GigaDevice Semiconductor Inc.

Arm® Cortex®-M3/4/23/33 32-bit MCU

Application Note
AN039
Table of Contents

Table of Contents ................................................................. 2
List of Figures ................................................................. 3
List of Tables ................................................................. 4
1. Introduction ........................................................................ 5
2. CRC check batch processing ........................................ 6
   2.1. Add CRC value batch processing required files ............... 6
   2.2. Configure batch processing ........................................ 11
3. Results ................................................................................. 14
4. Revision history ............................................................. 15
List of Figures

Figure 2-1. Uncheck Create flash image ................................................................. 11
Figure 2-2. Post-build steps settings ........................................................................ 12
Figure 2-3. Add link file ......................................................................................... 12
Figure 2-4. Select executable file ............................................................................ 13
Figure 3-1. Build information .................................................................................. 14
Figure 3-2. Memory information .............................................................................. 14
List of Tables

Table 4-1. Revision history

15
1. Introduction

In the IEC60730 self-check, it requires self-check on the on-chip flash of the mcu. In order to realize the automatic calculation and addition of the CRC value, it is necessary to add a CRC check step in the IDE. To this end, this application note describes how to add a CRC check batch method in the eclipse environment. The process is described as follows.
2. CRC check batch processing

In the eclipse environment, the function of automatically calculating CRC is not provided. In order to complete the flash self-check required by the IEC60730 standard, the CRC value is usually pre-calculated and added to the link file manually. This method is complicated to operate. For this reason, by adding Batch processing steps to complete the automatic calculation and addition of the CRC value.

2.1. Add CRC value batch processing required files

1. First download the Srecord tool, create a new bin folder in the project directory, and copy srec_cat.exe, srec_cmp.exe and srec_info.exe to this folder.

2. Add the gen_crc.bat and gd32e10x_flash.ld files in the project directory. Here we take the gd32e10x series chips as an example. The gen_crc.bat file is used to call the Srecord tool to calculate the CRC value and store the value at the end of the .hex file to facilitate the self-check of the entire flash and print it in the build window. CRC value calculated by Srecord tool. The command code of the file is as follows:

```batch
SET MAP_FILE=gd32e10x_iec_test.map
::-------------------get CRC address information line
SET TMP_FILE=crc_temp.txt
FINDSTR /R /C:"^ *.check_sum" %MAP_FILE%>%TMP_FILE%
SET /p crc_search=<%TMP_FILE%
DEL %TMP_FILE%
::-------------------CRC address
for /i "tokens=1 delims=" %%a in ("%crc_search%") do set crc_search=%%a
SET crc_search=%crc_search:.check_sum=%
for /i "tokens=1 delims= " %%a in ("%crc_search%") do set CRC_ADDR=%%a

SET /a CRC_ADDR_END=%CRC_ADDR%+4
::-------------------choose CRC32 or CRC16
FINDSTR /R /C:"^ .*crc_block_data_calculate" %MAP_FILE% > nul &&
call :OK||call :NO
goto :eof

:OK
::-------------------CRC32
..\bin\srec_cat.exe Project.hex -intel -crop 0x08000000 %CRC_ADDR% -fill 0xff 0x08000000 %CRC_ADDR% -stm32-l-e %CRC_ADDR% -o Project_checked.hex -
```
The gd32e10x_flash.ld file defines the loading address of each program segment and variable. Through the code shown below, the CRC value (CHECKSUM) is fixed at the end of the FLASH space.

```
ENTRY(Reset_Handler)

/* end of Stack */
estack = 0x20008000; /*ram size*/

/* memory map */
MEMORY
{
  FLASH (rx) : ORIGIN = 0x08000000, LENGTH = 128K /*flash size*/
  iec_test (wxa!ri) : ORIGIN = 0x20000000, LENGTH = 0xB0
  RAM (xrw) : ORIGIN = 0x200000B0, LENGTH = 0x7F50 /*32K*/
  flash_end (rxai!w) : ORIGIN = 0x0801FFC0, LENGTH = 0x40 /*flash size - 0x40*/
}
SECTIONS
```

The gd32e10x_flash.ld file defines the loading address of each program segment and variable. Through the code shown below, the CRC value (CHECKSUM) is fixed at the end of the FLASH space.

```
ENTRY(Reset_Handler)

/* end of Stack */
estack = 0x20008000; /*ram size*/

/* memory map */
MEMORY
{
  FLASH (rx) : ORIGIN = 0x08000000, LENGTH = 128K /*flash size*/
  iec_test (wxa!ri) : ORIGIN = 0x20000000, LENGTH = 0xB0
  RAM (xrw) : ORIGIN = 0x200000B0, LENGTH = 0x7F50 /*32K*/
  flash_end (rxai!w) : ORIGIN = 0x0801FFC0, LENGTH = 0x40 /*flash size - 0x40*/
}
SECTIONS
```
__stack_size = DEFINED(__stack_size) ? __stack_size : 2K;

.isr_vector:
{
   . = ALIGN(4);
   KEEP(*(.isr_vector))
   . = ALIGN(4);
} >FLASH

.text:
{
   . = ALIGN(4);
   *(.text)
   *(.text*)
   *(.glue_7)
   *(.glue_7t)
   *(.eh_frame)

   KEEP *(.init))
   KEEP *(.fini))

   . = ALIGN(4);
   /* the symbol '_etext' will be defined at the end of code section */
   _etext = .;
} >FLASH

.rodata:
{
   . = ALIGN(4);
   *(.rodata)
   *(.rodata*)
   . = ALIGN(4);
} >FLASH

.ARM.extab:
{
   *(.ARM.extab*.gnu.linkonce.armextab.*)
} >FLASH

.ARM:
{
   __exidx_start = .;
   *(.ARM.exidx*)
CRC check batch addition method for IEC60730 Flash self-check in eclipse environment

```c
__exidx_end = .;
}
FLASH

.ARM.attributes : { *(.ARM.attributes) } > FLASH

.preinit_array :
{
  PROVIDE_HIDDEN (__preinit_array_start = .);
  KEEP (*(.preinit_array*))
  PROVIDE_HIDDEN (__preinit_array_end = .);
} >FLASH

.init_array :
{
  PROVIDE_HIDDEN (__init_array_start = .);
  KEEP (*(SORT(.init_array.*)))
  KEEP (*(.init_array*))
  PROVIDE_HIDDEN (__init_array_end = .);
} >FLASH

.fini_array :
{
  PROVIDE_HIDDEN (__fini_array_start = .);
  KEEP (*(SORT(.fini_array.*)))
  KEEP (*(.fini_array*))
  PROVIDE_HIDDEN (__fini_array_end = .);
} >FLASH

._iec_classb(NOLOAD) :
{
  . = ALIGN(4);
  *(.ram_run_buf)
  *(.ram_run_buf*)
  *(.ram_run_ptr)
  *(.ram_run_ptr*)
  . = ALIGN(4);
  PROVIDE(_iec_start = .);

  *(.iec_test_ram)
  *(.iec_test_ram*)

  . = ALIGN(4);
```
CRC check batch addition method for IEC60730 Flash self-check in eclipse environment

```assembly
PROVIDE(_iec_end = .);
}>iec_test AT>iec_test

/* provide some necessary symbols for startup file to initialize data */
_sidata = LOADADDR(.data);
.data :
{
  . = ALIGN(4);
  /* the symbol '._sdata' will be defined at the data section end start */
  _sdata = .;
  *(.data)
  *(.data*)
  . = ALIGN(4);
  /* the symbol '._edata' will be defined at the data section end */
  _edata = .;
}>RAM AT> FLASH

.check_sum :
{
  . = ALIGN(64);
  PROVIDE( __Check_Sum = .);

  LONG(0x904eae51);
}>flash_end AT>flash_end

  . = ALIGN(4);
  .bss :
  {
    /* the symbol '_sbss' will be defined at the bss section start */
    _sbss = .;
    __bss_start__ = _sbss;
    *(.bss)
    *(.bss*)
    *(COMMON)
    . = ALIGN(4);
    /* the symbol '_ebss' will be defined at the bss section end */
    _ebss = .;
    __bss_end__ = _ebss;
  }>RAM

  PROVIDE ( end = _ebss );
  PROVIDE ( _end = _ebss );
```
2.2. Configure batch processing

1. After adding the above two files to the project directory, open the project, right-click the project name, and uncheck Create flash image in Properties->C/C++ Build->Setting->Toolchains. This is to use Batch process to generate executable files, as shown in Figure 2-1. Uncheck Create flash image.

Figure 2-1. Uncheck Create flash image

2. Right-click the project name->Properties->C/C++build->settings->build steps, and add the following command in Post-build steps:
CRC check batch addition method for IEC60730 Flash self-check in eclipse environment

\$\{cross\_prefix\}$cross\_objcopy$\{cross\_suffix\} -O ihex "gd32f30x_iec_test.elf" "Project.hex":.:\gen\_crc\_bat:. click apply, this command is to ensure that the operation of converting the .elf file into a .hex file is before the operation of the .bat file command, the setting method is as follows. 

Figure 2-2. Post-build steps settings

3. Right-click the project name->Properties->C/C++build->settings->GNU ARM Cross C Linker->General, add the link file here, as shown in Figure 2-3. Add link file, after modification Click Apply and Close.

Figure 2-3. Add link file
4. When debugging, you need to select the executable file as the Project.hex file generated after batch processing, in the Load executable option of the Run->Debug Configuration->GDB SEGGER J-Link Debugging->gd32e10x_iec_test Debug->Startup interface Choose Use file. As shown in Figure 2-4. Select executable file, click Apply. After the modification, the Project.hex file generated by batch processing is stored in the Debug folder in the workspace.

**Figure 2-4. Select executable file**

![Debug Configurations](image-url)
3. **Results**

After the configuration is completed, click Compile, and you can observe the build information in the Build Output window, as shown in *Figure 3-1. Build information*, showing that the CRC value has been stored in the location after 0x0801FFF0; click Debug, and query the address of 0x0801FFF0 in the memory observation window, as shown in *Figure 3-2. Memory information*, it can be seen that the value in the memory is consistent with the CRC value displayed in the Build Output window, and the CRC value calculation batch has been added successfully.

*Figure 3-1. Build information*

![Build information](image1)

*Figure 3-2. Memory information*

<table>
<thead>
<tr>
<th>Address</th>
<th>CRC Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001F7F0</td>
<td>FFFFFFFF</td>
</tr>
<tr>
<td>0001F7F4</td>
<td>FFFFFFFF</td>
</tr>
<tr>
<td>0001F7FE</td>
<td>FFFFFFFF</td>
</tr>
<tr>
<td>00020000</td>
<td>FFFFFFFF</td>
</tr>
<tr>
<td>00020004</td>
<td>FFFFFFFF</td>
</tr>
<tr>
<td>00020008</td>
<td>FFFFFFFF</td>
</tr>
<tr>
<td>0002000C</td>
<td>FFFFFFFF</td>
</tr>
</tbody>
</table>

![Memory information](image2)
4. Revision history

Table 4-1. Revision history

<table>
<thead>
<tr>
<th>Revision No.</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Initial Release</td>
<td>Oct.21, 2021</td>
</tr>
</tbody>
</table>
Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the "Product"), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as its suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as its suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.

© 2021 GigaDevice – All rights reserved