Application Note

SPD1179 Security Application Note

Revision 1 – October 2021

Contents

List of tables

List of figures

1 Introduction

To reduce the risk of software theft, many chips have on-chip program protection. The SPD1179 was designed with the importance of program protection in mind, providing users with strong and reliable program protection. These include Debug lock and random number protection.

2 Security Features

2.1 Debug Lock

The debug interface of the chip is used frequently when debugging or burning the program. This also means that the contents of the chip's internal memory can be read or modified via the debug interface. Therefore, the chip debug feature must be disabled during mass production. Otherwise, programs inside the chip can be obtained through the debug interface.

The debug interface of SPD1179 can be locked via setting the CHIP_SECURITY field of the Configuration Words. The description of the Configuration Words is shown in Table 1. Once the debug interface of SPD1179 was locked, the internal memories can’t be accessed through the debug interface. Additionally, the reading, programming and sector erasing of the internal memories via the bootloader in ROM are also disabled; the bootloader in ROM only support chip erasing of the Flash memory.

The debug interface of SPD1179 can be unlocked through the following methods:

- Chip erase of the entire internal Flash memory via the bootloader in ROM
- Enter correct un-security keys (equal to UNSECURITY_KEY field of the Configuration Words) via the bootloader in ROM. This method is unlock the debug interface temporarily. That means the debug interface will keep locked when the chip restarts up. If the value of UNSECURITY_KEY field is 0x00000000_00000000, it means the debug interface can’t be unlocked through entering the un-security keys.

Table 1: Description of the Configuration Words

Address	Name	Description
FLASH_END - 15	UNSECURITY_KEY	Keys (8 bytes) for unlocking chip debug interface (only valid when debug interface lock is enabled) All 0x00: the debug interface can’t be unlocked via entering UNSECURITY_KEY Others: the debug interface can be unlocked temporarily via entering UNSECURITY_KEY
FLASH_END - 7	CHIP_SECURITY	Chip debug interface lock word 0xFFFFFFFF: the chip debug interface would not be locked Others: the chip debug interface would be locked
FLASH_END - 3	WDT_ENABLE	Watchdog enable word 0xFFFFFFFF: Disable watchdog when chip start-up Others: Enable watchdog when chip start-up

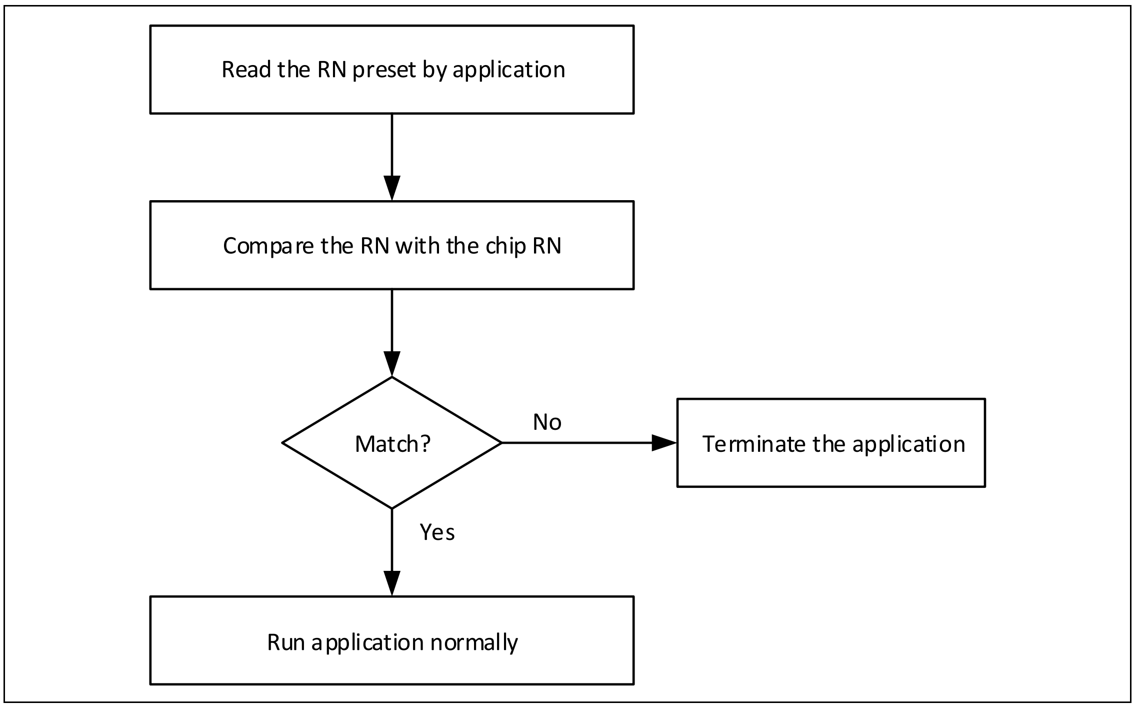
(1) The **FLASH_END** symbol represents the end address of the main flash memory.

2.2 Random Number Protect

Some chip crackers may have the ability to remove the chip package with a specific device, read out the data in Flash memory and then write it to an un-programmed chip for replication of the user's product. The SPD1179 also has the appropriate protection mechanism for this case.

SPD1179 is factory written to an 8-byte random number, which can't be modified. The customer's application can read and verify the random number from the chip at runtime. If the random number does not match the values preset by the application, the application is terminated. Because the random numbers of the chips are different from each other, the program will not work properly even if the chip cracker obtains program data from the Flash memory of a chip and then writes the program data to other SPD1179 chips. In this way, the customer's product cannot be replicated in bulk.

Figure 1: Flow Diagram of Random Number Protect in Application

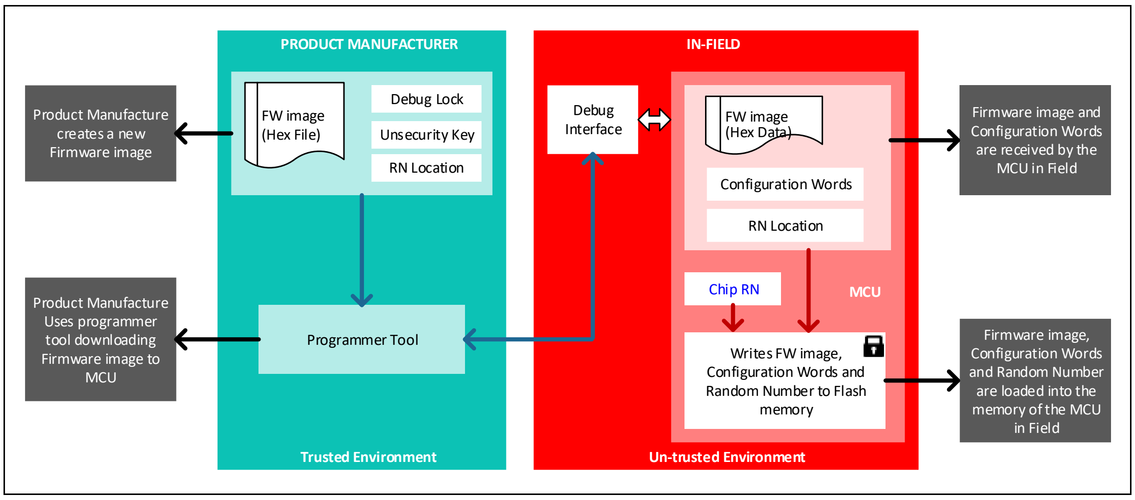

Read the RN preset by application
Compare the RN with the chip RN
Match?
No
Terminate the application
Yes
Run application normally

In Figure 1, the Random Number (RN) preset by application can be implemented through the programmer tool during product production. The programmer tool reads the random number of the target chip and writes it to the specific address of the target chip Flash memory. When the application starts up, it reads the preset Random Number from the specific address of the target chip Flash memory and compares it with the random number of the target chip.

3 Security Implementation

Figure 2 shows the steps to implement the security during in-field mass production.

Figure 2: Flow Diagram of Security Implementation during Production


PRODUCT MANUFACTURER
IN-FIELD
Product Manufacture creates a new Firmware image
FW image (Hex File)
Debug Lock
Unsecurity Key
RN Location
Debug Interface
FW image (Hex Data)
Configuration Words
RN Location
Firmware image and Configuration Words are received by the MCU in Field
Product Manufacture Uses programmer tool downloading Firmware image to MCU
Programmer Tool
Chip RN
MCU
Writes FW image, Configuration Words and Random Number to Flash memory
Firmware image, Configuration Words and Random Number are loaded into the memory of the MCU in Field
Trusted Environment
Un-trusted Environment

4 Revision history

Table 2: Document revision history

Date	Revision	Changes
21-Oct-2021	1	Initial release.