GigaDevice Semiconductor Inc.

GD32F1 & GD32F2
ARM® Cortex®-M3 32-bit MCU

Application Note
AN006

Table of Contents

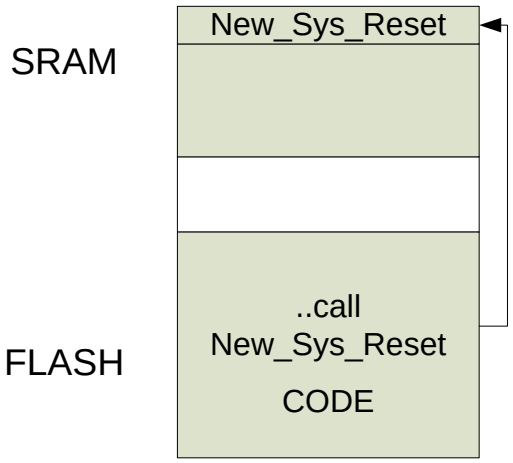
1 Introduction

The purpose of this application note is to avoid the breakdown risk when the MCU exposes to frequently reset. Generally, when a poor or unexpected condition occurs, a reset is generated according to the hardware or software design. And if the condition occurs frequently, then the reset is generated frequently. In order to achieve more reliable reset in complicated EMI environment, some methods of configuration are recommended as follows.

2 Methods of Configuration

2.1 For software reset

Software reset refers to triggering a MCU reset by code. The execution address of the software reset function is recommended to be configured in the SRAM address region.



Two steps are needed in this configuration.

First, assign a section name for the software reset function, when it is declared. And the implementation of the function contains writing the SCB->AIRCR register and a while loop.

For example, the implementation of the function is in a file called nvic_conf.c, and the code is as follows:

```
void Sys_Reset(void)  __attribute__((section ("New_Sys_Reset")));
void Sys_Reset(void)
{
    SCB->AIRCR = (NVIC_AIRCR_VECTKEY | (SCB->AIRCR & (0x700)) |
(1<<NVIC_SYSRESETREQ));         /* Keep priority group unchanged */
    while(1);
}
```

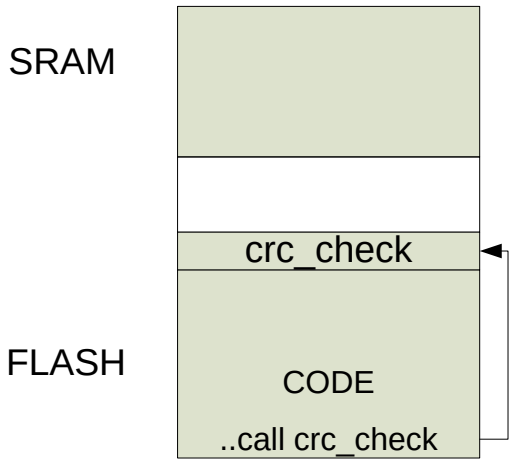
Second, change the execution address of the function. For example, modify the scatter file if

the project is developed in Keil IDE.

```
; *************************************************************
; *** Scatter-Loading Description File generated by uVision ***
; *************************************************************
LR_IROM1 0x08000000 0x00040000  {      ; load region size_region
  ER_IROM1 0x08000000 0x00040000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   *(+RO)
  }
  RW_IRAM1 0x20000000 0x0000A000  {  ; RW data
   .ANY (+RW +ZI)
  }
  ER_IRAM2 0x2000A000 UNINIT 0x00000400  {
    nvic_conf.o(New_Sys_Reset)
  }
}
```

2.2 For hardware reset

A CRC check of the FLASH content should be performed when the MCU starts up. Every time the MCU resets, a CRC calculation is carried out, and the value is compared to the correct CRC value stored when the firmware is downloaded. If a wrong CRC value is generated, the MCU should enter into the standby mode and wait for a wakeup event. When the MCU wakes up, a process of FLASH hardware correction is executed same as that when a power-on reset occurs.



Some tips should be complied with when writing the code.

1. A margin of load address should be kept between the CRC check code segment and the other code segment. For example, the load address of the CRC check code

segment can be configured at the end of the FLASH by modifying the scatter file. The implementation of the CRC check function is in a file called **crc_check.c**, and the scatter file presented as following.

```
; *************************************************************
; *** Scatter-Loading Description File generated by uVision ***
; *************************************************************
LR_IROM1 0x08000000 0x0003F000  {      ; load region size_region
  ER_IROM1 0x08000000 0x0003F000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   *(+RO)
  }
  RW_IRAM1 0x20000000 0x00010000  {  ; RW data
   .ANY (+RW +ZI)
  }
}

LR_IROM2 0x0803F000 0x1000  {
    ER_IROM2 0x0803F000 0x1000  {
      crc_check.o (+RO)
  }
}
```

2. The input context of CRC check should not include the CRC check code segment itself. That means only the other code segment is included.

3. The CRC check code segment should be called at the beginning of the Reset_Handler.

3 Revision history

Table 1. Revision history

Revision No.	Description	Date
1.0	Initial Release	November.11, 2016
1.1	Details updated	November.16, 2016