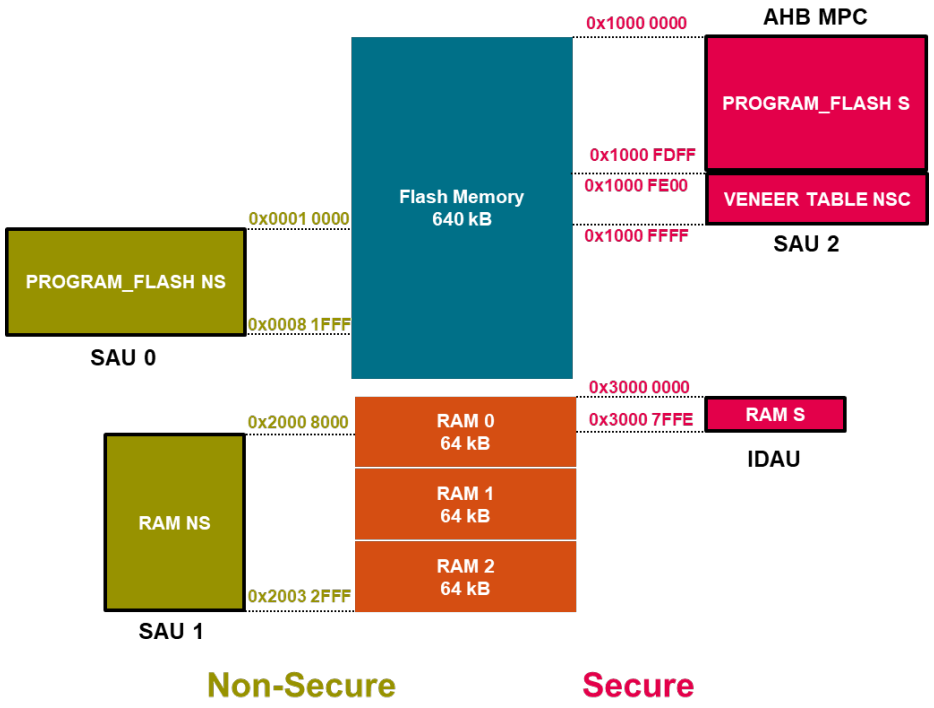


Figure 4. *Secure Lab Memory Map*

Inspect the Trusted Execution Environment(TEE) Tool

To assist you in the configuration of the environment NXP has developed a tool called the TEE configuration Tool. Let's look at it now.

1) In the MCUXpresso menu open the MCUXpresso IDE Users Guide. Once that is open navigate to the MCUXpresso Config Tool User's Guide → Trusted Executions Tool. Here you can learn about how the tool will assist you in configuration of your own application.
2) Below is an exert of the TEE user's guide.

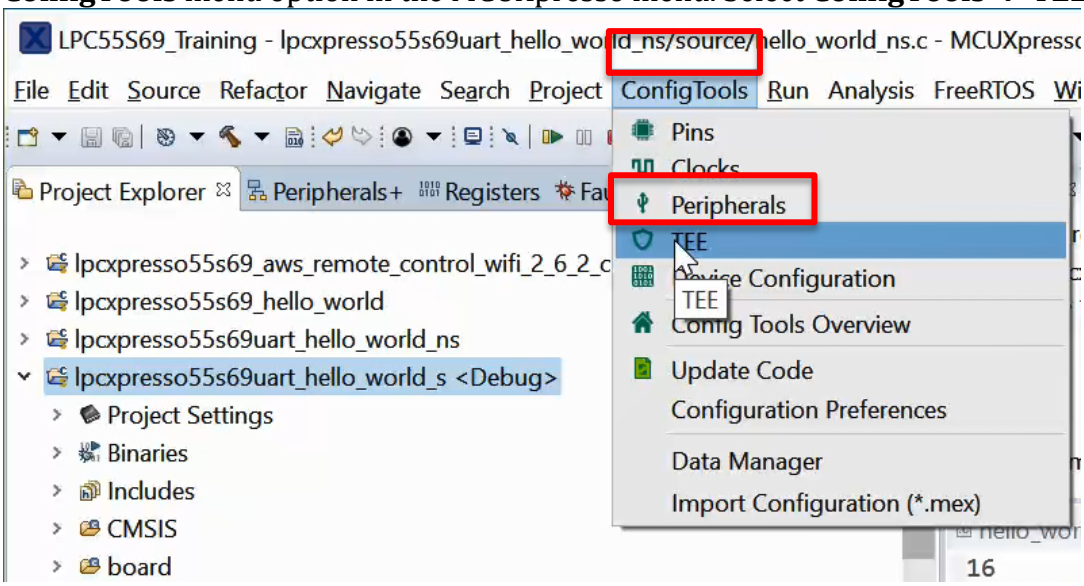
In the **Trusted Execution Environment**, or **TEE** tool, you can configure security policies of memory areas, bus masters, and peripherals, in order to isolate and safeguard sensitive areas of your application.

You can set security policies of different parts of your application in the **Security access configuration** and its sub-views, and review these policies in the **Memory map** and **Access overview** views. Use the **User Memory Regions** view to create a convenient overview of memory regions and their security levels.

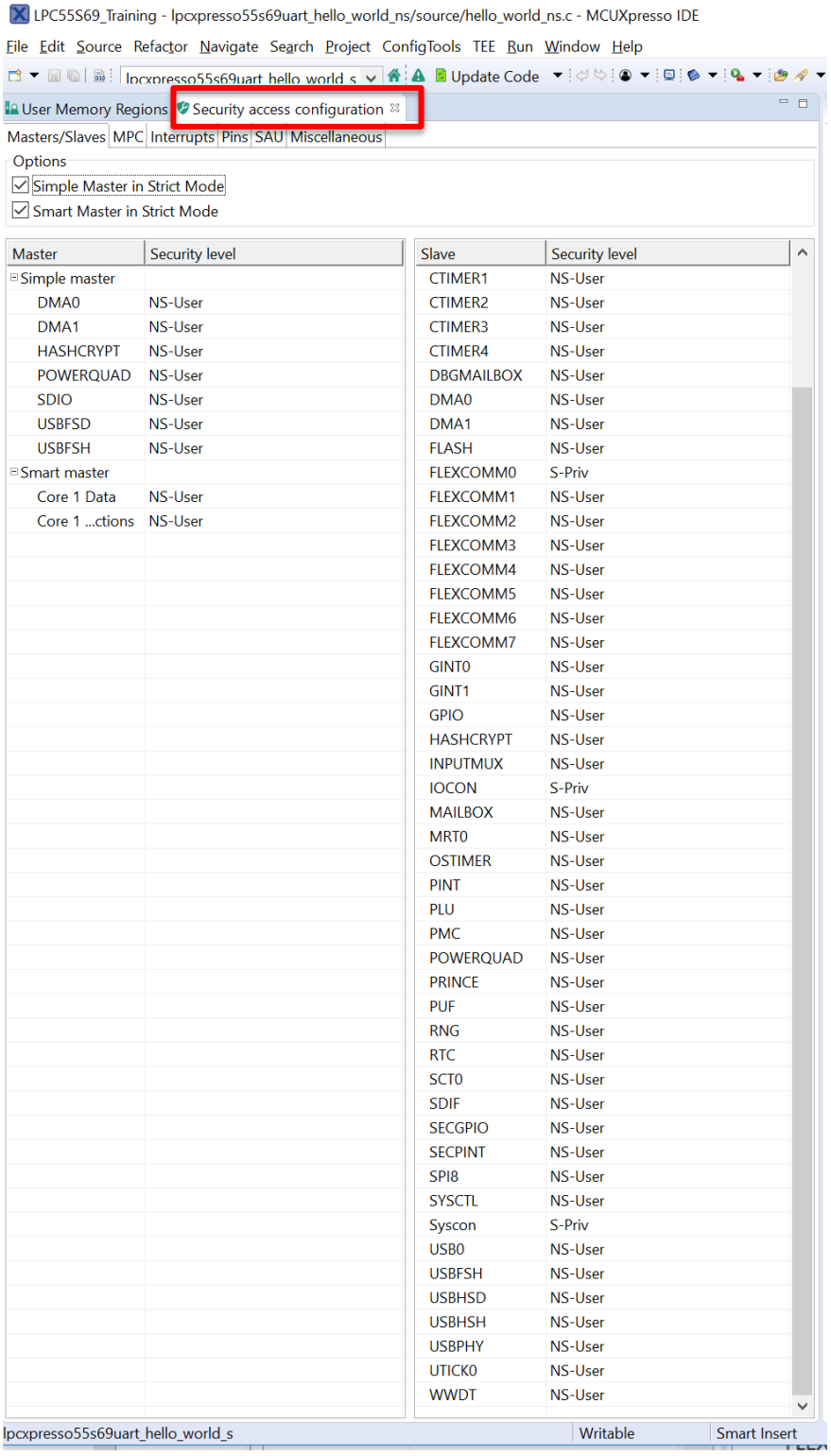
You can also view registers handled by the **TEE** tool in the **Registers** view, and inspect the code in the **Code Preview** tool.

Note: In order for your configuration to come into effect, make sure you have enabled the relevant enable secure check option in the **Miscellaneous** sub-view of the **Security access configuration** view.

1. At the project explorer select **lpcxpresso55s69_hello_world_s,** then look for the **ConfigTools** menu option in the MCUXpresso menu. Select **ConfigTools → TEE**



2. Select the **Security Access Configuration** tab and inspect the security levels of the cores and each module.



The secure project configures TrustZone after every RESET. The file trustzone→tzm_config.c contains one function **BOARD_InitTrustZone**, which configures

complete TrustZone environment. It includes SAU, MPU's, AHB secure controller and some TrustZone related registers from System Control Block. This function is called from **`SystemInitHook`** function, it means during system initialization and prior jumping into main.

SAU configuration

SAU Region	Name	Type	Start	End
0	CODE_FLASH_NS	NS	0x0001 0000	0x0008 1FFF
1	DATA_RAM_NS	NS	0x2000 8000	0x2003 2FFF
2	CODE_FLASH_NSC	NSC	0x1000 FE00	0x1000 FFFF

AHB MPC

Secure Flash – 0x00000000 to 0x0000FFFF – 64 kB
Secure RAM0 – 0x20000000 to 0x20007FFF – 32 kB
Secure RAM1 – 0x20010000 to 0x20017FFF – 32 kB

AHB PPC

Secure and privileged user access only: SYSCON, IOCON, FLEXCOMM0

S → NS → S transition

S → NS

The S project sets NS main stack, vector table, then ends jumping to the NS project

```
/* typedef for non-secure callback functions */
typedef void (*funcptr_ns) (void) __attribute__((cmse_nonsecure_call));

/* Set non-secure main stack (MSP_NS) */
__TZ_set_MSP_NS(*((uint32_t *)(NON_SECURE_START)));

/* Set non-secure vector table */
SCB_NS->VTOR = NON_SECURE_START;

/* Get non-secure reset handler */
ResetHandler_ns = (funcptr_ns)(*((uint32_t *)((NON_SECURE_START) + 4U)));

/* Jump to normal world (Correct way compare to Test 1) */
ResetHandler_ns();

while (1)
{
    /* This point should never be reached */
}
```

NS → S

After RESET the TurstZone AHB PPC configuration sets FLEXCOMM0 at lpcxpresso55s69_hello_world_s → trustzone → tzm_config.c with secured access only. FLEXCOMM0 is the UART handling PRINTFs. Therefore, just secure software is capable of printing things at the terminal.

```
/* Set FLEXCOMM0 as secure */
AHB_SECURE_CTRL->SEC_CTRL_AHB0_0_SLAVE_RULE = AHB_SECURE_CTRL_SEC_CTRL_AHB0_0_SLAVE_RULE_FLEXCOMM0_RULE(0x3U);
```

If you try to PRINTF from the NS project, a hard fault event will occur because you are trying to access to a peripheral marked as S from an NS environment. To prevent the hard fault event, the application needs a safe way to call S code, this gets done through the NSC Secure Gateway SG API.

Figure 4 shows how to do a NS → S transition to PRINTF. veneer_table.c defines all the S entry functions exported to the NS world. Both projects S and NS require the veneer_table.h header file. Besides sharing the veneer_table.h, the lpcxpresso55s69_hello_world_ns creates a connection to the lpcxpresso55s69_hello_world _s by adding the lpcxpresso55s69_hello_world_s_CMSE_lib.o library as a lpcxpresso55s69_hello_world_ns linker input.

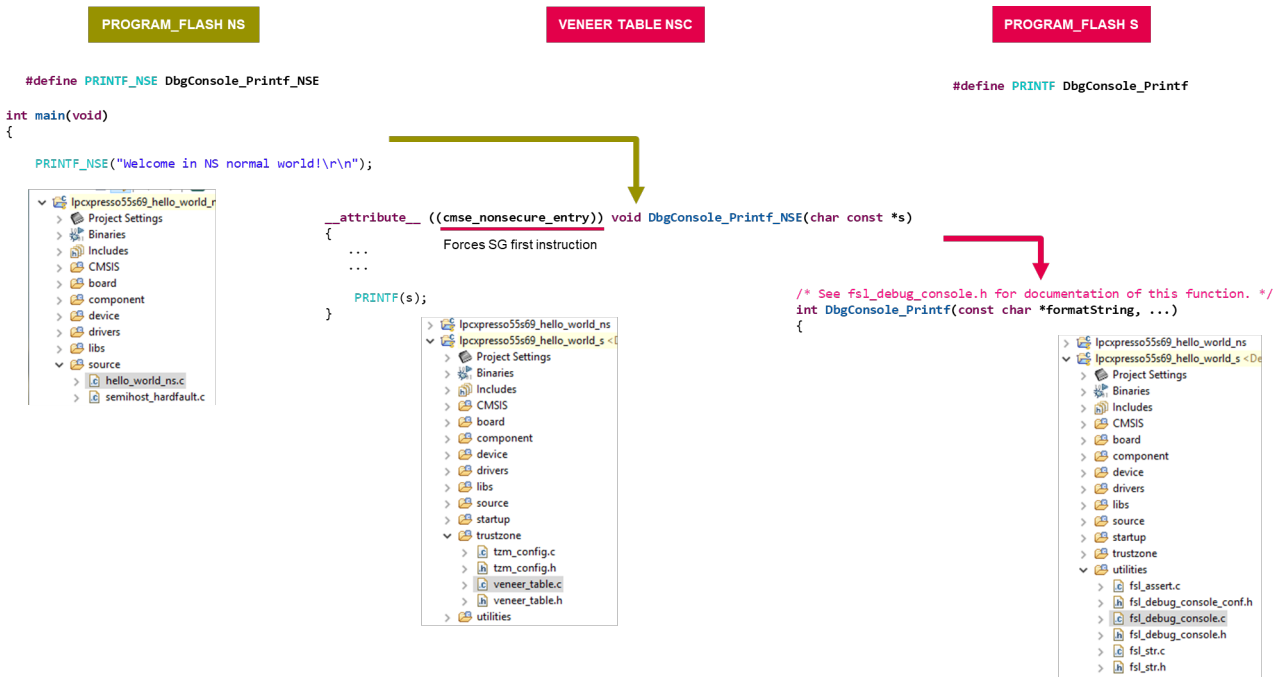
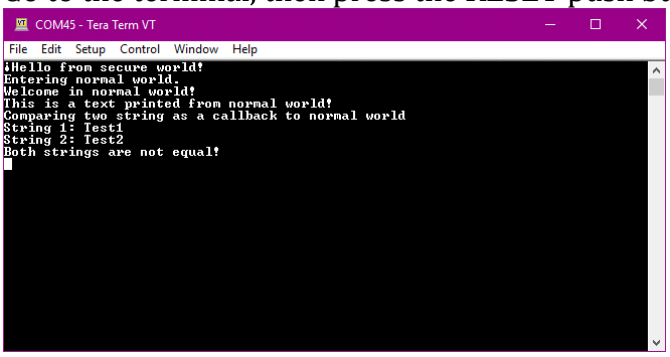


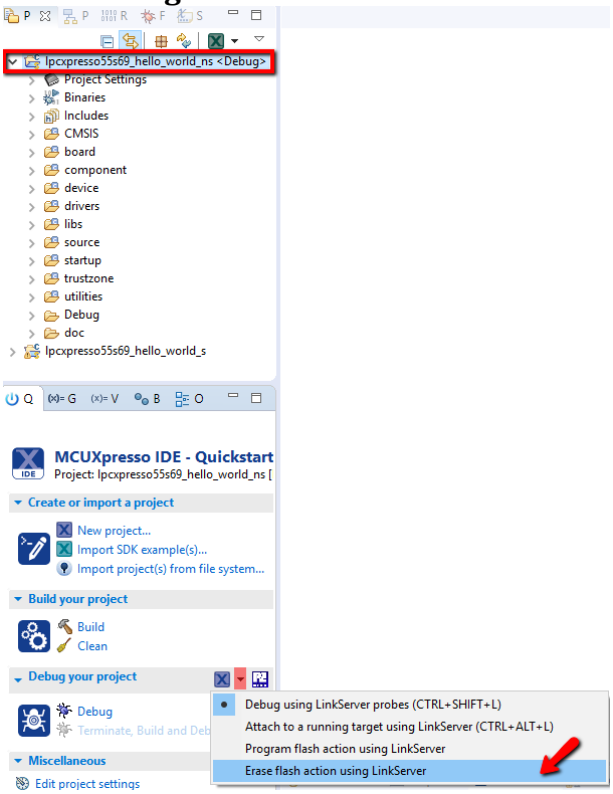
Figure 5. *PRINTF from NS*

3. Go to the terminal, then press the **RESET** push button S4

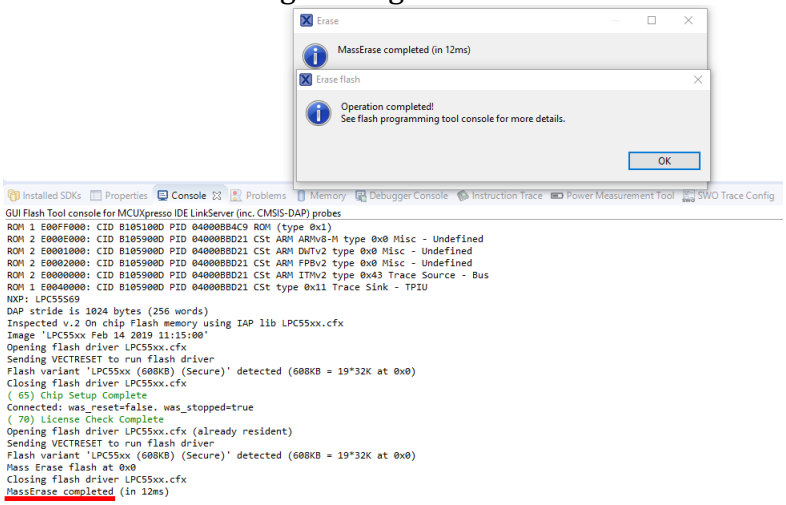


NOTE At this moment the LPC55S6x is running secure software, therefore it won't be able to connect to a Debugger. To work around this is necessary to do a mass-erase, if you forget to do this step you won't be able to flash any project.

4. Mass erase your device, select the **lpcxpresso55s69_hello_world_ns** at the project explorer, then using the Quick Start Panel look for the Debug your project section, expand the **Debug with Link Server drop-down menu** then choose **Erase flash action using LinkServer**

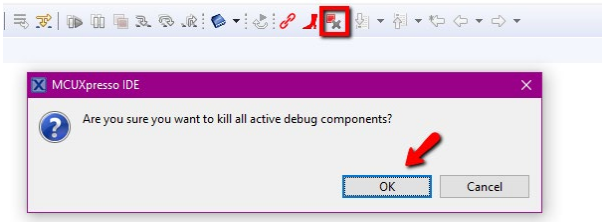


5. Wait until the program is successfully erased, to verify that the MassErase worked look for the following message at the console window



6. Click **OK** at the Operation completed! Window

7. To close the current debug session click **Cleanup Debug** → select **OK** when the window pop-up asks to kill active debug components



Additional Resources

LP55S6x User's Manual https://www.nxp.com/docs/en/user-guide/UM11126.pdf
TrustZone for cortex M-Arm https://www.arm.com/why-arm/technologies/trustzone-for-cortex-m
MCUXpresso IDE User's Guide – open from Help → MCUXpresso IDE User's Guide.