Enabling Infineon SEMPER™ flash with Octal interface on STM32N6 microcontroller

About this document

Scope and purpose

This application note describes how to enable Infineon SEMPER™ NOR flash memories with Octal interface with STM32N6 microcontroller from STMicroelectronics, by showing working examples of hardware connection, boot configuration, programming, and flash driver integration.

Intended audience

This document is intended for hardware and software designers who plan to integrate Infineon S28HL-T/S28HS-T SEMPER™ NOR flash memories with STM32N6 series of devices. Readers are expected to have usage experience of STM32 design tools.

Table of contents

About this document . . . 1
Table of contents . . . 2
1 Introduction . . . 3
1.1 Infineon SEMPER™ NOR flash with Octal interface . . . 3
1.2 External flash memory interface in STMicroelectronics STM32N6 . . . 3
1.3 Hardware and software setup . . . 3
2 Hardware setup . . . 4
2.1 SEMPER™ NOR flash with Octal interface pinout and signal description . . . 4
2.2 Connecting STM32N6 XSPI controller and SEMPER™ NOR flash . . . 5
3 STM32N6 boot configuration for SEMPER™ NOR flash with Octal interface . . . 6
3.1 Boot from SEMPER™ NOR flash with Octal interface . . . 6
3.2 I/O speed optimization . . . 6
4 Flash driver configuration and development for SEMPER™ NOR flash with Octal interface . . . 8
4.1 STM32CubeMX new project . . . 8
4.2 Clock configuration for XSPI controller . . . 9
4.3 XSPI configuration for SEMPER™ NOR flash with Octal interface . . . 11
4.4 Middleware packs configuration for user memory driver . . . 13
4.5 Import STM32 project into STM32CubeIDE . . . 15
4.6 Flash driver development for SEMPER™ NOR flash with Octal interface . . . 18
5 Programming to SEMPER™ NOR flash with Octal interface . . . 23
6 Summary . . . 26
References . . . 27
Revision history . . . 28
Disclaimer . . . 29

1 Introduction

The STM32N6 microcontroller series is a high-performance embedded processing platform designed to address demanding next-generation Edge AI, computer vision, graphics, and real-time control requirements.

This document describes the implementation of a 512 Mb density S28HS512T flash memory. It can be integrated with memory devices of both smaller and larger densities with only minor adjustments.

1.1 Infineon SEMPER™ NOR flash with Octal interface

Infineon S28HL-T/S28HS-T SEMPER™ family is serial NOR flash memory with Octal interface that delivers the performance required for STM32N6-based embedded systems. The STM32N6 supports high-speed XSPI external memory interfaces for serial NOR flash memories, enabling fast access to external non-volatile memory for boot code, application firmware, AI models, graphics assets, and data storage. The key features of S28HL-T/S28HS-T SEMPER™ NOR flash are:

- x8 interface operating up to 200 MHz DDR (400 MBps)
- JEDEC xSPI compliant interface
- Automotive qualified options up to AEC-Q100 Grade 1 (–40°C to +125°C)
- ASIL-D certified
- 256 Mb to 2 Gb densities

1.2 External flash memory interface in STMicroelectronics STM32N6

The STM32N6 family of devices is a high-performance MCU platform based on the Arm® Cortex®-M55 processor, targeting advanced embedded applications, such as Edge AI, computer vision, industrial automation, smart devices, and automotive sensing.

STM32N6 does not have internal, on-chip flash memory for application storage. Therefore, external non-volatile memory is essential for boot code, application firmware, AI models, graphics assets, and other persistent data. STM32N6 provides high-speed external memory interfaces, including XSPI, to interface external serial NOR flash memory. The key features of the XSPI controller are:

- x1 (single), x2 (dual), x4 (quad), x8 (octal), and x16 serial memory interface configurations
- Support for serial NOR flash memories and memory-mapped access
- Single Transfer Rate (STR) and Double Transfer Rate (DTR) operation modes
- Up to 200 MHz serial clock speed

For an in-depth description of the Octo-SPI, Hexadeca-SPI, and XSPI interfaces, see AN5050 from STMicroelectronics.

1.3 Hardware and software setup

The following hardware and software are used for the application in this document.

- Discovery kit with STM32N657X0 MCU (STM32N6570-DK) with flash memory replaced
- Infineon S28HS512T SEMPER™ NOR flash
- STM32CubeIDE 2.1.1 along with STM32CubeMX v6.17.0 and STM32CubeProgrammer v2.22.0

2 Hardware setup

2.1 SEMPER™ NOR flash with Octal interface pinout and signal description

SEMPER™ devices are provided in ball grid array (BGA), 1 mm pitch, 24-ball, 5x5 ball array footprint (Figure 1). The footprint and pin assignment are compatible with Octal SPI flash memories from different vendors. Table 1 describes input/output signals typically connected to STM32N6 device.

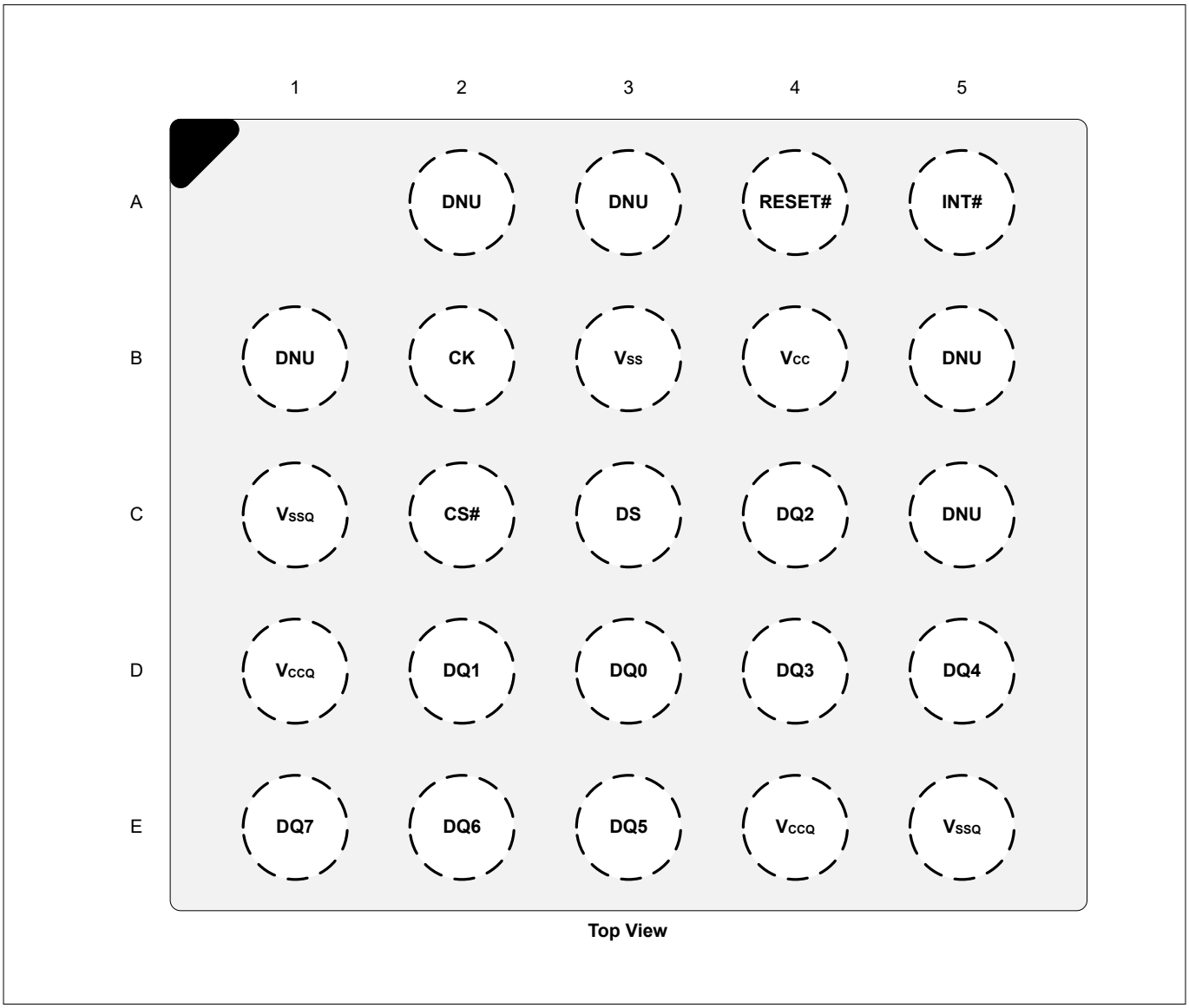

Figure 1 24-ball BGA pinout configuration

Table 1 Input/output signal description

Symbol	Type	Description
CS#	Input	**Chip Select:** Device is selected and in active state when CS# is $\overline{\text{LOW}}$
CK	Input	**Clock:** Provides the timing of the serial interface

(table continues...)

Table 1 (continued) Input/output signal description

Symbol	Type	Description
DS	Output	**Data Strobe:** Used for data read operations only and indicates that output data is valid
DQ[7..0]	Input/output	**Serial Data:** Bidirectional signals that transfer command, address, and data information
RESET#	Input	**Hardware Reset:** Resets device to default state

2.2 Connecting STM32N6 XSPI controller and SEMPER™ NOR flash

The STM32N6 XSPI controller has two independent ports (1 and 2) and each port has two chip selects. Figure 2 is an example of a connection between STM32N657X0 and S28HS512T SEMPER™ NOR flash. In this example, XSPI port 2, Chip Select 1 (NCS1) is used.

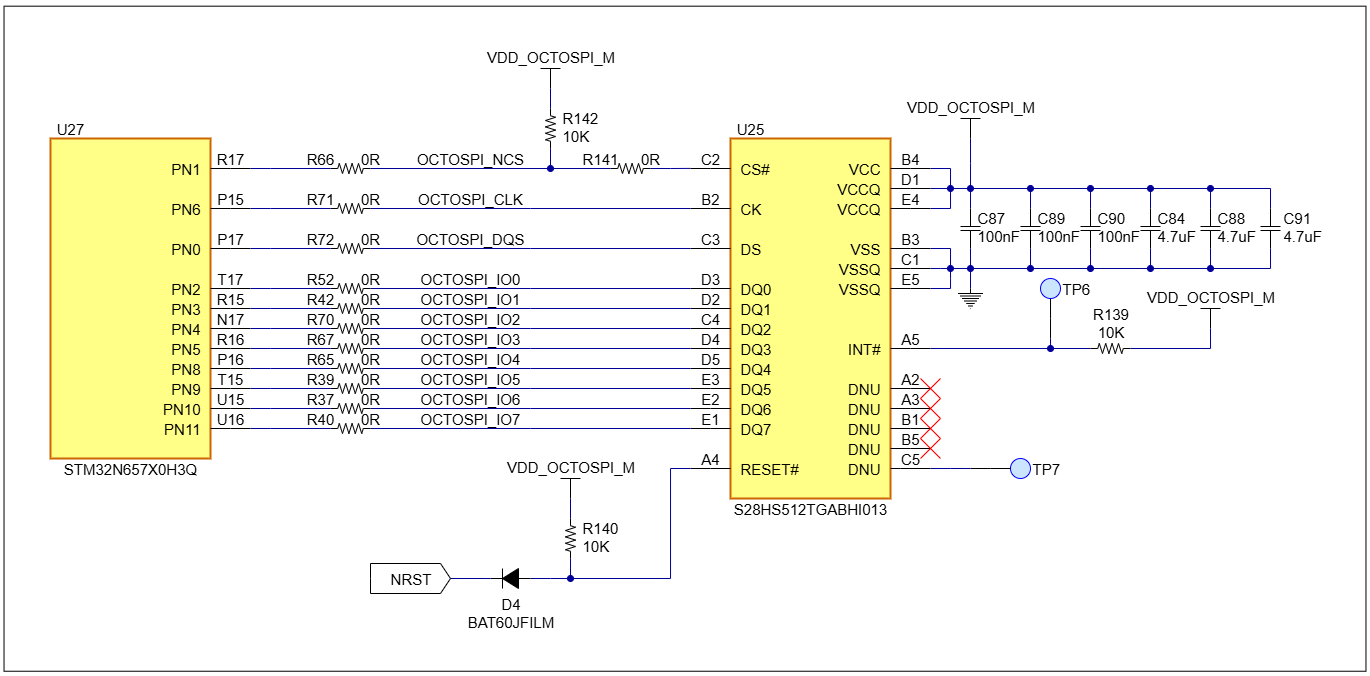

Figure 2 Connection between STM32N6 and S28HS512T SEMPER™ NOR flash

3 STM32N6 boot configuration for SEMPER™ NOR flash with Octal interface

STM32N6 has BootROM firmware and it loads the First Stage Boot Loader (FSBL) based on options specified through the one-time programmable (OTP) fuse bits. BootROM supports booting from external flash memory including xSPI NOR flash.

This section describes the configuration required to boot from SEMPER™ NOR flash with Octal interface and boot performance optimization by maximizing read throughput of SEMPER™ NOR flash.

For details on how to program the OTP fuse bits, as well as on boot configuration and I/O speed optimization, see How to program the OTP fuse bits in the STM32N6, AN5967, and RM0486.

3.1 Boot from SEMPER™ NOR flash with Octal interface

The boot interface and boot device are determined by the BOOT0 pin, BOOT1 pin, and OTP fuses (BOOTROM_CONFIG_n). During the application development phase, the boot mode can be selected using the BOOT0 and BOOT1 pins (SW2 and SW1 on the STM32N6570-DK). Table 2 shows the BOOT0 and BOOT1 settings for each boot mode, along with the corresponding switch settings on the STM32N6570-DK. Development boot mode is used to load firmware directly into SRAM or to program firmware into external flash memory. In this mode, the program is lost when the board is powered off because the STM32N6 microcontroller series does not have internal flash memory. Flash boot mode is used to boot firmware from external flash memory.

Table 2 Boot modes

BOOT0 (SW2)	BOOT1 (SW1)	Boot source
–	H	Development boot
L	L	Flash boot

The boot device can be changed by programming BOOTROM_CONFIG_2 (OTP11), but only once. In this case, there is no need to program the OTP fuse because if the boot source is not programmed in the OTP fuse, the default boot interface is single SPI.

Table 3 OTP fuse configuration for SEMPER™ NOR flash with Octal interface

OTP word	OTP bits	Settings for SEMPER™ NOR flash with Octal interface
OTP11 (BOOTROM_CONFIG_2)	[8:5]	0 (default) or 3: XSPI serial NOR (single SPI)

3.2 I/O speed optimization

To achieve the high-speed operation above 100 MHz with external flash memory, supplying 1.8 V to VDDIOx and the external flash memory is not sufficient. OTP124 must be programmed and additional firmware code must be added to leverage the firmware setting. On the STM32N6570-DK, VDDIO3 is used for the NOR flash interface. Therefore, OTP124[15] (HSLV_VDDIO3) must be programmed. Table 4 shows the HSLV_VDDIO3 setting required for high-speed operation.

Table 4 OTP fuse configuration for I/O speed optimization

OTP word	OTP bits	OTP bits name	Settings for high-speed frequency
OTP124 (HCONF1)	[15]	HSLV_VDDIO3	1

After programming the OTP fuse, ensure that the firmware applies the updated OTP setting. To perform, add the following function calls:

```
__HAL_RCC_PWR_CLK_ENABLE();
HAL_PWREx_EnableVddIO3();
HAL_PWREx_ConfigVddIORange(PWR_VDDIO3, PWR_VDDIO_RANGE_1V8);
```

For details on how to program the OTP fuse bits, as well as on boot configuration and I/O speed optimization, see How to program the OTP fuse bits in the STM32N6, AN5967, and RM0486.

4 Flash driver configuration and development for SEMPER™ NOR flash with Octal interface

STMicroelectronics provides the middleware packs for external memories, such as the External Memory Loader and External Memory Manager. These packs simplify the integration of external memories with the STM32N6 microcontroller. However, support for S28HL-T/S28HS-T SEMPER™ NOR flash requires certain modifications to the middleware. The following sections explain how to configure the XSPI controller and the middleware, and how to modify the middleware for S28HL-T/S28HS-T SEMPER™ NOR flash. The procedure is described as a workflow, from creating a new project to building the project.

4.1 STM32CubeMX new project

1. Install STM32CubeMX software
2. In STM32CubeMX software, select **File** > **New Project**

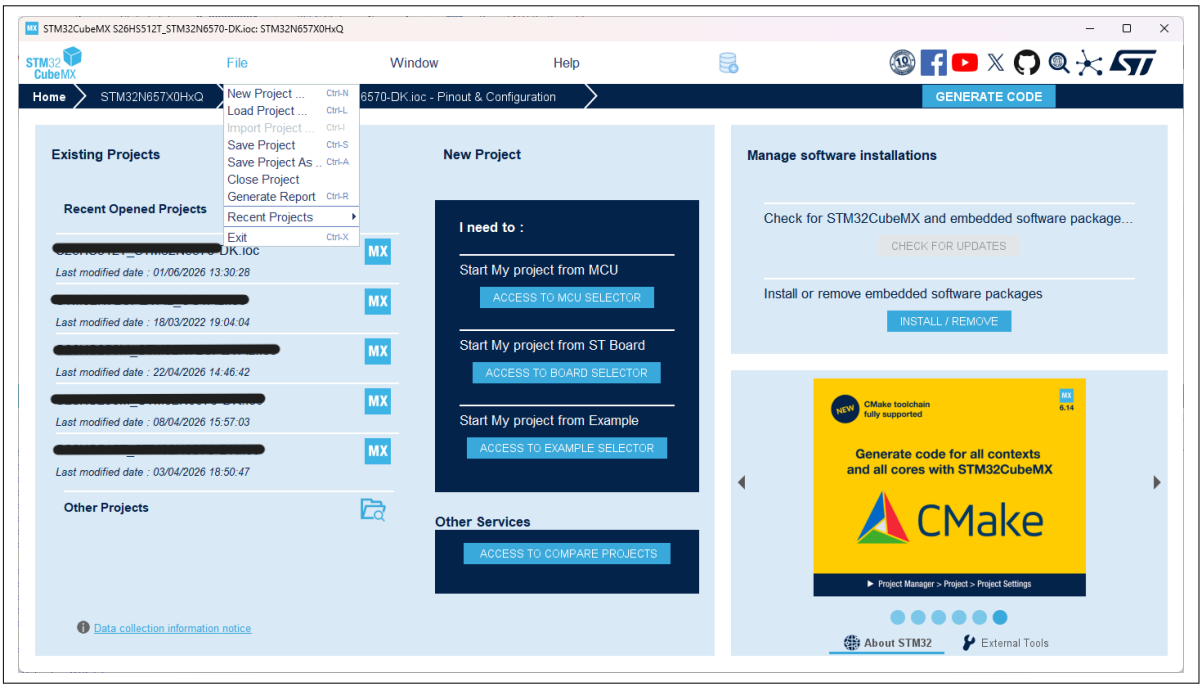

Figure 3 Create new STM32CubeMX project

3. In MCU/MPU Selector, fill out Commercial Part Number with STM32N657X0, select the MCU, and then click **Start Project**

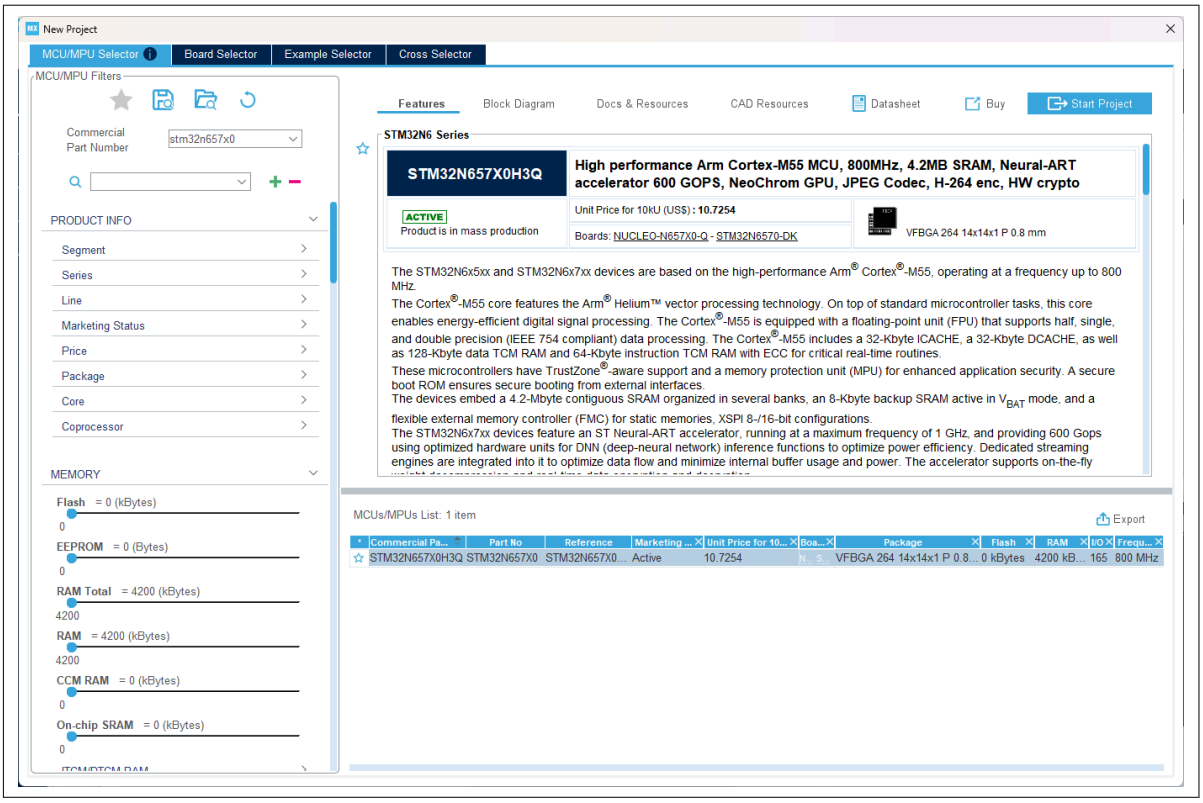

Figure 4 MCU/MPU Selector dialog

4.2 Clock configuration for XSPI controller

Use the clock configuration tool in STM32CubeMX to configure the clock for the external flash. Select the **Clock Configuration** tab at the top of the window.

To operate XSPI controller at 200 MHz, set the **AXI Clocks** to 200 MHz or higher and the **AHB Peripheral Clocks** to 200 MHz. These clocks can be adjusted by modifying the dividers and changing the Clock MUX output. Figure 5 and Figure 6 show example clock settings for generating a 200 MHz clock for the S28HS-T SEMPER™ NOR flash. The settings for PLL1 to PLL4 are the same.

To use an external crystal or another clock source, enable the High Speed Clock (HSE) in the **Pinout & Configuration** tab.

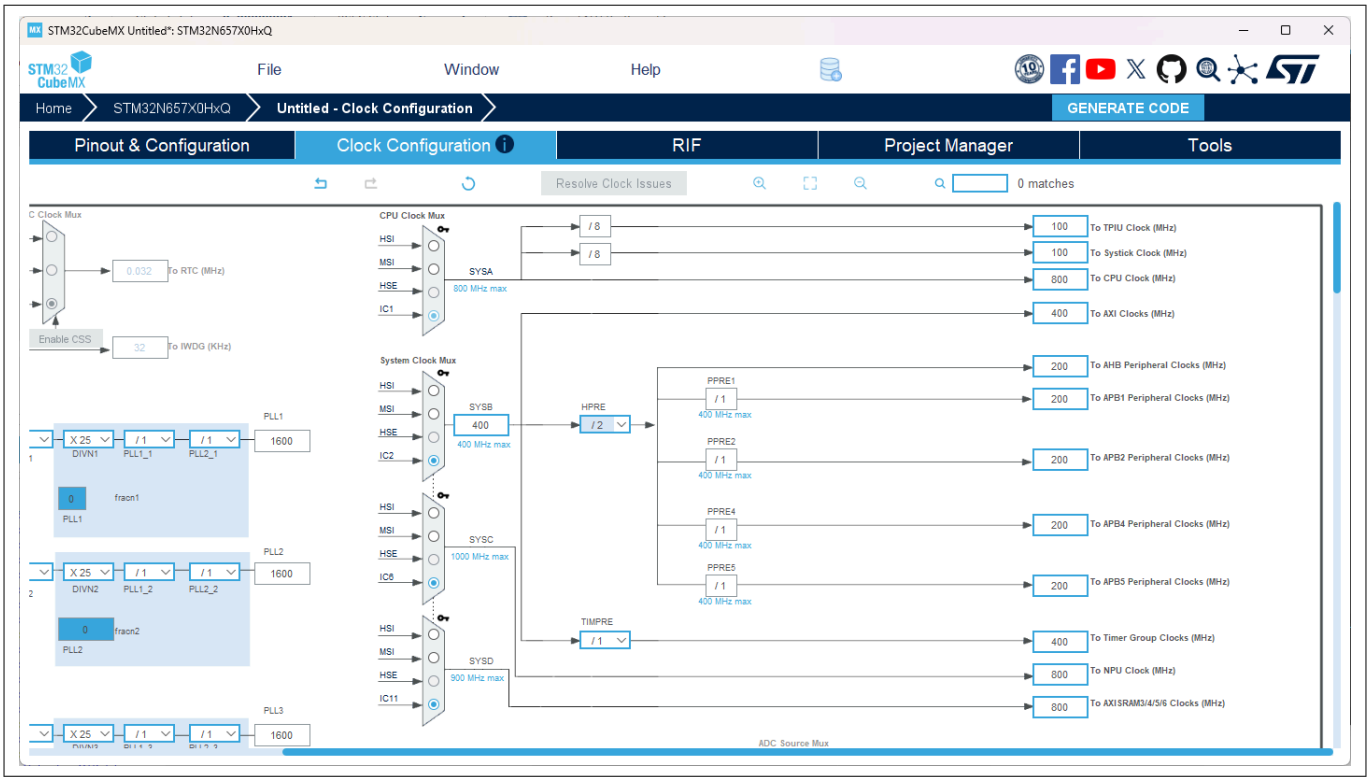

Figure 5 **Clock configuration for AXI and AHB peripheral clocks**

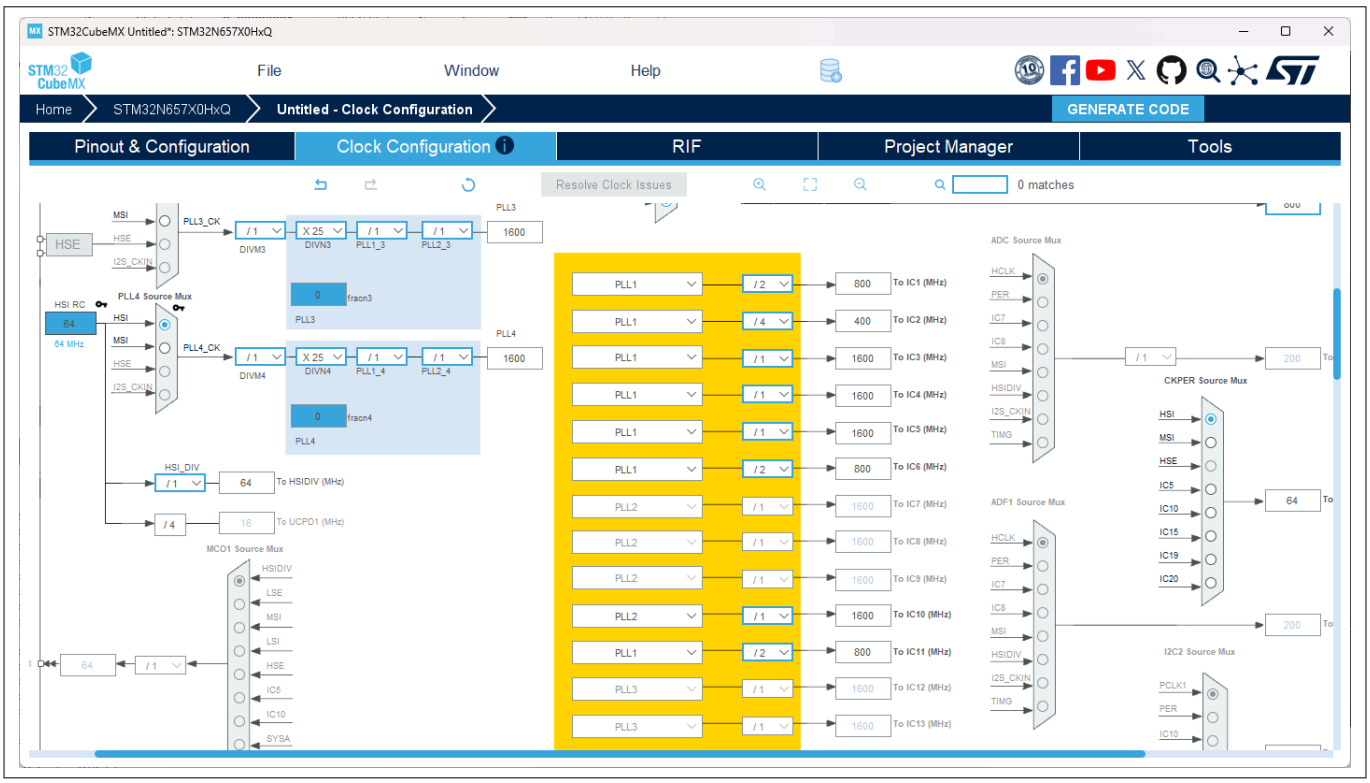

Figure 6 **Clock configuration for ICn**

4.3 XSPI configuration for SEMPER™ NOR flash with Octal interface

1. In STM32CubeMX, select the **Pinout & Configuration** tab, click **Connectivity**, and then click **XSPIM**. The XSPIM configuration window displays as shown in Figure 7. To use XSPI in the FSBL and External Memory Loader, select First Stage Boot Loader, External Memory Loader, and Direct mode. If you use XSPIM in Swapped mode or another mode, replace "XSPI2" with the appropriate term in the following sections.

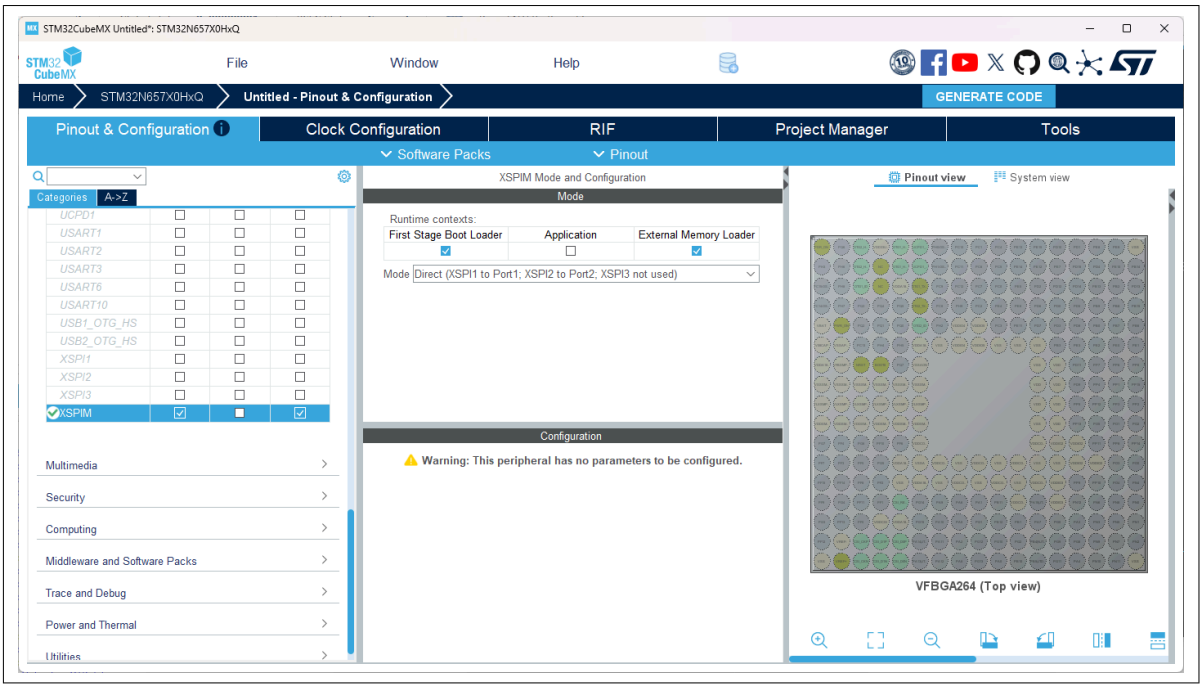

Figure 7 XSPIM configuration window

2. Select **XSPI2** as shown in Figure 8. To use XSPI2 in the FSBL and External Memory Loader, select First Stage Boot Loader and External Memory Loader. Configure the parameters as required

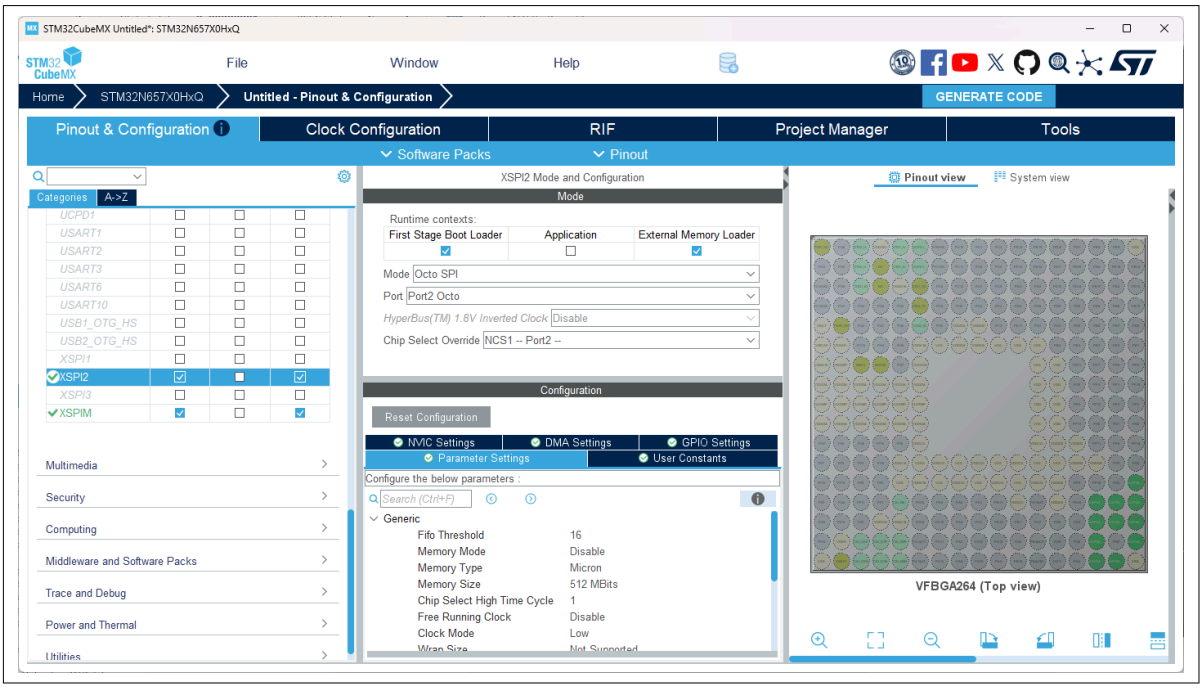

Figure 8 XSPI2 configuration window

Table 5 Key parameters in XSPI2 configuration

Parameter	Value
Mode	Octo SPI
Port	Port2 Octo
Chip Select Override	NCS1 -- Port2 --
Fifo Threshold	32
Memory Mode	Disable
Memory Type	Micron
Memory Size	512 MBits
Chip Select High Time Cycle	1
Free Running Clock	Disable
Clock Mode	Low
Wrap Size	Not Supported
Clock Prescaler	3
Sample Shifting	None
Delay Hold Quarter Cycle	Enable
Chip Select Boundary	Disabled
Maximum Transfer	0
Refresh Rate	0
Memory Select	NCS1
Switching Duration Clock Number	1

After the XSPI configuration, navigate to Clock Configuration to ensure that XSPI2 clock is 200 MHz as shown in Figure 9.

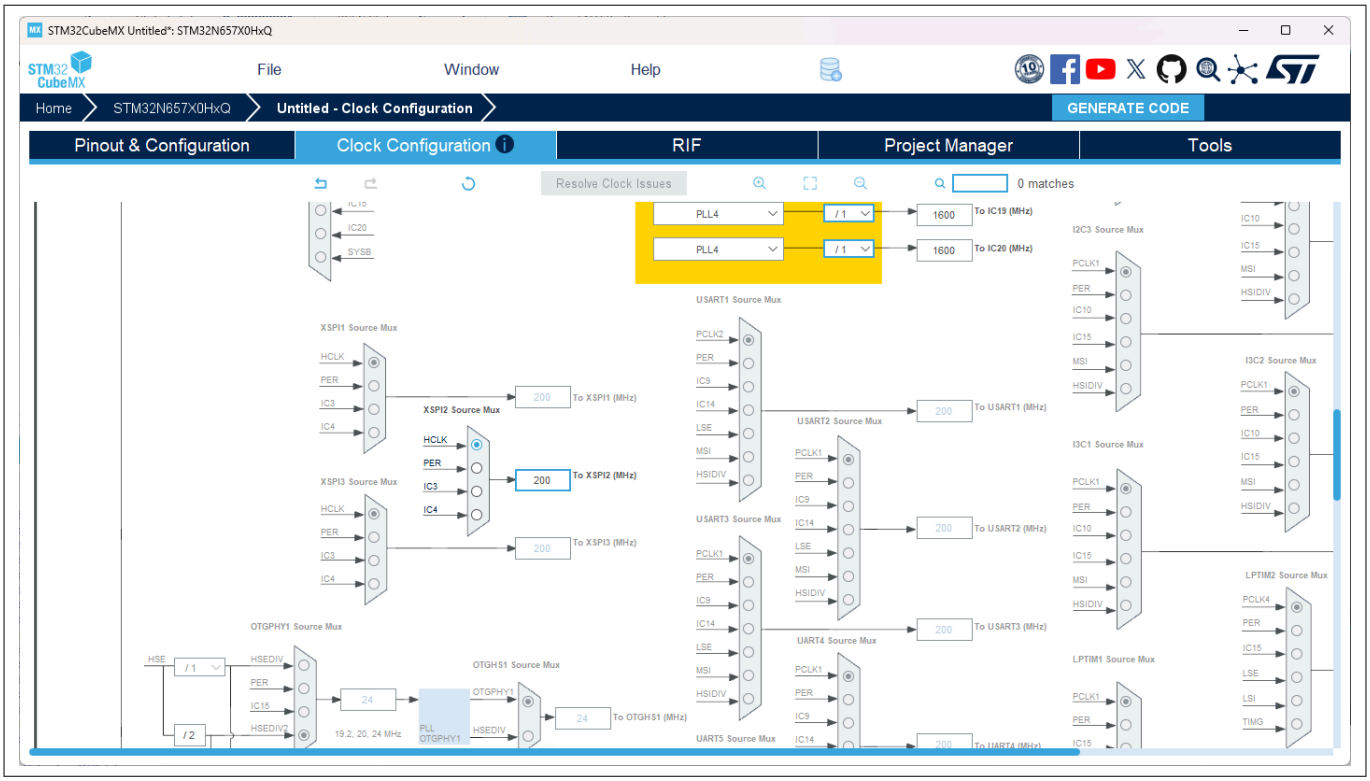

Figure 9 Clock configuration for XSPI2

4.4 Middleware packs configuration for user memory driver

1. In the **Pinout & Configuration** tab, click **Middleware and Software Packs**, and then select EXTMEM_MANAGER. EXTMEM_MANAGER configuration window displays as shown in Figure 10. Select FSBL and Activate External Memory Manager. In the **Boot usecase** tab, Select boot code generation and Configure the parameters for Boot and LRUN source. Table 6 describes the key parameters that must be configured to execute in the internal RAM. In the **Memory 1** tab, select EXTMEM_USER for Select the memory driver

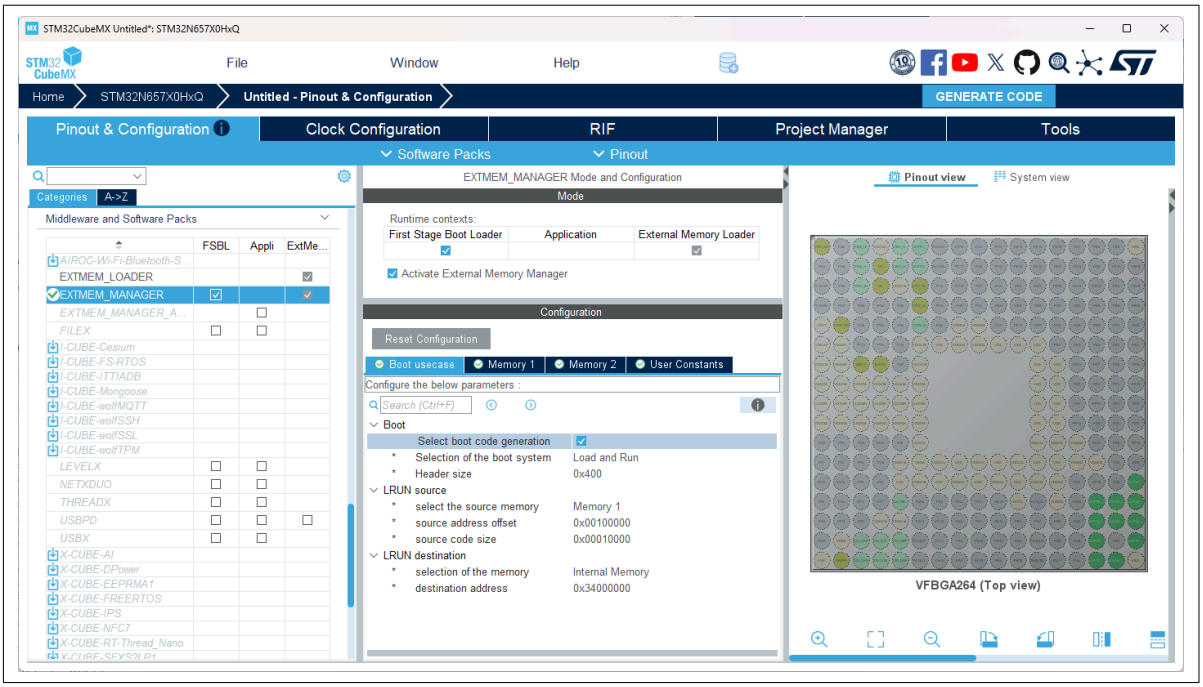

Figure 10 EXTMEM_MANAGER configuration window

Table 6 Key parameters in EXTMEM_MANAGER

Parameter	Value
Selection of the boot system	Load and Run
Header size	0x400
select the source memory	Memory 1
source address offset	0x00100000
source code size[1)]	0x00010000

2. Select **EXTMEM_LOADER** and **Activate External Memory Loader** parameters displays as shown in Figure 11. Configure the parameters as required. Table 7 describes the key parameters for EXTMEM_LOADER configurations

1 Depends on application code size.

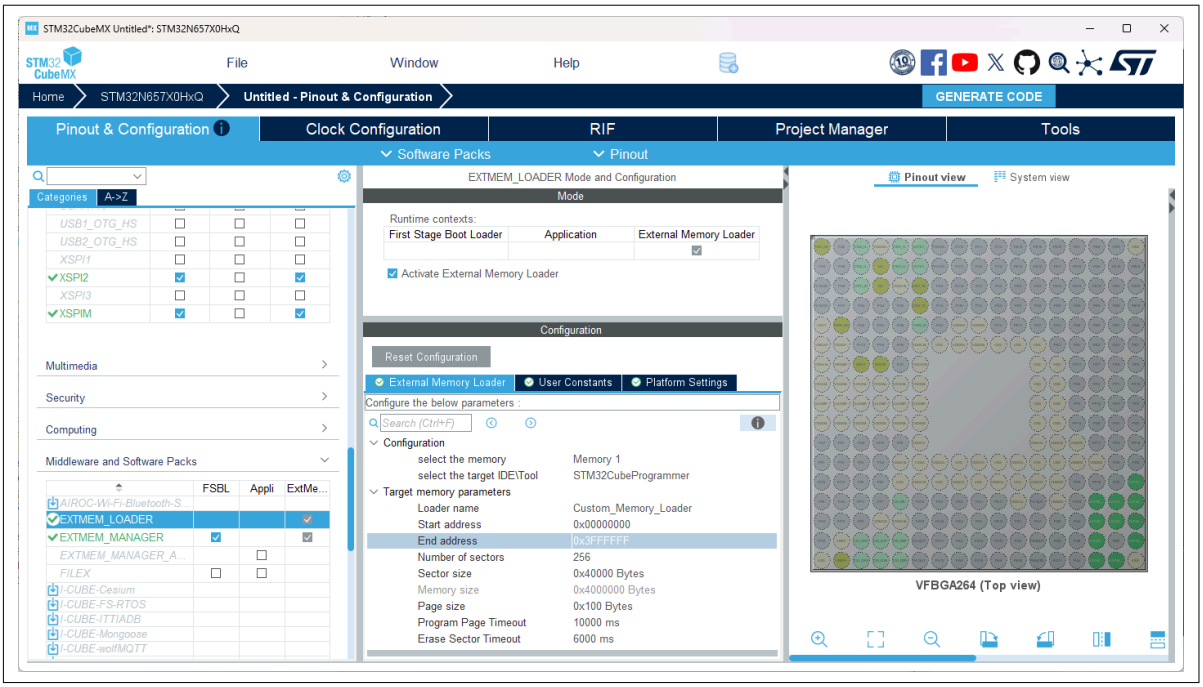

Figure 11 EXTMEM_LOADER configuration window

Table 7 Key parameters in EXTMEM_LOADER

Parameter	Value
select the memory	Memory 1
select the target IDE/Tool	STM32CubeProgrammer
Start address	0x70000000
End address	0x73FFFFFF
Number of sectors	256
Sector size	0x40000 Bytes
Page size	0x100 Bytes

4.5 Import STM32 project into STM32CubeIDE

1. In the **Project Manager** tab of STM32CubeMX, enter Project Name, select FSBL, Appli, and ExtMemLoader, and select STM32CubeIDE for Toolchain/IDE. Select **Code Generator** tab at the left of the window, and select for Generate peripheral initialization as a pair of '`.c/.h`' files per peripheral. Click **GENERATE CODE** at the right top of the window

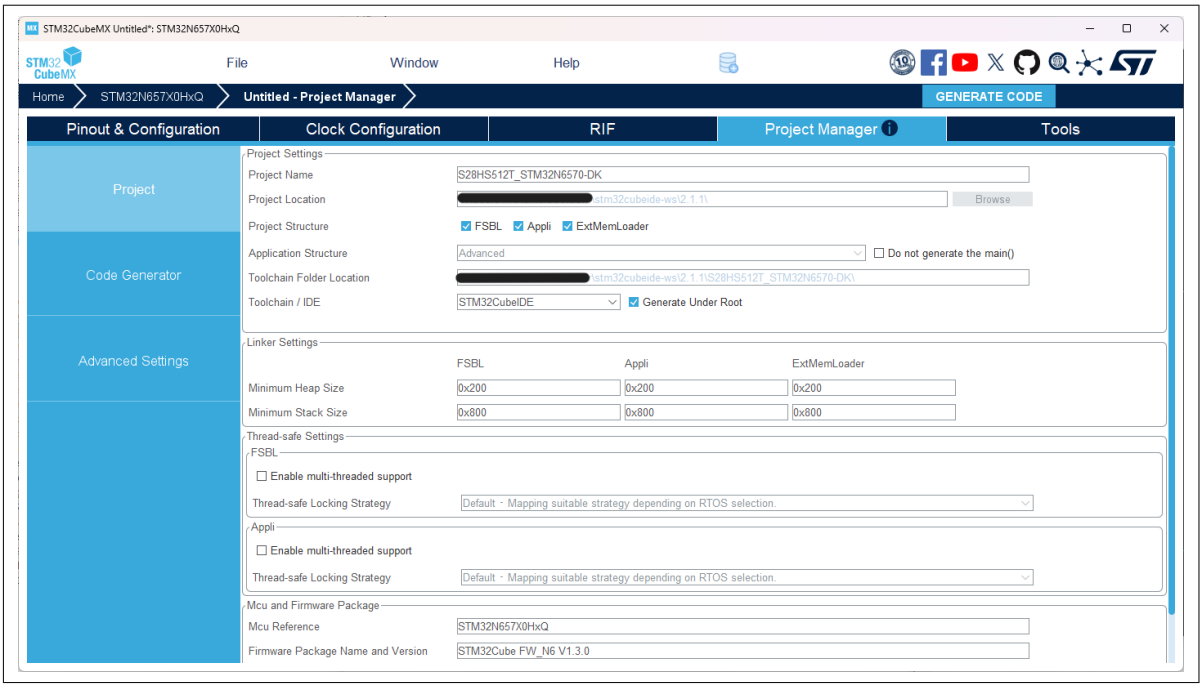

Figure 12 Project tab in Project Manager tab

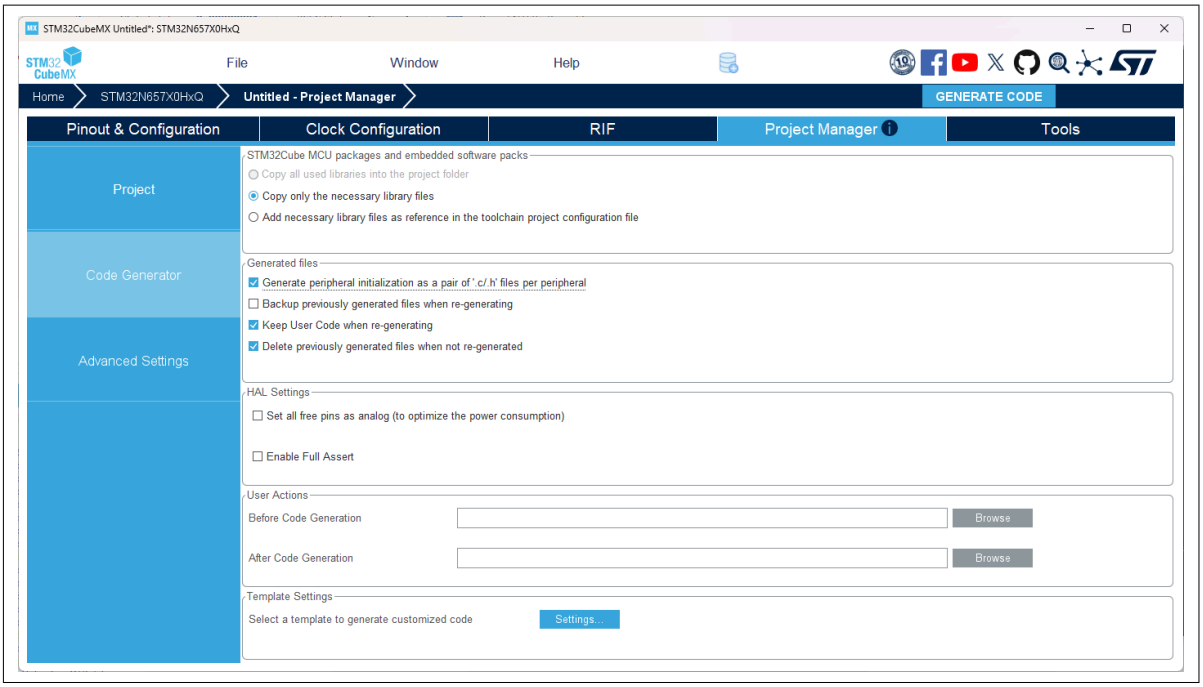

Figure 13 Code Generator tab in Project Manager tab

2. After the code generation, launch STM32CubeIDE at the workspace where the generated project directory exists. In STM32CubeIDE, select **File** > **STM32 Project Create/Import**, it displays the **Create / Import STM32 Project** window. Select STM32CubeMX/STM32CubeIDE Project under Import STM32 Project and click **Next**. In **Import Projects from File System or Archive** window, click **Directory...** to select Import source. Select all items under **Folder** and click **Finish** to import the project

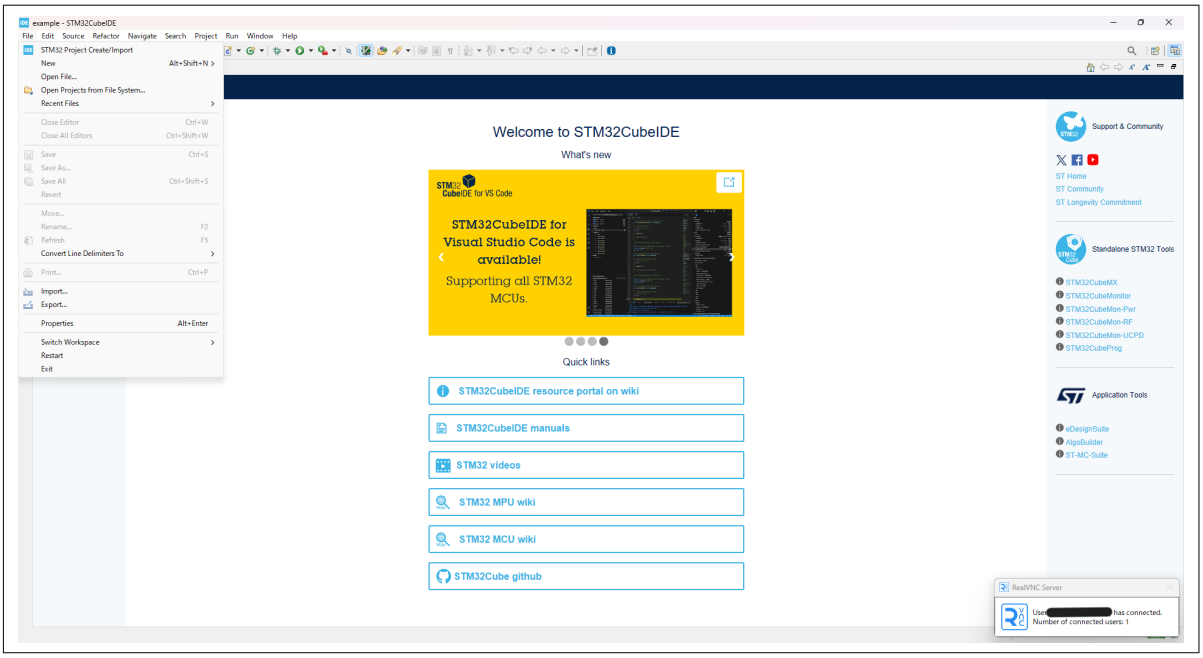

Figure 14 Import STM32 project into STM32CubeIDE

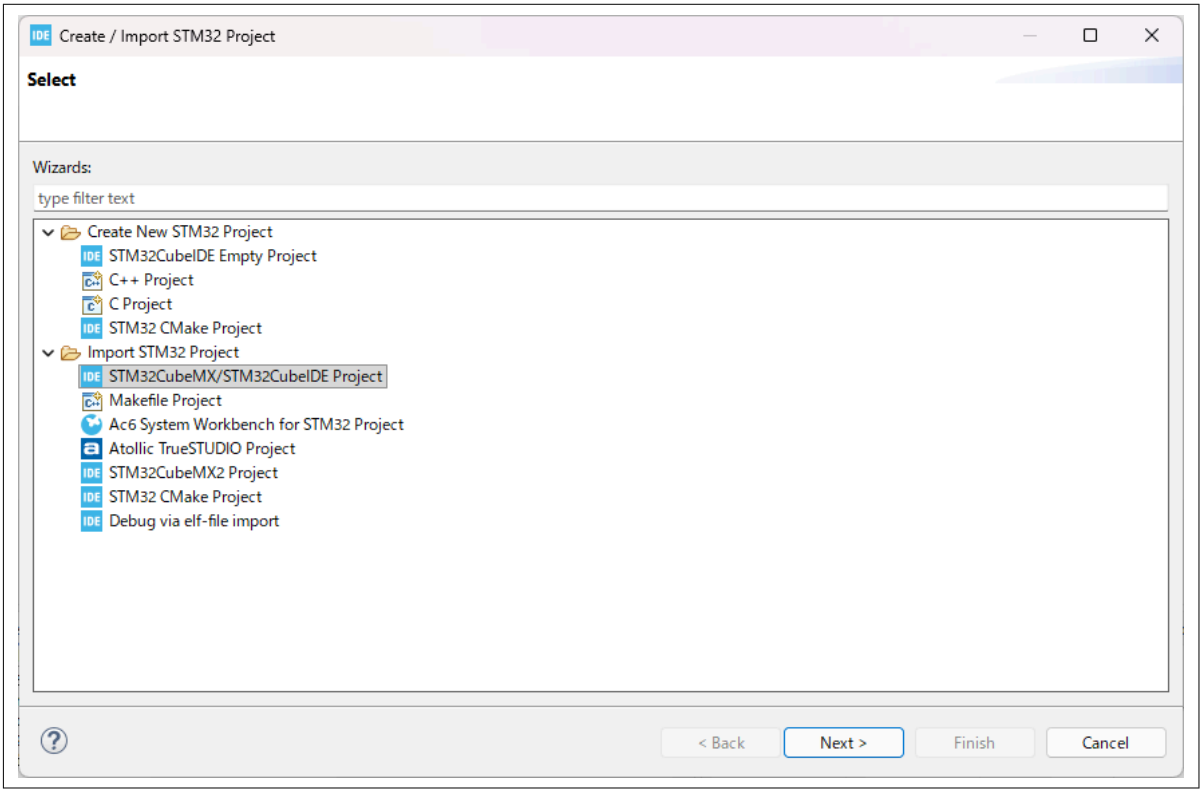

Figure 15 Create/Import STM32 Project window

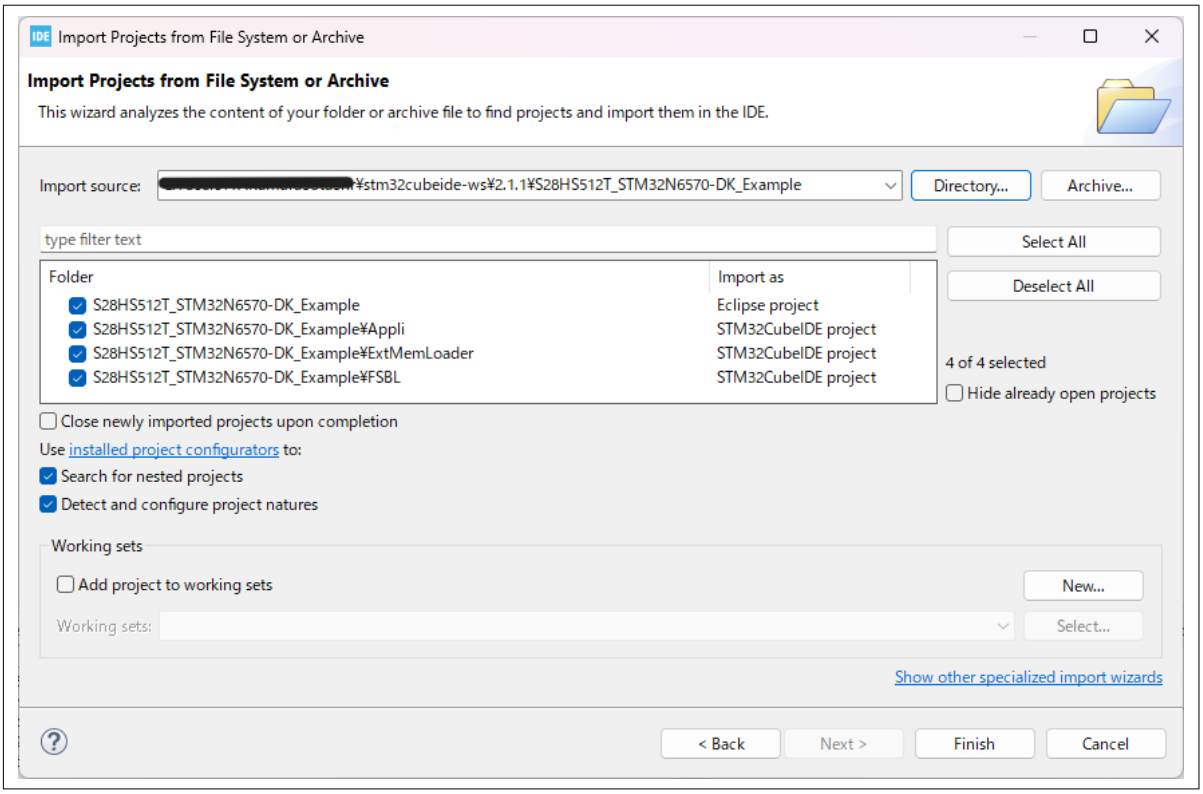

Figure 16 Import Projects from File System or Archive window

4.6 Flash driver development for SEMPER™ NOR flash with Octal interface

EXTMEM_MANAGER offers unified APIs for external memories, and EXTMEM_LOADER provides a loader that initializes, erases, writes, and verifies external memories from STM32CubeProgrammer.

This section describes how to implement a USER driver for SEMPER™ NOR flash with Octal interface using the EXTMEM_MANAGER APIs, and the modifications in the EXTMEM_LOADER. For details on command sequences and register settings, see the modified BSP layer in the example project.

1. EXTMEM_MANAGER: USER driver updates

The USER driver in EXTMEM_MANAGER provides only function prototypes. In the default implementation, these functions are declared with __weak and return EXTMEM_DRIVER_USER_NOTSUPPORTED. Therefore, support for S28HL-T/S28HS-T devices requires you to implement the required APIs by overriding these weak definitions. Implement the following functions:

```
EXTMEM_DRIVER_USER_Init()
EXTMEM_DRIVER_USER_Read()
EXTMEM_DRIVER_USER_Write()
EXTMEM_DRIVER_USER_EraseSector()
EXTMEM_DRIVER_USER_MassErase()
EXTMEM_DRIVER_USER_Enable_MemoryMappedMode()
EXTMEM_DRIVER_USER_Disable_MemoryMappedMode()
EXTMEM_DRIVER_USER_GetMapAddress()
```

To operate at 200 MHz, it is necessary to configure the configuration registers of the memory and the prescaler of the XSPI controller. Table 8 shows the configuration register values used in the SEMPER™

S28HS512T example project. For details on the configuration registers of S28HL-T/S28HS-T, see the datasheet. The prescaler must be set to '0' by using `HAL_XSPI_SetClockPrescaler()` after configuring the memory.

Table 8 Configuration register values to operate at 200 MHz

Bit number	Name	Value (binary)
CFR2x[3:0]	MEMLAT[3:0]	1010
CFR3x[7:6]	VRGLAT[1:0]	10
CFR5x[0]	OPI-IT	1
CFR5x[1]	SDRDDR	1

For example implementation, see the SEMPER™ S28HS512T example project and `Middlewares/S28HS512T/ExtMem_Manager/stm32_s28hs512t_driver.c` in that project.

2. `EXTMEM_LOADER`: Required API updates

The External Loader requires entry-point implementation for `Init()`, `Write()`, `MassErase()`, `SectorErase()`, and `Verify()`. STM32CubeMX generates templates for these functions in `Middlewares/ST/STM32_ExtMem_Loader/STM32Cube/stm32_loader_api.c`, and in most cases these templates can be used as-is. However, if the project uses the Bottom hybrid sector configuration, `SectorErase()` must be adapted so that all erase regions are handled correctly. SEMPER™ NOR flash supports both uniform and hybrid sector architectures, and the hybrid configuration can place the small sectors at the top or bottom of the address space depending on the selected memory configuration. See the SEMPER™ NOR flash datasheet on the Infineon website for details on the memory array organization and the address map. A practical approach is to override wrapper functions in `Middlewares/ST/STM32_ExtMem_Loader/core/memory_wrapper.c` because these are declared with `__weak`. In the example project, the

`memory_sectorerase()` wrapper function is overridden to support the Bottom hybrid sector configuration. The following code shows an example implementation:

```
#define STM32EXTLOADER_DEVICE_4K_SECTOR_SIZE   0x1000
#define    STM32EXTLOADER_DEVICE_128K_SECTOR_SIZE 0x20000
#define    STM32EXTLOADER_DEVICE_256K_SECTOR_SIZE 0x40000

/**
  * @brief   Sector erase.
  * @param   EraseStartAddress :  erase start address
  * @param   EraseEndAddress   :  erase end address
  * @param   SectorSize        :  sector size
  * @retval  Operation status
  */
MEM_STATUS memory_sectorerase(uint32_t EraseStartAddress, uint32_t EraseEndAddress,
uint32_t SectorSize)
{
  EraseStartAddress = EraseStartAddress & 0x03FFFFFFUL;
  EraseEndAddress = EraseEndAddress & 0x03FFFFFFUL;
  SectorSize = (EraseStartAddress < STM32EXTLOADER_DEVICE_128K_SECTOR_SIZE)
            ? STM32EXTLOADER_DEVICE_4K_SECTOR_SIZE
            : (EraseStartAddress < STM32EXTLOADER_DEVICE_256K_SECTOR_SIZE)
               ? STM32EXTLOADER_DEVICE_128K_SECTOR_SIZE
               : STM32EXTLOADER_DEVICE_256K_SECTOR_SIZE;
  uint32_t currentStartAddr = (SectorSize == STM32EXTLOADER_DEVICE_4K_SECTOR_SIZE)
                     ? EraseStartAddress & 0x1F000UL
                     : (SectorSize == STM32EXTLOADER_DEVICE_128K_SECTOR_SIZE)
                ? EraseStartAddress & 0x20000UL
                : EraseStartAddress & 0x3FC0000UL;
  uint32_t currentEndAddr = (EraseEndAddress < STM32EXTLOADER_DEVICE_128K_SECTOR_SIZE)
           ? (EraseEndAddress & 0x3FFF000UL) + STM32EXTLOADER_DEVICE_4K_SECTOR_SIZE
           : (EraseEndAddress < STM32EXTLOADER_DEVICE_256K_SECTOR_SIZE)
        ? (EraseEndAddress & 0x3FFF000UL) + STM32EXTLOADER_DEVICE_128K_SECTOR_SIZE
    : (EraseEndAddress & 0x3FFF000UL) + STM32EXTLOADER_DEVICE_256K_SECTOR_SIZE;
  MEM_STATUS retr = MEM_OK;

  do
  {
    if (EXTMEM_EraseSector(STM32EXTLOADER_DEVICE_MEMORY_ID, currentStartAddr,
SectorSize) != EXTMEM_OK)
    {
      retr = MEM_FAIL;
    }

    currentStartAddr += SectorSize;

    if (SectorSize != STM32EXTLOADER_DEVICE_256K_SECTOR_SIZE)
    {
      SectorSize = (currentStartAddr < STM32EXTLOADER_DEVICE_128K_SECTOR_SIZE)
              ? STM32EXTLOADER_DEVICE_4K_SECTOR_SIZE
              : (currentStartAddr < STM32EXTLOADER_DEVICE_256K_SECTOR_SIZE)
               ? STM32EXTLOADER_DEVICE_128K_SECTOR_SIZE
               : STM32EXTLOADER_DEVICE_256K_SECTOR_SIZE;
```

```
    }
  } while ((currentStartAddr < currentEndAddr) && (retr == MEM_OK));

  return retr;
}
```

3. `EXTMEM_LOADER`: Device information updates

External Memory Loader provides STM32CubeProgrammer with device information through the `StorageInfo` structure. This structure includes the device name, device type, start address, device size, page size, and sector layout. If the project uses mixed sector sizes, the sector information in `Middlewares/ST/STM32_ExtMem_Loader/STM32Cube/stm32_device_info.c` should be reviewed and updated manually to match the actual memory layout used by the driver. The following code shows an example:

```
/**
 * @brief this variable is used by the loading program to get memory information
 */
#if defined(__ICCARM__)
__root sStorageInfo const StorageInfo =
#else
__attribute__((used)) sStorageInfo const StorageInfo =
#endif                                          /* __ICCARM__*/
{
    STM32EXTLOADER_DEVICE_NAME,           // Device Name
    STM32EXTLOADER_DEVICE_TYPE,           // Device Type
    STM32EXTLOADER_DEVICE_ADDR,           // Device Start Address
    STM32EXTLOADER_DEVICE_SIZE,           // Device Size in Bytes
    STM32EXTLOADER_DEVICE_PAGE_SIZE,      // Programming Page Size in Bytes
    STM32EXTLOADER_DEVICE_INITIAL_CONTENT, // Initial Content of Erased Memory
    {
      {32, 0x1000},   // SA00 - SA31 (4 KB sectors)
      {1, 0x20000},   // SA32 (128 KB sector)
      {255, 0x40000}, // SA33 - SA287 (256 KB sectors)
      {0x00000000, 0x00000000}
    },
};
```

4. Maximizing read throughput in the FSBL

To maximize read throughput, you must optimize the firmware implementation. In the default FSBL implementation, the application is copied from the external NOR flash to the internal SRAM using a byte-by-byte loop. This approach can lower read throughput and increase boot time. Therefore, it is

recommended to replace this read method in `Middlewares/ST/STM32ExtMem_Manager/boot/stm32_boot_lrun.c` with `memcpy()`, as follows:

```
...

/* Includes ------------------------------------------------------------------*/
#include "stm32_boot_lrun.h"
#include <string.h>

...

BOOTStatus_TypeDef copyApplication(void)
{
  ...

  /* Get the map address of the source memory */
  switch (EXTMEM_GetMapAddress(EXTMEM_LRUN_SOURCE, &MapAddress))
  {
    case EXTMEM_OK :
    {
    /* Manage the copy in mapped mode */
    source = (uint8_t *)(MapAddress + EXTMEM_LRUN_SOURCE_ADDRESS);
    img_size = BOOT_GetApplicationSize((uint32_t) source);
    /* Copy from source to destination in mapped mode */
    /* Comment out the byte-by-byte loop
    for (uint32_t index = 0; index < img_size; index++)
    {
      destination[index] = source[index];
    }
    */
    /* Read with memcpy() in memory-mapped mode */
    memcpy((void *)destination, (void *)source, img_size);

...
```

5 Programming to SEMPER™ NOR flash with Octal interface

STM32CubeProgrammer is a programming software for STM32 devices. It can program external flash memories connected to an XSPI controller.

This section describes how to add SEMPER™ NOR flash into the external loader of STM32CubeProgrammer. The details about STM32CubeProgrammer are described in UM2237.

1. Build the external Memory Loader project in STM32CubeIDE

In STM32CubeIDE, open the Properties dialog for the ExtMemLoader project.

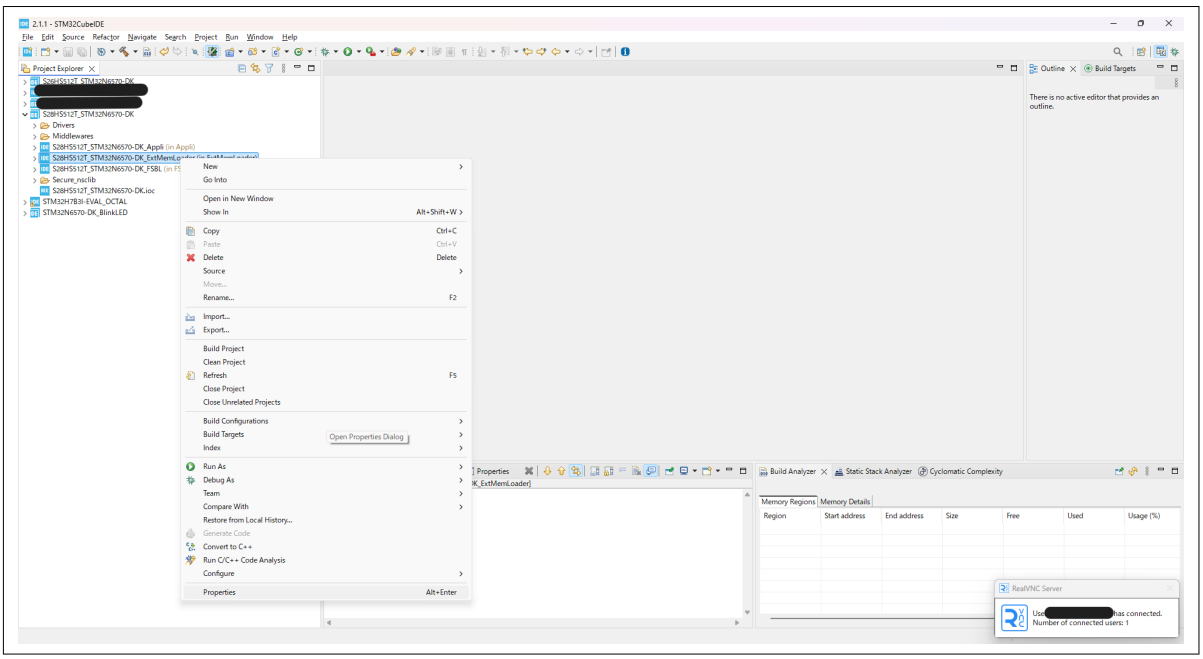

Figure 17 Open the Properties dialog for the ExtMemLoader project

Select **C/C++ Build** > **Settings**, and then select the **Build Steps** tab. Enter the following command in the **Post-build steps**:

```
cp "${BuildArtifactFileBaseName}.elf" "S28HS512T_STM32N6570-DK.stldr"
```

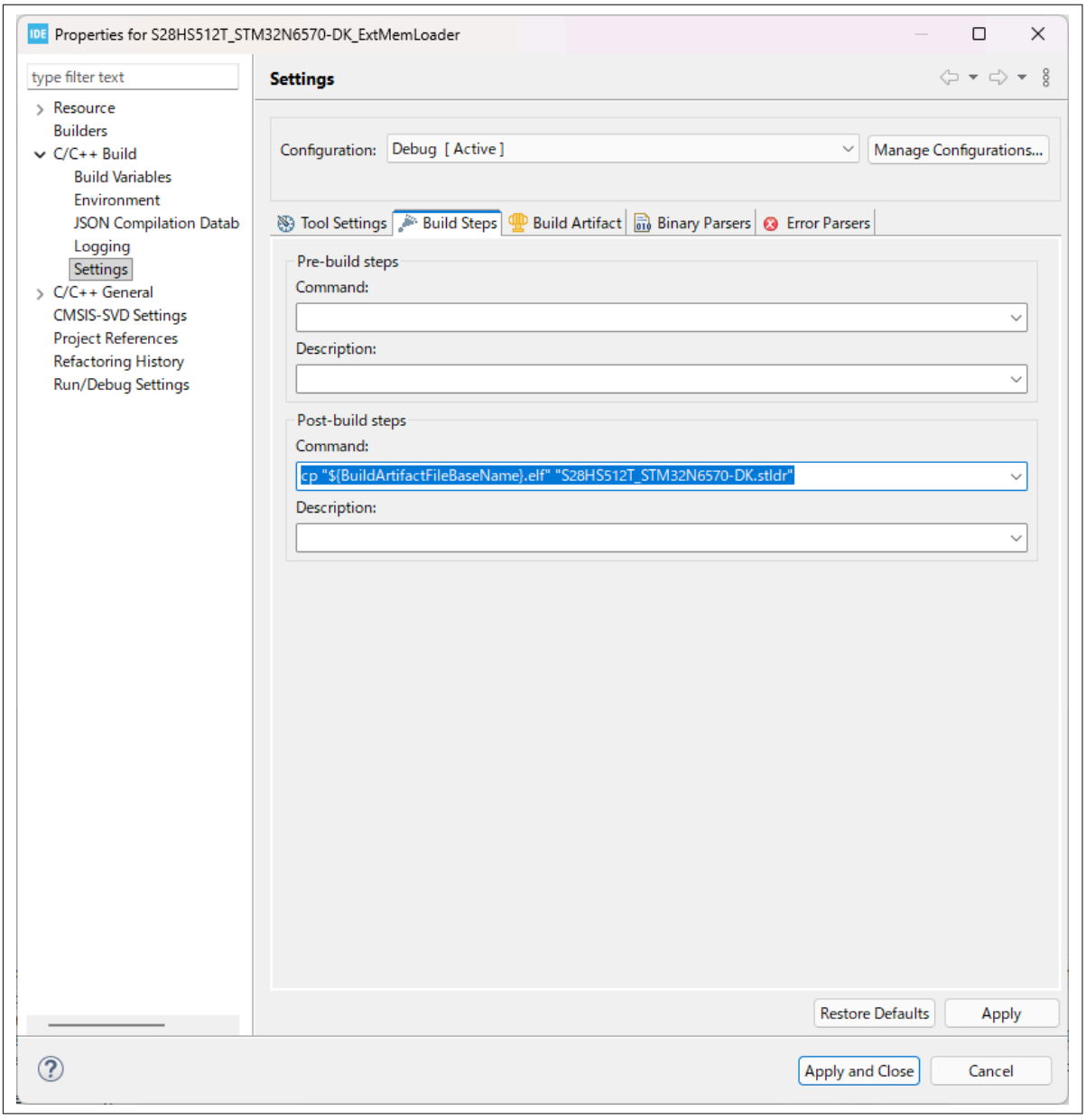

Figure 18 Enter the Post-build command

The `.stldr` file name is arbitrary and can be changed later. Click **Apply and Close** and build the project to generate the external loader file for STM32CubeProgrammer.

2. Copy the external loader file for STM32CubeProgrammer

   Copy the external loader file generated in the previous step into the `ExternalLoader` folder. The `.stldr` file is located in `ExtMemLoader\Debug` under the project folder. The `ExternalLoader` folder is located at the following path: `C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\ExternalLoader`.

3. Select the external loader for SEMPER™ NOR flash in STM32CubeProgrammer

   In STM32CubeProgrammer, click **EL** in the lower-left corner of the window to open the External Loaders list. Find and select the External Loader by the name. The SEMPER™ NOR flash can then be programmed by following the steps described in How to execute code from the external serial NOR using the STM32N6.

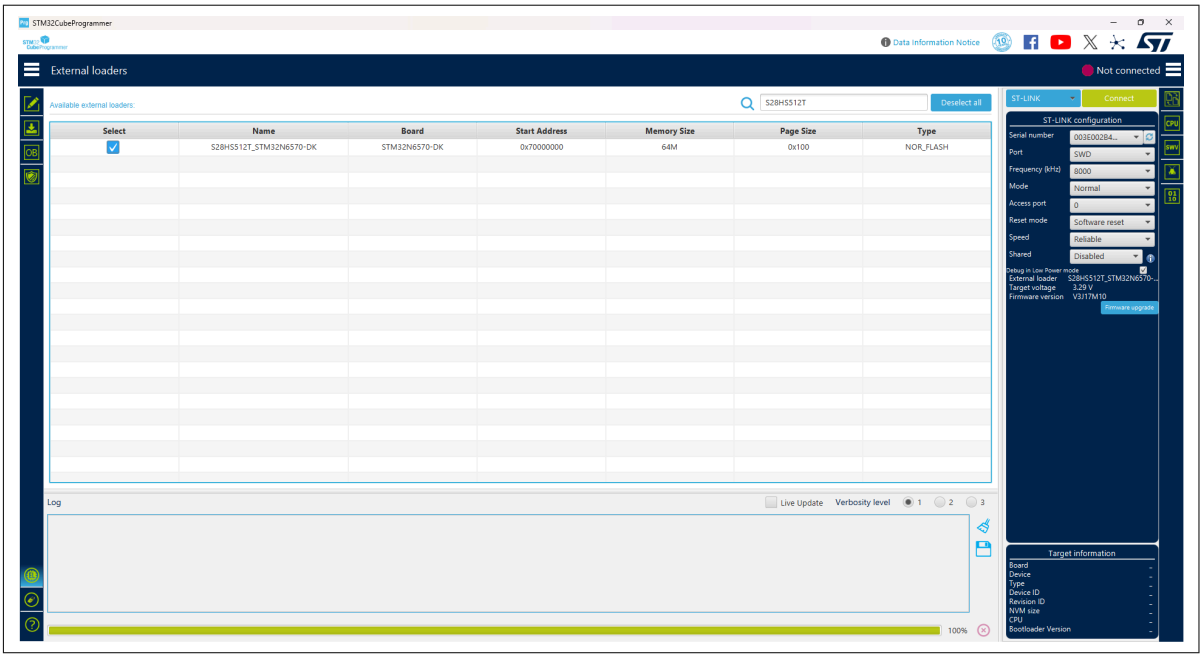

Figure 19 Select the external loader for SEMPER™ NOR flash

6 Summary

Infineon S28HL-T/S28HS-T SEMPER™ family devices can be used as external flash memory with the XSPI controller of the STMicroelectronics STM32N6 microcontroller. The BootROM firmware in STM32N6 can load the user FSBL, which can then configure the XSPI controller and load application from the SEMPER™ NOR flash operating at 200 MHz. STM32CubeProgrammer can support an algorithm for SEMPER™ NOR flash with Octal interface. To conclude, the middleware packs and GUI-based configuration tool in STM32CubeProgrammer help developers create a driver for SEMPER™ NOR flash and integrate the algorithm into both the FSBL and STM32CubeProgrammer.

References

[1] STMicroelectronics: *AN5050: Getting started with Octo-SPI, Hexadeca-SPI, and XSPI interfaces on STM32 MCUs*; Available online

[2] STMicroelectronics: *RM0486: STM32N647/657xx Arm®-based 32-bit MCUs reference manual*; Available online

[3] STMicroelectronics: *AN5967: Getting started with hardware development for STM32N6 MCUs*; Available online

[4] STMicroelectronics: *How to program the OTP fuse bits in the STM32N6*; Available online

[5] STMicroelectronics: *UM2237: STM32CubeProgrammer software description*; Available online

[6] Infineon Technologies AG: *SEMPER™ NOR flash webpage*; Available online

Revision history

Document revision	Date	Description of changes
**	2026-06-22	Initial release

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2026-06-22

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2026 Infineon Technologies AG

All Rights Reserved.

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-cfr1779365376313

Important notice

Products which may also include samples and may be comprised of hardware or software or both ("Product(s)") are sold or provided and delivered by Infineon Technologies AG and its affiliates ("Infineon") subject to the terms and conditions of the frame supply contract or other written agreement(s) executed by a customer and Infineon or, in the absence of the foregoing, the applicable Sales Conditions of Infineon. General terms and conditions of a customer or deviations from applicable Sales Conditions of Infineon shall only be binding for Infineon if and to the extent Infineon has given its express written consent.

For the avoidance of doubt, Infineon disclaims all warranties of non-infringement of third-party rights and implied warranties such as warranties of fitness for a specific use/purpose or merchantability. Infineon shall not be responsible for any information with respect to samples, the application or customer's specific use of any Product or for any examples or typical values given in this document.

If this document is marked as 'Preliminary', 'Target', 'Draft' or 'Proposal', Infineon reserves the right to change all information given in this document at any time without notice. Subject to the development and release of a Product for series supply by Infineon, the technical specifications of the Product are set forth in the relevant datasheet provided by Infineon without such marking.

The data contained in this document is exclusively intended for technically qualified and skilled customer representatives. It is the responsibility of the customer to evaluate the suitability of the Product for the intended application and the customer's specific use and to verify all relevant technical data contained in this document in the intended application and the customer's specific use. The customer is responsible for properly designing, programming, and testing the functionality and safety of the intended application, as well as complying with any legal requirements related to its use.

Unless otherwise explicitly approved by Infineon, Products may not be used in any application where a failure of the Products or any consequences of the use thereof can reasonably be expected to result in personal injury.

However, the foregoing shall not prevent the customer from using any Product in such fields of use that Infineon has explicitly designed and sold it for, provided that the overall responsibility for the application lies with the customer. Infineon expressly reserves the right to use its content for commercial text and data mining (TDM) according to applicable laws, e.g. Section 44b of the German Copyright Act (UrhG). If the Product includes security features:

Because no computing device can be absolutely secure, and despite security measures implemented in the Product, Infineon does not guarantee that the Product will be free from intrusion, data theft or loss, or other breaches ("Security Breaches"), and Infineon shall have no liability arising out of any Security Breaches.

If this document includes or references software:

The software is owned by Infineon under the intellectual property laws and treaties of the United States, Germany, and other countries worldwide. All rights reserved. Therefore, you may use the software only as provided in the software license agreement accompanying the software.

If no software license agreement applies, Infineon hereby grants you a personal, non-exclusive, non-transferable license (without the right to sublicense) under its intellectual property rights in the software (a) for software provided in source code form, to modify and reproduce the software solely for use with Infineon hardware products, only internally within your organization, and (b) to distribute the software in binary code form externally to end users, solely for use on Infineon hardware products. Any other use, reproduction, modification, translation, or compilation of the software is prohibited.

For further information on the Product, technology, delivery terms and conditions, and prices, please contact your nearest Infineon office or visit www.infineon.com.