

Application note

Getting started with STM32 MCU's and Arm®TrustZone® development

Introduction

This document aims to provide guidelines using the EWARM and MDKARM software tool-chains on STM32L5, STM32U3, and STM32U5 series microcontrollers.

This application note provides a basis for building and debugging secure and nonsecure applications for devices based on Arm[®] Cortex[®]-M33 (Armv8-M architecture).

This document first gives an overview of the Arm® Cortex®-M33 and the TrustZone® concept.

This application note then describes the way of use EWARM and MDKARM with STM32L5, STM32U3, and STM32U5 series microcontrollers when the TrustZone[®] is enabled through TZEN option bit.



1 General information

This document applies to the STM32L5, STM32U3, and STM32U5 series single-core Arm[®]Cortex[®]-M33 based microcontrollers with Arm[®]TrustZone[®].

Note:

Arm and TrustZone are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

arm

Reference documents

[1]	RM0438	Reference manual STM32L552xx and STM32L562xx advanced $\mathrm{Arm}^{\textcircled{R}}\text{-}based$ 32-bit MCUs
[2]	RM0456	Reference manual STM32U5 Series Arm [®] -based 32-bit MCUs
[3]	RM0487	Reference manual STM32U3 Series Arm®-based 32-bit MCUs
[4]	Reference Manual	Armv8-M Architecture Reference Manual available from the Arm® web site.

2 Arm[®] Cortex[®]-M33 core overview

The Arm[®] Cortex[®]-M33 is the first full-feature implementation of Armv8-M with TrustZone[®] secure technology and digital signal processing functionality. The processor supports a large number of flexible configuration options to facilitate the deployment of a wide range of applications, and offers a dedicated co-processor interface for accelerating frequently used compute intensive operations. The Cortex[®]-M33 delivers an optimized balance between performance, power, security, and productivity.



3 TrustZone[®] concept of the Armv8-M

The Cortex[®]-M33 processor with TrustZone[®] has two security states (see Figure 1) and a number of associated features:

- secure state
- nonsecure state
- four stacks and four stack pointer registers
- hardware stack-limit checking
- support for programmable MPU-like security attribution unit (SAU)
- interface for system security notification
- visibility of secure code from a nonsecure (NS) domain restricted to predefined entry points
- exception hardware automatically saves and clears secure register states when switching to nonsecure
- extensive banking of interrupt or exception control, SysTick
- memory protection unit for each of the secure and nonsecure parts.

Figure 1. Security state in Armv8-M



Note: When the TrustZone[®] is enabled, by default the system starts up in the secure state.



4 SAU / IDAU - TrustZone[®] concept

TrustZone[®] security is activated by the TZEN option bit in the FLASH_OPTR register. When the TrustZone[®] is enabled, the security attribution unit (SAU) and implementation defined attribution unit (IDAU) define the access permissions based on secure and nonsecure states.

- IDAU: provides a first memory partition as secure or, nonsecure callable attributes. The IDAU memory map partition is not configurable and is fixed by hardware implementation.
- SAU: eight regions, used to overwrite IDAU in order to set secure areas and confirm nonsecure ones.
- The security state is selected based firstly on the IDAU security attribute, then combined with SAU security attribution. The resulting security attribution is the highest security setting of either IDAU and SAU.
- The "secure" security attribution priority has the highest secure priority, then nonsecure callable has a lower secure priority and nonsecure has the lowest secure priority. Any undefined region is secure by default

When the TrustZone[®] security is activated, the default security state is, for:

- The CPU: the Cortex[®]-M33 is in secure state after reset. The boot address must be in a secure address.
- The memory map: SAU is fully secure after reset. The whole memory map is fully secure. Up to eight SAU configurable regions are available for security attributions.
- Flash memory:
 - Flash security area is defined by watermark user options. All flash is fully secure.
 - Flash block based features are nonsecure after reset. Even if all the flash memory is nonsecure through IDAU/SAU and through the flash secure watermark option bytes, it is possible to configure volatile secure areas using the flash memory block based feature: any page is programmable on the fly as secure mode, using the flash interface block based configuration registers.
 - SRAM: all the SRAM is secure after reset. memory protection block based controller (MPCBB) is secure.
 - Nonsecure memory view is identical to other Cortex[®]-M cores.
 - The secure memory space is divided into two memory types:
 - Secure: containing secure program code and data, such as stack and heap.
 - Nonsecure callable (NSC): contains entry functions (for example entry points for APIs), this is to
 prevent nonsecure application from branching into invalid entry points.



5 Debugging modes

5.1 Invasive debug

Invasive debug is defined as a debug process where the user controls and observes the processor activity. Most debug features are considered as invasive debug as they enable the user to halt the processor and modify its state.

DBGEN and SPIDEN controls have invasive debug permissions.

5.2 Non-invasive debug

Non-invasive debug is defined as a debug process where the user observes the processor but does not control it. The Embedded Trace Macrocell[™] (ETM) interface and the performance monitor registers are features of non-invasive debug.

NIDEN and SPNIDEN controls both have non-invasive debug permissions. Non-invasive debug is always permitted when invasive debug is permitted.

6 Debug access

6.1 Secure debug access

Secure debug access offers full visibility on all instruction execution, across all memory regions, and device peripherals. It allows the tracing and debugging of the secure and the nonsecure software running on the target. Debugging of secure firmware is only available in this mode.

Code running in secure state has access to both secure and nonsecure information.

6.2 Nonsecure debug access

The nonsecure debug view protects the secure memory and peripherals. These are invisible to the debugger in nonsecure mode. Debug and trace capabilities are limited to nonsecure system resources.



7 Flash memory protection

7.1 Readout protection level when TrustZone[®] is disabled

There are three readout protection levels as listed below:

- Level 0: all read/program/erase operations to and from the user Flash memory are allowed.
- Level 1: the Flash memory content is protected against debugger and potential malicious code stored in RAM.
- Level 2: all debug features are disabled, the boot from SRAM and from system memory are no longer available.

7.2 RDP level transition scheme when TrustZone[®] is disabled

The RDP level transition scheme when TZEN is cleared is illustrated in Figure 2.



Figure 2. RDP level transition scheme when TrustZone[®] is disabled (TZEN = 0)

7.3 Readout protection level when TrustZone[®] is enabled

In addition to the RDP levels mentioned previously is set, there is a new RDP level named 0.5 that allows the following features:

- All read and write operations to / from the nonsecure flash memory are possible. The debug access to secure area is prohibited. Debug access to nonsecure area remains possible.
- Nonsecure debug mode: nonsecure debug is possible when the CPU is in nonsecure state.





The RDP level transition scheme when TZEN is set is illustrated in Figure 3.





Note: RDP regression is only available through the debug interface or the system bootloader.

57/



8 Starting with secure/nonsecure project

EWARM and MDK-ARM provide very similar approaches to support STM32L5, STM32U3, and STM32U5 series microcontrollers. It is done using two separate projects: secure and nonsecure.

- Section 9 provides the MDK-ARM project instructions.
- Section 10 provides the instructions for EWARM.
- Section 11 provides the instructions for the CubeIDE.

Each section provides step by step instructions explaining the project setup of the secure and nonsecure parts using STM32L5/U5 series microcontrollers.

To begin with, use a template from STM32CubeL5 package (STM32Cube_FW_L5) that is composed from two sub-projects: one for the secure application part and the other for the nonsecure application part.

Before starting, the option bytes must be set using the STM32CubeProgrammer as detailed in the project readme.txt . This tool is available for download from *www.st.com* and illustrated in Figure 4.

Figure 4. Configuration of option bytes using STM32CubeProgrammer

m STM	32CubeProgrammer				– a ×
STM32	200			(19)	F 🖪 y 🔆 🗺
Cube	Programmer				
	Option bytes				Not connected
	TZEN		Unchecked : Global TrustZone security disabled Checked : Global TrustZone security enabled	ST-LINK	 Connect
			Hide protection first area enable		ST-LINK configuration
*	HDP1EN		Unchecked : No HDP area 1 Checked : HDP first area is enabled	Serial number	003500163137511533333639 👻 🧭
OB	HDP1_PEND	Value 0x0 Address 0x8000000	End page of first hide protection area	Port	SWD 👻
			Hide protection second area enable	Frequency (kHz)	24000 👻
	HDP2EN		Unchecked : No HDP area 2 Checked : HDP second area is enabled	Mode	Under reset 👻
	HDP2_PEND	Value 0x0 Address 0x8000000	End page of second hide protection area	Access port	0 ~
	NSBOOTADD0	Value 0x10000 Address 0x8000000	Non-secure Boot base address 0	Shared	Hardware reset
	NSBOOTADD1	Value 0x17f200 Address 0xbf90000	Non-secure Boot base address 1	External loader	-
		Value 0v19000 Address 0vs000000		Target voltage	3.27 V
	SECBOOTADD0	Value 0x10000 Address 0x000000	Secure boot base address 0	Firmware version	V3J3M2
			The boot is always forced to base address value programmed in SECBOOTADD0		Firmware upgrade
	BOOT_LOCK		Unchecked : Boot based on the pad/option bit configuration Checked : Boot forced from base address memory		
	▼ Segure Area 1				
	Name	Value	Description		
	SECWM1_PSTRT	Value 0x0 Address 0x8000000	Start page of first secure area		
	SECWM1_PEND	Value 0x7f Address 0x803f800	End page of first secure area		
	Write Protection 1				
	▼ Secure Δrea 2				
	Name	Value	Description		
	SECWM2_PSTRT	Value 0x1 Address 0x8040800	Start page of second secure area		
	SECWM2_PEND	Value 0x0 Address 0x8040000	End page of second secure area	J	
				<u> </u>	
			Apply Read	_	Device information
	Log		Verbosity level 🔍 1 🔍 2 🔍 3	Device	
-	16:09:16 : OPTION BYTE PROGR	AMMING VERIFICATION:		Type	-
(\mathbf{a})	16:09:19 : Disconnected from	essivity programmed i device.		Device ID	
_	16:09:19 : ST-LINK SN : 0035 16:09:19 : ST-LINK FW : V333	M2		Flash size	-
(?)				CPU	
\sim			8		



9 Using MDK-ARM for Cortex[®]-M33 with Trust Zone

The latest version of MDK-ARM (Keil[®]) is available for download from the official Arm[®] Keil[®] web site. MDK-ARM (Keil[®]) is installed by default in the "C:\Keil" directory on the PC local hard disk, the installer creates a start menu μ Vision[®] 5 shortcut.

The MDK-ARM v5.27.0.0 and STM32L562-DK disco board are used for this section.

9.1 Secure project settings

This section outlines the secure project settings.

1. Open the multi-projects workspace file: "Project.uvmpw" that allows the user to work on both projects at the same time. The open project appears in the project explorer as illustrated in Figure 5.

Project	Д	x
🖃 🌃 WorkSpace		
🖨 🍂 Project: Project_s		
😑 ᇶ STM32L562E-DK_s		
🗈 🚞 Doc		
🕀 😹 Drivers/BSP/STM32L562E-Discovery		
🕀 😹 Drivers/BSP/Components		
🕀 🧰 Drivers/STM32L5xx_HAL_Driver		
🕀 🧰 Drivers/CMSIS		
🕀 🧰 Example/MDK-ARM		
🕀 🧰 Example/User		
🗄 쓚 Project: Project_ns		
🖨 🔊 STM32L562E-DK_ns		
🕀 🧰 Doc		
🕀 🧰 Drivers/BSP/Components		
🕀 🧰 Drivers/BSP/STM32L562E-Discovery		
Drivers/STM32L5xx_HAL_Driver		
🕀 🧰 Drivers/CMSIS		
🗉 🧰 Example/MDK-ARM		
🕀 🧰 Example/User		

Figure 5. MDK-ARM project structure

2. Set project_s as active project, see Figure 6.



Project		д 🗙
🖃 📴 WorkSpace		
🗄 🔧 Project: Project_s		
🗄 쓚 Project: Project_ns	Set as Active Project	



3. Select the correct device by opening the configuration window and selecting: **Project / Options for Target / Device** then select the device from the list (see Figure 7).

Options for Target 'STM32L562E-DI	(_s')	<
Device Target Output Listing User	C/C++ (AC6) Asm Linker Debug Utilities	
Software Packs Vendor: STMicroelectronics Device: STM32L562ZETx Toolset: ARM	Software Pack Pack: Keil.STM32L5xx_DFP.1.0.9 URL: http://www.keil.com/pack	
Search:		
STM32L562QEIx STM32L562QEIxQ STM32L562RETx STM32L562VETx STM32L562VETxQ STM32L562VETxQ STM32L562ZETx STM32L562ZETxQ	The ARM Cortex-M33 is the most configurable of all Cortex-M processors. It is a full featured microcontroller class processor based on the ARMv8-M mainline architecture with ARM TrustZone security.	
	OK Cancel Defaults Help	

Figure 7. Device selection



- 4. From **Project / Options for Target / Target / Code Generation** section, select the "Software Model" as "Secure". Ensure the right memory area is selected. See Figure 8:
 - Secure Boot address : Flash at 0x0C000000 : secure Flash
 - Secure Boot address: SRAM1 at 0x30000000: secure SRAM

Figure 8. Project_s target options

V Options for Target 'Project_s' ×										
Device Target Output Listing User C/C++ (AC6) Asm Linker Debug Utilities										
STMicroelectronics STM32L552Z	E-Q	0	Code G	eneration - Compiler:	V6.11		.			
Operating system: None	Xtal (MHz): 12		Softw	are Model:	Secure Mo	ode	-			
System Viewer File: Image: Use MicroLIB Big Endian STM32L5x2.svd Image: Single Precision Image: Single Precision										
Read/Only Memory Areas	~	a .	-Read/	Write Memo	ory Areas	~				
BOM1:	Size	- C	derault	οπ-chip RAM1·	Start	Size				
ROM2:	<u> </u>	c		RAM2:		i —				
ROM3:	i —	C		RAM3:		i —				
on-chip	0x40000	œ		on-chip IRAM1:	0x20018000	0x18000				
IROM2: 0xC000000	0x40000	0		IRAM2:	0x30000000	0x18000				
	ОК	Can	cel	Defau	lts		Help			

5. Ensure that the secure nonsecure callable functions (NSC) object file "secure_nsclib.o" is defined in **Project / Options for Target / Linker** under **Misc Controls** section.

Use the [--import_cmse_lib_out ..\lib\nsclib_Secure.o] command to create the output library: nsclib_Secure.o.

This file, automatically generated during the build of the secure project, contains all the nonsecure callable functions declared with the prefix: __attribute__((cmse_nonsecure_entry)). See Figure 9.

Figure 9.	Project_	s	Linker	config	uration
-----------	----------	---	--------	--------	---------

W Options for Target 'Project_s'	×
Device Target Output Listing User C/C++ (AC6) Asm Linker Debug Utilities	
□ Use Memory Layout from Target Dialog X/O Base: □ Make RW Sections Position Independent R/O Base: □ Make RO Sections Position Independent R/W Base □ Don't Search Standard Libraries Isable Warnings:	
Scatter .\Project_s.sct	Edit
Miscimport_cmse_lib_out=.\\Secure_nsclib\secure_nsclib.o	× •
Linker -cpu=Cortex-M33 *.o control -library_type=microlibstrictscatter ".\Project_s.sct" string	Ŷ
OK Cancel Defaults	Help

Under scatter file section, check that this file contains the correct addresses as illustrated in Figure 9. This file is used by the Linker and determine how the memory layout is organized. A sample of a scatter file is given in Figure 10.

Figure 10. Scatter file sample

```
Project_s.sct
         1
     .
      *** Scatter-Loading Description File generated by uVision ***
  2
     .
     3
                                                ********
  4
     LR IROM2 0x0C000000 0x00040000 { ; ; load region size region
  5
      ER IROM2 0x0C000000 0x0003E000 { ; load address = execution address
  6
       *.o (RESET, +First)
  7
  8
       * (InRoot$$Sections)
  9
       .ANY (+RO)
 10
       .ANY (+XO)
 11
       ł
 12
      RW IRAM2 0x30000000 0x00018000 { ; RW data
 13
       .ANY (+RW +ZI)
 14
      1
 15 }
 16
 17
    LR IROM3 0x0C03E000 0x00002000 { ; load region size region
 18
      ER_IROM3 0x0C03E000 0x00002000 { ; load address = execution address
 19
       * (Veneer$$CMSE)
                                   ; check with partition.h
 20
      ł
 21
    }
 22
```

6. Select "ST-LINK Deb ugger" as the debugger from: Project / Options for Target / Debug. See Figure 11.

Figure 11. Target options debug



- If "ST-LINK Debugger" does not appear in the list:
- a. Go to C:\Keil install directory
- b. Open TOOLS.INI file and apply the following changes:
 - Look for [ARMADS]: All Armv8M based devices requires the processor SARMV8M.DLL. The TOOLS.INI file contains CPUDLL3 = SARMV8M.DLL (TDRV2, TDRV13, TDRV14, TDRV15, TDRV16). The ST-Link driver is registered as TDRV6 in this example and could vary depending on the project: TDRV6=STLink\ST-LINKIII-KEIL_SWO.dll ("ST-Link Debugger").
 - ii. Add the TDRV6 to the list in CPUDLL3= SARMV8M.DLL:CPUDLL3 = SARMV8M.DLL (TDRV2, TDRV6, TDRV13, TDRV14, TDRV15, TDRV16).



7. From "Debug" settings tab, ensure the debugger is connected as illustrated in Figure 12.

Cortex-M Target Driver Setup					×
Debug Trace Rash Download					
Debug Adapter Unit: ST-LINK/V3 ✓ Shareable ST-Link	SW De	IDCODE 0x0BE12477	Device Name ARM CoreSight SV	/-DP	Move Up
Serial Number: 002A001A3137511533333639 Version: HW: V3 FW: V3J3M2 Check version on start Target Com Port: SW Clock Req: 1.800 MHz Selected: 1 MHz Perfomance Profile: High Perf	C Au	utomatic Detection anual Configuration d Delete U	ID CODE: Device Name: odate IR Ien:	A	Down P: 0
Debug Connect & Reset Options Connect: under Reset 🔽 Reset: Autodetec IV Reset after Connect 🔽 Stop after Res	ct s et	Cache Optic	Code Downloa Memory Downloa	d Options / Code Download nload to Flash	
			ОК	Cancel	Apply

Figure 12. Debug configuration

From the "Flash Download" tab, select the correct Flash-loader (see Figure 13):

- "Download Function": sets the Flash operations.
- RAM for algorithm: defines the address space where programming algorithms are loaded and executed. Usually, the address space is located in on-chip RAM.
- "Program Algorithm": contains the Flash programming definitions.



Figure 13. Flash-loader settings

LOAD 5 3	C Erase Full Chip Erase Sectors Do not Erase	 ✓ Program ✓ Verify ✓ Reset and F 	Start: 0	x30000000 Size: 0x0800	1
Programmin	g Algorithm				
Descriptio	n	Device Size	Device Type	Address Range	
STM32L5	x 512 Dual 0C00	512k	On-chip Flash	0C000000H - 0C07FFFFH	
			Start: 0	x0C000000 Size: 0x000800	00
		Add	Remove		

9.2 Nonsecure project settings

1. Set project_ns as active project (see Figure 14).

Figure 14. Project_ns nonsecure project selection





2. Select the correct device by opening the configuration window: Select **Project / Options for Target** (see Figure 15).

W Options for Target 'STM32L562E-DK_s'	×
Device Target Output Listing User C/C++ (AC6) Asm Linker Debug Utilities	
Software Packs Vendor: STMicroelectronics Device: STM32L562ZETx Total in a SDM UBL:	
Search:	
STM32L562QEIx STM32L562QEIxQ STM32L562RETx STM32L562VETxQ STM32L562ZETx STM32L562ZETX STM32L562ZETX STM32L562ZETX STM32L562ZETX STM32L562ZETX STM32L562ZETX STM32L562XEX STM32L562XEX STM32L562XEX STM32L562XEX STM32L562XEX STM32L562XEX STM32L562XEX STM32L562XEX STM32L562XEX STM32L562XEX STM32L562XEX STM32L562XEX STM3ZEX STM3ZEX STM3ZEX STM3ZEX	< >
OK Cancel Defaults Help	

Figure 15. Device selection



- 3. Ensure the right memory area is selected from Project / Options for Target / Target:
 - Boot address 0: Flash at 0x08040000: nonsecure flash
 - Boot address 1: SRAM at 0x20018000: nonsecure SRAM

The software model must be set in nonsecure mode (see Figure 16).

Figure 16. Memory configuration

Options for Target 'Project_r	15'					X			
Device Target Output Listing User C/C++ (AC6) Asm Linker Debug Utilities									
STMicroelectronics STM32L552ZE-Q Code Generation ARM Compiler: V6.11									
	Xtal (MHz):	2.0	Software N	Model: Non-Secu	re Mode	-			
Operating system: None		•		,					
System Viewer File:			🔽 Use M	icroLIB	🗌 Big Endian				
STM32L5x2.svd			Floating Po	oint Hardware:	Single Precision	-			
Use Custom File									
Read/Only Memory Areas			Read/Write	Memory Areas					
default off-chip Start	Size	Startup	default off-	chip Start	Size	Nolnit			
ROM1:		0	🗆 RA	M1:					
ROM2:		0	🗆 RA	M2:					
ROM3:		0	□ RA	M3:					
on-chip			on-	chip					
IROM1: 0x8040000	0x40000	œ	IRA IRA	M1: 0x20018000	0x18000				
□ IROM2: 0xC000000	0x80000	0	IRA	M2: 0x3000000	0x30000				
•									
	ОК	Car	cel	Defaults		Help			



4. Add the import library from the secure project: this file is automatically included at link time in the nonsecure project. It allows the nonsecure part to call functions from the secure part (see Figure 17).

Options for Target 'Project_ns'	×
Device Target Output Listing User C/C++ (AC6) Asm Linker Debug Utilities	
✓ Use Memory Layout from Target Dialog X/O Base: ✓ Make RW Sections Position Independent R/O Base: ✓ Make RO Sections Position Independent R/W Base ✓ Don't Search Standard Libraries disable Warnings:	
Scatter File	Edit
Misc .//Secure_nsclib/secure_nsclib.o	~ ~
Linkercpu=Cortex-M33 *.o controllibrary_type=microlibstrictscatter ".\Objects\Project_ns.sct" string	Ŷ
OK Cancel Defaults	Help

Figure 17. Linker options

Under scatter file section, check that this file contains the correct addresses as shown in Figure 17. This file is used by the Linker and determines how the memory layout is organized. A sample of a scatter file is given in Figure 18.

Figure 1	18.	Scatter	file	sam	ple
----------	-----	---------	------	-----	-----

Project_ns.sct ; 1 ; *** Scatter-Loading Description File generated by uVision *** 2 3 4 LR IROM1 0x08040000 0x00040000 { ; ; load region size region 5 ER IROM1 0x08040000 0x00040000 { ; load address = execution address 6 7 *.o (RESET, +First) *(InRoot\$\$Sections) 8 9 .ANY (+RO) 10 .ANY (+XO) 11 ł RW_IRAM1 0x20018000 0x00018000 { ; RW data 12 13 .ANY (+RW +ZI) 14 } 15 }



5. Select "ST-LINK debugger" from Project / Options for Target / Debug (see Figure 19).

	Figure [•]	19. Debug settings	5
Asm Linker	Debug	Utilities	
🔍 Use: 🛛	ST-Link De	bugger	Settings

- 6. From Debug settings / Flash Download window (see Figure 20) select:
 - Download function: sets the flash operations
 - RAM for algorithm: defines the address space where programming algorithms are loaded and executed. Usually, the address space is located in the embedded RAM.
 - Program algorithm: contains the definitions for programming flash.

Figure 20. FlashLoader configuration

Cortex-M Target Driver Setup	×
Debug Trace Flash Download	
Download Function ○ Frase Full Chip ✓ Program ○ Erase Sectors ✓ Verify ○ Do not Erase ⊂ Reset and Run	
Description Device Size Device Type Address Range STM32L5x_512_NSecure_Fl 512k On-chip Flash 08000000H - 0807FFFFH	
Stat: [0.0000000 Size: [0.0000000	
Add Remove	
OK Cancel A	.pply





It is now possible to build both projects at the same time. From **Project / Batch Setup** (see Figure 21 and Figure 22) or from the icon available from the menu bar go to the batch setup menu and select both projects.

Note: The secure project must be built first in order to create the import library for the nonsecure project. In order to build the secure project before the nonsecure one, it must be first in order.



Figure 22. Project build ordering



Then, from the same menu, click on "Batch Build" to build both projects (see Figure 23).

Figure 23. Build both projects in one step

i 🖉 🔛 🖉	😻 🕶 🔜 🛛 🙀 Project_n:	;
Project	🍪 Batch Build	д 🗙
🖃 📴 WorkS	Batch Rebuild	
🗄 🔧 Pro	Batch Clean	
🗄 🖧 Pro	🧼 Batch Setup	



9.3 Execute from secure code to nonsecure code

Before downloading the projects, a connection to the STM32L562E-DK Discovery board must be made as follows:

 Connect the ST-LINKV3 programming and debugging tool on the Discovery board by plugging the USB cable to the board CN17 (ST-LINK USB connector). LD3 illuminates in red when the ST-LINKV3 is connected as illustrated in Figure 24.



Figure 24. STM32L562E-DK Discovery board in connected status

2. Select the **Project_ns** project as the active project then load the nonsecure binary code. Select the **Project_s** project as the active project then load the secure binary code. This is illustrated in Figure 25.

Figure 25. Load the nonsecure binary





3. Start a debug session by clicking the "Download and Debug" button in the toolbar illustrated in Figure 26.



Note:

The system always boots in secure code (main.c) at first and the secure application then launches the nonsecure application as illustrated in below.



Figure 27. Main.c sample code

At the end of secure function, the system switches from the secure state to the nonsecure state (see Figure 28).



Figure 28. Code switch to nonsecure code status



The secure status is provided from the status bar at the bottom of Keil® interface as illustrated in Figure 29.

Figure 29. CPU status

70 * 8; 71 * *; 72 * * 73 ⊡/** 74 * 8; 75 */ 76 */ 77 /***********************************		Ţ
<		>
ST-Link D	bugger Debug: Secure CPU: Non-Secure	t1: 0.00000000 sec L:50 C:1 CAP NUM SCRL OVR R/W



10 Using EWARM for Cortex M33 with TrustZone[®]

The latest version of IAR Embedded Workbench for Arm[®] (EWARM) is available to download from the official web site of IAR System.

This part uses EWARM v8.40.1 and STM32L562-DK disco board.

10.1 Secure project settings

To configure a secure project, the first step is to open "Multi-projects" workspace file: Project.eww that allows the user to work on both projects at the same time.

1. The open project appears in the project explorer view illustrated in Figure 30.

Figure 30. EWARM v8.40.1 project explorer view

Workspace	•	φ×
Project_s - STM32L562E-DK_Templates_TrustZone		~
Files	\$	•
🖽 🗂 Project		
	~	
-⊞ i Doc		
│		
Example		
U Gutput		
└────────────────────────────────────	~	
E Drivers		
Example		
Let Cutput		
Overview Project_s Project_ns		



2. Set project_s-STM32L562E-DK_Templates_TrustZone as active project as illustrated in Figure 31.

i igure 31. Setting the project to active status	Figure 31	. Setting	the	project	to	active	status
--	-----------	-----------	-----	---------	----	--------	--------

Workspace			-	ąх
Project_ns - STM32L562E-DK_Templates_	TrustZone			~
Files			¢	•
Project_s - STM32L562F Project_s - STM32L562F Doc Project_s Example Project_ns - STM32L Project_ns - STM32L	Options Make Compile Rebuild All		✓ ✓	
⊢⊞ ■ Doc -⊞ ■ Drivers -⊞ ■ Example -⊞ ■ Output	Clean C-STAT Static Analysis Stop Build Add	>		
	Remove Rename Version Control System Open Containing Folder	>		
	Set as Active			



3. Open the configuration window by selecting **Project-s / Options / General Options** and select the correct device from "Processor variant" section.

From "TrustZone" section, ensure that the mode selected is "Secure" and "TrustZone" checkbox is checked as shown in Figure 32.

Figure 32. Device selection

Options for node "Project_s" \times Category: General Options Static Analysis Runtime Checking Library Options 2 MISRA-C:2004 MISRA-C:1998 C/C++ Compiler Target Assembler Output Library Configuration Library Options 1 Output Converter Processor variant Custom Build Cortex-M33 ○ Core Build Actions Linker ST STM32L562QE-Q ∎. Device Debugger Simulator None CMSIS-Pack CADI CMSIS DAP Endian mode Floating point settings GDB Server Little I-jet FPU None \sim J-Link/J-Trace ○ Big TI Stellaris D registers BE32 Nu-Link BE8 PE micro ✓ TrustZone ST-LINK DSP Extension Third-Party Driver Mode Secure \sim TI MSP-FET Advanced SIMD (NEON) TI XDS 0K Cancel



 \times

4. From Project-s / Options / Linker / Config "Linker configuration file editor" section (see Figure 33):

- a. Click Edit to display the linker configuration file editor.
- b. Check the linker configuration file to make sure that the application has been linked to the right address:
 - Secure boot address : Flash at 0x0C000000 for the secure flash
 - Secure boot address: SRAM1 at 0x30000000 for the secure SRAM

Figure 33. Linker configuration

Options for node "Project_s"

stegoly.	,					Factory S	Settings
eneral Options							
tatic Analysis							
untime Checking	#1.6	D:		a 1	F b	.	• ••
C/C++ Compiler	#define	Diagno	ostics	Checksum	Encodings	Extra	Options
Assembler	Config	Library	Input	Optimizations	Advanced	Output	List
Dutput Converter	Linker	configuratio	on file —				
Custom Build		erride defa	ult				
Build Actions							
.inker	SP	ROJ_DIR	\$\stm32	562xx_flash_s.ic	af		
Debugger							
Simulator	Linker conf	iguration	file edit	or			×
CADI							
CMSIS DAP	Vector Tab	e Memor	y Regio	ns Stack/Heap	Sizes		
GDB Server							
I-jet			Start	:	End:		_
J-Link/J-Trace	ROM		0x0	C000000	0x0C03FF	FFF	
TI Stellaris							
Nu-Link	RAM		0x3	0000000	0x30017F	FF	
PE micro							- 1
ST-LINK							
Third-Party Driver					Г	Save	
TI MSP-FET					L	Jave	
TI XDS							_

This .icf file contains all the information required by the linker.



5. Open the debugger tab from: **Project / Options / Debugger**. From setup section, select ST-LINK as a debugger in the driver field (see Figure 34).

Options for node "Project_s" X Category: Factory Settings General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Output Converter Run to Custom Build Setup macros Build Actions Use macro file(s) Simulator OUSIS DAP GDB Server I-jet I-jet Device description file
Category: Factory Settings General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Download Images Extra Options Multicore Plugins Output Converter Run to Custom Build Build Actions Linker Debugger Simulator CADI CMSIS DAP Device description file
J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET



6. From the "Download" tab, ensure that "Use flash loader" is checked (see Figure 35).

Options for node "Project_s" × Category: Factory Settings General Options Static Analysis Runtime Checking Setup Download Images Extra Options Multicore Plugins C/C++ Compiler Assembler Verify download Output Converter Custom Build Suppress download **Build Actions** Use flash loader(s) Linker Override default .board file Debugger \$TOOLKIT_DIR\$\config\flashloader\ST\FlashSTM3. Simulator CADI Edit. CMSIS DAP GDB Server Perform mass erase before flashing I-jet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDS 0K Cancel

Figure 35. FlashLoader selection



7. The secure project must specify the nonsecure project output file as an extra image that must be loaded by the debugger. To do this, use: **Project / Options / Debugger / Images** and check the "Download extra image" check box (see Figure 36).



Options for node "Project_s"

 \times

Category:	Factory Settings
General Options Static Analysis Runtime Checking C/C++ Compiler	Setup Download Images Extra Options Multicore Plugins
Assembler	Download extra image
Output Converter Custom Build	Path: \$PROJ_DIR\$\STM32L562E-DK_ns\Exe\project_ns.i
Linker	Offset: 0 Debug info only
Debugger	Download extra image
CADI	Path:
CMSIS DAP GDB Server	Offset: Debug info only
I-jet J-Link/J-Trace	Download extra image
TI Stellaris	Path:
Nu-Link PE micro	Offset: Debug info only
ST-LINK Third-Party Driver	
TI XDS	
	OK Cancel

Debug info causes the debugger to only download debug information, and not the complete debug file.



- 8. From Project / Options / ST-LINK "Setup" tab, see Figure 37:
 - Select the "ST-LINK debugger".
 - Select the reset type:
 - System reset: resets the core and peripherals.
 - Core reset: resets the core via the VECTRESET bit; the peripheral units are not affected.
 - Software reset: sets PC to the program entry address.
 - Hardware reset: the probe toggles the nSRST/nRESET line on the JTAG connector to reset the device. This reset usually resets the peripheral units also.
 - Connect during reset: ST-LINK connects to the target while keeping Reset active. Reset is pulled low and remains low while connecting to the target.
 - Select the communication interface:
 - JTAG: to use the JTAG interface.
 - SWD: to uses the SWO interface, which uses fewer pins than JTAG. Select SWD if the serial-wire output (SWO) communication channel is to be used.
 - Select the Access Port:
 - Auto: automatically uses the access port 0 for Cortex[®]-M33.
 - Manually: specify the access port to be used.

Figure 37. Project setup

Options for node "Project_s"		×
Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator CADI CMSIS DAP GDB Server I-jet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDC	Setup Communication Breakpoints Emulator Serial no:	Factory Settings
	ОКС	ancel





10.2 Nonsecure project settings

Set project_s-STM32L562E-DK_Templates_TrustZone as active project

 Open the configuration window by selecting Project-s / Options/ General Options. In the "Target" tab, select the correct device from processor variant section (see Figure 38).
 From the TrustZone[®] section, ensure that the "Nonsecure" mode is selected and the TrustZone[®] box is checked.

Figure 38. Project set up: general options

Options for node "Project_ns"

 \times

Category:	
General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator CADI CMSIS DAP GDB Server I-jet J-Link/J-Trace TI Stellaris Nu-Link PE micro	Library Options 2 MISRA-C:2004 MISRA-C:1998 Target Output Library Configuration Library Options 1 Processor variant Ocore Cortex-M33 Image: Cortex-M33 O Device ST STM32L562QE-Q Image: Cortex-M33
	O CMSIS-Pack None Endian mode Floating point settings Ittle FPU
	BE32 BE8 D registers TrustZone
Third-Party Driver TI MSP-FET TI XDS	DSP Extension Advanced SIMD (NEON)
	OK Cancel



 \times

2. From Project-s / Options / Linker / Linker configuration file section (see Figure 39):

- Click edit to display the linker configuration file editor.
 - Check the linker configuration file to make sure that the application has been linked to the right address:
 - Boot address 0: Flash at 0x08040000 (nonsecure flash)
 - Boot address 1: SRAM at 0x20018000 (nonsecure SRAM).

Figure 39. Project linker configuration

Options for node "Project_ns"

itegory.	_				Factory 9	Setting:
eneral Options						
atic Analysis						
untime Checking		D :	C 1	-	F .	.
C/C++ Compiler	Hetine	Diagnostics	Checksum	Encodings	Extra	Options
Assembler	Config	Library Input	Optimizations	Advanced	Output	List
Output Converter	Linker	configuration file	,			
Custom Build		erride default				
Build Actions			2015C2	: _ f		1
inker	ar ar	NOJ_DIN\$\stri	ISZIS620X_filash_fis.	lici		
)ebugger			10			~
Simulator	Linker co	nfiguration fi	eeditor			
CADI		L. Manager	Decision at the			_
CMSIS DAP	Vector T	able Memory	Regions Stack/He	ap Sizes		
GDB Server			Start:	End		
1-jet	ROM		0x08040000	0x080	7EEEE	
J-LINK/J-Trace						
11 Stellaris Nu Link	PAM		0x20018000	0x200	DEEEE	
DE micro	North		0x20010000		2111	
Third-Darty Driver					0-1	
TT MCD_FET					Sav	/e
TTYDS						
11 AD3						



3. From **Project-s / Options / Linker** in the "Library" (see Figure 40). Add the imported library from the secure project. This file is automatically included at link time in the

nonsecure project. It allows the nonsecure part to call functions of the secure part.

Figure 40. Linker library setup

Options for node "Project_ns"

 \times

lategory:						Factory 9	Settings
General Options Static Analysis Runtime Checking C/C++ Compiler	#define	Diag	nostics	Checksum	Encodings	Extra (Options
Assembler Output Converter Custom Build Build Actions	Config	Library omatic run nal libraries	Input time librar s: (one pe	Optimizations y selection r line)	Advanced	Output	List
Debugger Simulator CADI CMSIS DAP GDB Server I-jet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET	\$PRO.	J_DIR\$\\ erride defau Entry symb No entry sy	Secure_r ult prograr ool [. ymbol	nsclib\secure_ns m entry iar_program_st	art		
TI XDS					OK		Cancel

4. The other configurations are similar to the secure project.





10.3 Build projects

Both projects are ready to be built.

1. Select **Project / Batch Build** or the icon available from the menu bar (see Figure 41).

Project - IAR Embedded Workbench IDE - Arm 8.40.1							
File Edit View	Proj	ject	ST-Link	Tools	Window	Help	-
1 🗅 🔛 🕋 🗌		Add	l Files				- < 0
Workspace		Add	d Group				
Project_ns - STM32L!		Imp	ort File Li	st			
Files		Add	l Project C	onnectio	on		
🗉 🗖 Project		Edit	t Configur	ations			
		Ren	nove				L
□ Doc		Cre	ate New P	roject			-
Examp		Ado	d Existing	Project			
Project_		Opt	tions			ALT+F7	stZone
■ Doc		Ver	sion Cont	rol Syster	m	•	
– ⊕ 🖬 Examp – ⊕ 🖬 Output	0	Mal	ke			F7	
	Ð	Cor	npile			CTRL+F7	
		Reb	uild All				
	_	Clea	an				
		Bat	ch build			F8	
		C-S	TAT Static	Analysis		•	
	8	Sto	p Build			CTRL+ATTN	

Figure 41. Project batch build



Note:

2. Add the two configurations to be built at the same time (see Figure 42).

The secure project must be built first in order to create the import library for the nonsecure project. In order to build the secure project before the nonsecure one, it must be first in the build order as illustrated below.

Edit Batch Build	× Edit Batch Build	
Name build	Name build	
Available configurations Project_ns - STM32L562E-DK_Templat Project_s - STM32L562E-DK_Template >> <<<<<>> Configurations to build (Creation of the second se	Available configurations	Configurations to build Project_s - STM32L562E-DK_Templat Project_ns - STM32L562E-DK_Templat
OK Cance	le	0K Cance

Figure 42. Project batch build order



10.4 Execute from secure code to nonsecure code

In order to execute any code, it has to be downloaded to the board as follows:

- 1. Before downloading the project, connect to the STM32L562E-DK Discovery board as follows (see Figure 43):
 - Connect the ST-LINKV3 programming and debugging tool to the Discovery board by plugging the USB cable to the CN17 ST-LINK USB connector of the board.
 - LD3 illuminates in red when the ST-LINKV3 is connected.

Figure 43. STM32L562E-DK Discovery board in connected status





 Select the Project_ns project as active project then load the nonsecure binary code. Start a debug session by clicking the download and debug button in the toolbar to program the flash memory and start debugging (see Figure 44).

Figure 44. Download and debug launch button



Note:

when trying to load the nonsecure application, the following warning messages are displayed.

Figure 45. Nonsecure application loading warning error message samples

De	ebug Log	
	Log	
	Tue Feb 11, 2020 16:09:31: Warning:	
	Tue Feb 11, 2020 16:09:31: Verify error at address 0x080400CF, target byte: 0x00, byte in file: 0x08	
	Tue Feb 11, 2020 16:09:31: Warning:	
	Tue Feb 11, 2020 16:09:31: Verify error at address 0x080400D0, target byte: 0x00, byte in file: 0xAD	
	Tue Feb 11, 2020 16:09:31: Warning:	
	Tue Feb 11, 2020 16:09:31: Verify error at address 0x080400D1, target byte: 0x00, byte in file: 0x0F	
	Tue Feb 11, 2020 16:09:31: Werning:	
	Tue Feb 11, 2020 16:09:31: Verify error at address 0x080400D2, target byte: 0x00, byte in file: 0x04	
	Tue Feb 11, 2020 16:09:31: Warning:	
	Tue Feb 11, 2020 16:09:31: Verify error at address 0x080400D3, target byte: 0x00, byte in file: 0x08	
	Tue Feb 11, 2020 16:09:31: Warning:	
	Tue Feb 11, 2020 16:09:31: Verify error at address 0x080400D4, target byte: 0x00, byte in file: 0xB1	
	Tue Feb 11, 2020 16:09:31: Werning:	
	Tue Feb 11, 2020 16:09:31: Verify error at address 0x080400D5, target byte: 0x00, byte in file: 0x0F	
	Tue Feb 11, 2020 16:09:31: Warning:	
	Tue Feb 11, 2020 16:09:31: Verify error at address 0x080400D6, target byte: 0x00, byte in file: 0x04	
	Tue Feb 11, 2020 16:09:31: Warning:	
	Tue Feb 11, 2020 16:09:31: Verify error at address 0x080400D7, target byte: 0x00, byte in file: 0x08	
	Tue Feb 11, 2020 16:09:31: Warning:	
	Tue Feb 11, 2020 16:09:31: Verify error at address 0x080400D8, target byte: 0x00, byte in file: 0x85	
	Tue Feb 11, 2020 16:09:31: Warning: Too many verify errors, only the first 200 are displayed	

This is an expected behavior, as in the verification phase, the debugger attempts to read back the loaded content and compares it with the compiled binary. The debugger generates a secure transaction in a nonsecure regions (@ 0x08040000 nonsecure flash) before SAU configuration. This access is not allowed and the content reads zero.

3. Select the Project_s project as active project then load the nonsecure binary then start a debug session.

Note:

The system always boots in secure code (main.c) at first and the secure application then launches the nonsecure application



4. The secure status is provided from secure register under CPU registers (see Figure 46).

Registers 1		→ ⋣ X
Find:		✓ Group: Current CPU Registers ✓
Name	Value	Access
RO	0x00000000	ReadWrite
R1	0x00000000	ReadWrite
R2	0x00000000	ReadWrite
R3	0x00000000	ReadWrite
R4	0x00000000	ReadWrite
R5	0x00000000	ReadWrite
R6	0x00000000	ReadWrite
R7	0x00000000	ReadWrite
R8	0xFFFFFFFF	ReadWrite
R9	0xFFFFFFFF	ReadWrite
R10	0xFFFFFFFF	ReadWrite
R11	0xFFFFFFFF	ReadWrite
R12	0xFFFFFFFF	ReadWrite
SP	0x30000818	ReadWrite
SPLIM	0x00000000	ReadWrite
LR	0xFFFFFFFF	ReadWrite
± ¤PSR	0x01000000	ReadWrite
∃ APSR	0x00000000	ReadWrite
∃ IPSR	0x00000000	ReadWrite
EPSR	0x01000000	ReadWrite
PC	0x0C000928	ReadWrite
H PRIMASK	0x00000000	ReadWrite
BASEPRI	0x00000000	ReadWrite
BASEPRI_NAX	0x00000000	ReadWrite
H FAULTHASK	0x00000000	ReadWrite
E CONTROL	0x00000000	ReadWrite
∃ IAPSR	0x00000000	ReadWrite
EAPSR	0x01000000	ReadWrite
IEPSR	0x01000000	ReadWrite
SECURE	0x00000001	ReadWrite
CYCLECOUNTER	0 SEC	
CCTIMER1	0 Rea	dWrite
CCTIMER2	0 Sect	urity state
CCSTEP	0 0: N 1: S	on-secure
	Righ	nt-click for more registers and options

Figure 46. Secure register location

0 = nonsecure

1 = secure





10.5 Connection issue to STM32L552ZE-Q when RDP is set to 0.5

The EWARM is able to connect to the device and debug the nonsecure application. To connect to the STM32L552ZE-Q, proceed as follows:

- 1. Setting the option bytes, illustrated in Figure 47:
 - TZEN = 1
 - DBANK = 1
 - SECWM2_STRT = 0x1
 - SECWM1_PEND = 0x0.

Figure 47. Configuration of option bytes using STM32CubeProgrammer v2.2.0

Prg STN	/I32CubeProgrammer				- 🗆 ×
STM32	Programmer			f 🖸 🦄	• 🛪 ភ
	Option bytes				Connected
	 Read Out Protection 			ST-LINK	Disconnect
	Name	Value	Description	CT. 171	
2	RDP	BB 👻	Read protection option byte The read protection is used to protect the software code stored in Flash memory. AA : Level 0, no protection 55 : Level 0.5. read protection not active, only non-secure debug access is possible. Only available when T	Serial number	K configuration
OB			BB : Level 1, read protection of memories CC : Level 2, chip protection	Port	SWD 🗸
			>	Frequency (kHz)	24000 -
	 BOR Level User Configuration 			Mode	Under reset 🔹
	▶ Secure Area 1			Access port	0
	▶ Write Protection 1			Reset mode	Hardware reset 🔹
	Secure Area 2			Shared	Dirabled
	Write Protection 2			Esternal leader	Uisabieu
			Apply Read	Target voltage	

- 2. Load the nonsecure binary (at 0x08040000) then load the secure binary (at 0x0C000000) as specified in the section above.
- 3. Using STM32CubeProgrammer to set RDP=0x55 to reduce debug to nonsecure (see Figure 48).

Figure 48. RDP=0.5

Prg STN	132CubeProgrammer				– 🗆 ×
STM32 Cube	Programmer			F 🕒 🗄	* 🛪 🌆
	Option bytes				Connected
	 Read Out Protection 			ST-LINK	 Disconnect
	Name	Value	Description		
			Read protection option byte The read protection is used to protect the software code stored in Flash memory.	ST-LIN	VK configuration
	RDP BB 👻	AA : Level 0, no protection		002A001A 💋	
OB			D): Level U.S. read protection not active, only non-secure debug access is possible. Only available when it BB: Level 1, read protection of memories CC: Level 2, chip protection	Port	SWD 🗸
			>	Frequency (kHz)	24000 👻
	BOR Level			Mode	Under reset <
	 Secure Area 1 			Access port	0 -
	Write Protection 1			Reset mode	Hardware reset 🔹
	Secure Area 2			Shared	
	Write Protection 2				
				External loader	-
			Apply Read	Target voltage	3.27 V



4. Change the Reset mode to software reset: **Project options / ST-LINK** in the "Setup" tab select "Software" from the "Reset" field as illustrated in Figure 49.

Options for node "Project_r	1S"	×
Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator CADI CMSIS DAP GDB Server I-jet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDS	Setup Communication Breakpoints Emulator ST-LINK/V3 Serial no: Shared mode Always prompt for probe sele Reset Software Interface Interface speed SWD Default Access Pot Auto Specify	Ection

Figure 49. Reset mode selection



5. Connect to the device in Hot-plug mode from: Project / Attach to the Running Target (see Figure 50).

😌 Project - IAR Em	bedd	ed V	Vorkbencl	h IDE - A	rm 8.40.1		
File Edit View	Proj	ject	ST-Link	Tools	Window	/ Help	_
1 🗅 🔛 🕋 🛛		Ado	d Files				-
Workspace	6	Ado	d Group				.tx
STM32L552E-EV_Te	[+]	Imp	ort File Li	st			
Files		Ado	d Project C	onnecti	on		
🗆 💼 Project ns		Edi	t Configui	rations			
⊢⊞ ∎ Doc	×	Rer	nove				
-⊕ ≡ Example	t)	Cre	ate New P	roject			Ę
└─⊞ 📹 Output	6	Ado	d Existing	Project			
	۵	Op	tions			ALT+F7	
		Ver	sion Cont	rol Syste	m)	•
	0	Ma	ke			F7	Ę
		Cor	mpile			CTRL+F7	
	0	Ret	ouild All				
	₫	Cle	an				
	ê	Bat	ch build			F8	
		C-S	TAT Static	Analysis)	
	8	Sto	p Build			CTRL+ATTN	
	0	Do	wnload ar	nd Debu	g	CTRL+D	
	٠	Del	bug witho	ut Dowr	nloading		
	\odot	Atta	ach to Rur	nning Tai	rget		
	-						

Figure 50. Attach to running target option

Note: IDEs do not support the nonsecure flash reprogramming in RDP level 0.5, only STM32Cubeprogrammer allows it.



11 Using CubeIDE for Cortex[®]-M33 with TrustZone[®]

This part is explained in the *Getting started with STM32 development in CubeIDE* (AN5394), which is available on *www.st.com*.

Revision history

Table 1. Document revision history

Date	Version	Changes
21-Feb-2020	1	Initial release.
01-Aug-2022	2	Updated Figure 42. Project batch build order
15-May-2023	3	Updated: • Product series to include STM32U5 series • Section 1: General information
12-Sep-2023	4	Updated: Title Section 1: General information
23-Jul-2024	5	Updated: Title Section 1: General information
21-Jan-2025	6	Updated: • Product series to include STM32U3 series



Contents

1	Gene	eral information	2		
2	Arm	[®] Cortex [®] -M33 core overview	3		
3	Trus	tZone [®] concept of the Armv8-M	4		
4	SAU	/ IDAU - TrustZone [®] concept	5		
5	Debu	ebugging modes			
	5.1	Invasive debug	6		
	5.2	Non-invasive debug	6		
6	Debug access				
	6.1	Secure debug access	7		
	6.2	Nonsecure debug access	7		
7	Flas	h memory protection	8		
	7.1	Readout protection level when $TrustZone^{\mathbb{R}}$ is disabled \dots	8		
	7.2	RDP level transition scheme when TrustZone [®] is disabled	8		
	7.3	Readout protection level when TrustZone [®] is enabled	8		
	7.4	RDP level transition scheme when $TrustZone^{\texttt{®}}$ is enabled \ldots	9		
8	Star	ting with secure/nonsecure project	10		
9	Usin	g MDK-ARM for Cortex [®] -M33 with Trust Zone	11		
	9.1	Secure project settings	11		
	9.2	Nonsecure project settings	17		
		9.2.1 Building a project	22		
	9.3	Execute from secure code to nonsecure code			
10	Usin	g EWARM for Cortex M33 with TrustZone [®]			
	10.1	Secure project settings			
	10.2	Nonsecure project settings	34		
	10.3	Build projects	37		
	10.4	Execute from secure code to nonsecure code	39		
	10.5	Connection issue to STM32L552ZE-Q when RDP is set to 0.5	42		
11	Usin	g CubeIDE for Cortex [®] -M33 with TrustZone [®]	45		
Rev	ision	history			
List	of fig	ures			



List of figures

Figure 1.	Security state in Armv8-M	. 4
Figure 2.	RDP level transition scheme when TrustZone [®] is disabled (TZEN = 0) $\dots \dots \dots$. 8
Figure 3.	RDP level transition scheme when TrustZone [®] is disabled (TZEN = 1) $\dots \dots \dots$. 9
Figure 4.	Configuration of option bytes using STM32CubeProgrammer	10
Figure 5.	MDK-ARM project structure	11
Figure 6.	Secure project selection.	11
Figure 7.	Device selection	12
Figure 8.	Project_s target options	13
Figure 9.	Project_s Linker configuration	14
Figure 10.	Scatter file sample	15
Figure 11.	Target options debug	15
Figure 12.	Debug configuration	16
Figure 13.	Flash-loader settings	17
Figure 14.	Project_ns nonsecure project selection	17
Figure 15.	Device selection	18
Figure 16.	Memory configuration	19
Figure 17.	Linker options.	20
Figure 18.	Scatter file sample	20
Figure 19.	Debug settings	21
Figure 20.	FlashLoader configuration	21
Figure 21.	Project batch setup	22
Figure 22.	Project build ordering	22
Figure 23.	Build both projects in one step	22
Figure 24.	STM32L562E-DK Discovery board in connected status	23
Figure 25.	Load the nonsecure binary	23
Figure 26.	Download and debug button.	24
Figure 27.	Main.c sample code	24
Figure 28.	Code switch to nonsecure code status.	25
Figure 29.	CPU status	25
Figure 30.	EWARM v8.40.1 project explorer view	26
Figure 31.	Setting the project to active status.	27
Figure 32.	Device selection	28
Figure 33.		29
Figure 34.	Project debugger setup	30
Figure 35.	FlashLoader selection	31
Figure 36.	Selecting the nonsecure output file as an extra image	32
Figure 37.	Project setup	33
Figure 38.	Project set up: general options	34
Figure 39.		35
Figure 40.		36
Figure 41.		37
Figure 42.		38
Figure 43.	STM32L562E-DK Discovery board in connected status	39
Figure 44.		40
Figure 45.		40
Figure 46.	Configuration of ontion buton using STM22CubeDrogrammer v2.2.0	41
Figure 47.		42
Figure 40.		42
Figure 49.		43
i iyure 30.		44

IMPORTANT NOTICE - READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2025 STMicroelectronics – All rights reserved