SMH Technologies
UNIVERSAL PRODUCTION
IN–SYSTEM PROGRAMMING

27/02/2019 Samuele Stefanoni

DC11032

AN00180: Cybersecurity

FlashRunner 2.0 is a Universal In-System Programmer, which feature cybersecurity features. This Application Note describes how to properly set up and understand FlashRunner 2.0 cybersecurity features.

1. Introduction

Cybersecurity is the protection of internet-connected systems, including hardware, software and data, from cyberattacks.

In micro controllers world, especially in industrial manufacturing area, this term has been reinterpreted and assumes several other meanings. Cybersecurity can be related to:

- **Antipiracy**: flashing process is contracted to a third-party company and customers want to be sure that firmware can't be stolen or tampered
- **Traceability**: flashing is usually part of board manufacturing process, which requires high standards in terms of traceability. Flashing outcome result, timestamp of the flashing process are only some of the information which needs to be traced.
- **Integrity**: customer requires proofs of a correct flashing process

2. Antipiracy

FlashRunner features firmware encryption tool which can be used as antipiracy method. Suppose to be a customer that wants to do board manufacturing process to a third-party company, which will use FlashRunner 2.0 to flash the boards.

Soon board manufacturer will require to the customer a firmware to flash into the board microcontroller(s). FlashRunner 2.0 requires firmware conversion in FRB proprietary format. Conversion can be achieved using our GUI Workbench software.

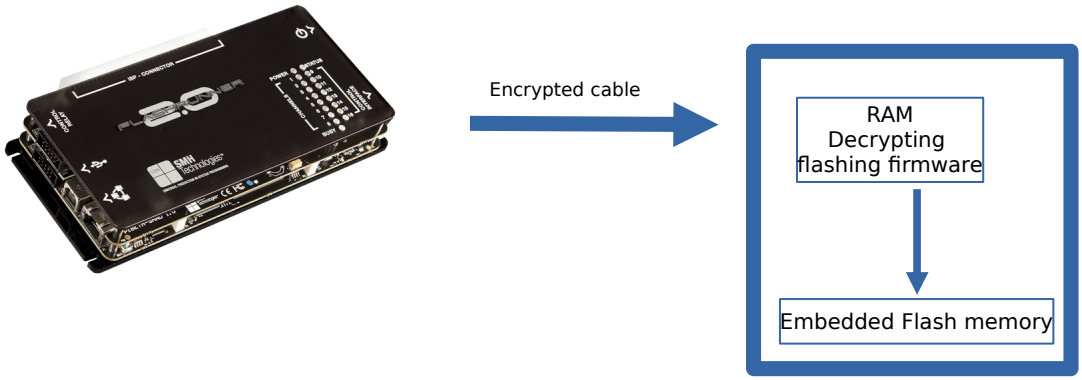
During this conversion process, FRB file can be converted into an encrypted file format, named FRS file (for more information on how to do it please check chapter 3.9 of FlashRunner 2.0 Programmer's Manual available on our website). Encryption method implemented is AES256 achieved through shared pass-phrase. This means that only GUI Workbench software has encryption key and only FlashRunner 2.0 has embedded decryption key, therefore FRS file can't be converted back to original data content, FRS file can only be stored into FlashRunner 2.0 memory.

This simple method provide a secure and efficient way to secure your software against stealing attempts, but it has a limitation: once FlashRunner 2.0 starts flashing, data over the cable between FlashRunner 2.0 and DUT is decrypted.

How can we keep data encrypted also over the cable? Implementing this feature is target dependent because it's related on how the device flashing specifications are designed.

Some devices have the capability to be flashed in-application, through a small RAM firmware which actually flashes the embedded memory. This give us the opportunity to design a customized flashing firmware: FlashRunner 2.0 sends encrypted data over the cable directly to the DUT which is loaded with our customized flashing firmware on RAM. Our customized flashing firmware contains the decryption key which will be used to decrypt data and flash embedded memory.

This method is target device dependent and must be considered a customized flashing feature.



3. Traceability

Board manufacturing processes require high standards also in terms of traceability. FlashRunner 2.0 has several method able to provide traceability to the whole system:

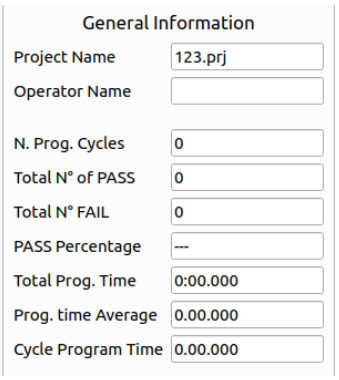
- Logging feature: provide whole history of commands executed. Each command is recorded with its own timestamp

```
01|2|000101-00:42:09|---#TPCMD CONNECT
01|2|000101-00:42:10|Requested ProtClock: 12500000.
01|2|000101-00:42:10|eMMC user data size: 59640 MB
01|2|000101-00:42:10|>|
```

- FlashRunner 2.0 protocol is based to a simple send-receive protocol. Each command sent will receive an answer back. Each project execution has a clear return state: PASS/FAIL
- Error return verbosity: each command has its own error return message, which is provided in full stack function return error

```
01|2|000430-22:30:06|---#TPCMD CONNECT
03|2|000430-22:30:06|Requested ProtClock: 20000000.
01|2|000430-22:30:06|Requested ProtClock: 20000000.
03|2|000430-22:30:11|00000040!|
03|4|000430-22:30:11|03|ERR--050000E3|(null)|[file ../emmc_hal.c, line 247, funct HAL_waitEoc()]
03|4|000430-22:30:11|03|ERR--050000D0|(null)|[file ../emmc.c, line 377, funct EMMC_readExtCsd()]
03|4|000430-22:30:11|03|ERR--050000D1|(null)|[file ../emmc.c, line 304, funct EMMC_initCard()]
03|4|000430-22:30:11|03|ERR--00000040|(null)|[file ../drv_api.c, line 650, funct CmdExe_Connect()]
03|4|000430-22:30:11|03|ERR--00000040|(null)|[file ../Src/pi-algo.c, line 388, funct cmd_TPCMD()]
01|2|000430-22:30:11|00000040!|
```

GUI Workbench tool provides easy statistics on total flashing cycles. Other implementations can be done on your own testing machine using our dll interfacing library.



4. Integrity

Every board manufacturing plant must assure to the customer that flashing process has been conducted correctly, and needs a proof of that.

For this reason we can split integrity chapter in two pieces:

- Avoid file corruption
- Assure flash matching on target device

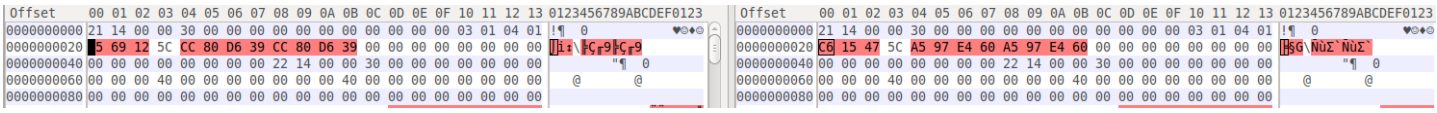
Avoid file corruption

Every customer firmware must be converted in FRB file before being used by FlashRunner 2.0. During the conversion process FlashRunner calculates a CRC number over the firmware data and save it inside frb file (it can be also easily found in frb conversion report). Once converted, frb can be downloaded into FlashRunner 2.0 storage memory and when loaded (only the first time), it's CRC is calculated at runtime by FlashRunner 2.0 and compared with the one saved inside frb file. If it doesn't match, FlashRunner 2.0 stops with an error.

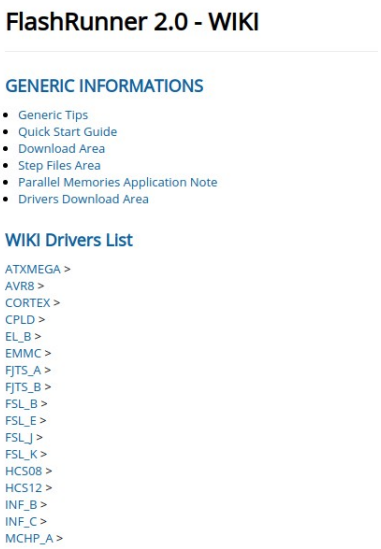
Assure flash matching on target device

Every FlashRunner driver contains as standard Verify ReadOut method which reads back content of flashed memory and compares it with the original firmware. Other methods, when it makes sense, are implemented: Verify Checksum method could be faster, as it compares digits representing the sum of the correct digits.

Another useful instrument during pre-production analysis is the DUMP method. Dump is implemented as standard command inside FlashRunner 2.0 driver. DUMP let read and save into a file the whole embedded memory content just flashed.



5. Additional features


FlashRunner 2.0 - WIKI

GENERIC INFORMATIONS

- Generic Tips
- Quick Start Guide
- Download Area
- Step Files Area
- Parallel Memories Application Note
- Drivers Download Area

WIKI Drivers List

ATXMEGA >
AVR8 >
CORTEX >
CPLD >
EL_B >
EMMC >
FJTS_A >
FJTS_B >
FSL_B >
FSL_E >
FSL_J >
FSL_K >
HCS08 >
HCS12 >
INF_B >
INF_C >
MCHP_A >

Several other features are available on our drivers, which implements specific driver security routines. Please register to our driver wiki http://www.smh-tech.com/login-form in order to have access to all our driver knowledge. Each driver has command description and full feature list.

Here's an example of RH850 driver cybersecuity features. In not already supported, additional features can be implemented on request

- **DISABLE_LOCK_BITS**

  Disable lock bits command

- **ENABLE_LOCK_BITS**

  Enable lock bits command

- **CHECK_ICU_S**

  This command returns the `ICU_S` mode Status

- **VALIDATE_ICU_S**

  This command validate the `ICU_S`. *Power on reset is necessary to be effective.*