life.augmented

DT0118
Design tip

A Procedure for Restoring Bootloader Functionality to the STNRG388A

9

By Dennis Nolan

Main component	
STNRG388A	Digital controller for power conversion applications with up to 6 programmable PWM generators, 96 MHz PLL

Purpose

The STNRG388A may have up to 128 non-volatile option bytes which control many aspects of the chip configuration. One of these aspects is whether or not the chip's bootloader functionality is enabled. While the default (fresh from the factory) configuration is to enable the bootloader, there are many scenarios in the life of a programmable chip which might leave the part with its bootloader disabled. The purpose of this note is to provide a simple, straight-forward, and reliable process by which the boot loading functionality can be restored. One piece of hardware, the ST-LINK programming dongle, will be required. This programmer is quite inexpensive and readily available from ST or its distributors. The software support is all free and available on the ST web site www.st.com. Just search for "stvp-stm8".

Hardware and software requirements

While the STVP is capable of programming the program flash memory (0x8000 to 0xFFFF), it is not capable of directly programming all of the required option bytes. Therefore, this process loads an executable into the program flash memory space which, when executed, uses In Application Programming to program the required option bytes.

Hardware Requirements:

- An eval board or prototype using the STNRG388A.
- An ST-LINK programming dongle with appropriate programming cable and USB cable.

Software Requirements:

- ST Visual Programmer (STVP-STM8) running on your PC.

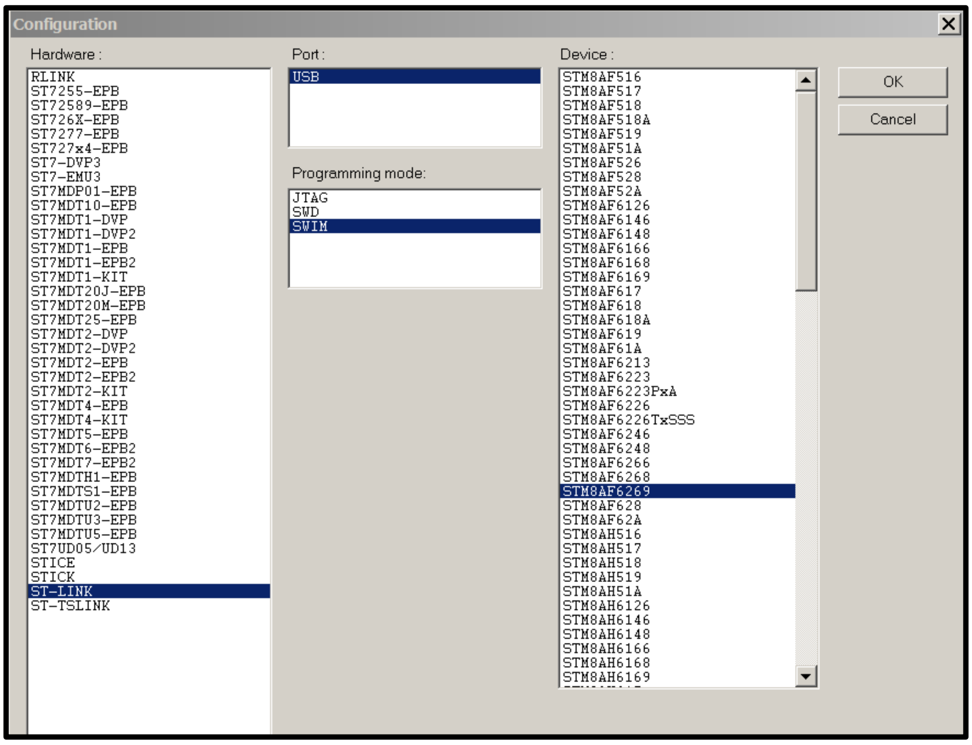
Step by step instructions

Create a simple Intel hex file by copying the following lines into a new plain txt file named “options-bl.hex” and save it in a convenient location.

```
:0480000082008080FA
:1080800035FF500235FF500335FF5004355650621E
:1080900035AE506255505F000072030000F63500A7
:1080A000800035AE50643556506455505F00007204
:1080B000070000F63580505B357F505C3511481560
:1080C00055505F000072050000F635EE4816555019
:1080D0005F000072050000F635AA487E55505F002B
:1080E000000072050000F63555487F55505F0000725C
:0480F000050000F691
:0F80F400A601C85000B70055000050002 0F281CF
:00000001FF
```

Open the STVP application, click on CONFIGURATION, and select the configuration as shown in Figure 1. Please note the STVP, while it is an excellent tool, is a legacy programmer which was never updated specifically for the STNRG388A. Nevertheless, the STM8AF6269 is sufficiently similar to the STM8 core built into the STNRG388A that this selection will work for our requirements. Also, be patient, STVP can take 15 to 20 seconds to start execution.

Figure 1.



Hardware : ST-LINK

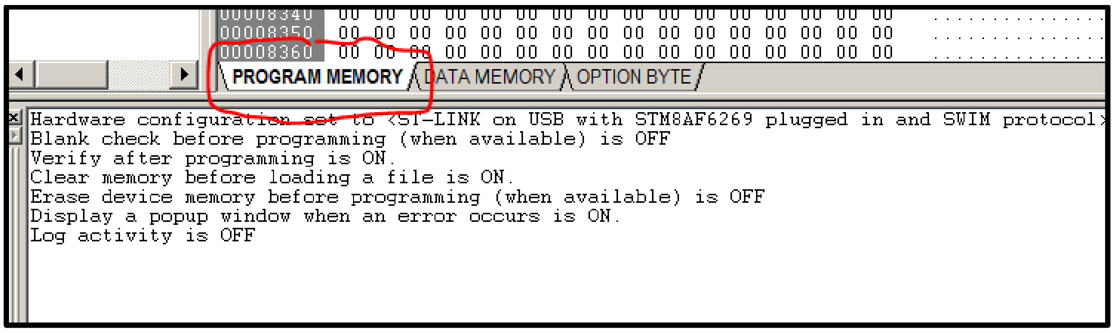
Port : USB

Programming mode: SWIM

Device : STM8AF6269

Select the PROGRAM MEMORY tab, as shown in Figure 2

Figure 2.



```
00008340  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00008350  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00008360  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

```
Hardware configuration set to <ST-LINK on USB with STM8AF6269 plugged in and SWIM protocol>
Blank check before programming (when available) is OFF
Verify after programming is ON.
Clear memory before loading a file is ON.
Erase device memory before programming (when available) is OFF
Display a popup window when an error occurs is ON.
Log activity is OFF
```

Select FILE->OPEN, navigate to wherever you have saved your “options-bl.hex” file, and select it.

The message window at the bottom of the screen should indicate that the file was loaded successfully.

Connect the ST_LINK programmer to the target PCB and power-up the board, as shown in Figure 3 and Figure 4.

Figure 3.

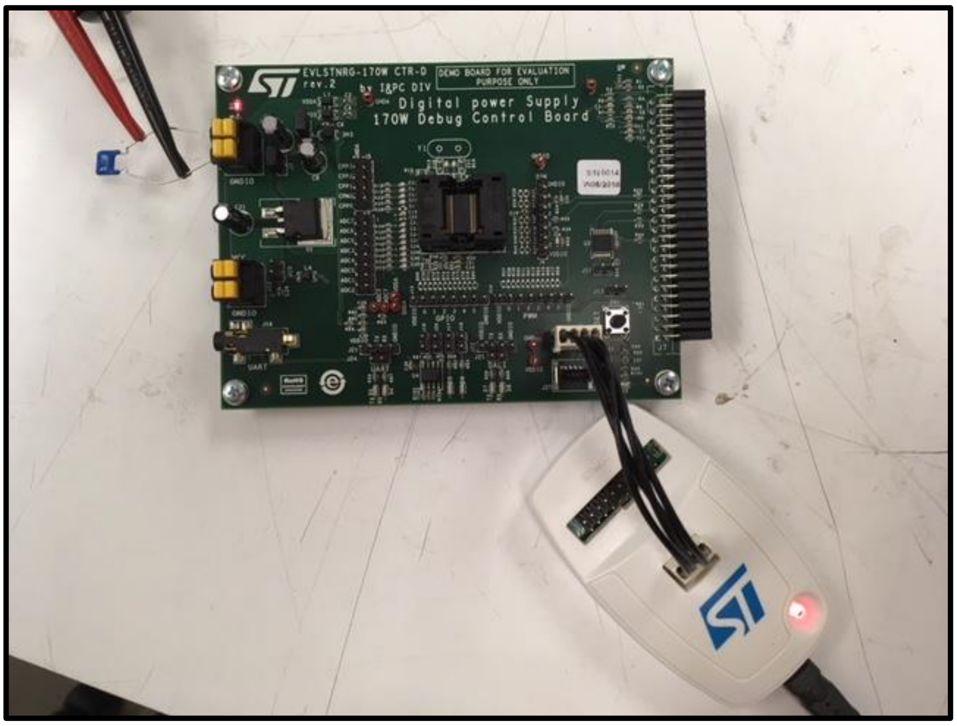
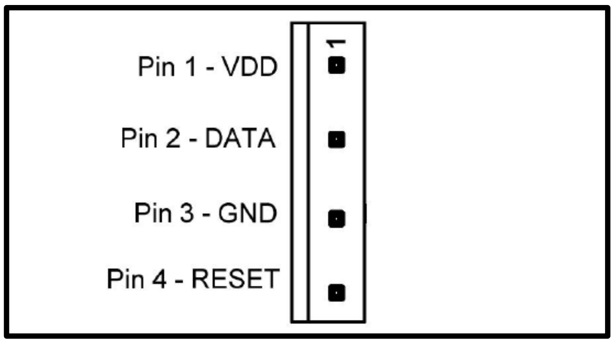


Figure 4.

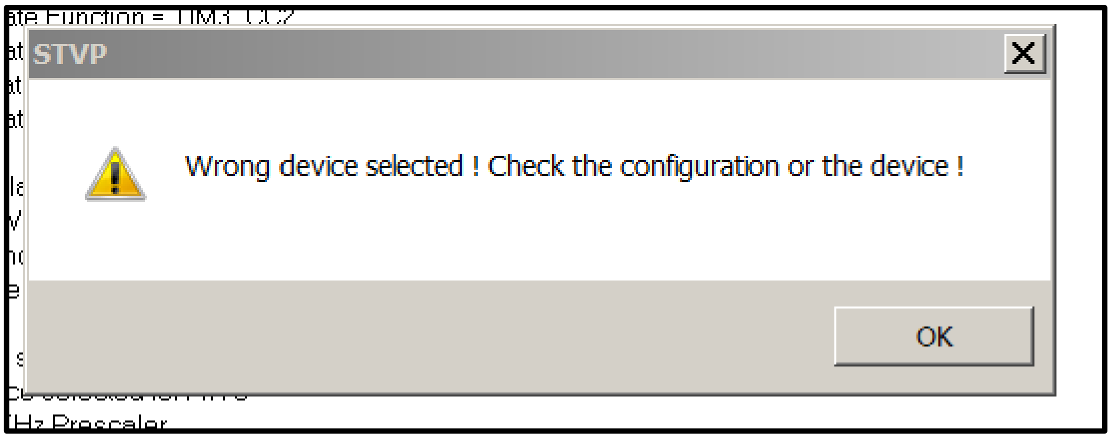


Select PROGRAM->Current Tab.

You will get the following warning, as shown in Figure 5, because the device ID does not match.

Ignore this and click OK.

Figure 5.

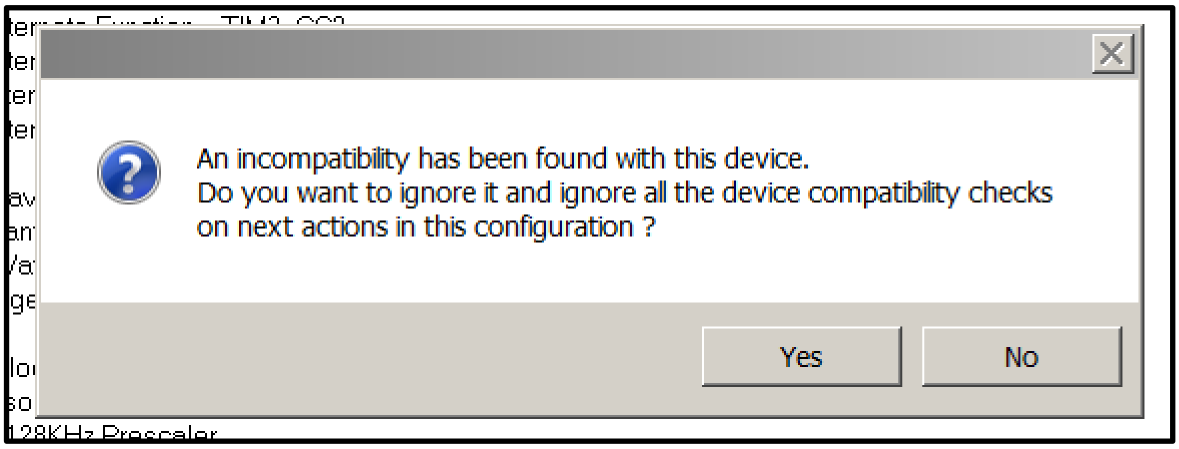


STVP

Wrong device selected ! Check the configuration or the device !

You will get the following dialog box.

Figure 6.

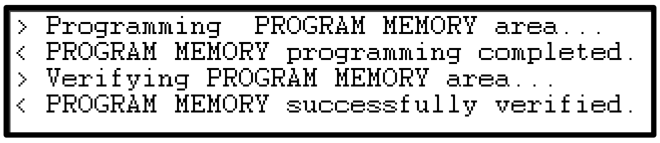


An incompatibility has been found with this device.
Do you want to ignore it and ignore all the device compatibility checks on next actions in this configuration ?

Click yes to bypass additional compatibility checks.

At the bottom of the message window, you should now receive the following message There is a small chance that you may get a verify error, but this should be ignored.

Figure 7.



```
> Programming  PROGRAM MEMORY area...
< PROGRAM MEMORY programming completed.
> Verifying PROGRAM MEMORY area...
< PROGRAM MEMORY successfully verified.
```

That's all there is to it! You can now power-down the board and disconnect the ST-LINK programmer. Cycling power to your board will reset and run the program to restore bootloading functionality. Note: Since the STNRG388A requires very little power, it is actually possible that the chip could receive enough current from the com port lines coming from your PC to keep it powered up and prevent the reset. To prevent this, it is good practice to also disconnect from the PC when you power down the board.

Since this program restores "virgin" status to the part, you do not need to meet the one second timeout period described in documentation of the general bootloader function. The part will wait indefinitely for a signal from the external bootloading device.

Troubleshooting

If you are having trouble getting the bootloader to function, you may have some question as to whether the chip programming process (using the STVP) has been successful. To test this functionality, and gain some confidence, you can use the text provided below, which is a slightly modified version of the first program. If you follow the same procedure to load this program, when it runs you will be able to observe a "hello world" signal at GPIO0.0 which is a 50% duty cycle square wave with a period of about 7 microseconds.

```
:0480000082008080FA
:1080800035FF500235FF500335FF50043556650621E
:1080900035AE506255505F000072030000F635AEF9
:1080A00050643556506455505F000072070000F66A
:1080B0003580505B357F505C3511481555505F0059
:1080C0000072050000F635EE481655505F0000724C
:1080D000050000F635AA487E55505F000072050085
:1080E00000F63555487F55505F000072050000F6D8
:0F80F000A601C85000B700550000500020F281D3
:00000001FF
```

Complete program source code

The following text is the entire source code (written for the Raisonance STM8 C compiler) for the first program. The only difference in the second program is that the line of code which writes 0 to the variable "firstflash" has been commented out. Clearing this flash memory byte at 0x8000 is necessary to put the chip back into "virgin" state but doing so also trashes the vector table. Because of this, the program actually destroys itself and will only execute once. With the line commented out, everything else runs and can continue to run whenever the board is reset. This includes the background loop at the end, which generates the hello world signal.

A word of warning. If you wish to compile and run this program it is self-contained, however, be sure that you turn off all optimization of the compiler you are using. In order to unlock memory, the STM8 family requires that two unique keys be written in quick succession to the same memory address. Even if the variable is declared volatile, many C compilers will eliminate the first write, considering it unnecessary.

```
// this program will enable bootloading with an infinite wait time
// this program is specifically for the STNRG388A

#define b0  1
#define b1  2
#define b2  4
#define b3  8
#define b4  16
#define b5  32
#define b6  64
#define b7  128

at 0x4815 volatile char ob4815 ;
at 0x4816 volatile char ob4816 ;
at 0x487E volatile char ob487e ;
at 0x487F volatile char ob487f ;

at 0x5000 volatile char gpio0odr;
at 0x5001 volatile char gpio0idr;
at 0x5002 volatile char gpio0ddr;
at 0x5003 volatile char gpio0cr1;
at 0x5004 volatile char gpio0cr2;

at 0x505B volatile char flashcr2;
at 0x505C volatile char flashNcr2;
at 0x505F volatile char flashiapsr;

at 0x5062 volatile char pukr;
at 0x5064 volatile char dukr;

at 0x8000 volatile char firstflash;

void main ( void )
{
// set up port 0
gpio0ddr = 255;
gpio0cr1 = 255;
gpio0cr2 = 255;

// unlock flash and clear first byte ( virgin part )
pukr = 0x56;
pukr = 0xAE;
while( (flashiapsr & b1) == 0 ) ;  // wait for memory to unlock
firstflash = 0;

// unlock EEPROM and option bytes
dukr = 0xAE;
dukr = 0x56;
while( (flashiapsr & b3) == 0 ) ;  // wait for memory to unlock

// enable writes to option bytes
flashcr2 = 0x80;
flashNcr2 = 0x7F;

// set up option bytes to enable bootloader
ob4815 = 0x11;
while( (flashiapsr & b2) == 0 ) ;  // wait for write to complete

ob4816 = 0xEE;
while( (flashiapsr & b2) == 0 ) ;  // wait for write to complete

ob487e = 0xAA;
while( (flashiapsr & b2) == 0 ) ;  // wait for write to complete

ob487f = 0x55;
while( (flashiapsr & b2) == 0 ) ;  // wait for write to complete

while(1)  // background loop
{
gpio0odr ^= 0x01;
}

}  // end of main
```

Support material

List any related support material such as evaluation boards, Gerber files etc. and any documents which might be useful to the customer, for example the datasheets, the evaluation board user manual etc.

Related design support material
EVLSTNRG-170W, 170W SMPS with digitally controlled PFC and resonant LLC stage using STNRG388A
ST-LINK/V2, ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32
Documentation
Datasheet, STNRG388A, Digital controller for power conversion applications with up to 6 programmable PWM generators, 96 MHz PLL

Revision history*

Date	Version	Changes
7/19/18	1	Initial release