Community

1

Search all content... Ctrl + K

STM32 MCUs

B.Montanari
ST Technical Moderator · 1 year ago

How to program the OTP fuse bits in the STM32N6

5 replies

Summary

This article provides a quick guide on what the STM32N6's OTP bits are and how to program them. It includes step-by-step instructions and explanation on configuring the OTP124, responsible for changing the VDDIOs' voltage to 1.8 V instead of 3.3 V using STM32CubeProgrammer. This allows the STM32N6570-DK to achieve the nominal speed with the external memories.

Introduction

The STM32N6 series microcontrollers use one-time programmable (OTP) memory to set various configuration parameters, ranging from hardware configuration to security features. OTP bits are read or programmed by the BSEC (boot and security control). The full description of them can be found in reference manual 0486, tables 18 - 20, named "OTP fuse description".

Among these OTP bits, it is possible to configure the STM32N6's independent VDDIOx voltage level. Specific I/O voltage settings can be configured for: VDD, VDDIO2, VDDIO3, VDDIO4, and VDDIO5, allowing the use at 3.3 V or 1.8 V. The IO ranges are not continuous, and they should be set using the OTP to achieve their nominal speed when used in the 1.8 V range. This is the case for the STM32N6570-DK board, where the external memories are tied to 1.8 V, but the default OTP for the VDDIO2 and VDDIO3 is 3.3 V.

The article provides a step-by-step guide on how to program the OTP124 of the STM32N6 MCU on the STM32N6570-DK board. Specifically focusing on bits 15 and 16, which are responsible for VDDIO3 and VDDIO2, respectively. The programming is made using STM32CubeProgrammer. By following this guide, you gain a comprehensive understanding of the process and ensure the correct configuration of these critical settings.

Feedback

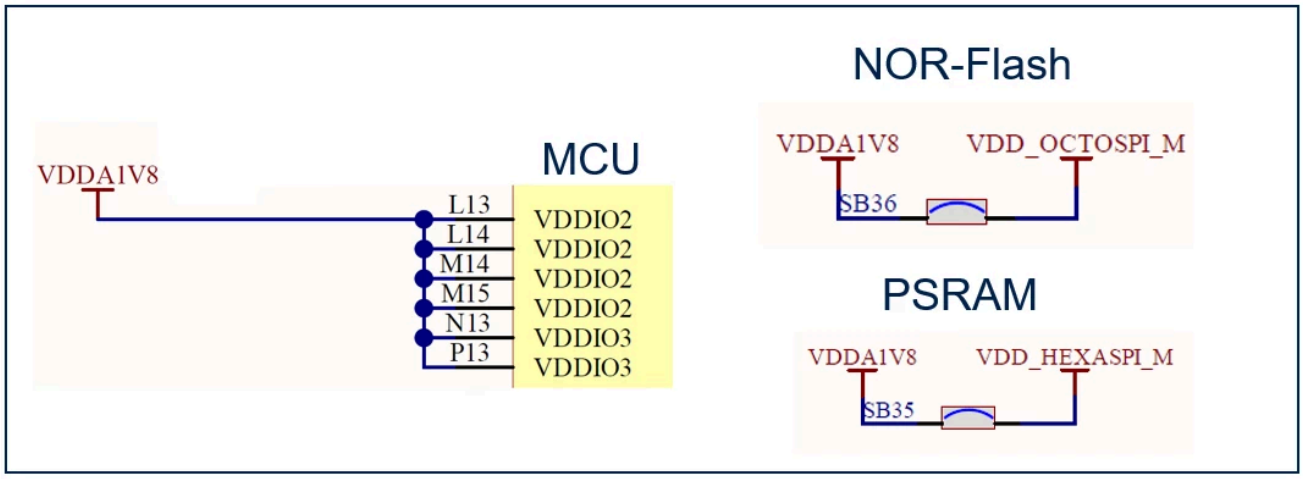
Ask AI

1. Hardware and OTP fuse

The STM32N6570-DK uses external memories at 1.8 V voltage and relevant I/Os should be configu from the 3.3 V default using firmware or software and OTP to get nominal speed.

Be aware that setting HSLV_VDDIOx and VDDIOxVRSEL bits while VDDIOx is in the 3.3 V range damages the device.

The relevant power supply connection on the discovery kit hardware is shown below:



This means that:

- XSPIM_P2 NOR-Flash (8-bit) → VDDIO3
- XSPIM_P1 PSRAM (16-bit) → VDDIO2

The OTP124 bits reflect the I/O domains voltage levels, as shown in the reference manual's table:

OTP124	-	HCONF1	-
	[0]	IWDG1_HW	IWDG1 start on reset
	[1]	IWDG1_FZ_STOP	IWDG1 freeze in Stop mode
	[2]	IWDG1_FZ_STANDBY	IWDG1 freeze in Standby mode
	[9:3]	Reserved	Reserved
	[10]	RST_STOP	Reset caused if the device is put in Stop mode
	[11]	RST_STDBY	Reset caused if the device is put in Standby mode
	[12]	SELINBORH	– 0: BOR disabled – 1: BOR = 2.7 V
	[13]	HSLV_VDDIO5	VDDIO5 I/O segment below 2.5 V for I/O mode. The I/O segment is used by SDMMC2 port.
	[14]	HSLV_VDDIO4	VDDIO4 I/O segment below 2.5 V for I/O mode (I/O segment used by SDMMC1 port)
	[15]	HSLV_VDDIO3	VDDIO3 I/O segment below 2.5 V for I/O mode (I/O segment used by XSPIM port 2)
	[16]	HSLV_VDDIO2	VDDIO2 I/O segment below 2.5 V for I/O mode (I/O segment used by XSPIM port 1)
	[17]	HSLV_VDD	Main I/O segment below 2.5 V for I/O mode
	[19:18]	Reserved	Reserved
	[20]	DFT_DISABLE	– 0: scan and bist available – 1: scan and bist only available on an O
	[31:21]	Reserved	Reserved

Ask AI

On the firmware level, the PWR_SVMCR3.VDDIOxVRSEL register bits must reflect bits of OTP wo
124, so the following sequence must be done before using VDDIO2 I/Os:

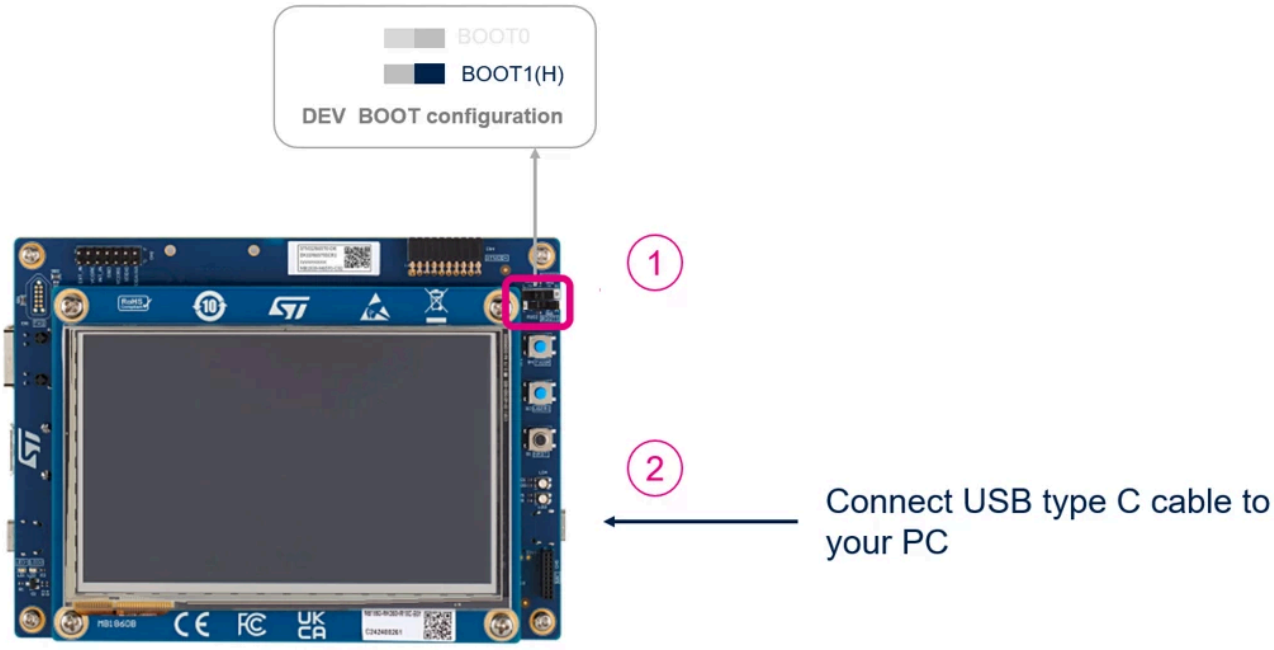
1. If VDDIO2 is independent from VDD:
2. a) Enable the VDDIO2VM by setting VDDIO2VMEN in PWR_SVMCR3.
3. b) Wait for the VDDIO2VM wake-up time.
4. c) Wait until VDDIO2RDY is set in PWR_SVMCR3.
5. d) Optional: Disable the VDDIO2VM for consumption saving.
6. e) If VDDIO2 is in the 1.8 V range: Set the VDDIO2 voltage range by setting VDDIO2VRSEL in PWR_SVMCR3.
7. Set VDDIO2SV in PWR_SVMCR3 to remove the VDDIO2 power isolation.

The same sequence must be done for the VDDIO3 I/Os, using the proper bits within the same register.

2. OTP fuse programming

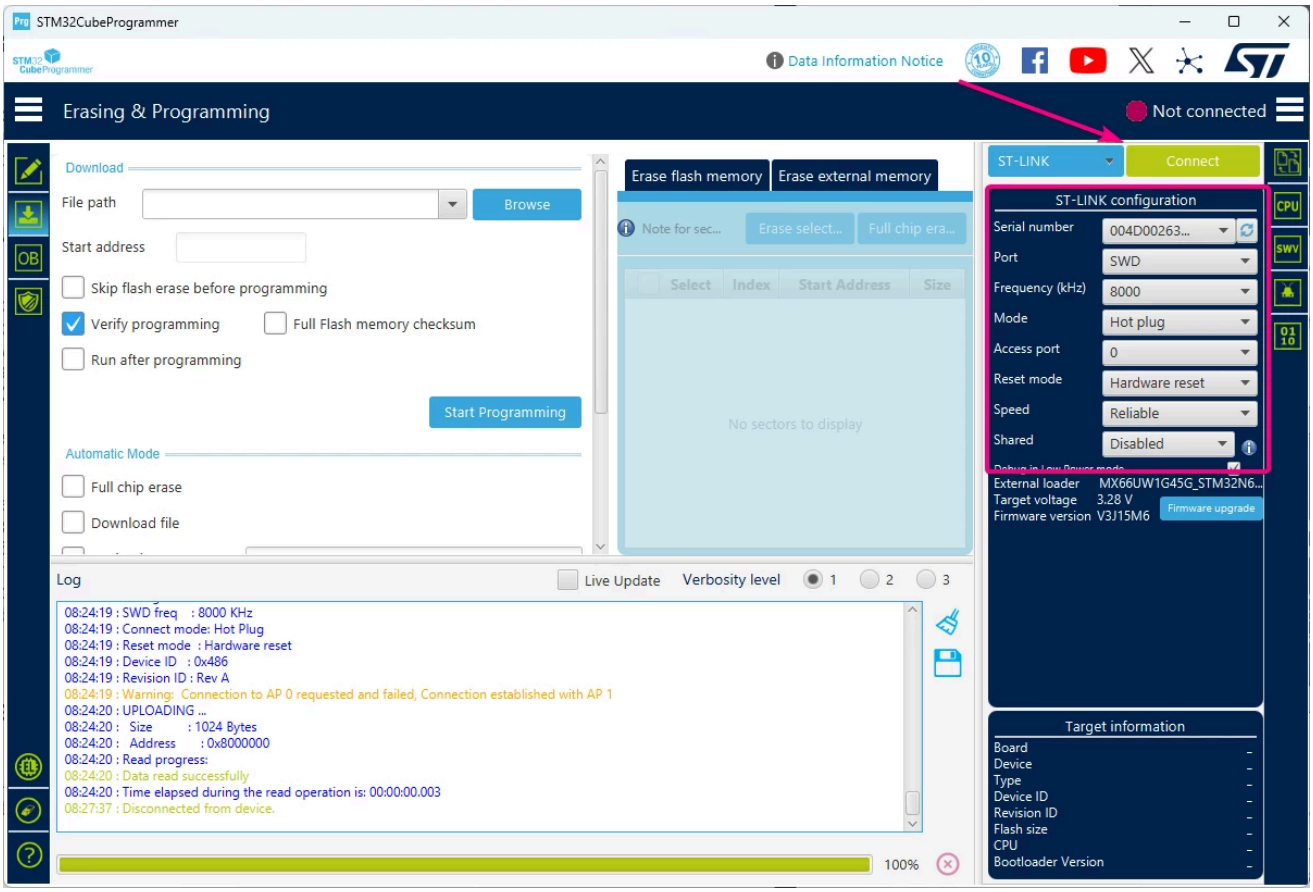
The board should be in Dev Mode to allow it to connect to the STM32CubeProgrammer and program the board. To ensure that the board is in Dev Mode, follow these steps:

1. Set the BOOT configuration to DEV BOOT by setting BOOT0 to LOW and BOOT1 to HIGH.
2. Connect the USB Type-C® cable to your PC.

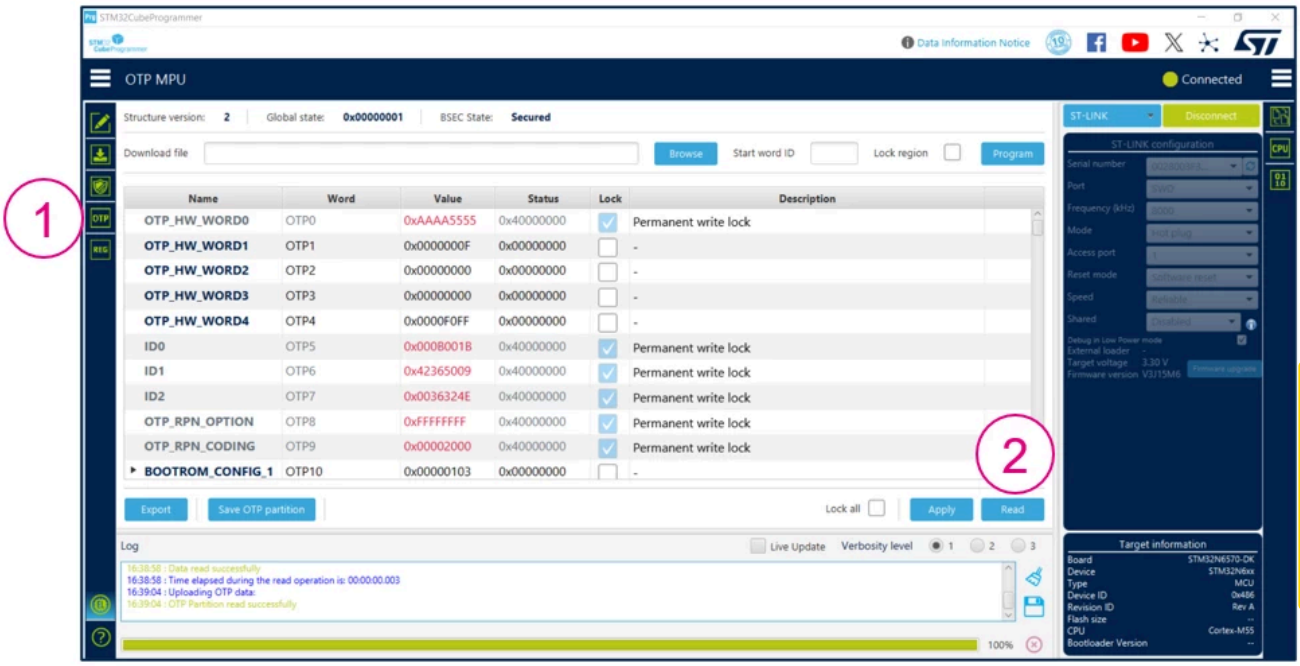


Once connected to the USB, open STM32CubeProgrammer and click **[Connect]**:

Ask AI

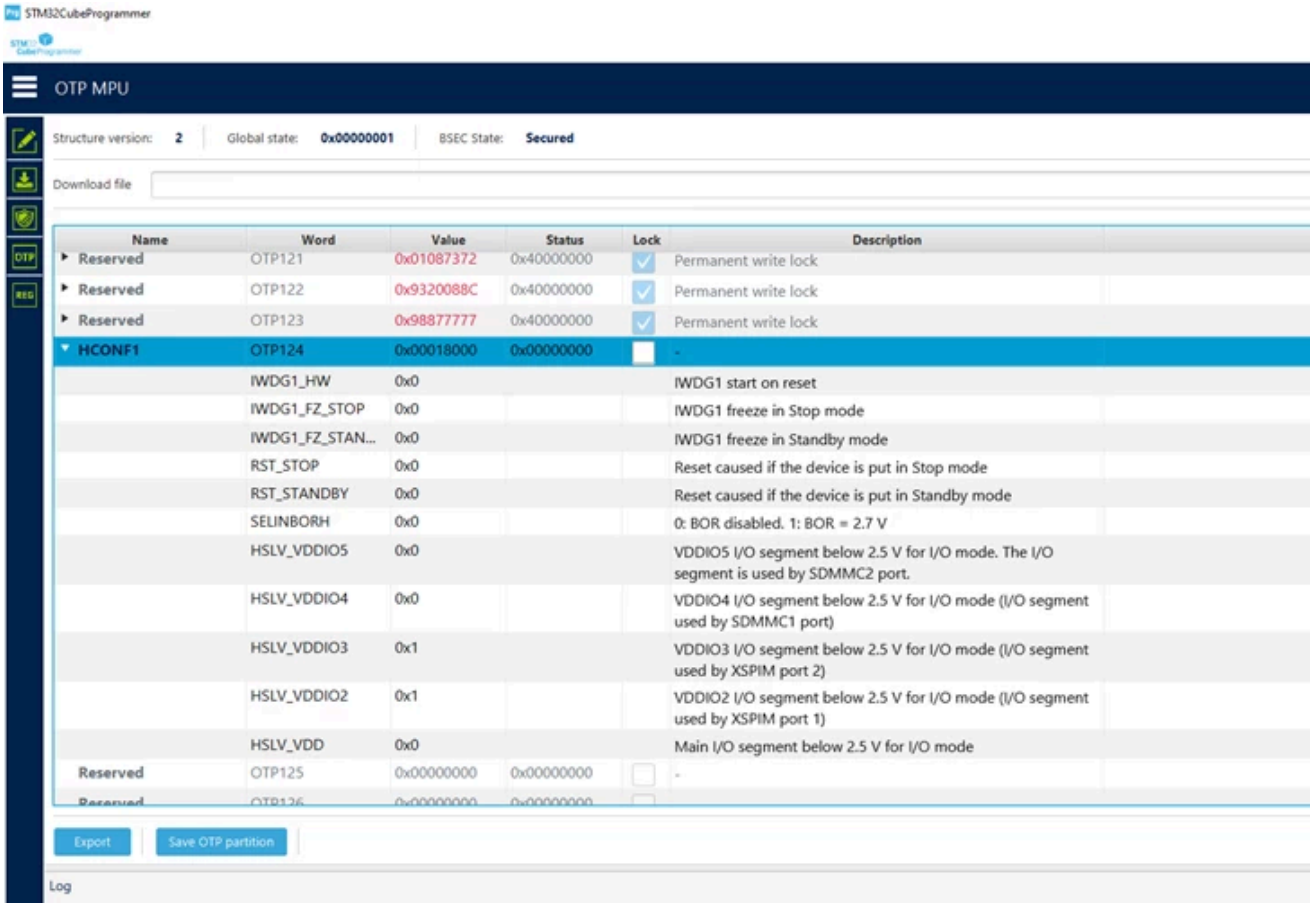


Now, go to the **OTP tab** and click on the **[Read]** button to refresh all values.

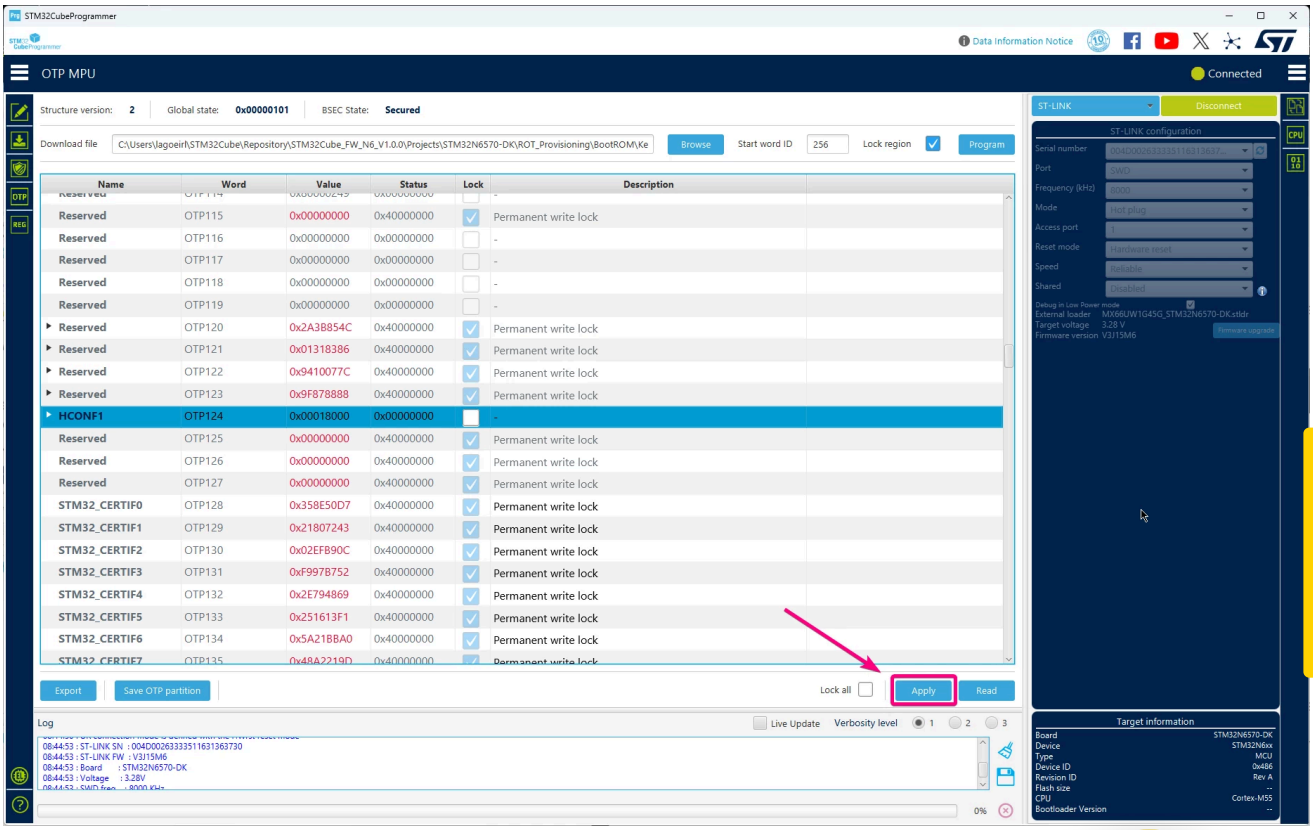


After that, edit the OTP124 (HCONF1) to be **0x0001 8000**, setting the HSLV_VDDIO3 (bit 15) and HSLV_VDDIO2 (bit 16) to 1.

Ask AI



Click on **[Apply]** to update the OTPs modification.



That concludes our example for STM32CubeProgrammer portion. Now, you can rea to verify if the OTP124 has changed.

Ask AI

The same process can be performed via firmware. An example is provided as part of the STM32N6 driver.

The last step that is needed, after performing the OTP change, is to ensure that the firmware can leverage the change. This can be done with a few function calls:

```
/* Set PWR configuration for IO speed optimization */
__HAL_RCC_PWR_CLK_ENABLE();
HAL_PWREx_EnableVddIO3();
HAL_PWREx_ConfigVddIORange(PWR_VDDIO3, PWR_VDDIO_RANGE_1V8);
HAL_PWREx_EnableVddIO2();
HAL_PWREx_ConfigVddIORange(PWR_VDDIO2, PWR_VDDIO_RANGE_1V8);
```

The template for the Discovery kit implements the OTP check. It is followed by the clock configuration to reach the maximum frequency of the external memory (200 MHz) and configure the VDDIO3 to 1.8 V. This example can be found here: STM32CubeN6/Projects/STM32N6570-DK/Templates/Template_FSBL_XIP at main · STMicroelectronics/STM32CubeN6 , specifically under the stm32n6xx_hal_msp.c file, in the function *HAL_XSPI_MspIni()*.

Conclusion

In this article, we explored how to program the OTP fuses in STM32N6 microcontrollers, specifically focusing on OTP124 bits 15 and 16, which are responsible for VDDIO3 and VDDIO2, respectively. By adjusting the hardware to the specific voltage settings and constraints, it is possible to ensure optimal performance and reliability of your embedded applications.

Related links

- Product page: discovery kit with STM32N657X0 MCU
- Reference manual 0486: STM32N647/657xx Arm®-based 32-bit MCUs
- Data brief: STM32N6570-DK - Discovery kit with STM32N657X0 MCU
- User manual 3249: Getting started with STM32CubeN6 for STM32N6 series

STM32CubeProgrammer **Flash** **STM32N6 series**

1

Leave a reply...

5 replies

Sort by: **Oldest first**

Will_Robertson
Associate III · 8 months ago

Ask AI

Hi @B.Montanari and @Laurids_PETERSEN ,

Thank you very much - we're planning to use external PSRAM with the STM32N6x7 for time critical functionality (data for neural network inference on the Neural_ART accelerator and buffers and reference frame for the VENC) for a new open source wildlife camera for our research on endangered European dormouse and squirrel species so your article has been a big help to us in understanding optimising external RAM performance:

https://new-homes-for-old-friends.cairnwater.com/quick-summary-new-homes-for-old-friends-belgium/

Many thanks!

Will

Mikk Leini
Senior · 8 months ago

Slightly confusing because on my DK the HSLV_VDDIO2 and 3 OTP bits are already set. Fancy graphics demo that was on board must have used external PSRAM and Flash at high speeds also.

1

Will_Robertson
Associate III · 8 months ago

Hi @Mikk Leini ,

Thank you very much - that's a good point - just checked them on my board and HSLV_VDIO3 and HSLV_VDIO2 are both set to 0x1 there as well - it looks like you're right - the demo that comes with the board or the STM32N6-GettingStarted-ObjectDetection example must set them. The STM32N6-GettingStarted-ObjectDetection certainly makes use of a large double buffer in PSRAM for the foreground and I'd guess that the demo that comes with the board also makes heavy use of PSRAM.

Will

double6318531
Visitor II · 8 months ago

Ask AI

Hi B.Montanari,

Thanks for your detail description about OTP operation. On our own PCBA , we use STM32N657L0. Now I try to modfiy OTP19 to config PA8 as BOOT1 pin , becasuse PA6 (default pin) used as SPI6_MISO， I use STM32cubeProgrammer to operate OTP19.

According to spec as below: i calculate OPT19 0x1020 0000

Bit Field (Bits) Name Value Remarks

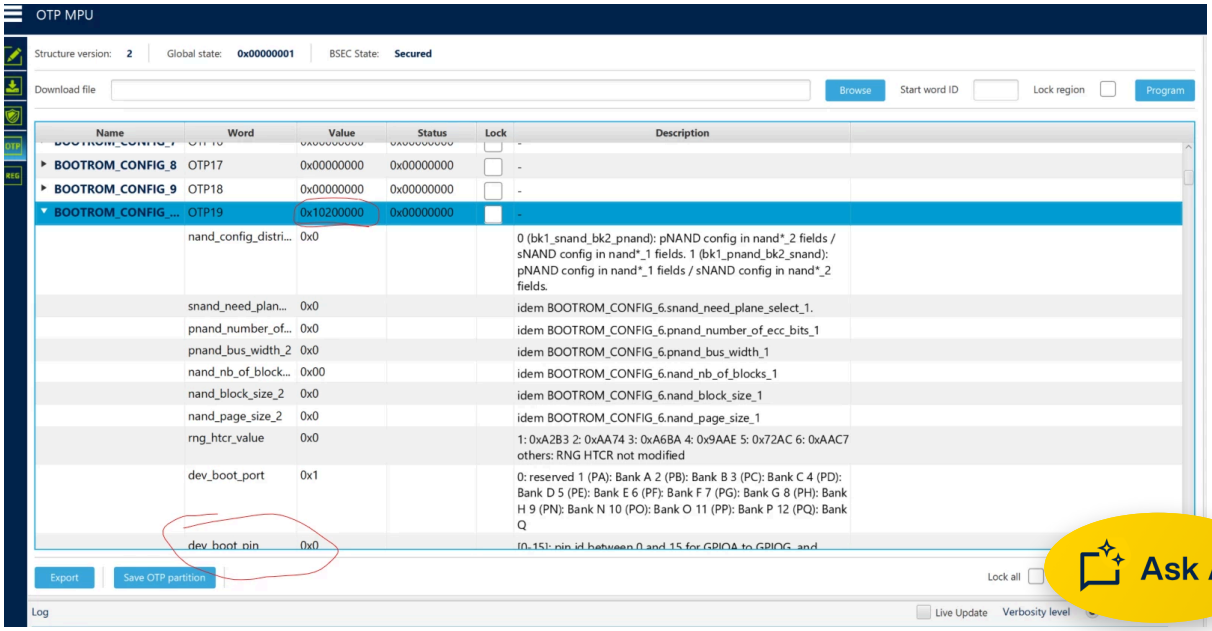
[17:0]	Reserved		All reserved bits must be written as 0
[20:18]	mg_hicr_value	0	You did not specify a value. Using the default "others", meaning "RNG HTCR not modified".
[24:21]	dev_boot_port	1	1 corresponds to **PA** (Bank A).
[28:25]	dev_boot_pin	8	For GPIOA, Pin ID 8 directly corresponds to the numerical value 8.
[31:29]	Reserved	0	All reserved bits must be written as 0

Ask AI

Table 18. OTP fuse description (lower OTP region) (continued)

OTP word	OTP bits	Name or description	Detailed description
OTP19	-	BOOTROM_CONFIG_10	-
	[17:0]	Reserved	Reserved
	[20:18]	rng_htcr_value	– 1: 0xA2B3 – 2: 0xAA74 – 3: 0xA6BA – 4: 0x9AAE – 5: 0x72AC – 6: 0xAAC7 – others: RNG HTCR not modified
	[24:21]	dev_boot_port	– 0: reserved – 1 (PA): Bank A – 2 (PB): Bank B – 3 (PC): Bank C – 4 (PD): Bank D – 5 (PE): Bank E – 6 (PF): Bank F – 7 (PG): Bank G – 8 (PH): Bank H – 9 (PN): Bank N – 10 (PO): Bank O – 11 (PP): Bank P – 12 (PQ): Bank Q
	[28:25]	dev_boot_pin	– [0-15]: pin ID between 0 and 15 for GPIOA to GPIOG, and GPIOP – [0-12]: pin ID between 0 and 12 for GPION – [0-8]: pin ID between 0 and 8 for GPIOH and GPIOQ – [0-5]: pin ID between 0 and 5 for GPIOO
	[31:29]	Reserved	Reserved
OTP20	-	BOOTROM_CONFIG_11	-
	[31:0]	oem_fsbl_monotonic_counter	[1-0xFFFF] → [1-32]: Value of monotonic counter is X, where X is the position of the most significant bit at 1
OTP21	-	BOOTROM_CONFIG_12	-
	[31:0]	oem_fsbl_monotonic_counter	[1-0xFFFF] → [33-64]: Value of monotonic counter is 32 + X, where X is the position of the most significant bit at 1
OTP22	-	BOOTROM_CONFIG_13	Reserved
OTP23	-	BOOTROM_CONFIG_14	Reserved

but i read OTP19 via STM32cubeProgrammer after configuring OPT19 to 0x10200000, the field dev_boot_pin is still 0 (display). anything wrong ?



1 reply

Richu Bini
Associate · 12 days ago

We were also facing the slimier issue when try to chnage the boot1 pin to PN10

ST Community highlights – April to June 2026



Related topics

How to debug STM32N6 TouchGFX applications using STM32CubeIDE
STM32 MCUs

First project on STM32N6750-DK
STM32 MCUs Boards and hardware tools

How to understand the STM32N6 using STM32Cubemx
STM32CubeMX (MCUs)

STM32N6 Nucleo XIP clock frequency problem.
STM32 MCUs Embedded software

STM32N6 - booting off Quad spi (OTP) settings
STM32 MCUs Products

Popular tags

STM32H7 series | STM32F4 series | STM32CubeMX | STM32CubeIDE | TouchGFX | STM32L4 series | STM32F7 series | USB | UART-USART | STM32Cube MCUs packages

Ask AI



About STMicroelectronics

Who we are

Investor relations

Sustainability

Innovation & technology

Careers

Suppliers to ST

Blog

General terms and conditions

Connect with us

Contact ST offices

Find sales offices & distributors

Community

Newsroom

Events & trainings

AI at ST

Browse

Shortcuts

Sitemap

Compliance, Ethics, Security & Privacy

Ethics and compliance

ST Ethics Hotline

ST's Code of Conduct

Security & privacy portal

Ask AI

Follow us

All rights reserved © 2026 STMicroelectronics | Terms of use | Sales terms & conditions | Trademarks | Privacy portal | Manage cookies

Ask AI