



STMicroelectronics Community > Resources > Knowledge base > STM32 MCUs >

How to add the STM32N6's header signature as post build

**B. Montanari**

ST Technical Moderator · 1 year ago



How to add the STM32N6's header signature as post build

1 reply · 3567 views

Summary

The first stage bootloader (FSBL) requires a header signature for the boot ROM to load the code from the external memory into the internal RAM. This allows the application's code to begin execution. Creating this header involves the STM32SigningTool's command line and adding this command as part of the building process can greatly facilitate development.

Introduction

On STM32N6 MCUs, the FSBL must be signed so the boot ROM can execute it in a secured-locked state. More information can be found on our wiki page: [Getting started with STM32N6 security - stm32mcu](#). However, the steps necessary to execute this process can become repetitive and time-consuming, especially when working on applications that require more than one binary. This article aims to introduce a simple postbuild script that allows for the entire signing process to occur after every build operation in the STM32CubeIDE. Additionally, it provides tools to understand and modify the script to suit other specifications.

1. The signing process

The STM32N6 microcontroller series is based on the Arm® Cortex®-M55 core, in which the firmware security is allowed by default. This means that every binary must be signed before being uploaded, ensuring the authenticity and integrity of the loaded images.

This process adds a base header that contains the signed binary size and the binary entry point address. It may also add an encryption extension header used for securing the binary. Additionally, a padding extension header to ensure that the total header size is fixed and the interrupt vector table is mapped correctly. For more information, consult the article [STM32N6 FSBL explained](#).

The signed binaries are generated using the STM32 Signing Tool software that can generate image files and private and public keys. This article will only focus on adding the header via a post build command, without any security. This software is installed as part of the [STM32CubeProgrammer](#).



2. The post build script

In this article, the script below is used to add the required binary from the current project. Starting from STM32CubeProgrammer version 2.21.0, a new parameter is needed at the end **"-align"**. **Please make sure you use the updated command, as follows:**

```
1 | cd "${ProjDirPath}/Debug" && echo y | "C:\Program
Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_SigningTool_CLI.exe"
-bin "${ProjName}.bin" -nk -of 0x80000000 -t fsbl -o "${ProjName}-Trusted.bin" -hv
2.3 -dump "${ProjName}-Trusted.bin" -align
```

Each command of the script above can be broken down to three main functionalities:

- **cd "\${ProjDirPath}/Debug"**: Change to the directory where the binary file is expected to be.
- **echo Project Name: \${ProjName}**: Prints the project name to verify the macro resolution.
- **echo y**: Types 'y' to allow the overwriting the previous signed binary.
- **"C:\ProgramFiles\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_SigningTool_CLI.exe" -bin "\${ProjName}.bin" -nk -of 0x80000000 -t fsbl -o "\${ProjName}-Trusted.bin" -hv 2.3 -dump "\${ProjName}-Trusted.bin" -align**: Runs the signing tool with the specified parameters and echo the outputs into the console.

The last command uses the STM32 Signing Tool parameters to set the proper configuration of the signed binary. The commands explanations are highlighted in the image below:

"C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_SigningTool_CLI.exe"
-bin Template_FSBL.bin **-nk** **-of** 0x80000000 **-t** fsbl **-o** Template_FSBL-Trusted.bin **-hv** 2.3 **-dump**
 Template_FSBL-Trusted.bin

Explanation :

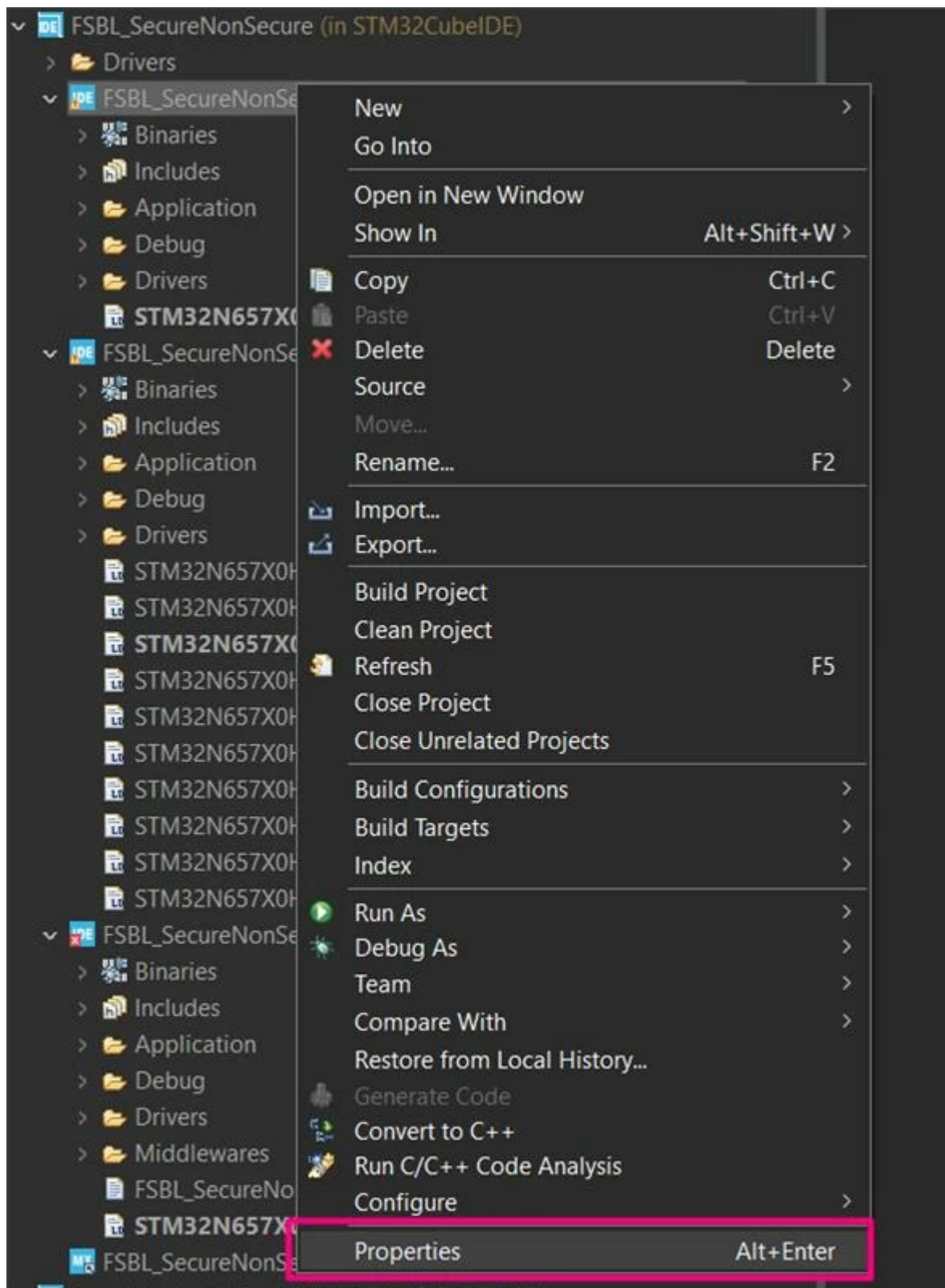
--binary-image	-bin	: Input binary image file path
--no-keys	-nk	: Adding empty header without key options.
--option-flags	-of	: Option flags of image
		b0=1: Authentication extension header
		b1=1: FSBL encryption extension header
		b31=1: Padding extension header
--type	-t	: Binary Type
--output	-o	: Output file path
--header-version	-hv	: Signing header version
--dump-header	-dump	: Parse and dump image header

3. Automating the process

This article assumes you have installed [STM32CubeMX](#) (6.13 or later), the latest version of the STM32N6 HAL driver, [STM32CubeProgrammer](#) (2.18 or later), and [STM32CubeIDE](#) (1.17.0 or later).

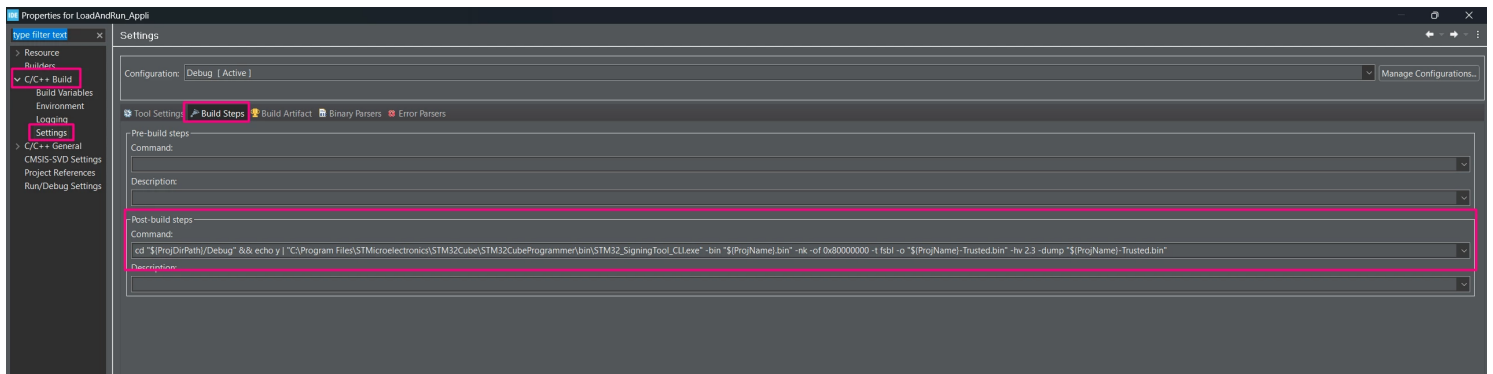
As stated, the manual signature process can be time-consuming and repetitive. However,  **Ask AI** can be completely automated by using the post build steps feature from STM32CubeIDE. The post build steps allow the execution of commands after the build process is completed.

First, after importing or creating a project, right-click on the project name and click on [Properties].

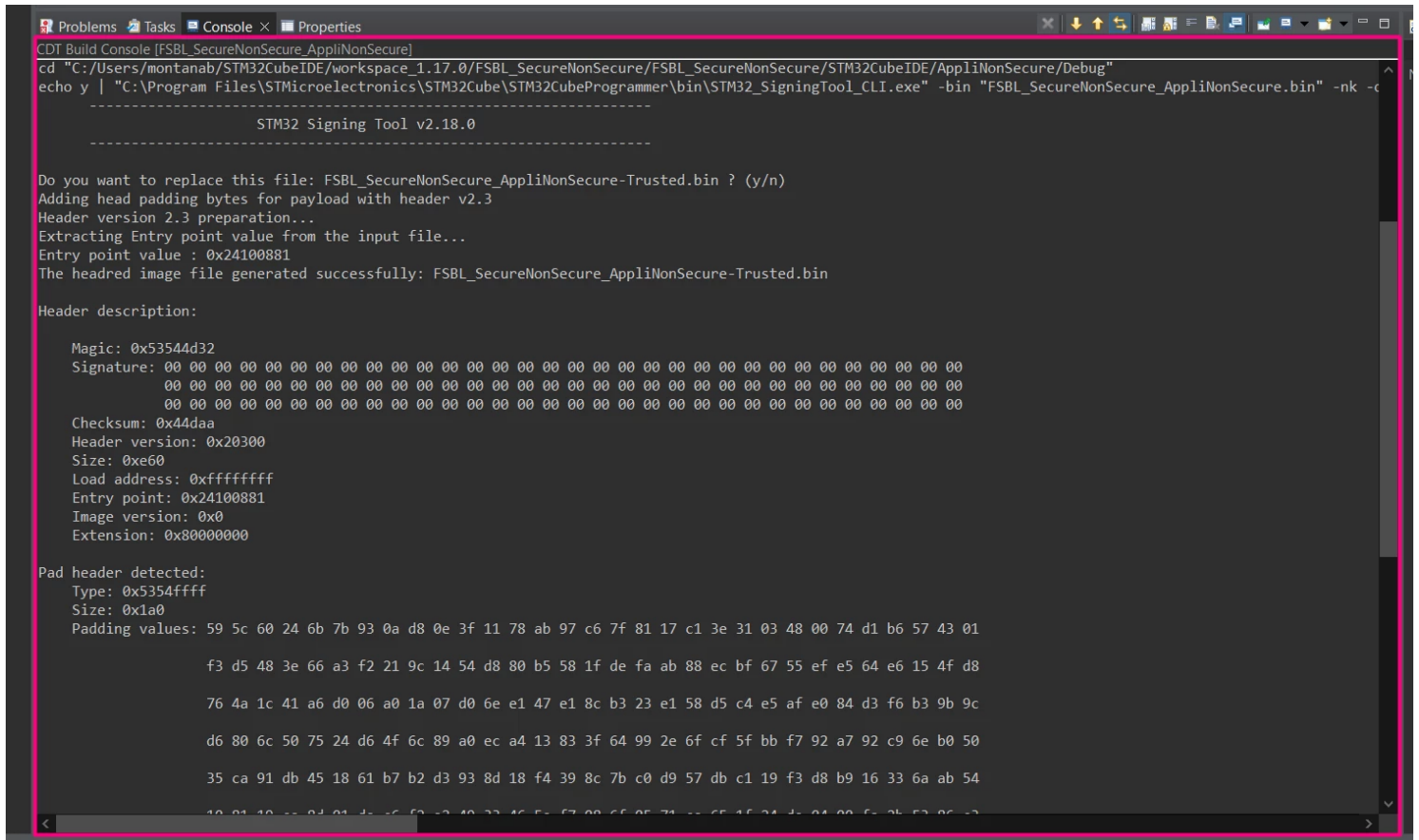


A window should open. In the left part of the menu, click on the arrow next to **[C/C++ Build]**, then select **[Settings]**. In the right part, a **[Settings]** configuration should appear. Select the **[Build Steps]** option. Finally, paste the command inside the **[Post-build steps]** text box, then apply the configurations and exit the window.





Now, whenever the project is built, the binary will have the header added as a post build action. The user can observe the message in the console output, as shown in the image below.



Most applications for the STM32N6 have two or more projects as part of the active workspace. It's important to emphasize that the post build command outlined can be implemented for all projects.

Conclusion

Post build scripts are crucial parts of a concise and modern programming environment. By understanding how to use them to our advantage, we save time and are able to create other automation with much less effort.

Related links

- [ST Wiki: STM32 Signing Tool](#)
- [Knowledge article: STM32N6 FSBL explained](#)

