CAN - Flash Documentation

For full explanation of prior needed concepts see:
https://docs.google.com/document/d/1RplHiQblHJfaAxLyOGzX1HnmwUP_ogc84lS5asevDy8/edit?usp=sharing

Terminology:

BOOT_ADD0 & BOOT_ADD1	An option byte (programmable setting) ; set in STM32 Cube Programmer
Boot Pin	A hardware pin on the STM32F767ZI ; controls which memory area to boot into
STM32 Cube Programmer	An application which allows the flashing, and memory manipulation of STM32 MCU's
System Bootloader	A bootloader made by STM located at memory address: 0x0010 0000. Enables flashing the MCU via ***CAN***, UART, SPI and more…

Setup:

1. Boot Option Bytes:
    a. There are two option bytes relevant to CAN Flash, these are BOOT_ADD0 and BOOT_ADD1
    b. Set BOOT_ADD0 to value 0x80 (address 0x0020 0000), and BOOT_ADD1 to value 0x40 (address 0x0010 0000) in Option Bytes of STM32CubeProgrammer. If option byte values are already configured as such, leave as is. See Figure 1 to see different boot addresses
    c. These options bytes are chosen by the firmware on reset via the boot pin. If the boot pin is LOW, BOOT_ADD0 is chosen and vice versa.
    d. <u>Verify:</u>
        i. Tie the boot pin to high ; choose BOOT_ADD1
        ii. In STM32 Cube Programmer: go into MCU Core tab > press **<u>any</u>** reset option
        iii. In STM32 Cube Programmer: go into MCU Core tab > press run and then halt
        iv. In STM32 Cube Programmer: go into MCU Core tab > check if the LR (instruction pointer) register is in the memory range of 0x1FFX XXXX
        v. If so, the MCU is currently in system bootloader

2. Read/Write protection option bytes:
    a. While BOOT_ADD0 and BOOT_ADD1 are the most important, other option bytes must be left in their **DEFAULT** to enable reading and writing to flash memory
        i. These are the write protection option bytes: nWRP# and read protection option byte: RDP
        ii. Ensure all these option bytes are left as their **default configuration**. See Figure 2 for default option bytes.
        iii. **For RDP: DO NOT CHANGE IT TO CC (Level 3 Protection) → DEBUG OPTIONS WILL BE DISABLED FOREVER**
    b. **Verify:**
        i. Check with Figure 2
        ii. In STM32 Cube Programmer : go into Erasing & Programming tab > do a full chip erase for flash memory
        iii. If it doesn't work, your read/write protections are set wrong

3. Can-prog download, and raspberry pi setup:
    a. In order to program the STM32 MCU with can-prog you will need a computer with a CAN interface → in testing we used a raspberry pi with a CAN Hat for our CAN bus setup
    b. For flashing with CAN we will need a bitrate of 125000 for our CAN network
        i. Using `ifconfig` you can check which networks are set up on your system → you should see either can0 or can1 ; either is fine to use, we will use can1
        ii. **Verify:** Using `ip -details -statistics link show can1` ; see if the CAN network is configured at the required bitrate of 125000
            1. If so, skip the steps below
        iii. Using `sudo ip link set can1 down` ; pull the CAN network down
        iv. Using `sudo ip link set can1 type can bitrate 125000` ; configure the CAN network to use the required bitrate of 125000
        v. Using `sudo ip link set can1 up` ; bring the CAN network up again
    c. Git clone this repo's version of can-prog and install

4. CAN - 2 Wiring, BOOT pin setup and CAN Transceiver:
    a. You will **need** a **CAN transceiver** for the STM32. We **require** the **CAN-2** interface on the STM32, specifically pins **PB_5** for CAN_RX and **PB_13** for CAN_TX.
    b. Drive BOOT pin high (See Figure 3)

c. Connect the CAN bus by connecting **CAN2_RD** from **PB_5** and **CAN2_TD** from **PB_13** into **CAN_RX** and **CAN_TX** on the CAN transceiver. Note that there are other CAN2_RD and CAN2_TD pins, but it must be from PB_5 and PB_13. Also note that there are multiple PB_5 and PB_13 pins, and that PB_5 and **PB_13** must be connected to the ones in Figure 4.a OR Figure 4.b, but must not be combined (i.e. do not connect PB_5 from Figure 4.a, and PB_13 from Figure 4.b)

5. Vector Table Offset (OPTIONAL):
    a.

Procedure:

1. Write the binary to the board
    a. i.e ***canprog -n can0 -f bin stm32 write [file_name] -a 0x08000000***
        i. -n → interface name
        ii. -f → file format (hex, bin)
        iii. -a → address to write → default is 0x0800 0000 DON'T CHANGE
        iv. [file_name] → file to write
2. Run the flashed application
    a. i.e ***canprog -n can0 stm32 go -a 0x08000000***

<u>Notes:</u>

- Unlock, lock, and speed commands with can-prog tool are not necessary
    - The unlock and lock commands change the readout protection and option bytes ; in our use case we do not need to lock the chip after flashing → therefore do not use
    - The speed command changes the bitrate of the CAN-2 interface on the STM32, however the default is already set correctly upon system reset → therefore do not use

Appendix

Bits 31:16 **BOOT_ADD1[15:0]**: Boot base address when Boot pin =1

BOOT_ADD1[15:0] correspond to address [29:14],
The boot memory address can be programmed to any address in the range 0x0000 0000 to 0x2004 FFFF with a granularity of 16KB.
Example:
BOOT_ADD1 = 0x0000: Boot from ITCM RAM (0x0000 0000)
BOOT_ADD1 = 0x0040: Boot from System memory bootloader (0x0010 0000)
BOOT_ADD1 = 0x0080: Boot from Flash on ITCM interface (0x0020 0000)
BOOT_ADD1 = 0x2000: Boot from Flash on AXIM interface (0x0800 0000)
BOOT_ADD1 = 0x8000: Boot from DTCM RAM (0x2000 0000)
BOOT_ADD1 = 0x8004: Boot from SRAM1 (0x2002 0000)
BOOT_ADD1 = 0x8013: Boot from SRAM2 (0x2004 C000)

Bits 15:0 **BOOT_ADD0[15:0]**: Boot base address when Boot pin =0

BOOT_ADD0[15:0] correspond to address [29:14],
The boot base address can be programmed to any address in the range 0x0000 0000 to 0x2004 FFFF with a granularity of 16KB.
Example:
BOOT_ADD0 = 0x0000: Boot from ITCM RAM (0x0000 0000)
BOOT_ADD0 = 0x0040: Boot from System memory bootloader (0x0010 0000)
BOOT_ADD0 = 0x0080: Boot from Flash on ITCM interface (0x0020 0000)
BOOT_ADD0 = 0x2000: Boot from Flash on AXIM interface (0x0800 0000)
BOOT_ADD0 = 0x8000: Boot from DTCM RAM (0x2000 0000)
BOOT_ADD0 = 0x8004: Boot from SRAM1 (0x2002 0000)
BOOT_ADD0 = 0x8013: Boot from SRAM2 (0x2004 C000)

Figure 1

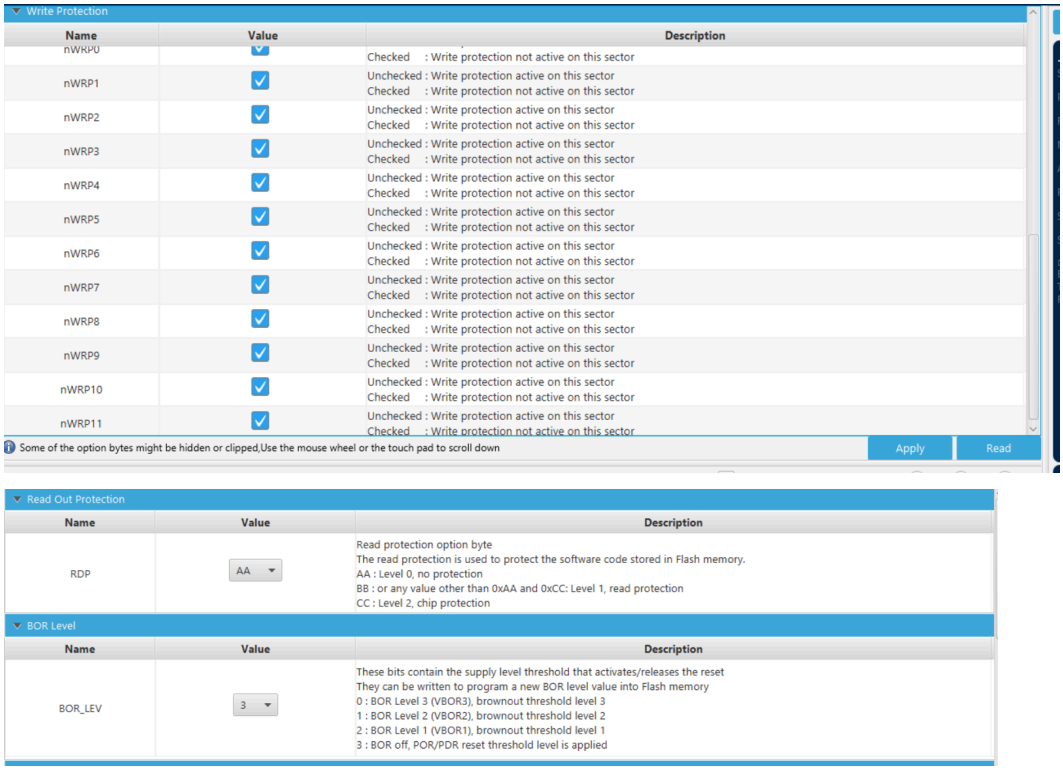

Figure 2: Default Option Bytes

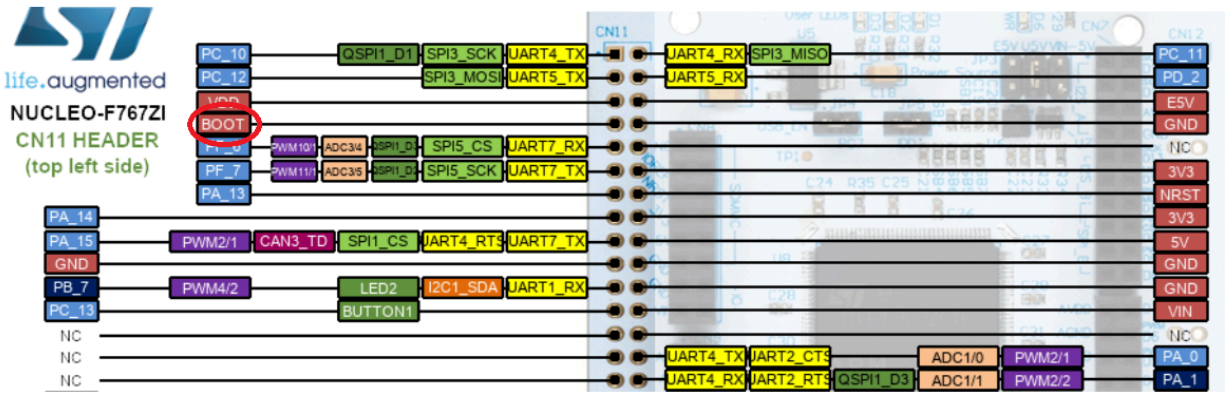

Figure 3: Boot pin pinout for STM32F767z

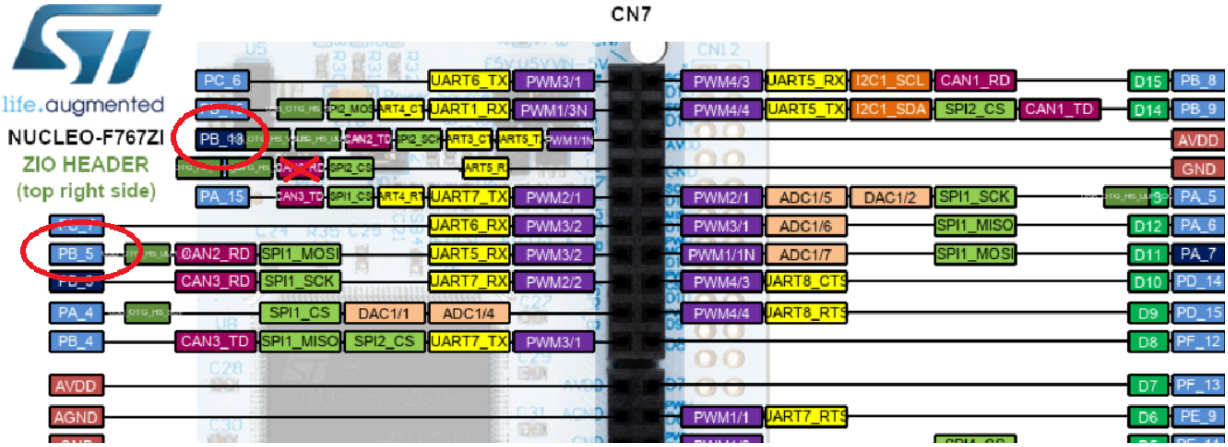

Figure 4. a: PB_5 and PB_13 pinout

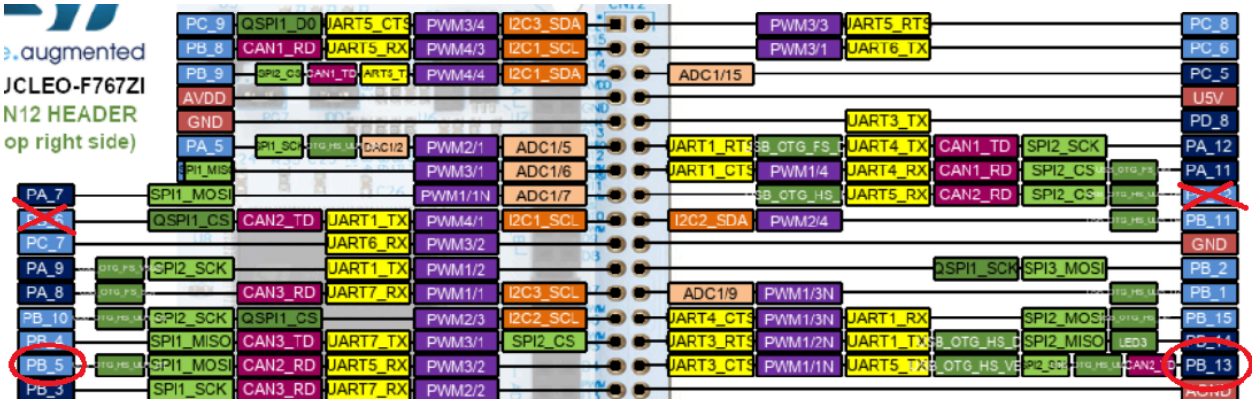

Figure 4. b: PB_5 and PB_13 pinout