GigaDevice Semiconductor Inc.

Migration from GD32E230 series to GD32F3x0 series

Application Note
AN046

Table of Contents

List of Figures

List of Tables

1. Introduction

This application note is designed to help you quickly migrate applications from GD32E230xx series MCU to GD32F3x0 series MCU.

In order to make better use of the information in this application note, you need to download it from the website www.GD32MCU.com, such as datasheet, user manual, official code and various development tools.

2. Introduction to hardware differences

The package types of GD32E230xx series include: TSSOP20, LGA20, QFN28, QFN32, LQFP32 and LQFP48; The package types of GD32F3x0 series include:TSSOP20 (GD32F330/F310xx series only), QFN28, QFN32, LQFP32(GD32F310xx series only), LQFP48, LQFP64(GD32F330/F350xx series only). The chip pins of the same package of the two series are compatible, see ***Figure 2-1. Comparison diagram of LQFP48 package of GD32F3x0 and GD32E230xx***, ***Figure 2-2. Comparison diagram of QFN32 package of GD32F3x0 and GD32E230xx***, ***Figure 2-3. Comparison diagram of QFN28 package of GD32F3x0 and GD32E230xx***, ***Figure 2-4. Comparison diagram of TSSOP20 package of GD32F330/F310xx and GD32E230xx***, ***Figure 2-5. Comparison diagram of LQFP32 package of GD32F310xx and GD32E230xx***.

1. In the package of TSSOP20 and QFN28, PA9 and PA10 of GD32E230xx series can be mapped to PA11 and PA12. GD32F3x0 series does not have this function.
2. LQFP48 package's pin 1 is V_{DD} on GD32E230xx series and Vbat on GD32F3x0, that is, GD32E230xx does not support power down RTC.

Figure 2-1. Comparison diagram of LQFP48 package of GD32F3x0 and GD32E230xx

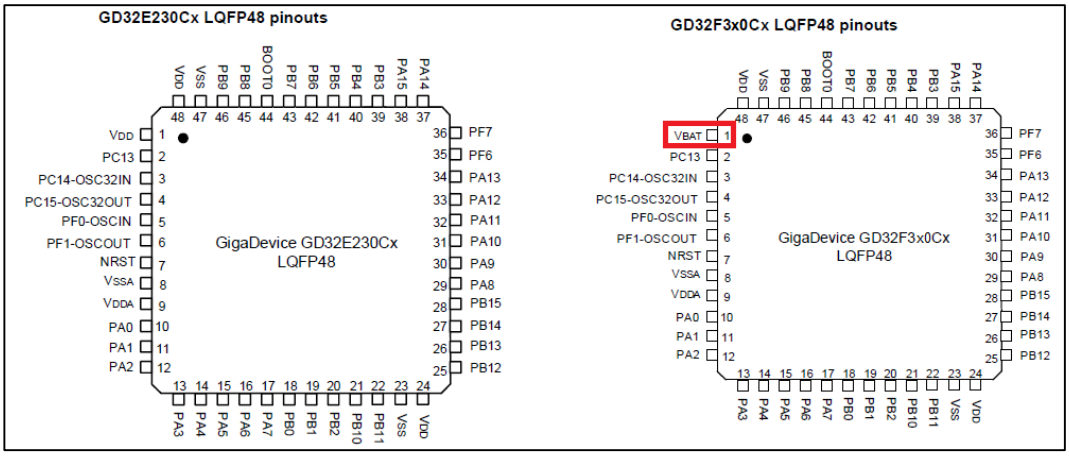


Figure 2-2. Comparison diagram of QFN32 package of GD32F3x0 and GD32E230xx

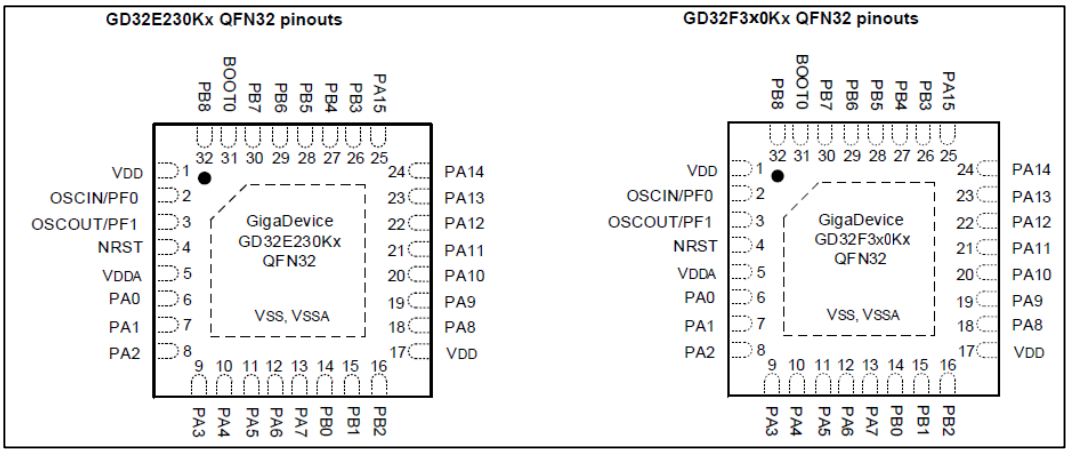


Figure 2-3. Comparison diagram of QFN28 package of GD32F3x0 and GD32E230xx

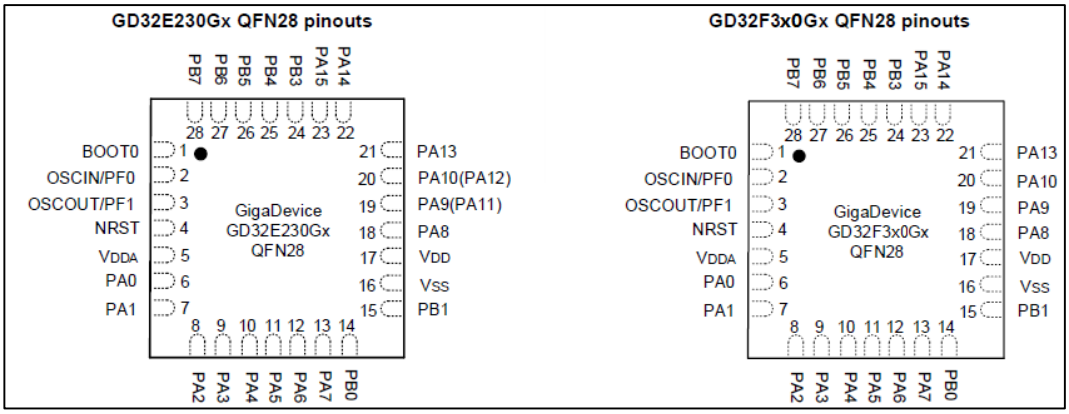


Figure 2-4. Comparison diagram of TSSOP20 package of GD32F330/F310xx and GD32E230xx

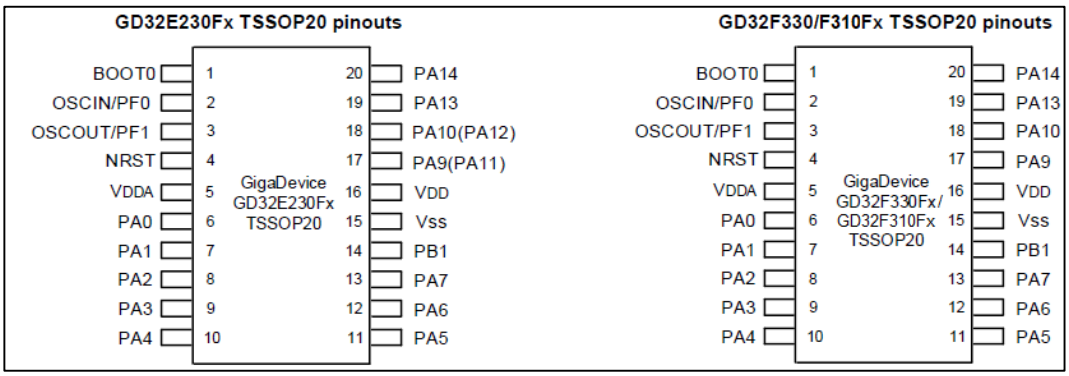
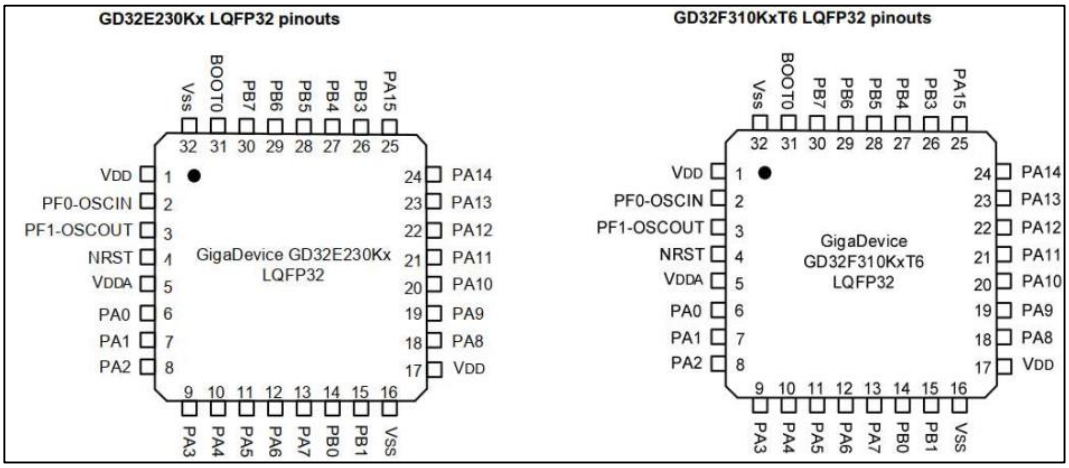


Figure 2-5. Comparison diagram of LQFP32 package of GD32F310xx and GD32E230xx



3. Comparison of resource and peripheral addresses

The resources of GD32F3x0 series and GD32E230xx series are slightly different：

1. TIMER1 is added to GD32F3x0 series, but TIMER5 is cut out(GD32F350xx has this peripheral). GD32E230xx series has TIMER5, but there is no TIMER1;
2. GD32E230xx and GD32F350xx series has a comparator, there is no one in GD32F330/F310xx;
3. GD32E230xx series adds 1K OTP area, which is not available in GD32F3x0 series;
4. GD32F350xx series has USBFS, HDMI-CEC and DAC peripheral, GD32F330/F310xx series and GD32E230xx series do not have these peripheral.

Please check for details in ***Table 3-1. GD32F3x0 series and gd32E230xx series resources comparison overview*** and ***Table 3-2. GD32F3x0 series and GD32E230xx series peripheral address comparison overview***.

Table 3-1. GD32F3x0 series and gd32E230xx series resources comparison overview

Peripheral	GD32F310xx	GD32F330xx	GD32F350xx	GD32E230xx
Core	Cortex-M4	Cortex-M4	Cortex-M4	Cortex-M23
Flash	16K-64K	16K-128K	16K-128K	16K-64K
RAM	4K-8K	4K-16K	4K-16K	4K-8K
Frequency	72MHz	84MHz	108MHz	72MHz
GPTM(32bit)	0	1	1	0
GPTM(16bit)	4/5	4/5	5	4/5
AdvTM	1	1	1	1
BaseTM	0	0	1	1
U(S)ART	1/2	1/2	1/2	1/2
I2C	1/2	1/2	1/2	1/2
SPI	1/2	1/2	1/2	1/2
I2S	1	0	1	1
USBFS	0	0	1	0
HDMI-CEC	0	0	1	0
TSI	0	0	1	0
COMP	0	0	2	1
ADC	1(9)/1(10)	1(9)/1(10)/1(16)	1(10)/1(16)	1(9)/1(10)
DAC	0	0	1	0
Operating Voltage	2.6-3.6V	2.6-3.6V	2.6-3.6V	1.8-3.6V
Temperature Range	-40-85℃	-40-85℃	-40-85℃	-40-85℃

Note: The above "/" represents a variety of situations, which need to be distinguished according to the specific chip part number.

Table 3-2. GD32F3x0 series and GD32E230xx series peripheral address comparison

overview

Peripheral	BUS	GD32F3x0	GD32E230xx
GPIOF	AHB2	0X48001400	0X48001400
GPIOD		0X48000C00	-
GPIOC		0X48000800	0x48000800
GPIOB		0X48000400	0X48000400
GPIOA		0X48000000	0X48000000
USBFS	AHB1	0X50000000	-
TSI		0X40024400	-
CRC		0X40023000	0X40023000
FMC		0X40022000	0X40022000
RCU		0X40021000	0X40021000
DMA		0X40020000	0X40020000
DBG	APB2	0xE0042000	0X40015800
TIMER16		0X40014800	0X40014800
TIMER15		0X40014400	0X40014400
TIMER14		0X40014000	0X40014000
USART0		0X40013800	0X40013800
SPI0/I2S0		0X40013000	0X40013000
TIMER0		0X40012C00	0X40012C00
ADC		0X40012400	0X40012400
EXTI		0X40010400	0X40010400
SYSCFG+CMP		0X40010000	0X40010000
CTC	APB1	0X4000C800	-
CEC		0X40007800	-
DAC		0X40007800	-
PMU		0X40007000	0X40007000
I2C1		0X40005800	0X40005800
I2C0		0X40005400	0X40005400
USART1		0X40004400	0X40004400
SPI1		0X40003800	0X40003800
FWDGT		0X40003000	0X40003000
WWDGT		0X40002C00	0X40002C00
RTC		0X40002800	0X40002800
TIMER13		0X40002000	0X40002000
TIMER5		0X40001000	0X40001000
TIMER2		0X40000400	0X40000400
TIMER1		0X40000000	-
SRAM		0x20000000	0x20000000
Option Byte		0x1FFFF800	0x1FFFF800
Main Flash		0x08000000	0x08000000
System Memory		0x1FFFEC00	0x1FFFEC00

OTP		-	0x1FFF7000

4. Comparison of development tools

GD32F3x0 can be developed by using Keil4 and Keil5 of MDK for arm. It is recommended to install version 4.74 or above when using Keil4; Using Keil5, it is recommended to install version 5.20 or above. You can also use IAR for ARM development. It is recommended to install IAR 6.3 or above, As shown in ***Table 4-1. Comparison of IDE environment between GD32F3x0 series and GD32E230xx series***.

Table 4-1. Comparison of IDE environment between GD32F3x0 series and GD32E230xx series

MCU series	GD32F3x0	GD32E230xx
KEIL	Both Keil4 and Keil5 support	KEIL 5.25 or above
IAR	IAR 6.3 or above	IAR 8.1 or above

GD32F3x0 can be developed with debugging tools such as J-LINK, ULINK and GD-LINK. As shown in ***Table 4-2. Comparison of GD32F3x0 series and GD32E230xx series debugging tools***.

Table 4-2. Comparison of GD32F3x0 series and GD32E230xx series debugging tools

MCU series	GD32F3x0	GD32E230xx
JLINK	JLINK ob, V8 and V9 all support	Only JLINK V9 and above
ULINK	support	support
GDLINK	support	support

5. Software environment settings

5.1. Using Keil to develop GD32F3x0

At present, the common MDK for arm versions on the market include Keil4 and Keil5: it is recommended to install version 4.74 or above for Keil4 and version 5.20 or above for Keil5.

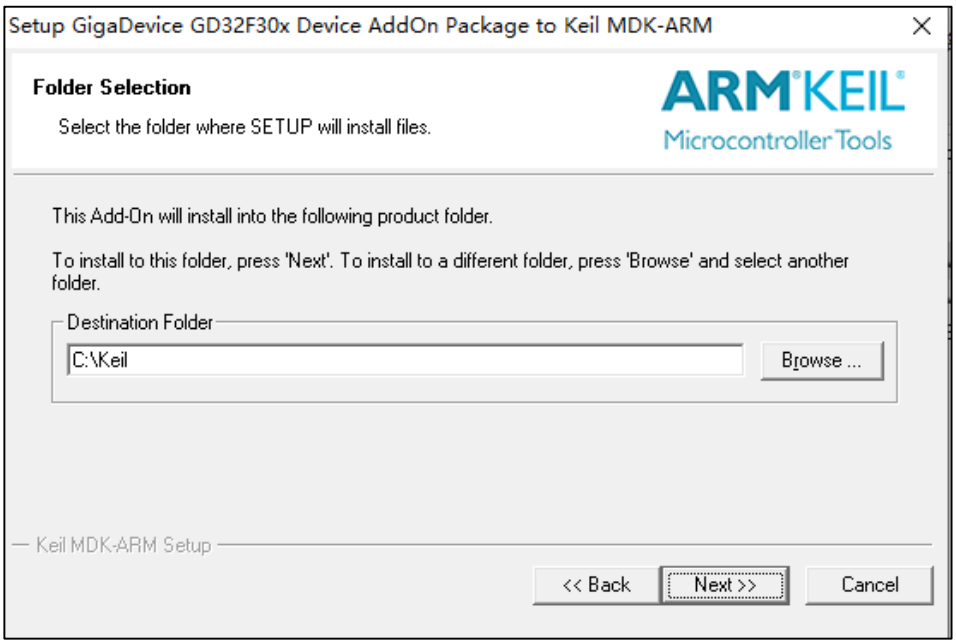
5.1.1. Add GD32F3x0 MCU device in Keil4

1. Download GD32F3x0 series plug-ins from gd32mcu website.

Figure 5-1. GD32F3x0 plug-in package details

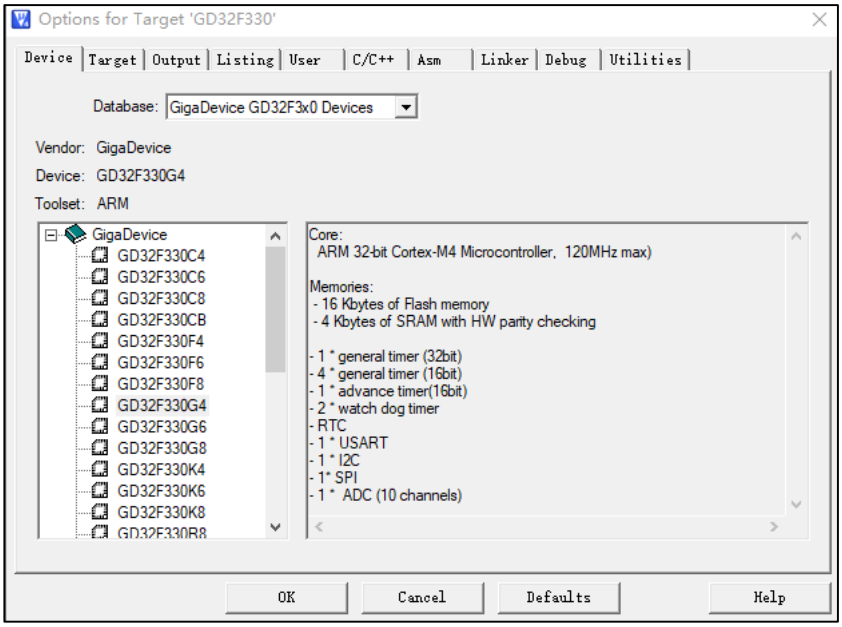
2. Double click the installation file to install the plug-in to the directory of Keil4. Generally, it will be selected by default. If Keil4 and Keil5 are both installed, it needs to be selected manually.

Figure 5-2. Installation diagram of GD32F3x0 Series MCU plug-in package (Keil4)



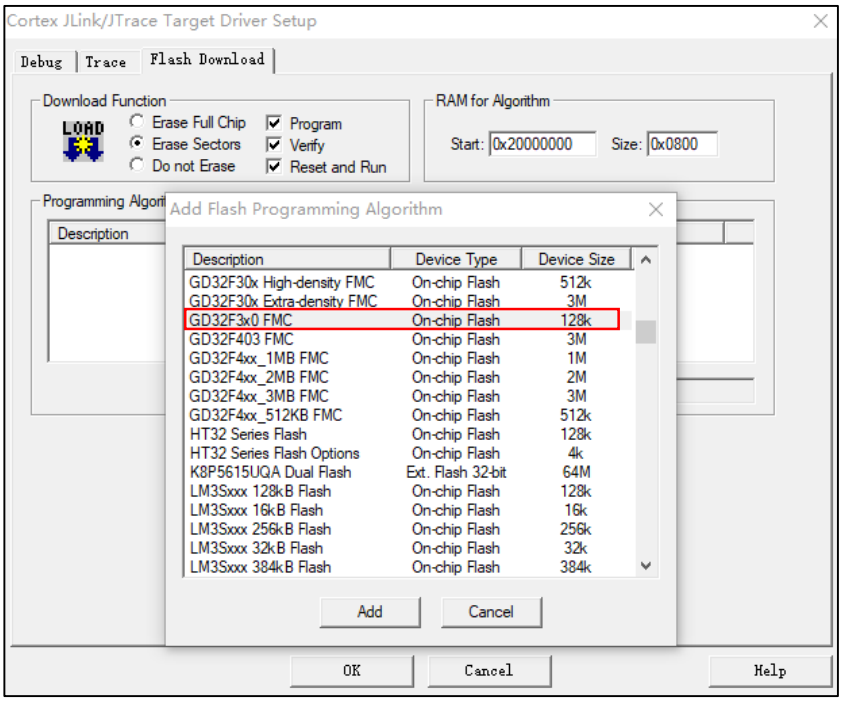
3. After successful installation, reopen Keil4, and the drop-down option of "Database" can appear in "Options for Target ->Device". Click to view the GD32F3x0 part number.

Figure 5-3. Successful installation of GD32F3x0 Series MCU plug-in package (Keil4)



4. For the smooth progress of subsequent debugging, it is recommended to check whether there is a download algorithm under the installation path. You can check it in the following way: open a project, select the device as GD32F3x0, and then "Options for Target -> Debug ->Settings -> Flash Download-> Add". If there is a flash download algorithm of GD32F3x0 in the drop-down option, the installation is successful.

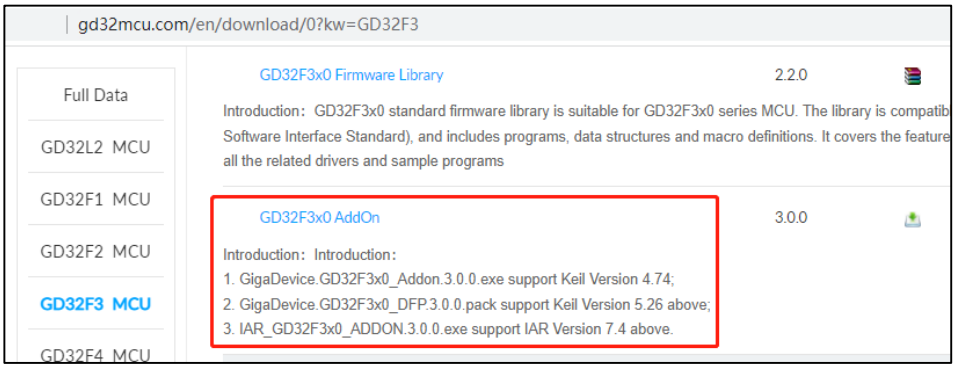
Figure 5-4. GD32F3x0 series flash algorithm file selection diagram (Keil4)



5.1.2. Add GD32F3x0 MCU device in Keil5

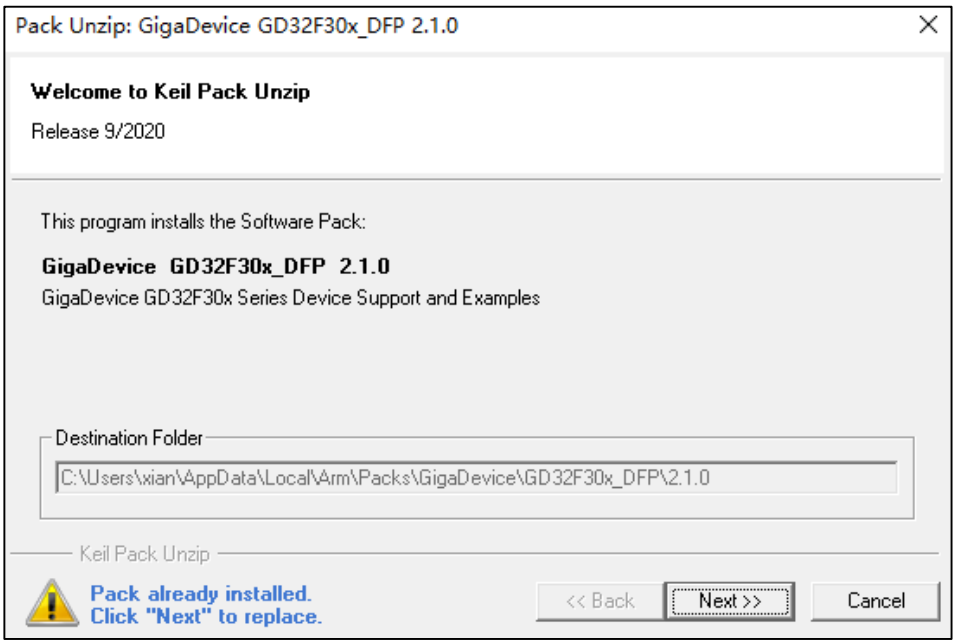
1. Download GD32F3x0 series plug-ins from gd32mcu website.

Figure 5-5. GD32F3x0 plug-in package details



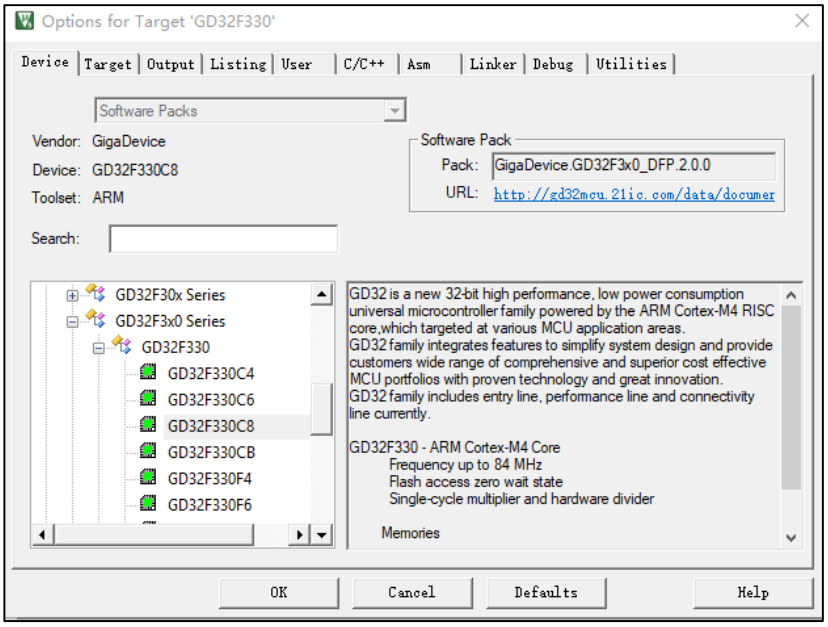
2. Extract and install it into the directory of Keil5.

Figure 5-6. Installation diagram of GD32F3x0 Series MCU plug-in package (Keil5)



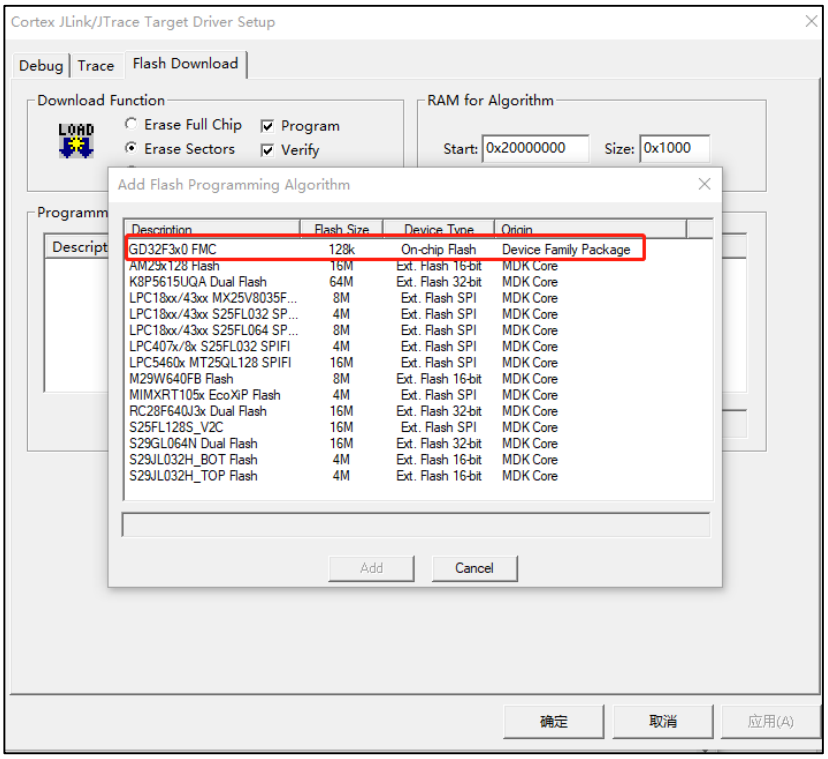
3. After installation, reopen Keil5 project, and you can find GD32F3x0 device in "Options for Target ->Device".

Figure 5-7. Successful installation of GD32F3x0 Series MCU plug-in package (Keil5)



4. Add the flash algorithm in "Options for Target -> Debug ->Settings ->Flash Download", and the algorithm of GD32F3x0 will appear, which indicates that the installation is successful. Debug and download is now availabled.

Figure 5-8. GD32F3x0 series flash algorithm file selection diagram (Keil5)



5. Open Keil5 project file in Keil4 environment

   If Keil5 environment is not installed, Keil4 environment can also be used to compile Keil5 project files. The method is to modify the suffix of the project file, and change the suffix

"xxxx. uvprojx" of Keil5 project file to "xxxx. uvproj", then Keil4 can be used for development.

6. Open Keil4 project file in Keil5 environment

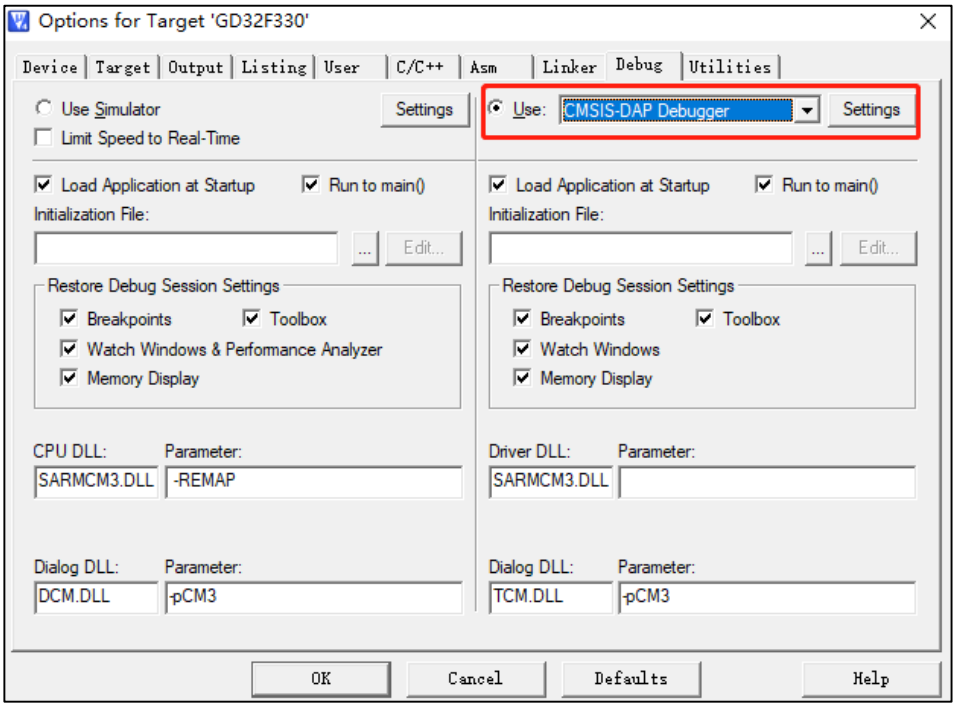
   If you use Keil5 environment to open the Keil4 project file, There will be no MCU devices found. In this case, you can directly modify the suffix of the Keil4 project file "xxxx.uvproj" to "xxxx.uvprojx", then Keil5 can be used for development.

5.2. Debugging and simulating GD32F3x0 with GD-Link

Debugging and simulating GD MCU with GD-Link,The hardware needs to be connected to the development board with GD-Link tool, and the specific configuration of the IDE is as follows.

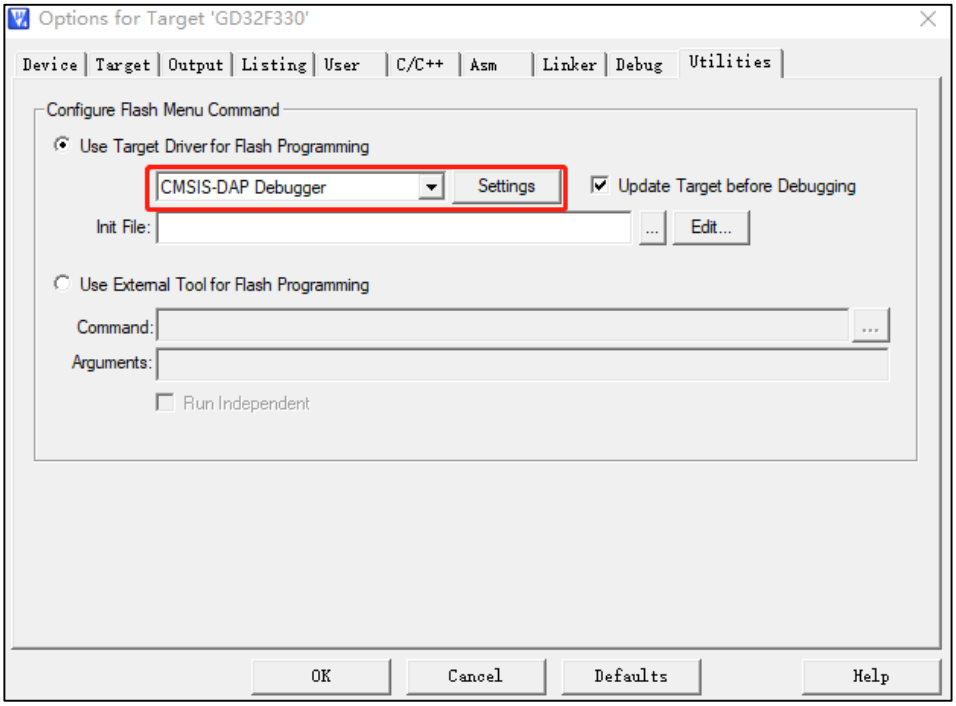
1. Open a GD32F3x0 project and select "CMSIS-DAP Debugger" in "Options for Target -> Debug". Some customers reported that this drive option could not be found because the MDK version is too low and only Keil4 The "CMSIS-DAP Debugger" option is only supported for versions above 4.74 and Keil5.

Figure 5-9. Select the "CMSIS-DAP Debugger" option in the Debug interface (Keil4)



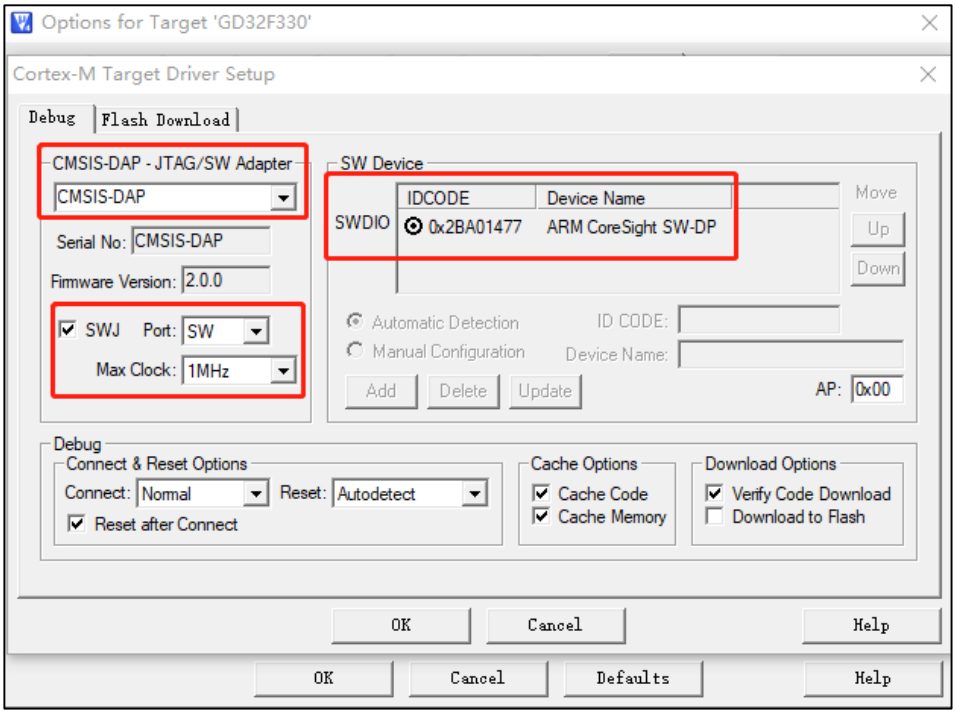
2. In "Options for Target > Utilities", we also have to choose "CMSIS-DAP Debugger" option.

Figure 5-10. Select the "CMSIS-DAP Debugger" option in the Utilities interface (Keil4)



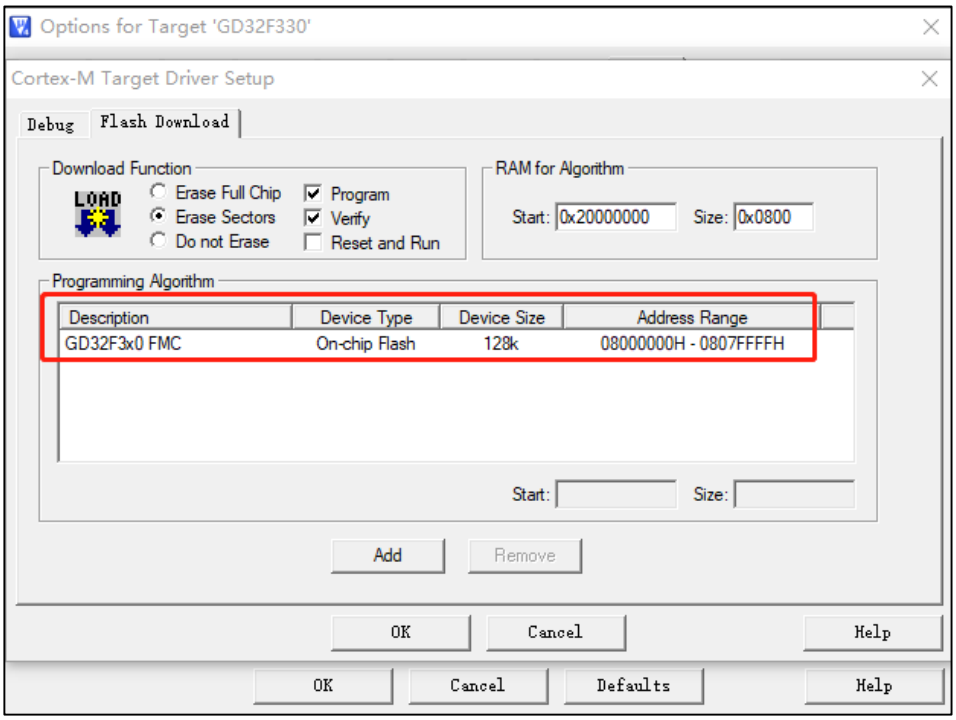
3. In the "Options for Target > Debug ->Settings" check SWJ and Port select SW. "0xXBAXXXXX" will appear in the idcode in the right box, indicates that the target MCU device is successfully connected.

Figure 5-11. GD-Link tool successfully connected to the target MCU(Keil4)



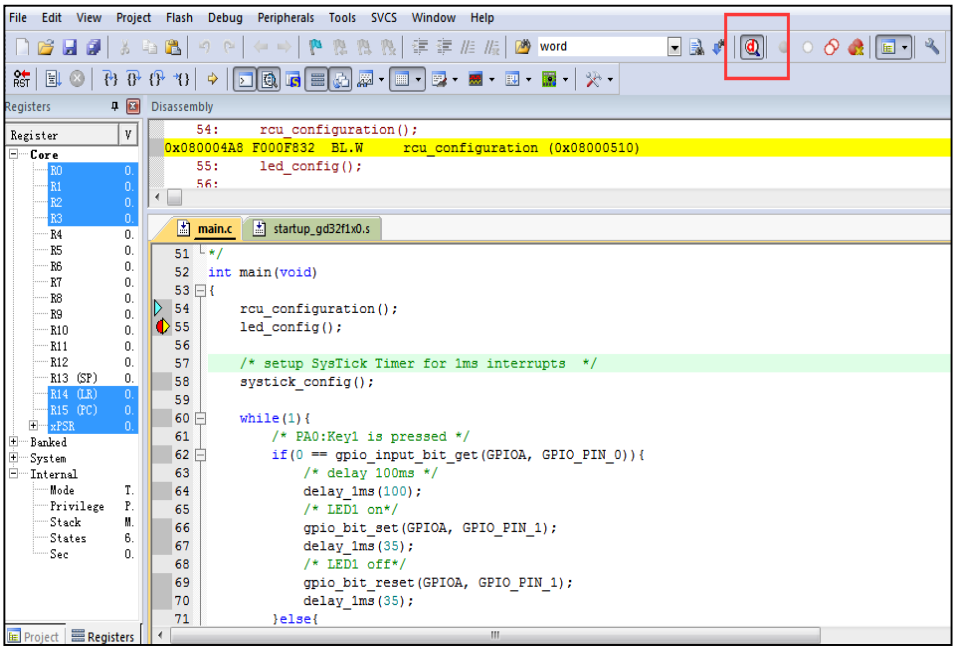
4. Add the flash algorithm of GD32F3x0 in "Options for Target -> Debug ->Settings -> Flash Download".

Figure 5-12. Schematic diagram of adding flash algorithm file(Keil4)



5. Click the shortcut in the red box in ***Figure 5-13. Schematic diagram of GD-Link simulation(Keil4)*** to start debugging, and you can use GD-Link for simulation.

Figure 5-13. Schematic diagram of GD-Link simulation(Keil4)



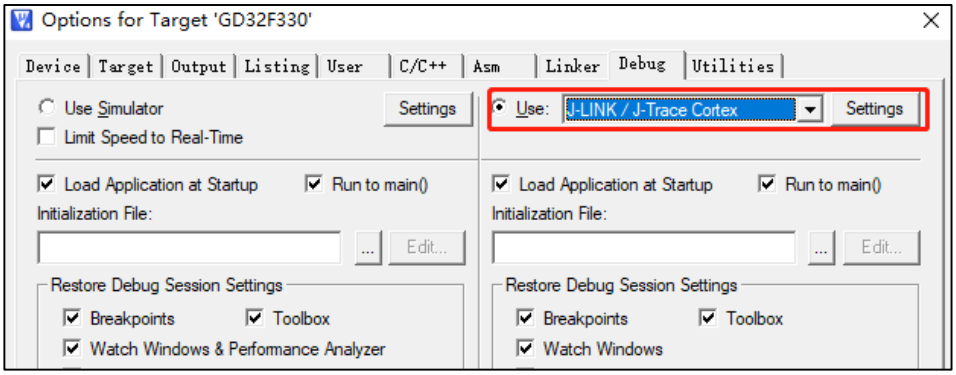
5.3. Debugging and simulating GD32F3x0 with J-Link

Debugging and simulating GD MCU with J-Link, The hardware needs to be connected to the

development board with J-Link tool, and the specific configuration of the IDE is as follows:

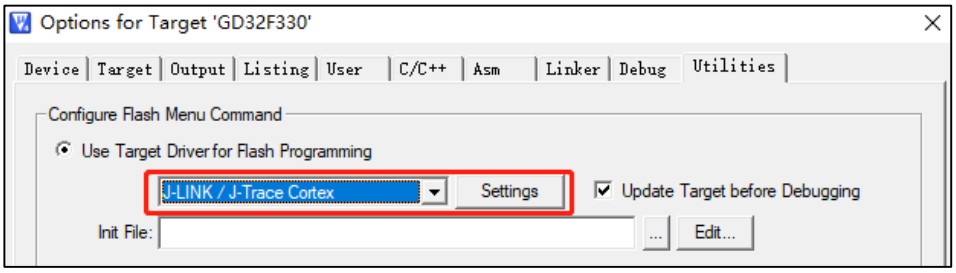
1. Open a GD32F3x0 project file and select "J-LINK/J-Trace Corte " in "Options for Target -> Debug".

Figure 5-14. Select the "J-LINK/J-Trace Cortex" option in the Debug interface (Keil4)



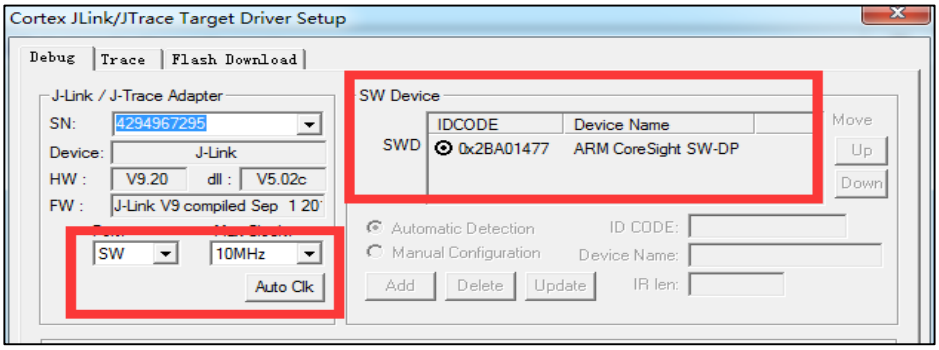
2. In "Options for Target > Utilities", we also have to choose "J-LINK/J-Trace Cortex" option.

Figure 5-15. Select the "J-LINK/J-Trace Cortex" option in the Utilities interface (Keil4)



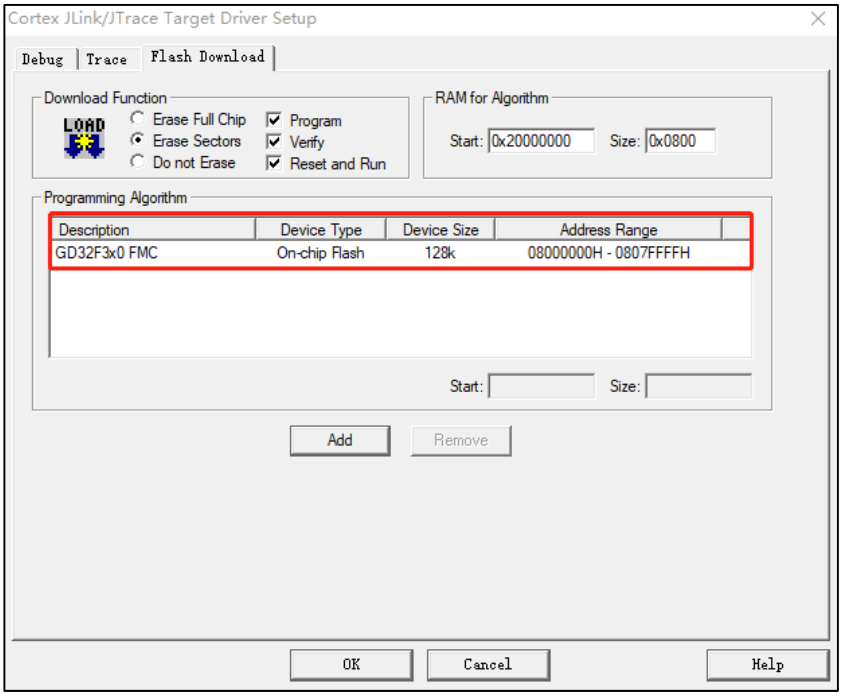
3. In the "Options for Target > Debug ->Settings" Port select SW. "0xXBAXXXXX" will appear in the idcode in the right box, indicates that the target MCU device is successfully connected.

Figure 5-16. J-Link tool successfully connected to the target MCU(Keil4)



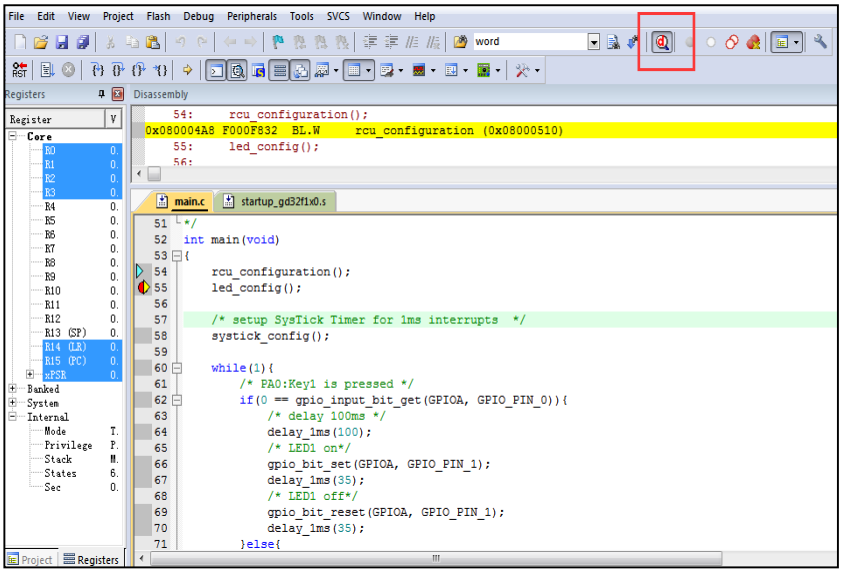
4. Add the flash algorithm of GD32F3x0 in "Options for Target -> Debug ->Settings -> Flash Download".

Figure 5-17. Schematic diagram of adding flash algorithm file(Keil4)



5. Click the shortcut in the red box in ***Figure 5-18. Schematic diagram of J-Link simulation(Keil4)*** to start debugging, and you can use J-Link for simulation.

Figure 5-18. Schematic diagram of J-Link simulation(Keil4)



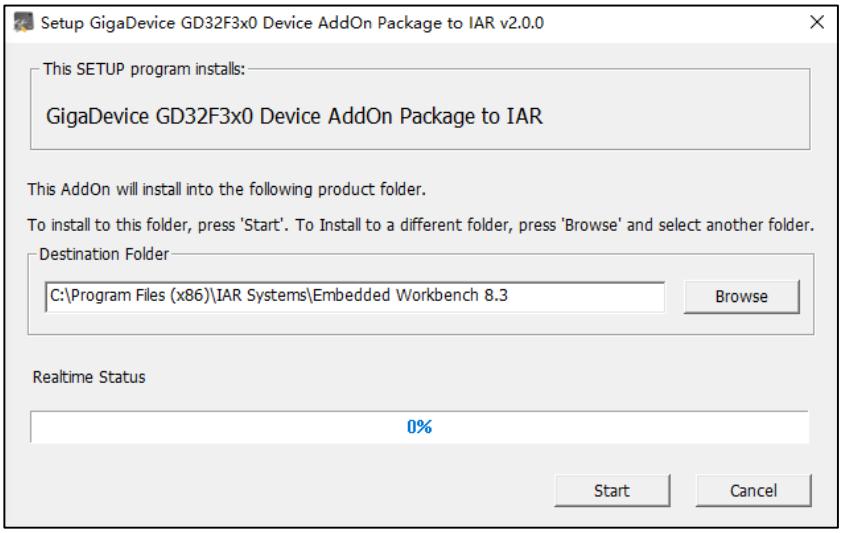
5.4. Using IAR to develop GD32F3x0

There are many versions of IAR, and the compatibility between versions is not good. If you use it for the first time, it is recommended to install versions above 7.3. After installing IAR, add the device of GD according to this document for debugging

5.4.1 Add gd32F3x0 MCU device in IAR

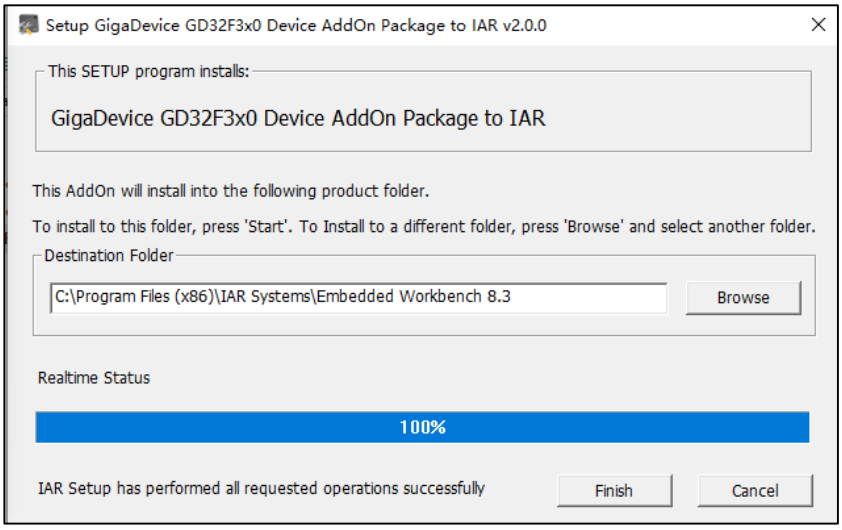
1. Download GD32F3x0 series plug-ins: IAR_GD32F3x0_ADDON_2.0.0.exe.
2. Run IAR_ GD32F3x0_ ADDON_ 2.0.0. exe, click start to start installing the plug-in.

Figure 5-19. Installation diagram of GD32F3x0 Series MCU plug-in package(IAR)



3. After the installation is successful, click Finish to end the plug-in installation.

Figure 5-20. Successful installation of GD32F3x0 Series MCU plug-in package (IAR)



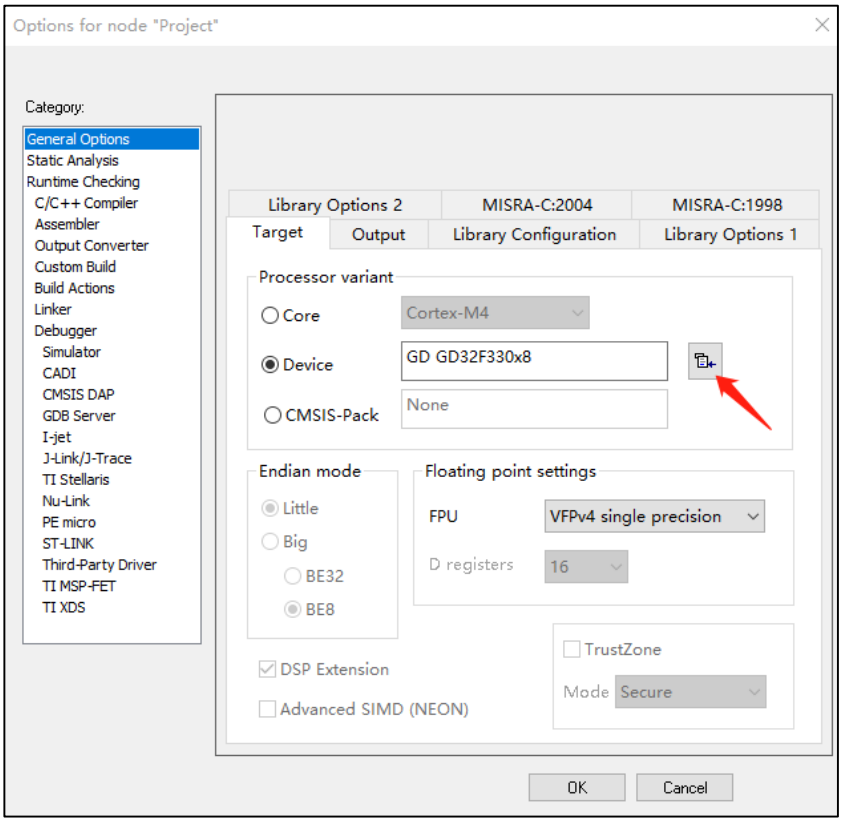
5.4.2 Debugging GD32F3x0 in IAR

In the previous section, we have added the plug-in of GD32F3x0 series. In this section, we will introduce how to use it.

1. There are two ways to use IAR to compile GD MCU. One is to use the existing project for modification, and the other is to re-establish the project. Here, we will not introduced how to establish the project. The project establishment of GD is consistent with that of other

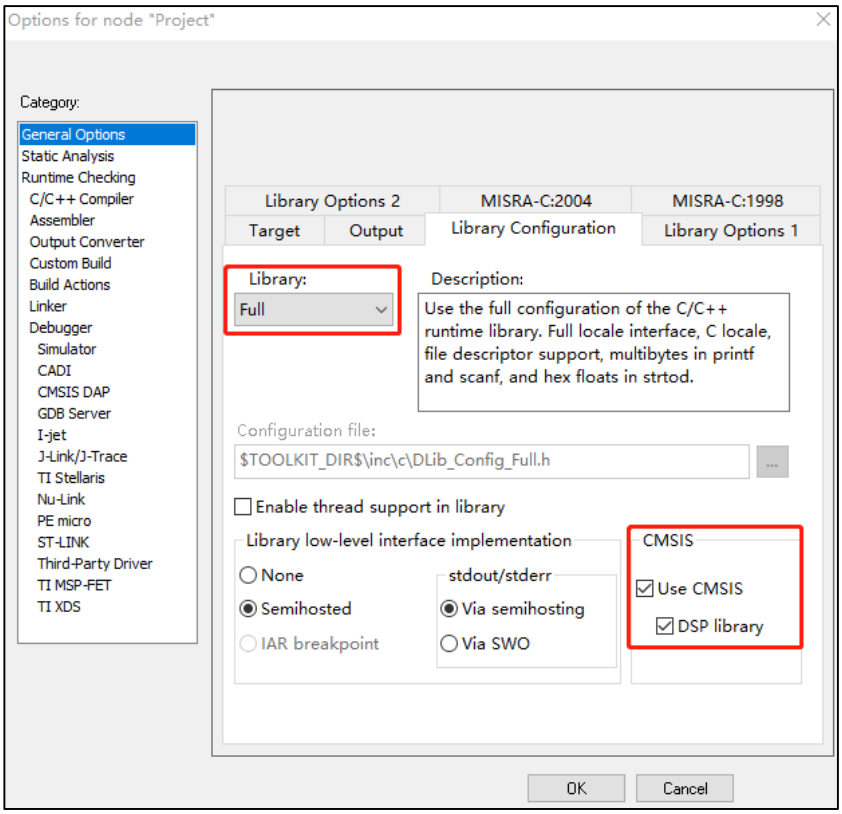
platforms.

Figure 5-21. Select the GD device in the IAR "Options" interface



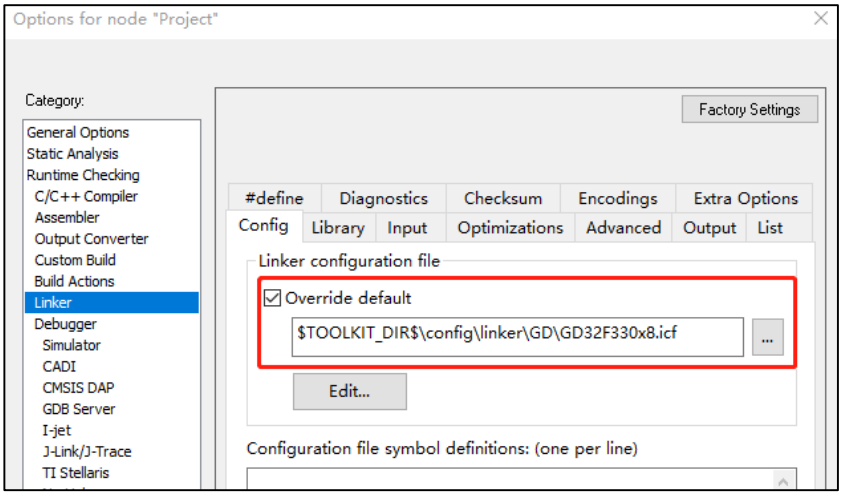
2. IAR after version 6.1 does not need to add CMSIS files (core_cm3.c and core_cm3.h), but you need to check use CMSIS in "General Options->Library Configuration". If the software code uses printf function, you also need to modify the"Library" to"FULL".

Figure 5-22. Add CMSIS file in IAR "Options" interface



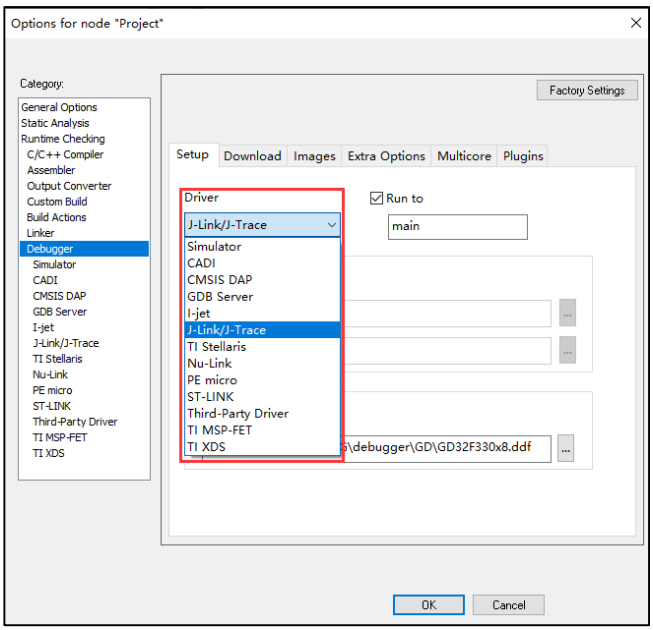
3. The Link file of the chip will be selected by default according to the device when establishing the project, but you should still have the habit of checking before compiling. Check whether the ICF file is configured and correct.

Figure 5-23. Add ICF file in IAR "Options" interface



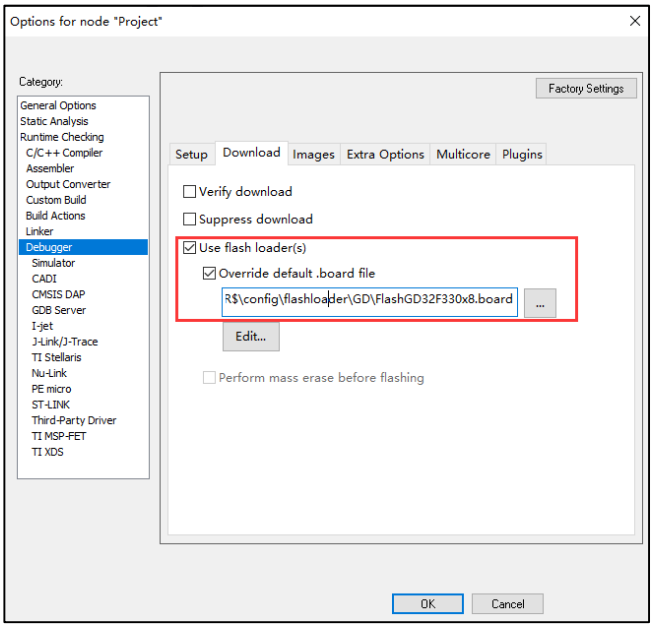
4. Configure the "Debugger->Setup" option. The newly created project is simulator option by default. If debugging is required, you need to choose according to the actual situation: use GD-Link to select CMSIS DAP (poor compatibility, not recommended under IAR) or J-Link to select J-Link/J-Trace.

Figure 5-24. Select the debugger tool in the IAR "Options" interface



5. Configure the "Debugger->Download" option. The new project may not be configured with the download option. If we need to debug the code, we must check the "User flash loader" option and ensure that the "board file" is accurate, otherwise the program cannot be downloaded to the chip normally.

Figure 5-25. Configure flash loader in IAR "Options" interface

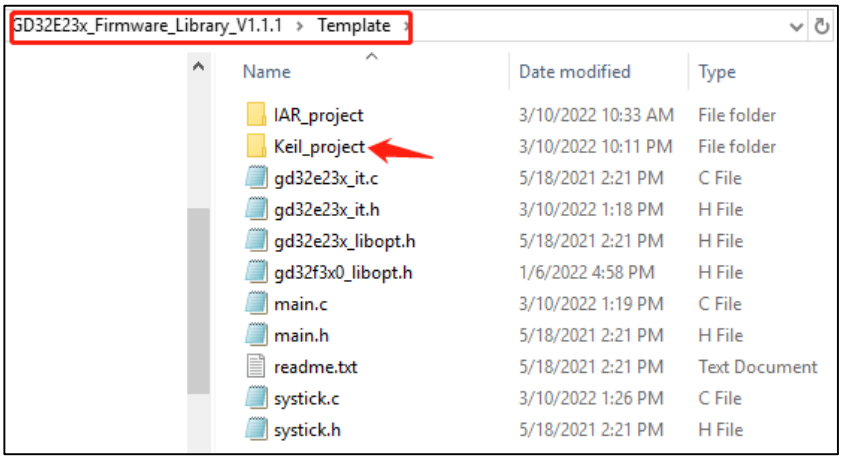


6. Steps of GD32E23x firmware library adapting to GD32F3x0 Series MCU

This chapter will use GD32E23x_Firmware_Library_V1.1.1 take the project in the template as an example to introduce how to adapt GD32F3x0 series MCU.

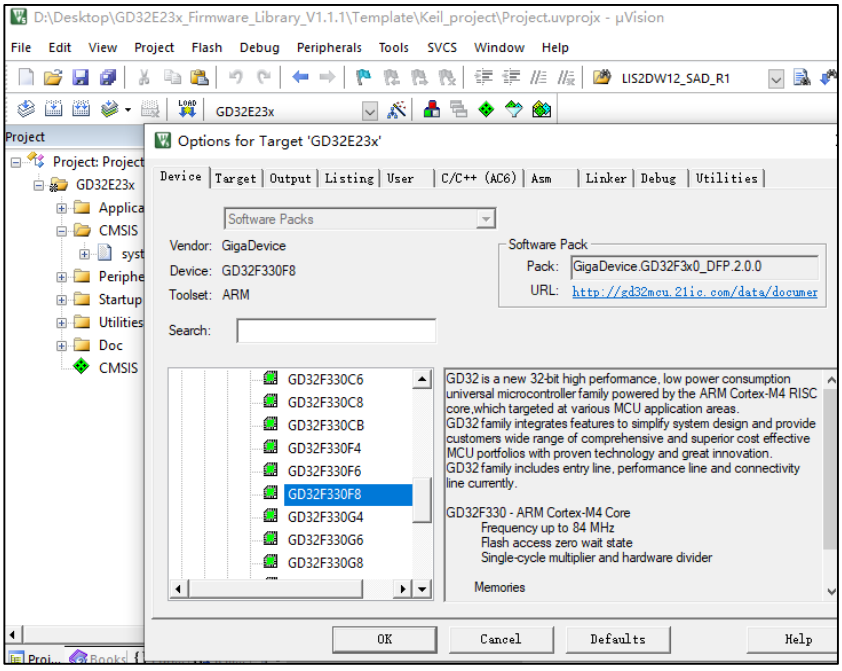
1. Open Keil project.

Figure 6-1. Open GD32E23x Keil project



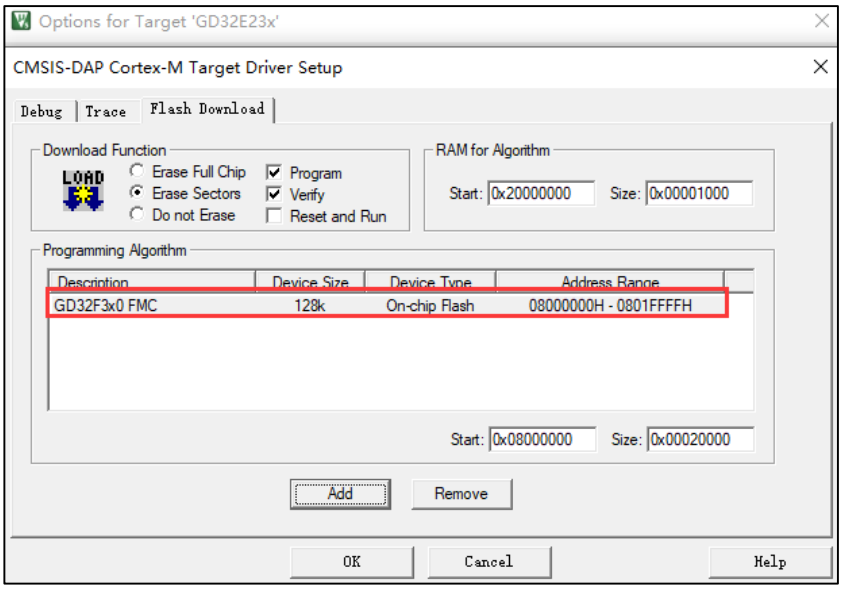
2. After opening the project, "Options for Target -> Device", select GD32F3x0 MCU part number.

Figure 6-2. Select GD32F3x0 device in GD32E23x project



3. Add the flash algorithm of GD32F3x0 in "Options for Target -> Debug ->Settings -> Flash Download".

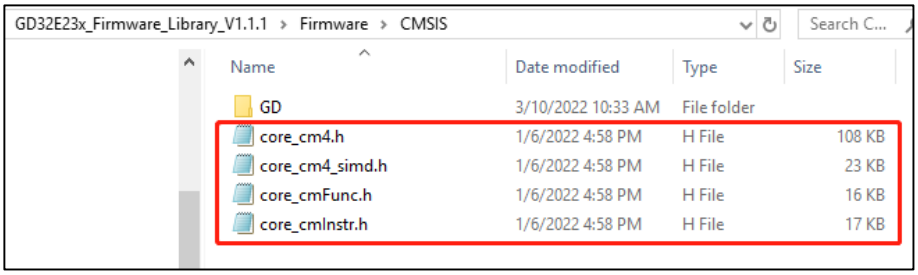
Figure 6-3. Add the flash algorithm of GD32F3x0



4. Copy Cortex M4 kernel files to:

x:\ GD32E23x_Firmware_Library_V1.1.1\Firmware\CMSIS.

Figure 6-4. Add Cortex M4 kernel files to GD32E23x firmware library



5. Modify the contents of the " gd32e23x.h" in GD32E23x firmware library.

Figure 6-5. Modify the contents of "gd32e23x.h"

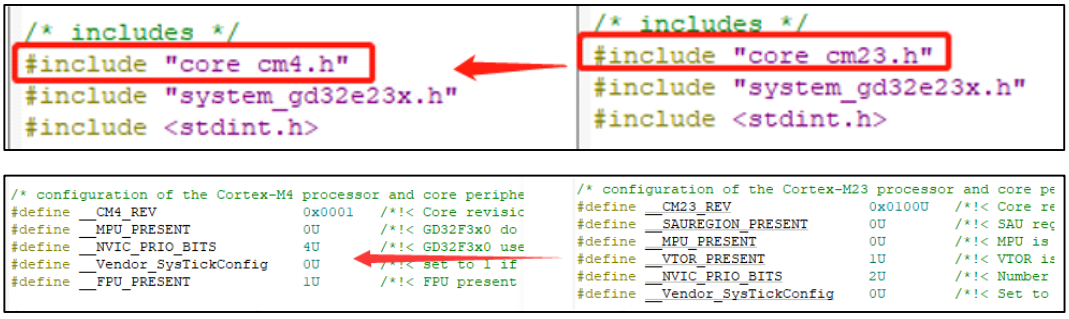


Table 6-1. Modify the contents of "gd32e23x.h"

After modification	Before modification
#include "core_cm23.h"	#include "core_cm23.h"
#define __CM4_REV 0x0001	#define __CM23_REV 0x0100U

#define __MPU_PRESENT 0U	#define __SAUREGION_PRESENT 0U
#define __NVIC_PRIO_BITS 4U	#define __MPU_PRESENT 0U
#define __Vendor_SysTickConfig 0U	#define __VTOR_PRESENT 1U
#define __FPU_PRESENT 1U	#define __NVIC_PRIO_BITS 2U
	#define __Vendor_SysTickConfig 0U

6. GD32E230xx does not support interrupt grouping, so there is no "void nvic_priority_group_set (uint32_t nvic_prigroup)" function in the firmware library. We need to add corresponding content in the firmware library.

Table 6-2. Modify the contents of "gd32e23x_misc.h"

```
 /* priority group - define the pre-emption priority and the subpriority */
#define NVIC_PRIGROUP_PRE0_SUB4        ((uint32_t)0x00000700)
#define NVIC_PRIGROUP_PRE1_SUB3        ((uint32_t)0x00000600)
#define NVIC_PRIGROUP_PRE2_SUB2        ((uint32_t)0x00000500)
#define NVIC_PRIGROUP_PRE3_SUB1        ((uint32_t)0x00000400)
#define NVIC_PRIGROUP_PRE4_SUB0        ((uint32_t)0x00000300)

 /* set the priority group */
void nvic_priority_group_set(uint32_t nvic_prigroup);
```

Table 6-3. Modify the contents of "gd32e23x_misc.c"

```
void nvic_priority_group_set(uint32_t nvic_prigroup)
{
    /* set the priority group value */
    SCB->AIRCR = NVIC_AIRCR_VECTKEY_MASK | nvic_prigroup;
}
```

7. GD32E230xx only supports level 4 priority, not sub priority. GD32F3x0 supports both priority and sub priority. The corresponding contents need to be modified in the firmware library.

Table 6-4. Modify the contents of "gd32e23x_misc.h"

```
 /* enable NVIC request */
void nvic_irq_enable(uint8_t nvic_irq, uint8_t nvic_irq_pre_priority, uint8_t
nvic_irq_sub_priority);
```

Table 6-5. Modify the contents of "gd32e23x_misc.c"

```
void nvic_irq_enable(uint8_t nvic_irq,
                     uint8_t nvic_irq_pre_priority,
                     uint8_t nvic_irq_sub_priority)
{
    uint32_t temp_priority = 0x00U, temp_pre = 0x00U, temp_sub = 0x00U;
    /* use the priority group value to get the temp_pre and the temp_sub */
    switch ((SCB->AIRCR) & (uint32_t)0x700U) {
    case NVIC_PRIGROUP_PRE0_SUB4:
```

```
        temp_pre = 0U;
        temp_sub = 0x4U;
        break;
    case NVIC_PRIGROUP_PRE1_SUB3:
        temp_pre = 1U;
        temp_sub = 0x3U;
        break;
    case NVIC_PRIGROUP_PRE2_SUB2:
        temp_pre = 2U;
        temp_sub = 0x2U;
        break;
    case NVIC_PRIGROUP_PRE3_SUB1:
        temp_pre = 3U;
        temp_sub = 0x1U;
        break;
    case NVIC_PRIGROUP_PRE4_SUB0:
        temp_pre = 4U;
        temp_sub = 0x0U;
        break;
    default:
        nvic_priority_group_set(NVIC_PRIGROUP_PRE2_SUB2);
        temp_pre = 2U;
        temp_sub = 0x2U;
        break;
    }
    /* get the temp_priority to fill the NVIC->IP register */
    temp_priority = (uint32_t)nvic_irq_pre_priority << (0x4U - temp_pre);
    temp_priority |= nvic_irq_sub_priority &(0x0FU >> (0x4U - temp_sub));
    temp_priority = temp_priority << 0x04U;
    NVIC->IP[nvic_irq] = (uint8_t)temp_priority;
    /* enable the selected IRQ */
    NVIC->ISER[nvic_irq >> 0x05U] = (uint32_t)0x01U << (nvic_irq & (uint8_t)0x1FU);
}
```

8. The flash of GD32F3x0 is zero waiting. GD32E230xx series needs to configure the waiting cycle, so the function of waiting cycle can be removed

Table 6-6. Remove the function of waiting period in GD32E23x project

```
~~FMC_WS = (FMC_WS & (~FMC_WS_WSCNT)) | WS_WSCNT_2;~~
```

9. The flash of GD32E230xx supports 32-bit and 64-bit programming, and the flash of GD32F3x0 supports 32-bit word and half word programming. If 64-bit programming is used in the application code, it needs to be modified to 32-bit word or half word programming, and half word programming needs to be added to the GD32E230xx firmware library

Table 6-7. Add half word programming to "gd32e23x_fmc.h" of GD32E23x project

```
/* FMC program a half word at the corresponding address */
fmc_state_enum fmc_halfword_program(uint32_t address, uint16_t data);
```

Table 6-8. Add half word programming to "gd32e23x_fmc.c" of GD32E23x project

```
fmc_state_enum fmc_halfword_program(uint32_t address, uint16_t data)
{
    fmc_state_enum fmc_state = fmc_ready_wait(FMC_TIMEOUT_COUNT);

    if(FMC_READY == fmc_state){
        /* set the PG bit to start program */
        FMC_CTL |= FMC_CTL_PG;
        REG16(address) = data;
        /* wait for the FMC ready */
        fmc_state = fmc_ready_wait(FMC_TIMEOUT_COUNT);
        /* reset the PG bit */
        FMC_CTL &= ~FMC_CTL_PG;
    }
    /* return the FMC state */
    return fmc_state;
}
```

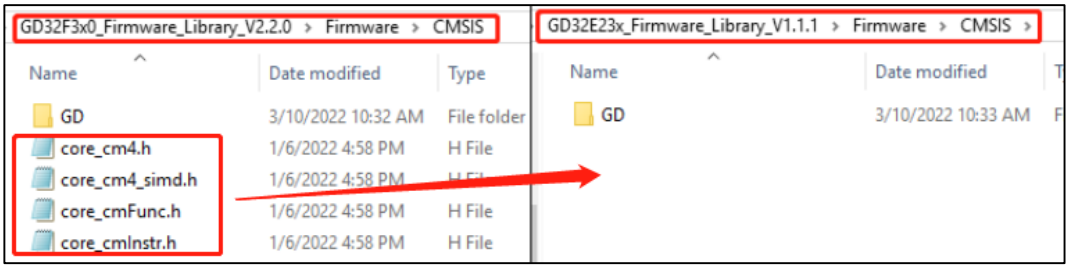
10. If TIMER5 is used in the project, because GD32F3x0 remove this TIMER5(Except GD32F350xx), the code of TIMER5 needs to be changed to other timer.
11. Compile GD32E23x project, so far, you can use the modified GD32E23x firmware library for software development in GD32F3x0 series MCU.

7. Steps to replace GD32E23x project library with GD32F3x0 Library

This chapter will use the projects in "GD32E23x_Firmware_Library_V1.1.1\Template" and "GD32F3x0_Firmware_Library_V2.2.0\Template" as examples.

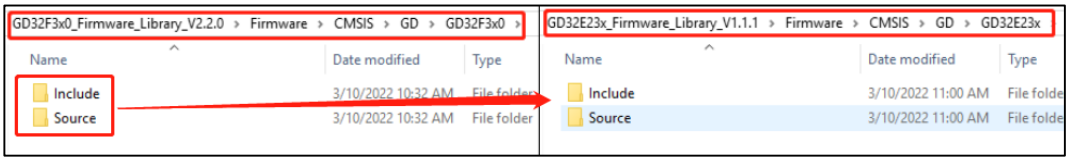
1. Copy the files in "GD32F3x0_Firmware_Library_V2.2.0\Firmware\CMSIS" to the "GD32E23x_Firmware_Library_V1.1.1\Firmware\CMSIS" folder.

Figure 7-1. Copy h file in CMSIS of GD32F3x0 firmware library to GD32E23x



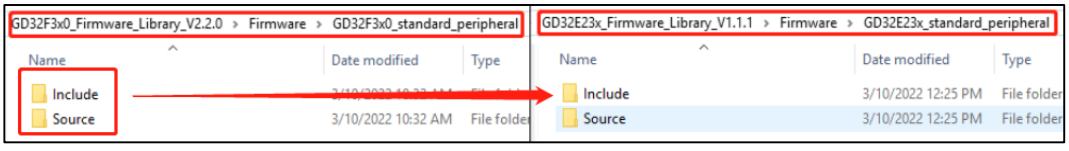
2. Copy the Iclude and Source folders in "GD32F3x0_Firmware_Library_V2.2.0\Firmware\CMSIS\GD\GD32F3x0" and replace them to the "GD32E23x_Firmware_Library_V1.1.1\Firmware\CMSIS\GD\GD32E23x" folder.

Figure 7-2. Copy and replace the Include and Source files in CMSIS under GD32F3x0 firmware library into GD32E23x firmware library



3. Copy the Iclude and Source folders in "GD32F3x0_Firmware_Library_V2.2.0\Firmware\GD32F3x0_standard_peripheral" and replace them to the "GD32E23x_Firmware_Library_V1.1.1\Firmware\GD32E23x_standard_peripheral" folder.

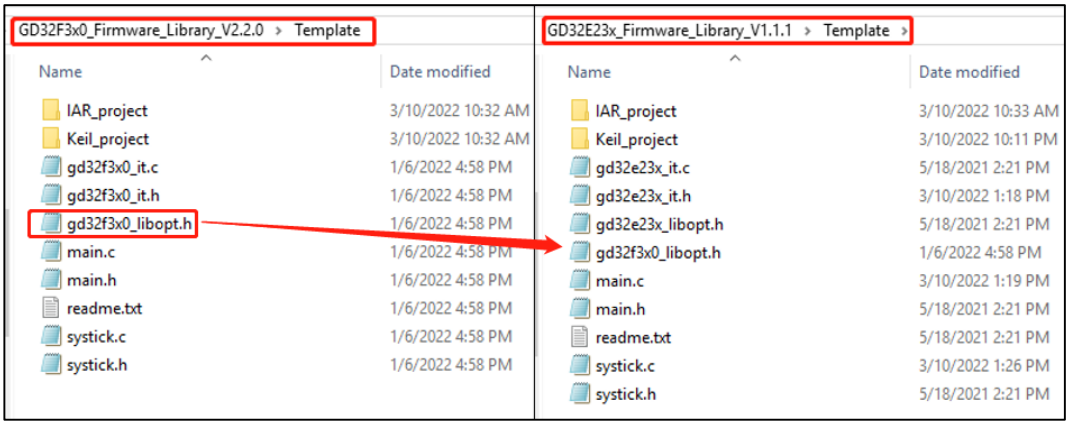
Figure 7-3. Copy and replace the Include and Source files in standard_peripheral under GD32F3x0 firmware library into GD32E23x firmware library



4. Copy the "gd32f3x0_libopt.h" file in "GD32F3x0_Firmware_Library_V2.2.0\Template" into "GD32E23x_Firmware_Library_V1.1.1\Template".

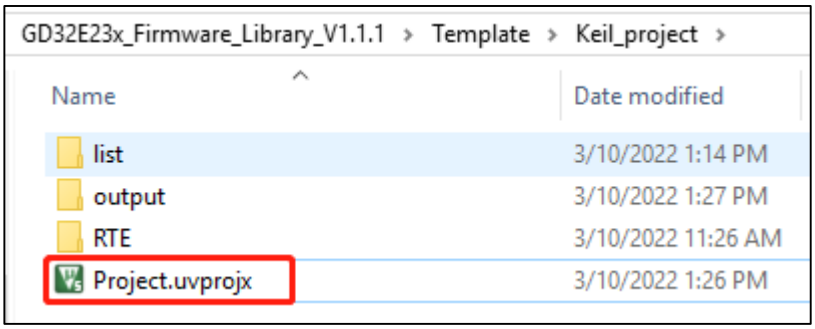
Figure 7-4. Copy the "gd32f3x0_libopt.h" file in GD32F3x0 firmware library into

GD32E23x firmware library



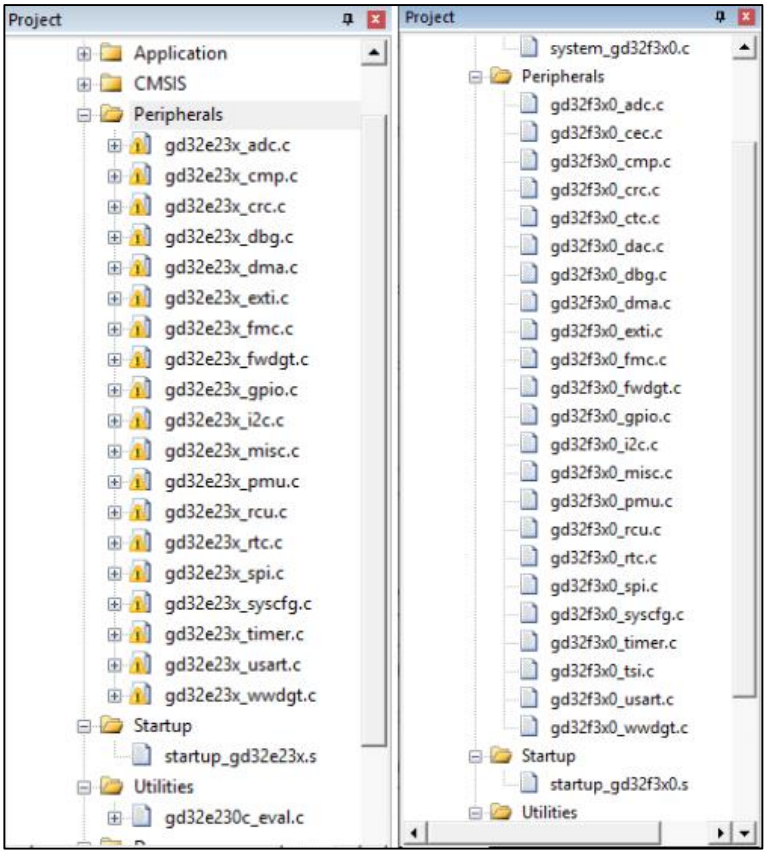
5. Open the Keil project under the template file in the GD32E23x firmware library.

Figure 7-5. Open the Keil project under the template file in the GD32E23x firmware library



6. A yellow triangle mark on the left side of the engineering interface indicates that the original file no longer exists because the old file has been replaced in the previous file replacement steps. At this time, you only need to remove all the files marked in yellow. Among them, "gd32e230c_eval. c" is the supporting configuration of the development board. If it is not used in the actual project, it can be transplanted, and then add the corresponding GD32F3x0 files.

Figure 7-6. Remove the Yellow marked files and add new files



7. Modify the "#include "gd32e23x.h" "statement contained in the" main. c" and "systick. c" files in the project to "#include "gd32f3x0.h"" statement, and delete the "#include"gd32e230c_eval. h"" statement. Then reselect the MCU device and flash algorithm.

Figure 7-7. Modify the contents of "main.c", "systick.c" files

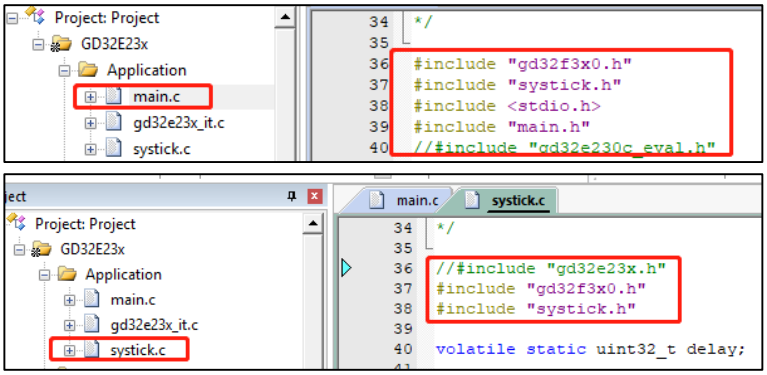


Figure 7-8. Reselect GD32F3x0 MCU device

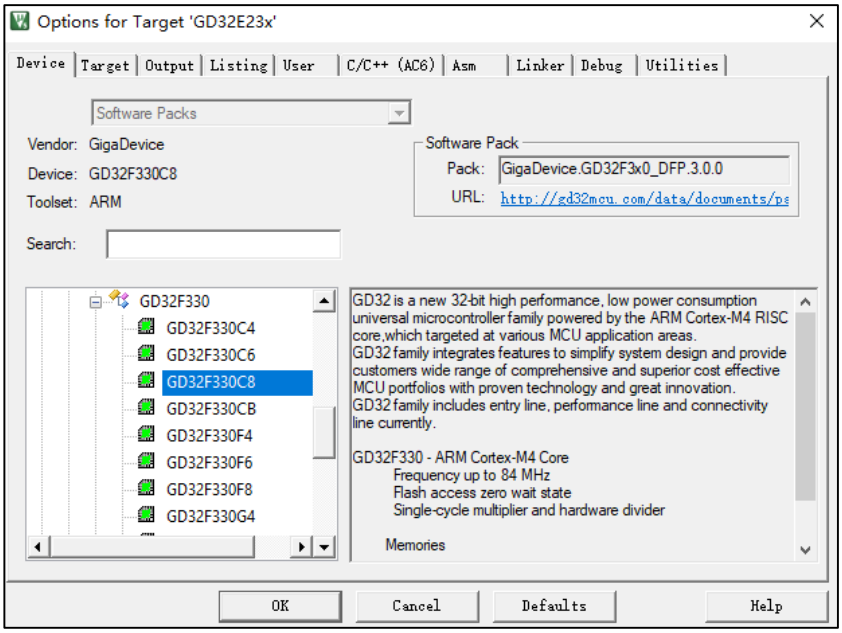
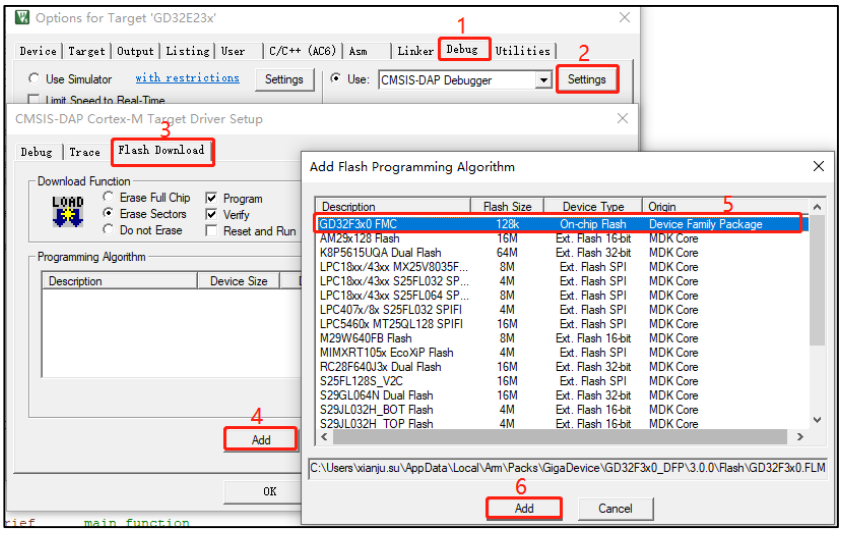


Figure 7-9. Reselect GD32F3x0 Flash algorithm



8. Since GD32E230xx does not support the bit length of the configuration priority group, after transplanting the GD32F3x0 library, when there is a configuration of using interrupt in the application code, the application code needs to add the "void nvic_priority_group_set (uint32_t nvic_prigroup)" function.

Table 7-1. nvic_priority_group_set function

```
/* set the priority group */
void nvic_priority_group_set(uint32_t nvic_prigroup);
```

Moreover, GD32E230xx only supports level 4 preemption priority and does not support sub priority. Therefore, after transplantation, the interrupt enabling function needs to be changed to the function shown in ***Table 7-2. nvic_irq_enable function***.

Table 7-2. nvic_irq_enable function

```
/* set the priority group */
void nvic_irq_enable(uint8_t nvic_irq, uint8_t nvic_irq_pre_priority, uint8_t nvic_irq_sub_priority);
```

9. If TIMER5 is used in the project, because GD32F3x0 remove this TIMER5(Except GD32F350xx), the code of TIMER5 needs to be changed to other timer.
10. Compile the project. If there is an error, modify it according to the prompt. Usually, the prompt is that "#include "gd32e23x.h"" in the code is not modified to "#include "gd32f3x0.h"", and modify it according to the prompt. So far, the project has been transplanted successfully, and the development of GD32F3x0 Series MCU can be carried out.

8. Revision history

Table 8-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Mar.15 2022

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.