life.augmented

STM32H7R-S Debug authentication

Introduction

- STM32H7R-S implements a new mechanism for the protection of the code stored in internal flash.
- Let’s see the impact of this change

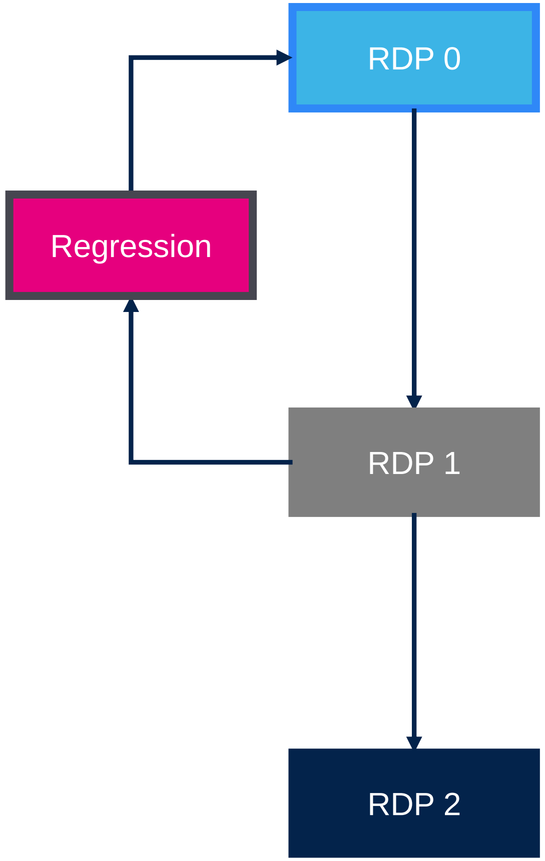
life.augmented

Reminder of RDP protection on legacy STM32

- RDP 0 : Open state dedicated to development

- RDP 1
  - Firmware in flash is protected from readout
  - Debugger can attach and read ram content
  - Possible regression to RDP0 with automatic flash erase
  - State mostly used because of this regression capability

- RDP2
  - No debugger access, no possible regression



STM32H7R-S evolution

- RDP levels replaced by PRODUCT_STATE
  - Option byte in both cases
  - RDP values fixed for RDP 0 (0xAA) and RDP 2 (0xCC). All other values mean RDP 1
  - PRODUCT_STATE have fixed value for each state. No default

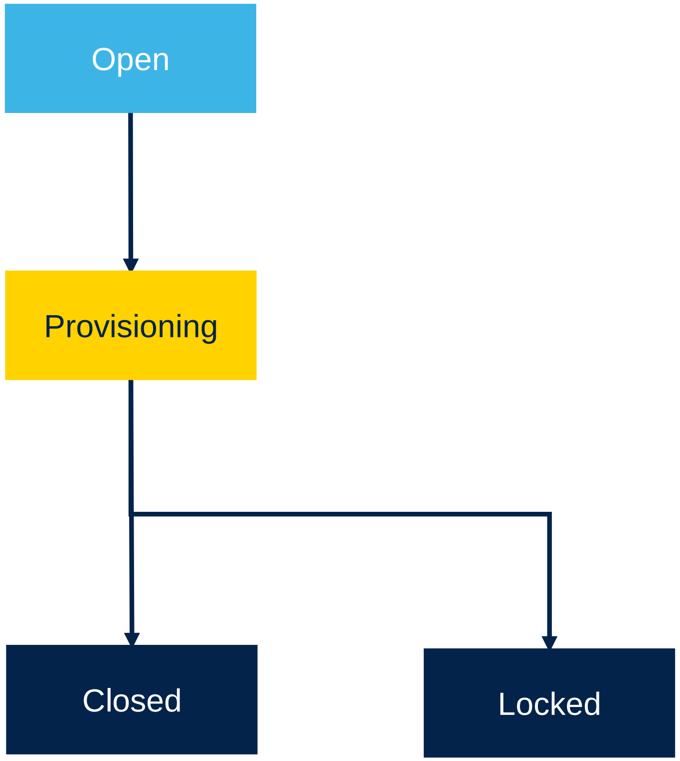
- Debug Authentication to control device regression and/or debugging link reopening
  - JTAG dedicated access point
  - ADAC protocol defined by ARM
  - 2 possible methods :
    - Password used for regression (not covered here)
    - Certificate used for regression and debugging capabilities

Product State

- Open state dedicated to development

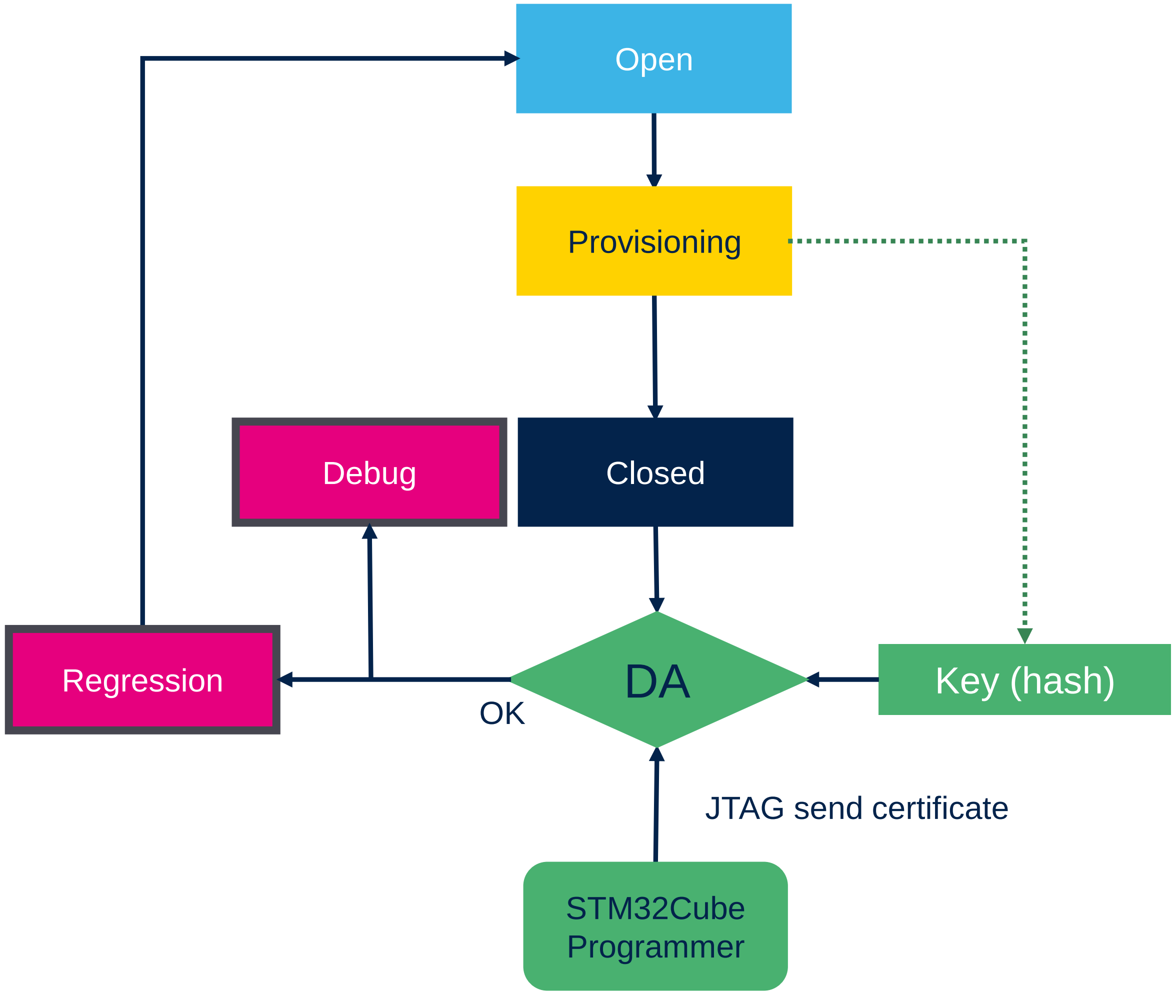
- Provisioning allows transmitting specific file containing keys and data to be provisioned

- Closed and Locked are used in the field to protect device.
  - Closed state allows debug authentication
  - Locked state is definitive



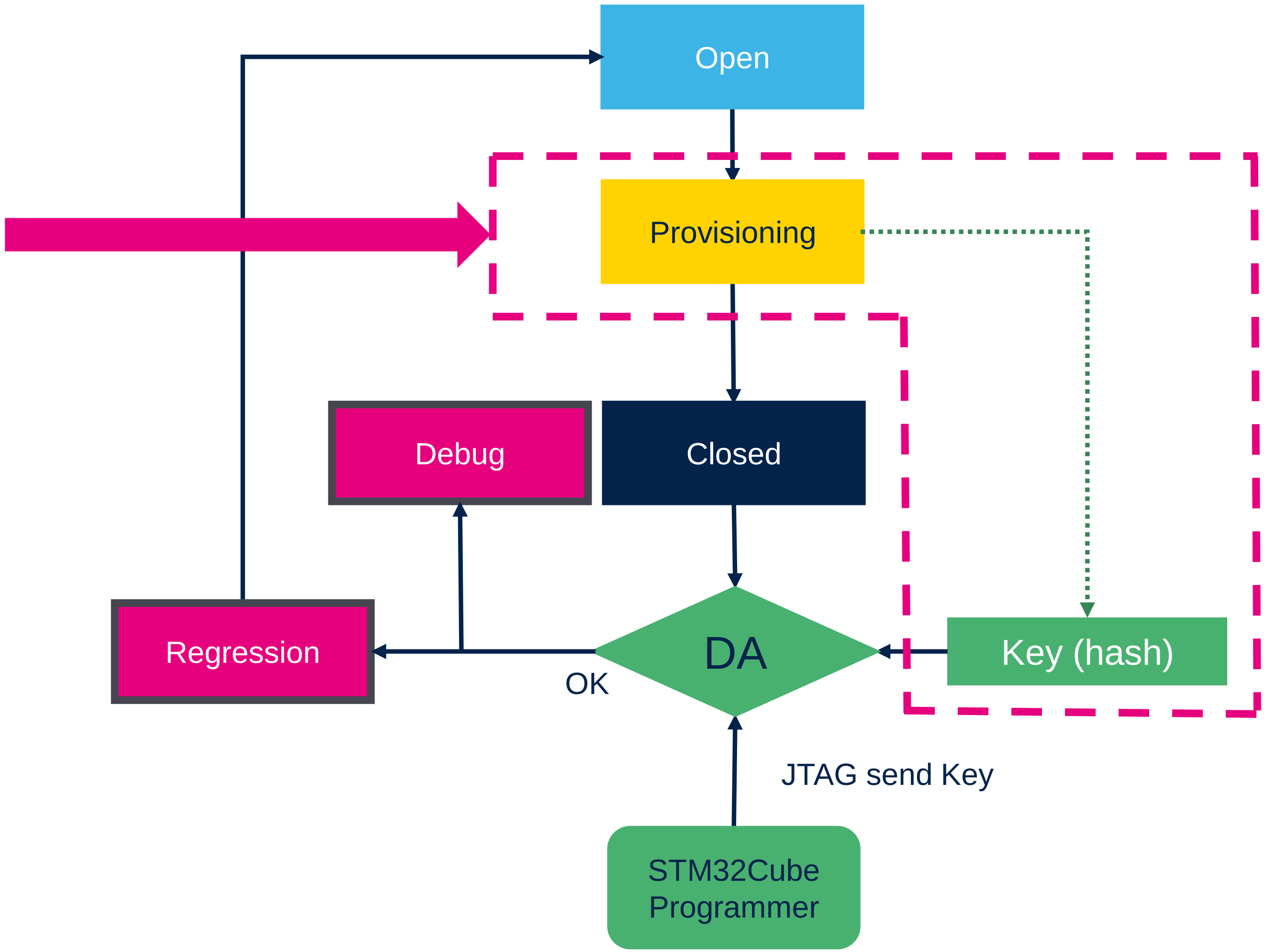
Debug Authentication

- Firmware can be flashed in open state
- Provisioning is used in production to transmit auth key to the secure storage
- Close device : no more debug access
- Field return : use certificate to open the device securely through JTAG/SWD interface using dedicated access point.



Impact on production process

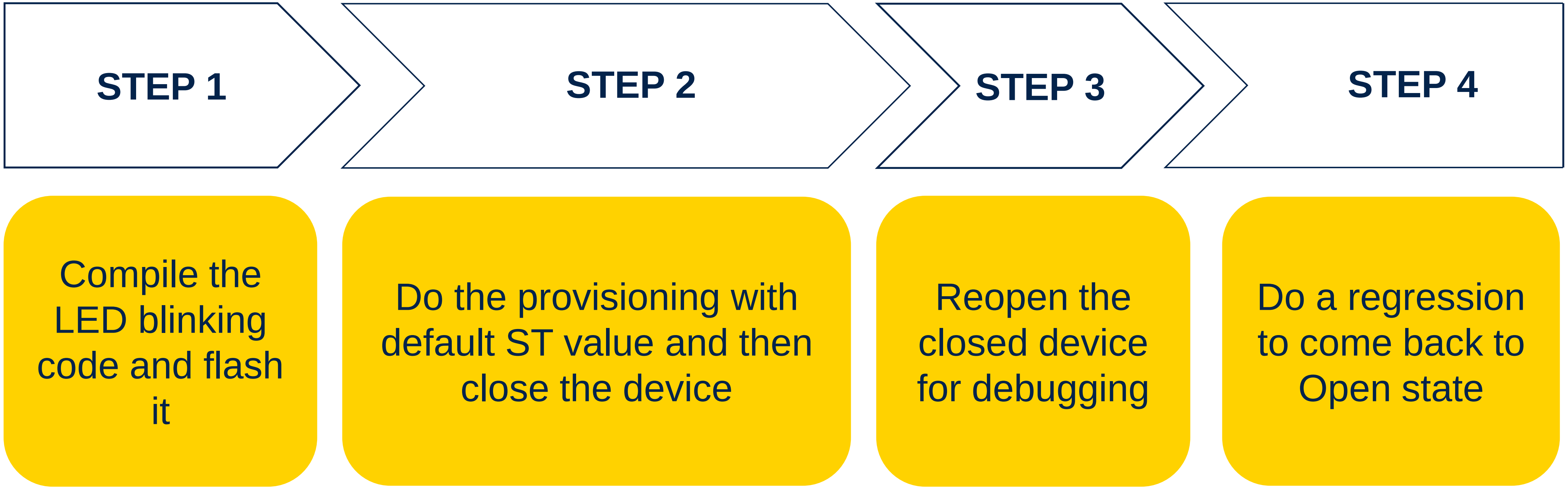
New step required to have the capability to perform regression or to reopen the debug



Demo Hands-on purpose

- Simple LED blink application
- This demo shows the steps to provision key in order to enable the debug reopening and regression capabilities.
- Material is available on GitHub
  https://github.com/ST-TOMAS-Examples-Security/stm32h7rs_debug_authentication

Key Steps for the Hands-On Exercise



STEP 1

Compile the LED blinking code and flash it

STEP 2

Do the provisioning with default ST value and then close the device

STEP 3

Reopen the closed device for debugging

STEP 4

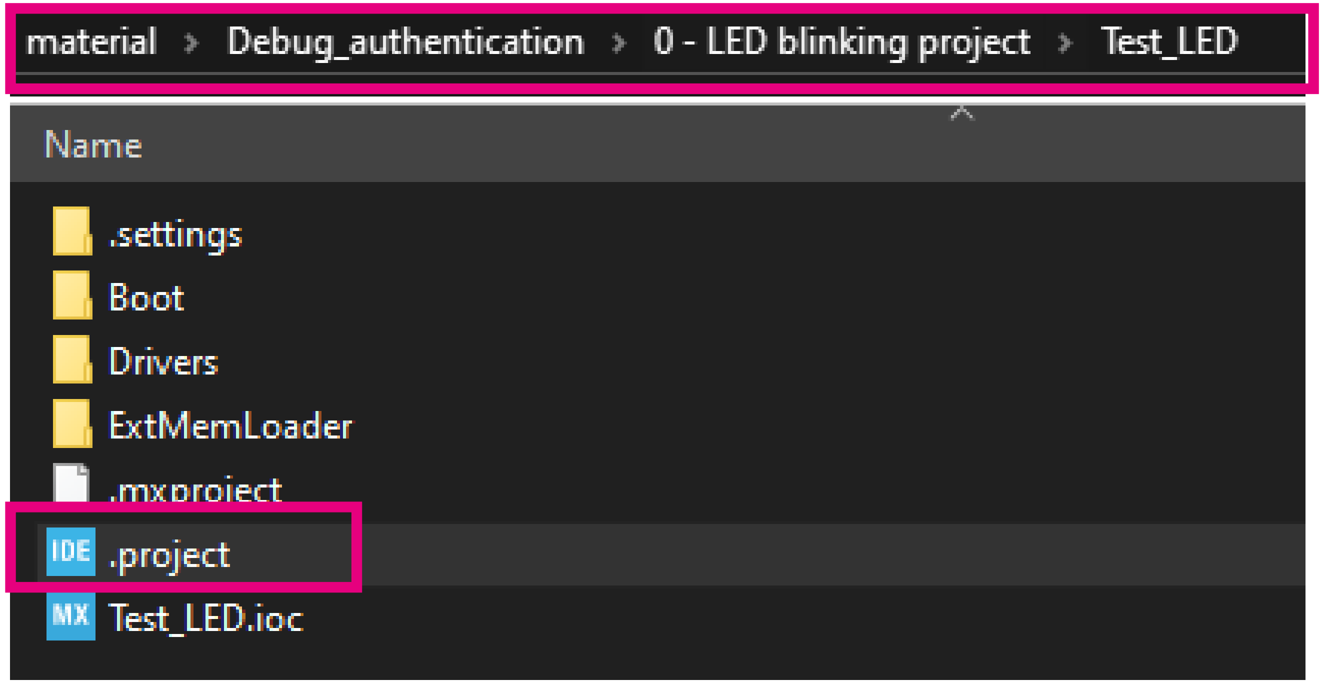
Do a regression to come back to Open state

Key Steps for the Hands-On Exercise

STEP 1

Compile the LED blinking code and flash it

- Open with CUBE IDE the project :
  0-LED blinking project\Test_LED\.project



material › Debug_authentication › 0 - LED blinking project › Test_LED

Name

.settings

Boot

Drivers

ExtMemLoader

.mxproject

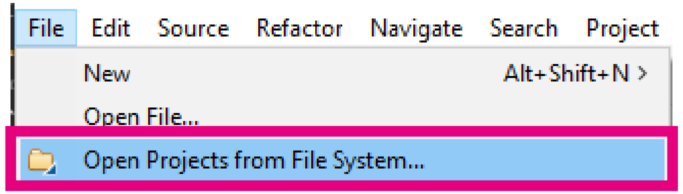
.project

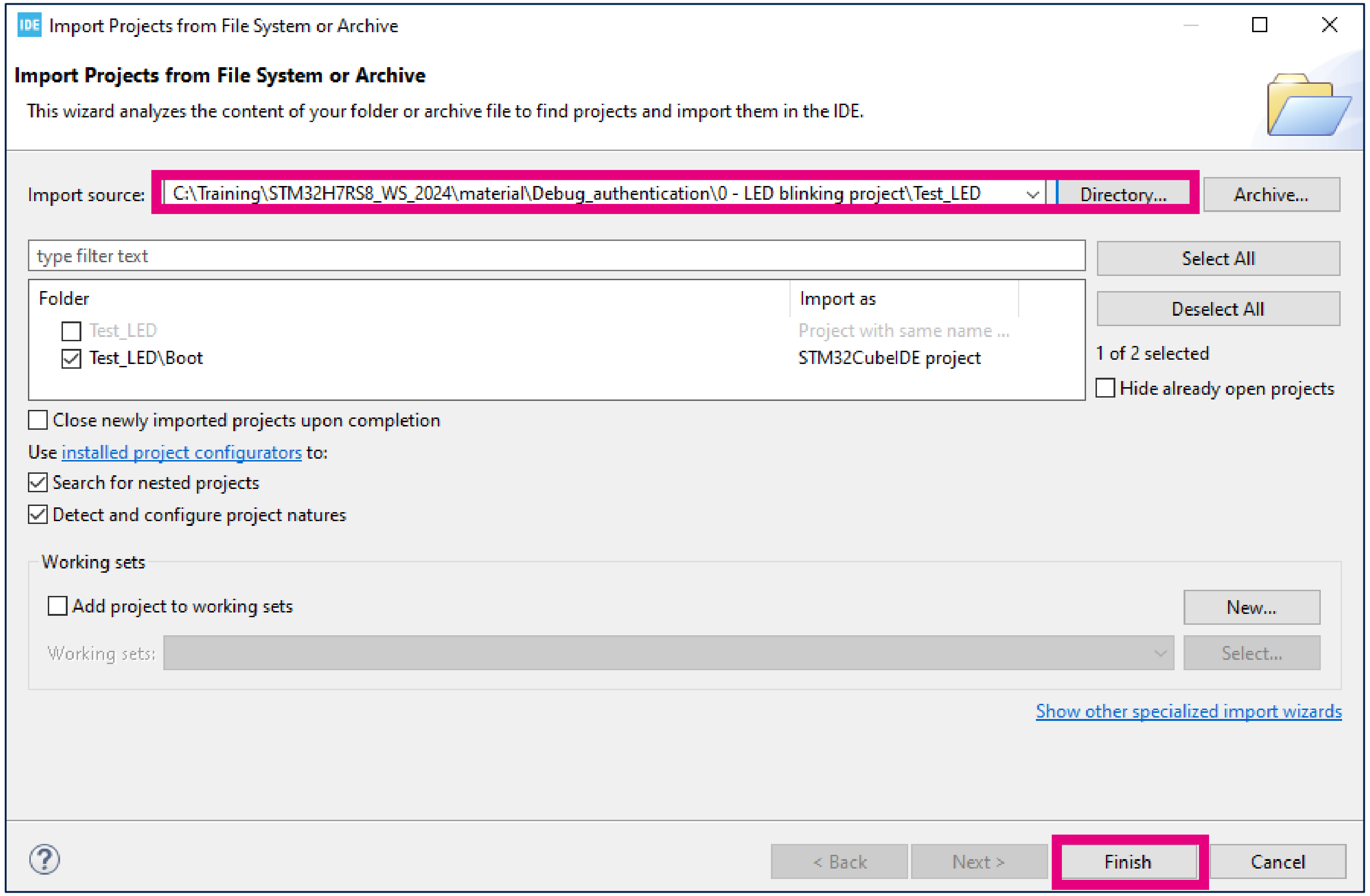
Test_LED.ioc

Key Steps for the Hands-On Exercise

STEP 1

Compile the LED blinking code and flash it





Import Projects from File System or Archive

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

Import source: C:\Training\STM32H7RS8_WS_2024\material\Debug_authentication\0 - LED blinking project\Test_LED

type filter text

Folder	Import as
□ Test_LED	Project with same name ...
☑ Test_LED\Boot	STM32CubeIDE project

1 of 2 selected

□ Hide already open projects

□ Close newly imported projects upon completion

Use installed project configurators to:

☑ Search for nested projects

☑ Detect and configure project natures

Working sets

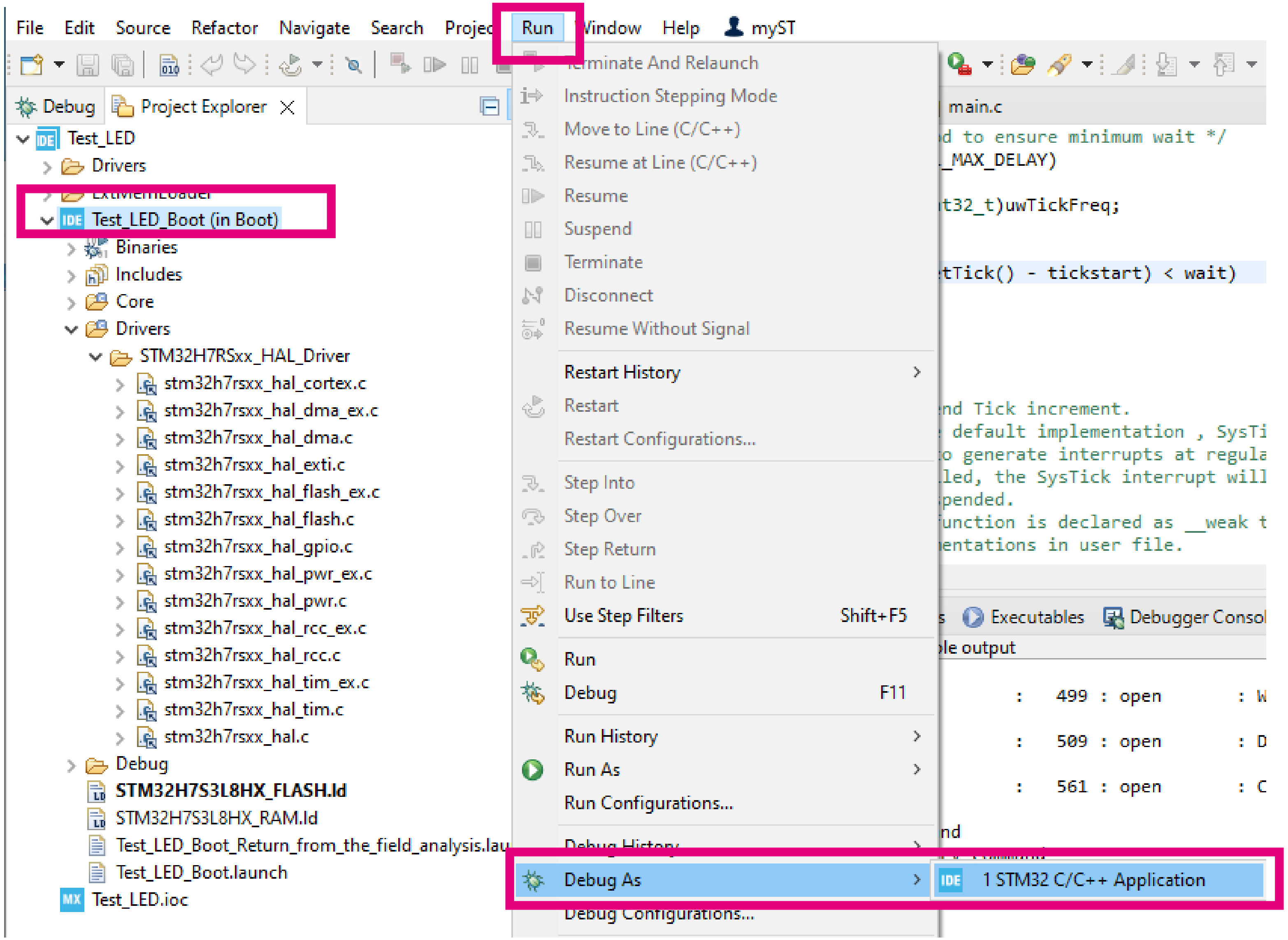
□ Add project to working sets

Working sets:

Key Steps for the Hands-On Exercise

STEP 1

Compile the LED blinking code and flash it



```
d to ensure minimum wait */
_MAX_DELAY)

t32_t)uwTickFreq;

tTick() - tickstart) < wait)

nd Tick increment.
 default implementation , SysTi
o generate interrupts at regula
led, the SysTick interrupt will
pended.
unction is declared as __weak t
entations in user file.
```

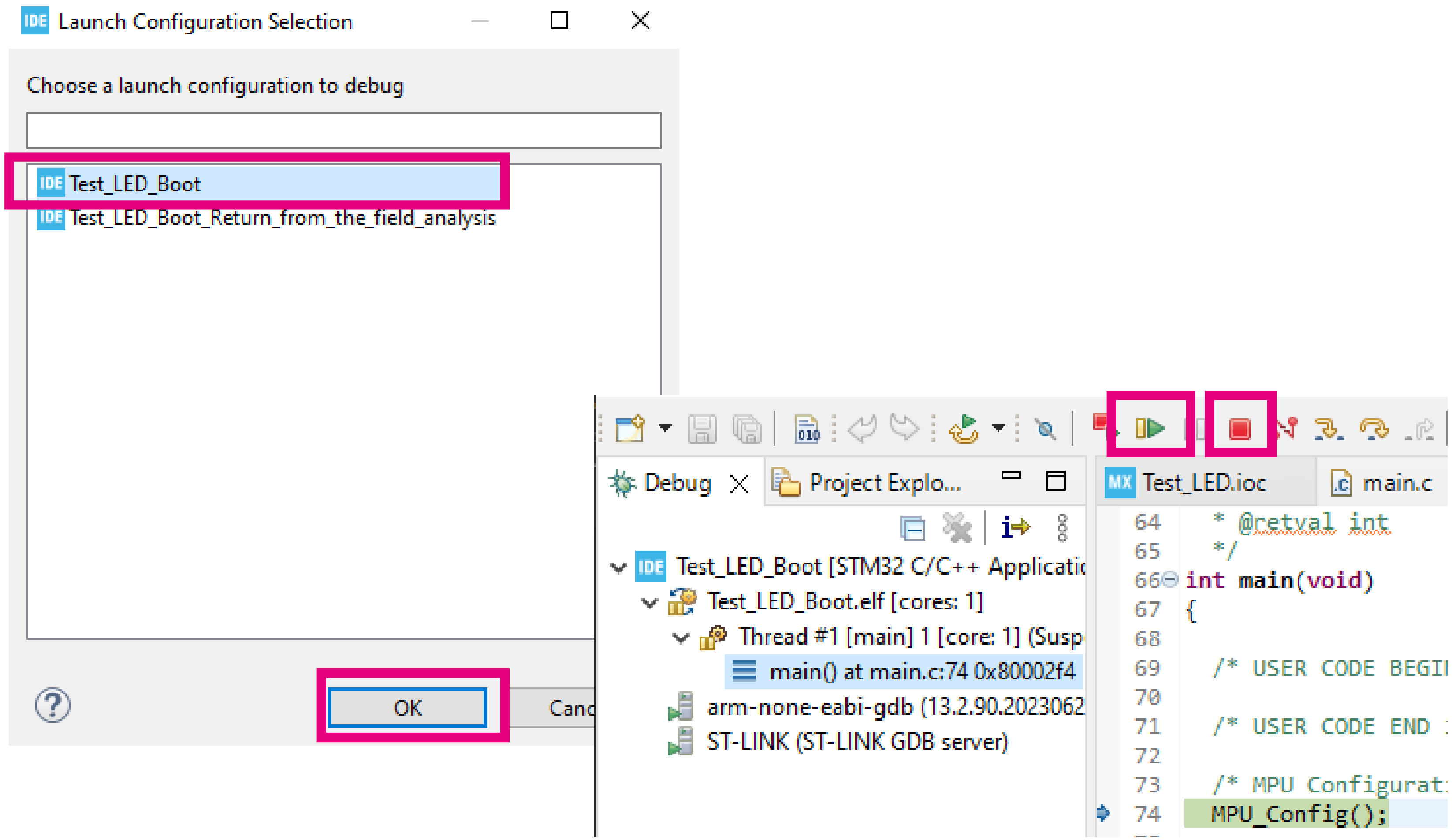
```
:   499 : open    : W
:   509 : open    : D
:   561 : open    : C
nd
```

life.augmented

Key Steps for the Hands-On Exercise

STEP 1

Compile the LED blinking code and flash it



Launch Configuration Selection

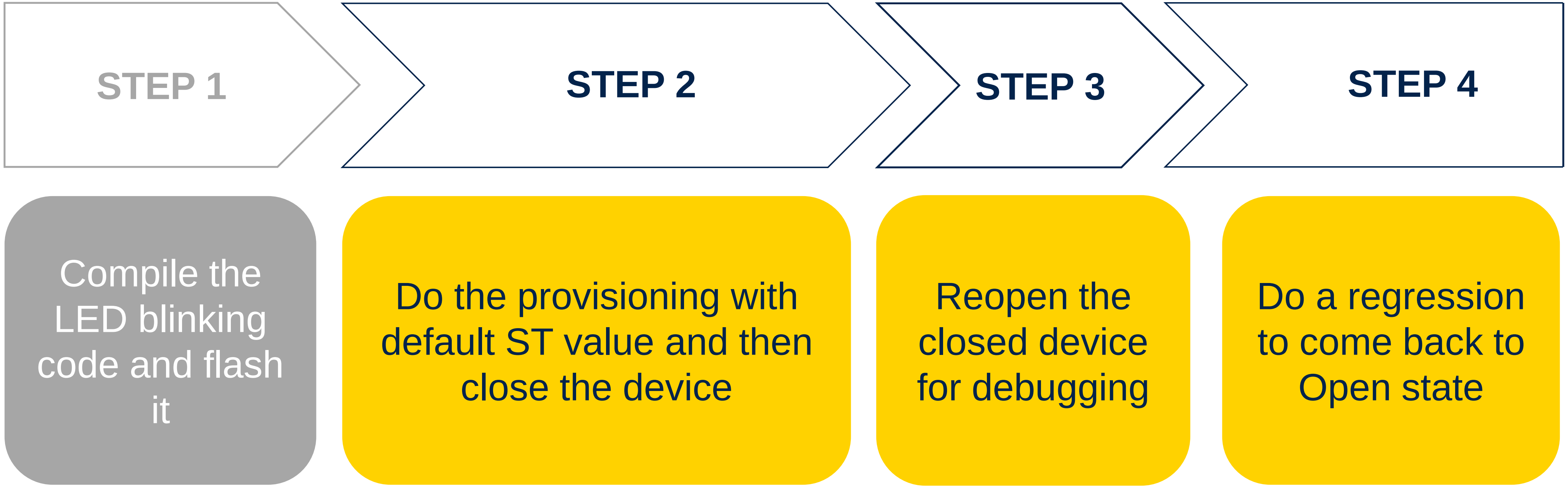
Choose a launch configuration to debug

- Test_LED_Boot
- Test_LED_Boot_Return_from_the_field_analysis

```
Test_LED_Boot [STM32 C/C++ Applicatio
  Test_LED_Boot.elf [cores: 1]
    Thread #1 [main] 1 [core: 1] (Susp
      main() at main.c:74 0x80002f4
  arm-none-eabi-gdb (13.2.90.2023062
  ST-LINK (ST-LINK GDB server)
```

```
  * @retval int
  */
int main(void)
{

  /* USER CODE BEGI

  /* USER CODE END :

  /* MPU Configurat:
  MPU_Config();
```

Key Steps for the Hands-On Exercise



STEP 1

Compile the LED blinking code and flash it

STEP 2

Do the provisioning with default ST value and then close the device

STEP 3

Reopen the closed device for debugging

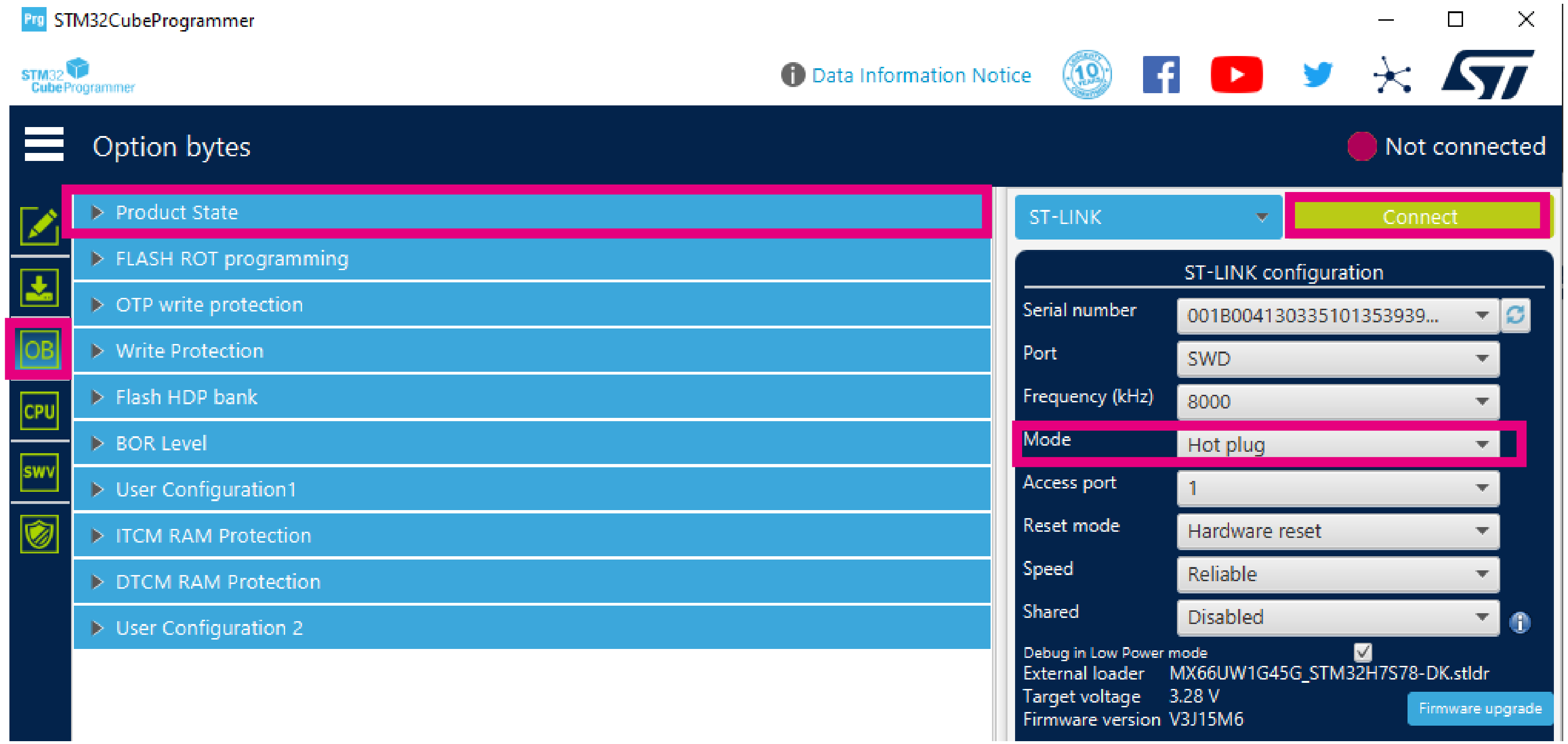
STEP 4

Do a regression to come back to Open state

Key Steps for the Hands-On Exercise

STEP 2.1

Do the provisioning with default ST value



STM32CubeProgrammer

Option bytes

Not connected

- Product State
- FLASH ROT programming
- OTP write protection
- Write Protection
- Flash HDP bank
- BOR Level
- User Configuration1
- ITCM RAM Protection
- DTCM RAM Protection
- User Configuration 2

ST-LINK | Connect

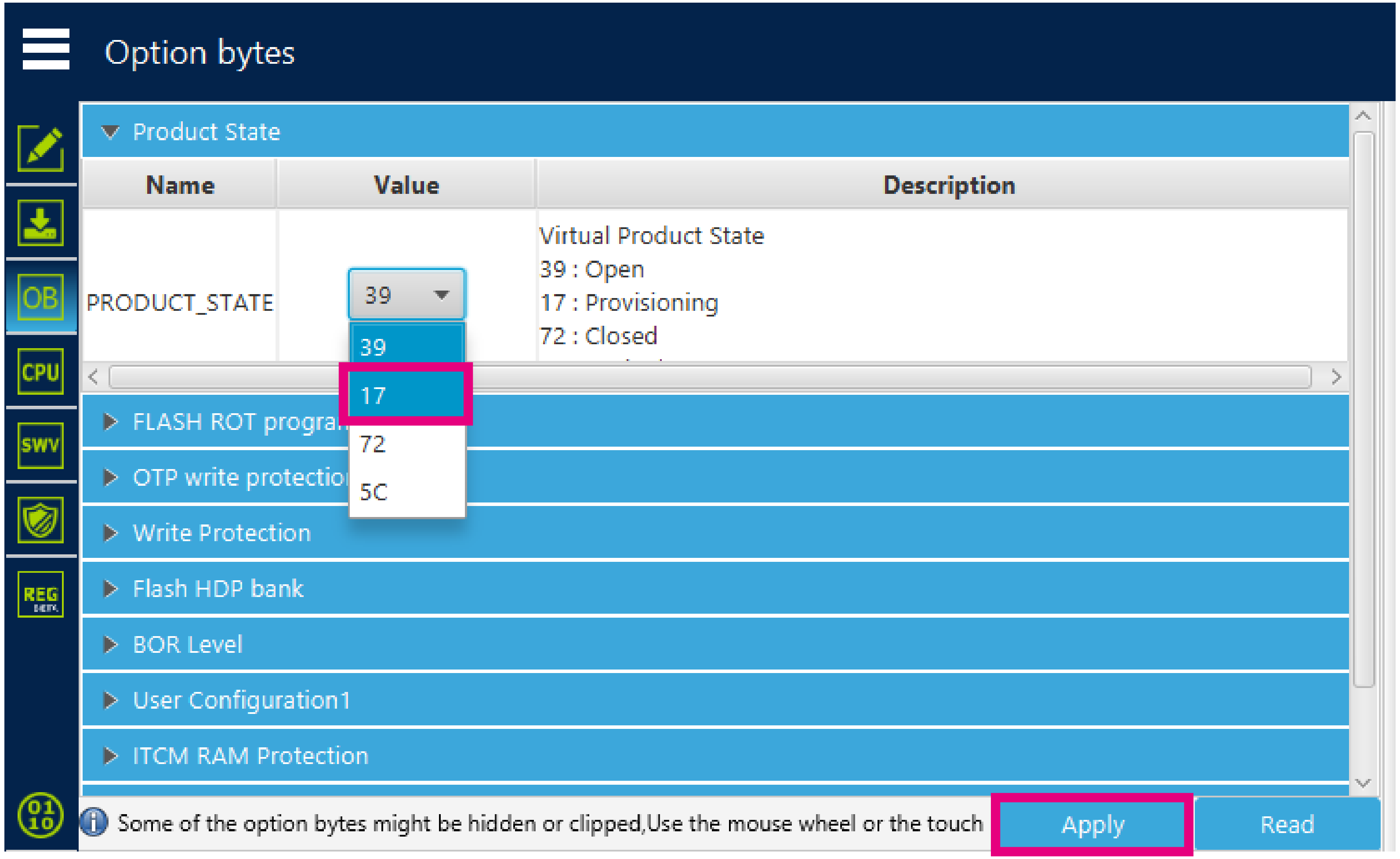
ST-LINK configuration

Serial number: 001B004130335101353939...
Port: SWD
Frequency (kHz): 8000
Mode: Hot plug
Access port: 1
Reset mode: Hardware reset
Speed: Reliable
Shared: Disabled
Debug in Low Power mode ☑
External loader MX66UW1G45G_STM32H7S78-DK.stldr
Target voltage 3.28 V
Firmware version V3J15M6

Key Steps for the Hands-On Exercise

STEP 2.1

Do the provisioning with default ST value



Option bytes

Product State

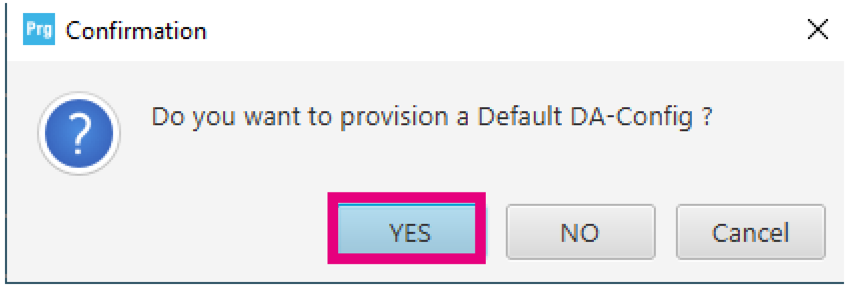
Name	Value	Description
PRODUCT_STATE	39	Virtual Product State 39 : Open 17 : Provisioning 72 : Closed

Some of the option bytes might be hidden or clipped,Use the mouse wheel or the touch

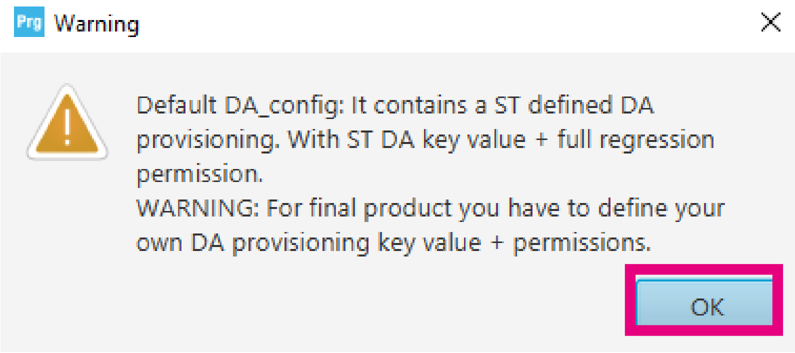
Key Steps for the Hands-On Exercise

STEP 2.1

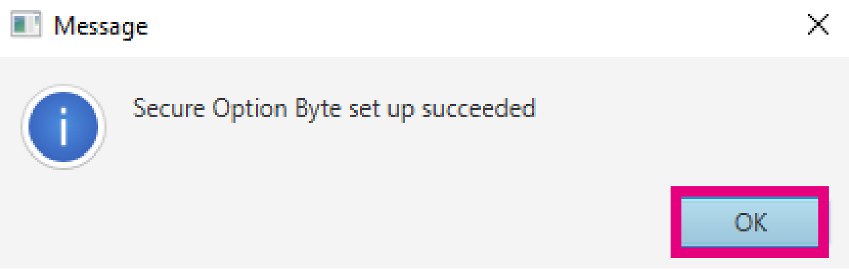
Do the provisioning with default ST value and then close the device


Confirmation

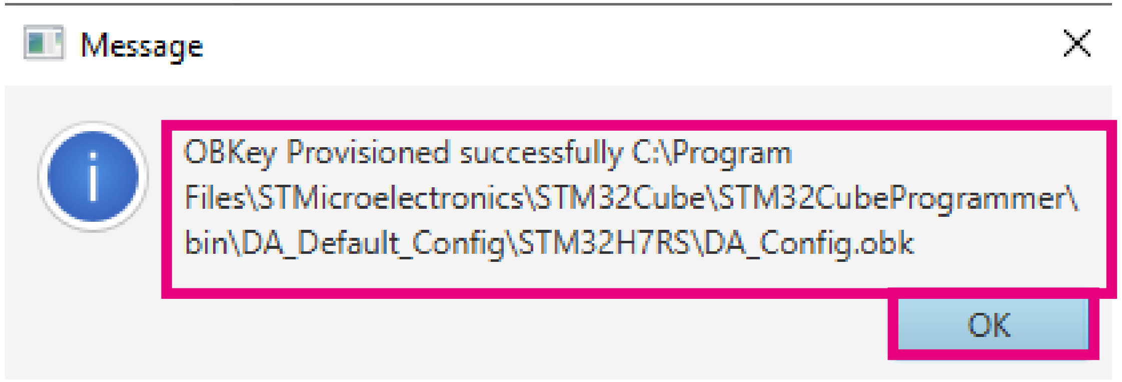
Do you want to provision a Default DA-Config ?


Warning

Default DA_config: It contains a ST defined DA provisioning. With ST DA key value + full regression permission.
WARNING: For final product you have to define your own DA provisioning key value + permissions.

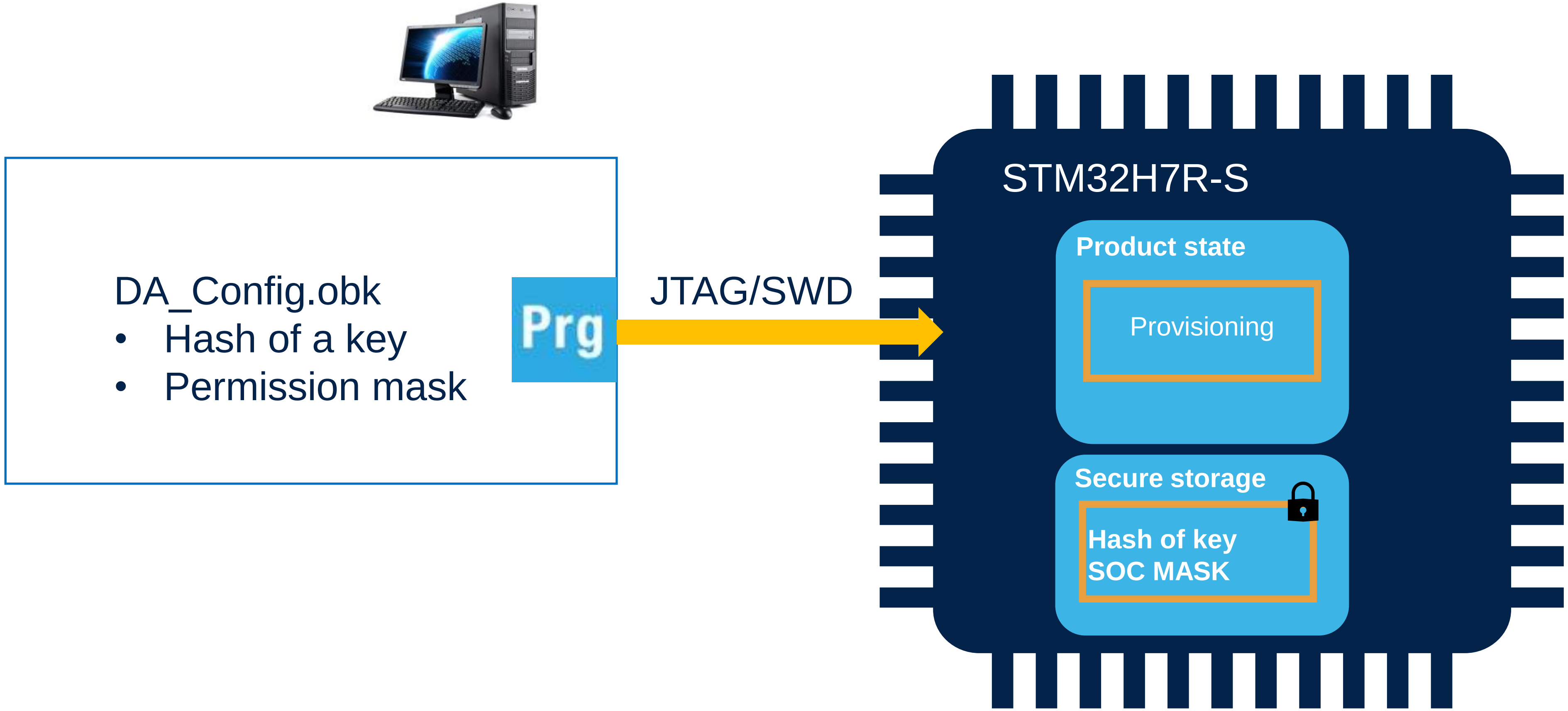

Message

Secure Option Byte set up succeeded


Message

OBKey Provisioned successfully C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\DA_Default_Config\STM32H7RS\DA_Config.obk

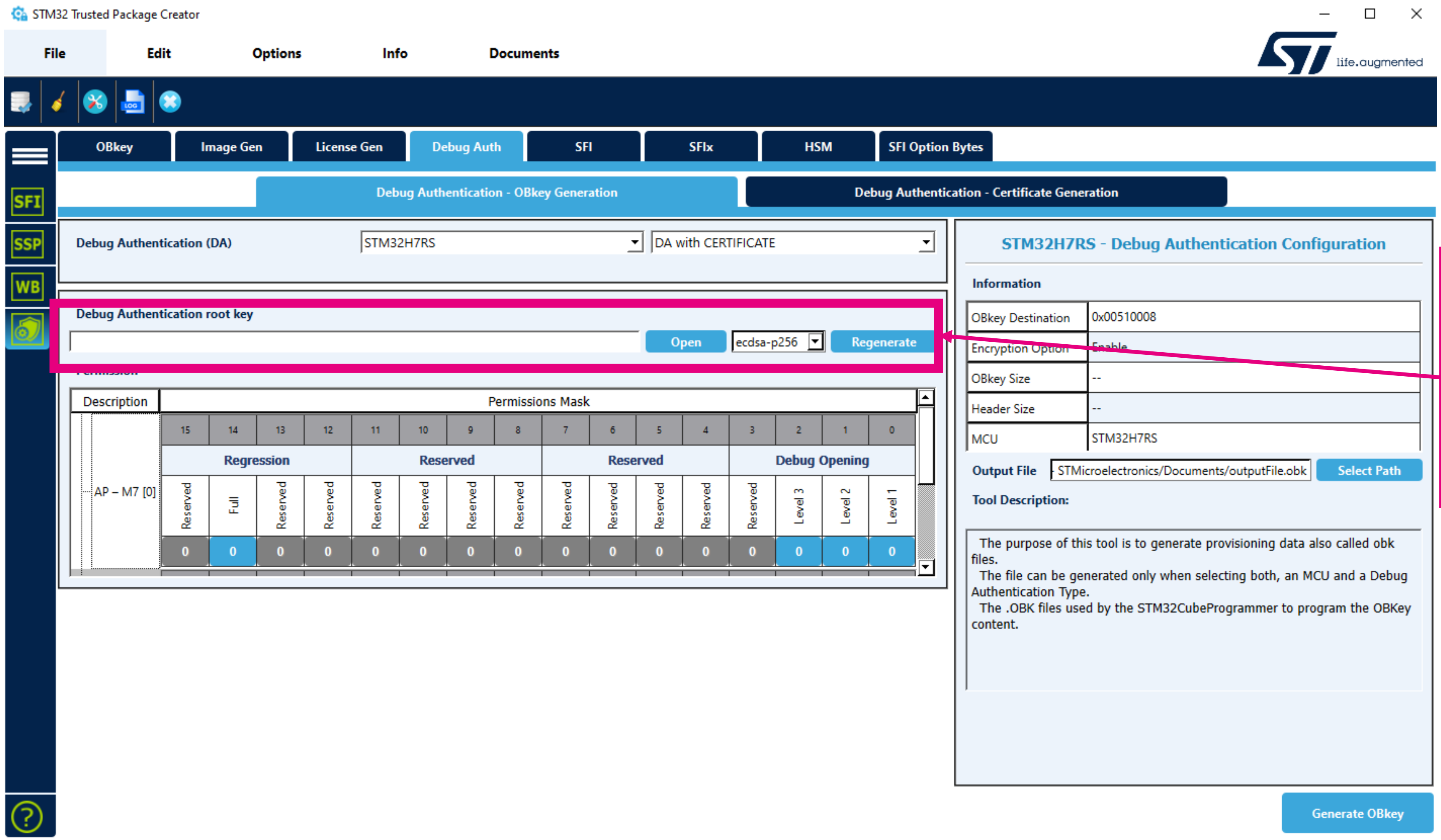
STM32H7RS Security
Debug Authentication Provisioning



DA_Config.obk
- Hash of a key
- Permission mask

How to generate a Debug Authentication obk file ?
Input : an ecc key

STM32 Trusted Package Creator



Debug Authentication (DA): STM32H7RS | DA with CERTIFICATE

Debug Authentication root key: ecdsa-p256

Permission

Description	Permissions Mask															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Regression				Reserved				Reserved				Debug Opening			
AP – M7 [0]	Reserved	Full	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Level 3	Level 2	Level 1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM32H7RS - Debug Authentication Configuration

Information

OBkey Destination	0x00510008
Encryption Option	Enable
OBkey Size	--
Header Size	--
MCU	STM32H7RS

Output File: STMicroelectronics/Documents/outputFile.obk

Tool Description:

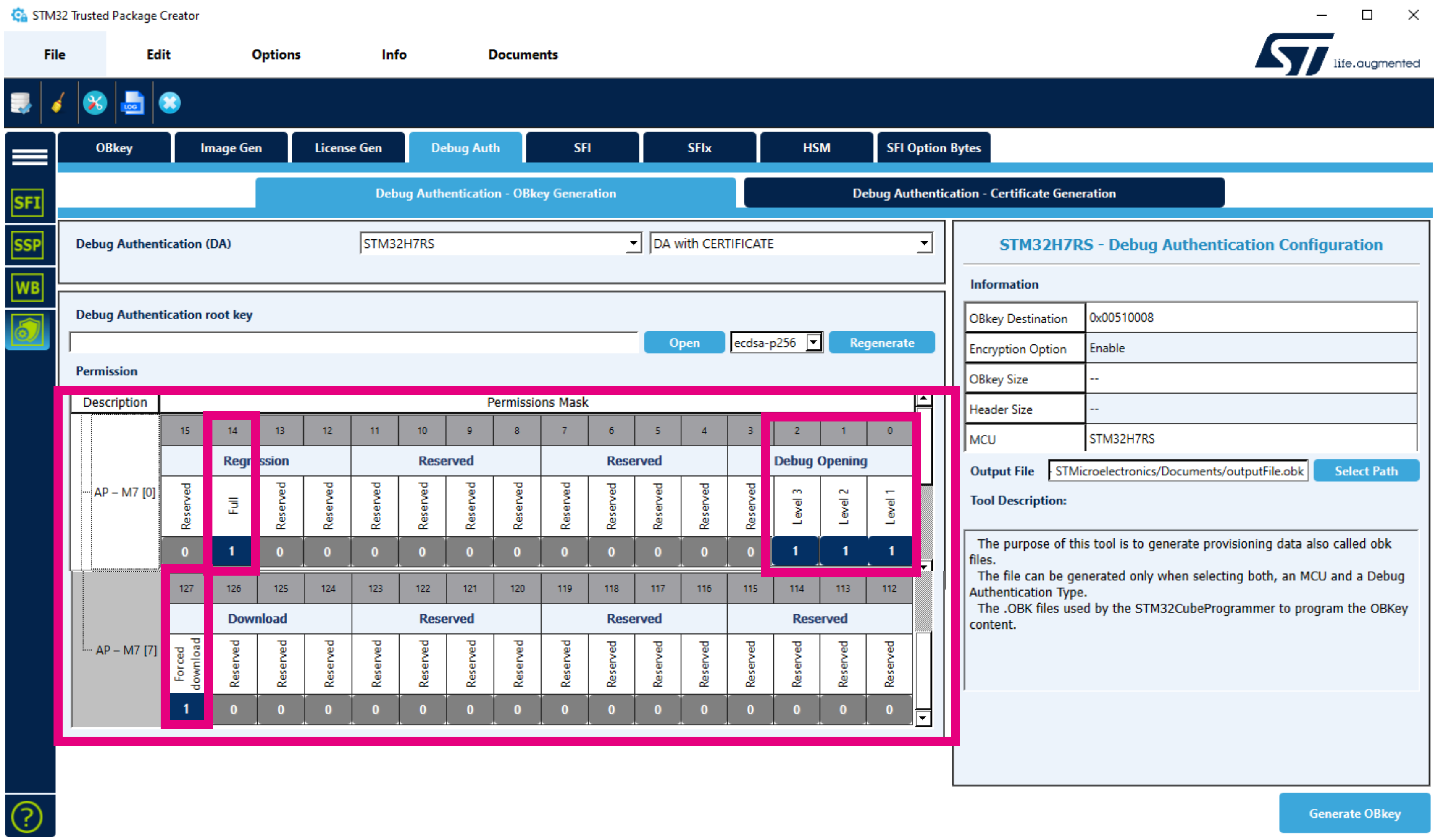
The purpose of this tool is to generate provisioning data also called obk files.
The file can be generated only when selecting both, an MCU and a Debug Authentication Type.
The .OBK files used by the STM32CubeProgrammer to program the OBKey content.

An ECC Key to generate the hash of its public key.

The private key will be needed to reopen the device

How to generate a Debug Authentication obk file ?
Input : the soc mask

STM32 Trusted Package Creator



STM32 Trusted Package Creator

Debug Authentication (DA): STM32H7RS | DA with CERTIFICATE

Debug Authentication root key: ecdsa-p256

Permission

Description	Permissions Mask															
AP – M7 [0]	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Regression				Reserved				Reserved				Debug Opening			
	Reserved	Full	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Level 3	Level 2	Level 1
	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
AP – M7 [7]	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
	Download				Reserved				Reserved				Reserved			
	Forced download	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM32H7RS - Debug Authentication Configuration

Information

OBkey Destination	0x00510008
Encryption Option	Enable
OBkey Size	--
Header Size	--
MCU	STM32H7RS

Output File: STMicroelectronics/Documents/outputFile.obk

Tool Description:

The purpose of this tool is to generate provisioning data also called obk files.
The file can be generated only when selecting both, an MCU and a Debug Authentication Type.
The .OBK files used by the STM32CubeProgrammer to program the OBKey content.

SOC MASK
Maximum action via debugging link when device will be Closed

Regression capability

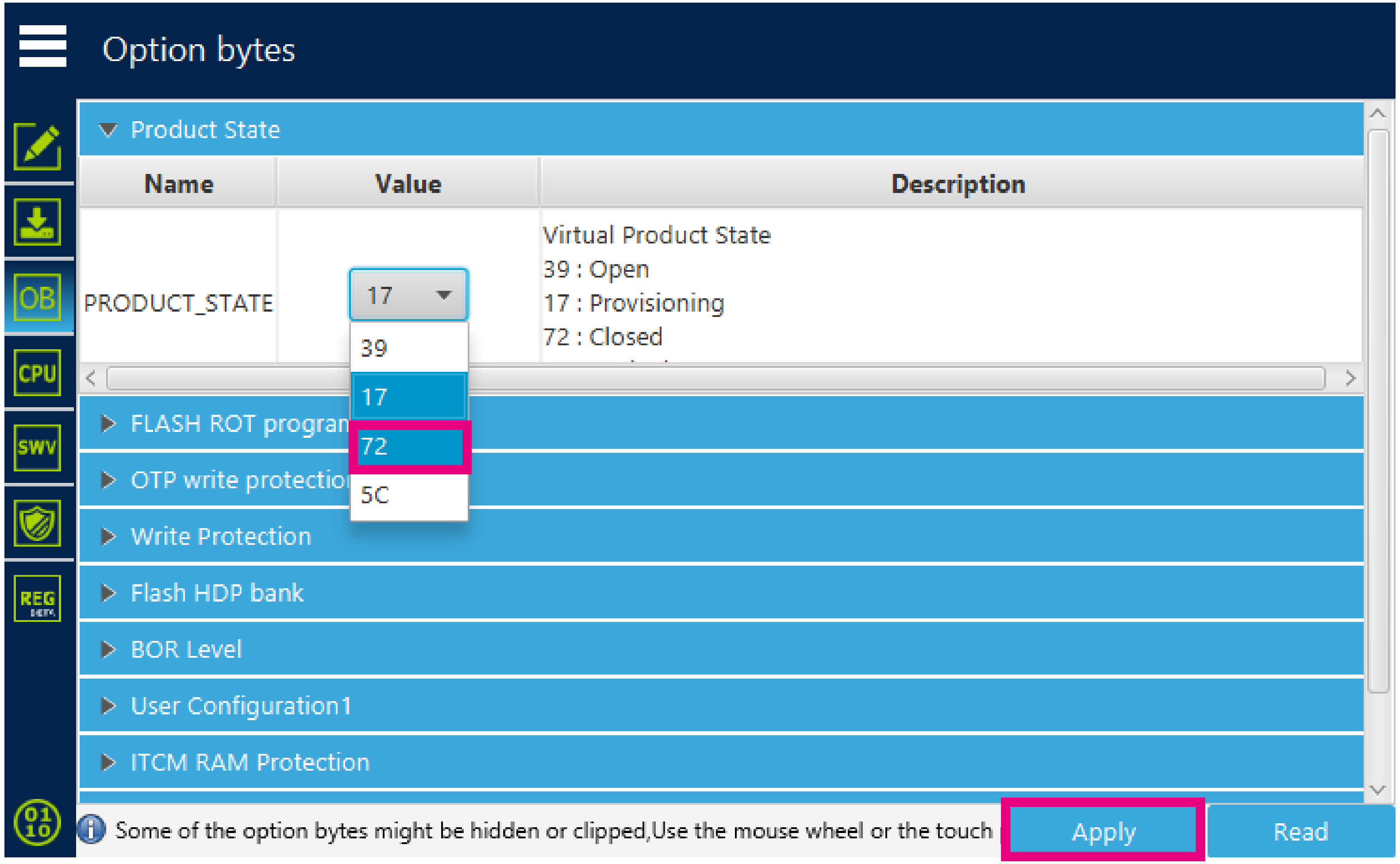
Debug capability

Force to activate bootloader capability

Key Steps for the Hands-On Exercise

STEP 2.2

Close the device



Option bytes

Product State

Name	Value	Description
PRODUCT_STATE	17	Virtual Product State 39 : Open 17 : Provisioning 72 : Closed

FLASH ROT progran

OTP write protectio

Write Protection

Flash HDP bank

BOR Level

User Configuration1

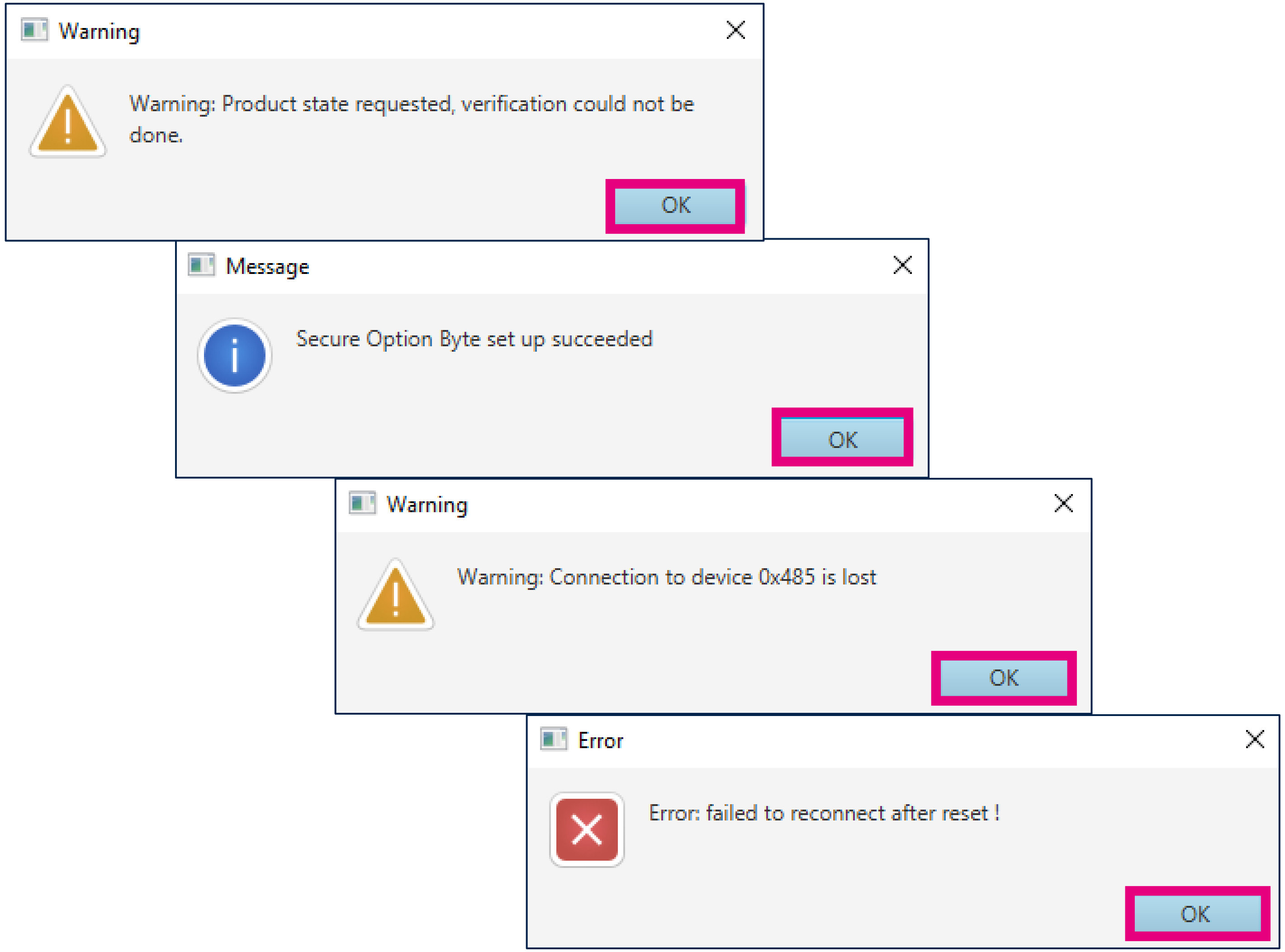
ITCM RAM Protection

Some of the option bytes might be hidden or clipped,Use the mouse wheel or the touch

STM32H7RS Security Debug Authentication Provisioning

STEP 2.2

Close the device



Warning

Warning: Product state requested, verification could not be done.

Message

Secure Option Byte set up succeeded

Warning

Warning: Connection to device 0x485 is lost

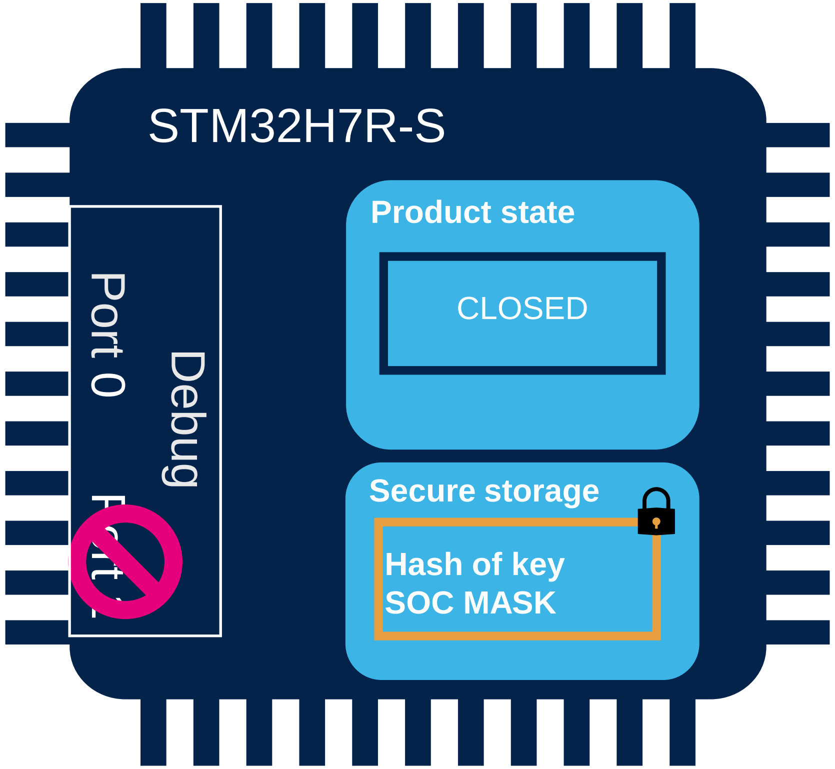
Error

Error: failed to reconnect after reset !

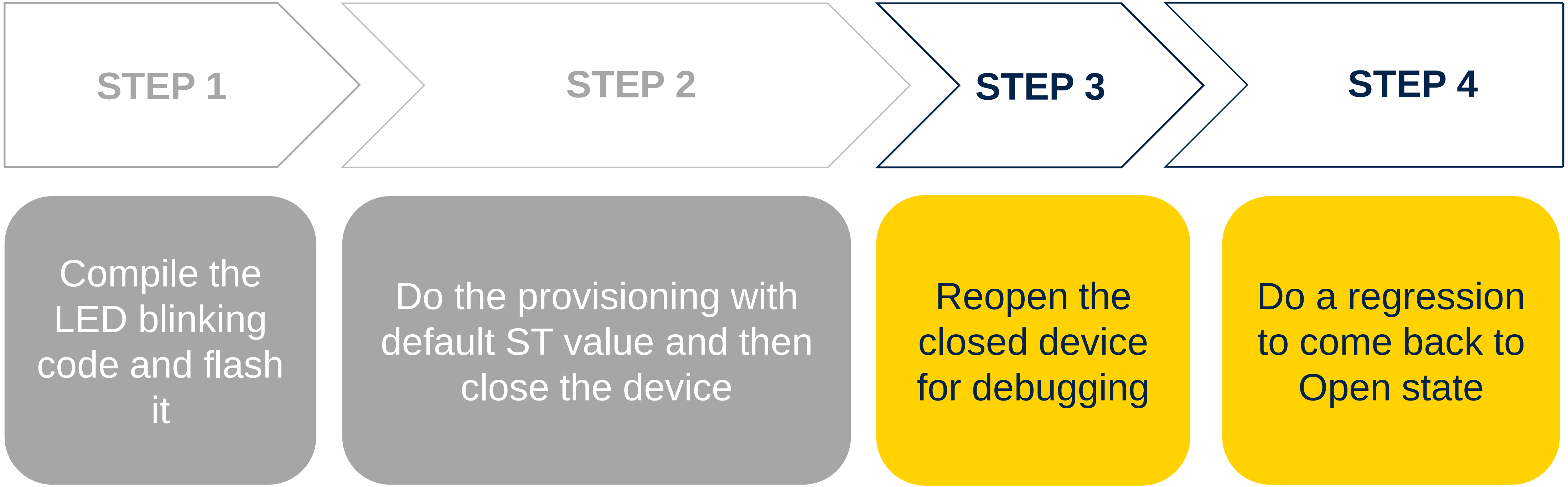
STM32HRS Security
Debug Authentication Process

STEP 2.2

Close the device



Key Steps for the Hands-On Exercise



STEP 1

Compile the LED blinking code and flash it

STEP 2

Do the provisioning with default ST value and then close the device

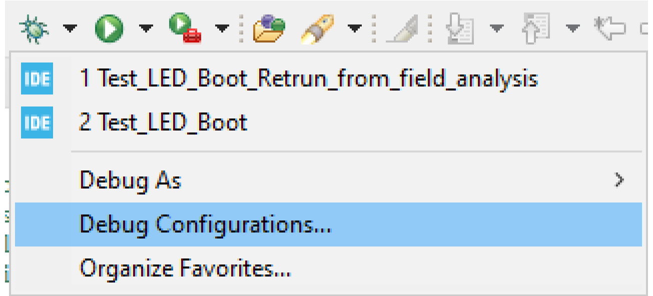
STEP 3

Reopen the closed device for debugging

STEP 4

Do a regression to come back to Open state

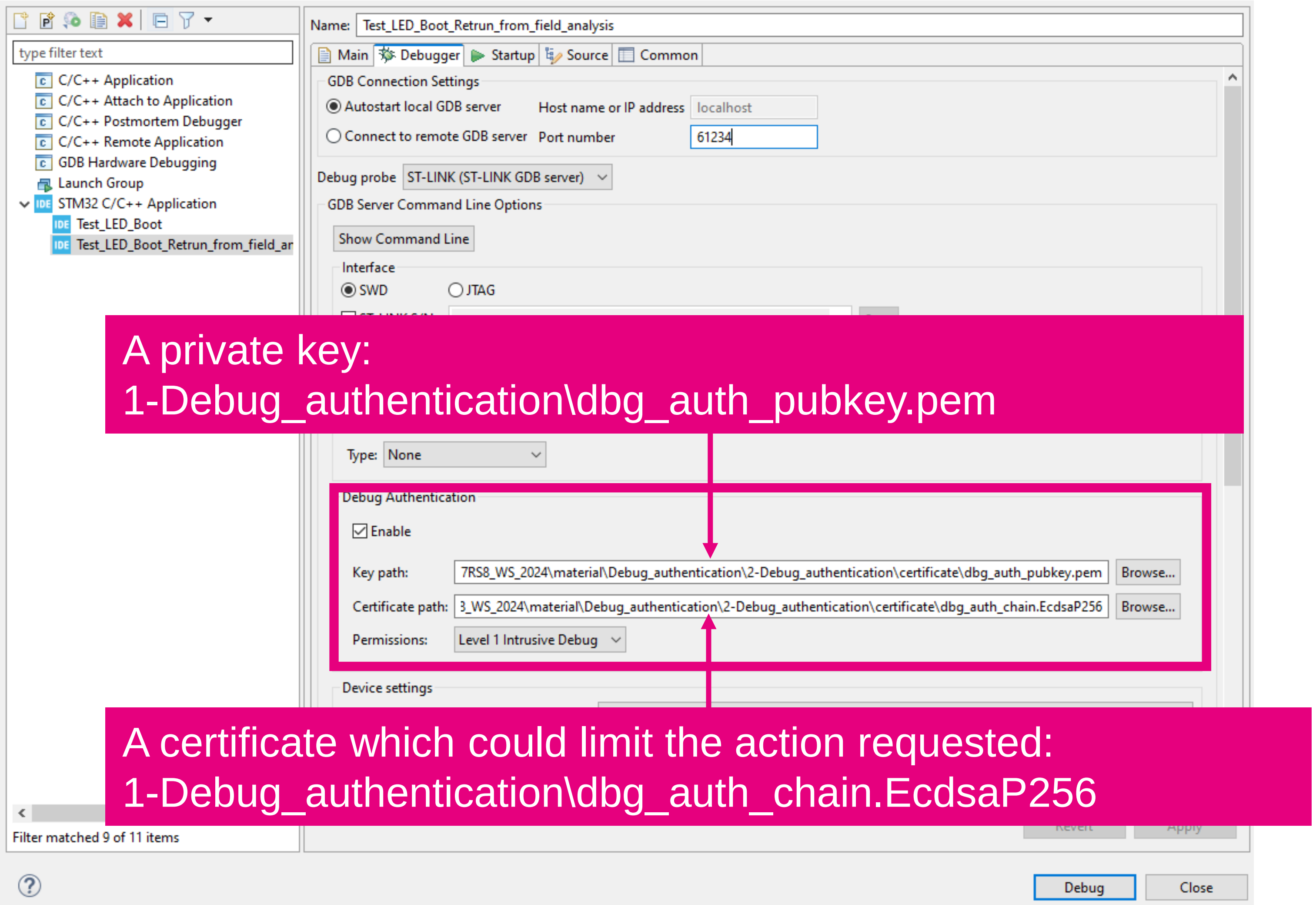
STM32H7RS Security
Debug Authentication reopening



1 Test_LED_Boot_Retrun_from_field_analysis
2 Test_LED_Boot
Debug As
Debug Configurations...
Organize Favorites...

STEP 3

Reopen the closed device for debugging



type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- Launch Group
- STM32 C/C++ Application
  - Test_LED_Boot
  - Test_LED_Boot_Retrun_from_field_ar

Filter matched 9 of 11 items

Name: Test_LED_Boot_Retrun_from_field_analysis

Main | Debugger | Startup | Source | Common

GDB Connection Settings

● Autostart local GDB server — Host name or IP address: localhost

○ Connect to remote GDB server — Port number: 61234

Debug probe: ST-LINK (ST-LINK GDB server)

GDB Server Command Line Options

Show Command Line

Interface: ● SWD ○ JTAG

Type: None

Debug Authentication

☑ Enable

Key path: 7RS8_WS_2024\material\Debug_authentication\2-Debug_authentication\certificate\dbg_auth_pubkey.pem — Browse...

Certificate path: 3_WS_2024\material\Debug_authentication\2-Debug_authentication\certificate\dbg_auth_chain.EcdsaP256 — Browse...

Permissions: Level 1 Intrusive Debug

Device settings

Revert | Apply

Debug | Close

A private key:
1-Debug_authentication\dbg_auth_pubkey.pem

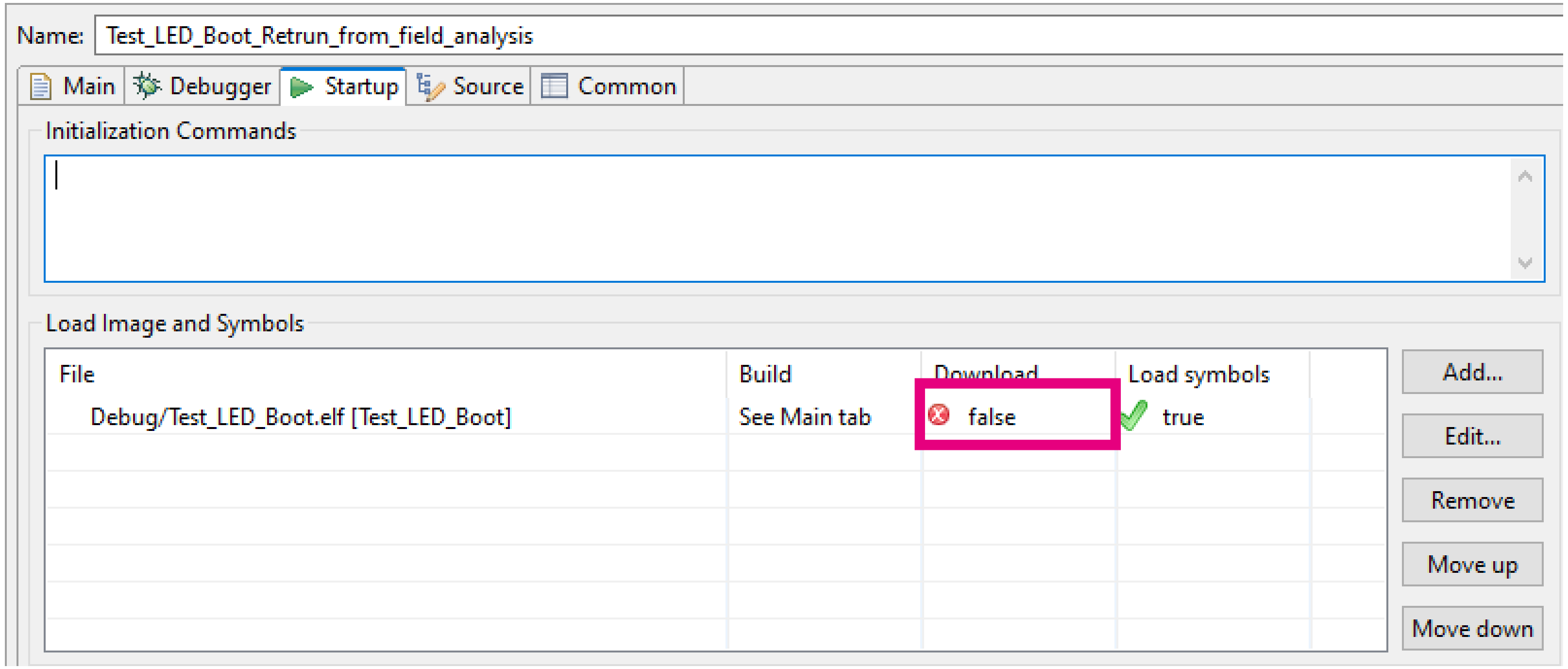
A certificate which could limit the action requested:
1-Debug_authentication\dbg_auth_chain.EcdsaP256

life.augmented

STM32H7RS Security
Debug Authentication Provisioning

STEP 3

Reopen the closed device for debugging



Name: Test_LED_Boot_Retrun_from_field_analysis

Initialization Commands

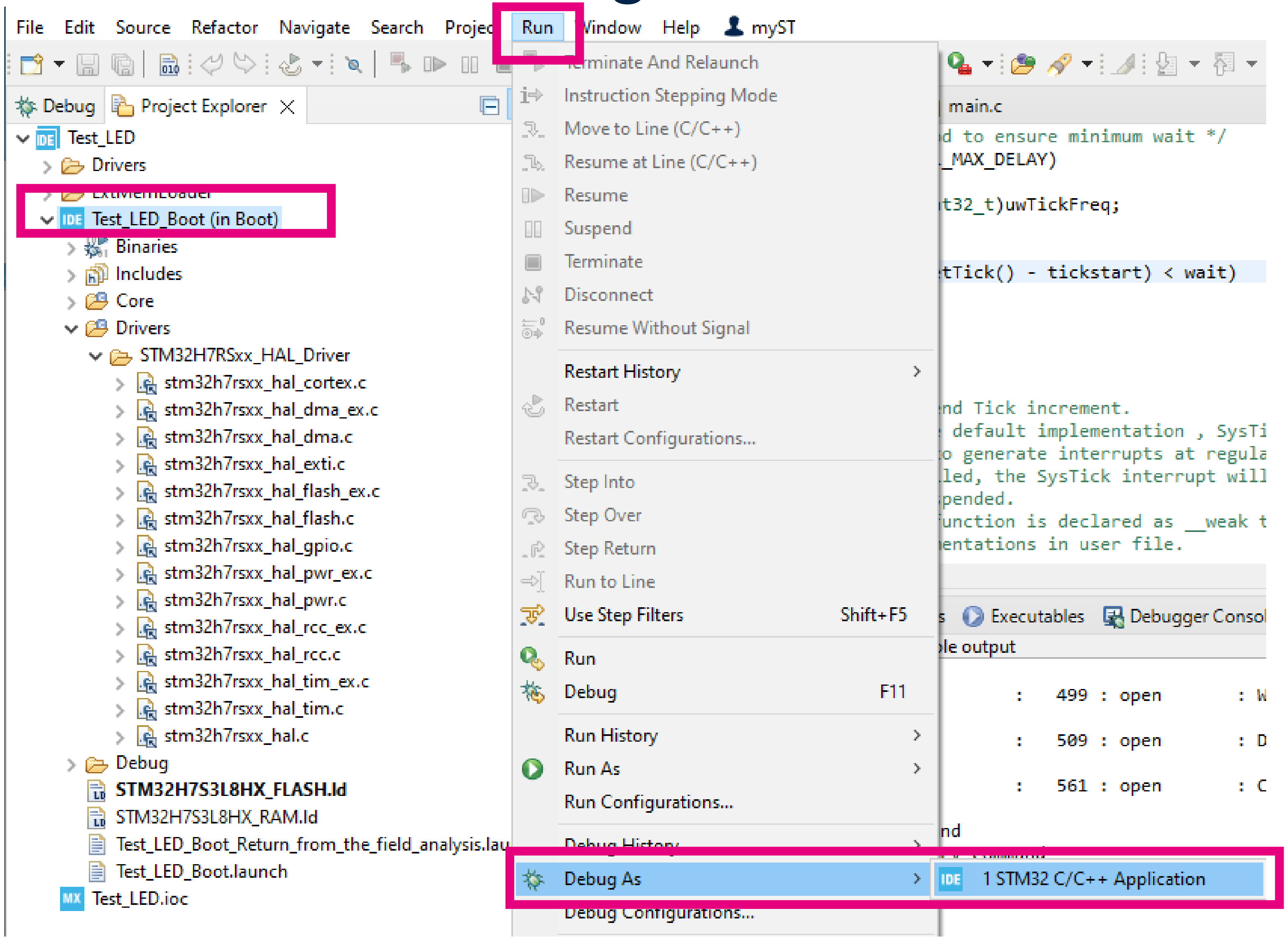
Load Image and Symbols

File	Build	Download	Load symbols
Debug/Test_LED_Boot.elf [Test_LED_Boot]	See Main tab	☒ false	✓ true

STM32H7RS Security
Debug Authentication reopening

STEP 3

Reopen the closed device for debugging



```
d to ensure minimum wait */
_MAX_DELAY)

t32_t)uwTickFreq;

tTick() - tickstart) < wait)

nd Tick increment.
 default implementation , SysTi
o generate interrupts at regula
led, the SysTick interrupt will
pended.
unction is declared as __weak t
entations in user file.
```

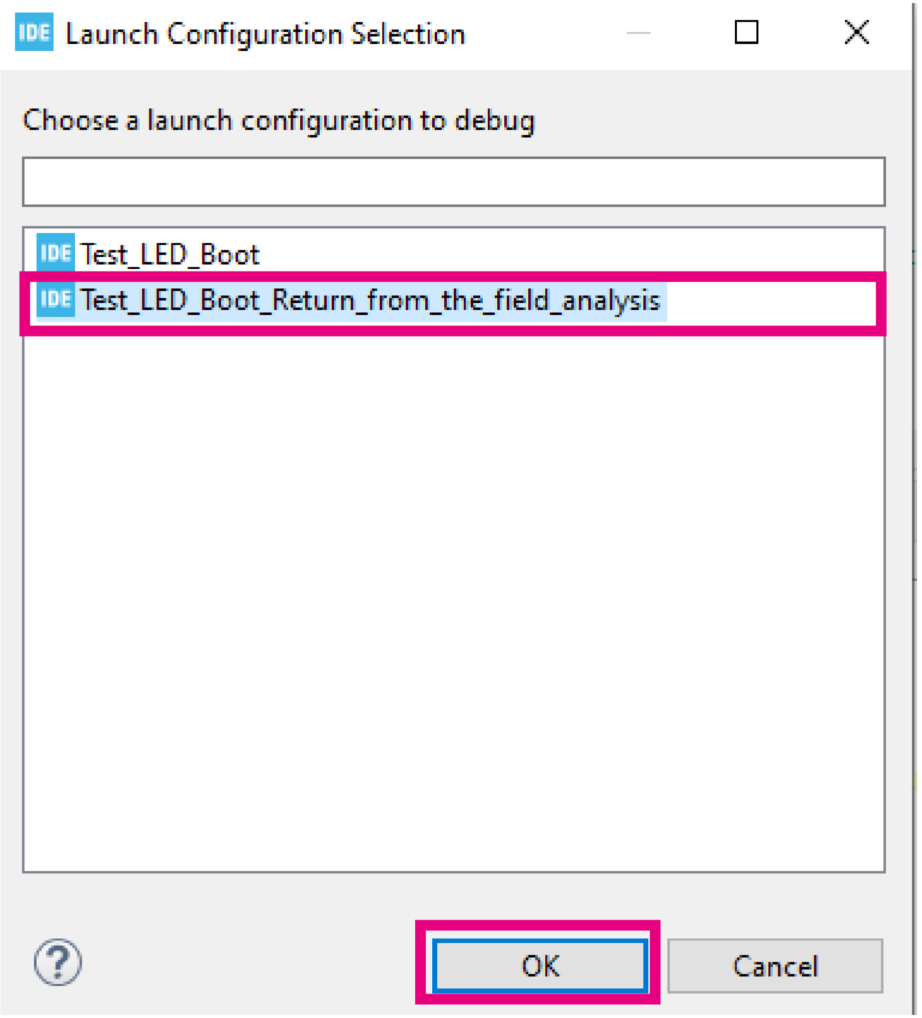
```
le output

    :   499 : open      : W

    :   509 : open      : D

    :   561 : open      : C

nd
```

life.augmented

STM32H7RS Security
Debug Authentication reopening

STEP 3

Reopen the closed device for debugging


Launch Configuration Selection

Choose a launch configuration to debug

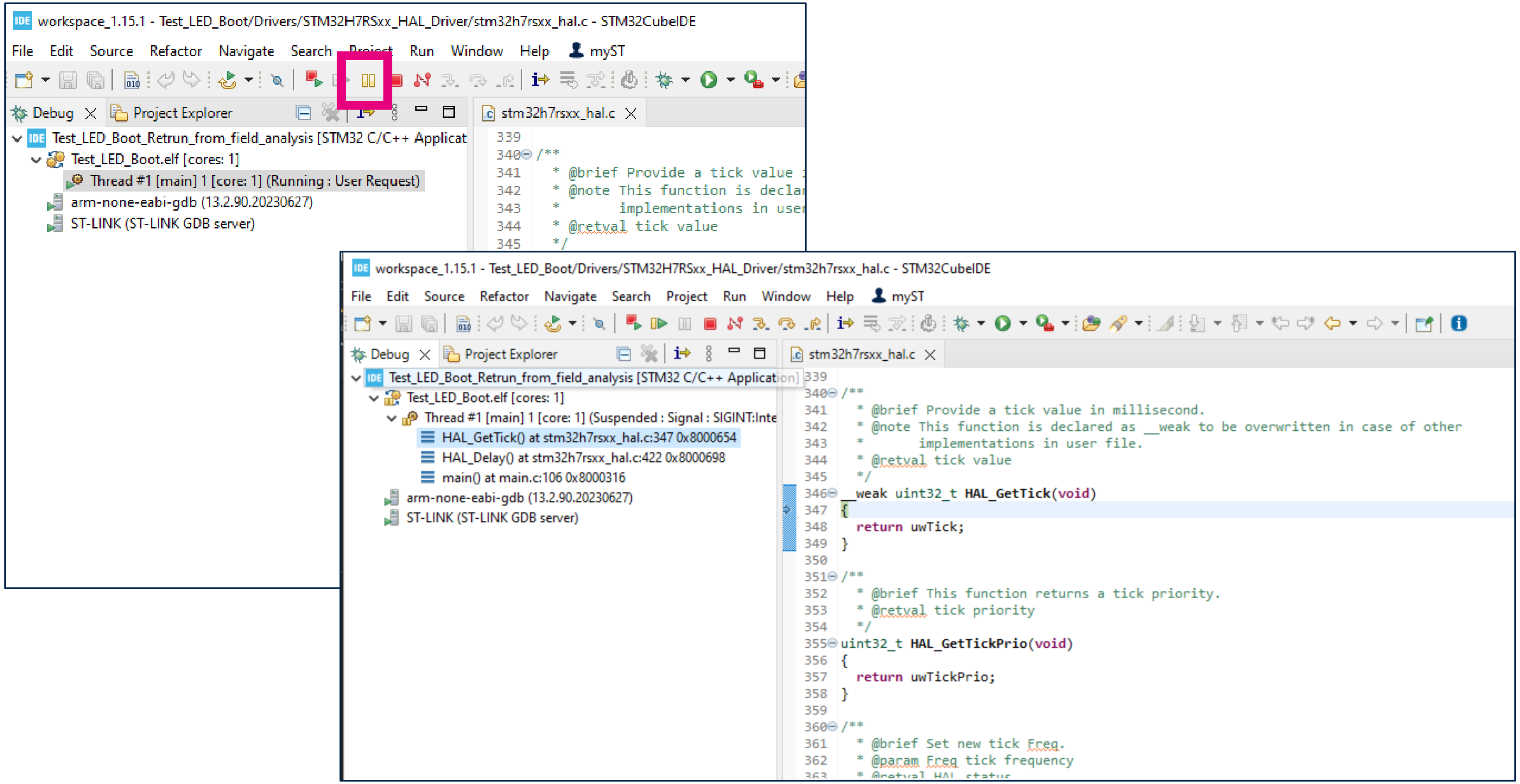
Test_LED_Boot

Test_LED_Boot_Return_from_the_field_analysis

STM32H7RS Security
Debug Authentication reopening

STEP 3

Reopen the closed device for debugging



workspace_1.15.1 - Test_LED_Boot/Drivers/STM32H7RSxx_HAL_Driver/stm32h7rsxx_hal.c - STM32CubeIDE

```
Test_LED_Boot_Retrun_from_field_analysis [STM32 C/C++ Applicat
  Test_LED_Boot.elf [cores: 1]
    Thread #1 [main] 1 [core: 1] (Running : User Request)
  arm-none-eabi-gdb (13.2.90.20230627)
  ST-LINK (ST-LINK GDB server)
```

```
/**
  * @brief Provide a tick value
  * @note This function is declar
  *       implementations in user
  * @retval tick value
  */
```

workspace_1.15.1 - Test_LED_Boot/Drivers/STM32H7RSxx_HAL_Driver/stm32h7rsxx_hal.c - STM32CubeIDE

```
Test_LED_Boot_Retrun_from_field_analysis [STM32 C/C++ Application]
  Test_LED_Boot.elf [cores: 1]
    Thread #1 [main] 1 [core: 1] (Suspended : Signal : SIGINT:Inte
      HAL_GetTick() at stm32h7rsxx_hal.c:347 0x8000654
      HAL_Delay() at stm32h7rsxx_hal.c:422 0x8000698
      main() at main.c:106 0x8000316
  arm-none-eabi-gdb (13.2.90.20230627)
  ST-LINK (ST-LINK GDB server)
```

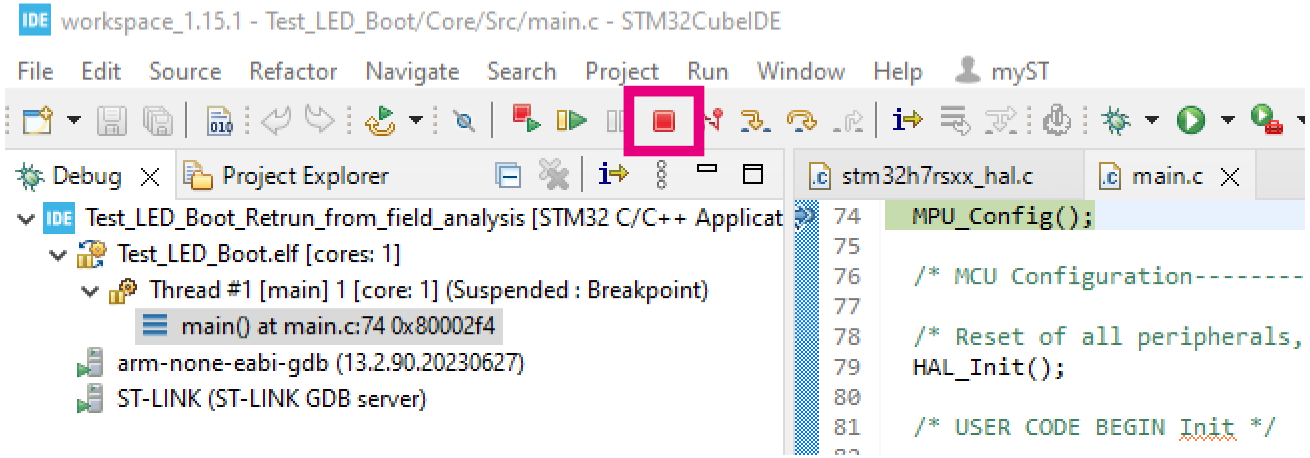
```

/**
  * @brief Provide a tick value in millisecond.
  * @note This function is declared as __weak to be overwritten in case of other
  *       implementations in user file.
  * @retval tick value
  */
__weak uint32_t HAL_GetTick(void)
{
  return uwTick;
}

/**
  * @brief This function returns a tick priority.
  * @retval tick priority
  */
uint32_t HAL_GetTickPrio(void)
{
  return uwTickPrio;
}

/**
  * @brief Set new tick Freq.
  * @param Freq tick frequency
  * @retval HAL status
```

life.augmented

STM32H7RS Security
Debug Authentication reopening

STEP 3

Reopen the closed device for debugging

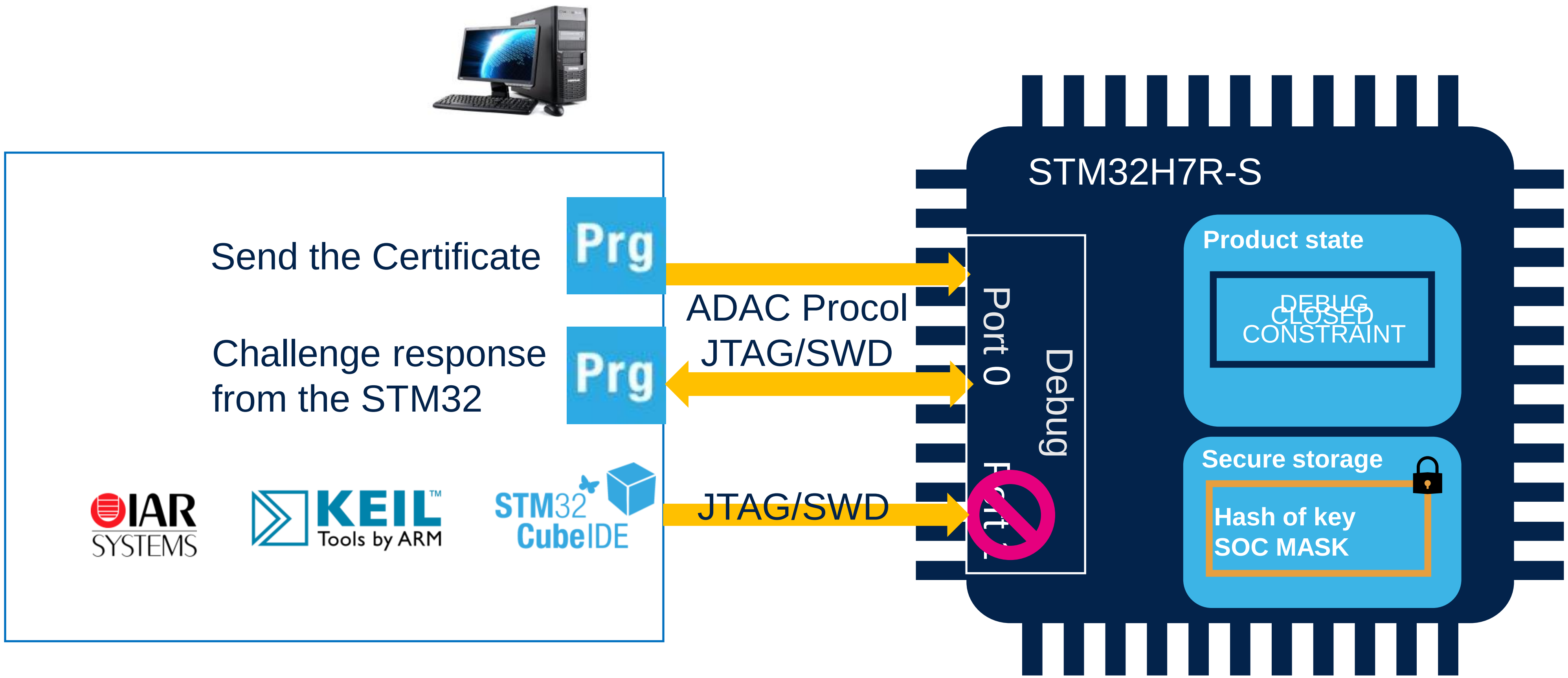


workspace_1.15.1 - Test_LED_Boot/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help myST
Debug
Project Explorer
stm32h7rsxx_hal.c
main.c

Test_LED_Boot_Retrun_from_field_analysis [STM32 C/C++ Applicat
Test_LED_Boot.elf [cores: 1]
Thread #1 [main] 1 [core: 1] (Suspended : Breakpoint)
main() at main.c:74 0x80002f4
arm-none-eabi-gdb (13.2.90.20230627)
ST-LINK (ST-LINK GDB server)

```
MPU_Config();

/* MCU Configuration--------

/* Reset of all peripherals,
HAL_Init();

/* USER CODE BEGIN Init */
```

STM32H7RS Security
Debug Authentication Process



Send the Certificate

Prg

Challenge response from the STM32

Prg

ADAC Procol JTAG/SWD

JTAG/SWD

IAR SYSTEMS

KEIL Tools by ARM

STM32 CubeIDE

STM32H7R-S

Port 0

Port 1

Debug

Product state

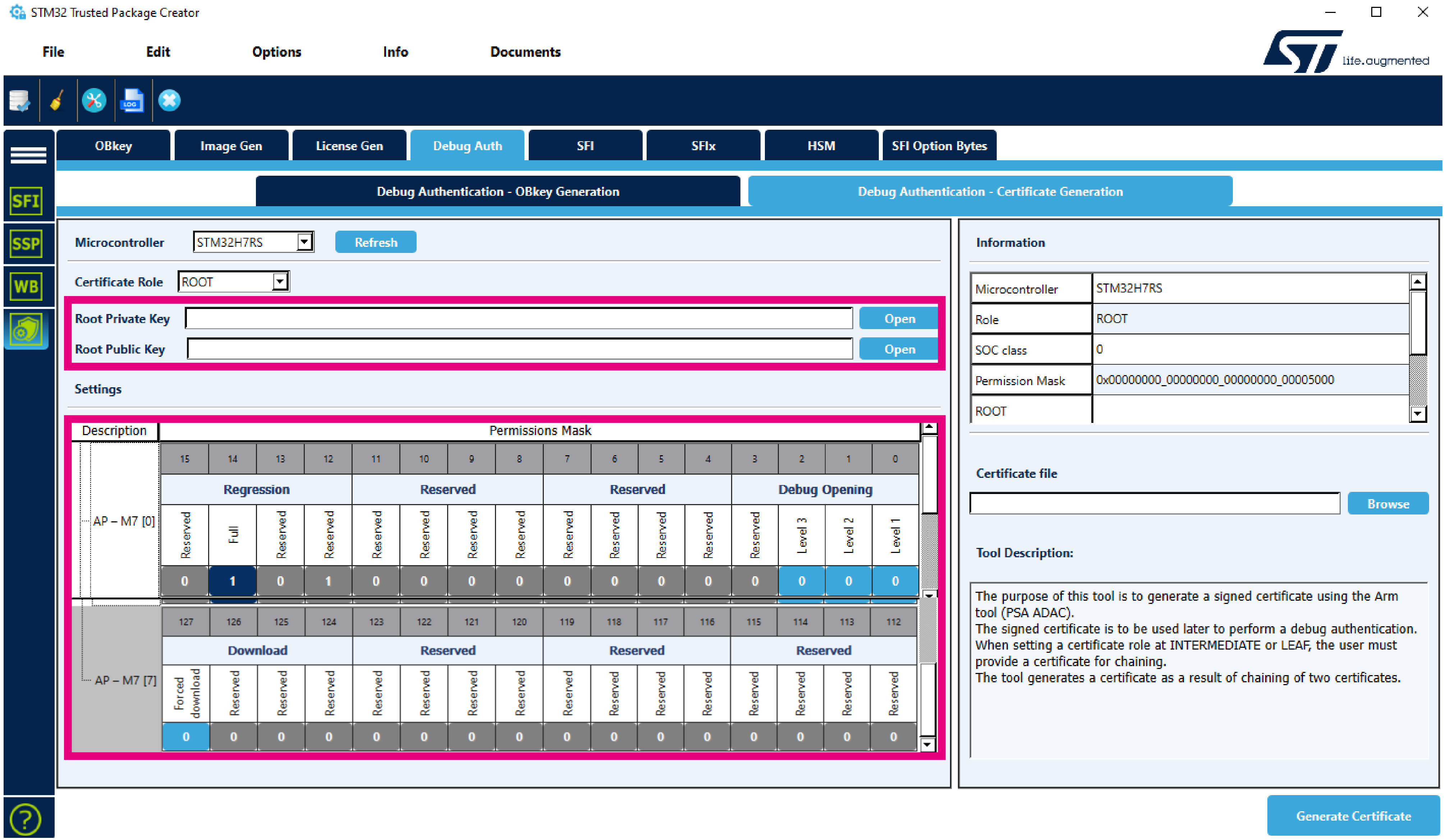
DEBUG CLOSED CONSTRAINT

Secure storage

Hash of key
SOC MASK

FYI : Debug port1 is open until the next power cycle but can also be closed thanks a command

How was generate the certificate ?



Microcontroller: STM32H7RS

Certificate Role: ROOT

Root Private Key:

Root Public Key:

Settings

Description	Permissions Mask															
AP – M7 [0]	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Regression				Reserved				Reserved				Debug Opening			
	Reserved	Full	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Level 3	Level 2	Level 1
	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
AP – M7 [7]	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
	Download				Reserved				Reserved				Reserved			
	Forced download	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Information

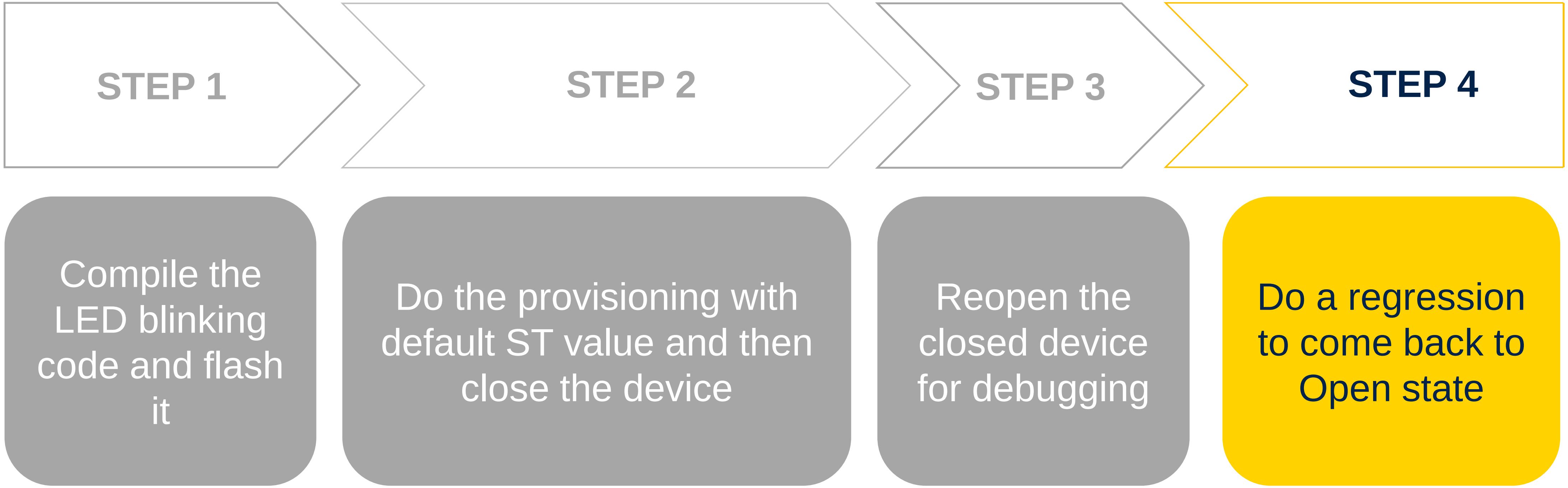
Microcontroller	STM32H7RS
Role	ROOT
SOC class	0
Permission Mask	0x00000000_00000000_00000000_00005000
ROOT	

Certificate file

Tool Description:

The purpose of this tool is to generate a signed certificate using the Arm tool (PSA ADAC).
The signed certificate is to be used later to perform a debug authentication.
When setting a certificate role at INTERMEDIATE or LEAF, the user must provide a certificate for chaining.
The tool generates a certificate as a result of chaining of two certificates.

Key Steps for the Hands-On Exercise



STEP 1

Compile the LED blinking code and flash it

STEP 2

Do the provisioning with default ST value and then close the device

STEP 3

Reopen the closed device for debugging

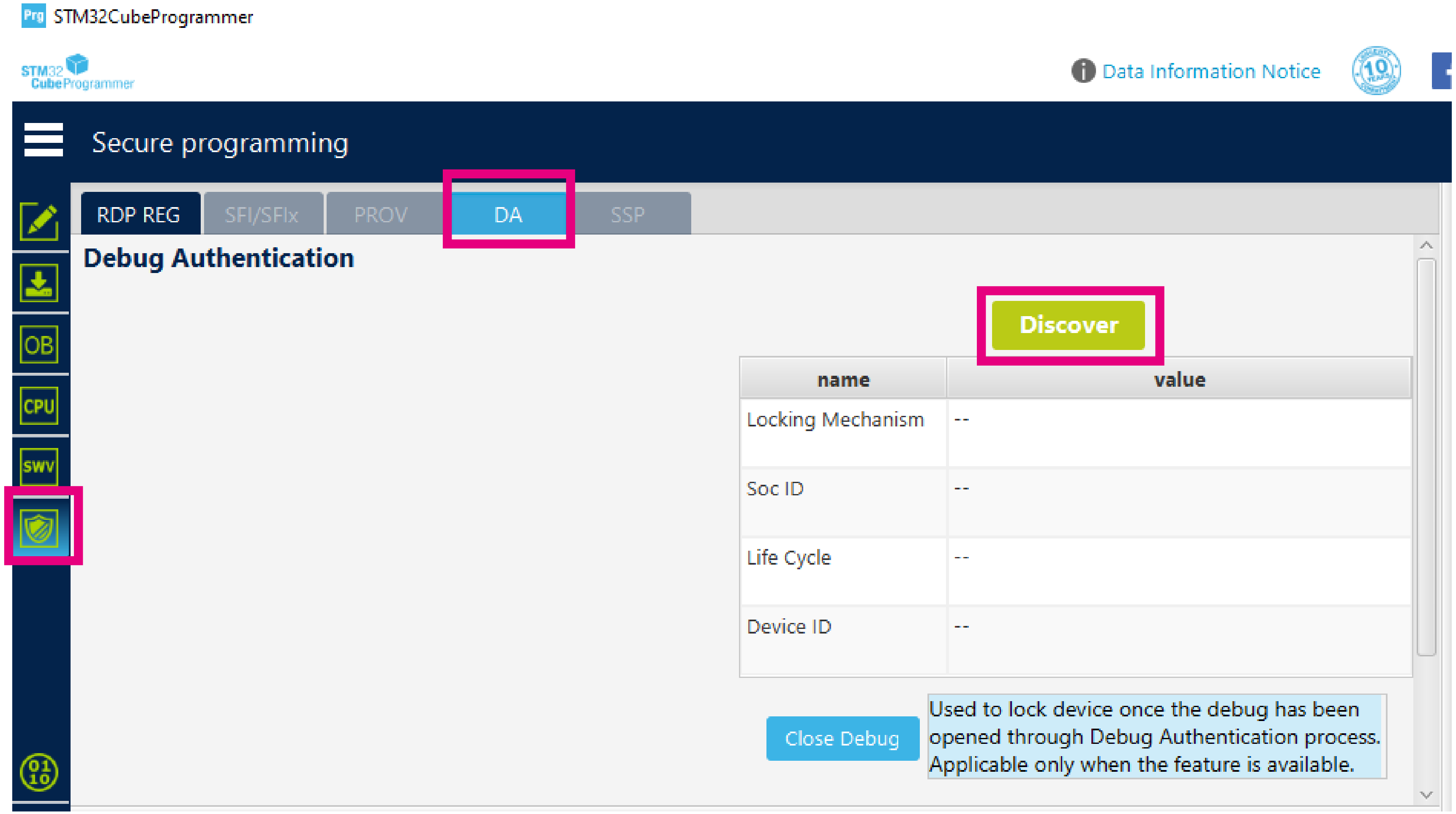
STEP 4

Do a regression to come back to Open state

Let’s do a regression

STEP 4

Do a regression to come back to Open state



Debug Authentication

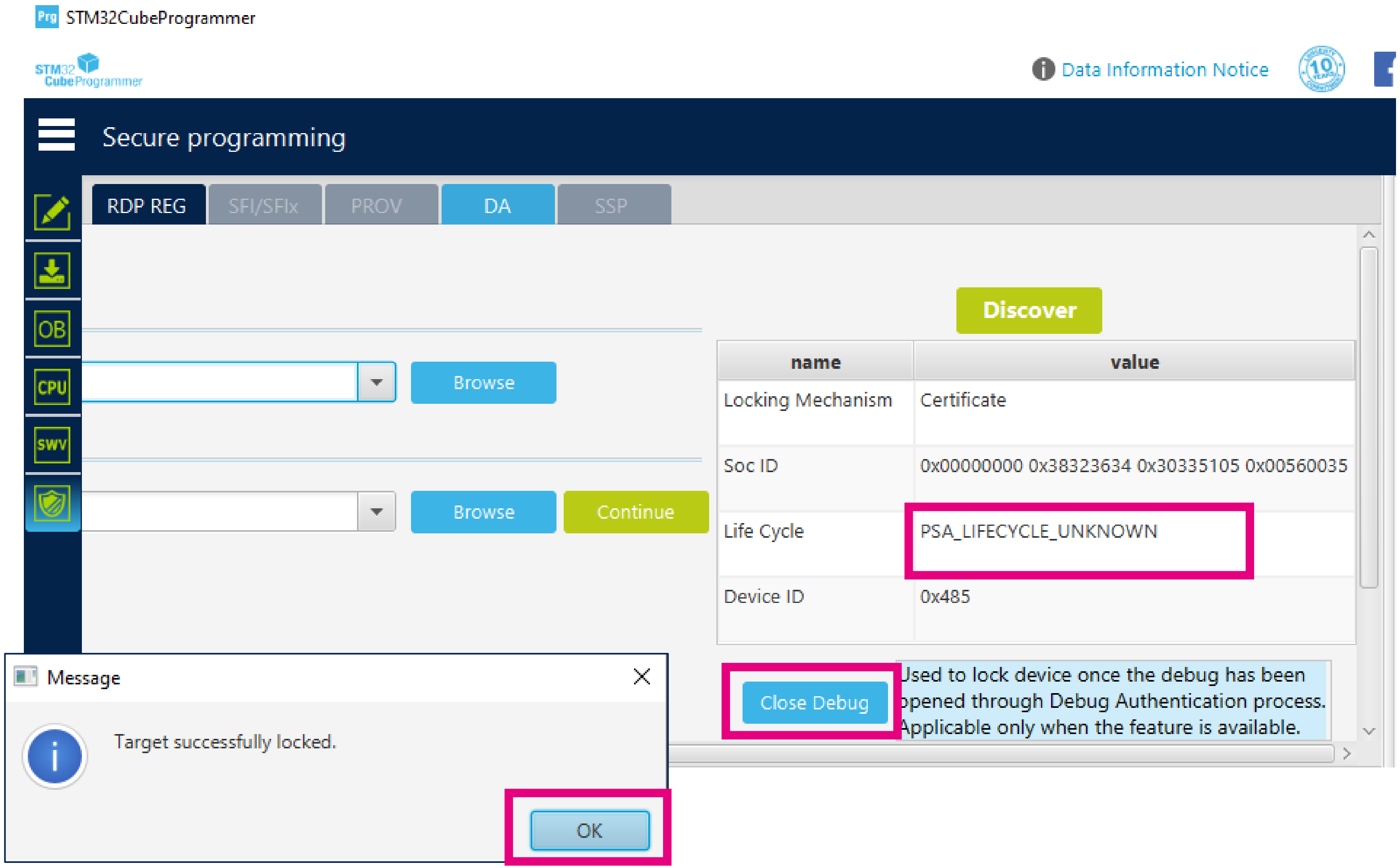
name	value
Locking Mechanism	--
Soc ID	--
Life Cycle	--
Device ID	--

Used to lock device once the debug has been opened through Debug Authentication process. Applicable only when the feature is available.

Let’s do a regression

STEP 4

Do a regression to come back to Open state



name	value
Locking Mechanism	Certificate
Soc ID	0x00000000 0x38323634 0x30335105 0x00560035
Life Cycle	PSA_LIFECYCLE_UNKNOWN
Device ID	0x485

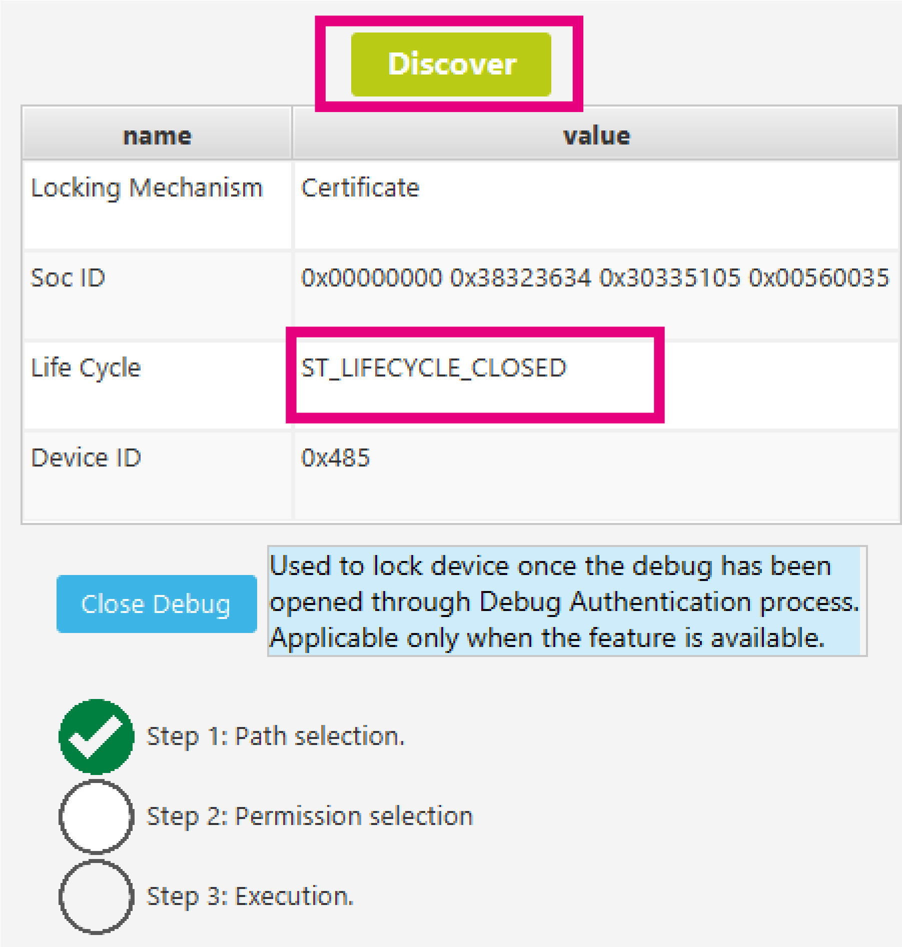
Used to lock device once the debug has been opened through Debug Authentication process. Applicable only when the feature is available.

Target successfully locked.

Let’s do a regression

STEP 4

Do a regression to come back to Open state



name	value
Locking Mechanism	Certificate
Soc ID	0x00000000 0x38323634 0x30335105 0x00560035
Life Cycle	ST_LIFECYCLE_CLOSED
Device ID	0x485

Used to lock device once the debug has been opened through Debug Authentication process. Applicable only when the feature is available.

✓ Step 1: Path selection.

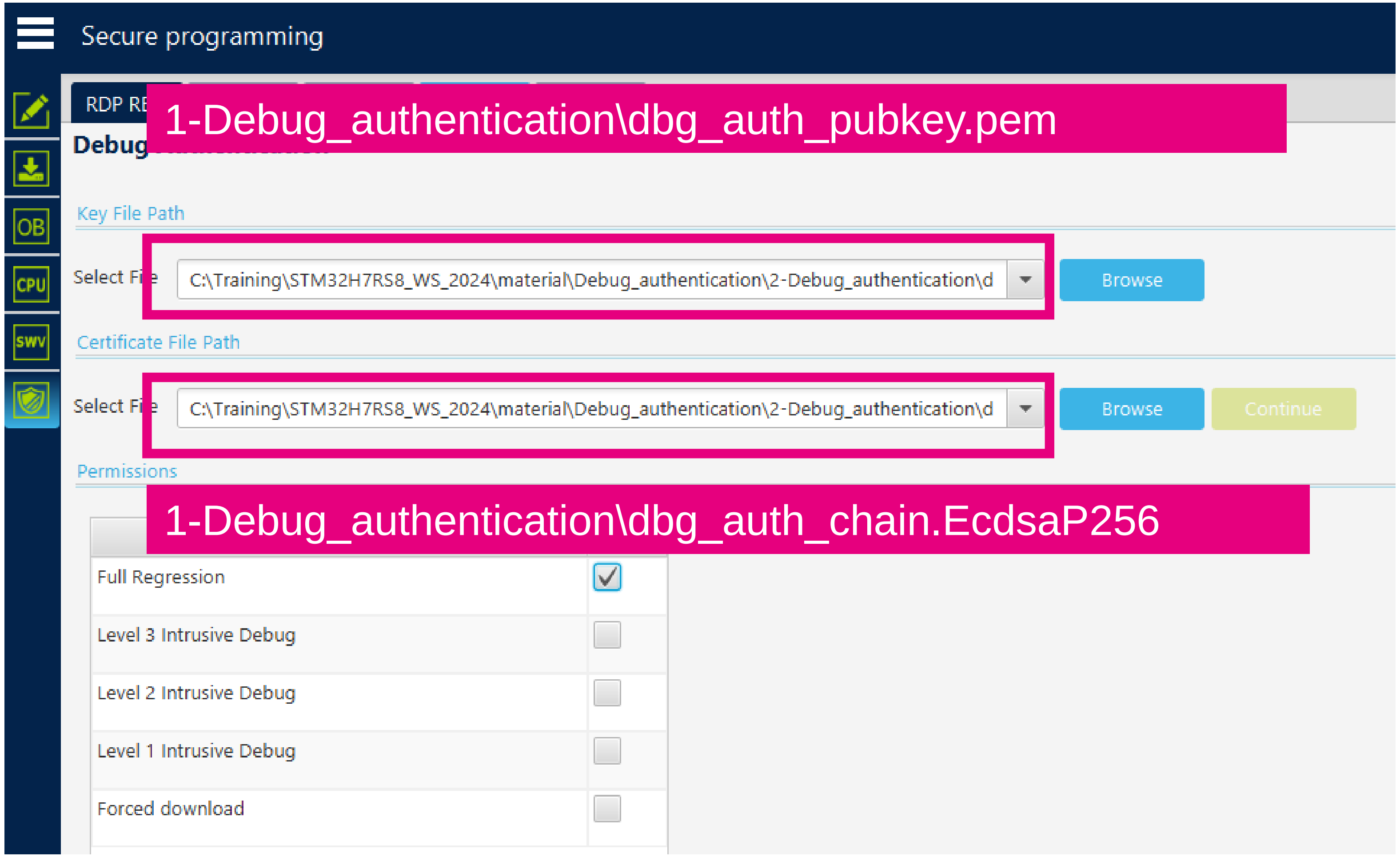
○ Step 2: Permission selection

○ Step 3: Execution.

Let’s do a regression

STEP 4

Do a regression to come back to Open state



1-Debug_authentication\dbg_auth_pubkey.pem

Key File Path

Select File: C:\Training\STM32H7RS8_WS_2024\material\Debug_authentication\2-Debug_authentication\d

Certificate File Path

Select File: C:\Training\STM32H7RS8_WS_2024\material\Debug_authentication\2-Debug_authentication\d

1-Debug_authentication\dbg_auth_chain.EcdsaP256

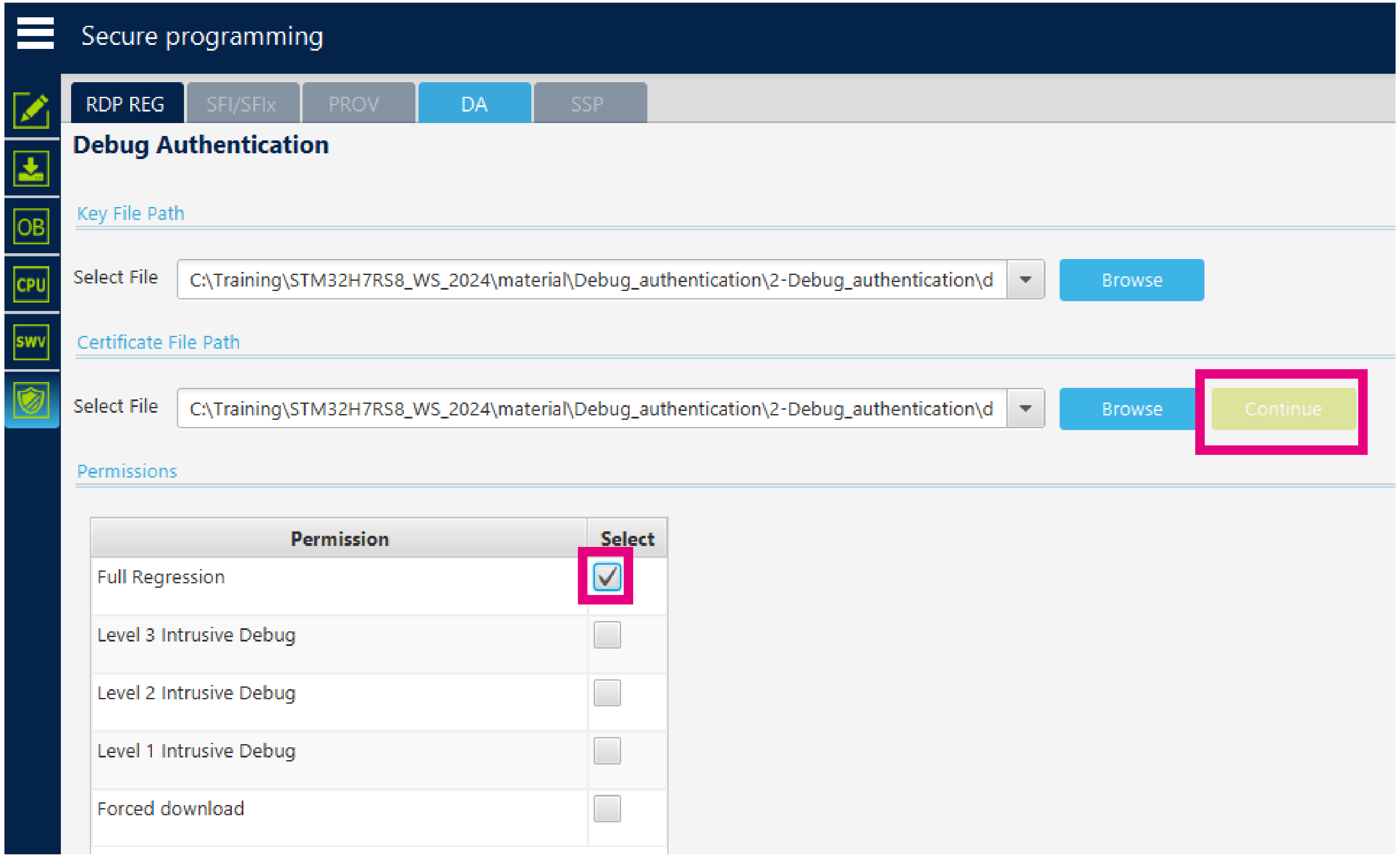
Permissions

Full Regression	☑
Level 3 Intrusive Debug	□
Level 2 Intrusive Debug	□
Level 1 Intrusive Debug	□
Forced download	□

Let’s do a regression

STEP 4

Do a regression to come back to Open state



Secure programming

Debug Authentication

Key File Path

Select File: C:\Training\STM32H7RS8_WS_2024\material\Debug_authentication\2-Debug_authentication\d

Certificate File Path

Select File: C:\Training\STM32H7RS8_WS_2024\material\Debug_authentication\2-Debug_authentication\d

Permissions

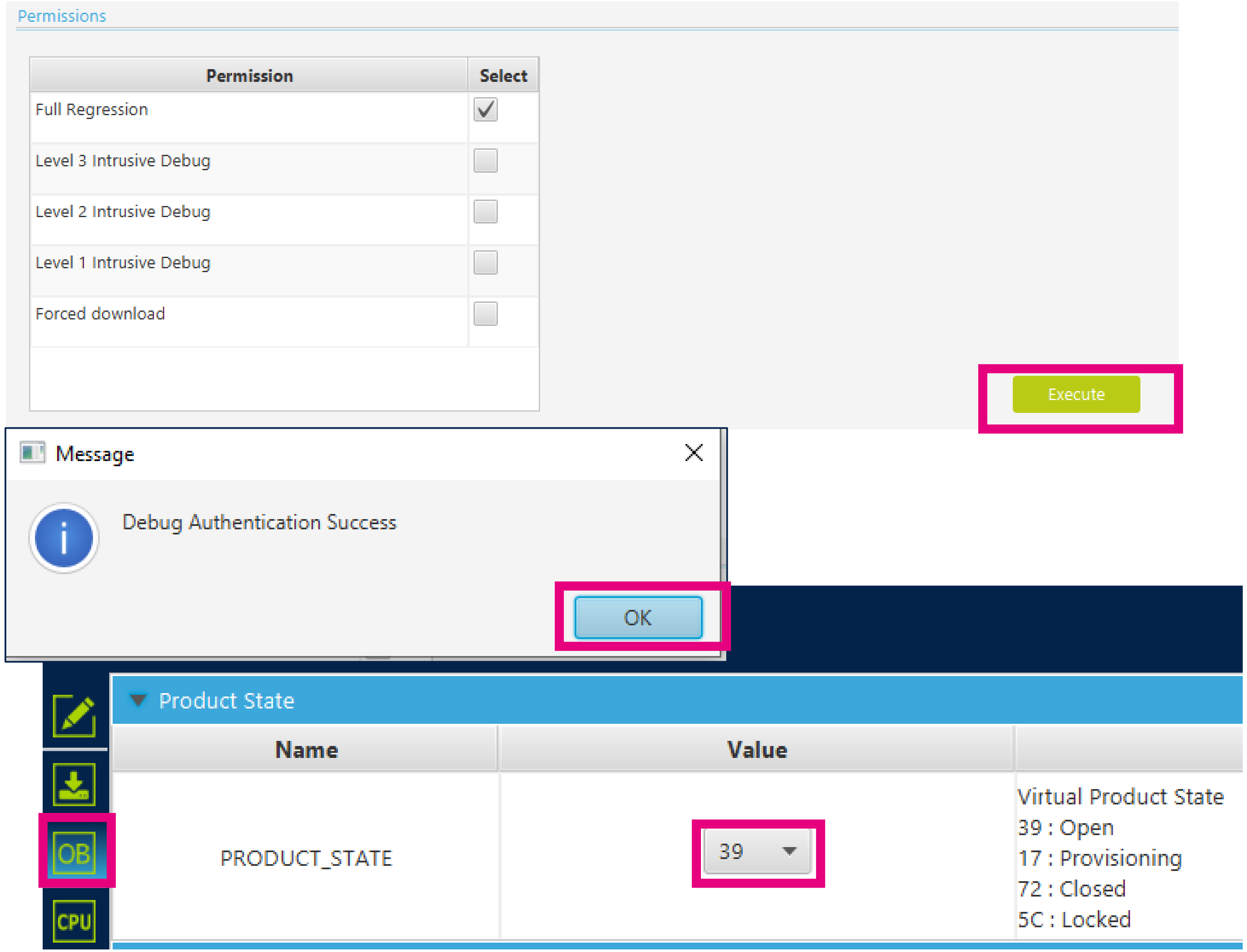
Permission	Select
Full Regression	☑
Level 3 Intrusive Debug	□
Level 2 Intrusive Debug	□
Level 1 Intrusive Debug	□
Forced download	□

life.augmented

Let’s do a regression

STEP 4

Do a regression to come back to Open state



Permissions

Permission	Select
Full Regression	☑
Level 3 Intrusive Debug	□
Level 2 Intrusive Debug	□
Level 1 Intrusive Debug	□
Forced download	□

Message

Debug Authentication Success

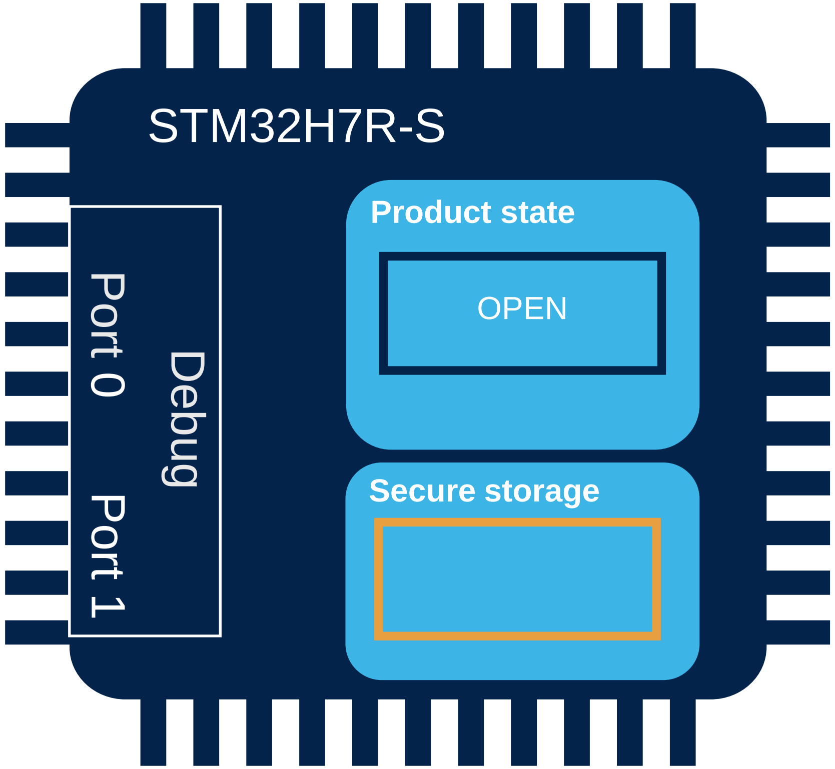
Product State

Name	Value	
PRODUCT_STATE	39	Virtual Product State 39 : Open 17 : Provisioning 72 : Closed 5C : Locked

STM32H7RS Security
Debug Authentication Process

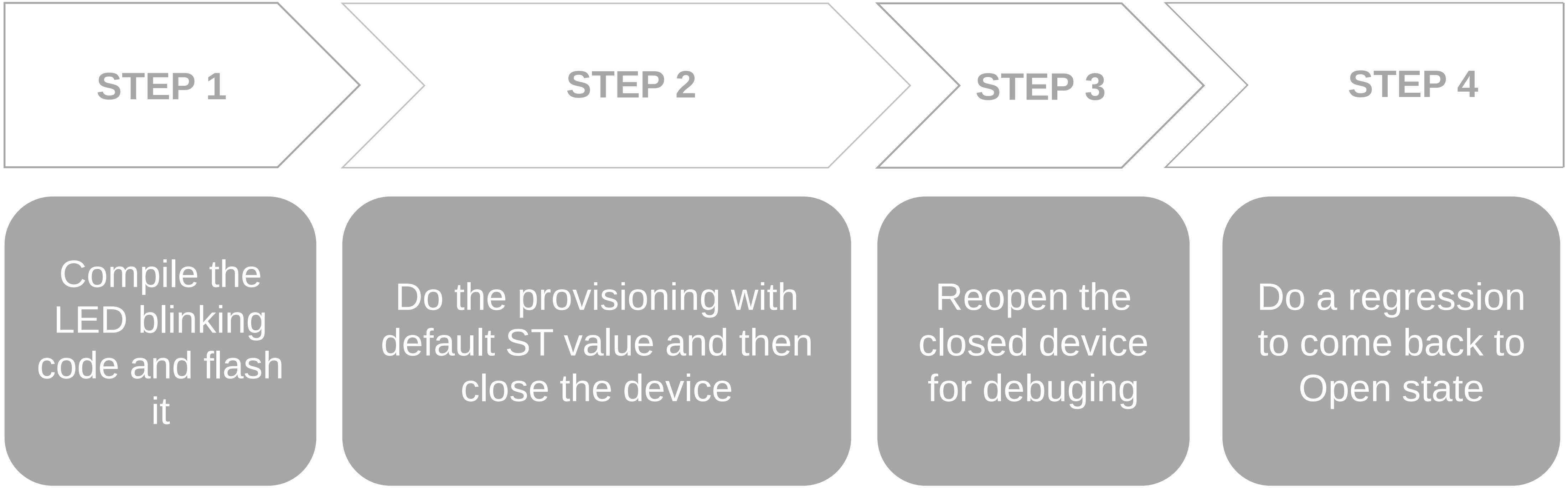
STEP 4

Do a regression to come back to Open state



User flash has been erased
Secure storage has been erased

Key Steps for the Hands-On Exercise



STEP 1

Compile the LED blinking code and flash it

STEP 2

Do the provisioning with default ST value and then close the device

STEP 3

Reopen the closed device for debuging

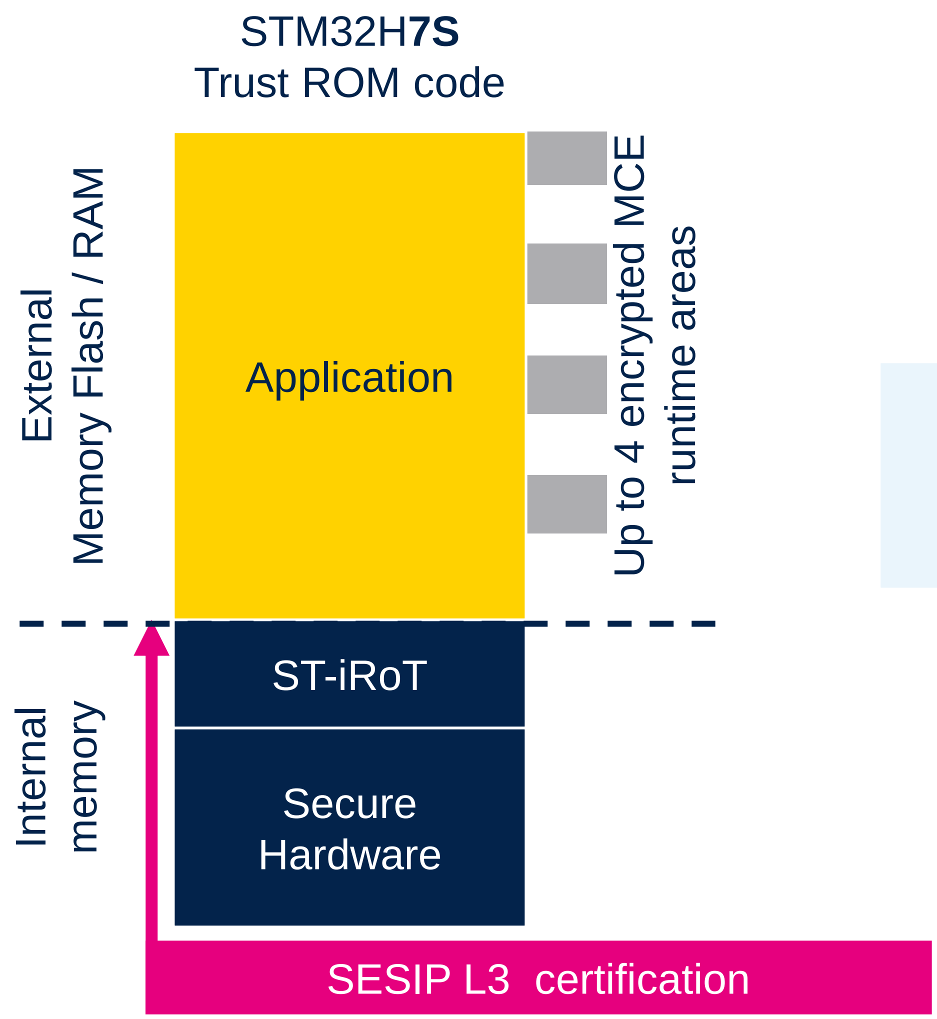
STEP 4

Do a regression to come back to Open state

Take away

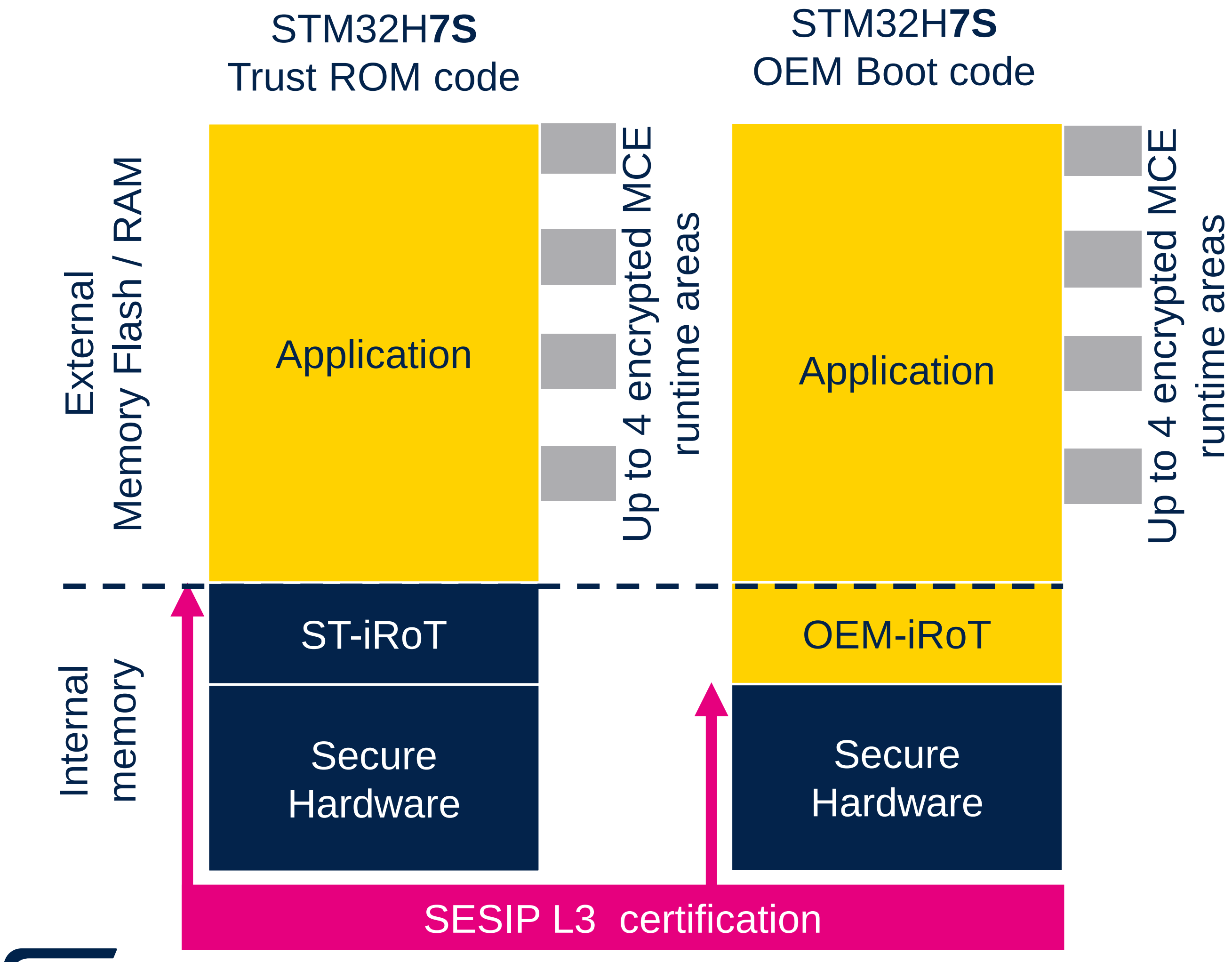
- STM32H7R-S implement Debug Authentication feature to control regression and debug reopening with certificate
- The provisioning adds a step in production
- This new mechanism increases the robustness of the flash protection but allows a reopening of the device and in a secure way.
- STM32H7RS introduced many other security features in STM32H7 family like…

Secure boot with secure firmware update capabilities !



STiROT
A ROMed secure boot with secure update capability

Secure boot with secure firmware update capabilities !

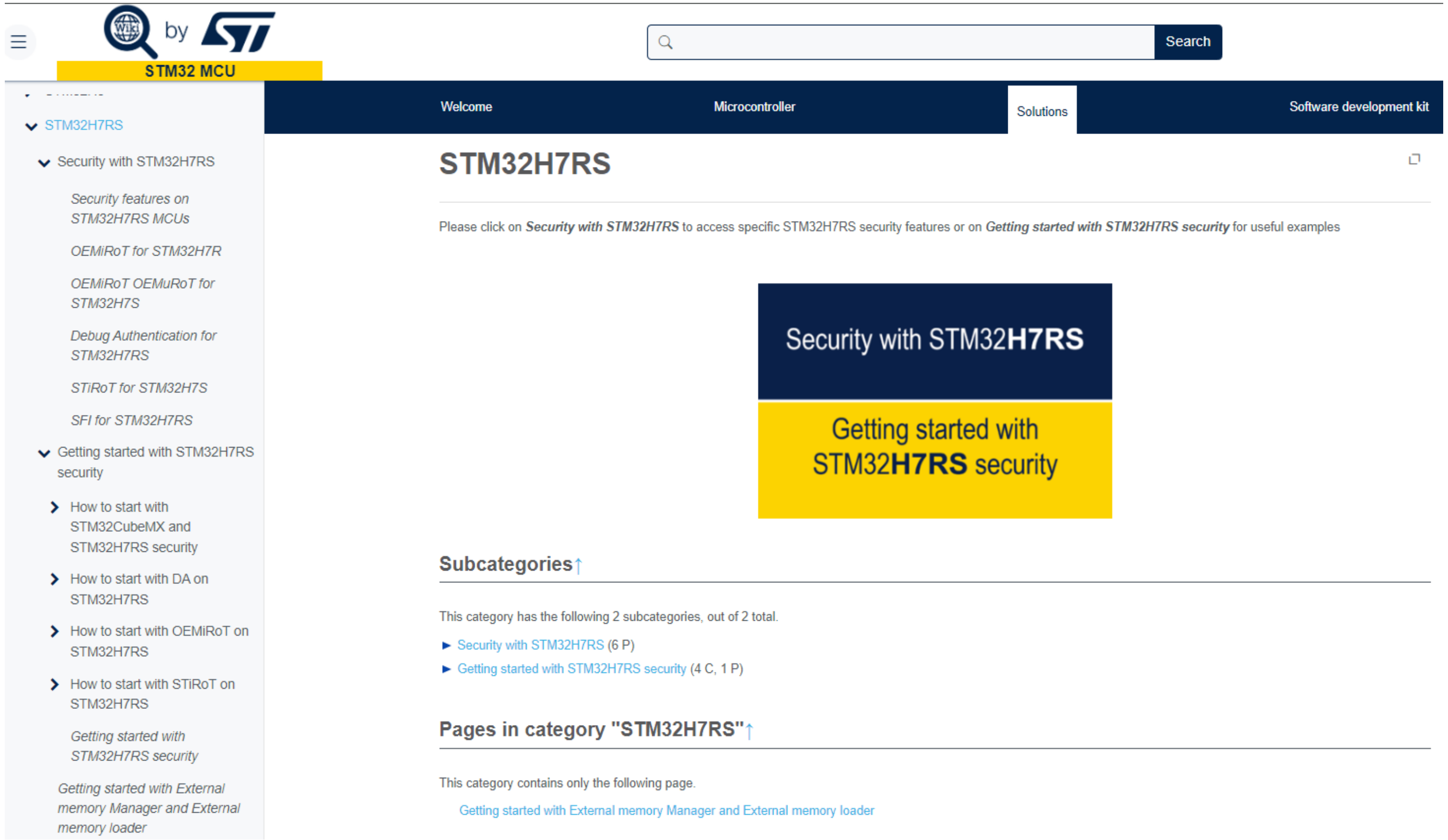


OEMiRoT
Open source code example of secure boot with secure update capabilities

life.augmented

TM32H7RS scalable security !

https://wiki.st.com/stm32mcu/wiki/Category:STM32H7RS



- STM32H7RS
  - Security with STM32H7RS
    - *Security features on STM32H7RS MCUs*
    - *OEMiRoT for STM32H7R*
    - *OEMiRoT OEMuRoT for STM32H7S*
    - *Debug Authentication for STM32H7RS*
    - *STiRoT for STM32H7S*
    - *SFI for STM32H7RS*
  - Getting started with STM32H7RS security
    - How to start with STM32CubeMX and STM32H7RS security
    - How to start with DA on STM32H7RS
    - How to start with OEMiRoT on STM32H7RS
    - How to start with STiRoT on STM32H7RS
    - *Getting started with STM32H7RS security*
  - *Getting started with External memory Manager and External memory loader*

STM32H7RS

Please click on ***Security with STM32H7RS*** to access specific STM32H7RS security features or on ***Getting started with STM32H7RS security*** for useful examples

Security with STM32**H7RS**

Getting started with STM32**H7RS** security

Subcategories↑

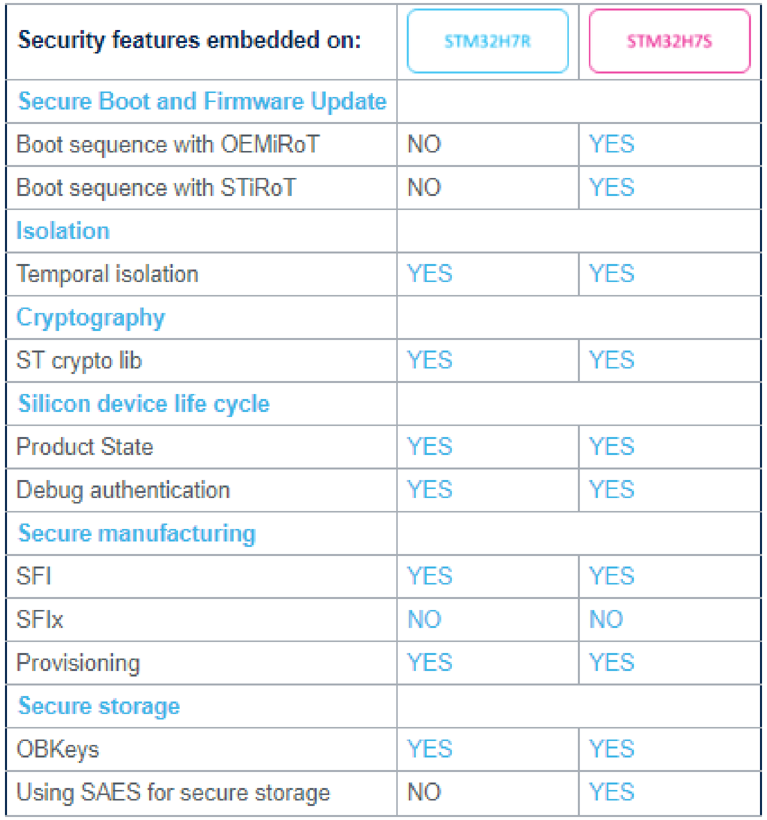
This category has the following 2 subcategories, out of 2 total.

- ► Security with STM32H7RS (6 P)
- ► Getting started with STM32H7RS security (4 C, 1 P)

Pages in category "STM32H7RS"↑

This category contains only the following page.

Getting started with External memory Manager and External memory loader



Security features embedded on:	STM32H7R	STM32H7S
Secure Boot and Firmware Update		
Boot sequence with OEMiRoT	NO	YES
Boot sequence with STiRoT	NO	YES
Isolation		
Temporal isolation	YES	YES
Cryptography		
ST crypto lib	YES	YES
Silicon device life cycle		
Product State	YES	YES
Debug authentication	YES	YES
Secure manufacturing		
SFI	YES	YES
SFIx	NO	NO
Provisioning	YES	YES
Secure storage		
OBKeys	YES	YES
Using SAES for secure storage	NO	YES

STM32H7S targeting SESIP/PSA L3 !

life.augmented

Our technology starts with You

life.augmented