芯海通用 MCU 应用笔记

芯海通用 MCU 基于 GCC 编译开发应用

V1.0

摘要

本应用笔记旨在帮助指导用户针对芯海通用 MCU 基于 GCC 环境的快速开发。芯海科技通用 MCU 提供的 pack 开发包都是仅支持芯海 CSU、MDK 或 IAR 通用集成的 IDE 工具，如果客户需要 GCC 开发，需要增加相关的开发文件。本文档中的代码部分为 GCC 开发所需文件的模版，用户可根据对应芯片型号的规格进行修改，仅供参考。

版本

历史版本	修改内容	日期
V1.0	初版生成	2022-04-06

目录

1 环境安装

GCC 编译需要支持 ARM-GCC 的编译环境和 make 命令，本文档以 windows 系统为例。

可从 ARM 官网下载 gcc Windows 32-bit 可执行文件：

https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads

make 命令在 linux 环境下直接进行编译。如果是 windows 环境，可以使用 cygwin 模拟环境，安装参考链接：

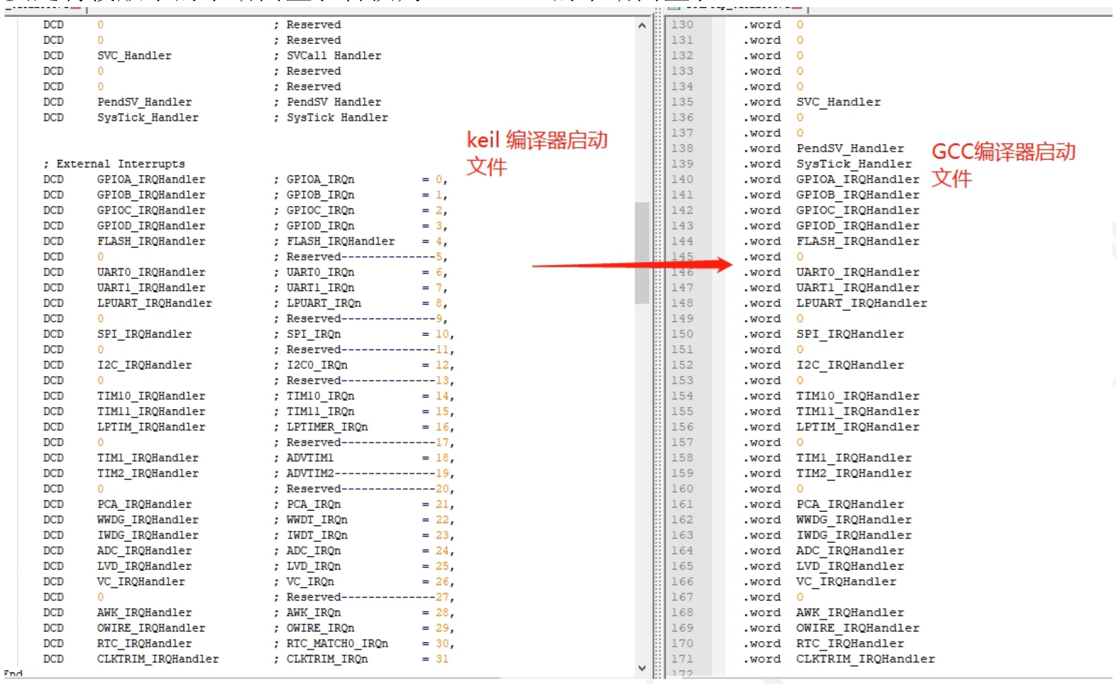
https://blog.csdn.net/thankseveryday/article/details/105653330?utm_medium=distribute.pc_aggpage_search_result.none-task-blog-2~aggregatepage~first_rank_ecpm_v1~rank_v31_ecpm-1-105653330.pc_agg_new_rank&utm_term=cygwin%E5%AE%89%E8%A3%85make&spm=1000.2123.3001.4430

2 GCC 编译环境的目录结构

```
|——Board          ;example
|   |——bsp
|   |——Common
|   |——Example
|——build          ;build result
|——CMSIS          ;cortex M0 head file
|——HAL_Driver     ;CS32L010 drive source and head file
|   |——inc
|   |——src
|——Include        ;public head file
|——Source         ;public source file
|   |——GCC        ;startup .s file
|——cs32l010.ld    ;link file
|——Makefile       ;compile file
```

3 启动.s 文件编写

以 CS32L010 为例，这里将启动文件中原始的中断向量表按格式修改为 ARM GCC 所需要的格式,主要是将模版中的中断向量表替换为 CS32L010 的中断向量表。



```
        DCD     0                          ; Reserved
        DCD     0                          ; Reserved
        DCD     SVC_Handler                ; SVCall Handler
        DCD     0                          ; Reserved
        DCD     0                          ; Reserved
        DCD     PendSV_Handler             ; PendSV Handler
        DCD     SysTick_Handler            ; SysTick Handler


        ; External Interrupts
        DCD     GPIOA_IRQHandler           ; GPIOA_IRQn          = 0,
        DCD     GPIOB_IRQHandler           ; GPIOB_IRQn          = 1,
        DCD     GPIOC_IRQHandler           ; GPIOC_IRQn          = 2,
        DCD     GPIOD_IRQHandler           ; GPIOD_IRQn          = 3,
        DCD     FLASH_IRQHandler           ; FLASH_IRQHandler    = 4,
        DCD     0                          ; Reserved--------------5,
        DCD     UART0_IRQHandler           ; UART0_IRQn          = 6,
        DCD     UART1_IRQHandler           ; UART1_IRQn          = 7,
        DCD     LPUART_IRQHandler          ; LPUART_IRQn         = 8,
        DCD     0                          ; Reserved--------------9,
        DCD     SPI_IRQHandler             ; SPI_IRQn            = 10,
        DCD     0                          ; Reserved--------------11,
        DCD     I2C_IRQHandler             ; I2C0_IRQn           = 12,
        DCD     0                          ; Reserved--------------13,
        DCD     TIM10_IRQHandler           ; TIM10_IRQn          = 14,
        DCD     TIM11_IRQHandler           ; TIM11_IRQn          = 15,
        DCD     LPTIM_IRQHandler           ; LPTIMER_IRQn        = 16,
        DCD     0                          ; Reserved--------------17,
        DCD     TIM1_IRQHandler            ; ADVTIM1             = 18,
        DCD     TIM2_IRQHandler            ; ADVTIM2---------------19,
        DCD     0                          ; Reserved--------------20,
        DCD     PCA_IRQHandler             ; PCA_IRQn            = 21,
        DCD     WWDG_IRQHandler            ; WWDT_IRQn           = 22,
        DCD     IWDG_IRQHandler            ; IWDT_IRQn           = 23,
        DCD     ADC_IRQHandler             ; ADC_IRQn            = 24,
        DCD     LVD_IRQHandler             ; LVD_IRQn            = 25,
        DCD     VC_IRQHandler              ; VC_IRQn             = 26,
        DCD     0                          ; Reserved--------------27,
        DCD     AWK_IRQHandler             ; AWK_IRQn            = 28,
        DCD     OWIRE_IRQHandler           ; OWIRE_IRQn          = 29,
        DCD     RTC_IRQHandler             ; RTC_MATCH0_IRQn     = 30,
        DCD     CLKTRIM_IRQHandler         ; CLKTRIM_IRQn        = 31
End
```

```
  .word  0
  .word  0
  .word  0
  .word  0
  .word  0
  .word  SVC_Handler
  .word  0
  .word  0
  .word  PendSV_Handler
  .word  SysTick_Handler
  .word  GPIOA_IRQHandler
  .word  GPIOB_IRQHandler
  .word  GPIOC_IRQHandler
  .word  GPIOD_IRQHandler
  .word  FLASH_IRQHandler
  .word  0
  .word  UART0_IRQHandler
  .word  UART1_IRQHandler
  .word  LPUART_IRQHandler
  .word  0
  .word  SPI_IRQHandler
  .word  0
  .word  I2C_IRQHandler
  .word  0
  .word  TIM10_IRQHandler
  .word  TIM11_IRQHandler
  .word  LPTIM_IRQHandler
  .word  0
  .word  TIM1_IRQHandler
  .word  TIM2_IRQHandler
  .word  0
  .word  PCA_IRQHandler
  .word  WWDG_IRQHandler
  .word  IWDG_IRQHandler
  .word  ADC_IRQHandler
  .word  LVD_IRQHandler
  .word  VC_IRQHandler
  .word  0
  .word  AWK_IRQHandler
  .word  OWIRE_IRQHandler
  .word  RTC_IRQHandler
  .word  CLKTRIM_IRQHandler
```

```
/*******************************************************************************
* File Name          : startup_cs32l010.s
* Author             : Software Team
* Version            : V1.0.0
* Date               : 23-12-2021
* Description        : CS32L010 vector table for GCC
*                      toolchain.
*                      This module performs:
*                      - Set the initial SP
*                      - Set the initial PC == Reset_Handler
*                      - Set the vector table entries with the exceptions ISR address
*                      - Configure the clock system
*                      - Branches to __main in the C library (which eventually
*                        calls main()).
*                      After Reset the CortexM0+ processor is in Thread mode,
*                      priority is Privileged, and the Stack is set to Main.
********************************************************************************/
  .syntax unified
  .cpu cortex-m0
  .fpu softvfp
  .thumb

.global  g_pfnVectors
.global  Default_Handler


/* start address for the initialization values of the .data section.
defined in linker script */
.word  _sidata
/* start address for the .data section. defined in linker script */
.word  _sdata
/* end address for the .data section. defined in linker script */
.word  _edata
/* start address for the .bss section. defined in linker script */
.word  _sbss
/* end address for the .bss section. defined in linker script */
.word  _ebss
/* stack used for SystemInit_ExtMemCtl; always internal RAM used */

/**
```

```
 * @brief  This is the code that gets called when the processor first
 *         starts execution following a reset event. Only the absolutely
 *         necessary set is performed, after which the application
 *         supplied main() routine is called.
 * @param  None
 * @retval : None
*/

  .section .text.Reset_Handler
  .weak Reset_Handler
  .type Reset_Handler, %function
Reset_Handler:
  ldr   r0, =_estack
  mov   sp, r0          /* set stack pointer */

ApplicationStart:
/* Copy the data segment initializers from flash to SRAM */
  movs r1, #0
  b LoopCopyDataInit

CopyDataInit:
  ldr r3, =_sidata
  ldr r3, [r3, r1]
  str r3, [r0, r1]
  adds r1, r1, #4

LoopCopyDataInit:
  ldr r0, =_sdata
  ldr r3, =_edata
  adds r2, r0, r1
  cmp r2, r3
  bcc CopyDataInit
  ldr r2, =_sbss
  b LoopFillZerobss
/* Zero fill the bss segment. */
FillZerobss:
  movs r3, #0
  str  r3, [r2]
  adds r2, r2, #4


LoopFillZerobss:
  ldr r3, = _ebss
  cmp r2, r3
  bcc FillZerobss

/* Call the clock system intitialization function.*/
    bl  SystemInit

/* Call the application's entry point.*/
  bl main

LoopForever:
    b LoopForever

.size Reset_Handler, .-Reset_Handler

/**
 * @brief  This is the code that gets called when the processor receives an
 *         unexpected interrupt.  This simply enters an infinite loop, preserving
 *         the system state for examination by a debugger.
 * @param  None
 * @retval None
*/
    .section  .text.Default_Handler,"ax",%progbits
Default_Handler:
Infinite_Loop:
  b  Infinite_Loop
  .size  Default_Handler, .-Default_Handler
/******************************************************************************
*
* The minimal vector table for a Cortex M0. Note that the proper constructs
* must be placed on this to ensure that it ends up at physical address
* 0x0000.0000.
*
```

```
*******************************************************************************/
  .section  .isr_vector,"a",%progbits
 .type  g_pfnVectors, %object
 .size  g_pfnVectors, .-g_pfnVectors

g_pfnVectors:
 .word  _estack
 .word  Reset_Handler
 .word
 .word  NMI_Handler
 .word  HardFault_Handler
 .word  0
 .word  0
 .word  0
 .word  0
 .word  0
 .word  0
 .word  0
 .word  SVC_Handler
 .word  0
 .word  0
 .word  PendSV_Handler
 .word  SysTick_Handler
 .word  GPIOA_IRQHandler
 .word  GPIOB_IRQHandler
 .word  GPIOC_IRQHandler
 .word  GPIOD_IRQHandler
 .word  FLASH_IRQHandler
 .word  0
 .word  UART0_IRQHandler
 .word  UART1_IRQHandler
 .word  LPUART_IRQHandler
 .word  0
 .word  SPI_IRQHandler
 .word  0
 .word  I2C_IRQHandler
 .word  0
 .word  TIM10_IRQHandler
 .word  TIM11_IRQHandler
 .word  LPTIM_IRQHandler
 .word  0
 .word  TIM1_IRQHandler
 .word  TIM2_IRQHandler
 .word  0
 .word  PCA_IRQHandler
 .word  WWDG_IRQHandler
 .word  IWDG_IRQHandler
 .word  ADC_IRQHandler
 .word  LVD_IRQHandler
 .word  VC_IRQHandler
 .word  0
 .word  AWK_IRQHandler
 .word  OWIRE_IRQHandler
 .word  RTC_IRQHandler
 .word  CLKTRIM_IRQHandler

/*******************************************************************************
*
* Provide weak aliases for each Exception handler to the Default_Handler.
* As they are weak aliases, any function with the same name will override
* this definition.
*
*******************************************************************************/

 .weak  NMI_Handler
 .thumb_set NMI_Handler,Default_Handler
 .weak  HardFault_Handler
 .thumb_set HardFault_Handler,Default_Handler
 .weak  SVC_Handler
 .thumb_set SVC_Handler,Default_Handler
 .weak  PendSV_Handler
 .thumb_set PendSV_Handler,Default_Handler
 .weak  SysTick_Handler
 .thumb_set SysTick_Handler,Default_Handler
 .weak  GPIOA_IRQHandler
```

```
.thumb_set GPIOA_IRQHandler,Default_Handler
.weak  GPIOB_IRQHandler
.thumb_set GPIOB_IRQHandler,Default_Handler
.weak  GPIOC_IRQHandler
.thumb_set GPIOC_IRQHandler,Default_Handler
.weak  GPIOD_IRQHandler
.thumb_set GPIOD_IRQHandler,Default_Handler
.weak  FLASH_IRQHandler
.thumb_set FLASH_IRQHandler,Default_Handler
.weak  UART0_IRQHandler
.thumb_set UART0_IRQHandler,Default_Handler
.weak  UART1_IRQHandler
.thumb_set UART1_IRQHandler,Default_Handler
.weak  LPUART_IRQHandler
.thumb_set LPUART_IRQHandler,Default_Handler
.weak  SPI_IRQHandler
.thumb_set SPI_IRQHandler,Default_Handler
.weak  I2C_IRQHandler
.thumb_set I2C_IRQHandler,Default_Handler
.weak  TIM10_IRQHandler
.thumb_set TIM10_IRQHandler,Default_Handler
.weak  TIM11_IRQHandler
.thumb_set TIM11_IRQHandler,Default_Handler
.weak  LPTIM_IRQHandler
.thumb_set LPTIM_IRQHandler,Default_Handler
.weak  TIM1_IRQHandler
.thumb_set TIM1_IRQHandler,Default_Handler
.weak  TIM2_IRQHandler
.thumb_set TIM2_IRQHandler,Default_Handler
.weak  PCA_IRQHandler
.thumb_set PCA_IRQHandler,Default_Handler
.weak  WWDG_IRQHandler
.thumb_set WWDG_IRQHandler,Default_Handler
.weak  IWDG_IRQHandler
.thumb_set IWDG_IRQHandler,Default_Handler
.weak  ADC_IRQHandler
.thumb_set ADC_IRQHandler,Default_Handler
.weak  LVD_IRQHandler
.thumb_set LVD_IRQHandler,Default_Handler
.weak  VC_IRQHandler
.thumb_set VC_IRQHandler,Default_Handler
.weak  AWK_IRQHandler
.thumb_set AWK_IRQHandler,Default_Handler
.weak  OWIRE_IRQHandler
.thumb_set OWIRE_IRQHandler,Default_Handler
.weak  RTC_IRQHandler
.thumb_set RTC_IRQHandler,Default_Handler
.weak  CLKTRIM_IRQHandler
.thumb_set CLKTRIM_IRQHandler,Default_Handler
```

4 .ld 文件编写

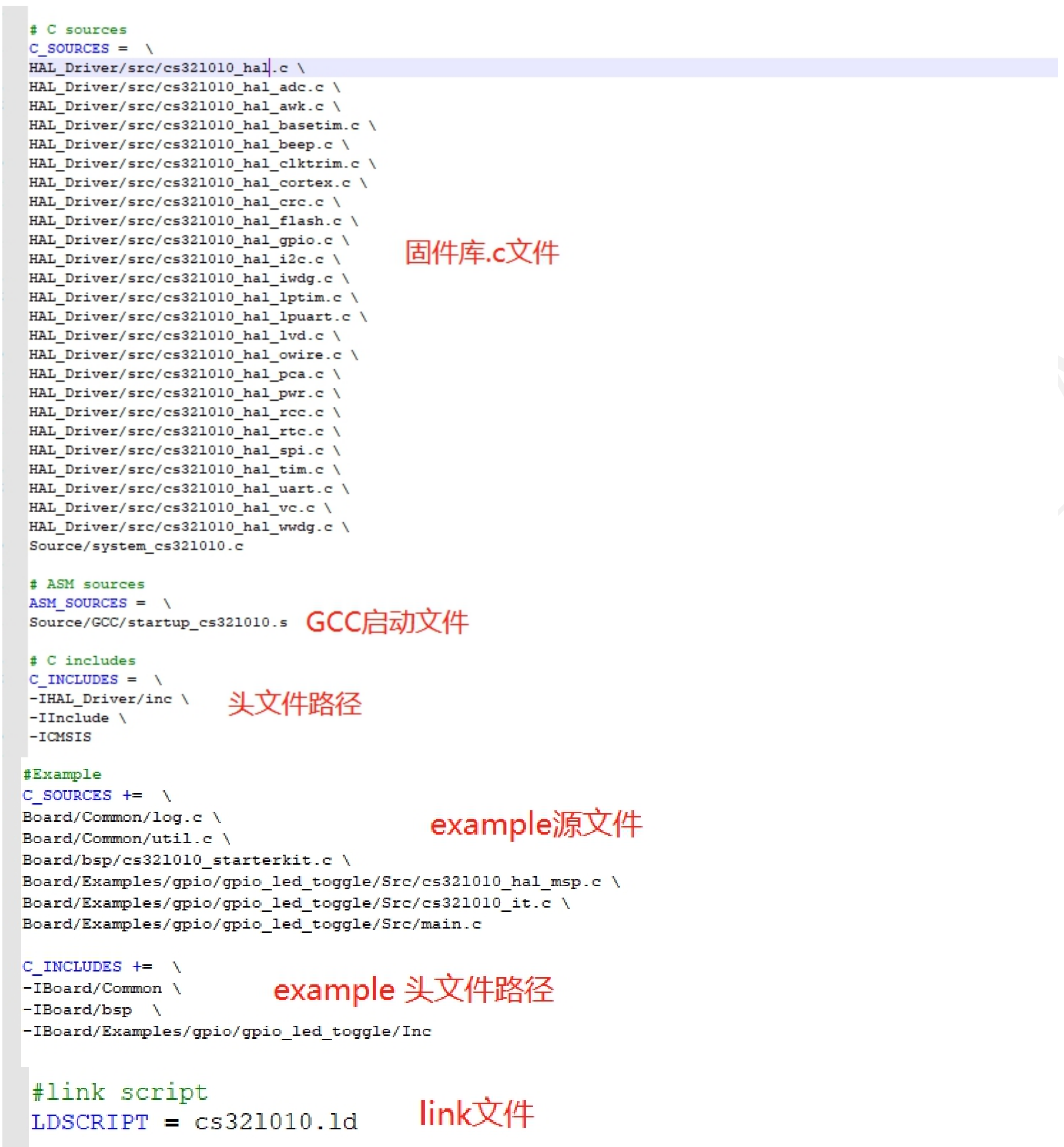
.ld 文件主要用于 GCC 编译链接，需要设置 MCU RAM 和 ROM 的起始地址和大小，以及堆栈的大小，以 CS32L010 为例，只需将模版中标红的内容设置为和实际芯片一致即可。

```
/* cs32l010 ld file for gcc compile*/

/* Entry Point */
ENTRY(Reset_Handler)

/* Highest address of the user mode stack */
_estack = 0x20001000;    /* end of RAM */
/* Generate a link error if heap and stack don't fit into RAM */
_Min_Heap_Size = 0x200;      /* required amount of heap  */
_Min_Stack_Size = 0x200; /* required amount of stack */

/* Specify the memory areas */
MEMORY
{
RAM (xrw)      : ORIGIN = 0x20000000, LENGTH = 4K
FLASH (rx)      : ORIGIN = 0x00000000, LENGTH = 64K
}
```

```
/* Define output sections */
SECTIONS
{
 /* The startup code goes first into FLASH */
 .isr_vector :
 {
  . = ALIGN(4);
  KEEP(*(.isr_vector)) /* Startup code */
  . = ALIGN(4);
 } >FLASH

 /* The program code and other data goes into FLASH */
 .text :
 {
  . = ALIGN(4);
  *(.text)           /* .text sections (code) */
  *(.text*)          /* .text* sections (code) */
  *(.glue_7)         /* glue arm to thumb code */
  *(.glue_7t)        /* glue thumb to arm code */
  *(.eh_frame)       /*                        */

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

  . = ALIGN(4);
  _etext = .;        /* define a global symbols at end of code */
 } >FLASH

 /* Constant data goes into FLASH */
 .rodata :
 {
  . = ALIGN(4);
  *(.rodata)         /* .rodata sections (constants, strings, etc.) */
  *(.rodata*)        /* .rodata* sections (constants, strings, etc.) */
  . = ALIGN(4);
 } >FLASH

 .ARM.extab   : { *(.ARM.extab* .gnu.linkonce.armextab.*) } >FLASH
 .ARM : {
  __exidx_start = .;
  *(.ARM.exidx*)
  __exidx_end = .;
 } >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

 /* used by the startup to initialize data */
 _sidata = LOADADDR(.data);

 /* Initialized data sections goes into RAM, load LMA copy after code */
 .data :
 {
  . = ALIGN(4);
  _sdata = .;        /* create a global symbol at data start */
  *(.data)           /* .data sections */
  *(.data*)         /* .data* sections */

  . = ALIGN(4);
  _edata = .;        /* define a global symbol at data end */
} >RAM AT> FLASH


/* Uninitialized data section */
. = ALIGN(4);
.bss :
{
  /* This is used by the startup in order to initialize the .bss secion */
  _sbss = .;         /* define a global symbol at bss start */
  __bss_start__ = _sbss;
  *(.bss)
  *(.bss*)
  *(COMMON)

  . = ALIGN(4);
  _ebss = .;         /* define a global symbol at bss end */
  __bss_end__ = _ebss;
} >RAM

/* User_heap_stack section, used to check that there is enough RAM left */
._user_heap_stack :
{
  . = ALIGN(8);
  PROVIDE ( end = . );
  PROVIDE ( _end = . );
  . = . + _Min_Heap_Size;
  . = . + _Min_Stack_Size;
  . = ALIGN(8);
} >RAM



/* Remove information from the standard libraries */
/DISCARD/ :
{
  libc.a ( * )
  libm.a ( * )
  libgcc.a ( * )
}

.ARM.attributes 0 : { *(.ARM.attributes) }
}
```

5 Makefile 编写

Makefile 文件用于编译生成需要的 hex 和 bin 文件，以 CS32L010 为例，Makefile 文件中需要添加 CS32L010 的固件库.c 文件和对应的头文件路径，准备好的 GCC 启动文件.s 和链接文件.ld，如下图所示。



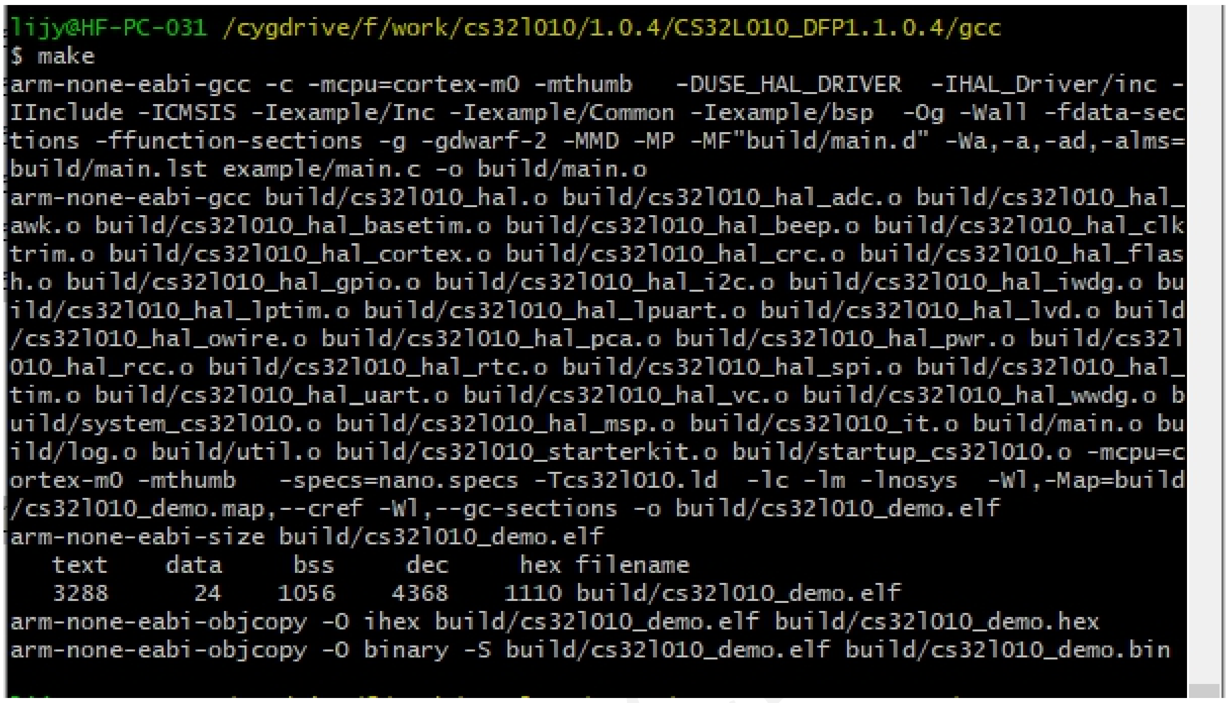
```
# C sources
C_SOURCES =  \
HAL_Driver/src/cs321010_hal.c \
HAL_Driver/src/cs321010_hal_adc.c \
HAL_Driver/src/cs321010_hal_awk.c \
HAL_Driver/src/cs321010_hal_basetim.c \
HAL_Driver/src/cs321010_hal_beep.c \
HAL_Driver/src/cs321010_hal_clktrim.c \
HAL_Driver/src/cs321010_hal_cortex.c \
HAL_Driver/src/cs321010_hal_crc.c \
HAL_Driver/src/cs321010_hal_flash.c \
HAL_Driver/src/cs321010_hal_gpio.c \
HAL_Driver/src/cs321010_hal_i2c.c \
HAL_Driver/src/cs321010_hal_iwdg.c \
HAL_Driver/src/cs321010_hal_lptim.c \
HAL_Driver/src/cs321010_hal_lpuart.c \
HAL_Driver/src/cs321010_hal_lvd.c \
HAL_Driver/src/cs321010_hal_owire.c \
HAL_Driver/src/cs321010_hal_pca.c \
HAL_Driver/src/cs321010_hal_pwr.c \
HAL_Driver/src/cs321010_hal_rcc.c \
HAL_Driver/src/cs321010_hal_rtc.c \
HAL_Driver/src/cs321010_hal_spi.c \
HAL_Driver/src/cs321010_hal_tim.c \
HAL_Driver/src/cs321010_hal_uart.c \
HAL_Driver/src/cs321010_hal_vc.c \
HAL_Driver/src/cs321010_hal_wwdg.c \
Source/system_cs321010.c

# ASM sources
ASM_SOURCES =  \
Source/GCC/startup_cs321010.s

# C includes
C_INCLUDES =  \
-IHAL_Driver/inc \
-IInclude \
-ICMSIS

#Example
C_SOURCES +=  \
Board/Common/log.c \
Board/Common/util.c \
Board/bsp/cs321010_starterkit.c \
Board/Examples/gpio/gpio_led_toggle/Src/cs321010_hal_msp.c \
Board/Examples/gpio/gpio_led_toggle/Src/cs321010_it.c \
Board/Examples/gpio/gpio_led_toggle/Src/main.c

C_INCLUDES +=  \
-IBoard/Common \
-IBoard/bsp  \
-IBoard/Examples/gpio/gpio_led_toggle/Inc

#link script
LDSCRIPT = cs321010.ld
```

```
#CS32L010 makefile template
TARGET = cs321010_demo
#define file format and name

# debug build?
DEBUG = 1
# optimization
OPT = -Og

BUILD_DIR = build

# C sources
C_SOURCES =  \
HAL_Driver/src/cs321010_hal.c \
HAL_Driver/src/cs321010_hal_adc.c \
HAL_Driver/src/cs321010_hal_awk.c \
HAL_Driver/src/cs321010_hal_basetim.c \
HAL_Driver/src/cs321010_hal_beep.c \
HAL_Driver/src/cs321010_hal_clktrim.c \
HAL_Driver/src/cs321010_hal_cortex.c \
HAL_Driver/src/cs321010_hal_crc.c \
HAL_Driver/src/cs321010_hal_flash.c \
HAL_Driver/src/cs321010_hal_gpio.c \
HAL_Driver/src/cs321010_hal_i2c.c \
HAL_Driver/src/cs321010_hal_iwdg.c \
HAL_Driver/src/cs321010_hal_lptim.c \
HAL_Driver/src/cs32l010_hal_lpuart.c \
HAL_Driver/src/cs32l010_hal_lvd.c \
HAL_Driver/src/cs32l010_hal_owire.c \
HAL_Driver/src/cs32l010_hal_pca.c \
HAL_Driver/src/cs32l010_hal_pwr.c \
HAL_Driver/src/cs32l010_hal_rcc.c \
HAL_Driver/src/cs32l010_hal_rtc.c \
HAL_Driver/src/cs32l010_hal_spi.c \
HAL_Driver/src/cs32l010_hal_tim.c \
HAL_Driver/src/cs32l010_hal_uart.c \
HAL_Driver/src/cs32l010_hal_vc.c \
HAL_Driver/src/cs32l010_hal_wwdg.c \
Source/system_cs32l010.c

# ASM sources
ASM_SOURCES =  \
Source/GCC/startup_cs32l010.s

# C includes
C_INCLUDES =  \
-IHAL_Driver/inc \
-IInclude \
-ICMSIS

#Example
C_SOURCES +=  \
example/cs32l010_hal_msp.c \
example/cs32l010_it.c \
example/main.c \
example/Common/log.c \
example/Common/util.c \
example/bsp/cs32l010_starterkit.c

C_INCLUDES +=  \
-Iexample/Inc \
-Iexample/Common \
-Iexample/bsp

# binaries
PREFIX = arm-none-eabi-
# The gcc compiler bin path can be either defined in make command via GCC_PATH variable (> make GCC_PATH=xxx)
# either it can be added to the PATH environment variable.
ifdef GCC_PATH
CC = $(GCC_PATH)/$(PREFIX)gcc
AS = $(GCC_PATH)/$(PREFIX)gcc -x assembler-with-cpp
CP = $(GCC_PATH)/$(PREFIX)objcopy
SZ = $(GCC_PATH)/$(PREFIX)size
else
CC = $(PREFIX)gcc
AS = $(PREFIX)gcc -x assembler-with-cpp
CP = $(PREFIX)objcopy
SZ = $(PREFIX)size
endif
HEX = $(CP) -O ihex
BIN = $(CP) -O binary -S

# CFLAGS
# cpu
CPU = -mcpu=cortex-m0

# mcu
MCU = $(CPU) -mthumb $(FPU) $(FLOAT-ABI)

# c compile symbol
CFLAGS = $(MCU) $(DEFS) $(INCDIRS) -std=c99 -Wall -Wfatal-errors \
        -MMD -fdata-sections -ffunction-sections
ASFLAGS = $(CFLAGS) $(AS_DEFS)

#c define
C_DEFS =  \
-DUSE_HAL_DRIVER \

# compile gcc flags
ASFLAGS = $(MCU) $(AS_DEFS) $(AS_INCLUDES) $(OPT) -Wall -fdata-sections -ffunction-sections
```

```
CFLAGS = $(MCU) $(C_DEFS) $(C_INCLUDES) $(OPT) -Wall -fdata-sections -ffunction-sections

ifeq ($(DEBUG), 1)
CFLAGS += -g -gdwarf-2
endif

# Generate dependency information
CFLAGS += -MMD -MP -MF"$(@:%.o=%.d)"

#link script
LDSCRIPT = cs32l010.ld

# libraries
LIBS = -lc -lm -lnosys
LIBDIR =
LDFLAGS = $(MCU) -specs=nano.specs -T$(LDSCRIPT) $(LIBDIR) $(LIBS) -Wl,-Map=$(BUILD_DIR)/$(TARGET).map,--cref -Wl,--gc-
sections

# default action: build all
all: $(BUILD_DIR)/$(TARGET).elf $(BUILD_DIR)/$(TARGET).hex $(BUILD_DIR)/$(TARGET).bin

#object rule
OBJECTS = $(addprefix $(BUILD_DIR)/,$(notdir $(C_SOURCES:.c=.o)))
vpath %.c $(sort $(dir $(C_SOURCES)))
# list of ASM program objects
OBJECTS += $(addprefix $(BUILD_DIR)/,$(notdir $(ASM_SOURCES:.s=.o)))
vpath %.s $(sort $(dir $(ASM_SOURCES)))

$(BUILD_DIR)/%.o: %.c Makefile | $(BUILD_DIR)
	$(CC) -c $(CFLAGS) -Wa,-a,-ad,-alms=$(BUILD_DIR)/$(notdir $(<:.c=.lst)) $< -o $@

$(BUILD_DIR)/%.o: %.s Makefile | $(BUILD_DIR)
	$(AS) -c $(CFLAGS) $< -o $@

$(BUILD_DIR)/$(TARGET).elf: $(OBJECTS) Makefile
	$(CC) $(OBJECTS) $(LDFLAGS) -o $@
	$(SZ) $@

$(BUILD_DIR)/%.hex: $(BUILD_DIR)/%.elf | $(BUILD_DIR)
	$(HEX) $< $@

$(BUILD_DIR)/%.bin: $(BUILD_DIR)/%.elf | $(BUILD_DIR)
	$(BIN) $< $@

$(BUILD_DIR):
	mkdir $@

# clean up
clean:
	-rm -fR $(BUILD_DIR)

# dependencies
-include $(wildcard $(BUILD_DIR)/*.d)
```

6 编译测试

在准备好启动.s 文件，链接.ld 文件和 Makefile 后，执行 make 编译生成需要的 hex 和 bin 文件，如下图所示。



```
lijy@HF-PC-031 /cygdrive/f/work/cs32l010/1.0.4/CS32L010_DFP1.1.0.4/gcc
$ make
arm-none-eabi-gcc -c -mcpu=cortex-m0 -mthumb   -DUSE_HAL_DRIVER  -IHAL_Driver/inc -
IInclude -ICMSIS -Iexample/Inc -Iexample/Common -Iexample/bsp  -Og -Wall -fdata-sec
tions -ffunction-sections -g -gdwarf-2 -MMD -MP -MF"build/main.d" -Wa,-a,-ad,-alms=
build/main.lst example/main.c -o build/main.o
arm-none-eabi-gcc build/cs32l010_hal.o build/cs32l010_hal_adc.o build/cs32l010_hal_
awk.o build/cs32l010_hal_basetim.o build/cs32l010_hal_beep.o build/cs32l010_hal_clk
trim.o build/cs32l010_hal_cortex.o build/cs32l010_hal_crc.o build/cs32l010_hal_flas
h.o build/cs32l010_hal_gpio.o build/cs32l010_hal_i2c.o build/cs32l010_hal_iwdg.o bu
ild/cs32l010_hal_lptim.o build/cs32l010_hal_lpuart.o build/cs32l010_hal_lvd.o build
/cs32l010_hal_owire.o build/cs32l010_hal_pca.o build/cs32l010_hal_pwr.o build/cs32l
010_hal_rcc.o build/cs32l010_hal_rtc.o build/cs32l010_hal_spi.o build/cs32l010_hal_
tim.o build/cs32l010_hal_uart.o build/cs32l010_hal_vc.o build/cs32l010_hal_wwdg.o b
uild/system_cs32l010.o build/cs32l010_hal_msp.o build/cs32l010_it.o build/main.o bu
ild/log.o build/util.o build/cs32l010_starterkit.o build/startup_cs32l010.o -mcpu=c
ortex-m0 -mthumb   -specs=nano.specs -Tcs32l010.ld  -lc -lm -lnosys  -Wl,-Map=build
/cs32l010_demo.map,--cref -Wl,--gc-sections -o build/cs32l010_demo.elf
arm-none-eabi-size build/cs32l010_demo.elf
   text    data     bss     dec     hex filename
   3288      24    1056    4368    1110 build/cs32l010_demo.elf
arm-none-eabi-objcopy -O ihex build/cs32l010_demo.elf build/cs32l010_demo.hex
arm-none-eabi-objcopy -O binary -S build/cs32l010_demo.elf build/cs32l010_demo.bin
```