**GigaDevice Semiconductor Inc.** 

Arm<sup>®</sup> Cortex<sup>®</sup>-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  |    |
| Figure 2-3. Add link file              |    |
| Figure 2-4. Select executable file     |    |
| Figure 3-1. Build information          |    |
| Figure 3-2. Memory information         |    |



# **List of Tables**

| e 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:

SET MAP FILE=qd32e10x 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 /f "tokens=1 delims=(" %%a in ("%crc search%") do set crc search=%%a SET crc\_search=%crc\_search:.check\_sum=% for /f "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 Oxff 0x08000000 %CRC\_ADDR% -stm32-I-e %CRC\_ADDR% -o Project\_checked.hex -



| intel   |
|---|
| \bin\srec_cat.exe Project.hex -intel -crop 0x08000000 %CRC_ADDR%            |
| Project_checked.hex -intel -crop %CRC_ADDR% %CRC_ADDR_END% -o               |
| Project.hex -intel  |
| \bin\srec_cat.exe Project.hex -intel -crop %CRC_ADDR% %CRC_ADDR_END%        |
| -o -hex-dump  |
| del Project_checked.hex   |
| goto :eof   |
|   |
| :NO   |
| ::CRC16   |
| \bin\srec_cat.exe Project.hex -intel -crop 0x08000000 %CRC_ADDR% -fill 0xff |
| 0x08000000 %CRC_ADDR% -crc16-I-e %CRC_ADDR% -POLYnomial ccitt -XMODEM       |
| -o Project_checked.hex -intel   |
| \bin\srec_cat.exe Project.hex -intel -crop 0x08000000 %CRC_ADDR%            |
| Project_checked.hex -intel -crop %CRC_ADDR% %CRC_ADDR_END% -o               |
| Project.hex -intel  |
| \bin\srec_cat.exe Project.hex -intel -crop %CRC_ADDR% %CRC_ADDR_END%        |
| -o -hex-dump  |
| del Project_checked.hex   |
| goto :eof   |
|   |
| exit  |

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*)
```



```
__exidx_end = .;
  } >FLASH
.ARM.attributes : { *(.ARM.attributes) } > FLASH
.preinit_array
                  2
{
  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 (*(.fini_array*))
  KEEP (*(SORT(.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);
```



```
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
                11
{
  . = 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 );
```



## AN039 CRC check batch addition method for IEC60730 Flash self-check in eclipse environment

```
.stack ORIGIN(RAM) + LENGTH(RAM) - __stack_size - 0x18:
{
    *(.stack_ov_test)
    *(.stack_ov_test*)
    . = 0x18;
    PROVIDE( _heap_end = . );
    . = __stack_size + 0x18;
    PROVIDE( _sp = . );
}
/* input sections */
GROUP(libgcc.a libc.a libm.a libnosys.a)
```

# 2.2. Configure batch processing

 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 <u>Figure 2-1. Uncheck Create flash image</u>.

| Figure | 2-1. | Uncheck  | Create | flash | image |
|--------|------|----------|--------|-------|-------|
| riguic | 2-1. | Olicheck | Orcale | nasn  | mage  |

| Environment                                      |                    |  |
|--|--------------------|--|
| Logging  | 🛞 Tool Settings    | ) Toolchains 📕 Devices 🎤 Build Steps 🚇 Build Artifact 🗟 Binary Pa 🔹 🕨                                |
| Settings   |                    |  |
| Tool Chain Editor                                | Name:              | GNU Tools for ARM Embedded Processors (arm-none-eabi-gcc)  |
| > C/C++ General                                  | Architecture:      | ARM (AArch32) v  |
| Linux Tools Path                                 | Prefix:            | arm-none-eabi-   |
| > MCU  | Prefix:            | arm-hone-eabl-   |
| Project References                               | Suffix:            |  |
| Refactoring History<br>Run/Debug Settings        | C compiler:        | gcc  |
| > Task Repository                                | C++ compiler:      | g++  |
| Task Tags  | Archiver:          | ar   |
| <ul> <li>Validation</li> <li>WikiText</li> </ul> | Hex/Bin converter: | objcopy  |
|  | Listing generator: | objdump  |
|  | Size command:      | size   |
|  | Build command:     | make   |
|  | Remove command:    | rm   |
|  |                    |  |
|  | Toolchain path:    | D:\Software\gnumcueclipse-4.3.1-oxygen-2\eclipse\eclipse_toolchain\GNU Tools                         |
|  |                    | (to change it use the <u>global</u> or <u>workspace</u> preferences pages or the <u>project</u> prop |
|  | Build tools path:  | D:\Software\gnumcueclipse-4.3.1-oxygen-2\eclipse\eclipse_toolchain\Build Tools                       |
|  |                    | (to change it use the <u>global</u> or <u>workspace</u> preferences pages or the <u>project</u> prop |
|  |                    |  |
|  | Create flash imag  | je   |
|  | Create extended    | listing  |
|  | ✓ Print size       |  |
|  |                    |  |

2. Right-click the project name->Properties->C/C++build->settings->build steps, and add the following command in Post-build steps:

11



### AN039 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 shown.

#### Figure 2-2. Post-build steps settings

| Properties for gd32e10x_ie   | ec_test – 🗆 X   |
|--|---|
|  | Settings 🗘 🕆 🗸 🔻  |
| <ul> <li>&gt; Resource<br/>Builders</li> <li>&gt; C/C++ Build<br/>Build Variables<br/>Environment<br/>Logging<br/>Settings<br/>Tool Chain Editor</li> <li>&gt; C/C++ General<br/>Linux Tools Path</li> <li>&gt; MCU<br/>Project References<br/>Refactoring History<br/>Run/Debug Settings</li> <li>&gt; Task Repository<br/>Task Tags</li> <li>&gt; Validation<br/>WikiText</li> </ul> | Configuration: Debug [Active] 			Manage Configurations  Tool Settings Toolchains Devices Puild Steps Puild Artifact Binary Parsers E E   Pre-build steps Command: Post-build steps Command: [\$(cross_objcopy)\$(cross_suffix) -O ihex *gd32e10x_iec_test.elf* *Project.hex*,\\gen_crc.bat; v  Description: |
| ?  | Apply and Close Cancel  |

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

#### Figure 2-3. Add link file

| Properties for gd32e10x_i   | -   | - 🗆 X  |
|---|---|--|
| <ul> <li>Properties for gd32e10x j</li> <li>type filter text</li> <li>Resource<br/>Builders</li> <li>C/C++ Build<br/>Build Variables</li> <li>Environment<br/>Logging</li> <li>Settings<br/>Tool Chain Editor</li> <li>C/C++ General<br/>Linux Tools Path</li> <li>MCU<br/>Project References<br/>Refactoring History<br/>Run/Debug Settings</li> <li>Task Repository<br/>Task Tags</li> <li>Validation<br/>WikiText</li> </ul> | Settings         Configuration:       Debug [ Active ]         Tool Settings       Toolchains       Devices         Target Processor       Script files (-T)         Optimization       Script files (-T)         Warnings       Debugging         S GNU ARM Cross Assembler       Preprocessor         Warnings       Marnings         Warnings       Miscellaneous         S Muscellaneous       S Warnings         Marnings       Includes         Includes       Includes         Includes       Includes         Includes       Includes | ↓ ↓ ↓ ✓ Manage Configurations rtifact in Binary Parsers is Error Parsers Image State of the second s |
|   | Optimization     Overal     Second Constraints     Second Constraints     Second Constraints     Second Constraints     Second Constraints     Second Constraints   |  |



## AN039 CRC check batch addition method for IEC60730 Flash self-check in eclipse environment

4. When debugging, you need to select the executable file as the Project.hex file generated after batch processing, in the Load excutable 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   |   | ×  |
|--|---|----|
| Create, manage, and run configuratio   | ons   | ñ  |
|  | ال  | ~  |
| Image: Second secon | Name:     gd32e10x_iec_test Debug       Image: The startup     Image: Superior Startup       Image: The startup     Image: Superice Startup       Image: The startup | ^  |
| <ul> <li>C/C++ Attach to Application</li> <li>C/C++ Postmortem Debugger</li> <li>C/C++ Remote Application</li> <li>GDB Hardware Debugging</li> <li>GDB OpenOCD Debugging</li> </ul>  | <ul> <li>☑ Enable flash breakpoints</li> <li>☑ Enable semihosting Console routed to: ☑ Telnet □ GDB client</li> <li>☑ Enable SWO CPU freq: 0</li> <li>Hz. SWO freq: 0</li> <li>Hz. Port mask: 0x1</li> </ul>  |    |
| C GDB OpenOCD Debugging<br>C GDB OpenOCD Debugging<br>C GDB PyOCD Debugging<br>C GDB QEMU Debugging<br>C GDB SEGGER J-Link Debugging   | Load Symbols and Executable   |    |
| C gd32e10x_iec_test Debug<br>G Launch Group<br>► Launch Group (Deprecated)   | Load symbols       ☑ Load symbols       ④ Use file:         Workspace   File System   |    |
|  | Symbols offset (hex):   |    |
|  | Ouse project binary:         gd32e10x_iec_test.elf           Ouse file:         \${workspace_loc:\gd32e10x_iec_test\Debug\Project.hex}           Workspace_noc:\gd32e10x_iec_test\Debug\Project.hex}         Workspace  |    |
|  | Executable offset (hex):          Runtime Options         RAM application (reload after each reset/restart)   |    |
|  | Run/Restart Commands  | ~  |
| < > Filter matched 13 of 17 items  | Revert Appl   | /  |
| ?  | Debug   | se |



## 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

E:\SVN\IEC60730\IEC\_Test\_GD32E103\V2.0\GD32E103V\_EVAL\_Demo\_Suites\Projects\IEC\_TEST\eclipse\Debug>..\bin\sr E:\SVN\IEC60730\IEC\_Test\_GD32E103\V2.0\GD32E103V\_EVAL\_Demo\_Suites\Projects\IEC\_TEST\eclipse\Debug>..\bin\sr 0801FFC0: B5 22 93 65 #5".e E:\SVN\IEC60730\IEC\_Test\_GD32E103\V2.0\GD32E103V\_EVAL\_Demo\_Suites\Projects\IEC\_TEST\eclipse\Debug>del Proje E:\SVN\IEC60730\IEC\_Test\_GD32E103\V2.0\GD32E103V\_EVAL\_Demo\_Suites\Projects\IEC\_TEST\eclipse\Debug>del Proje

#### Figure 3-2. Memory information

| 0x0801ffc0 : | 0x0801ffc0 : 0x801FFC0 <hex> 🖄 🖕 New Renderings</hex> |          |          |          |  |
|--------------|---|----------|----------|----------|--|
| Address      | 0 - 3   | 4 - 7    | 8 - B    | C - F    |  |
| 0801FF90     | FFFFFFF   | FFFFFFFF | FFFFFFFF | FFFFFFF  |  |
| 0801FFA0     | FFFFFFF   | FFFFFFF  | FFFFFFF  | FFFFFFF  |  |
| 0801FFB0     | FFFFFFFF  | FFFFFFFF | FFFFFFF  | FFFFFFF  |  |
| 0801FFC0     | B5229365  | FFFFFFF  | FFFFFFFF | FFFFFFF  |  |
| 0801FFD0     | FFFFFFF   | FFFFFFFF | FFFFFFF  | FFFFFFF  |  |
| 0801FFE0     | FFFFFFF   | FFFFFFF  | FFFFFFF  | FFFFFFF  |  |
| 0801FFF0     | FFFFFFF   | FFFFFFF  | FFFFFFFF | FFFFFFFF |  |
| 08020000     | FFFFFFF   | FFFFFFF  | FFFFFFFF | FFFFFFFF |  |
| 08020010     | FFFFFFF   | FFFFFFF  | FFFFFFFF | FFFFFFFF |  |
| 08020020     | FFFFFFF   | FFFFFFF  | FFFFFFFF | FFFFFFF  |  |



# 4. Revision history

#### Table 4-1. Revision history

| Revision No. | Dscription      | Date         |
|--------------|-----------------|--------------|
| 1.0          | Initial Release | Oct.21, 2021 |



#### **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 it's 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 it's 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