AN5556

Application note

Getting started with IAR Embedded Workbench® and MDK-ARM software development for STM32WL5x/Ex MCUs

Introduction

This application note provides guidelines on how to used the software toolchains IAR Embedded Workbench® for Arm® (EWARM) and MDK-ARM from Keil®, with the STM32WL5x/Ex dual-core microcontrollers (MCUs).

1 Prerequisites

The following tools must be installed before starting this tutorial :

- EWARM IDE available for download from the official IAR System® web site
- MDK-ARM IDE available for download from the official Arm® Keil® web site
- STM32CubeProgrammer (STM32CubeProg) available for download from the official ST web site
- STM32CubeWL MCU Package available for download from the official ST web site
- ST-LINK server available for download from the official ST web site

The documents listed below may also help:

- STM32WL5x reference manual (RM0453)
- STM32WL55xx STM32WL54xx datasheet (DS13293)
- Cortex®-M4 and M0+ Technical Reference Manuals (available on Arm® web site)

Note: *Arm is a registered trademark of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.*

The Arm word and logo are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved.

2 STM32WL5x/Ex architecture

The STM32WL5x/Ex dual-core microcontrollers use an heterogeneous core architecture, that is composed of a Cortex-M4 core (named CPU1) and a Cortex-M0+ core (named CPU2).

These two cores boot always separately. The dual-core debug allows a simultaneous debugging of both cores using a single hardware debug probe.

2.1 Access port

The devices contain two access ports (AP):

- AP0: CPU1 debug access port (AHB-AP)
- AP1: CPU2 debug access port (AHB-AP)

2.2 CPU2 boot

The CPU2 boots after the CPU1 has set the C2BOOT bit in the power control register 4 (PWR_CR4). The C2BOOT value is retained in Standby mode and the CPU2 boots accordingly when exit from Standby. When security is enabled through the option byte, the CPU2 also boots when an illegal access has been detected.

The CPU2 can boot from the system memory (RSS/SFI) or anywhere in the user Flash memory (0x0800 0000), SRAM1 (0x2000 0000) or SRAM2 (0x2000 8000).

2.3 Debug support

The figure below shows the logical partitioning of the debug infrastructure, with two interfaces: serial wire and JTAG debug port.

Figure 1. Block diagram of debug support infrastructure

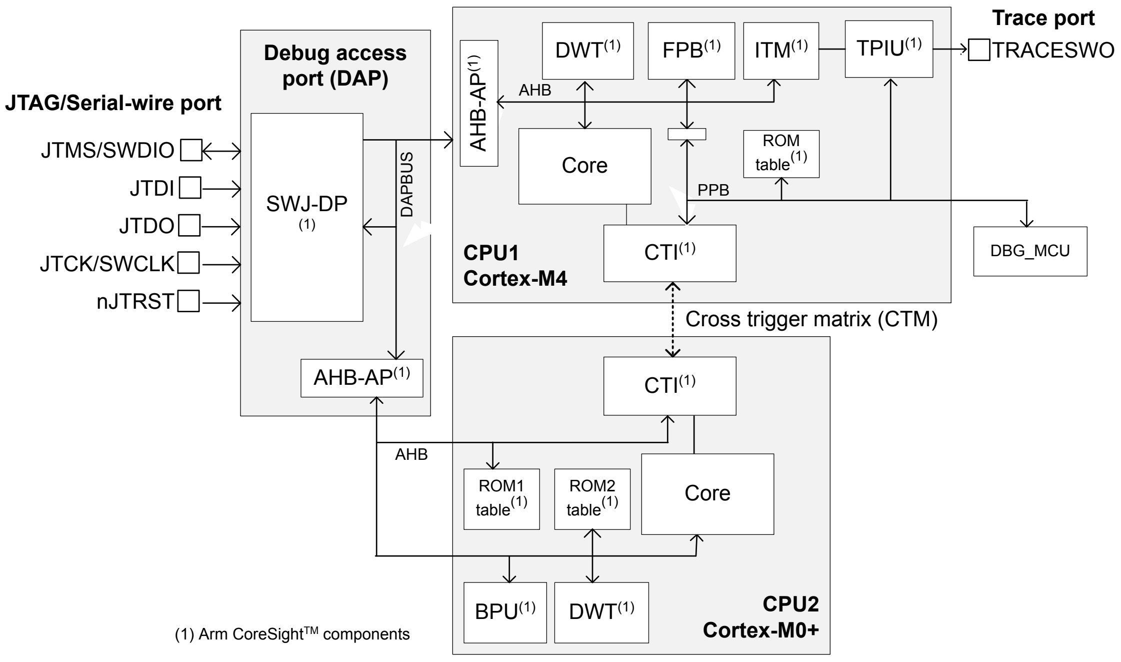


Table 1. **Debug features of Cortex-M cores**

CPU1 (Cortex-M4)	CPU2 (Cortex-M0+)
• ROM table • System control space (SCS) • Breakpoint unit (FPB) • Data watchpoint and trace unit (DWT) • Instrumentation trace macrocell (ITM) • Trace port interface unit (TPIU) • Cross trigger interface (CTI)	• ROM table • System control space (SCS) • Breakpoint unit (FPB) • Data watchpoint and trace unit (DWT) • Cross trigger interface (CTI)

The device-level debug features are controlled in the DBGMCU registers that are only accessible by the CPU1.

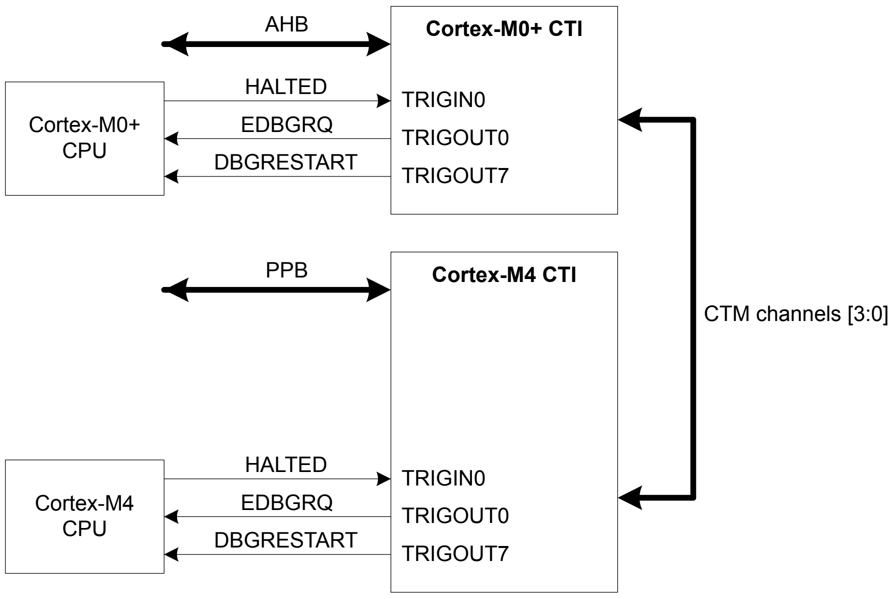
2.4 Cross-trigger interface (CTI)

There are two CTI components in the STM32WL5x/Ex: one dedicated to the CPU1 and one dedicated to the CPU2.

These CTIs are connected to each other via the cross-trigger matrix (CTM) and they enable events from various sources to trigger the debug activity.

For example, a breakpoint reached in one of the processor cores can stop the other processor.

Figure 2. **Embedded cross trigger**



2.5 Global security controller (GTZC)

The STM32WL5x/Ex devices support security with isolation between:

- a **secure world** managed by the CPU2, where usually security sensitive applications are run and critical resources are located
- a **non-secure or public world** handled by the CPU1, where non-secure transactions are used

2.6 Secure configuration (ESE/FSD option bits)

All or a part of the Flash memory and SRAM1, SRAM2 memories can be made secure, exclusively accessible for read and write by secured master: CPU2 and DMA channels (that have been configured as secured).

The secure CPU2 can only execute from secure areas, whereas the CPU1 can only execute from non-secure areas.

Note: *The CPU2 security is enabled when part or all of the Flash memory is secure (FSD = 0) or when option byte loading fails. In this case, the ESE bit in FLASH_OPTR is set.*

Change the CPU2 security mode

The CPU2 security is easily enabled by loading the user option FSD with 0 (security applied in any RDP level). The CPU2 security start address can be modified by the secure CPU2, by loading a new user option SFSA. To totally disable the security, the CPU2 sets the FSD bit to 1.

Note: Prior to removing security from part or all of the memory, a good practice consists in page erasing the Flash memory part that becomes non-secure.

Non-secure CPU1 and secure CPU2 can both remove security, by clearing the ESE bit to 0 in FLASH_OPTR and regressing the RDP level from Level 1 to Level 0. In this case, the main Flash memory backup registers (RTC_BKPxR), SRAM1, SRAM2 and PKA SRAM are erased.

Secure Flash memory area

The secure Flash memory area has a 2-Kbyte granularity sector, and is defined by the secure Flash start-address user option, controlled from the SFSA[6:0] in FLASH_SFR. In the same register, the FSD bit controls if the Flash memory security is enabled. When enabled, the whole system security framework is activated.

Secure SRAM areas

The SRAM1 and SRAM2 areas are only secure when the Flash memory security is enabled (ESE = 1 and FSD = 0).

The secure SRAM1 and SRAM2 areas have a 1-Kbyte granularity. The backup SRAM2 can be configured as secure from the BRSD bit and its start address from SBRSA[4:0] in FLASG_SRRVR. The non-backup SRAM can be configured as secure from the NBRSD bit and its start address from SNBRSA[4:0] in FLASG_SRRVR.

Debug access

The selection for CPU2 debug access is independent from security:

- When the system is non-secure (ESE = 0), CPU1 and CPU2 can enable and disable the CPU2 debug access, via the DDS bit in FLASH_SFR. In this case, the C2SWDBGEN bit is not writable and its default value is debug enabled. The CPU2 DDS debug is enabled/disabled after restarting OBL.
- When the system is secure (ESE = 1), the CPU2 debug access can only be enabled by the secure CPU2 via DDS and C2SWDBGEN bits. The CPU2 debug can be disabled directly by the CPU2 via the DDS bit, and indirectly by both CPUs when regressing ESE. The CPU2 debug is enabled/disabled after restarting OBL.

3 Using EWARM

The EWARM (IAR Embedded Workbench for Arm) is installed by default in the folder `C:\Program Files (x86)\IAR Systems`.

The STM32WL5x/Ex dual-core devices are officially supported (without need of ST patches) on EWARM starting from version 8.50.9

In this section, EWARM v8.50.6 with an internal patch for STM32WL5x/Ex (EWARMv8_STM32WLxx_V4.7), and a project template from STM32Cube_FW_WL_V1.0.0 are used.

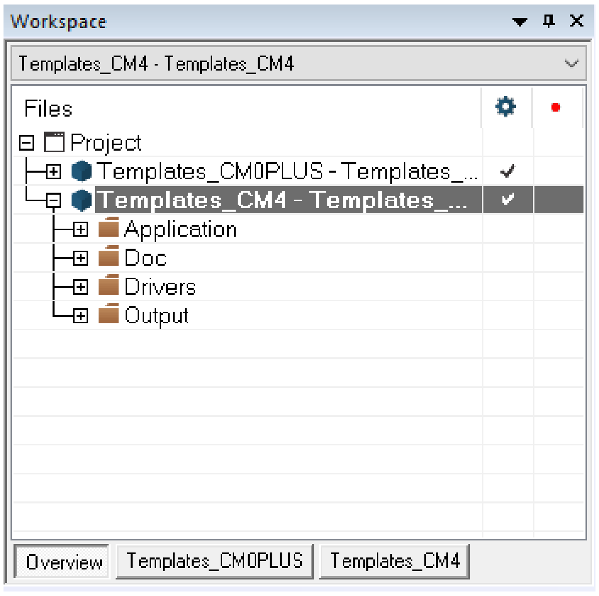
3.1 Dual-core debug steps (EWARM)

3.1.1 CM4 project settings (EWARM)

This section outlines the settings of the CM4 project, and uses a project template from STM32Cube_FW_WL_V1.0.0, named *Templates_CM4*.

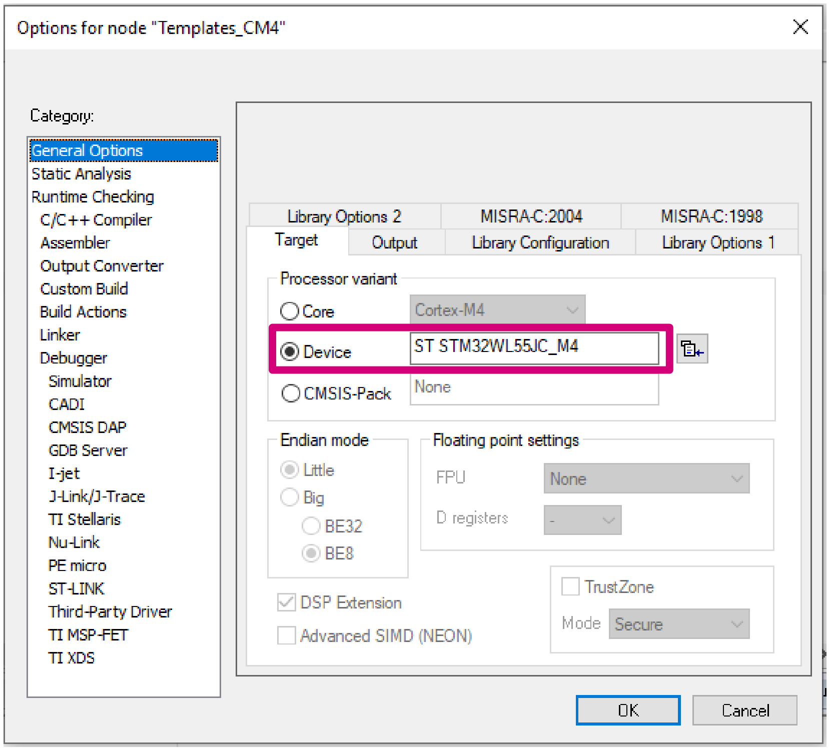
1. Open the *Template DualCore* project under `\STM32Cube_FW_WL_V1.0.0\Projects\NUCLEO-WL55JC\Templates\DualCore\EWARM`.
   This project is used to work on both cores at the same time. This project appears now in the *Project Explorer* view as shown in the figure below.

Figure 3. **EWARM project explorer view**


Workspace
Templates_CM4 - Templates_CM4
Files
Project
Templates_CM0PLUS - Templates_...
Templates_CM4 - Templates_...
Application
Doc
Drivers
Output
Overview
Templates_CM0PLUS
Templates_CM4

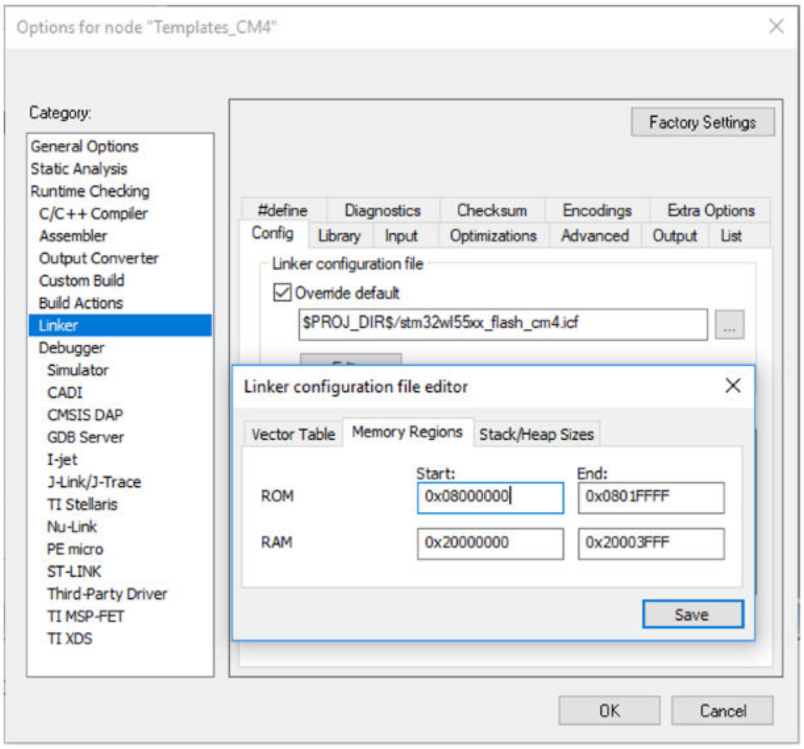
2. Set the *Templates_CM4* project as active, and ensure that the settings are compatible with the project options below.
   a. Go to *Project* > *Options*. On the *General Options*, select the STM32WL55JC_M4 device (see the figure below).

Figure 4. STM32WL55JC_M4 device selection



   b. Go to the *Linker* tab -> *Config Linker configuration file editor* section. Click *Edit* to display the *Linker configuration file editor*. Check that the application has been linked to the right address: Boot from main Flash memory at 0x0800 0000 and Boot from SRAM memory at 0x2000 0000.

Figure 5. **CM4 project - Linker configuration**



Linker configuration file

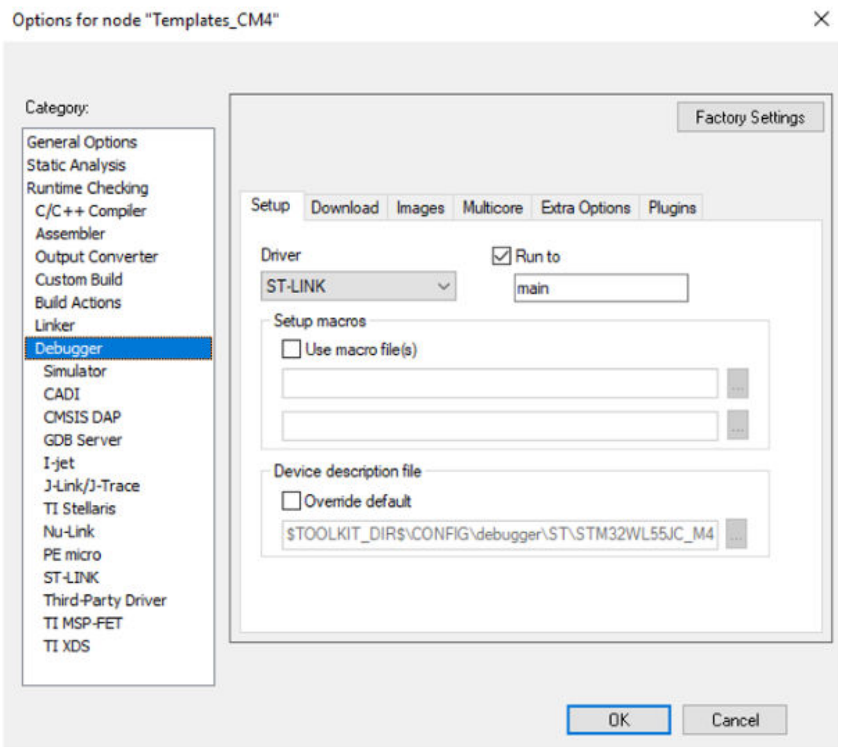
☑ Override default

$PROJ_DIR$/stm32wl55xx_flash_cm4.icf

	Start:	End:
ROM	0x08000000	0x0801FFFF
RAM	0x20000000	0x20003FFF

c. From *Debugger* tab, select *ST-LINK* as a debugger in the *Driver* field (the NUCLEO-WL55JC board contains an embedded debugger STLINK-V3).

Figure 6. **Debugger setup**



Driver: ST-LINK

☑ Run to: main

Setup macros

☐ Use macro file(s)

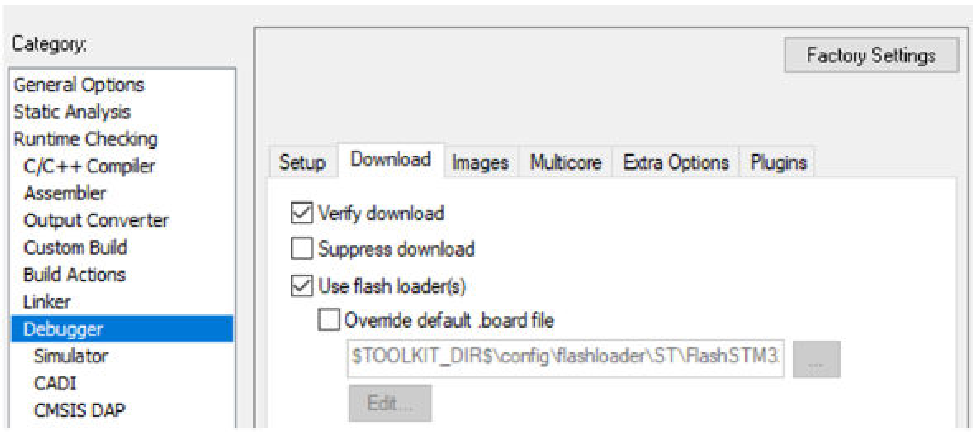
Device description file

☐ Override default

$TOOLKIT_DIR$\CONFIG\debugger\ST\STM32WL55JC_M4

d. Go to *Debugger -> Download* tab and tick the *Use flash loader(s)* check box.

Figure 7. **Flash loader selection**



Category: Debugger

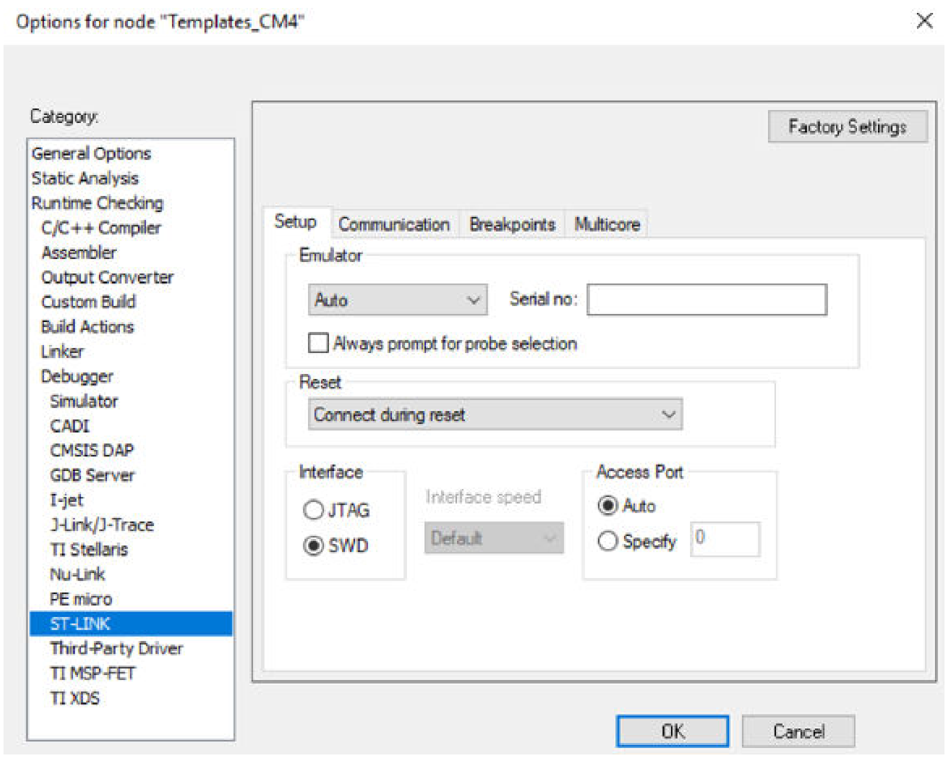
Download tab:

- [✓] Verify download
- [ ] Suppress download
- [✓] Use flash loader(s)
- [ ] Override default .board file

$TOOLKIT_DIR$\config\flashloader\ST\FlashSTM3

e. Go to *Project -> Options -> ST-LINK -> Setup*.

Figure 8. **ST-LINK setup**



Options for node "Templates_CM4"

Category: ST-LINK

Setup tab:

- Emulator: Auto
- Serial no:
- [ ] Always prompt for probe selection
- Reset: Connect during reset
- Interface: JTAG [✓ SWD]
- Interface speed: Default
- Access Port: [✓ Auto] Specify 0

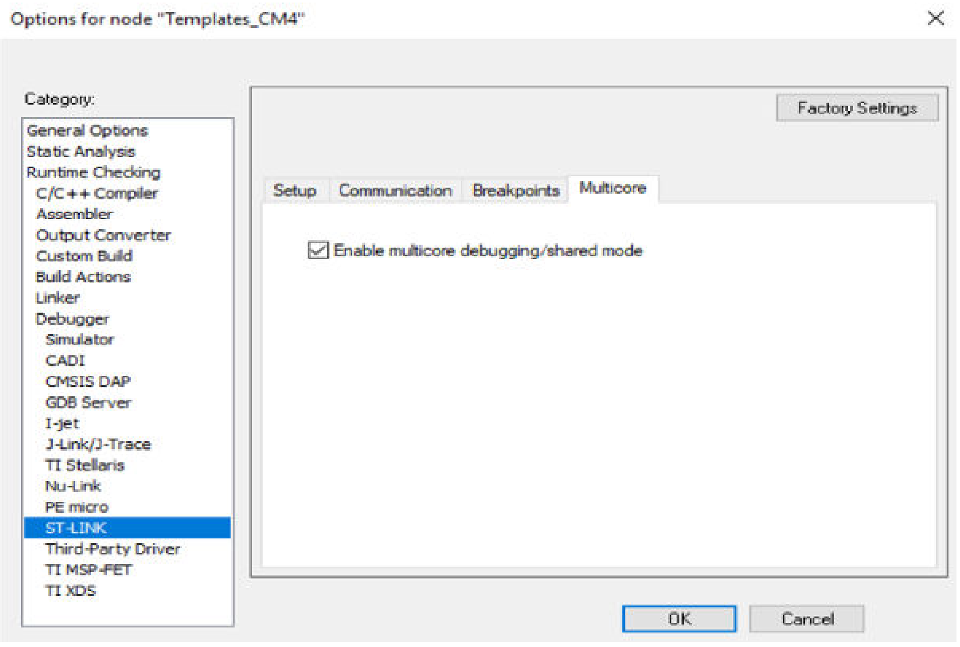
f. Select the *Access Port*:
   - *Auto*: The access port 0 for Cortex-M4 is automatically used.
   - *Manually*: It is possible to manually select the access port (by putting 0 with CM4, for example).

g. Select *SWD* in the communication *Interface* to use the serial-wire output (SWO) communication channel (fewer pins than JTAG).

h. Select the *Reset* type: With *Connect during reset*, the ST-LINK connects to the target while keeping reset active. Reset is pulled low and remains low while connecting to the target.

i. Go to the *Multicore* tab to enable the use of the *shared mode* to debug both cores simultaneously.

Figure 9. Shared mode activation


Options for node "Templates_CM4"

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

Factory Settings

Setup | Communication | Breakpoints | Multicore

☑ Enable multicore debugging/shared mode

OK | Cancel

Note: To set up for a multicore debugging using an ST-LINK debug probe, install the latest ST-LINK server version available on www.st.com.

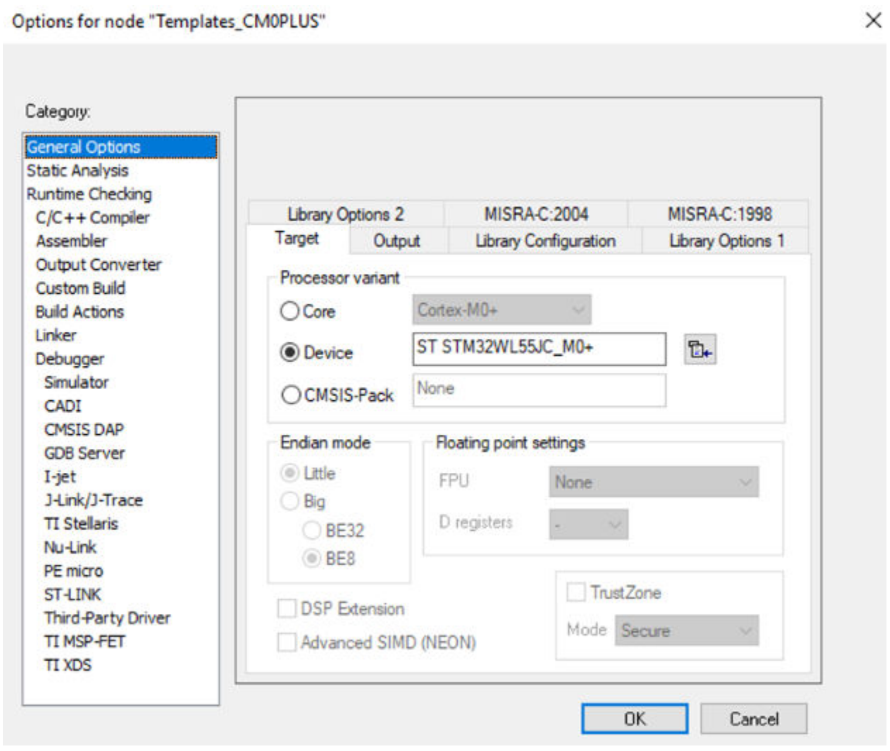
3.1.2 CM0+ project settings (EWARM)

This section outlines the settings of the CM0+ project, and uses a project template from STM32Cube_FW_WL_V1.0.0, named *Templates_CM0PLUS*.

Important: The CPU2 (Cortex-M0+) boots after the CPU1 (Cortex-M4) has set the C2BOOT bit in the power control register 4 (PWR_CR4). This allows the CPU1 to initialize the system after a reset or wakeup from system low-power mode, before booting the CPU2.

1. Open in another instance the *Templates_CM0PLUS* project under `\STM32Cube_FW_WL_V1.0.0\Projects\NUCLEO-WL55JC\Templates\DualCore\EWARM`, and ensure that the settings are compatible with the options below.
   a. Go to *Project* -> *Options*. On the *General Options*, select the STM32WL55JC_M0+ device (see the figure below).

Figure 10. ST32WL55JC_M0+ device selection



Options for node "Templates_CM0PLUS"

Category: General Options

Target tab — Processor variant: Device: ST STM32WL55JC_M0+

Core: Cortex-M0+

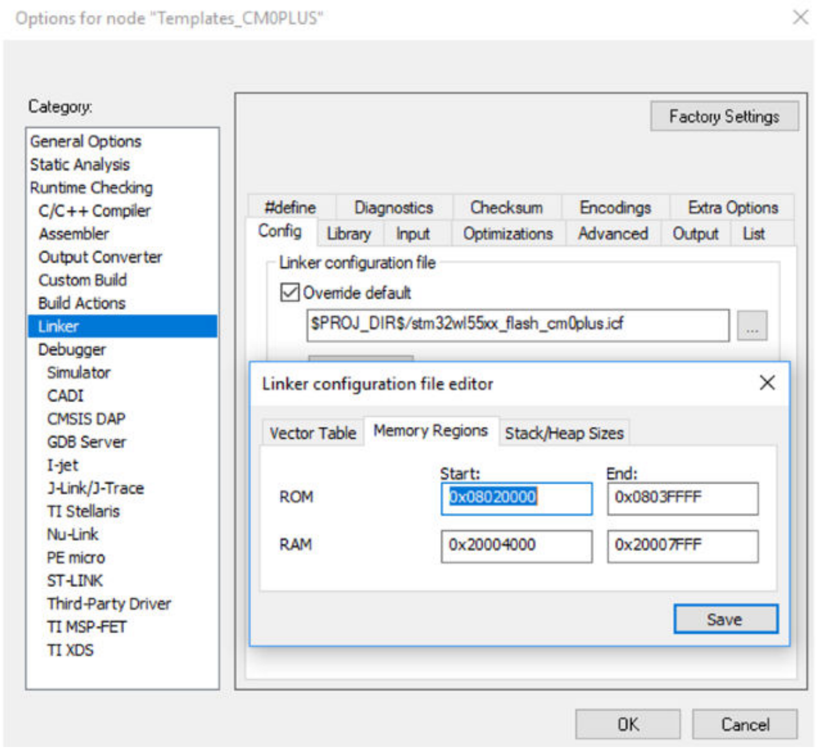
CMSIS-Pack: None

Endian mode: Little

Floating point settings: FPU: None

   b. Go to the *Linker* tab -> Config *Linker configuration file editor* section. Click *Edit* to display the *linker configuration file editor*. Check that the application has been linked to the right address: Boot from main Flash memory at 0x0802 0000 and Boot from SRAM memory at 0x2000 4000.

Figure 11. CM0+ project - Linker configuration



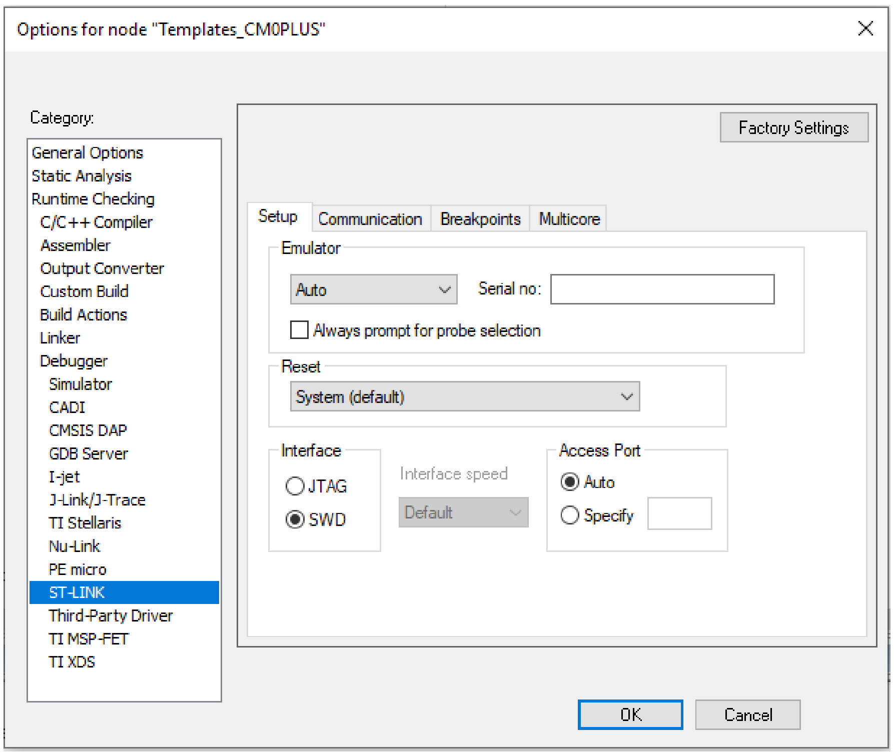
Linker configuration file
☑ Override default
$PROJ_DIR$/stm32wl55xx_flash_cm0plus.icf

Linker configuration file editor

	Start:	End:
ROM	0x08020000	0x0803FFFF
RAM	0x20004000	0x20007FFF

c. From *Debugger* tab, select *ST-LINK* as a debugger in the *Driver* field.
d. Go to *Debugger -> Download* tab and tick the *Use flash loader(s)* check box.
e. Go to *Project -> Options -> ST-LINK -> Setup*.

Figure 12. ST-LINK setup



Emulator: Auto
Serial no:
☐ Always prompt for probe selection
Reset: System (default)
Interface: ( ) JTAG [✓ SWD]
Interface speed: Default
Access Port: [✓ Auto] ( ) Specify

f. Select the *Access Port*:
- *Auto*: The access port 1 for Cortex-M0+ is automatically used.
- *Manually*: It is possible to manually select the access port (by putting 1 with CM0+, for example) .

g. Select *SWD* in the communication *Interface* to use the SWO communication channel.

h. Change the *Reset type* from software to system reset (resets the core and peripherals).

Note: This example uses software reset by default as it was developed with an old IDE version that contains some limitation with system reset

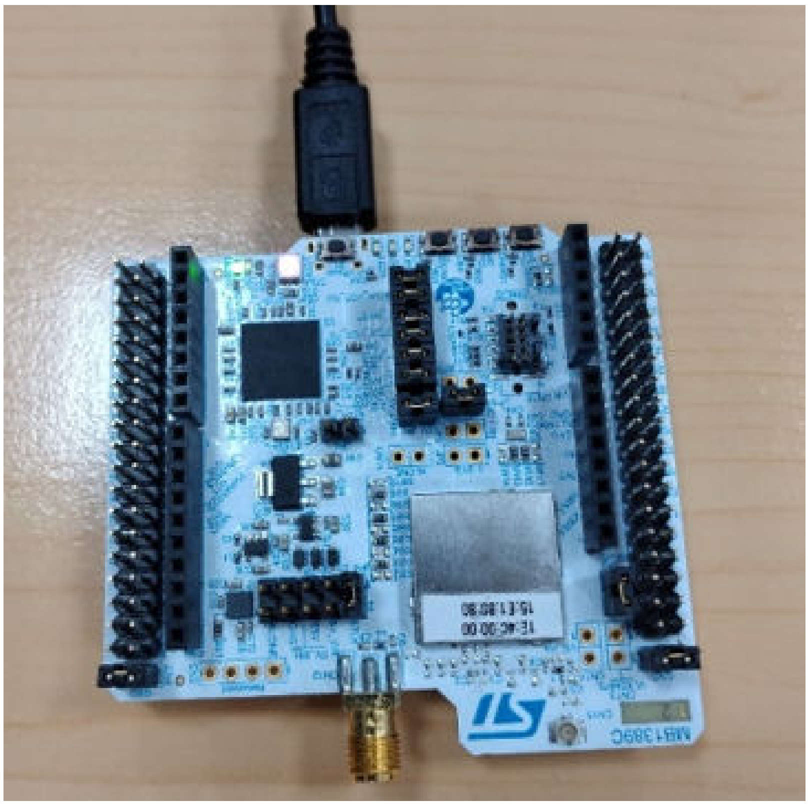
i. Go to the *Multicore* tab to enable the shared mode by checking the *Enable multicore debugging/shared mode* option .

3.1.3 Loading and debugging both projects (EWARM)

Before downloading the project, connect to the NUCLEO-WL55JC board:

- Connect the STLINK-V3E programming and debugging tool on the NUCLEO-WL55JC board.
- Plug the USB cable to the CN1 USB STLINK connector of the board.
- LED6 lights up red when the ST-LINK is connected.

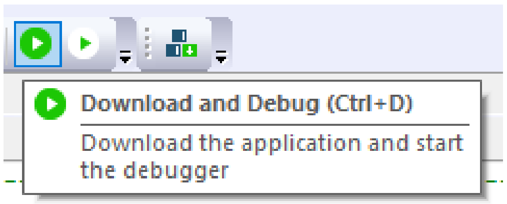
Figure 13. **STM32WL55JC NUCLEO board in connected status**



Note: This figure is not contractual.

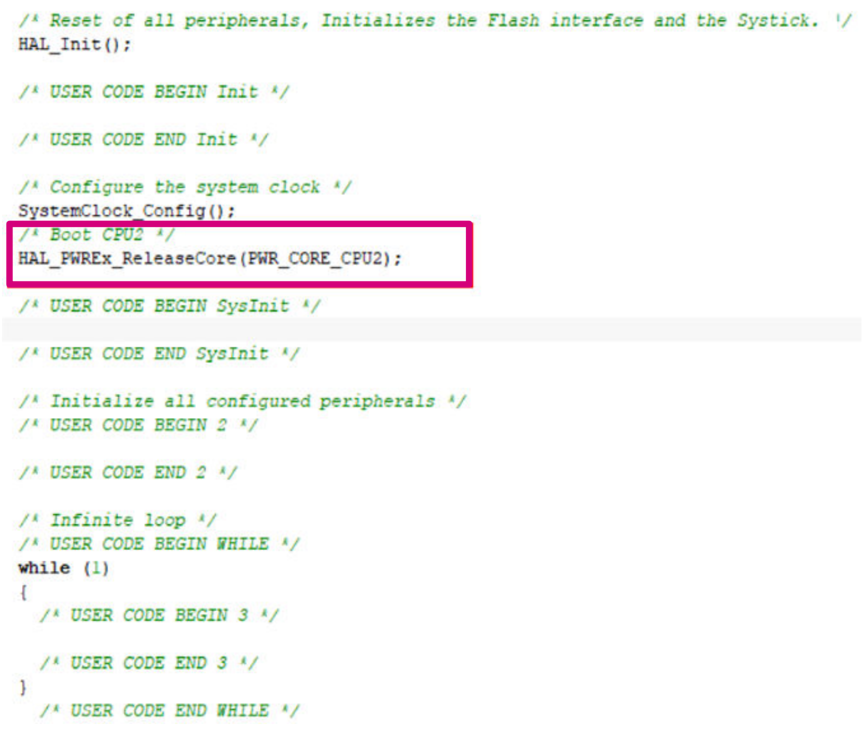
1. Download the *Templates_CM4* project and start a debug session by pressing the download and debug button.

Figure 14. **Download and Debug button**


Download and Debug (Ctrl+D)
Download the application and start the debugger

2. Execute the code until the instruction that sets the C2BOOT (see the figure below). This project boots the Cortex-M0+ by setting the C2BOOT bit in PWR_CR4.

Figure 15. **Release the CPU2 from holding**



```
/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();
/* Boot CPU2 */
HAL_PWREx_ReleaseCore(PWR_CORE_CPU2);

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE BEGIN 3 */

  /* USER CODE END 3 */
}
  /* USER CODE END WHILE */
```

3. Download the *Templates_CM0PPLUS* project.
4. Start a debug session.

3.2 Known limitations (EWARM)

- When using the software reset option, the first time, to download with the Cortex-M0+ (Flash memory empty), the download is done with success but there is a HardFault message that is displayed all along the debug session.
- The system reset option is not functional from debug session: a HardFault error is generated when applying a system reset from a debug session.

4 Using MDK-ARM

The MDK-ARM (from Keil) is installed by default in the `C:\Keil` directory: the installer creates a start menu µVision® 5 shortcut.

In this section, MDK-ARM v5.32 with an internal pack for STM32WL5x/Ex (Keil.STM32WLxx_DFP.1.0.8), and a project template from STM32Cube_FW_WL_V1.0.0 are used.

Note: All required packs are available for download from the official Arm Keil web site.

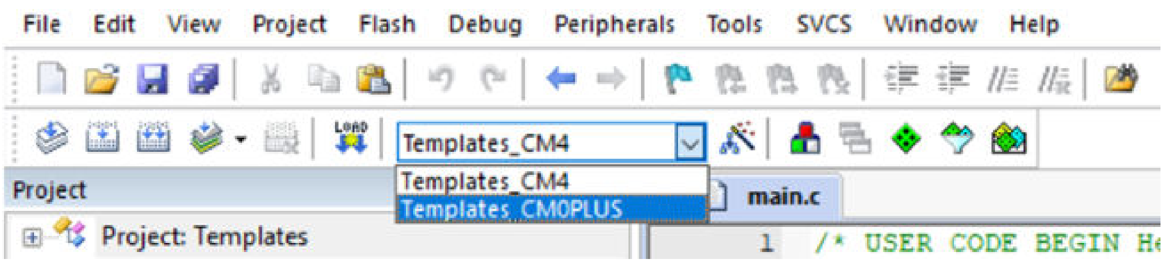
4.1 Dual-core debug steps (MDK-ARM)

4.1.1 CM4 project settings (MDK-ARM)

This section outlines the settings of the CM4 project, and uses a project template from STM32Cube_FW_WL_V1.0.0 named *Templates_CM4*.

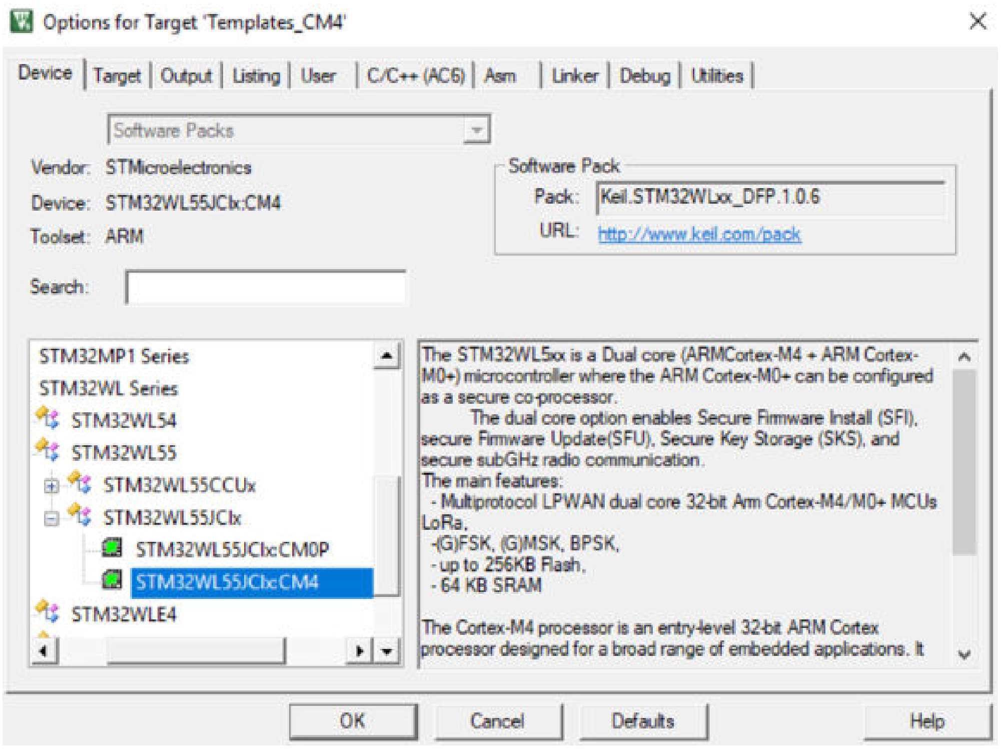
1. Open the *Template DualCore* project under `\STM32Cube_FW_WL_V1.0.0\Projects\NUCLEO-WL55JC\Templates\DualCore\MDK-ARM`.
   This project is used to work on both cores at the same time. This project appears now in the *Project Explorer* view as shown in the figure below.

Figure 16. Project explorer view



2. Set the *Templates_CM4* project as active, and ensure that the settings are compatible with the options below.
   a. Select the correct device by opening the *Configuration* window and selecting:
      Project -> Options for Target -> Device then select the STM32WL55JCIx:CM4 device from the list.

Figure 17. STM32WL55JCIx - CM4 device selection



Vendor: STMicroelectronics
Device: STM32WL55JClx:CM4
Toolset: ARM
Search:
Pack: Keil.STM32WLxx_DFP.1.0.6
URL: http://www.keil.com/pack

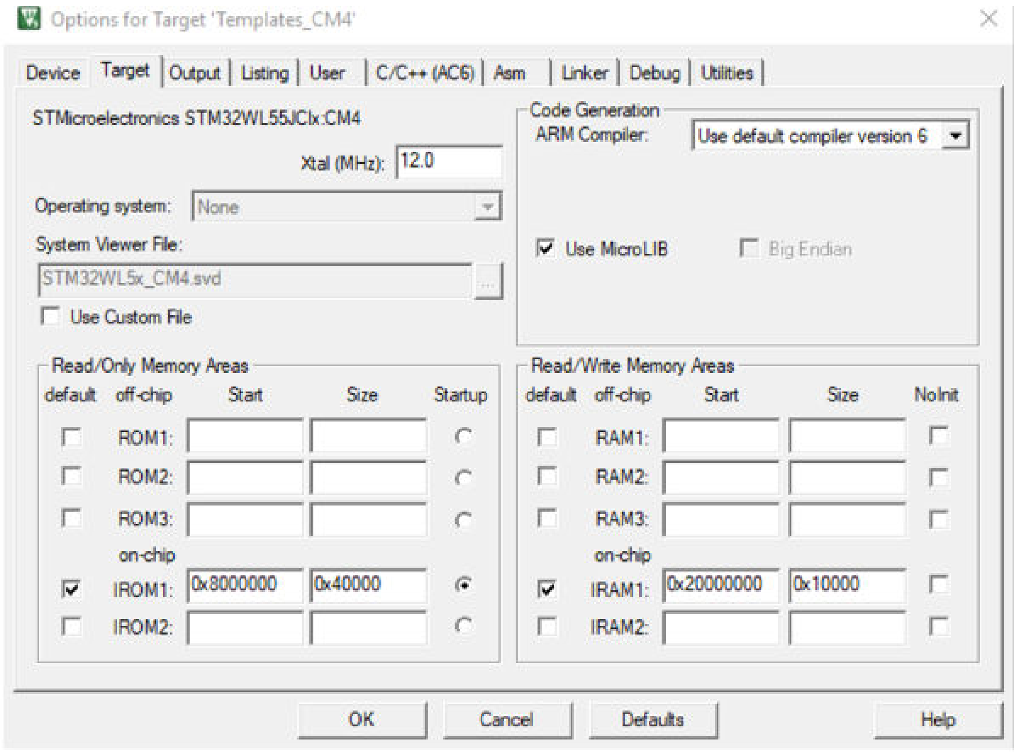
STM32MP1 Series
STM32WL Series
STM32WL54
STM32WL55
STM32WL55CCUx
STM32WL55JClx
STM32WL55JClx:CM0P
STM32WL55JClx:CM4
STM32WLE4

The STM32WL5xx is a Dual core (ARMCortex-M4 + ARM Cortex-M0+) microcontroller where the ARM Cortex-M0+ can be configured as a secure co-processor.
The dual core option enables Secure Firmware Install (SFI), secure Firmware Update(SFU), Secure Key Storage (SKS), and secure subGHz radio communication.
The main features:
- Multiprotocol LPWAN dual core 32-bit Arm Cortex-M4/M0+ MCUs LoRa,
-(G)FSK, (G)MSK, BPSK,
- up to 256KB Flash,
- 64 KB SRAM

The Cortex-M4 processor is an entry-level 32-bit ARM Cortex processor designed for a broad range of embedded applications. It

   b. From *Project -> Options for Target -> Target -> Read/Only Memory Areas* section, ensure that the right memory area is selected: Boot from main Flash memory at 0x0800 0000 and Boot from SRAM1 memory at 0x2000 0000.

Figure 18. Memory area selection



Options for Target 'Templates_CM4'

Device | Target | Output | Listing | User | C/C++ (AC6) | Asm | Linker | Debug | Utilities

STMicroelectronics STM32WL55JCIx:CM4

Xtal (MHz): 12.0

Operating system: None

System Viewer File: STM32WL5x_CM4.svd

☐ Use Custom File

Code Generation
ARM Compiler: Use default compiler version 6

☑ Use MicroLIB ☐ Big Endian

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
☑	IROM1:	0x8000000	0x40000	●
☐	IROM2:			○

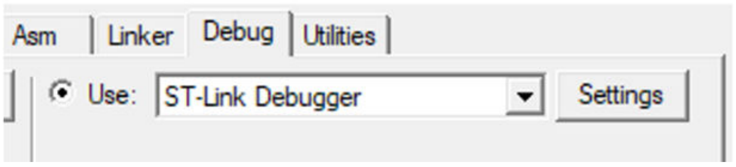
Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
☑	IRAM1:	0x20000000	0x10000	☐
☐	IRAM2:			☐

OK | Cancel | Defaults | Help

c. Select *ST-LINK Debugger* as the debugger from *Project -> Options* for *Target -> Debug*.
The NUCLEO-WL55JC board contains an embedded debugger ST-LINK V3.

Figure 19. Debug probe selection

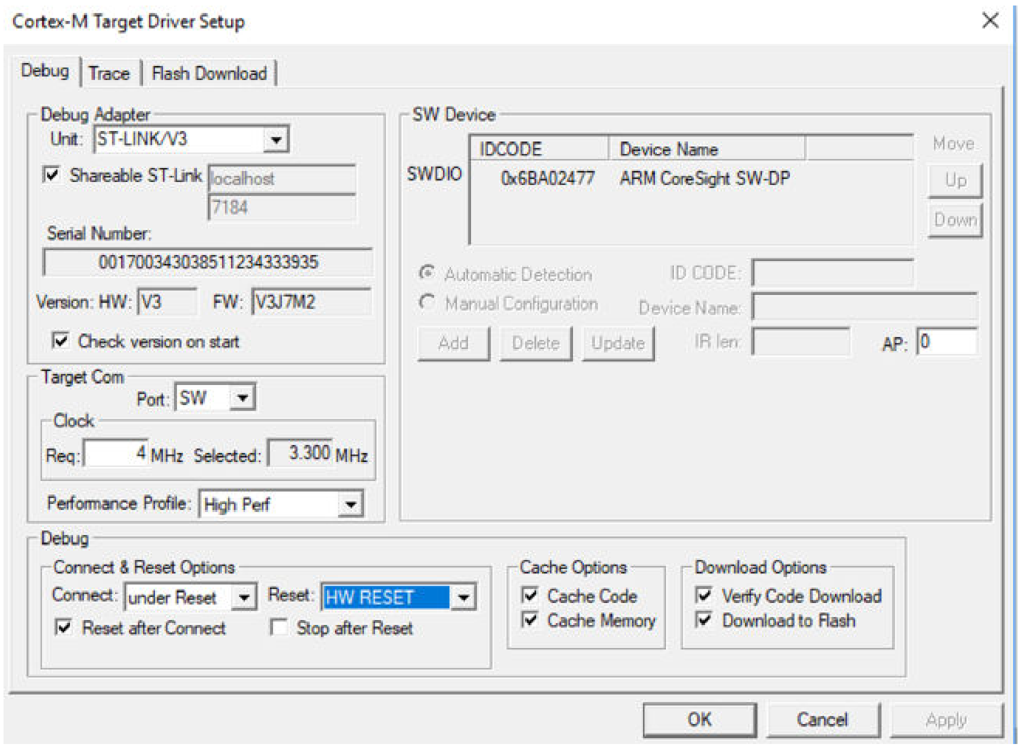


Asm | Linker | Debug | Utilities

● Use: ST-Link Debugger | Settings

d. Go to *Debugger -> Settings* and check that the debugger is connected as in the figure below.

Figure 20. **Debug settings tab**



Cortex-M Target Driver Setup

Debug | Trace | Flash Download

Debug Adapter
Unit: ST-LINK/V3
[✓] Shareable ST-Link localhost 7184
Serial Number: 001700343038511234333935
Version: HW: V3 FW: V3J7M2
[✓] Check version on start

Target Com
Port: SW
Clock
Req: 4 MHz Selected: 3.300 MHz
Performance Profile: High Perf

SW Device

	IDCODE	Device Name
SWDIO	0x6BA02477	ARM CoreSight SW-DP

[✓ Automatic Detection] Manual Configuration
ID CODE:
Device Name:
IR len:
AP: 0

Debug
Connect & Reset Options
Connect: under Reset Reset: HW RESET
[✓] Reset after Connect
[ ] Stop after Reset

Cache Options
[✓] Cache Code
[✓] Cache Memory

Download Options
[✓] Verify Code Download
[✓] Download to Flash

e. Enable the dual-core debug by checking *Shareable ST-Link* in order to be able to debug both cores simultaneously.

Note: To set up a multicore debugging using an ST-LINK debug probe, install the latest ST-LINK server.

f. Select the *Access Port* (port 0 for Cortex-M4).

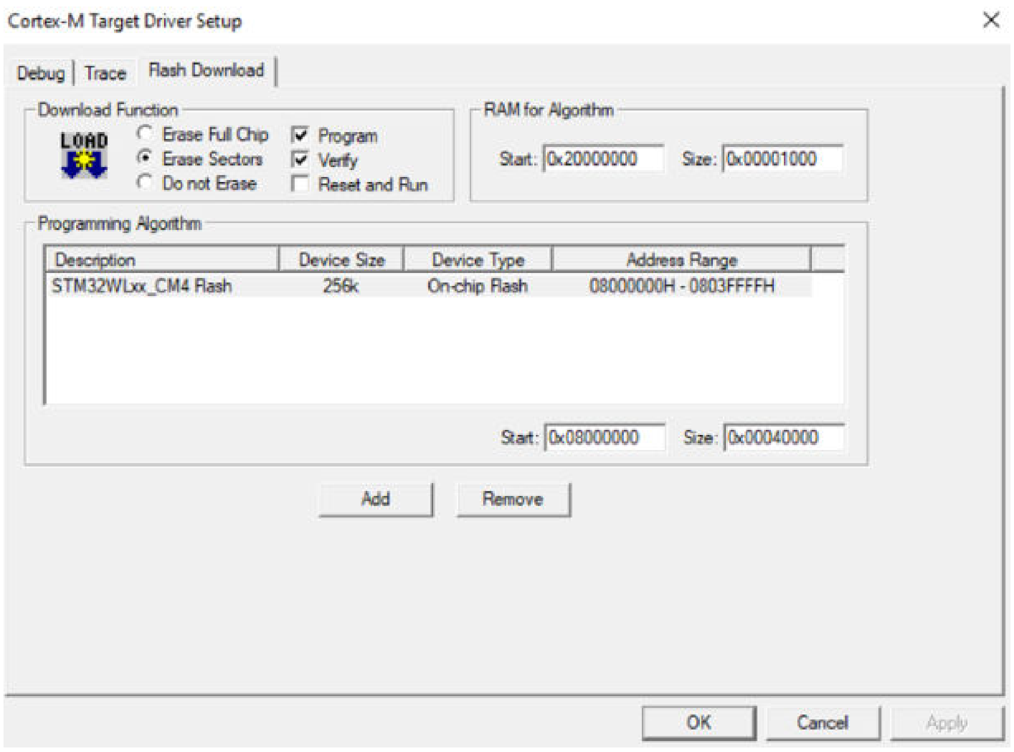
g. Select the communication *Interface* as SWD to use the serial-wire output (SWO) communication channel (fewer pins than JTAG).

h. Select the *Connect and Reset Options*.
- *Connect under reset* holds the hardware reset (*HW RESET*) signal active while connecting to the device.
- *HW RESET* performs a hardware reset by asserting the hardware reset (*HW RESET*) signal.

i. Select the *Download Options*.
- *Verify Code Download* stops the CPU after the currently executed instruction.
- *Download to Flash* downloads the code to all memory area.

j. From the *Flash Download* window shown in the figure below:
- *Download Function* is used to set 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 on-chip RAM.
- *Program Algorithm* contains the definitions for programming the Flash memory.

Figure 21. CM4 Flash loader settings



Cortex-M Target Driver Setup

Download Function
- ( ) Erase Full Chip
- (●) Erase Sectors
- ( ) Do not Erase
- [✓] Program
- [✓] Verify
- [ ] Reset and Run

RAM for Algorithm
Start: 0x20000000 Size: 0x00001000

Programming Algorithm

Description	Device Size	Device Type	Address Range
STM32WLxx_CM4 Flash	256k	On-chip Flash	08000000H - 0803FFFFH

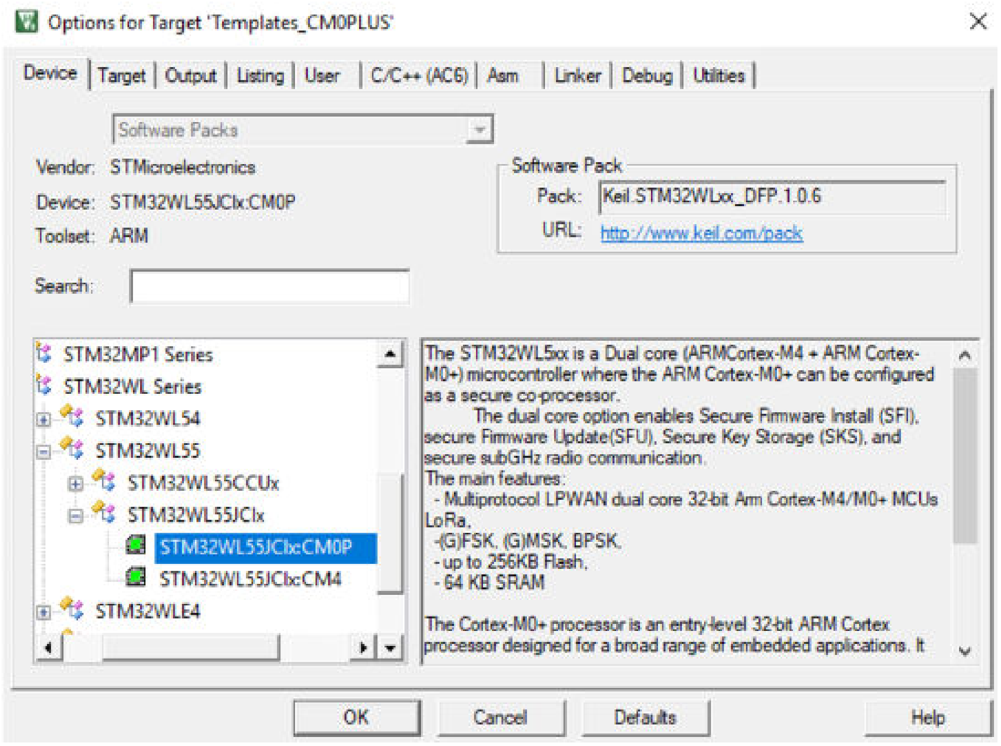
Start: 0x08000000 Size: 0x00040000

4.1.2 CM0+ project settings (MDK-ARM)

This section outlines the settings of the CM0+ project, and uses a project template from STM32Cube_FW_WL_V1.0.0 named *Templates_CM0PLUS*.

1. Open the *Templates_CM0PLUS* project in another instance and ensure that the settings are compatible with the options below.
   a. Go to *Project -> Options for Target 'Templates_CM0PLUS'*
   b. Select the correct device by opening the *Configuration* window and selecting *Project -> Options for Target -> Device* then select the STM32WL55JCIx:CM0P device from the list.

Figure 22. STM32WL55JCIx - CM0+ device selection



Options for Target 'Templates_CM0PLUS'

Device | Target | Output | Listing | User | C/C++ (AC6) | Asm | Linker | Debug | Utilities

Software Packs

Vendor: STMicroelectronics
Device: STM32WL55JCIx:CM0P
Toolset: ARM

Software Pack
Pack: Keil.STM32WLxx_DFP.1.0.6
URL: http://www.keil.com/pack

Search:

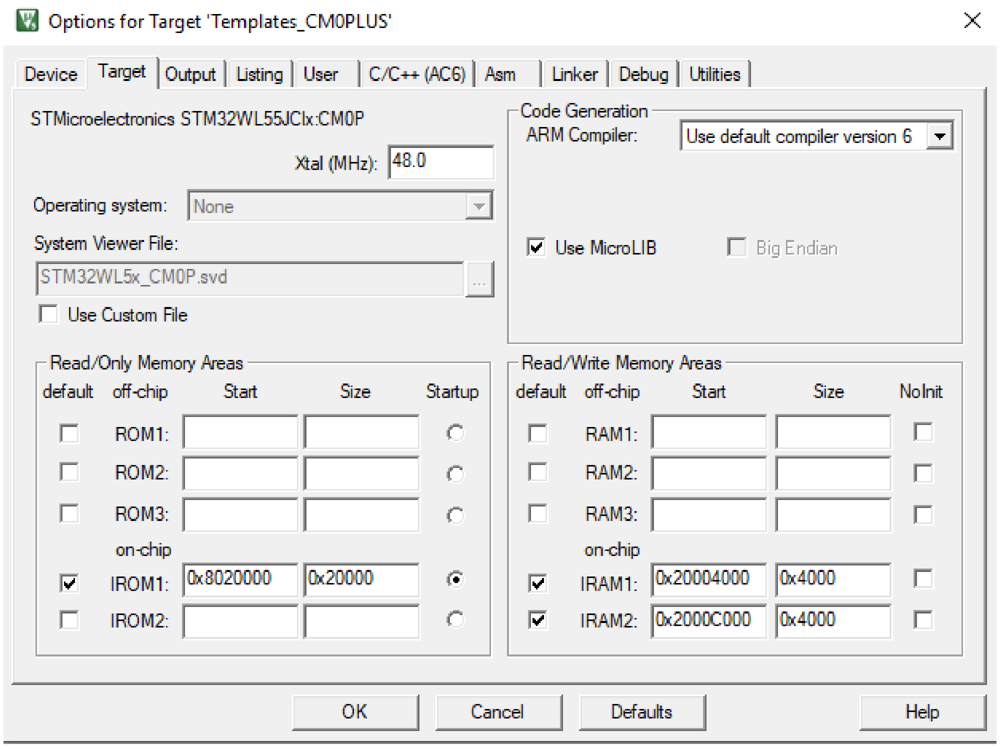
- STM32MP1 Series
- STM32WL Series
  - STM32WL54
  - STM32WL55
    - STM32WL55CCUx
    - STM32WL55JCIx
      - STM32WL55JCIx:CM0P
      - STM32WL55JCIx:CM4
  - STM32WLE4

The STM32WL5xx is a Dual core (ARMCortex-M4 + ARM Cortex-M0+) microcontroller where the ARM Cortex-M0+ can be configured as a secure co-processor.
The dual core option enables Secure Firmware Install (SFI), secure Firmware Update(SFU), Secure Key Storage (SKS), and secure subGHz radio communication.
The main features:
- Multiprotocol LPWAN dual core 32-bit Arm Cortex-M4/M0+ MCUs LoRa,
-(G)FSK, (G)MSK, BPSK,
- up to 256KB Flash,
- 64 KB SRAM

The Cortex-M0+ processor is an entry-level 32-bit ARM Cortex processor designed for a broad range of embedded applications. It

   c. From *Project -> Options for Target -> Target -> Read/Only Memory Areas* section, ensure that the right memory area is selected: Boot from main Flash memory at 0x0802 0000 and Boot from SRAM1 memory at 0x2000 4000.

Figure 23. **Memory area selection**


Options for Target 'Templates_CM0PLUS'

STMicroelectronics STM32WL55JCIx:CM0P

Xtal (MHz): 48.0

Operating system: None

System Viewer File: STM32WL5x_CM0P.svd

☐ Use Custom File

Code Generation
ARM Compiler: Use default compiler version 6

☑ Use MicroLIB ☐ Big Endian

Read/Only Memory Areas

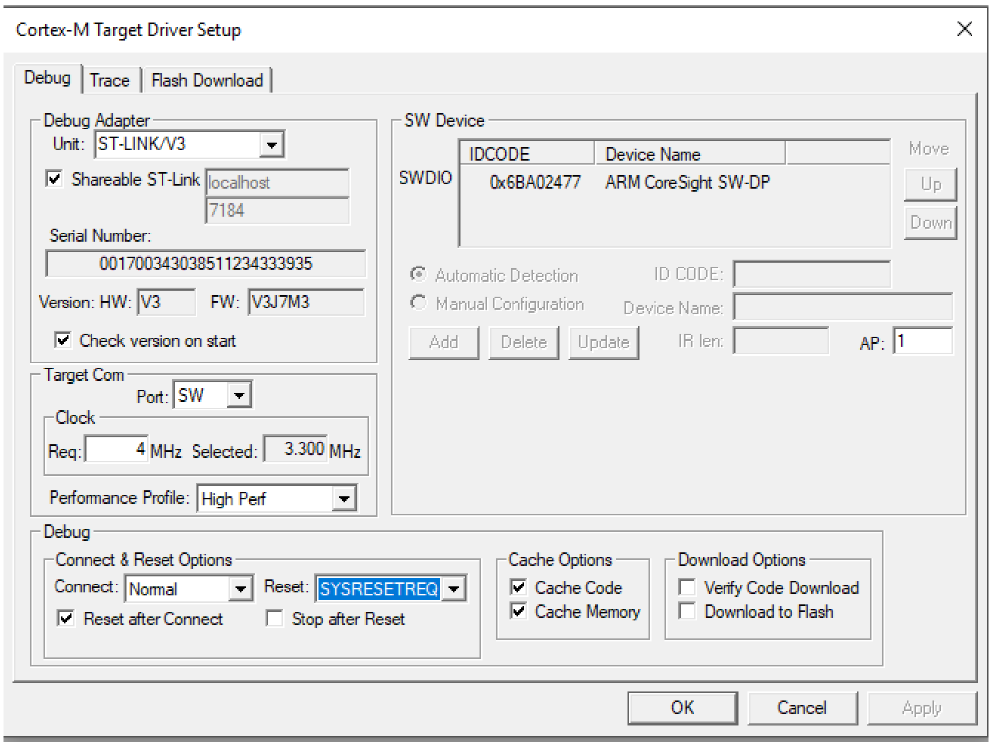
default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
☑	IROM1:	0x8020000	0x20000	●
☐	IROM2:			○

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
☑	IRAM1:	0x20004000	0x4000	☐
☑	IRAM2:	0x2000C000	0x4000	☐

d. Go to *Debugger -> Settings* and check that the debugger is connected as in the figure below.

Figure 24. **Debugger settings**


Cortex-M Target Driver Setup

Debug Adapter
Unit: ST-LINK/V3
☑ Shareable ST-Link localhost 7184
Serial Number: 001700343038511234333935
Version: HW: V3 FW: V3J7M3
☑ Check version on start

Target Com
Port: SW
Clock
Req: 4 MHz Selected: 3.300 MHz
Performance Profile: High Perf

SW Device

	IDCODE	Device Name
SWDIO	0x6BA02477	ARM CoreSight SW-DP

● Automatic Detection ID CODE:
○ Manual Configuration Device Name:

IR len: AP: 1

Debug
Connect & Reset Options
Connect: Normal Reset: SYSRESETREQ
☑ Reset after Connect ☐ Stop after Reset

Cache Options
☑ Cache Code
☑ Cache Memory

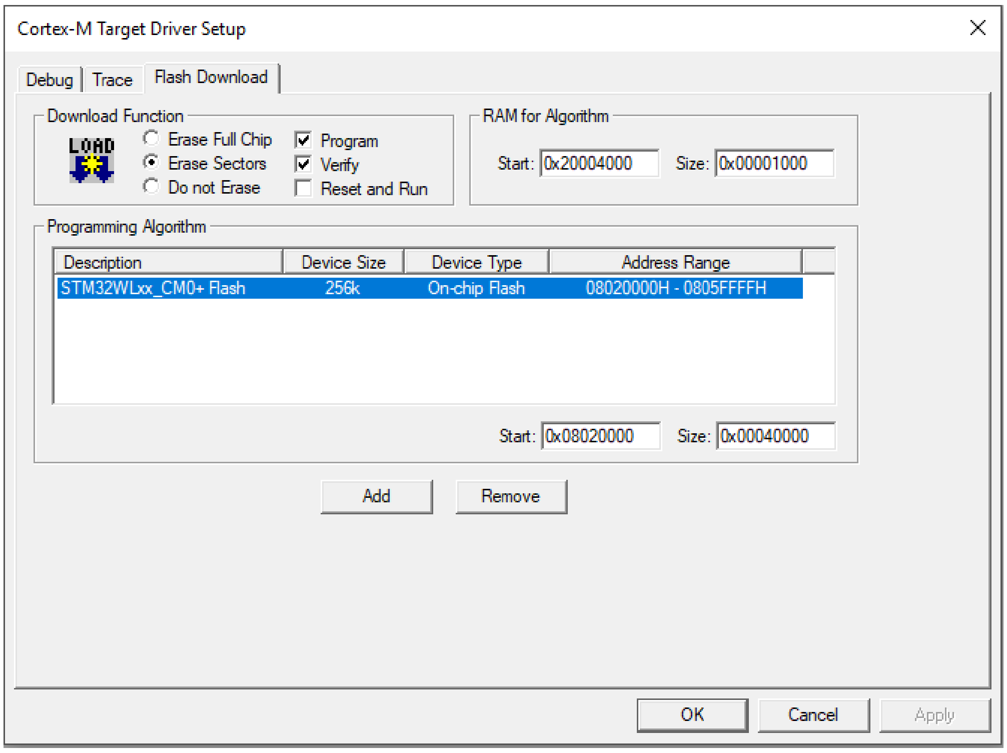
Download Options
☐ Verify Code Download
☐ Download to Flash

e. Enable the dual-core debug by checking *Shareable ST-Link* in order to be able to debug both cores simultaneously.

f. Select the *Access Port* (port 1 for Cortex-M0+).

g. Select the communication *Interface* as SWD to use the serial-wire output (SWO) communication channel (fewer pins than JTAG).
h. Select the *Connect and Reset Options*.
 - *Connect Normal* stops the CPU at the currently executed instruction after connecting.
 - *SYSRESETREQ* performs a software reset by setting the SYSRESETREQ bit. The Cortex-M0+ core and the on-chip peripherals are reset (the only reset mode supported with Cortex-M0+).
i. Select the *Download Options*.
 - *Verify Code Download* stops the CPU after the currently executed instruction.
 - *Download to Flash* downloads the code to all memory area.
j. From the *Flash Download* window shown in the figure below:
 - *Download Function* is used to set 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 on-chip RAM.
 - *Program Algorithm* contains the definitions for programming Flash memory.

Figure 25. **CM0+ Flash loader settings**



Download Function: Erase Full Chip [✓ Erase Sectors] Do not Erase; Program [✓], Verify [✓], Reset and Run [ ]

RAM for Algorithm: Start: 0x20004000 Size: 0x00001000

Programming Algorithm

Description	Device Size	Device Type	Address Range
STM32WLxx_CM0+ Flash	256k	On-chip Flash	08020000H - 0805FFFFH

Start: 0x08020000 Size: 0x00040000

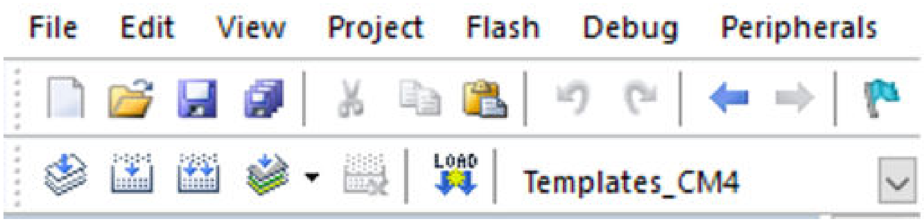
4.1.3 Loading and debugging both projects (MDK-ARM)

Before downloading the project, connect to the NUCLEO-WL55JC board (see Figure 13)

- Connect the STLINK-V3E programming and debugging tool on the NUCLEO-WL55JC board.
- Plug the USB cable to the CN1 USB STLINK connector of the board.
- LED6 lights up red when the ST-LINK is connected.

1. Build and download the *templates_CM4* project using *Build and Load* button

Figure 26. **Build and Load button**



2. Start a debug session, then execute the application until the instruction that sets the C2BOOT (see the figure below). This project boots the Cortex-M0+ by setting the C2BOOT bit in PWR_CR4.

Figure 27. **Start debug session**

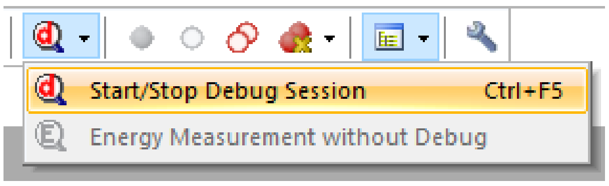
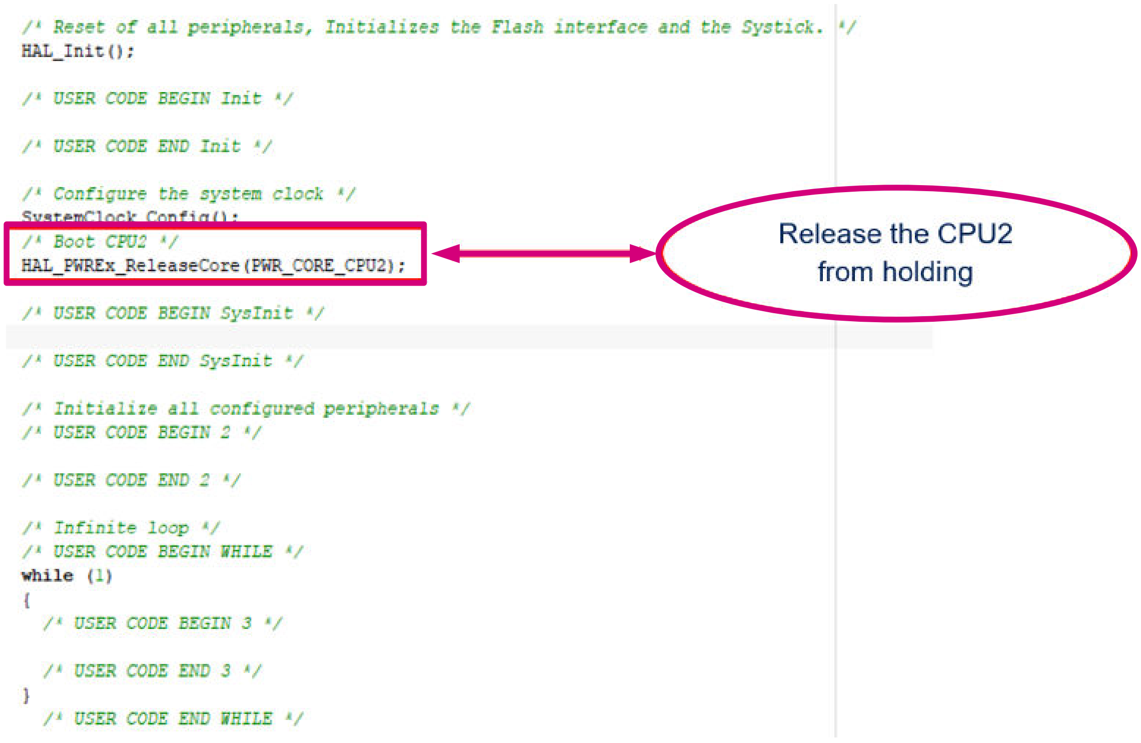

Start/Stop Debug Session Ctrl+F5
Energy Measurement without Debug

Figure 28. **Main.c sample code**



```
/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();
/* Boot CPU2 */
HAL_PWREx_ReleaseCore(PWR_CORE_CPU2);

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE BEGIN 3 */

  /* USER CODE END 3 */
}
  /* USER CODE END WHILE */
```

3. Build and download the *Templates_CM0PLUS* project.
4. Launch a debug session.

Note:

- *Connect to the two cores simultaneously in shareable mode is only possible if the option "Stop after Reset” is disabled to be sure that the C2BOOT is not reset. The Cortex-M0 only boots after the Cortex-M4 has set C2BOOT in PWR_CR4.*
- *It is possible to connect to each core separately (without shared mode) , but the C2BOOT bit must be set before connecting to the Cortex-M0+.*

5 Debug and programming use cases with EWARM and MDK-ARM

This section details a specific use case by describing the support of the STM32WL5x/Ex devices within EWARM and MDK-ARM.

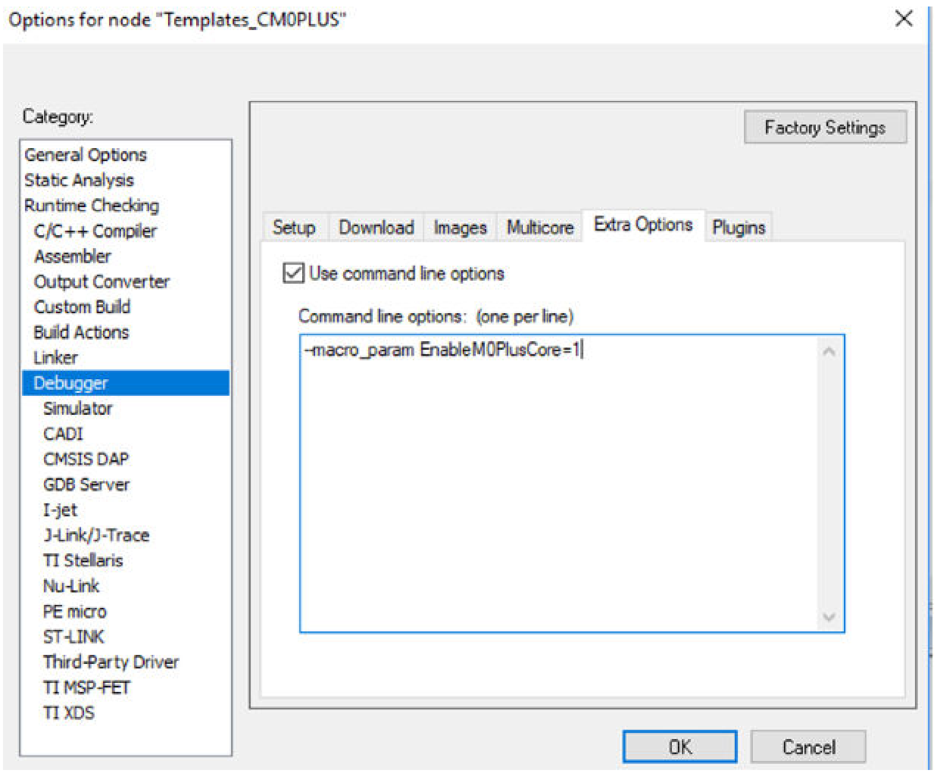
5.1 How to connect and download a Cortex-M0+ application while the full Flash memory is empty (EWARM)

Important: *Before connecting to the Cortex-M0+ core, ensure that the C2BOOT bit in PWR_CR4 is enabled.*

With the following steps, a Cortex-M0+ application can be downloaded while the full Flash memory is empty.

1. The C2BOOT must be automatically set before the AP1 connection for the download step, by adding the extra option `--macro_param EnableM0PlusCore =1` from *Project option -> Debugger -> Extra options*.

Figure 29. Enable Cortex-M0+ core



☑ Use command line options

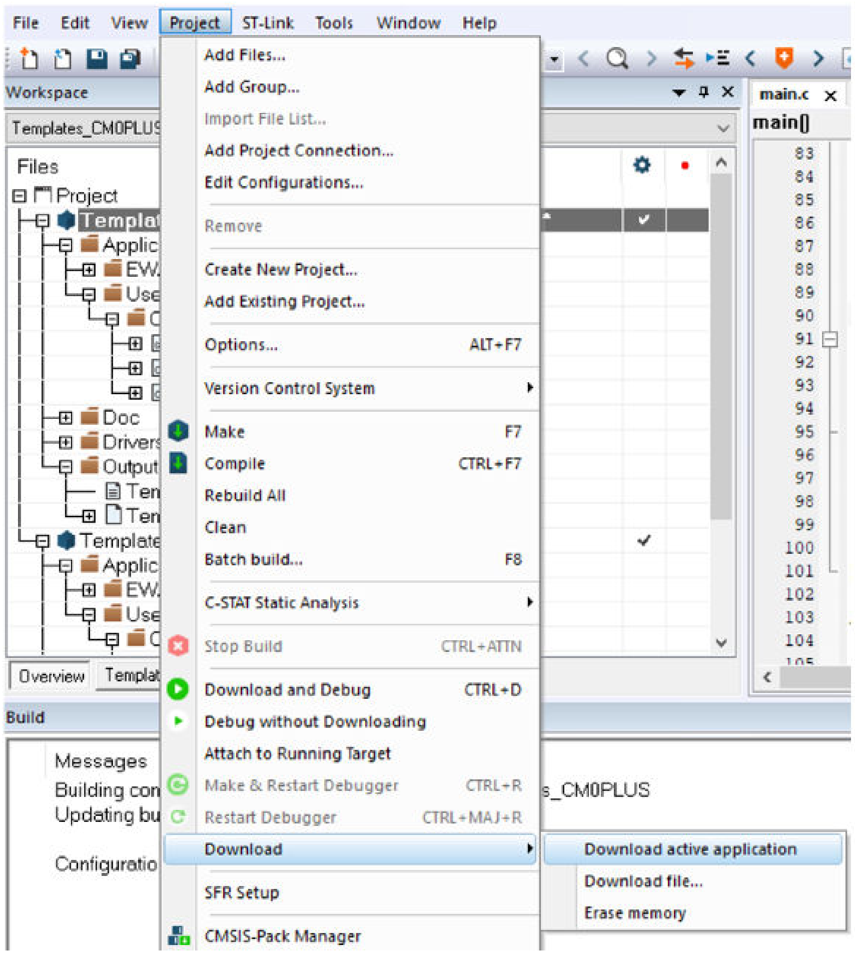
Command line options: (one per line)

```
-macro_param EnableM0PlusCore=1
```

2. Go to the *Project -> Option -> ST-LINK -> Setup* and select the reset mode as system reset.
3. Build the application and go to *Project -> Download -> Download active application* to program the Cortex-M0+ application within the Flash memory at 0x0802 0000.

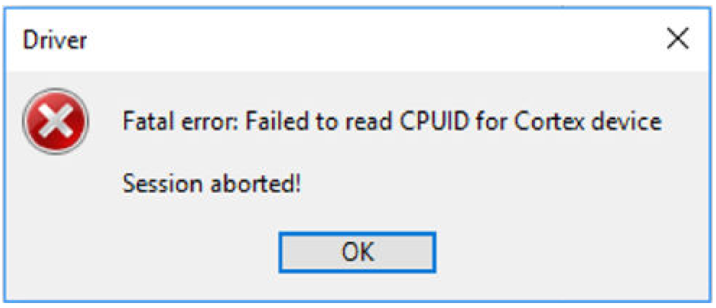
4. Download the active application to the target without launching a full debug session by using *Project -> Download -> Download active application*.

Figure 30. **Download active application**



The following error message indicates that the C2BOOT is not set. The IDE tries to connect with several attempts if necessary, until a timeout of 1 s (to wait eventual C2BOOT enabling from the application), else the IDE returns an error to indicate that the AP1 is not accessible.

Figure 31. **Error - AP1 not accessible**


Driver

Fatal error: Failed to read CPUID for Cortex device

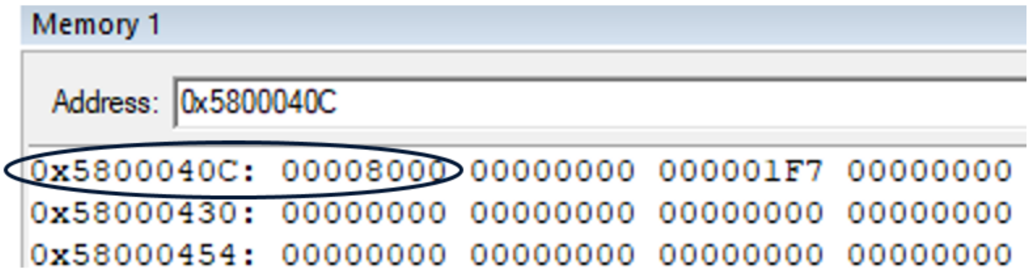
Session aborted!

5.2 How to connect and download a Cortex-M0+ application while the full Flash memory is empty (MDK-ARM)

With the following steps, a Cortex-M0+ application can be downloaded while the full Flash memory is empty.

1. The C2BOOT must be automatically set before the AP1 connection for the download step.
   - Connect to AP0 using MDK-ARM (see Section 4.1.1).
   - Open the *Memory* window from *View > Memory views*.
   - Write 0x0000 8000 in PWR_CR4 (at the address 0x5800 040C).

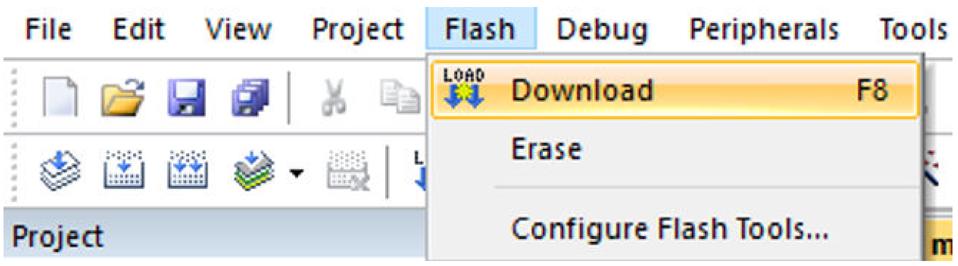
Figure 32. **Enable CPU2 using MDK-ARM**



```
Address: 0x5800040C
0x5800040C: 00008000 00000000 000001F7 00000000
0x58000430: 00000000 00000000 00000000 00000000
0x58000454: 00000000 00000000 00000000 00000000
```

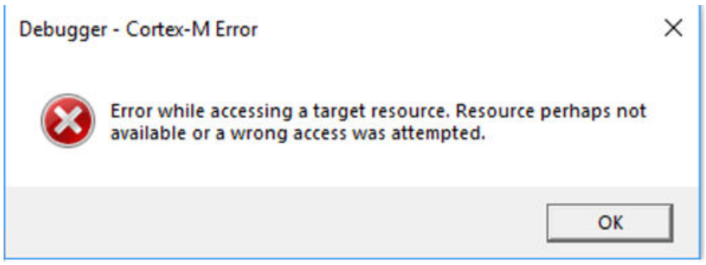
2. Go to the *Debug* tab and check that the debugger is connected like in Figure 24.
3. Build the application and go to *Flash -> Download* to program the active application to the target without launching a full debug session.

Figure 33. **Download option**



When trying to do the same manipulation before setting C2BOOT, an error message appears that indicates that the AP1 is not accessible.

Figure 34. **AP1 not accessible**


Debugger - Cortex-M Error

Error while accessing a target resource. Resource perhaps not available or a wrong access was attempted.

OK

6 Secure programming

This section describes the steps needed to enable the security in order to protect the memory areas (Flash memory, SRAM1 and SRAM2) from being accessed by any non-authorized bus master.

SRAM1 and SRAM2 areas are only secure when the Flash memory security is enabled (ESE = 1 and FSD = 0).

Secure Flash memory

The secure Flash memory area:

- starts at Flash memory base address + (SFSA[6:0] x 0x0800) included
  where SFSA[6:0] is the secure Flash memory start address and contains the start address of the first 2-Kbyte page of the secure Flash memory area.
- ends at Flash memory last address

Note: When the CPU2 security is enabled, the minimum CPU2 secure area size is one sector (2 Kbytes).

For example, with a CPU2 secure area from 0x0802 7000 (included) to 0x0803 FFFF (included), FLASH_SFR must be programmed with SFSA = 0x4E.

The ESE flag in FLASH_OPTR indicates if the CPU2 security is enabled. Any CPU1 access to a CPU2 security area triggers RDERR or WRPERR flag error.

Secure non-backup SRAM1

The secure non-backup SRAM1 area:

- starts at non-backup SRAM1 base address + (SNBRSA[4:0] x 0x0400) included
  where SNBRSA[4:0] is the secure non-backup SRAM1 start address and contains the start address of the first 1-Kbyte page of the secure non-backup SRAM1 area.
- ends at non-backup SRAM1 last address

For example, with a CPU2 secure SRAM1 area from 0x2000 6C00 (included) to 0x2000 7FFF (included), FLASH_SRRVR must be programmed with SNBRSA = 0x1B.

Any CPU1 read access returns zero data. A write access to a CPU2 secure SRAM1 area is discarded and generates an illegal access event.

Note: When NBRSD is set to 1, the SRAM1 is non-secure.

Secure SRAM2

The secure SRAM2 area:

- starts at SRAM2 base address + (SBSRA[4:0] x 0x0400) included
  where SBSRA[4:0] is the secure SRAM2 start address and contains the start address of the first 1-Kbyte page of the secure SRAM2 area.
- ends at SRAM2 last address

For example, with a CPU2 secure SRAM2 area from 0x2000 A800 (included) to 0x2000 FFFF (included), FLASH_SRRVR must be programmed with SBSRA = 0x0A.

Any CPU1 read access returns zero data. A write access to a CPU2 secure SRAM2 area is discarded and generates an illegal access event.

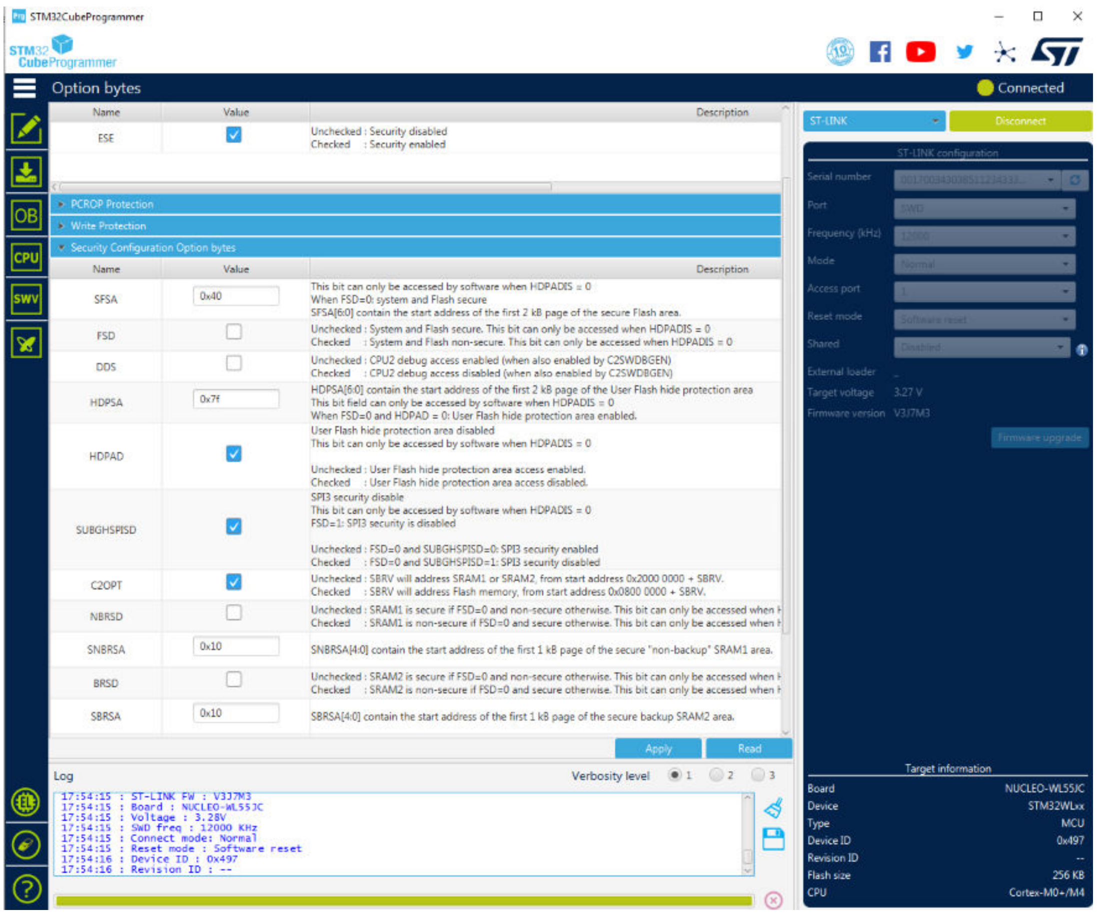
Note: When BRSD is set to 1, the SRAM2 is non-secure.

Option-byte setup

Before securing the system and memories as detailed in the next sections, the option bytes must be configured as follows, using the STM32CubeProgrammer tool:

- FSD = 0 to enable the overall system security
- SFSA[6:0] = 0x40 in FLASH_SFR to secure the second half of the Flash memory area
- SNBRSA[4:0] = 0x10 and NBRSD = 0 in FLASH_SRRVR to secure the second half of the non-backup SRAM1
- SBRSA[4:0] = 0x10 and BRSD = 0 in FLASH_SRRVR to secure the second half of the backup SRAM2

Figure 35. Option-byte configuration



Name	Value	Description
ESE	✓	Unchecked : Security disabled Checked : Security enabled

Name	Value	Description
SFSA	0x40	This bit can only be accessed by software when HDPADIS = 0 When FSD=0: system and Flash secure SFSA[6:0] contain the start address of the first 2 kB page of the secure Flash area.
FSD	☐	Unchecked : System and Flash secure. This bit can only be accessed when HDPADIS = 0 Checked : System and Flash non-secure. This bit can only be accessed when HDPADIS = 0
DDS	☐	Unchecked : CPU2 debug access enabled (when also enabled by C2SWDBGEN) Checked : CPU2 debug access disabled (when also enabled by C2SWDBGEN)
HDPSA	0x7f	HDPSA[6:0] contain the start address of the first 2 kB page of the User Flash hide protection area This bit field can only be accessed by software when HDPADIS = 0 When FSD=0 and HDPAD = 0: User Flash hide protection area enabled.
HDPAD	✓	User Flash hide protection area disabled This bit can only be accessed by software when HDPADIS = 0 Unchecked : User Flash hide protection area access enabled. Checked : User Flash hide protection area access disabled.
SUBGHSPISD	✓	SPI3 security disable This bit can only be accessed by software when HDPADIS = 0 FSD=1: SPI3 security is disabled Unchecked : FSD=0 and SUBGHSPISD=0: SPI3 security enabled Checked : FSD=0 and SUBGHSPISD=1: SPI3 security disabled
C2OPT	✓	Unchecked : SBRV will address SRAM1 or SRAM2, from start address 0x2000 0000 + SBRV. Checked : SBRV will address Flash memory, from start address 0x0800 0000 + SBRV.
NBRSD	☐	Unchecked : SRAM1 is secure if FSD=0 and non-secure otherwise. This bit can only be accessed when H Checked : SRAM1 is non-secure if FSD=0 and secure otherwise. This bit can only be accessed when H
SNBRSA	0x10	SNBRSA[4:0] contain the start address of the first 1 kB page of the secure "non-backup" SRAM1 area.
BRSD	☐	Unchecked : SRAM2 is secure if FSD=0 and non-secure otherwise. This bit can only be accessed when H Checked : SRAM2 is non-secure if FSD=0 and secure otherwise. This bit can only be accessed when H
SBRSA	0x10	SBRSA[4:0] contain the start address of the first 1 kB page of the secure backup SRAM2 area.

Log   Verbosity level [✓ 1] 2 3

```
17:54:15 : ST-LINK FW : V3J7M3
17:54:15 : Board : NUCLEO-WL55JC
17:54:15 : Voltage : 3.28V
17:54:15 : SWD freq : 12000 KHz
17:54:15 : Connect mode: Normal
17:54:15 : Reset mode : Software reset
17:54:16 : Device ID : 0x497
17:54:16 : Revision ID : --
```

ST-LINK configuration

Serial number	001700343038511234333...
Port	SWD
Frequency (kHz)	12000
Mode	Normal
Access port	1
Reset mode	Software reset
Shared	Disabled
External loader	_
Target voltage	3.27 V
Firmware version	V3J7M3

Target information

Board	NUCLEO-WL55JC
Device	STM32WLxx
Type	MCU
Device ID	0x497
Revision ID	--
Flash size	256 KB
CPU	Cortex-M0+/M4

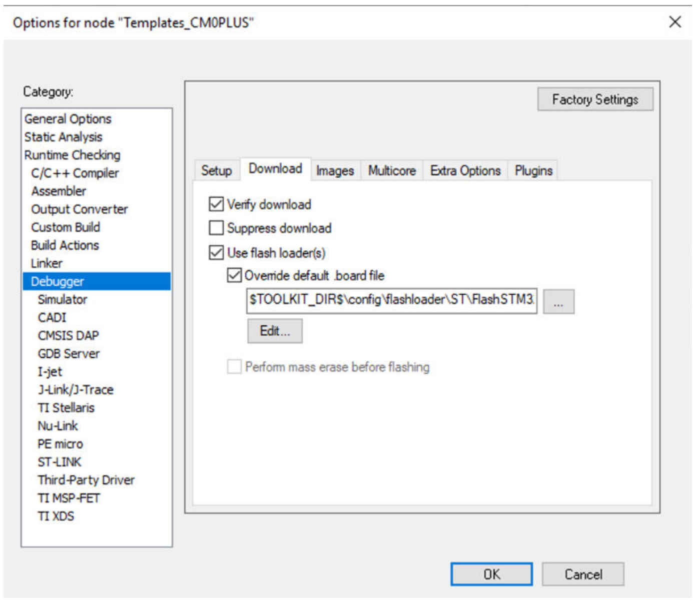
6.1 Secure programming using EWARM

The steps below are needed to perform a secure programming using the NUCLEO_WL55JC board and EWARM:

1. Ensure that all option bytes are configured as in the introduction of Section 6: Secure programming.
2. Open the CM4 project and ensure that all project options are configured as described in Section 3.1.1.
3. Open the CM0+ project and ensure that all project options are configured as described in Section 3.1.2.

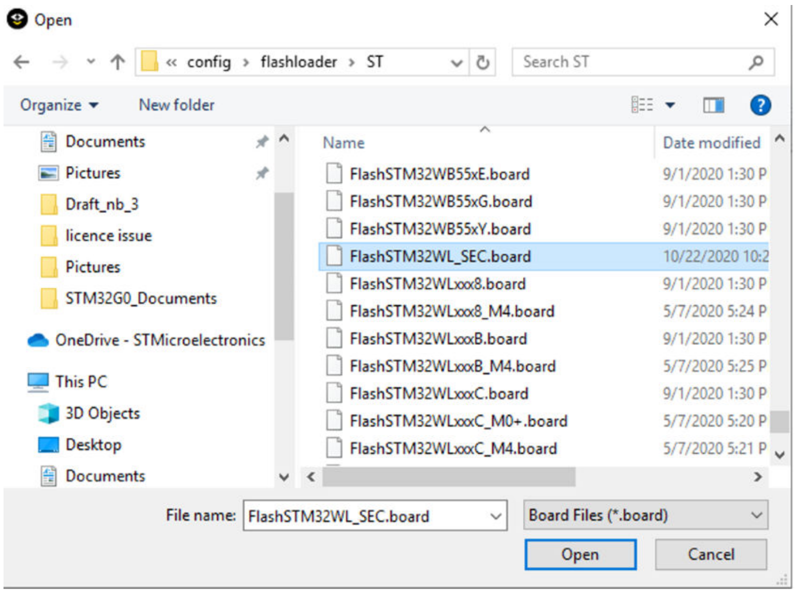
4. Go to *Project options -> Debugger -> Download* tab from the CM0+ project and replace the flashloader selected by *FlashSTM32WL_SEC.board*, that is available under `C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.4_3\arm\config\flashloader\ST`.

Figure 36. Change the default flashloader



☑ Verify download
☐ Suppress download
☑ Use flash loader(s)
☑ Override default .board file
$TOOLKIT_DIR$\config\flashloader\ST\FlashSTM3
☐ Perform mass erase before flashing

Figure 37. Select the secure flashloader



Name	Date modified
FlashSTM32WB55xE.board	9/1/2020 1:30 P
FlashSTM32WB55xG.board	9/1/2020 1:30 P
FlashSTM32WB55xY.board	9/1/2020 1:30 P
FlashSTM32WL_SEC.board	10/22/2020 10:2
FlashSTM32WLxxx8.board	9/1/2020 1:30 P
FlashSTM32WLxxx8_M4.board	5/7/2020 5:24 P
FlashSTM32WLxxxB.board	9/1/2020 1:30 P
FlashSTM32WLxxxB_M4.board	5/7/2020 5:25 P
FlashSTM32WLxxxC.board	9/1/2020 1:30 P
FlashSTM32WLxxxC_M0+.board	5/7/2020 5:20 P
FlashSTM32WLxxxC_M4.board	5/7/2020 5:21 P

File name: FlashSTM32WL_SEC.board
Board Files (*.board)

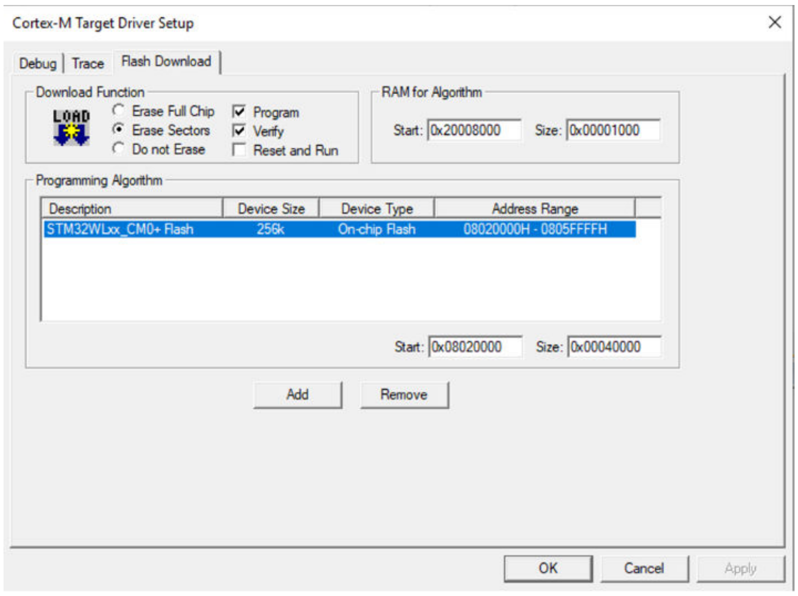
5. Build and download the Cortex-M4 application.
6. Set the C2BOOT to perform CPU2 (Cortex-M0+) boot in PWR_CR4.
7. Build and download the Cortex-M0+ application: the Flash memory programming is performed using the JTAG interface.

6.2 Secure programming using MDK-ARM

The steps below are needed to perform a secure programming using the NUCLEO_WL55JC board and MDK-ARM:

1. Ensure that all option bytes are configured as in the introduction of Section 6.
2. Open the CM4 project and ensure that all project options are configured as described in Section 4.1.1.

3. Open the CM0+ project and ensure that all project options are configured as described in Section 4.1.2.
4. Go to *Project options -> Debugger > Flash Download* tab and update the flashloader execution address in *RAM for Algorithm* with a secure SRAM address.


Cortex-M Target Driver Setup

Debug | Trace | Flash Download

Download Function: ( ) Erase Full Chip, (●) Erase Sectors, ( ) Do not Erase; [✓] Program, [✓] Verify, [ ] Reset and Run

RAM for Algorithm: Start: 0x20008000 Size: 0x00001000

Programming Algorithm

Description	Device Size	Device Type	Address Range
STM32WLxx_CM0+ Flash	256k	On-chip Flash	08020000H - 0805FFFFH

Start: 0x08020000 Size: 0x00040000

Figure 38. **Change the flashloader execution address**

5. Build and download the Cortex-M4 application.
6. Set the C2BOOT to perform CPU2 (Cortex-M0+) boot in PWR_CR4.
7. Build and download the Cortex-M0+ application: the Flash memory programming is performed using the JTAG interface.

Revision history

Table 2. **Document revision history**

Date	Version	Changes
2-Dec-2020	1	Initial release.
19-Feb-2026	2	Title updated.

Contents

List of tables

List of figures