



STMicroelectronics Community > Resources > Knowledge base > STM32 MCUs >

STM32N6 boot ROM explained**B. Montanari**

ST Technical Moderator · 1 year ago



STM32N6 boot ROM explained

1 reply · 11211 views

Summary

The STM32N6 boot ROM is a critical component that initiates the boot process for the STM32N6 microcontroller. It performs various functions such as system initialization, reset source detection, and boot memory device selection. It supports various external memory devices and facilitates code downloading over serial interfaces like USART and USB. The boot ROM ensures a secure boot process, image authentication, and optional image decryption. During development, the DEV boot mode allows for direct execution of applications or the FSBL from internal RAM, aiding in debugging and testing.

Introduction

This article explains how the boot ROM works in the STM32N6 microcontroller, which is based on the Arm® Cortex®-M55 core. We look at what the boot ROM does, how it interacts with other parts of the system, and how it helps get your microcontroller up and running.

1. What does the boot ROM do?

The boot ROM is the first code that runs when the STM32N6 is powered on or reset. It lives in the on-chip ROM and kicks off the boot process. Here are its main capabilities:

- **System initialization:** Gets the basic system up and running.
- **Reset detection:** Figures out why the system reset and what to do next.
- **Bootstrapping:** Loads the initial code from various types of external memory.
- **Code downloading:** Downloads code over serial interfaces (USART, USB).
- **Security:** Validates signed images and supports several security features.

When the STM32N6 starts, the boot ROM follows a specific sequence of steps: it detects the reset source, selects the boot device, and performs secure boot authentication. During development, you can use the DEV boot mode to make debugging easier.

1.1 boot ROM code flow diagram

The boot ROM code flow diagram shows the main activities and possible branches. All possible sequence of steps taken during the boot process, including system initialization, reset source detection, boot device



STM32 Sidekick



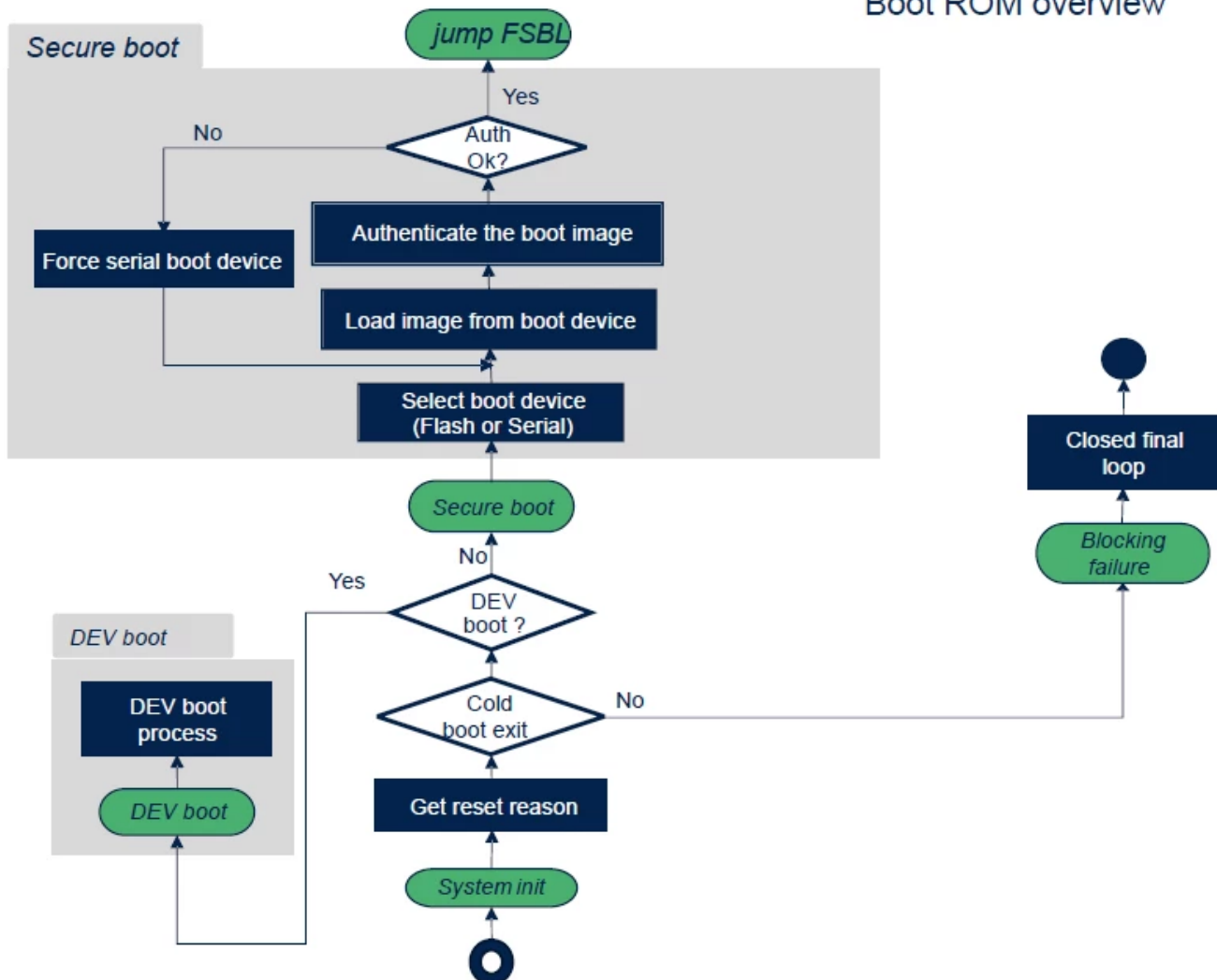
Have a question on STM32?
Get answers from our AI
assistant.



Feedback

selection, secure boot authentication, and image loading. This is the treasure map on how the boot ROM works. During most of the development phase, the user is in DEV boot mode. This is achieved by using the BOOT pins on the hardware level to make a particular branch path. The image below shows the boot ROM overview:

Boot ROM overview



DT75062V1

The DEV boot allows the debug to work, and we can activate it by setting the BOOT1 pin to VDDIO. The BOOT0 pin state does not matter to enter this mode.



STM32 Sidekick

Have a question on STM32?
Get answers from our AI assistant.





Flash Boot
 BOOT0=L (VSS)
 BOOT1=L (VSS)



DEV Boot
 BOOT0=X
 BOOT1=H (VDDIO)

The boot ROM runs whenever the STM32N6 is reset. It can handle different types of resets, such as system resets and key provisioning resets. The boot configuration is determined by the register bits BOOTSR[0:1], which reflect the level of the external boot pins as latched at reset. Which means that the BOOT0 and BOOT1 pins are read during a system reset, not just upon a power cycle.

Depending on the detected type of reset, different branches within the boot ROM code implementation are executed. The following logical reset types are applicable and distinguished by the boot ROM code:

- SYSTEM: Logical system reset
- ST_KEY_PROVISIONING: Logical reset of the ST key provisioning stage condition with SFT and PIN resets

1.2 Execution of special boot branches

1.2.1 Dev boot execution

The dev boot mode is executed only in the CLOSED_UNLOCKED life cycle, protecting assets and reopening debug secure and nonsecure.

1.2.2 Blocking failure execution

The blocking failure scenario occurs during boot ROM code execution if an error happens, clearing and locking sensitive data, and sending UART status traces.

2. Utilizing DEV boot mode

The DEV boot mode can be utilized during the development phase to load applications from RAM. This mode is executed only in the CLOSED_UNLOCKED life cycle. The life cycle is mentioned later in this article. In this scenario, the boot ROM debugs secure and nonsecure, and then goes into an endless loop. As the boot mode is selected using the Boot1 pin.

Another useful feature is that by using the DEV boot mode. Developers can directly load and execute applications or the FSBL from the internal RAM to facilitate the debugging experience and test during the development phase.



STM32 Sidekick

Have a question on STM32?
 Get answers from our AI
 assistant.

Ask your question...



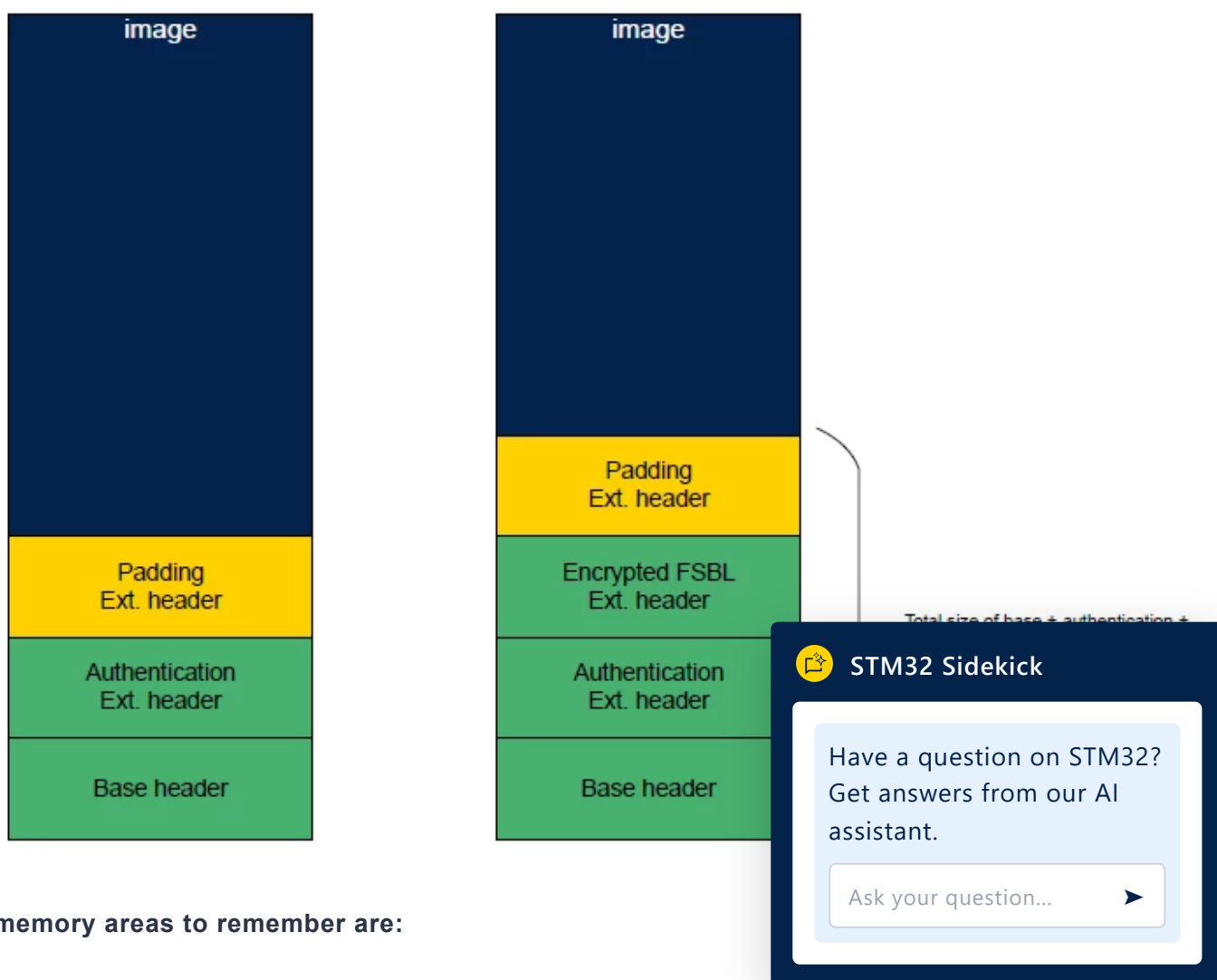
3. Supported boot memory devices

The boot memory device is the external flash memory device attached to the STM32N6 MCU on which the First stage bootloader (FSBL) is located. This means that the boot ROM can load the FSBL from various external memory devices, including:

- sNOR x4 and x8 flash devices
- HyperFlash™
- SD memory card
- eMMC memory card

The boot ROM code handles its data in the AXI SRAM2 internal RAM by configuring the RISAF2 hardware functional block, which is dedicated to RISAF for AXI SRAM2 internal memory. The boot ROM code configures seven regions inside RISAF2 and dynamically sets each region depending on the boot ROM code phase being executed. This means that it uses the AXI SRAM2 internal RAM to handle its data and configures the RISAF2 hardware block to manage memory regions dynamically. The layout of this internal RAM for the STM32N6 project is as follows:

Figure 10. Headers examples



Key memory areas to remember are:

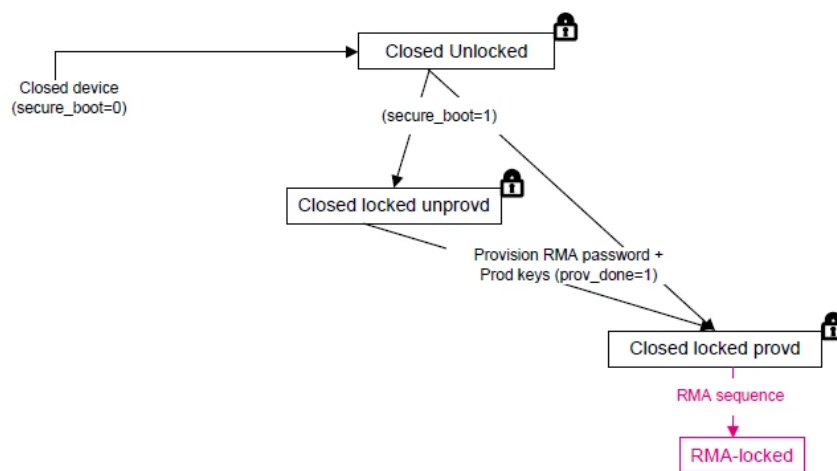
- **512 KB area for FSBL:** Stores the FSBL image in internal RAM including the header.
- **1 KB download buffer header:** Holds metadata about the downloaded FSBL image.

The boot ROM code uses the 512KB area in the AXI SRAM2 to store the FSBL image. This area is crucial in order to load the FSBL from an external boot memory device into the internal RAM, where it can be executed. The download buffer header, located at 0x34180200, is used to store the header information of the downloaded FSBL image. This header includes metadata such as the image length, entry point, and other relevant details required for the boot process.

4. Supported life cycle

The life cycle defines the states that determine the behavior and available features of the STM32N6 MCU device during the different steps of its life. The value of fuses determines the life cycle state. The BSEC function controls these fuses. The boot ROM code follows this life cycle to provide the expected features according to the life cycle state.

In-field states



Debug/engi tests

DT75061V1

5. Supported serial boot interfaces

The boot ROM code provides functionality for downloading code over a serial boot interface into the internal RAM of the STM32N6 MCU. Typically, downloading code over a serial boot interface is used to update the FSBL stored on a flash-type attached boot memory device. The boot ROM code supports the following types of serial boot interfaces:

- USART type of serial interface using three different USART hardware
- USB interface (USB 2.0 HS)

6. Secure boot

6.1 Secure boot overview

The boot ROM code oversees the first stage in the trust chain, implementing a secure boot process that includes parameter checks, image authentication, and jumping to the payload.

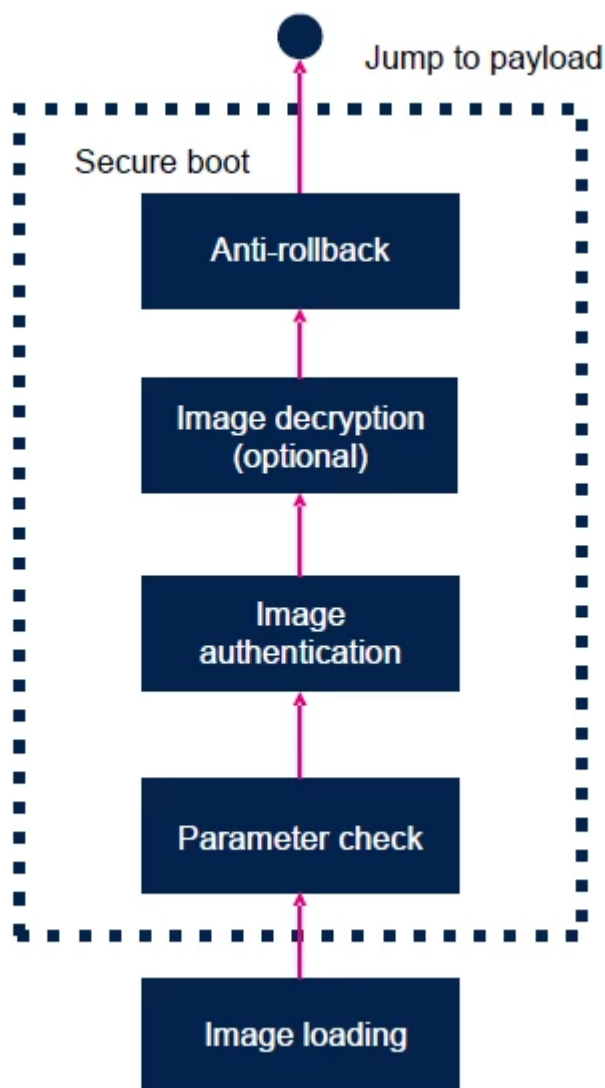


STM32 Sidekick

Have a question on STM32?
Get answers from our AI
assistant.

Ask your question...





6.2 Parameter Checks

The boot ROM code checks all parameters needed for authentication. This includes the presence of headers extension flag, verifying headers parameters and geometry, and ensuring the image version is equal to or greater than the OTP monotonic counter.

- **Image authentication**

The image authentication process includes signature verification, key version update.

- **Signature verification**

The boot ROM code implements ECDSA 256 and ECDSA 384 algorithm.

- **ECDSA key revocation**

If the signature verification is successful, the boot ROM code applies

- **Image decryption**



STM32 Sidekick

Have a question on STM32?
Get answers from our AI
assistant.

Ask your question...



The boot ROM code follows a sequence for FSBL decryption, including deriving the decryption key, decrypting the FSBL, calculating the plain FSBL SHA256, and comparing the 128 MSB bits.

- **FSBL version update**

If the FSBL version is higher than the OTP monotonic counter, the boot ROM code updates the OTP to match the FSBL version.

7. Error and trace logging

The boot ROM code manages errors and traces, using a dedicated GPIO to communicate statuses and writing binary traces to memory. In case of a blocking failure, the boot ROM code writes a UART log error to the debug GPIO pin. Note that PG10 (AF11) is connected to BOOTFAILED, so you can see a toggle LED in case of failure.

Conclusion

The STM32N6 boot ROM is essential to start the microcontroller. It initializes the system, handles resets, selects the boot device, and ensures a secure boot process. During development, the DEV boot mode makes debugging easier. The boot ROM supports various memory devices and serial interfaces, providing flexibility and security for your applications.

Related links

- [User manual 3234: How to proceed with boot ROM on STM32N6 MCUs](#)
- [User manual 3300: Discovery kit with STM32N657X0 MCU](#)
- [Reference manual 0486: STM32N647/657xx Arm®-based 32-bit MCUs](#), chapters 4, 7, 10, and 78

STM32N6 series

FSBL

Like 2 Reply Follow Share



Leave a reply...

1 reply



Mikk Leini
Senior · 4 months ago

A tip that took me at least a while to figure out.

To execute FSBL over USB DFU, boot into serial boot mode (pulling BOOT1 high when powering up or restarting) and use this STM32CubeProgrammer CLI command:

STM32 Sidekick

Have a question on STM32?
Get answers from our AI assistant.

Ask your question...

```
STM32_Programmer_CLI.exe -c port=usb1 -d firmware.bin 0x1 -detach
```

It copies firmware.bin into AXISRAM2 and if it is valid (checked by Boot ROM) it gets executed. It only remains there until next restart.

So if somebody happens to corrupt FSBL in NOR Flash on CLOCKED_LOCKED_PROVD MCU it gives a chance to recover the device. The FSBL that runs in RAM must write NOR Flash with new FSBL (from some data interface) or with itself.

 Like  Reply

ST Community highlights – April to June 2026



1 month ago

Related topics

STM32N6 Boot Pins

STM32 MCUs Boards and hardware tools

First project on STM32N6750-DK

STM32 MCUs Boards and hardware tools

How to understand the STM32N6 using STM32CubeMX

STM32CubeMX (MCUs)

TouchGFX for STM32N6570 - "Cannot Run Target" and "Program and Run Target" Disabled

STM32 MCUs TouchGFX and GUI

Issue after modifying linker script to use larger RAM region on STM32N6570-DK

STM32 MCUs Embedded software

Popular tags

STM32H7 series

STM32CubeMX

STM32F4 series

S

STM32L4 series

USB

STM32F7 series

STM32MP15



STM32 Sidekick

Have a question on STM32?
Get answers from our AI
assistant.

Ask your question...

