Nintendo Alarmo
“1stloader” firmware

Rev. 2
2025/02/09

Author: Tim Schuerewegen

What can you do with the “1stloader” firmware?

- Launch original Alarmo encrypted “system” and “factory” firmware.

- Access the 3GB and 512MB file systems on the eMMC via USB.

- Install custom Alarmo themes.

- Install and launch homebrew.

- Launch homebrew from the USB RAM disk.

- Autoboot whatever you want.

Nintendo Alarmo

The Nintendo Alarmo contains an STM32 MCU, 4GB eMMC, 32MB external RAM, 240x320 LCD, speaker, ambient light sensor, Wi-Fi, radar, speaker, and more.

The STM32 MCU has 128KB of flash. It contains a secure firmware that cannot be dumped.

The eMMC contains a 3GB and 512MB file system. It contains encrypted firmware files and theme files.

Boot process

The very simplified Nintendo Alarmo boot process is as follows:

1. The STM32 MCU executes the firmware stored on the STM32 flash. It mounts the 3GB file system, reads and decrypts the “2ndloader.bin” firmware, and executes it.

2. The “2ndloader.bin” firmware mounts the 3GB file system, extracts the “a.bin” firmware from the encrypted “system.shpac” archive, and executes it.

3. The “a.bin” firmware initializes all the hardware, displays a loading screen, and then displays the Alarmo clock.

STM32 security

The STM32 MCU offers several security features and they are configured as follows:

- The RDP level is set to 1.

- The security bit is enabled.

- The secure area covers the entire 128KB of flash.

Due to the configured STM32 security it is not possible to read the contents of the STM32 flash or connect via ST-LINK/SWD before the “2ndloader.bin” firmware gets executed. Should the “2ndloader.bin” file become missing or corrupt, it is very likely that the STM32 flash firmware will crash and thus making it impossible to connect via ST-LINK/SWD and repair the damage. The STM32 MCU will be (semi-)permanently bricked unless you can find a way to reprogram the eMMC by desoldering the chip and soldering wires to it.

Secret AES key and IV

The “1stloader” firmware needs to know the secret Alarmo AES key and IV or else it cannot decrypt and run the original Alarmo clock firmware. It is possible to extract the AES key and IV from the STM32 MCU thanks to some very smart people.

https://github.com/GaryOderNichts/alarmo

How to remove the STM32 security?

In order to install a different STM32 flash firmware you first need to remove the STM32 security. It will also erase the STM32 flash. This process is dangerous and can permanently brick the STM32 MCU when not done correctly. If this happens then no repairs are possible and you will have to replace the STM32 MCU. You have been warned.

The process to remove the STM32 security is described in section 2.4.2 (step 1) and section 2.4.3 (step 2) of the AN5601 “HDP secure area for STM32H7B3xx microcontrollers” application note. This process is very dangerous and can permanently brick the STM32 MCU. You have been warned, again.

https://www.st.com/resource/en/application_note/an5601-hdp-secure-area-for-stm32h7b3xx-microcontrollers-stmicroelectronics.pdf

I have created a video where you see me using STM32CubeProgrammer to remove the STM32 security and installing the “1stloader” firmware.

https://youtu.be/oPXoK3NROqo

Note 1: The first time I perform step 1 (section 2.4.2) it does not work, but the second time it does.

Note 2: I use values “0xff” and “0x00” for the secure area start/end. Those values will work too, because “0xff” is higher than “0x00”. That is very important.

How to install the “1stloader” firmware

1. Put the secret Alarmo AES key and IV in the “1stloader.cfg” file. If you do not have this key/IV then do not continue.

2. Use STM32CubeProgrammer to remove the STM32 security. This will erase the STM32 flash.

3. Program the STM32 flash with the “1stloader” firmware .hex file.

4. Use “USB DISK EMMC” to get access to the 3GB and 512MB file systems and make a complete backup of all files and folders on both file systems. Keep this backup safe in case you ever need to restore something.

5. Copy the “1stloader.cfg” file to the “settings” folder on the 512MB file system.

6. Create a “homebrew” folder on the 3GB file system. This is where you can put homebrew .bin and .shpac files.

7. Unplug the USB cable and plug it back in.

8. Use “LAUNCHER” to launch “system.shpac”. This is the Alarmo clock firmware. If it does not work then either the “1stloader.cfg” file is missing or the AES key/IV is wrong.

How do I launch homebrew from USB RAM disk?

- Copy “a.bin” or “a.shpac” to the USB RAM disk.

- Copy an empty “MarkFile” file to the USB RAM disk to launch the homebrew.

Explanation of “1stloader.cfg” options

```
start            = autoboot
aes_key          = 0123456789abcdef0123456789abcdef
aes_iv           = 0123456789abcdef0123456789abcdef
color_text       = ffffff
color_background = ff0000
lcd_backlight    = 255
autoboot_file    = 0:/contents/system.shpac
autoboot_time    = 6
show_format_emmc = 0
```

start - specifies what to show on boot, possible values are "autoboot", "launcher", "usb_disk_emmc" and "usb_disk_ram".

aes_key and ***aes_iv*** - secret AES key and initialization vector necessary to decrypt the original Alarmo system/factory firmware.

color_text and ***color_background*** - colors to use for the text and background.

lcd_backlight - backlight brightness, possible values from 0 to 255.

autoboot_file - specifies what autoboot should launch, possible values are .bin or .shpac filenames on either the 3GB or 512MB file system.

autoboot_time - countdown time in seconds for autoboot, possible values are 0 to 100.

show_format_emmc - shows or hide the “FORMAT EMMC” option, possible values are 0 or 1.