

How to execute code from external memory on STM32F7 and STM32H750/H7B0/H730 MCUs

Introduction

There is an increased demand for applications able to support new and complex features, and as a consequence there is an increased demand for devices with a bigger flash-memory area.

The use of external flash memory provides higher storage capabilities with comparable performance levels while supplying a cost efficient solution for the demand of an increased flash-memory area.

STM32F7x0 Value line, STM32H750 Value line, STM32H7B0 Value line, and STM32H730 Value line devices respond to the market demand with a reduced inner flash-memory area.

This application note describes the steps needed to build an application on these Value line devices, with code execution from external memory.

It provides details on how to boot from internal flash memory, and then jump to user-application execution from an external memory.

Related documents

Available from the STMicroelectronics website at www.st.com:

- *STM32Cube MCU Package for STM32F7 Series with HAL, low-layer drivers and dedicated middleware* databrief (DB2601)
- *STM32Cube MCU Package for STM32H7 Series with HAL and dedicated middleware* databrief (DB3259)
- *STM32F75xxx and STM32F74xxx advanced Arm®-based 32-bit MCUs* reference manual (RM0385)
- *STM32H743/753 advanced Arm®-based 32-bit MCUs* reference manual (RM0433)
- *STM32H7A3/B3 and STM32H7B0 Value line advanced Arm®-based 32-bit MCUs* reference manual (RM0455)
- *STM32H723/733, STM32H725/735 and STM32H730 Value line advanced Arm®-based 32-bit MCUs* reference manual (RM0468)
- *STM32F7 Series system architecture and performance* application note (AN4667)
- *Quad-SPI (QSPI) interface on STM32 microcontrollers* application note (AN4760)
- *Getting started with STM32H7x3 hardware development* application note (AN4938)
- *Getting started with STM32F7 Series MCU hardware development* application note (AN4661)
- *Octo-SPI (OSPI) interface on STM32 microcontrollers* application note (AN5050)
- *STM32CubeProgrammer software description* user manual (UM2337)
- *Evaluation board with STM32H7B3LI MCU* user manual (UM2662)
- *Getting started with STM32CubeH7 for STM32H7 Series* user manual (UM2204)



1 General information

This document applies to Arm®-based devices.

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

arm

2 External memory code execution overview

2.1 External memory code execution principle

The STM32CubeF7 v1.12.0 and the STM32CubeH7 v1.8.0 firmware packages provide several applications to demonstrate how to boot from internal Flash memory and how to configure the external memories and jump to user application (located on the external memory). Two possible use cases are available: XiP and BootROM.

- The XiP use case is intended for "eXecute in Place" from external Flash memory (QSPI/OSPI or FMC-NOR Flash memory). The user-application code must be linked with the target execution memory-address (external QSPI/OSPI or FMC-NOR Flash memory).
- The BootROM use case is intended to demonstrate how to boot from the internal Flash memory, configure the external RAM memories (SDRAM, SRAM or OSPI-RAM), copy user-application binary from the code storage area (an SDCARD or an SPI-Flash memory) to the external SDRAM, external SRAM or external OSPI-RAM, and then jump to the user application. The user-application code must be linked with the target execution memory address (external SDRAM, SRAM or OSPI-RAM).

The applications described in the table below are available on the firmware package under \Applications\ExtMem_CodeExecution for the following boards:

- 32F723EDISCOVERY board for the STM32F730 devices
- STM32756G-EVAL board for the STM32F750 devices
- STM32H743I-EVAL board for the STM32H750 devices
- STM32H7B3I_EVAL and STM32H7B3I-DK board for the STM32H7B0 devices
- STM32H735G-DK board for the STM32H730 devices

Table 1. Application details

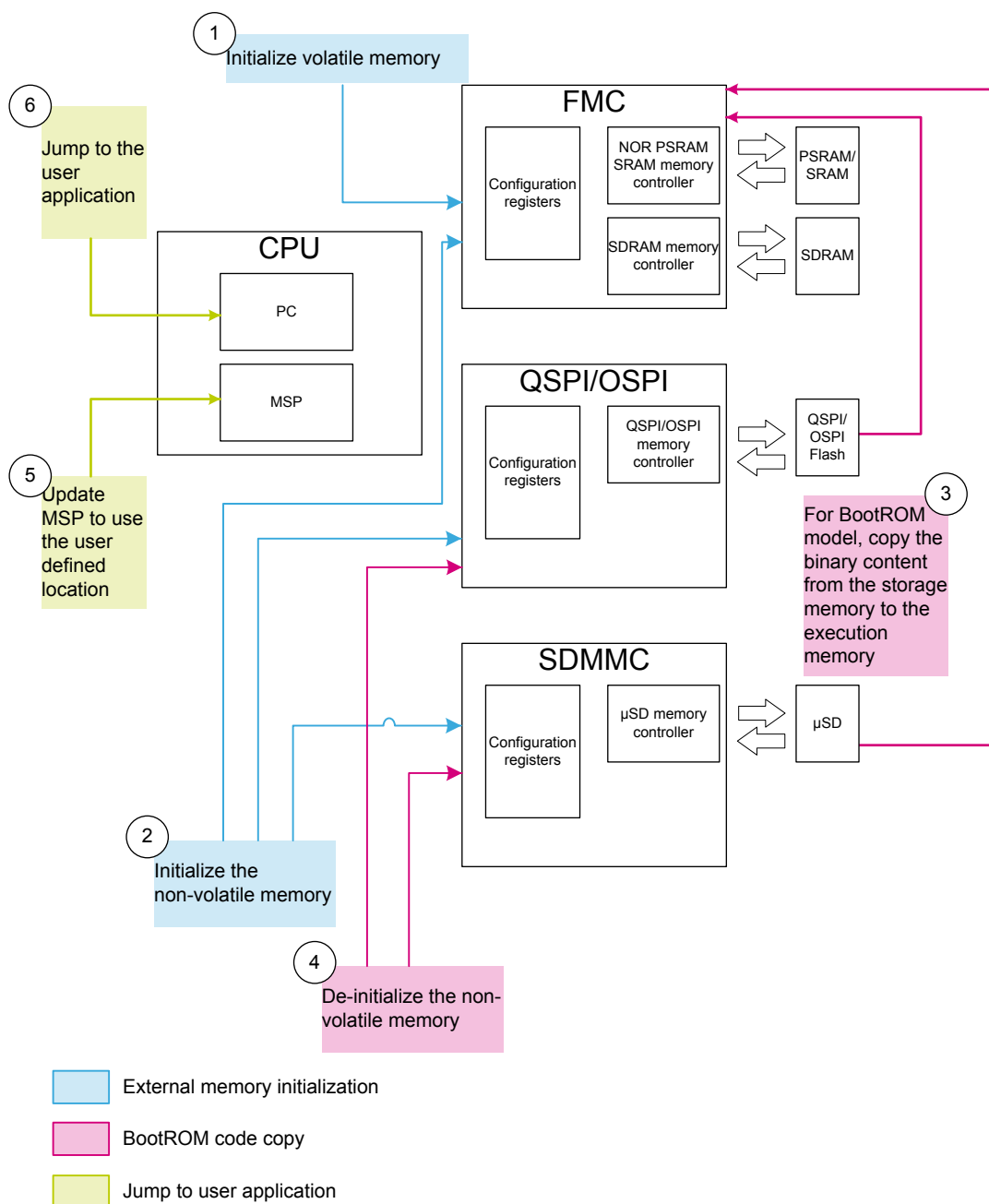
Application	Description
ExtMem_Boot	Shows how to boot from internal Flash memory, configure external memories and then jump to user application located on external memory. The user can select QSPI/OSPI Flash memory, FMC-NOR Flash memory, external SDRAM, external SRAM or external OSPI-RAM for code execution.
ExtMem_Application\LedToggling	Sample application running from external Flash memory (QSPI/OSPI Flash memory or FMC-NOR Flash memory), external SRAM, external SRAM or external OSPI-RAM
ExtMem_Application\FreeRTOS	Sample FreeRTOS application with execution from external Flash memory (QSPI/OSPI Flash memory or FMC-NOR Flash memory), external SRAM, external SRAM or external OSPI-RAM

The *external memory boot* application is in charge of initializing the required resources to make the external memories available and ready to use. This application initializes the required resources as per the user configuration (see [Section 3.3: Configuration](#)).

The *external memory boot* application must setup the main stack pointer and configure the application to be executed on external memory. This type of boot schema enables the support of sizable user applications.

The *external memory boot* application ensures that any resources that are no longer needed after the setup phase are reset or free before jumping to the user application. The figure below presents an overview of this boot schema.

Figure 1. External memory code boot schema



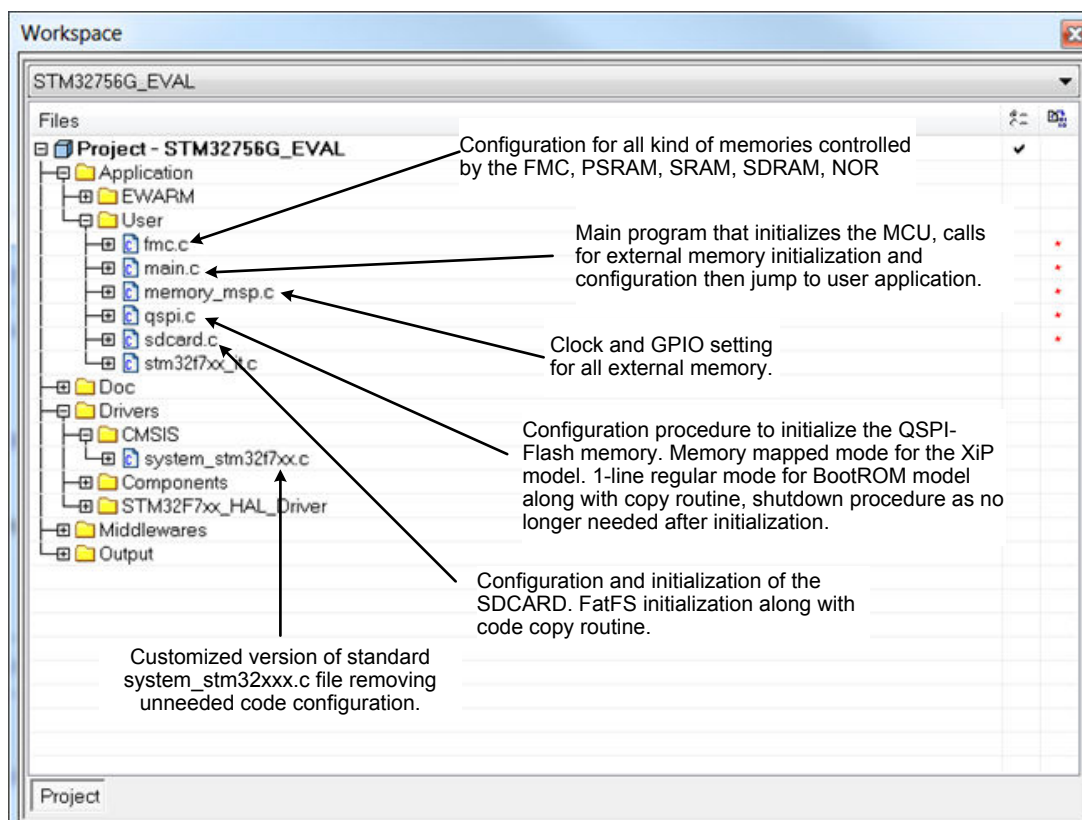
2.2 External memory boot application description

The *external memory boot* application contains a set of source files on the STM32CubeF7/H7 package which is tailored to match the supported configuration for each hardware platform.

For the STM32H7B0 Value line and the STM32H730 Value line devices, the settings of the clock and GPIO for the external memories are done within the BSP drivers for each memory. The `qspi.c` file is replaced by `ospi.c` file.

The figure below shows an example of the superset of all files for all the supported configurations.

Figure 2. External memory boot application superset of source files



3 Supported boot model

The application supports two types of execution models:

- Execute in place support (XiP support)
- BootROM support

The users must select the configuration matching their needs by tuning the *memory.h* header file.

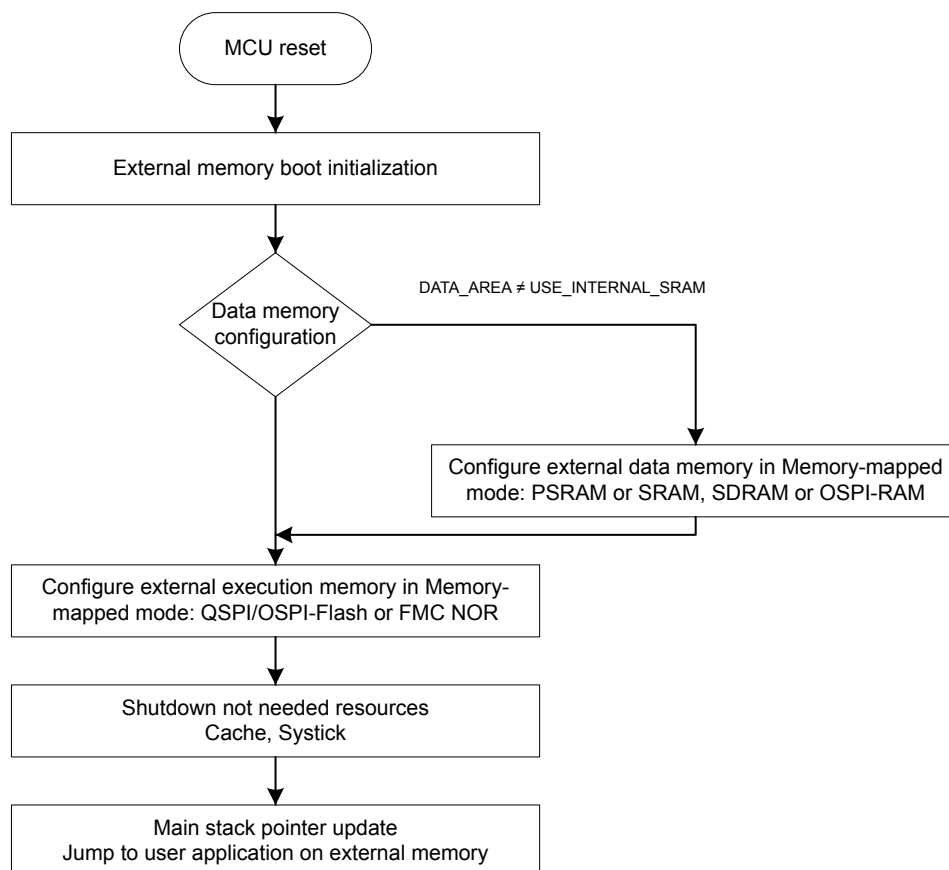
3.1 Execute in place (XiP) support

The XiP model is based on code execution directly from the external non-volatile memory that is used for code storage. This execution model requires memory-mapped support to grant the CPU with direct access to the executed-code user application. The XiP model is available on external NOR/QSPI/OSPI Flash memory through the FMC/QSPI/OSPI interfaces.

Based on the user configuration in the *memory.h* file, the *External memory boot* application configures **one** of the following volatile memories: SDRAM, SRAM, PSRAM, OSPI-RAM or internal SRAM. In this model the volatile memory is used for data only.

The following flowchart illustrates the operational flow for the XiP model.

Figure 3. XiP model operational flow



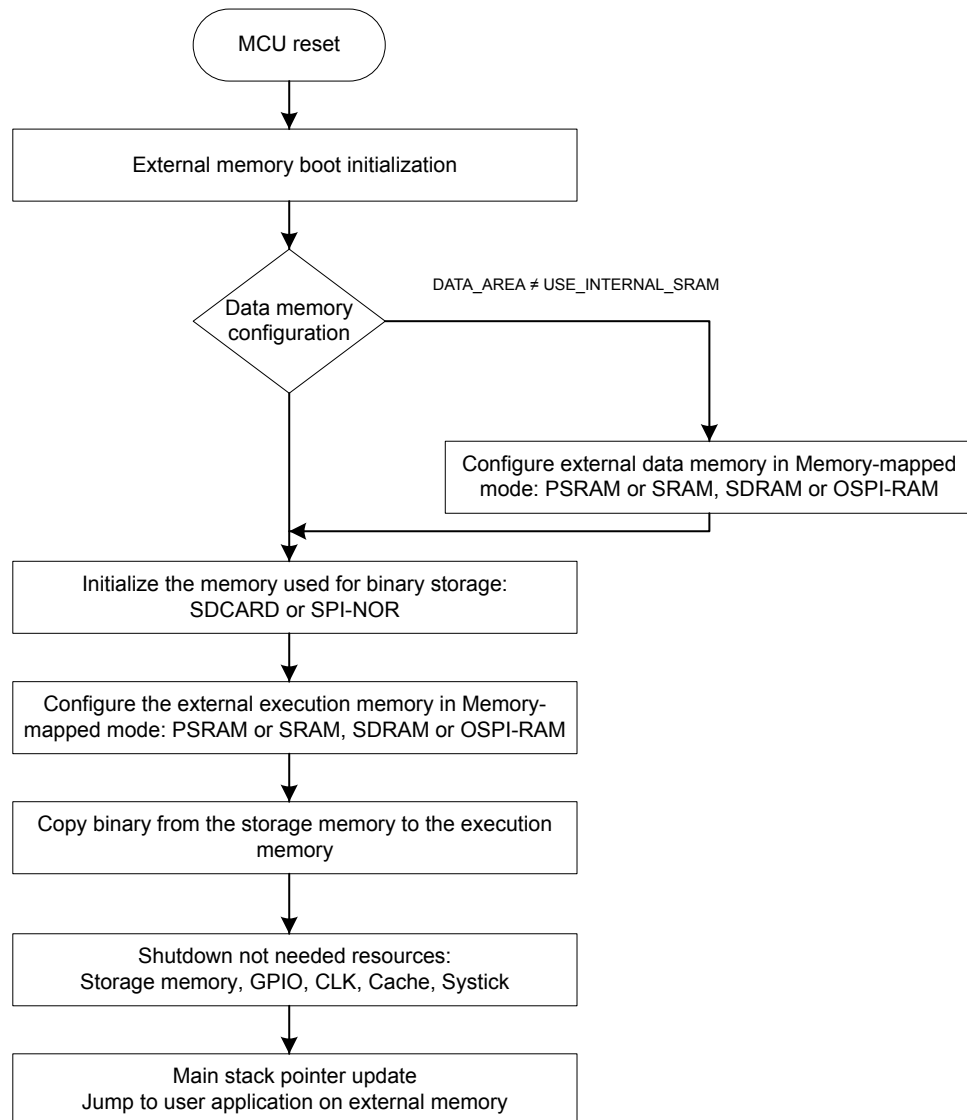
3.2 BootROM support

The BootROM model is based on code execution from a chosen volatile memory. This execution model is suitable when binary data is stored in a memory that has no memory-mapped interface (like for SDCARD). This model is also suitable when binary data is stored in a memory with low throughput (like for SPI-NOR (emulated using QSPI/OSPI with 1- line)).

Based on the user configuration in the *memory.h* file, the *external memory boot* application configures **two** of the following volatile memories: SDRAM, SRAM, PSRAM, OSPI-RAM or internal SRAM. In this model, binary data is copied from a non-volatile memory to one volatile memory prior to the execution by the *external memory boot* application. The second volatile memory is used for data.

The following flowchart illustrates the operational flow for the BootROM model.

Figure 4. BootROM model operational flow



3.3 Configuration

The user configuration is defined by the following *defines*:

- **DATA_AREA:** it is used to specify the volatile memory that is used for data holding. Supported memories (depending on the board used) are:
 - USE_EXTERNAL_SDRAM: external SDRAM is used for data holding
 - USE_EXTERNAL_SRAM: external SRAM is used for data holding
 - USE_EXTERNAL_PSRAM: external PSRAM is used for data holding
 - USE_INTERNAL_SRAM: internal SRAM is used for data holding
 - USE_EXTERNAL_RAM : external OSPI-RAM is used for data holding
- **CODE_AREA:** it is used to specify the execution location of the user application. This area can be a volatile memory for the BootROM schemas or a non-volatile for the XiP schemas. The supported memories (depending on the hardware used) are:
 - XiP model: BINARY_AREA must be undefined:
 - USE_QSPI: QSPI Flash is used for code execution
 - USE_OSPI: OSPI Flash is used for code execution
 - USE_NOR: FMC-NOR is used for code execution
 - BootROM model: BINARY_AREA must be defined
 - USE_EXTERNAL_SDRAM: external SDRAM is used for code execution
 - USE_EXTERNAL_SRAM: external SRAM is used for code execution
 - USE_EXTERNAL_PSRAM: external PSRAM is used for code execution
 - USE_INTERNAL_SRAM: internal SRAM is used for code execution
 - USE_EXTERNAL_RAM : external OSPI-RAM is used for code execution
- **BINARY_AREA:** is defined in the BootROM model only. It is used to specify the location of the binary containing the user application. Additional *defines* are needed depending on the chosen configuration. Supported memories (depending on the hardware used):
 - USE_SPI_NOR: SPI NOR Flash is used for binary storage
 - BINARY_BASE_OFFSET: offset of the binary within SPI NOR Flash
 - BINARY_SIZE: size of the binary image
 - USE_SDCARD: SDCard is used for binary storage
 - BINARY_FILENAME: name of the binary file to be executed

The user must make sure that the selected memories contain code and data to cover at least a proper user application startup. Afterwards, the user application can initialize any other memory needed.

3.4 Summary of external memories part numbers

The following table summarizes the part numbers of the external memories used versus the board and boot model. As there is not a dedicated board for devices of STM32F7x0 Value line, STM32H750 Value line, the STM32H7B0 Value line and the STM32H730 Value line, the boards (with compatible devices) that are used are:

- 32F723EDISCOVERY is used to emulate the STM32F730 devices.
- STM32F756G-EVAL is used to emulate the STM32F750 devices.
- STM32H743I-EVAL is used to emulate the STM32H750 devices.
- STM32H7B3I-EVAL and STM32H7B3I-DK are used to emulate the STM32H7B0 devices.
- STM32H735G-DK is used to emulate the STM32H730 devices.

Table 2. External memories used on each board by boot model

Boot model	Memory	32F723EDISCOVERY with STM32F730 devices	STM32F756G-EVAL with STM32F750 devices	STM32H743I-EVAL with STM32H750 devices	STM32H7B3I-EVAL with STM32H7B0 devices	STM32H7B3I-DK with STM32H7B0 devices	STM32H735G-DK with STM32H730 devices
XiP	QSPI:OSPI Flash memory	MX25L51245GZ2I-08G (bus width : 4 lines)	N25Q512A13GSF40E (bus width : 4 lines)	Two Quad-SPI Flash MT25QL512ABB8ESF-0SIT Or One twin Quad-SPI Flash MT25TL01GHBB8ESF-0SIT (bus width : 8 lines)	Octo-SPI Flash NOR MX25LM51245G (bus width: 8 lines)	Octo-SPI Flash NOR MX25LM51245G (bus width: 8 lines)	Octo-SPI Flash NOR MX25LM51245G (bus width: 8 lines)
	NOR Flash memory (on FMC)	-	PC28F128M29E WL A (bus width : 16-bit)	MT28EW128ABA1 L PC-OSIT	MT28EW128ABA1 LPC-OSIT	-	-
BootROM	SPI-NOR (emulated with QSPI 1 line)	-	N25Q512A13GSF40E (bus width : 2 lines)	Two Quad-SPI Flash MT25QL512ABB8ESF-0SIT Or One twin Quad-SPI Flash MT25TL01GHBB8ESF-0SIT (bus width : 2 lines)	Octo-SPI Flash NOR MX25LM51245G (bus width: 1 line)	Octo-SPI Flash NOR MX25LM51245G (bus width: 1 line)	Octo-SPI Flash NOR MX25LM51245G (bus width: 1 line)
	SDCARD	-	Native support	Transceiver IP4856CX25/C_Module_REV	HSP051-4M10	HSP051-4M10	HSP051-4M10
Volatile memory	Internal SRAM	Native support	Native support	Native support	Native support	Native support	Native support
	External SRAM	-	IS61WV102416BL L -10ML I (bus width : 16-bit)	IS61WV102416BL L -10ML I (bus width : 16-bit)	IS42S32800J-6B LI (bus width : 32-bit)	-	-
	External SDRAM	-	IS42S32800G-6BL I (bus width : 32-bit)	IS42S32800G-6BL I (bus width : 32-bit)	IS42S32800J-6B LI (bus width : 32-bit)	IS42S32800J-6B LI (bus width : 16-bit)	-
	External PSRAM	IS66WV51216EBLL-55BLI	-	-	-	-	-

Boot model	Memory	32F723EDISCOVERY with STM32F730 devices	STM32756G-EVAL with STM32F750 devices	STM32H743I-EVAL with STM32H750 devices	STM32H7B3I-EVAL with STM32H7B0 devices	STM32H7B3I-DK with STM32H7B0 devices	STM32H735G-DK with STM32H730 devices
Volatile memory		(bus width : 16-bit)					
	External OSPI-RAM	-	-	-	-	-	OSPI HyperRAM S70KL1281

4 Resources constraints to be considered

Any resources that are no longer needed after initialization (interruption, ongoing transfers, unused pins) must be released before jumping to the user application. This must be done to avoid an extra power consumption and to limit any interference with the user application. Especially for the BootROM model, as the peripherals used for binary storage are no longer required, they must be reset.

The user must consider the amount of resources used by the *external memory boot* application in order to ensure that the external memory interface remains up and running. The resources constraints are linked to:

- The allocation and configuration of the pins
- The configuration of the interface (QSPI/OSPI IP register must not be modified, the FMC IP register can be partially updated)
- The RCC configuration to avoid IP reset clock disabling and clock frequency/source update in a harmful way.

The pin allocation table below is provided as reference and is valid for a pin selection according to the used board. Other pin selection could be used based on the available alternate functions.

Table 3. Pins allocated for each memory by board

Memory/ board	32F723EDISCOVERY with STM32F730 devices	STM32756G-EVAL with STM32F750 devices	STM32H743I-EVAL with STM32H750 devices	STM32H7B3I- EVAL with STM32H7B0 devices	STM32H7B3I-DK with STM32H7B0 devices	STM32H735G-DK with STM32H730 devices
QSPI/OSPI Flash memory	PB(2,6), PC(9,10), PD13, PE2	PB(2,6), PF(6,8,9)	PB2, PC11, PF(6,7,8,9), PG(6,9,14), PH(2,3)	PB2, PC(1,5), PD6, PF(6,7,8, 9), PG(6,14), PH3	PG(6,9), PB2, PC(1,5), PD(11,7), PF(6,7,9), PH3	PB2, PD(4,5,7,11,12,13), PE2, PG(6,9), PF10
NOR Flash memory (on FMC)	-	PD(0,1,4,5,6,7,8,9, 10,11,12,13,14,15) PE(2,3,4,5,6,7,8,9, 10,11,12,13,14,15) PF(0,1,2,3,4,5,12,1 3,14,15) PG(0,1,2,3,4,5)	PD(0,1,4,5,6,7,8,9, 10,11,12,13,14,15) PE(2,3,4,5,6,7,8,9, 10,11,12,13,14,15) PF(0,1,2,3,4,5,12,1 3,14,15) PG(0,1,2,3,4,5)	PC(6,7), PD(0,1,4,5,8,9,10,1 1,12,13,14,15), PE(0,1,7,8,9,10,11, 12,13,14,15), PF(0,1,2,3,4,5,12,1 3,14,15), PG(0,1,2,3,4,5)	-	-
External SRAM	-	PD(0,1,3,4,5,8,9,10 ,11,12,13,14,15) PE(0,1,3,4,7,8,9,10 ,11,12,13,14,15) PF(0,1,2,3,4,5,12,1 3,14,15) PG(0,1,2,3,4,5,10)	PD(0,1,3,4,5,8,9,10 ,11,12,13,14,15) PE(0,1,3,4,7,8,9,10 ,11,12,13,14,15) PF(0,1,2,3,4,5,12,1 3,14,15) PG(0,1,2,3,4,5,6,9, 10,12,13,14)	PD(0,1,4,5,8,9,10,1 1,12,13,14,15), PE(0,1,7,8,9,10,11, 12,13,14,15), PF(0,1,2,3,4,5,12,1 3,14,15), PG(0,1,2,3,4,5,10)	-	-
External SDRAM	-	PD(0,1,8,9,10,14,1 5) PE(0,1,7,8,9,10,11, 12,13,14,15) PF(0,1,2,3,4,5,11,1 2,13,14,15) PG(0,1,4,5,8,15) PH(2,3,5,8,9,10,11, 12,13,14,15) PI(0,1,2,3,4,5,6,7,9, 10)	PD(0,1,8,9,10,14,1 5) PE(0,1,7,8,9,10,11, 12,13,14,15) PF(0,1,2,3,4,5,11,1 2,13,14,15) PG(0,1,2,3,4,5,8,15) PH(5,6,7,8,9,10,11, 12,13,14,15) PI(0,1,2,3,4,5,6,7,9, 10)	PA7, PD(0,1,8,9,10,14,1 5), PE(0,1,7,8,9,10,11, 12,13,14,15), PF(0, 1,2,3,4,5,11,12,13,1 4,15), PG(0,1,4,5,8,15), PH(6,7)	PD(0,1,8,9,10,14,1 5), PE(0,1,7,8,9,10,11, 12,13,14,15), PF(0,1,2,3,4,5,11,1 2,13,14,15), PG(0,1,4,5,8,15), PH(5,6,7)	-
External PSRAM	PD(0,1,4,5,7,8,9,10 ,11,12,14,15) PE(0,1,7,8,9,10,11, 12,13,14,15) PF(0,1,2,3,4,5,12,1 3,14,15) PG(0,1,2,3,4,5)	-	-	-	-	-
External OSPI-RAM	-	-	-	-	-	PF(0,1,3,4,12), PG(0,1,10,11,12)

The following table summarizes the resources that must be kept unmodified. It describes a list of peripherals (or part of peripheral) that must not be modified in order to avoid the unavailability of external storage. The mentioned peripherals must not be reset or clock disabled, nor reconfigured in a manner that can alter their behavior.

Note: Some elements might change based on the external memory boot application configuration chosen for the selected board. and on the platform's hardware.

Note: Single QSPI mode is not supported with the external loaders as long as the external memories are all Twin Quad SPI Flash

Table 4. Peripherals required by memory type

	32F723EDISCOVERY with STM32F730 devices	STM32756G-EVAL with STM32F750 devices	STM32H743I-EVAL with STM32H750 devices	STM32H7B3I-EVAL with STM32H7B0 devices	STM32H7B3I-DK with STM32H7B0 devices	STM32H735G-DK with STM32H730 devices
QSPI/OSPI Flash memory	QSPI1 (0x9000000)	QSPI1 (0x9000000)	Dual QSPI Mode QSPI is not supported for this application	OSPI (octal mode)	OSPI (octal mode)	OSPI (octal mode)
NOR Flash (on FMC)	-	FMC-NOR (No FMC-PSRAM / FMC-SRAM)	FMC-NOR (No FMC-PSRAM / FMC-SRAM)	FMC-NOR (No FMC-SRAM)	-	-
External SRAM	-	FMC-SRAM (No FMC-PSRAM / FMC-NOR)	FMC-SRAM (No FMC-PSRAM / FMC-NOR)	FMC-SRAM (No FMC-NOR)	-	-
External SDRAM	-	FMC-SDRAM (200 MHz maximum system frequency)	FMC-SDRAM	FMC-SDRAM	FMC-SDRAM	-
External PSRAM	FMC-SRAM (No FMC-SRAM / FMC-NOR)	-	-	-	-	-
External OSPI-RAM	-	-	-	-	-	OSPI (linear burst)

5 Description of the external memory user application

5.1 Required updates

The external memory application is based on a specific boot schema, which is different from the standard one and which supports a smooth transition from the on-chip application to the off-chip application.

There are two updates that must be done by the user as the location of the application has changed:

- Ensure the usage of the required linker file with memory mapping that corresponds to the selected boot option.
- Update the settings of VTOR to use the right address.

5.2 Load and debug

The five boards 32F723EDISCOVERY, STM32756G-EVAL, STM32H743I-EVAL, STM32H7B3I-EVAL, STM32H7B3I-DK and STM32H735G-DK have a loader for external non-volatile memories. Those loaders are provided within the STM32CubeF7/H7 as:

- Patch for EWARM IDE (not required when using EWARM v8.50.6 or higher)
- Dedicated pack for MDK-ARM IDE

The XiP model provides a seamless load and debug experience similar to an internal Flash debugging. For SW4STM32 IDE and STM32Cube IDE the STM32CubeProgrammer must be used for application loading on external Flash memories.

In BootROM model, the application is compiled and linked for execution from an external volatile memory:

- External SDRAM : linker address 0xD0000000 for STM32H750 Value line and STM32H7B0 Value line, and 0xC0000000 for STM32F7x0 Value line
- External SRAM : linker address 0x68000000 for STM32H750 Value line, STM32H7B0 Value line, and STM32F7x0 Value line
- External OSPI-RAM : linker address 0x70000000 for STM32H730 Value line

The application binary must be then stored either into the SPI_NOR Flash memory or into the SDCARD. It is up to the boot application to copy the user application from the storage area to the execution RAM area.

As consequence, the application's load schema cannot be handled by the IDE (MDK-ARM or EWARM) external memory Flash loader (as the application's link address and storage address are different).

Depending on the **BINARY_AREA** define (specified in the "memory.h" file of the boot application), this model requires the use of the two different loading schemas below:

- **SPI_NOR**
The user application must be stored into the SPI-NOR Flash memory at the address 0x90000000. It has to be done using the STM32CubeProgrammer. The output of the application must be in binary format in order to be able to specify a different load address which is the SPI-Flash address. See details in the figure below.
- **SDCARD**
The user must manually copy the binary file, output of the build, into the SDCARD that is used to store the user application, then plug the SDCARD into the evaluation board.

The figure below shows the steps to be followed to load and debug:

Figure 5. STM32CubeProgrammer

Figure 5 shows the STM32CubeProgrammer interface with several key components highlighted by numbered callouts:

- 1:** Connect button (top right)
- 2:** Download button (left sidebar)
- 3:** External loaders list (bottom table)
- 4:** File path field (top left)
- 5:** Start address field (center)
- 6:** Start Programming button (bottom right)

The External loaders list (Table 1) contains the following data:

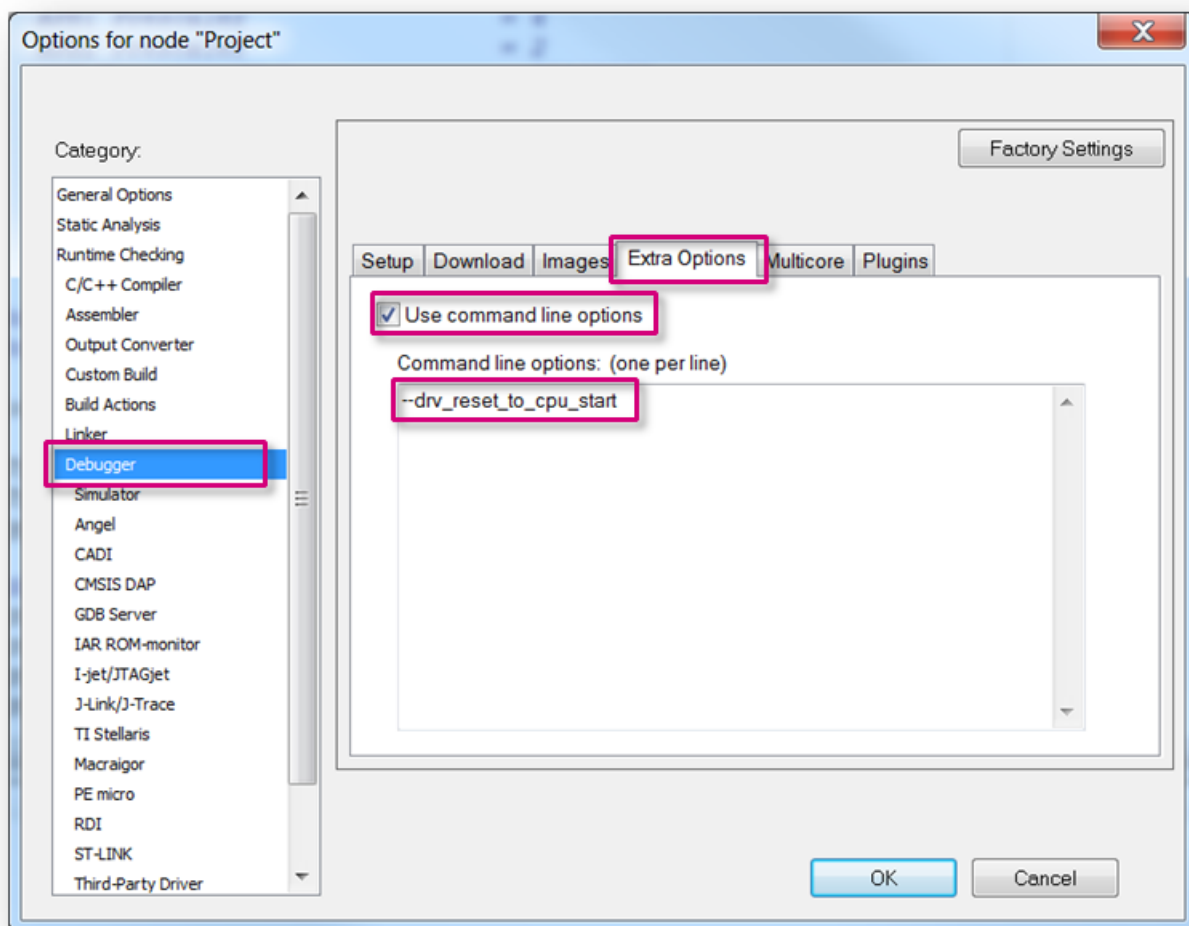
Loader	Device	Start Address	Size	Flash Size	Flash Type
MT25T101G_STM32H7xxI-EVAL	STM32H7xxI-EVAL	0x90000000	128M	0x100	NOR_FLASH
MT25QL512A_STM32756G-EVAL	STM32756G-EVAL	0x90000000	64M	0x10	NOR_FLASH
MX25LM51245G_STM32H7B3I-EVAL	STM32H7B3I-EVAL	0x90000000	64M	0x1000	NOR_FLASH
MX25LM51245G_STM32H7B3I_DISCO	STM32H7B3I	0x90000000	64M	0x1000	NOR_FLASH
MX25LM51245G_STM32H735G-DK	STM32H735G-DK	0x90000000	64M	0x1000	NOR_FLASH

5.3 Debug using EWARM IDE

A special precaution is needed with the EWARM IDE when debugging the user application that is running from the external memory. EWARM overrides the default CPU reset value of the PC (program counter) by the one given in the user application (an address value within the external execution memory).

In this boot schema the user application PC address remains inaccessible until the *external memory boot* application is executed (so the external memory is ready and memory mapped via the FMC or QSPI/OSPI). A hardfault is generated if the EWARM jumps directly to the start point of the user application. To avoid the hardfault, the user must add the "--drv_reset_to_cpu_start" command line in the debugger options as shown in the figure below. This setting prevents the EWARM from forcing the PC and gives place to the *external memory boot* application to configure the external memory before jumping to the user application.

Figure 6. Debugger command line options



6 Performance characterization

When executing from external memory the performances are impacted due to the external Flash memory latency and the longer instruction/data path. By using the STM32F7x0 Value line, STM32H750 Value line and STM32H7B0 Value line devices, this impact is reduced thanks to the Cortex-M7 L1-cache.

The table below summarizes the EEMBC® CoreMark® scores achieved for each combination of ROM/RAM. The best performances can be achieved when executing from the internal Flash memory. Nevertheless the loss is significantly reduced when execution from an external memory.

These figures illustrate the impact on CPU performance when operating from external memories. The internal Flash configuration score is provided as reference.

Table 5. EEMBC® CoreMark® score per configuration

ROM memory	RAM memory	32F723EDISCOVERY with STM32F730 devices (I/D Cache 8K/8K)	STM32756G-EVAL with STM32F750 devices (I/D Cache 4K/4K)	STM32H743I-EVAL with STM32H750 devices (I/D Cache 16K/16K)	STM32H7B3I-EVAL with STM32H7B0 devices (I/D Cache 16K/16K)	STM32H735G-DK with STM32H730 devices (I/D Cache 16K/16K)
QSPI/OSPI Flash memory	Internal SRAM	1089	948	2020	1414	2777
	External SRAM	-	940	1972	1380	-
	External SDRAM	-	871	1972	1380	-
	External PSRAM	1079	-	-	-	-
	External RAM	-	-	-	-	2713
NOR Flash memory	Internal SRAM	-	906	2020	1414	-
	External SRAM	-	899	1972	1380	-
	External SDRAM	-	833	1972	1380	-
External SRAM	Internal SRAM	-	1016	2020	1414	-
External SDRAM	Internal SRAM	-	989	2020	1414	-
External RAM	Internal SRAM	-	-	-	-	2777
Internal Flash	Internal SRAM	1092	1082	2020	1414	2777

Revision history

Table 6. Document revision history

Date	Version	Changes
11-Jul-2018	1	Initial release.
16-Jun-2020	2	<p>Added STM32H7B0 Value line in the whole document.</p> <p>Updated cover.</p> <p>Updated Section 2.1: External memory code execution principle.</p> <p>Updated Section 2.2: External memory boot application description.</p> <p>Updated Section 3.1: Execute in place (XiP) support.</p> <p>Updated Section 3.3: Configuration.</p> <p>Added STM32H7B3I-EVAL and STM32H7B3I-DK with STM32H7B0 devices in:</p> <ul style="list-style-type: none"> Table 2. External memories used on each board by boot model. Table 3. Pins allocated for each memory by board. Table 4. Peripherals required by memory type. Updated Section 5.2: Load and debug. Table 5. EEMBC® CoreMark® score per configuration.
03-Sep-2020	3	<p>Added STM32H730 Value line, external OSPI-RAM, STM32H735G-DK in the whole document:</p> <ul style="list-style-type: none"> Updated cover. Updated Section 2.1: External memory code execution principle. Updated Section 2.2: External memory boot application description. Updated Section 3.1: Execute in place (XiP) support. Updated Section 3.2: BootROM support. Updated Section 3.3: Configuration. Updated Table 2. External memories used on each board by boot model. Updated Table 3. Pins allocated for each memory by board. Updated Table 4. Peripherals required by memory type. Updated Section 5.2: Load and debug. Updated Table 5. EEMBC® CoreMark® score per configuration.
20-Jan-2026	4	Updated document title.

Contents

1	General information	2
2	External memory code execution overview	3
2.1	External memory code execution principle	3
2.2	External memory boot application description	5
3	Supported boot model	6
3.1	Execute in place (XiP) support	6
3.2	BootROM support	6
3.3	Configuration	8
3.4	Summary of external memories part numbers	9
4	Resources constraints to be considered	11
5	Description of the external memory user application	14
5.1	Required updates	14
5.2	Load and debug	14
5.3	Debug using EWARM IDE	15
6	Performance characterization	17
	Revision history	18
	List of tables	20
	List of figures	21

List of tables

Table 1.	Application details	3
Table 2.	External memories used on each board by boot model	9
Table 3.	Pins allocated for each memory by board	12
Table 4.	Peripherals required by memory type	13
Table 5.	EEMBC® CoreMark® score per configuration	17
Table 6.	Document revision history	18

List of figures

Figure 1.	External memory code boot schema	4
Figure 2.	External memory boot application superset of source files	5
Figure 3.	XiP model operational flow	6
Figure 4.	BootROM model operational flow	7
Figure 5.	STM32CubeProgrammer	15
Figure 6.	Debugger command line options.	16

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgment.

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

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

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

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

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

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

© 2026 STMicroelectronics – All rights reserved