GigaDevice Semiconductor Inc.

基于 IAR 环境下 RISC-V 内核 IEC60730 认证库移植

应用笔记

AN071

目录

图索引

表索引

1. 简介

目前基于 RISC-V 内核的 GD32 MCU 支持两种开发环境 Eclipse 和 IAR，其中基于 Eclipse 环境的 IEC60730 认证库已开发完成，为了提高 IEC60730 认证库的多样性，需要在 IAR 环境中移植 IEC60730 认证库。本文将介绍在 IAR 环境中移植 IEC60730 认证库所遇到的问题和对应的解决方法。

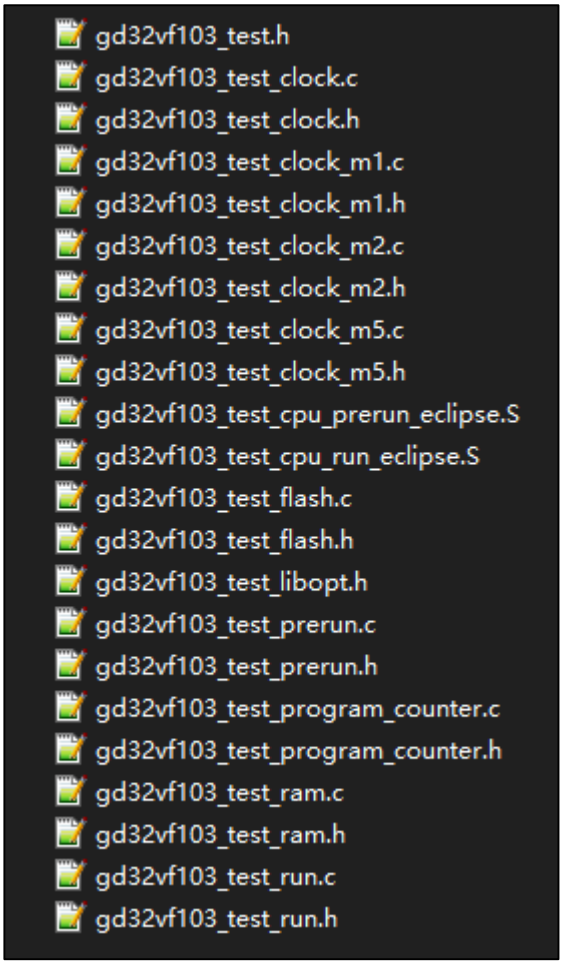
2. 基于 IAR 环境和 RISC-V 内核的 IEC60730 认证库移植

2.1. IEC60730 认证库移植平台

本文基于 GD32VF103V-EVAL V1.0 开发板移植 IEC60730 认证库，通过该认证程序实现对 MCU 不同模块（CLOCK、CPU、FLASH、RAM、Watchdog）的功能检测。

IAR 环境移植 IEC60730 认证库基于 Eclipse 环境下的 IEC60730 认证库工程，主要包括以下文件如*图 2-1. IEC60730 认证库源代码*所示。

图 2-1. IEC60730 认证库源代码

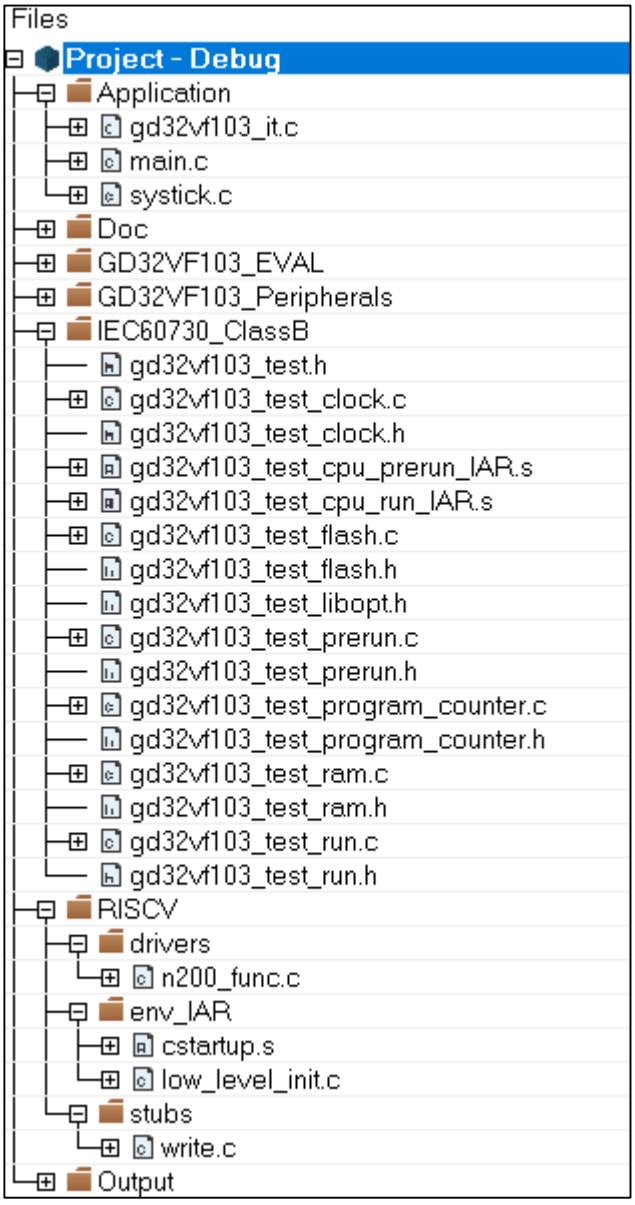


2.2. 新建 IAR 工程

在包含 eclipse 环境的 IEC60730 工程的文件夹下新增 EWRISCV 文件夹，然后在 IAR（IAR EW for RISC-V 1.40.1）中新建工程，添加工程所需文件，文件添加完成后，工程所需添加的文件目录如*图 2-2. IAR 工程目录*所示。其中，文件 gd32vf103_test_cpu_prerun_IAR.s 和 gd32f30x_test_cpu_run_IAR.s 是直接通过修改*图 2-1. IEC60730 认证库源代码*中的文件

gd32vf103_test_cpu_prerun_eclipse.S 和 gd32vf103_test_cpu_run_eclipse.S 的命名得来。

图 2-2. IAR 工程目录



2.3. 修改 cstartup.s 文件

IEC60730 中功能检测分为两个阶段，系统启动自检和运行自检，因此需要修改 cstartup.s 代码，在程序运行到 main()函数之前先调用 test_prerun()函数，从而完成系统启动自检，添加之后代码如*表 2-1. 添加 test_prerun()函数*所示。其中应注意__iar_data_init2()函数应在调用 test_prerun()函数之后再次调用，若不调用在 test_prerun()函数执行过程中会破坏__iar_data_init2()函数对数据的初始化，导致 main()函数中 printf()函数无法输出打印信息。

表 2-1. 添加 test_prerun()函数

```
        EXTERN  test_prerun
        CfiCall test_prerun
```

```
call      test_prerun

beq       a0, zero, ?cstart_call_main

// Reinitialize the data segment
EXTERN    __iar_data_init2
CfiCall __iar_data_init2
call      __iar_data_init2
```

2.4. 修改 gd32vf103_test_cpu_prerun_IAR.s 和 gd32vf103_test_cpu_run_IAR.s 文件

完成 2.3 节修改后对工程进行编译，gd32vf103_test_cpu_prerun_IAR.s 和 gd32vf103_test_cpu_run_IAR.s 文件会报错，提示语法错误，出错指令为结束返回指令：ret，为此首先将“ret”命令修改为“end”指令，编译通过。在后续调试中发现，仅添加“ret”指令或者“end”指令，程序均无法正常运行，需将两条指令配合使用，即先“ret”后“end”。

对比 ARM 内核下 CPU 检测 IAR 环境的汇编文件，还需要在 gd32vf103_test_cpu_prerun_IAR.s 和 gd32vf103_test_cpu_run_IAR.s 文件中增加保存通用寄存器的相关代码，否则会导致后续代码运行失败。代码修改之后如 *表 2-3. gd32vf103_test_cpu_run_IAR.s 代码*所示，红色标注的代码为修改和添加的代码。

表 2-2. gd32vf103_test_cpu_prerun_IAR.s 代码

```
    SECTION factor_def:CODE:NOROOT(2)

    PUBLIC test_cpu_prerun
    IMPORT test_fail_reset

GENERAL_FACTOR1   EQU   0xAAAAAAAA
GENERAL_FACTOR2   EQU   ~GENERAL_FACTOR1

SP_FACTOR1        EQU   0xAAAAAAA8
SP_FACTOR2        EQU   0x55555554

LOG_REGBYTES      EQU 3
REGBYTES          EQU (1 << LOG_REGBYTES)
;/*
;   \brief   cpu test in prerun time
;   \param   none
;   \retval TypeState: ERROR or SUCCESS
;*/
test_cpu_prerun:
```

```
    addi sp, sp, -20*REGBYTES
    sw x1,   0*REGBYTES(sp)
    sw x4,   1*REGBYTES(sp)
    sw x5,   2*REGBYTES(sp)
    sw x6,   3*REGBYTES(sp)
    sw x7,   4*REGBYTES(sp)
    sw x10, 5*REGBYTES(sp)
    sw x11, 6*REGBYTES(sp)
    sw x12, 7*REGBYTES(sp)
    sw x13, 8*REGBYTES(sp)
    sw x14, 9*REGBYTES(sp)
    sw x15, 10*REGBYTES(sp)
    ……
    ……
    ……
    lw x1,   0*REGBYTES(sp)
    lw x6,   2*REGBYTES(sp)
    lw x7,   3*REGBYTES(sp)
    lw x10, 4*REGBYTES(sp)
    lw x11, 5*REGBYTES(sp)
    lw x12, 6*REGBYTES(sp)
    lw x13, 7*REGBYTES(sp)
    lw x14, 8*REGBYTES(sp)
    lw x15, 9*REGBYTES(sp)
    addi sp, sp, 20*REGBYTES

    li      t0,1
    mv      a0,t0                       // SUCCESS = 1

    ret
    end
```

表 2-3. gd32vf103_test_cpu_run_IAR.s 代码

```
    SECTION factor_def:CODE:NOROOT(2)

    PUBLIC test_cpu_run
    IMPORT test_fail_reset

GENERAL_FACTOR1   EQU   0xAAAAAAAA
GENERAL_FACTOR2   EQU   ~GENERAL_FACTOR1

LOG_REGBYTES       EQU 3
REGBYTES           EQU (1 << LOG_REGBYTES)
```

```
;/*
;   \brief   cpu test in prerun time
;   \param   none
;   \retval TypeState: ERROR or SUCCESS
;*/
test_cpu_prerun:

    addi sp, sp, -20*REGBYTES
    sw x1,   0*REGBYTES(sp)
    sw x4,   1*REGBYTES(sp)
    sw x5,   2*REGBYTES(sp)
    sw x6,   3*REGBYTES(sp)
    sw x7,   4*REGBYTES(sp)
    sw x10, 5*REGBYTES(sp)
    sw x11, 6*REGBYTES(sp)
    sw x12, 7*REGBYTES(sp)
    sw x13, 8*REGBYTES(sp)
    sw x14, 9*REGBYTES(sp)
    sw x15, 10*REGBYTES(sp)
    ……
    ……
    ……
    lw x1,   0*REGBYTES(sp)
    lw x6,   2*REGBYTES(sp)
    lw x7,   3*REGBYTES(sp)
    lw x10, 4*REGBYTES(sp)
    lw x11, 5*REGBYTES(sp)
    lw x12, 6*REGBYTES(sp)
    lw x13, 7*REGBYTES(sp)
    lw x14, 8*REGBYTES(sp)
    lw x15, 9*REGBYTES(sp)
    addi sp, sp, 20*REGBYTES

    li      t0,1
    mv      a0,t0                   // SUCCESS = 1

    ret
    end
```

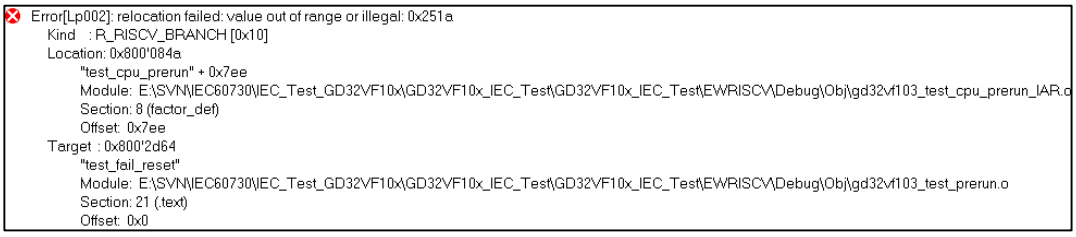
2.5. 修改分散加载文件

在对工程进行编译前需要对比其他 ARM 内核系列的分赛加载文件，修改本工程所需的分散加

载文件，标准的分散加载文件可以在 IAR 的安装目录…\riscv\config\linker\GigaDevice 下找到。

完成上述步骤后进行编译，会出现以下错误在如*图 2-3. 编译窗口提示*所示。报错原因为 RISC-V 汇编指令中的条件跳转指令，B 类指令的跳转时，PC 的寻址范围是(+ / −)4KB，因此，解决方法是利用分散加载将 test_cpu_prerun()函数、test_cpu_run()函数和 test_fail_rest()函数定位到 FLASH 空间 4KB 以内。

图 2-3. 编译窗口提示



为了实现上述分散加载，首先将包含 test_cpu_prerun()函数、test_cpu_run()函数和 test_fail_rest()函数的 gd32vf103_test_cpu_prerun_IAR.o、gd32vf103_test_cpu_run_IAR.o 和 gd32vf103_test_prerun.o 文件分散加载到 4KB 空间内，但是由于包含 test_fail_rest()函数的.o 文件 gd32vf103_test_prerun.o 所需的空间较大，导致三个.o 文件超过了 4KB 空间。查询 IAR 的 Help 手册，可对.c 文件中的函数单独分配存储位置，如*图 2-4. test_fail_rest()函数分散加载*所示。

图 2-4. test_fail_rest()函数分散加载

```
*/
void test_fail_reset(void) @ ".TEST_FAIL_RESET"
{
```

经过上述处理之后，对 gd32vf103_test_cpu_prerun_IAR.o、gd32vf103_test_cpu_run_IAR.o 和. TEST_FAIL_RESET 进行分散加载，其所需空间不超过 4KB，完成对比修改和函数空间位置修改之后，分散加载代码如*表 2-4. 分散加载代码*所示。

表 2-4. 分散加载代码

```
//////////////////////////////////////////////////////////////////////////////
// RISC-V ilink configuration file.

define exported symbol _link_file_version_2 = 1;
define exported symbol _auto_vector_setup = 1;
define exported symbol _max_vector = 96;
define exported symbol _CLINT = 1;

define memory mem with size = 4G;

/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__   = 0x0801FFFF;
```

```
define symbol __ICFEDIT_region_ROM1_start__ = 0x08004000;
define symbol __ICFEDIT_region_ROM1_end__   = 0x08004fff;


define symbol __ICFEDIT_region_RAM_start__ = 0x200000B0;
define symbol __ICFEDIT_region_RAM_end__   = 0x20007FFF;
define symbol __ICFEDIT_region_IECTEST_PARAM_start__   = 0x20000040;
define symbol __ICFEDIT_region_IECTEST_PARAM_end__      = 0x200000B0;

/*Sizes*/
define symbol __ICFEDIT_size_stack_ov_test__      = 0x18;
/**** End of ICF editor section. ###ICF###*/

export symbol   __ICFEDIT_region_ROM_start__;
export symbol   __ICFEDIT_region_ROM_end__;
export symbol   __ICFEDIT_region_RAM_start__;
export symbol   __ICFEDIT_region_RAM_end__;
export symbol   __ICFEDIT_region_IECTEST_PARAM_start__;
export symbol   __ICFEDIT_region_IECTEST_PARAM_end__;



define region ROM_region32 = mem:[from __ICFEDIT_region_ROM_start__ to
__ICFEDIT_region_ROM_end__];
define region RAM_region32 = mem:[from __ICFEDIT_region_IECTEST_PARAM_end__ to
__ICFEDIT_region_RAM_end__];
define region ROM1_region32 = mem:[from __ICFEDIT_region_ROM1_start__ to
__ICFEDIT_region_ROM1_end__];

/*-Symbols-*/
define symbol __region_RAM_RUN_BUF_start__   = 0x20000004;
define symbol __region_RAM_RUN_PTR__satrt__ = 0x20000030;
define symbol __region_IEC_TEST_RAM_start__   = 0x20000040;

/*-Memory Regions-*/
define region RAM_BUF_region = mem:[from __region_RAM_RUN_BUF_start__ to 0x2000002F];
define region RAM_PTR_region = mem:[from __region_RAM_RUN_PTR__satrt__ to 0x2000003F];
define region IEC_TEST_VAR_region    = mem:[from __region_IEC_TEST_RAM_start__ to
0x200000AF];
/**** End of ICF editor Regions. ###ICF###*/

initialize by copy { rw };
```

```
do not initialize   { section *.noinit,
                      section   STACK_OV_TEST,
                      section RAM_RUN_BUF,
                      section RAM_RUN_PTR,
                      section IEC_TEST_RAM};
place in ROM1_region32          { object gd32vf103_test_cpu_prerun_IAR.o,
                                  section .TEST_FAIL_RESET,
                                  object          gd32vf103_test_cpu_run_IAR.o};          //Make
test_cpu_prerun(), test_cpu_run() and test_fail_reset() no more than 0x1000 in the flash by scatter
loading

define block CSTACK with alignment = 16, size = CSTACK_SIZE { };
define block HEAP     with alignment = 16, size = HEAP_SIZE     { };
define block STACK_OV_TEST   with alignment = 8, size = __ICFEDIT_size_stack_ov_test__
{ };

define block MVECTOR with alignment = 128, size = _max_vector*4 { ro section .mintvec };

place in IEC_TEST_VAR_region
                      { rw data section IEC_TEST_RAM   };

place in RAM_BUF_region
                      { rw data section RAM_RUN_BUF   };

place in RAM_PTR_region
                      { rw data section RAM_RUN_PTR   };

if (isdefinedsymbol(_uses_clic))
{
   define block MINTERRUPT with alignment = 128 { ro section .mtext };
   define block MINTERRUPTS { block MVECTOR,
                              block MINTERRUPT };
}
else
{
   define block MINTERRUPTS with maximum size =   64k { ro section .mtext,
                                                        midway block MVECTOR };
}

define block RW_DATA with static base GPREL { rw data };
keep { symbol __iar_cstart_init_gp }; // defined in cstartup.s
keep { ro section .alias.hwreset };
```

```
"CSTARTUP32" : place at start of ROM_region32 { ro section .alias.hwreset,
                                                ro section .cstartup };


"ROM32":place in ROM_region32          { ro,
                                         block MINTERRUPTS }
                                 except  {object gd32vf103_test_cpu_prerun_IAR.o,
                                          section .TEST_FAIL_RESET,
                                          object gd32vf103_test_cpu_run_IAR.o };
          place at end of ROM_region32  { ro section .checksum };

"RAM32":place in RAM_region32          { block RW_DATA,
                                         block HEAP,
                                         block CSTACK,
                                         block STACK_OV_TEST };
```

2.6. 修改 RAM 检测代码

由于 RAM 检测会破坏堆栈内容，所以在对 RAM 进行操作前，需对堆栈进行保存，检测完成之后对堆栈进行恢复，对比 ARM 内核检测代码，首先获取 SP，然后保存堆栈内容。通过编译反映了两种架构下的 SP 读函数不兼容，需要重新实现 SP 读取函数。查阅 RISC-V 内核指令，通过 mscratch 寄存器获取 SP，首先通过内联函数将 SP 写入 mscratch 寄存器，然后读取 mscratch 寄存器的值，从而实现 SP 的读取，代码实现如*表 2-5. SP 读取函数*所示。

表 2-5. SP 读取函数

```
void write_sp(void){     asm("csrrw sp, mscratch,sp");}
void read_sp(void){      asm("csrrw sp, mscratch,sp");}
write_sp();
ptr_stack = (uint32_t *)read_csr(CSR_MSCRATCH)+8;
read_sp();
```

在堆栈内容保存完成之后，编译运行程序，程序执行跳转正常，但 printf()函数不能正常打印，单步执行调试，将 RAM 检测起始地址修改为 0x20000140 时，可以正常打印，说明在 RAM 起始地址存储了重要信息，为此在 RAM 检测之前，不仅需要保存堆栈，还是需要保存起始地址的数据，为此在 RAM 检测代码中加入下列代码，如*表 2-6. 起始地址数据保护代码*所示。

表 2-6. 起始地址数据保护代码

```
        ptr_stack = (uint32_t *)(RAM_START+128);
        /* store the value of    RAM (0x2000 0000 - 0x2000 0200) into the end of RAM */
        for(i = 128; i != 0; i--) {
            *(__IO uint32_t *)(RAM_END + i+384) = *ptr_stack;
            ptr_stack--;
        }
    ……
```

```
/* restore the value of RAM (0x2000 0000 - 0x2000 0200) from the end of RAM */
ptr_stack = (uint32_t *)(RAM_START+128);

for(i = 128; i != 0; i--) {
      *ptr_stack = *(__IO uint32_t *)(RAM_END + i + 384);
       ptr_stack--;
}
```

3. 代码测试

编译运行工程，从超级端打印窗口输出的打印信息可以看出，程序运行正常各模块功能检测中。

图 3-1. 超级端输出打印信息

```
>>>>>>>>>>>>>>  IEC60730 Test Board Init Success  <<<<<<<<<<<<<<


CPU Test(PreRun) Success!

... Power reset or software reset, next step --> FWDGT reset test ...


>>>>>>>>>>>>>>  IEC60730 Test Board Init Success  <<<<<<<<<<<<<<


CPU Test(PreRun) Success!

FWDGT reset
... FWDGT reset test OK, next step --> WWDGT reset test ...


>>>>>>>>>>>>>>  IEC60730 Test Board Init Success  <<<<<<<<<<<<<<


CPU Test(PreRun) Success!

FWDGT reset
WWDGT reset

... WWDGT reset test OK, WDGT test completed ...


Full RAM Test Success!


FLASH CRC32 Test(PreRun) Success!

Clock Frequency Test Success!

Program counter test(PreRun) Success!


****************************** Main program starts
******************************


FLASH CRC(Run-Time) Test running! Next Address --> 0x08000080
```

4. 版本历史

表 4-1. 版本历史

版本号.	说明	日期
1.0	首次发布	2022 年 9 月 20 日

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.