GigaDevice Semiconductor Inc.

GD32MCU
ARM® Cortex™-M3 32-bit MCU

应用指南 V2.0
ANxxx

目录

1. 简介

在 GD32 MCU 中，flash 有 Code Area 和 Data Area。Code Area 是运行代码区，Data Area 是数据区。它们都可以运行程序。

例如 GD32F103ZE，flash 大小为 512k。有 256K Code Area 和 256K Data Area。可以通过分散加载将不同函数或常量放置 Code Area、Data Area 区域内。

当工程师对代码运行速度有要求时，也可以通过将代码加载到 RAM 中运行，以达到预期运行效果。

2. 描述

以 GD32F103ZE 为例，分别用 Keil 和 IAR 工具实现：将函数放置某个地址、将常量放置某个地址、将函数放在 RAM 中运行的三种效果。

1、将 led_toggle()函数放在 0x08040000 地址后。

2、将 tempbuf[1024]常量放在 0x08020000 地址后。

3、将 void led_flow(void) 函数在 RAM 中运行，放在 0x20008000 地址后面。

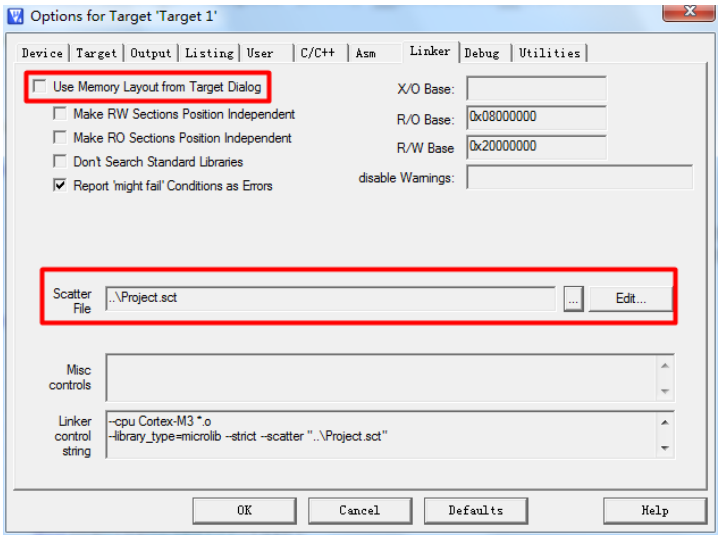
3. Keil IDE example

3.1. 将函数放置某个地址

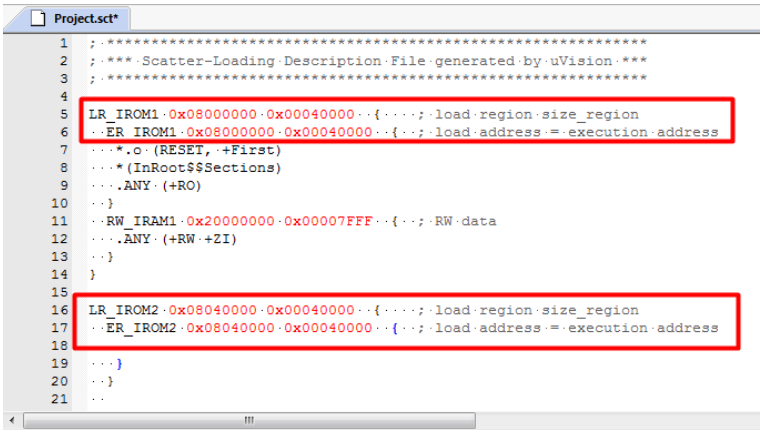
生成.sct 文件

单击 MDK 的 Option -> linker 取消勾选“Use memory layout from target Dialog”。

单击“Scatter file”中的“Edit” ， keil 工程会出现“.sct”文件。



修改.sct 文件，将 512k flash 分成 LR_IROM1 和 LR_IROM2 两个加载区域，分别为 256K 的 flash。



将 void led_toggle(void)函数添加到 LR_IROM2 地址内。

led.o 表示 led.c 文件生成的.o 文件。led_toggle 表示所添加的函数。

```
Project.sct
; *************************************************************
; *** 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)
   .ANY (+RO)
  }
  RW_IRAM1 0x20000000 0x00007FFF  {  ; RW data
   .ANY (+RW +ZI)
  }
}

LR_IROM2 0x08040000 0x00040000  {    ; load region size_region
  ER_IROM2 0x08040000 0x00040000  {  ; load address = execution address
   led.o (led_toggle)  ;led_toggle() Place after 256k flash

   }
}
```

void led_toggle(void)函数需要添加“__attribute__”，代码如下。
void led_toggle(void) __attribute__((section ("led_toggle")));其中“led_toggle”名称可以任意。

```
led.c   Project.sct   Project.map
#include "led.h"

#if defined ( __CC_ARM )

    void led_toggle(void)  __attribute__((section ("led_toggle")));
    void led_flow(void)    __attribute__((section ("led_flow")));
    const uint8_t  tempbuf [1024] __attribute__((at(0x08020000))) = {0};

#else
    void led_toggle(void)@"led_toggle";
    void led_flow(void) @"led_flow";
    const uint8_t  tempbuf [1024] @0x080200000  = {0};
#endif
```

双击工程名“Tartget1” 生成.map 文件，查看是否加载成功。

```
C:\Users\lhzhai\Desktop\王工\GD32F103ZE_Scatterloading\Template\MDK-ARM\Project.uvproj - µVision4
File  Edit  View  Project  Flash  Debug  Peripherals  Tools  SVCS  Window  Help
Project
Target 1
  Application
    main.c
    gd32f10x_it.c
    systick.c
    led.c
    uart.c
  CMSIS
    system_gd32f10x.c
  GD32F10x_Peripherals
    gd32f10x_adc.c
    gd32f10x_bkp.c
led.c   Project.sct   Project.map
#include "led.h"

#if defined ( __CC_ARM )

    void led_toggle(void)  __attribute__((sec
    void led_flow(void)    __attribute__((sec
    const uint8_t  tempbuf [1024] __attribute
#else
    void led_toggle(void)@"led_toggle";
    void led_flow(void) @"led_flow";
    const uint8_t  tempbuf [1024] @0x08020000
```

查看.map 文件。led_toggle 函数 在地址 0x08040000 后。说明加载成功。

```
led.c   Project.sct   Project.map
    Execution Region RW_IRAM1 (Base: 0x20000000, Size: 0x00002008, Max: 0x00007fff, ABSOLUTE)

    Base Addr    Size        Type   Attr      Idx    E Section Name        Object

    0x20000000   0x00000004  Data   RW        4202    .data                mc_w.l(stdout.o)
    0x20000004   0x00000004  PAD
    0x20000008   0x00002000  Zero   RW        3918    STACK                startup_gd32f10x_hd.o


  Load Region LR_IROM2 (Base: 0x08040000, Size: 0x0000006c, Max: 0x00040000, ABSOLUTE)

    Execution Region ER_IROM2 (Base: 0x08040000, Size: 0x0000006c, Max: 0x00040000, ABSOLUTE)

    Base Addr    Size        Type   Attr      Idx    E Section Name        Object

    0x08040000   0x0000006c  Code   RO         252    led_toggle           led.o


==============================================================================

Image component sizes


      Code (inc. data)   RO Data    RW Data    ZI Data      Debug   Object Name

       156          0          0          0          0       1984   gd32f10x_gpio.o
```

3.2. 将常量放置某个地址

将 tempbuf[1024]放置到 0x08020000，添加如下代码。此时可以不用修改.sct 文件。const uint8_t tempbuf [1024] __attribute__((at(0x08020000))) = {0};

```
led.c   Project.sct   Project.map
1   #include "led.h"
2
3   #if defined ( __CC_ARM )
4
5       void led_toggle(void)  __attribute__((section ("led_toggle")));
6       void led_flow(void)    __attribute__((section ("led_flow")));
7       const uint8_t  tempbuf [1024]  __attribute__((at(0x08020000)))  = {0};
8
9   #else
10      void led_toggle(void)@"led_toggle";
11      void led_flow(void) @"led_flow";
12      const uint8_t  tempbuf [1024] @0x080200000  = {0};
13  #endif
14
```

查看.map 文件。常量 tempbuf[1024]在地址 0x08020000 后。说明加载成功。

```
led.c   Project.sct   Project.map
    usart_deinit                          0x080007a5   Thumb Code   136  gd32f10x_usart.o(i.usart_deinit)
    usart_enable                          0x08000835   Thumb Code    10  gd32f10x_usart.o(i.usart_enable)
    usart_flag_get                        0x0800083f   Thumb Code    30  gd32f10x_usart.o(i.usart_flag_get)
    usart_gpio_config                     0x0800085d   Thumb Code    32  uart.o(i.usart_gpio_config)
    usart_interrupt_enable                0x08000881   Thumb Code    26  gd32f10x_usart.o(i.usart_interrupt_enable)
    usart_receive_config                  0x0800089b   Thumb Code    16  gd32f10x_usart.o(i.usart_receive_config)
    usart_transmit_config                 0x080008ab   Thumb Code    16  gd32f10x_usart.o(i.usart_transmit_config)
    led_flow                              0x080008bd   Thumb Code   180  led.o(led_flow)
    Region$$Table$$Base                   0x08000978   Number        0  anon$$obj.o(Region$$Table)
    Region$$Table$$Limit                  0x08000998   Number        0  anon$$obj.o(Region$$Table)
    tempbuf                               0x08020000   Data       1024  led.o(.ARM.__AT_0x08020000)
    led_toggle                            0x08040001   Thumb Code    96  led.o(led_toggle)
    __stdout                              0x20000000   Data           4  stdout.o(.data)
    __initial_sp                          0x20002008   Data           0  startup_gd32f10x_hd.o(STACK)
```

3.3. 将函数放置 RAM 中运行

在 led.c 文件将 led_flow(void)申明，代码添加如下。

void led_flow(void) __attribute__((section ("led_flow")));

```
led.c   Project.sct   Project.map
1   #include "led.h"
2
3   #if defined ( __CC_ARM )
4
5       void led_toggle(void)    __attribute__ ((section ("led_toggle")));
6       void led_flow(void)      __attribute__ ((section ("led_flow")));
7       const uint8_t  tempbuf [1024] __attribute__((at(0x08020000))) = {0};
8
9   #else
10      void led_toggle(void)@"led_toggle";
11      void led_flow(void) @"led_flow";
12      const uint8_t  tempbuf [1024] @0x080200000  = {0};
13  #endif
```

修改.sct 文件，将 mcu 的 64K RAM 分成 RW_IRAM1 和 RW_IRAM2 两个区，分别 32K 大小 。在 LR_IROM2 增加 RW_IRAM2 执行区域和 led_flow 函数，如图。

```
led.c   Project.sct*   Project.map
2   ; *** Scatter-Loading Description File generated by uVision ***
3   ; *************************************************************
4
5   LR_IROM1 0x08000000 0x00040000  {    ; load region size_region
6     ER_IROM1 0x08000000 0x00040000  {  ; load address = execution address
7       *.o (RESET, +First)
8       *(InRoot$$Sections)
9       .ANY (+RO)
10    }
11    RW_IRAM1 0x20000000 0x00007FFF  {  ; RW data
12      .ANY (+RW +ZI)
13    }
14  }
15
16  LR_IROM2 0x08040000 0x00040000  {    ; load region size_region
17    ER_IROM2 0x08040000 0x00040000  {  ; load address = execution address
18      led.o (led_toggle)  ;led_toggle() Place after 256k flash
19
20    }
21
22    RW_IRAM2 0x20008000 0x00008000  {  ; RW data
23      led.o (led_flow);
24    }
25
26  }
```

前32K RAM

后32K RAM

添加led_flow函数

查看.map 文件，led_flow 函数在地址 0x20008000 后。说明加载成功。

```
led.c   Project.sct   Project.map
    Execution Region ER_IROM2 (Base: 0x08040000, Size: 0x0000006c, Max: 0x00040000, ABSOLUTE)

    Base Addr    Size        Type   Attr      Idx    E Section Name        Object

    0x08040000   0x0000006c  Code   RO          252    led_toggle          led.o


    Execution Region RW_IRAM2 (Base: 0x20008000, Size: 0x000000d0, Max: 0x00008000, ABSOLUTE)

    Base Addr    Size        Type   Attr      Idx    E Section Name        Object

    0x20008000   0x0000000a  Ven    RO         4237    Veneer$$Code        anon$$obj.o
    0x2000800a   0x0000000a  Ven    RO         4238    Veneer$$Code        anon$$obj.o
    0x20008014   0x000000bc  Code   RO          251    led_flow            led.o
```

3.4. 将程序中所有 const 快速放置在 data 区域

专门划分出一块区域 0x08040000-0x08080000 放置 const 数据

```c
#include "gd32f10x.h"
#include <stdlib.h>
#include <stdio.h>
#include "uart.h"
#include "led.h"

const uint8_t constdata[1024] = {0};
uint8_t temp;

void rcu_config(void);

/*!
    \brief      main function
    \param[in]  none
    \param[out] none
    \retval     none
*/
int main(void)
{

    rcu_config();
    nvic_config();
    usart_gpio_config();
    usart_config();
    led_init();
    /* Output a message on using printf function */
    printf("\n\r Scatter loading is start \n\r");
    temp = constdata[0] + 1;
    while(1){
        led_toggle();
        led_toggle();
        led_flow();
        led_flow();
    }
}
```

修改 sct 文件如下：

```
;*************************************************************
;*** 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)
   .ANY (+RO)
  }
  RW_IRAM1 0x20000000 0x00007FFF  {  ; RW data
   .ANY (+RW +ZI)
  }
}

LR_IROM2 0x08040000 0x00040000  {    ; load region size_region
  ER_IROM2 0x08040000 0x00040000  {  ; load address = execution address
  .ANY (+const)
   led.o (led_toggle)  ;led_toggle() Place after 256k flash

   }

   RW_IRAM2 0x20008000 0x00008000  {  ; RW data
   led.o (led_flow);
   }

  }
```

代码编译出来效果如下：

main.c led.c Project.sct Project.map

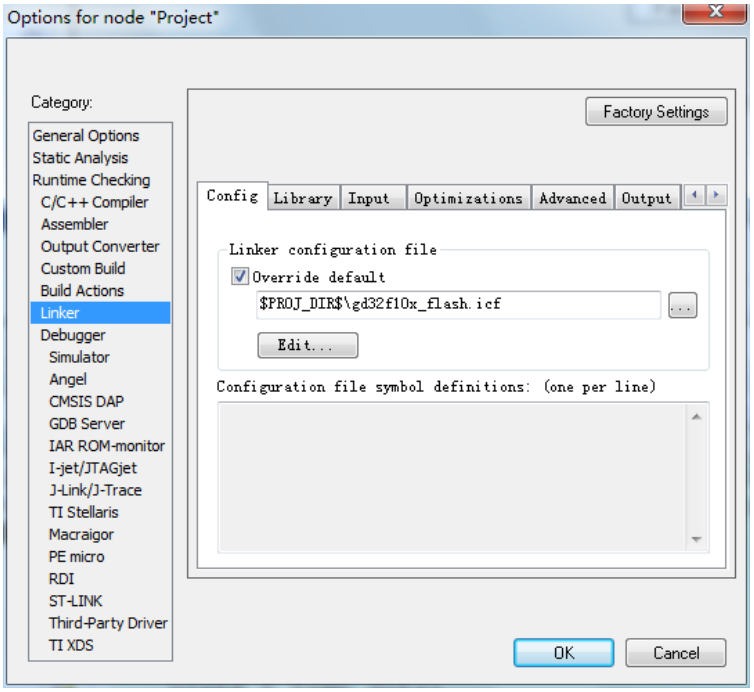
```
  Load Region LR_IROM2 (Base: 0x08040000, Size: 0x0000053c, Max: 0x00040000, ABSOLUTE)

    Execution Region ER_IROM2 (Base: 0x08040000, Size: 0x0000046c, Max: 0x00040000, ABSOLUTE)

    Base Addr    Size         Type   Attr      Idx    E Section Name        Object

    0x08040000   0x0000006c   Code   RO          256    led_toggle          led.o
    0x0804006c   0x00000400   Data   RO            6    .constdata          main.o


    Execution Region RW_IRAM2 (Base: 0x20008000, Size: 0x000000d0, Max: 0x00008000, ABSOLUTE)

    Base Addr    Size         Type   Attr      Idx    E Section Name        Object

    0x20008000   0x0000000a   Ven    RO         4241    Veneer$$Code        anon$$obj.o
    0x2000800a   0x0000000a   Ven    RO         4242    Veneer$$Code        anon$$obj.o
    0x20008014   0x000000bc   Code   RO          255    led_flow            led.o


===============================================================================
```

4. IAR IDE example

4.1. 将函数放置某个地址

打开配置文件.icf

Option -> linker -> config 勾选“override default”。单击“Edit”进行修改。



修改 icf 文件，将 512K flash 分成 ROM 和 ROM1，各 256K 加载区，添加代码如下：

define symbol __ICFEDIT_region_ROM_start__ = 0x08000000;

define symbol __ICFEDIT_region_ROM_end__ = 0x0803FFFF;

define symbol __ICFEDIT_region_ROM1_start__ = 0x08040000;

define symbol __ICFEDIT_region_ROM1_end__ = 0x0807FFFF;

define region ROM1_region=mem:[from__ICFEDIT_region_ROM1_start__to __ICFEDIT_region_ROM1_end__]

```
uart.c | led.c | led.h | main.c | gd32f10x_flash.icf | gd32f10x_it.c | Project.map | gd32f10x_rcu.c | startup_gd32f10x_hd.s
/*###ICF### Section handled by ICF editor, don't touch! ****/
/*-Editor annotation file-*/
/* IcfEditorFile="$TOOLKIT_DIR$\config\ide\IcfEditor\cortex_v1_0.xml" */
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x08000000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__   = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__     = 0x0803FFFF;  //change
define symbol __ICFEDIT_region_ROM1_start__  = 0x08040000;    //add
define symbol __ICFEDIT_region_ROM1_end__    = 0x0807FFFF;    //add

define symbol __ICFEDIT_region_RAM_start__ = 0x20000000;  //
define symbol __ICFEDIT_region_RAM_end__   = 0x20007FFF;  //change

define symbol __ICFEDIT_region_RAM1_start__ = 0x20008000;  //add
define symbol __ICFEDIT_region_RAM1_end__   = 0x2000FFFF;  //add
/*-Sizes-*/
define symbol __ICFEDIT_size_cstack__ = 0x200;
define symbol __ICFEDIT_size_heap__   = 0x200;
/**** End of ICF editor section. ###ICF###*/


define memory mem with size = 4G;

define region ROM_region   = mem:[from __ICFEDIT_region_ROM_start__   to __ICFEDIT_region_ROM_end__];

define region ROM1_region   = mem:[from __ICFEDIT_region_ROM1_start__  to __ICFEDIT_region_ROM1_end__]; //add

define region RAM_region   = mem:[from __ICFEDIT_region_RAM_start__   to __ICFEDIT_region_RAM_end__];
define region RAM1_region   = mem:[from __ICFEDIT_region_RAM1_start__   to __ICFEDIT_region_RAM1_end__];//add
```

修改 icf 文件，将 void led_toggle(void)函数放置在地址 0x08040000 后，添加代码如下。

place at address mem:0x08040000 { readonly section led_toggle };

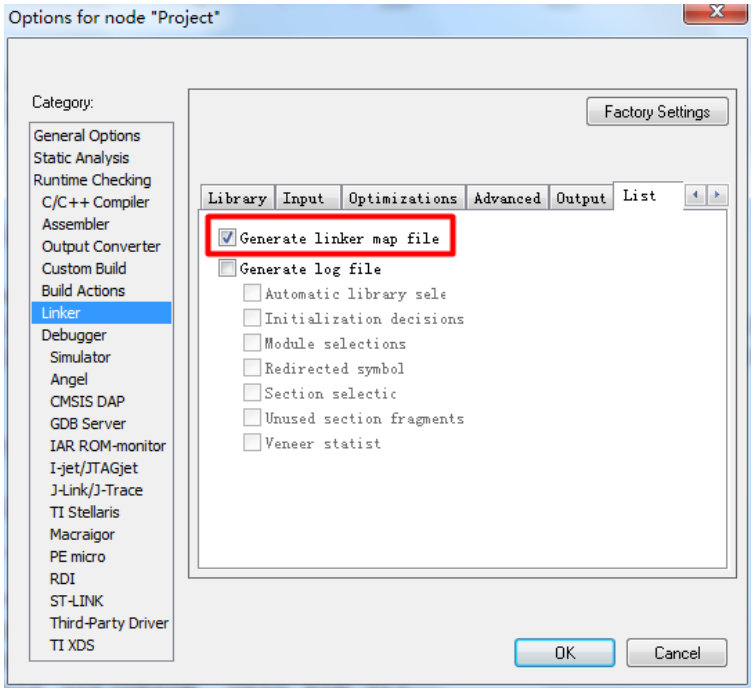
```
uart.c | led.c | led.h | main.c | gd32f10x_flash.icf | gd32f10x_it.c | Project.map | gd32f10x_rcu.c | startup_gd32f10x_hd.s
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x08000000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__   = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__     = 0x0803FFFF;  //change
define symbol __ICFEDIT_region_ROM1_start__  = 0x08040000;    //add
define symbol __ICFEDIT_region_ROM1_end__    = 0x0807FFFF;    //add

define symbol __ICFEDIT_region_RAM_start__ = 0x20000000;  //
define symbol __ICFEDIT_region_RAM_end__   = 0x20007FFF;  //change

define symbol __ICFEDIT_region_RAM1_start__ = 0x20008000;  //add
define symbol __ICFEDIT_region_RAM1_end__   = 0x2000FFFF;  //add
/*-Sizes-*/
define symbol __ICFEDIT_size_cstack__ = 0x200;
define symbol __ICFEDIT_size_heap__   = 0x200;
/**** End of ICF editor section. ###ICF###*/


define memory mem with size = 4G;

define region ROM_region   = mem:[from __ICFEDIT_region_ROM_start__   to __ICFEDIT_region_ROM_end__];

define region ROM1_region   = mem:[from __ICFEDIT_region_ROM1_start__  to __ICFEDIT_region_ROM1_end__]; //add

define region RAM_region   = mem:[from __ICFEDIT_region_RAM_start__   to __ICFEDIT_region_RAM_end__];
define region RAM1_region   = mem:[from __ICFEDIT_region_RAM1_start__   to __ICFEDIT_region_RAM1_end__];//add

define block CSTACK    with alignment = 8, size = __ICFEDIT_size_cstack__   { };
define block HEAP      with alignment = 8, size = __ICFEDIT_size_heap__     { };

initialize by copy { readwrite };
initialize by copy { readwrite,section led_flow };//add
do not initialize  { section .noinit };

place at address mem:__ICFEDIT_intvec_start__ { readonly section .intvec };

place in ROM_region   { readonly };
place at address mem:0x08040000 {  readonly section led_toggle };//put led_toggle at specified address
//place at address mem:0x08040000 {  section .rodata };
place at address mem:0x20008000 {   section led_flow };//add

place in RAM_region   { readwrite,
                        block CSTACK, block HEAP };
```

在 led.c 文件中添加函数属性。添加代码如下。

void led_toggle(void)@ “led_toggle”其中“led_toggle”名称可以任意。

```
uart.c | led.c | led.h | main.c | gd32f10x_flash.icf | gd32f10x_it.c | Project.map | gd32f10x_rcu.c | startup_gd32f10x_hd.s
#include "led.h"

uint8_t temp_data;

#if defined ( __CC_ARM )

    void led_toggle(void)  __attribute__((section ("led_toggle")));
    void led_flow(void)    __attribute__((section ("led_flow")));
    const uint8_t  tempbuf [1024] __attribute__((at(0x08020000)))   = {0};

#else
    void led_toggle(void)@"led_toggle";
    void led_flow(void) @"led_flow";
    const uint8_t  tempbuf [1024] @0x08020000  = {0};
#endif
```

生成.map 文件，查看是否加载成功。在 IAR 中 Option -> linker -> List 勾选“generate linker map file”。



查看.map 文件。led_toggle 函数 在地址 0x08040000 后。说明加载成功。

```
uart.c | led.c | led.h | main.c | gd32f10x_flash.icf | gd32f10x_it.c | Project.map
                          0x080006e1          Code  Gb  ABImemcpy_unaligned.o [4]
__iar_unaligned___aeabi_memcpy8
                          0x080006e1          Code  Gb  ABImemcpy_unaligned.o [4]
__iar_zero_init3          0x08000737   0x22   Code  Gb  zero_init3.o [4]
__low_level_init          0x08000f3f    0x4   Code  Gb  low_level_init.o [3]
__vector_table            0x08000000          Data  Gb  startup_gd32f10x_hd.o [1]
_call_main                0x08000f31          Code  Gb  cmain.o [4]
_exit                     0x08000f49          Code  Gb  cexit.o [4]
_main                     0x08000f3b          Code  Gb  cmain.o [4]
constdata                 0x08000130  0x400   Data  Gb  main.o [1]
exit                      0x08000f43    0x4   Code  Gb  exit.o [3]
fputc                     0x08000d9d   0x20   Code  Gb  main.o [1]
gpio_bit_reset            0x08000d3d    0x4   Code  Gb  gd32f10x_gpio.o [1]
gpio_bit_set              0x08000d39    0x4   Code  Gb  gd32f10x_gpio.o [1]
gpio_init                 0x08000c8d   0xac   Code  Gb  gd32f10x_gpio.o [1]
led_flow                  0x20008001   0xc4   Code  Gb  led.o [1]
led_init                  0x08000e41   0x28   Code  Gb  led.o [1]
led_toggle                0x08040001   0x74   Code  Gb  led.o [1]
main                      0x08000d49   0x38   Code  Gb  main.o [1]
nvic_config               0x08000e29   0x16   Code  Gb  uart.o [1]
nvic_irq_enable           0x08000bbf   0xbe   Code  Gb  gd32f10x_misc.o [1]
nvic_priority_group_set
```

4.2. 将常量放置某个地址

将 tempbuf[1024]放置到 0x08020000，添加如下代码。此时可以不用修改.icf 文件。const uint8_t tempbuf [1024] __attribute__((at(0x08020000))) = {0};

```
uart.c | led.c | led.h | main.c | gd32f10x_flash.icf | gd32f10x_it.c | Project.map | gd32f10x_rcu.c | startup_gd32f10x_hd.s
#include "led.h"

uint8_t temp_data;

#if defined ( __CC_ARM )

    void led_toggle(void)   __attribute__((section ("led_toggle")));
    void led_flow(void)     __attribute__((section ("led_flow")));
    const uint8_t  tempbuf [1024] __attribute__((at(0x08020000)))   = {0};

#else
    void led_toggle(void)@"led_toggle";
    void led_flow(void) @"led_flow";
    const uint8_t  tempbuf [1024] @0x08020000  = {0};
#endif
```

查看.map 文件。函数 tempbuf 在地址 0x08020000 后。说明加载成功。

```
uart.c | led.c | led.h | main.c | gd32f10x_flash.icf | gd32f10x_it.c | Project.map | gd32f10x_rcu.c | startup_gd32f10x_hd.s
                          0x08000171   0x20  Code  Gb  gd32f10x_rcu.o [1]
rcu_periph_reset_enable
                          0x08000151   0x20  Code  Gb  gd32f10x_rcu.o [1]
strlen                    0x08000639        Code  Gb  strlen.o [4]
system_clock_108m_hxtal
                          0x080006e1   0xb4  Code  Lc  system_gd32f10x.o [1]
system_clock_config       0x08000681    0x8  Code  Lc  system_gd32f10x.o [1]
temp_data                 0x20000090    0x1  Data  Gb  led.o [1]
tempbuf                   0x08020000  0x400  Data  Gb  led.o [1]
usart_baudrate_set        0x080003d9   0x90  Code  Gb  gd32f10x_usart.o [1]
usart_config              0x08000ab1   0x34  Code  Gb  uart.o [1]
usart_data_transmit       0x080004a7    0x8  Code  Gb  gd32f10x_usart.o [1]
usart_deinit              0x08000359   0x80  Code  Gb  gd32f10x_usart.o [1]
usart_enable              0x0800047d    0xa  Code  Gb  gd32f10x_usart.o [1]
usart_flag_get            0x080004af   0x20  Code  Gb  gd32f10x_usart.o [1]
usart_gpio_config         0x08000a91   0x20  Code  Gb  uart.o [1]
usart_interrupt_enable    0x080004cf   0x1c  Code  Gb  gd32f10x_usart.o [1]
usart_receive_config      0x08000497   0x10  Code  Gb  gd32f10x_usart.o [1]
usart_transmit_config     0x08000487   0x10  Code  Gb  gd32f10x_usart.o [1]
```

4.3. 将函数放在 RAM 中运行

在 led.c 文件设置 led_flow 属性，添加代码如下。void led_flow(void) @"led_flow";

```
uart.c | led.c | led.h | main.c | gd32f10x_flash.icf | gd32f10x_it.c | Project.map | gd32f10x_rcu.c | startup_gd32f10x_hd.s
#include "led.h"

uint8_t temp_data;

#if defined ( __CC_ARM )

    void led_toggle(void)   __attribute__((section ("led_toggle")));
    void led_flow(void)     __attribute__((section ("led_flow")));
    const uint8_t  tempbuf [1024] __attribute__((at(0x08020000)))   = {0};

#else
    void led_toggle(void)@"led_toggle";
    void led_flow(void) @"led_flow";
    const uint8_t  tempbuf [1024] @0x08020000  = {0};
#endif
```

修改.sct 文件，将 MCU 的 64K RAM 分成 RAM 和 RW_IRAM1 两个区，分别 32K 大小。

uart.c | led.c | led.h | main.c | gd32f10x_flash.icf | gd32f10x_it.c | Project.map | gd32f10x_rcu.c | startup_gd32f10x_hd.s

```
/*###ICF### Section handled by ICF editor, don't touch! ****/
/*-Editor annotation file-*/
/* IcfEditorFile="$TOOLKIT_DIR$\config\ide\IcfEditor\cortex_v1_0.xml" */
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x08000000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__   = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__     = 0x0803FFFF;  //change
define symbol __ICFEDIT_region_ROM1_start__  = 0x08040000;    //add
define symbol __ICFEDIT_region_ROM1_end__    = 0x0807FFFF;    //add

define symbol __ICFEDIT_region_RAM_start__  = 0x20000000;  //
define symbol __ICFEDIT_region_RAM_end__    = 0x20007FFF;  //change

define symbol __ICFEDIT_region_RAM1_start__ = 0x20008000;  //add
define symbol __ICFEDIT_region_RAM1_end__   = 0x2000FFFF;  //add
/*-Sizes-*/
define symbol __ICFEDIT_size_cstack__ = 0x200;
define symbol __ICFEDIT_size_heap__   = 0x200;
/**** End of ICF editor section. ###ICF###*/
```

将函数从 flash copy 到 RAM 中，添加代码如下。

initialize by copy { readwrite,section led_flow };

uart.c | led.c | led.h | main.c | gd32f10x_flash.icf | gd32f10x_it.c | Project.map | gd32f10x_rcu.c | startup_gd32f10x_hd.s

```
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x08000000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__   = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__     = 0x0803FFFF;  //change
define symbol __ICFEDIT_region_ROM1_start__  = 0x08040000;    //add
define symbol __ICFEDIT_region_ROM1_end__    = 0x0807FFFF;    //add

define symbol __ICFEDIT_region_RAM_start__  = 0x20000000;  //
define symbol __ICFEDIT_region_RAM_end__    = 0x20007FFF;  //change

define symbol __ICFEDIT_region_RAM1_start__ = 0x20008000;  //add
define symbol __ICFEDIT_region_RAM1_end__   = 0x2000FFFF;  //add
/*-Sizes-*/
define symbol __ICFEDIT_size_cstack__ = 0x200;
define symbol __ICFEDIT_size_heap__   = 0x200;
/**** End of ICF editor section. ###ICF###*/


define memory mem with size = 4G;

define region ROM_region   = mem:[from __ICFEDIT_region_ROM_start__   to __ICFEDIT_region_ROM_end__];

define region ROM1_region   = mem:[from __ICFEDIT_region_ROM1_start__ to __ICFEDIT_region_ROM1_end__]; //add

define region RAM_region   = mem:[from __ICFEDIT_region_RAM_start__   to __ICFEDIT_region_RAM_end__];
define region RAM1_region   = mem:[from __ICFEDIT_region_RAM1_start__   to __ICFEDIT_region_RAM1_end__];//add

define block CSTACK    with alignment = 8, size = __ICFEDIT_size_cstack__   { };
define block HEAP      with alignment = 8, size = __ICFEDIT_size_heap__     { };

initialize by copy { readwrite };
initialize by copy { readwrite,section led_flow };//add
do not initialize  { section .noinit };
```

将 void led_flow(void)函数放置地址 0x20008000 后，需要增加如下函数。如图。

place at address mem:0x20008000 { section led_flow };

```
uart.c | led.c | led.h | main.c | gd32f10x_flash.icf | gd32f10x_it.c | Project.map
define symbol __ICFEDIT_region_ROM_start__ = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__   = 0x08003FFF;//change
define symbol __ICFEDIT_region_ROM1_start__ = 0x08040000;//add
define symbol __ICFEDIT_region_ROM1_end__   = 0x0807FFFF;//add

define symbol __ICFEDIT_region_RAM_start__ = 0x20000000;//
define symbol __ICFEDIT_region_RAM_end__   = 0x20007FFF;//change

define symbol __ICFEDIT_region_RAM1_start__ = 0x20008000;//add
define symbol __ICFEDIT_region_RAM1_end__   = 0x2000FFFF;//add
/*-Sizes-*/
define symbol __ICFEDIT_size_cstack__ = 0x200;
define symbol __ICFEDIT_size_heap__   = 0x200;
/**** End of ICF editor section. ###ICF###*/


define memory mem with size = 4G;

define region ROM_region   = mem:[from __ICFEDIT_region_ROM_start__   to __ICFEDIT_region_RC

define region ROM1_region   = mem:[from __ICFEDIT_region_ROM1_start__ to __ICFEDIT_region_RC

define region RAM_region   = mem:[from __ICFEDIT_region_RAM_start__   to __ICFEDIT_region_RA
define region RAM1_region   = mem:[from __ICFEDIT_region_RAM1_start__   to __ICFEDIT_region_

define block CSTACK    with alignment = 8, size = __ICFEDIT_size_cstack__   { };
define block HEAP      with alignment = 8, size = __ICFEDIT_size_heap__     { };

initialize by copy { readwrite };
initialize by copy { readwrite,section led_flow };//add
do not initialize  { section .noinit };

place at address mem:__ICFEDIT_intvec_start__ { readonly section .intvec };



place in ROM_region   { readonly };
//place in ROM1_region   { readonly object led.o};//put led.c in ROM1
//place in ROM1_region   { readonly section led_toggle};
//place at address mem:0x08040000 { readonly  object led.o };//put led.c at specified addres
place at address mem:0x08040000 {  readonly section led_toggle };//put led_toggle at specifi

place at address mem:0x20008000 {   section led_flow };//add

place in RAM_region   { readwrite,
                        block CSTACK, block HEAP };
```

查看.map 文件，函数 led_flow 在地址 0x20008000 后说明加载成功。

```
uart.c | led.c | led.h | main.c | gd32f10x_flash.icf | gd32f10x_it.c | Project.map | gd32f10x_rcu.c | startup_gd32f10x_hd.s
_call_main                0x08000c09          Code  Gb  cmain.o [4]
_exit                     0x08000c21          Code  Gb  cexit.o [4]
_main                     0x08000c13          Code  Gb  cmain.o [4]
exit                      0x08000c1b    0x4   Code  Gb  exit.o [3]
fputc                     0x08000a67   0x20   Code  Gb  main.o [1]
gpio_bit_reset            0x08000a11    0x4   Code  Gb  gd32f10x_gpio.o [1]
gpio_bit_set              0x08000a0d    0x4   Code  Gb  gd32f10x_gpio.o [1]
gpio_init                 0x08000961   0xac   Code  Gb  gd32f10x_gpio.o [1]
led_flow                  0x20008001   0xc4   Code  Gb  led.o [1]
led_init                  0x08000b05   0x28   Code  Gb  led.o [1]
led_toggle                0x08040001   0x78   Code  Gb  led.o [1]
main                      0x08000a1d   0x2e   Code  Gb  main.o [1]
nvic_config               0x08000aed   0x16   Code  Gb  uart.o [1]
nvic_irq_enable           0x080007bf   0xbe   Code  Gb  gd32f10x_misc.o [1]
nvic_priority_group_set
```

4.4. 将程序中的 const data 快速的放置到某个区域

相关语法如下：place at address mem:0x08040000 { section .rodata };

uart.c | led.c | led.h | main.c | gd32f10x_flash.icf | gd32f10x_it.c | Project.map

```
define symbol __ICFEDIT_region_RAM1_start__ = 0x20008000;//add
define symbol __ICFEDIT_region_RAM1_end__   = 0x2000FFFF;//add
/*-Sizes-*/
define symbol __ICFEDIT_size_cstack__ = 0x200;
define symbol __ICFEDIT_size_heap__   = 0x200;
/**** End of ICF editor section. ###ICF###*/


define memory mem with size = 4G;

define region ROM_region   = mem:[from __ICFEDIT_region_ROM_start__   to __ICFEDIT_region_ROM_end__];

define region ROM1_region   = mem:[from __ICFEDIT_region_ROM1_start__ to __ICFEDIT_region_ROM1_end__]; //add

define region RAM_region   = mem:[from __ICFEDIT_region_RAM_start__   to __ICFEDIT_region_RAM_end__];
define region RAM1_region   = mem:[from __ICFEDIT_region_RAM1_start__   to __ICFEDIT_region_RAM1_end__];//ad

define block CSTACK    with alignment = 8, size = __ICFEDIT_size_cstack__   { };
define block HEAP      with alignment = 8, size = __ICFEDIT_size_heap__     { };

initialize by copy { readwrite };
initialize by copy { readwrite,section led_flow };//add
do not initialize  { section .noinit };

place at address mem:__ICFEDIT_intvec_start__ { readonly section .intvec };



place in ROM_region   { readonly };
//place in ROM1_region   { readonly object led.o};//put led.c in ROM1
//place in ROM1_region   { readonly section led_toggle};
//place at address mem:0x08040000 { readonly  object led.o };//put led.c at specified address
//place at address mem:0x08040000 {  readonly section led_toggle };//put led_toggle at specified address
place at address mem:0x08040000 {  section .rodata };
place at address mem:0x20008000 {   section led_flow };//add

place in RAM_region   { readwrite,
                        block CSTACK, block HEAP };
```

运行效果如下：

uart.c | led.c | led.h | main.c | gd32f10x_flash.icf | gd32f10x_it.c | Project.map

```
_exit                    0x08000c85          Code  Gb  cexit.o [4]
 main                    0x08000c77          Code  Gb  cmain.o [4]
 constdata               0x08040040   0x400  Data  Gb  main.o [1]
 exit                    0x08000c7f     0x4  Code  Gb  exit.o [3]
 fputc                   0x08000a71    0x20  Code  Gb  main.o [1]
 gpio_bit_reset          0x08000a11     0x4  Code  Gb  gd32f10x_gpio.o [1]
 gpio_bit_set            0x08000a0d     0x4  Code  Gb  gd32f10x_gpio.o [1]
 gpio_init               0x08000961     0xac Code  Gb  gd32f10x_gpio.o [1]
 led_flow                0x20008001    0xc4  Code  Gb  led.o [1]
 led_init                0x08000b15    0x28  Code  Gb  led.o [1]
 led_toggle              0x08000b61    0x74  Code  Gb  led.o [1]
 main                    0x08000a1d    0x38  Code  Gb  main.o [1]
 nvic_config             0x08000afd    0x16  Code  Gb  uart.o [1]
 nvic_irq_enable         0x080007bf    0xbe  Code  Gb  gd32f10x_misc.o [1]
 nvic_priority_group_set
                         0x080007b5     0xa  Code  Gb  gd32f10x_misc.o [1]
 obuf                    0x20000040    0x50  Data  Lc  xfiles.o [3]
 out                     0x080004eb    0x18  Code  Lc  xprintftiny.o [3]
 printf                  0x08000b3d    0x24  Code  Gb  printf.o [3]
 putchar                 0x08000cbd     0xc  Code  Gb  putchar.o [3]
 rcu_clock_freq_get      0x08000191   0x130  Code  Gb  gd32f10x_rcu.o [1]
 rcu_config              0x08000a55    0x1c  Code  Gb  main.o [1]
 rcu_periph_clock_enable
                         0x08000131    0x20  Code  Gb  gd32f10x_rcu.o [1]
 rcu_periph_reset_disable
                         0x08000171    0x20  Code  Gb  gd32f10x_rcu.o [1]
 rcu_periph_reset_enable
                         0x08000151    0x20  Code  Gb  gd32f10x_rcu.o [1]
 strlen                  0x08000639          Code  Gb  strlen.o [4]
 system_clock_108m_hxtal
                         0x080006e1    0xb4  Code  Lc  system_gd32f10x.o [1]
 system_clock_config     0x08000681     0x8  Code  Lc  system_gd32f10x.o [1]
 temp                    0x20000090     0x1  Data  Gb  main.o [1]
 tempbuf                 0x80200000   0x400  Data  Gb  led.o [1]
 usart_baudrate_set      0x080003d9    0x90  Code  Gb  gd32f10x_usart.o [1]
 usart_config            0x08000ac1    0x34  Code  Gb  uart.o [1]
 usart_data_transmit     0x080004a7     0x8  Code  Gb  gd32f10x_usart.o [1]
```

5. 历史版本

Revision No.	Description	Date
1.0	初始版本	2017 年 4 月 20 日
2.0	适用于新库	2017 年 12 月 14 日