

GigaDevice Semiconductor Inc.

GD32W51x

Arm[®] Cortex[®]-M33 32-bit MCU

快速开发指南

1.1 版本

(2022 年 3 月)

目录

目录.....	1
图索引.....	3
表索引.....	5
1. 认识开发板.....	6
1.1. 开发板实物图.....	6
1.1.1. EVAL 开发板.....	6
1.1.2. START 开发板.....	7
1.2. 启动模式.....	7
1.3. 调试器接口.....	8
1.4. 查看日志.....	8
2. 搭建开发环境.....	10
2.1. Keil MDK5 安装.....	10
2.2. IAR 安装.....	11
2.3. CMAKE+GCC 安装.....	11
2.3.1. JLINK 驱动安装.....	12
3. 开发需知.....	13
3.1. SDK 执行程序组.....	13
3.2. SDK 配置.....	14
3.2.1. Platform 配置.....	14
3.2.2. SRAM 布局.....	15
3.2.3. FLASH 布局.....	15
3.2.4. 固件版本号.....	16
3.2.5. APP 配置.....	16
3.3. 固件生成和下载.....	16
3.4. 正确日志示例.....	16
4. KEIL 工程.....	17
4.1. 工程组.....	17
4.2. 工程配置.....	18
4.2.1. Device 选择.....	18
4.2.2. 调试器配置.....	19
4.3. 编译.....	21
4.4. 下载.....	23

4.5.	调试.....	23
5.	IAR 工程.....	25
5.1.	工程组.....	25
5.2.	工程配置.....	25
5.2.1.	Device 选择.....	26
5.2.2.	调试器配置.....	26
5.3.	编译.....	27
5.4.	下载.....	30
5.5.	调试.....	30
6.	GCC 工程.....	33
6.1.	工程组.....	33
6.2.	编译.....	33
6.3.	下载.....	33
6.4.	调试.....	34
6.4.1.	启动 GDB server.....	34
6.4.2.	启动 GDB Client 进行调试.....	35
7.	常见问题.....	36
7.1.	DAPLINK 盘识别.....	36
7.2.	KEIL JLINK 调试 CM33.....	37
7.3.	DAP 芯片固件升级.....	38
7.4.	No image 错误.....	39
7.5.	代码跑在 SRAM.....	39
8.	版本历史.....	41

图索引

图 1-1. EVAL 开发板实物图.....	6
图 1-2. START 开发板实物图.....	7
图 1-3. J-LINK 引脚.....	8
图 1-4. 串口配置.....	9
图 2-1. 安装 KEIL GD32W51x PACK.....	10
图 2-2. 安装 IAR GD32W51x Addon.....	11
图 2-3. 下载 JLINK 驱动.....	12
图 2-4. 安装 JLINK 驱动.....	13
图 3-1. 启动过程.....	13
图 3-2. Platform 配置.....	14
图 3-3. SRAM 布局.....	15
图 3-4. FLASH 布局.....	15
图 3-5. 固件版本号.....	16
图 3-6. 设备和驱动器列表.....	16
图 3-7. 工程启动信息.....	17
图 4-1. KEIL 工程组界面.....	18
图 4-2. KEIL 选项 Device 页面.....	19
图 4-3. KEIL 选项 Debug 页面.....	19
图 4-5. KEIL 选项 JLINK Settings Debug 页面.....	20
图 4-6. KEIL 选项 Flash Download 页面.....	21
图 4-7. 切换当前工程.....	22
图 4-8. 批量编译.....	22
图 4-9. 批量编译设置.....	23
图 4-9. KEIL 烧录结果.....	23
图 4-10. 调试初始化文件.....	24
图 4-11. KEIL 调试界面.....	24

图 5-1. IAR 工程组	25
图 5-2. IAR 选项 Target 页面	26
图 5-3. IAR 选项 Debugger 页面	27
图 5-4. IAR 编译	28
图 5-5. IAR 批量编译设置	29
图 5-6. IAR Build Actions	30
图 5-7. IAR 烧录结果	30
图 5-8. IAR 调试按钮	31
图 5-9. IAR 开始调试	31
图 5-10. IAR 多镜像调试	32
图 6-1. 创建 HOME 环境变量	34
图 6-2. 启动 JLINK GDB Server	35
图 6-3. GDB 调试窗口	36
图 6-4. 常用命令	36
图 7-1. Mbed 串口驱动安装	37
图 7-2. 设备管理器串口列表	37
图 7-3. KEILDEBUG 选项	38
图 7-4. 添加 KEIL 支持的 TOOL	38
图 7-5. MAINTENANCE 盘	39
图 7-6. NSPE MAP 文件	39

表索引

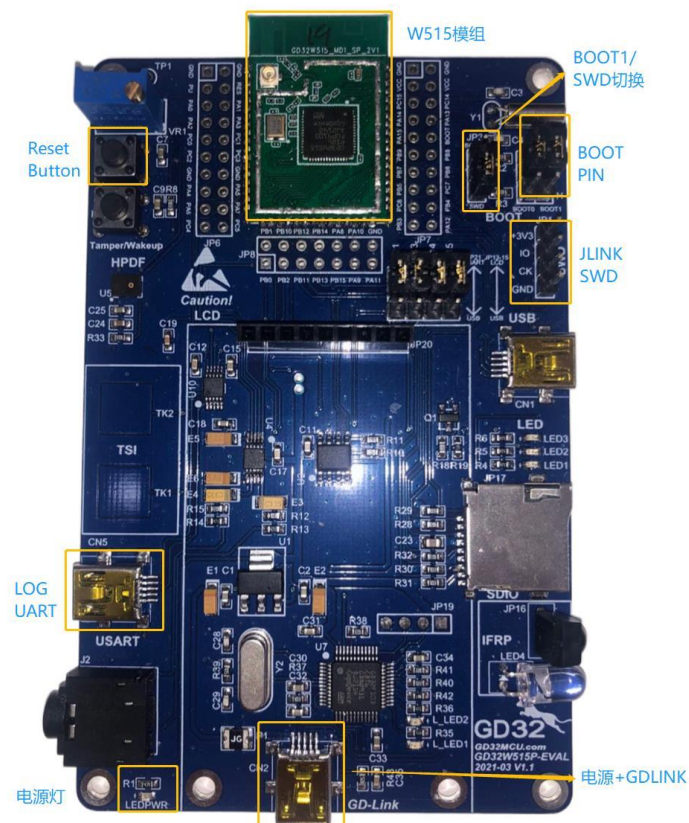
表 1-1. 启动模式.....	8
表 8-1. 版本历史.....	41

1. 认识开发板

1.1. 开发板实物图

1.1.1. EVAL开发板

图 1-1. EVAL 开发板实物图



EVAL 开发板由底板和模组组成，模组搭载了 GD32W51x WiFi 芯片，底板提供了众多外设测试口，例如：SDIO, I2S, IFRP, TSI 等等。图中 GD32W51x 芯片内部通过 SIP (System In a Package) 的方式封装集成了 FLASH。也可以选择外挂 FLASH，模组尺寸不变。

对于开发者来说，可能主要关注开发板以下几个部分，都已经在图中标注出来。

- 启动模式 (Boot PIN);
- 供电口 (电源);
- 查看日志 (LOG UART);
- 调试器接口 (GDLink 或者 JLINK);
- 重启 (Reset Button)。

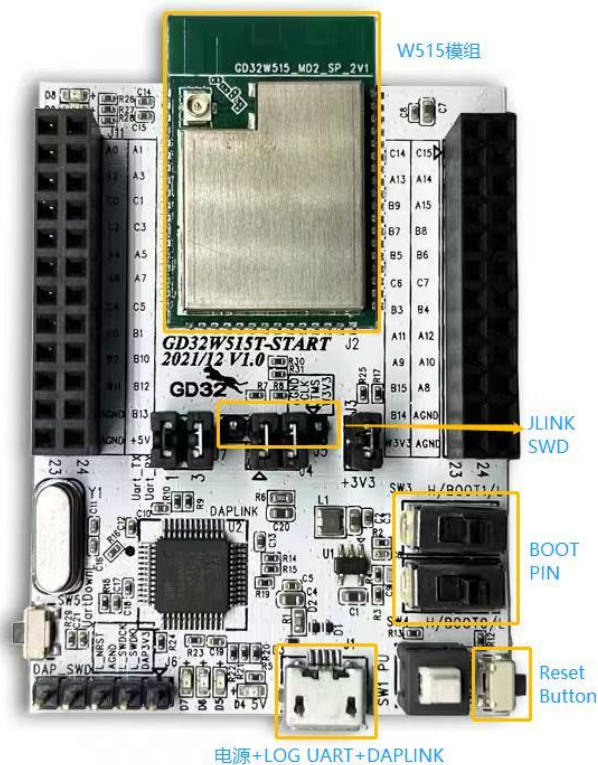
需要注意的是，JLINK SWD CK PIN 跟 BOOT1 PIN 脚复用，当想要切换启动模式时，需要将“BOOT1/SWD 切换”的跳帽切到上方 BOOT1 那一头。如果需要 Jlink 调试，则需要将跳帽

切到下方 SWD 那一头。

1.1.2. START开发板

START 开发板由底板和模组组成，模组搭载了 GD32W51x WiFi 芯片。GD32W51x WiFi 芯片有四款，可以选择内置 SIP FLASH 或者外挂 FLASH，可以选择 36 PIN 封装或者 56 PIN 封装。

图 1-2. START 开发板实物图



对于开发者来说，可能主要关注开发板的以下几个部分，都已经在图中标注出来。

- 启动模式（Boot PIN）；
- 供电口（电源）；
- 查看日志（LOG UART）；
- 调试器接口（DAPLINK 或者 JLINK）；
- 重启（Reset Button）。

1.2. 启动模式

GD32W51x 可以选择从 ROM 启动，FLASH 启动或者 SRAM 启动。

开发板 BOOT PIN 框内的 BOOT0 和 BOOT1 两根引脚的高低选择决定了启动模式，见[表 1-1. 启动模式](#)。更多关于启动模式的说明请参考文档《GD32W51x_User_Manual》。

表 1-1. 启动模式

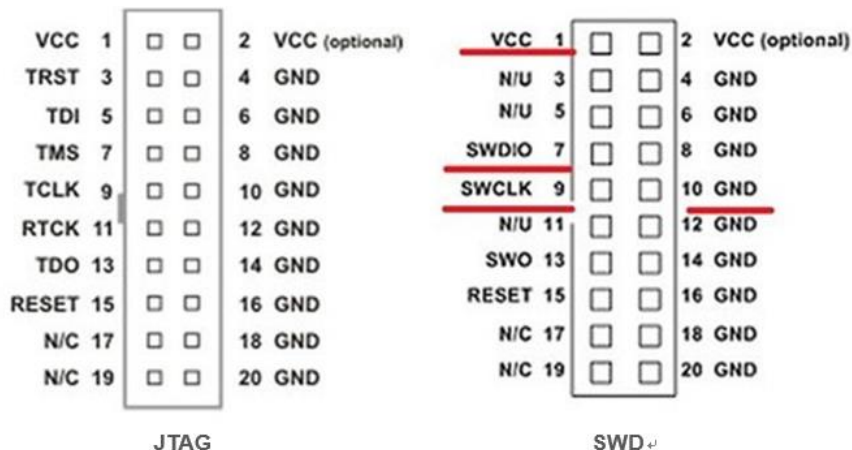
	BOOT1 pin 接高	BOOT1 pin 接低
BOOT0 引脚接高	SRAM	ROM (LegacyBootloader)
BOOT0 引脚接低	FLASH	

1.3. 调试器接口

对于 EVAL 开发板，可以选择 GD-LINK 或者 JLINK，开发板自带 GD-LINK 调试器，只需要接上电源口 Mini USB，就可以使用了。支持 JLINK SWD 模式，引出四根 PIN 分别为+3V3、IO、CK、GND，对应 JLINK 仿真器引脚 VCC、SWDIO、SWCLK、GND，参见[图 1-3. J-LINK 引脚](#)。

对于 START 开发板，可以选择 DAPLINK 或者 JLINK，开发板自带 DAPLINK 调试器（GD32F303），只需要接上电源口 Micro USB，就能使用，DAP 芯片还集成了 UART 功能，所以只需要一根 USB 线，就可以完成供电、调试和查看日志。JLINK SWD 引出了四根 PIN 分别为 AGND、JCLK、JTWS 和 W3V3，利用杜邦线将上述四根引脚和 J-LINK 仿真器对应引脚（如[图 1-3. J-LINK 引脚](#)所示）GND、SWCLK、SWDIO 和 VCC 相连，即可通过 J-LINK SWD 模式烧录和调试代码。将引脚 JCLK 和 JTWS 与下侧两引脚通过跳线帽连接，即可通过 DAPLINK 下载和调试代码。[图 1-2. START 开发板实物图](#)中展示的是透过 DAPLINK 调试。

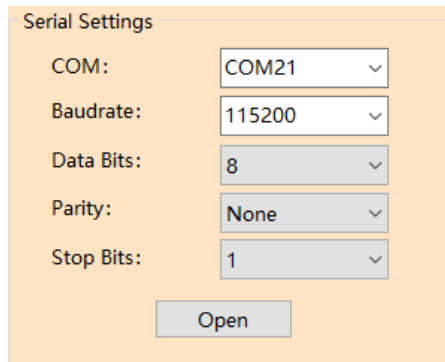
图 1-3. J-LINK 引脚



1.4. 查看日志

使用 MiniUSB 线连接 EVAL 开发板，或者使用 MicroUSB 线连接 START 开发板，PC 端使用串口工具并根据[图 1-4. 串口配置](#)的参数配置并连接，就可以使用串口输出日志了。

图 1-4. 串口配置



Serial Settings

COM:	COM21	▼
Baudrate:	115200	▼
Data Bits:	8	▼
Parity:	None	▼
Stop Bits:	1	▼

Open

2. 搭建开发环境

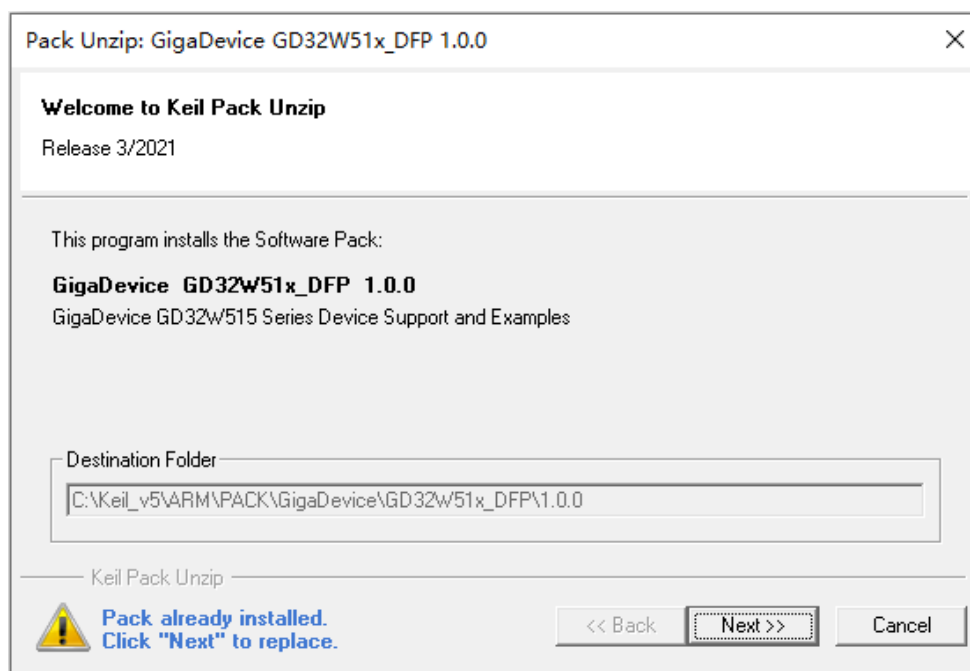
在编译和烧录固件之前，需要搭建开发环境。

目前常用的开发工具主要有三种，分别是 KEIL、IAR 和 GCC。我们 SDK 对三种开发工具都已经支持。三种工具各有所长，可以根据需求来选择。

2.1. Keil MDK5 安装

- 下载
 - 官网地址：<https://www2.keil.com/mdk5>，请选择 MDK525 及以上版本。
 - 如果需要向前兼容 MDK4，还需要下载 MDKCM5xx.EXE，下载地址为：<https://www2.keil.com/mdk5/legacy>。
- 安装
 - 安装选项可以选择默认值，KEIL MDK5 即安装在 C:/Keil_v5 目录下。
- GD32W51x PACK
 - PACK 文件位于 SDK 内 GD32W51x_Addon / KEIL 文件夹下，GigaDevice.GD32W51x_DFP_1.x.x.pack。
 - 双击运行，点击按钮“Next”，如[图 2-1. 安装 KEIL GD32W51x PACK](#)所示。完成后 PACK 文件就安装到了 C:/Keil_v5/ARM/PACK/GigaDevice 下。

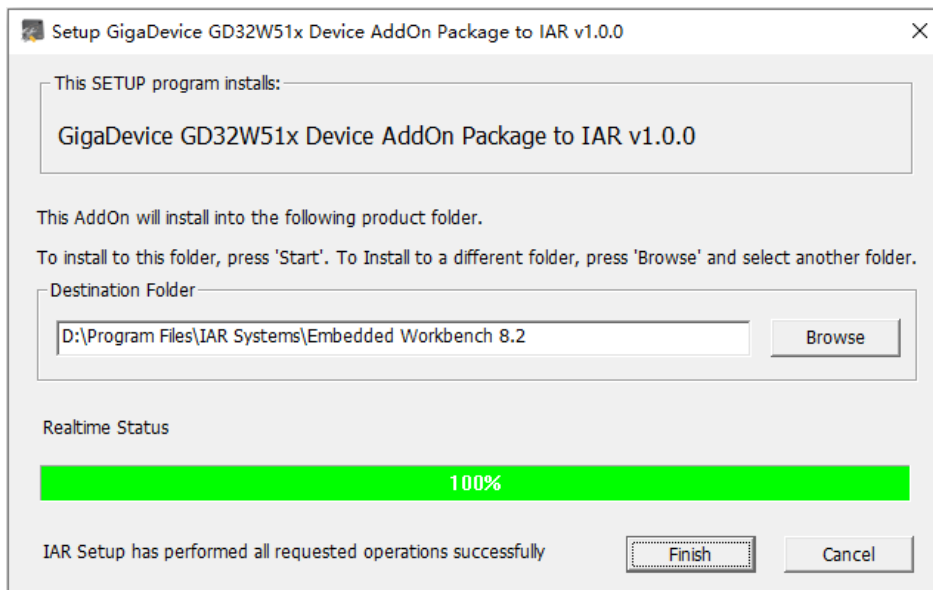
图 2-1. 安装 KEIL GD32W51x PACK



2.2. IAR 安装

- 下载
 - 官网地址: <https://www.iar.com/products/architectures/arm>。
 - 请选择 8.32 及以上版本, 以便更好地支持 ARM CM33。
- 安装
 - 可以选择默认安装, IAR 将安装在 C:/Program Files (x86)/ IAR Systems/ Embedded Workbench 8.2 下。
- GD32W51x Addon
 - Addon 位于 SDK 内 GD32W51x_Addon/IAR 文件夹下, IAR_GD32W51x_ADDON_1.x.x.exe。
 - 右键选择“以管理员身份运行”, 将路径选择到 IAR 安装路径, 点击按钮 Start 后, 完成安装, 见[图 2-2. 安装 IAR GD32W51x Addon](#)所示。

图 2-2. 安装 IAR GD32W51x Addon



2.3. CMAKE+GCC 安装

GCC 环境, 我们选择搭配 CMAKE 和 MAKE, 使用 GCC Toolchain 来进行编译。

- 安装 CMAKE
 - 下载地址: <https://cmake.org/download/>;
 - 请选择 3.15 及以上版本, 建议使用 cmake-3.20.3-windows-x86_64。
- 安装 MAKE
 - 下载地址: <http://ftp.gnu.org/gnu/make/>;
 - 建议使用 make-3.81。
- 安装 Toolchain

- 下载地址: <https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads>;
- 请选择 gcc-arm-none-eabi-7-2018-q2-update-win32 及以上版本, 建议使用 gcc-arm-none-eabi-9-2020-q2-update-win32。
- 添加到系统路径
 - 按照默认路径安装成功后, 将以下路径添加到系统环境变量中的 PATH 中。如果更换安装路径, 请将相应安装路径添加到系统路径中。
 - a) C :/ Program Files (x86) / GnuWin32 / bin
 - b) C :/ Program Files (x86) / GNU Tools Arm Embedded / 9 2020-q2-update / bin
 - c) C :/ Program Files / CMake / bin

如果调试器选择 JLINK, 需要安装 JLINK 驱动, 见 [2.3.1 JLINK 驱动安装](#)。如果使用 GD-LINK 或者 DAPLINK, 则使用 OpenOCD 进行下载和调试。OpenOCD 不需要安装, 执行文件见 GD32W51x_Addon / OpenOCD / bin / openocd.exe。

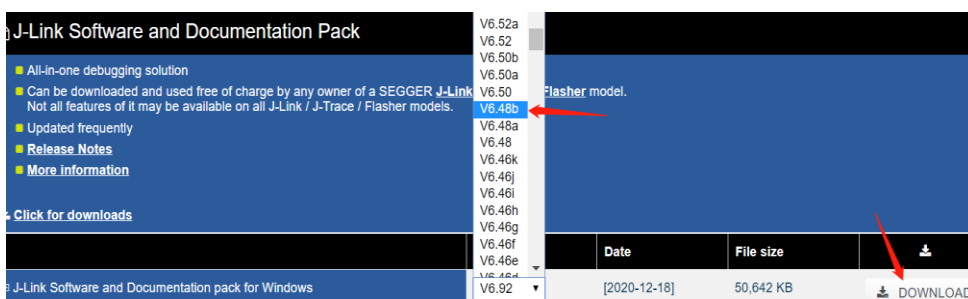
2.3.1. JLINK驱动安装

开发者可以选择使用 JLINK 进行开发调试, 如果不使用 JLINK, 可略过此节。

如果未安装 JLINK 驱动或者版本较低, 可以按照以下方法下载并安装 JLINK 驱动。

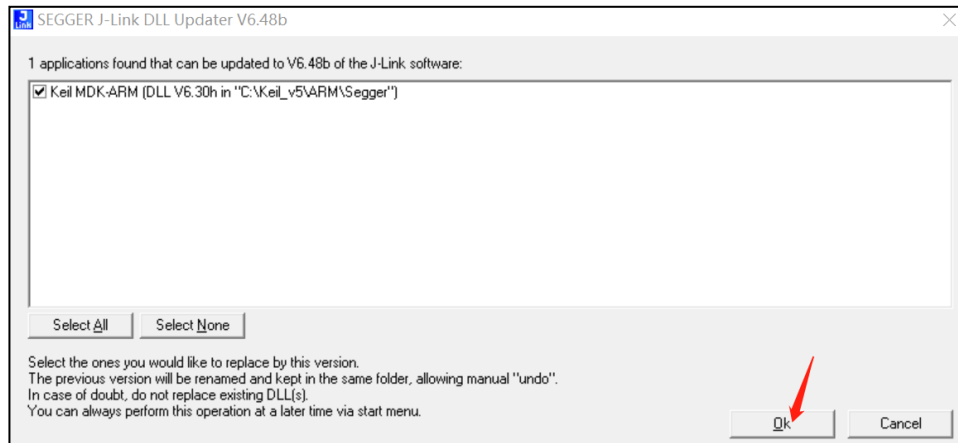
- 下载 JLINK 驱动
 - 进入官网: <https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack>
 - 下载 JLink_Windows_V648b 驱动, 如 [图 2-3. 下载 JLINK 驱动](#) 所示, 选择 Windows V6.48b 版本, 然后点 DOWNLOAD, 在弹出的界面选择接受协议并直接下载。

图 2-3. 下载 JLINK 驱动



- 安装 JLINK 驱动
 - 双击 JLink_Windows_V648b.exe 安装 Jlink 驱动, 除了 [图 2-4. 安装 JLINK 驱动](#) 界面选择 ok 外, 其余选择默认值。请确认将 JLink.exe 安装在 C :/ Program Files (x86) / SEGGER / JLink / JLink.exe。

图 2-4. 安装 JLINK 驱动



■ JLINK flash 烧录配置

- 为了方便在 GCC 开发环境中，使用 JLINK 烧录 flash，请进入软件包 GD32W51x_RELEASE / scripts / 目录，使用“管理员权限”运行 setup_jlink.bat 文件，就会将 flms 目录下的 FLASH 烧录算法文件和 JLinkDevices.xml 拷贝替换到 JLink 安装目录 C:/Program Files (x86)/SEGGER/JLink/中。

3. 开发需知

在着手开发之前，先了解下 SDK 执行程序组成员有哪些，如何正确配置 SDK，以及最终固件生成位置和构成。

对 SDK 进行编译和调试可以根据不同的开发工具 KEIL / IAR / GCC 选择相应的章节来了解。

3.1. SDK 执行程序组

SDK 最终生成的执行程序主要有两个：一个是 MBL_NS (Main Bootloader)，一个是 WIFI_IOT (有些配置文件中称为 NSPE，为了与 TrustZone 使能模式下的称呼保持一致)。它们最终都将被烧写到 FLASH 运行。上电之后，程序将从 MBL_NS 的 Reset_Handler 启动，然后跳转到 WIFI_IOT 主程序运行，如[图 3-1. 启动过程](#)所示。

图 3-1. 启动过程



3.2. SDK 配置

3.2.1. Platform 配置

配置文件为 GD32W51x_RELEASE / config / platform_def.h，主要内容见[图 3-2. Platform 配置](#)。

图 3-2. Platform 配置

```

41: #define CONFIG_PLATFORM          PLATFORM_ASIC_32W51X
42:
43: #ifndef CONFIG_PLATFORM
44: #error "CONFIG_PLATFORM must be defined!"
45: #elif CONFIG_PLATFORM >= PLATFORM_ASIC_32W51X
46: #define CONFIG_PLATFORM_ASIC
47: #else
48: #define CONFIG_PLATFORM_FPGA
49: #endif
50:
51: #define PLATFORM_BOARD_32W515T_START    0
52: #define PLATFORM_BOARD_32W515P_EVAL    1
53: #ifndef CONFIG_PLATFORM_ASIC
54: #define CONFIG_BOARD          PLATFORM_BOARD_32W515P_EVAL
55: #endif
56:
57: #define XIP_FLASH_SIP    0
58: #define XIP_FLASH_EXT    1
59: #define CONFIG_XIP_FLASH          XIP_FLASH_SIP
60:
61: #ifndef CONFIG_XIP_FLASH
62: #error "CONFIG_XIP_FLASH must be defined!"
63: #elif (CONFIG_XIP_FLASH == XIP_FLASH_EXT)
64: #define QSPI_FLASH_1_LINE    0
65: #define QSPI_FLASH_2_LINES    1
66: #define QSPI_FLASH_4_LINES    2
67: #define QSPI_FLASH_MODE          QSPI_FLASH_4_LINES
68: #endif
69:
70: #define CRYSTAL_26M    0
71: #define CRYSTAL_40M    1
72: #define PLATFORM_CRYSTAL          CRYSTAL_40M
73:
74: #if defined(CONFIG_BOARD) && (CONFIG_BOARD == PLATFORM_BOARD_32W515P_EVAL)
75: #define LOG_UART    USART2
76: #else
77: #define LOG_UART    USART1
78: #endif
79:
80: #ifndef CONFIG_PLATFORM_FPGA
81: #define RFAD_SPI    SPI0
82: #endif
83:
84: #define TOTAL_SRAM_SIZE          (448 * 1024)
85:
86: #define CONFIG_HW_SECURITY_ENGINE

```

- 如果是 EVAL 开发板，请选择：
 - #define CONFIG_BOARD PLATFORM_BOARD_32W515P_EVAL
- 如果是 START 开发板，请选择：
 - #define CONFIG_BOARD PLATFORM_BOARD_32W515T_START

- 如果是内置 SIP FLASH, 请选择:
 - #define CONFIG_XIP_FLASH XIP_FLASH_SIP
- 如果是外挂 EXT FLASH, 请选择:
 - #define CONFIG_XIP_FLASH XIP_FLASH_EXT
- 如果晶振选择 40 MHz, 请选择:
 - #define PLATFORM_CRYSTAL CRYSTAL_40M
- 如果晶振选择 26 MHz, 请选择:
 - #define PLATFORM_CRYSTAL CRYSTAL_26M

3.2.2. SRAM布局

配置文件为 GD32W51x_RELEASE / config / config_gdm32_ntz.h。修改以下宏定义值, 可以对可执行程序段 MBL 及 NSPE 占用的 SRAM 空间进行规划。这些值是偏移地址, 基地址定义在该文件开头处。

这里的 NSPE 就是指 WIFI_IOT 执行程序段, Non-Secure Process Environment (NSPE), 是 TrustZone 使能后对 WIFI_IOT 执行程序段的命名, 与 Secure Process Environment (配置文件中看到的 PROT) 相对, 表示 WIFI_IOT 运行在非安全环境中。

标注 “!Keep unchanged!” 的行不能修改, 否则会影响 ROM 中代码 MbedTLS 的运行。

图 3-3. SRAM 布局

```

/* SRAM LAYOUT */
#define RE_MBL_DATA_START      0x200          /* !Keep unchanged! */
#define RE_NSPE_DATA_START    0x200          /* !For SIP flash! */
    
```

每个可执行程序段内部的 SRAM 空间规划, 可以查看 Project 选项的 Linker 页面中的 Scatter File, 文件中的宏定义可以在 xxx_region.h 中找到, 例如 nspe_region.h。

3.2.3. FLASH布局

配置文件为 GD32W51x_RELEASE / config / config_gdm32_ntz.h。修改以下宏定义值, 可以对可执行程序段 MBL 及 NSPE 占用的 FLASH 空间进行规划。这些值是偏移地址, 基地址定义在该文件开头处。NSPE 的解释见 [3.2.2 SRAM 布局](#)。

标注 “!Keep unchanged!” 的行不能修改, 否则会影响工程运行。

图 3-4. FLASH 布局

```

/* FLASH LAYEROUT */
#define RE_VTOR_ALIGNMENT      0x200          /* !Keep unchanged! */
#define RE_MBL_OFFSET          0x0           /* !Keep unchanged! */
#define RE_SYS_STATUS_OFFSET   0x8000       /* !Keep unchanged! */
#define RE_IMG_0_PROT_OFFSET   0xA000
#define RE_IMG_0_NSPE_OFFSET   0xA000
#define RE_IMG_1_PROT_OFFSET   0x100000
#define RE_IMG_1_NSPE_OFFSET   0x100000
#define RE_IMG_1_END_OFFSET    0x200000
    
```


每个可执行程序段内部的 FLASH 空间规划，可以查看 Project 选项的 Linker 页面中的 Scatter File，文件中的宏定义可以在 xxx_region.h 中找到，例如 nspe_region.h。

3.2.4. 固件版本号

配置文件为 GD32W51x_RELEASE / config / config_gdm32_ntz.h。修改以下宏定义值，可以指定版本号。但是影响将来用户升级的版本号只有 RE_NSPE_VERSION。

MBL 只能本地升级，NSPE 可以支持在线升级，SDK 发布的版本号与 RE_NSPE_VERSION 保持一致，在 UART 启动日志中可以看到该版本号打印，如：“SDK version: V1.0.0”。

图 3-5. 固件版本号

```
/* FW_VERSION */
#define RE_MBL_VERSION          0x01000000
#define RE_NSPE_VERSION        0x01000000
```

3.2.5. APP配置

配置文件为 GD32W51x_RELEASE / NSPE / WIFI_IOT / app / app_cfg.h。可以选择是否打开一些应用，例如：ATCMD，SSL 示例，阿里云，FATFS 等等。

3.3. 固件生成和下载

编译后脚本自动生成的 BIN 和 HEX 文件统一放置在目录 GD32W51x_RELEASE / scripts / images / 中。其中 image-*.bin 用于生产或升级。

Image-all.bin 中包含可执行程序段 MBL_NS 和 WIFI_IOT，该固件可用于生产，烧录到空白 FLASH 中。

Image-ota.bin 仅包含 WIFI_IOT（NSPE），该固件可用于升级。

关于下载，除了通过 IDE 工具下载外，对于 START 开发板，还可以通过 U 盘拖拽的方法进行下载。将开发板通过 USB 线插入电脑，可以看到[图 3-6. 设备和驱动器列表](#)所示 DAPLINK 盘。将 image-all.bin 文件，直接拷贝进 DAPLINK 盘，就能完成对 GD32W51x 芯片的 FLASH 烧写。也支持 HEX 文件下载。

图 3-6. 设备和驱动器列表



3.4. 正确日志示例

在固件组（MBL_NS+WIFI_IOT）下载成功后，打开串口工具，按下开发板上的 Reset 键，可

以看到[图 3-7. 工程启动信息](#)。如果出现异常，请查阅[常见问题](#)，看能否找到帮助。

图 3-7. 工程启动信息

```
GIGA DEVICE
MBL: Boot from Image 0.
SDK first message for GDM32W51x
SDK git revision: v1.0.1-1-g59fcaae3-59fcaae3d82d9aa8
SDK version: V1.0.1
SDK build date: 2022/02/28 12:09:00

System reset mode: pin,
System clock is 180000000
WiFi SW init OK.
WiFi RF init OK.
WiFi BB config OK.
WiFi RF calibration OK.
WiFi MAC address: 76:ba:ed:1c:cb:47
wifi netlink: device opened!
Fatfs: mount succeed

#
```

4. KEIL 工程

本章将介绍如何在 KEIL 下编译和调试 SDK。

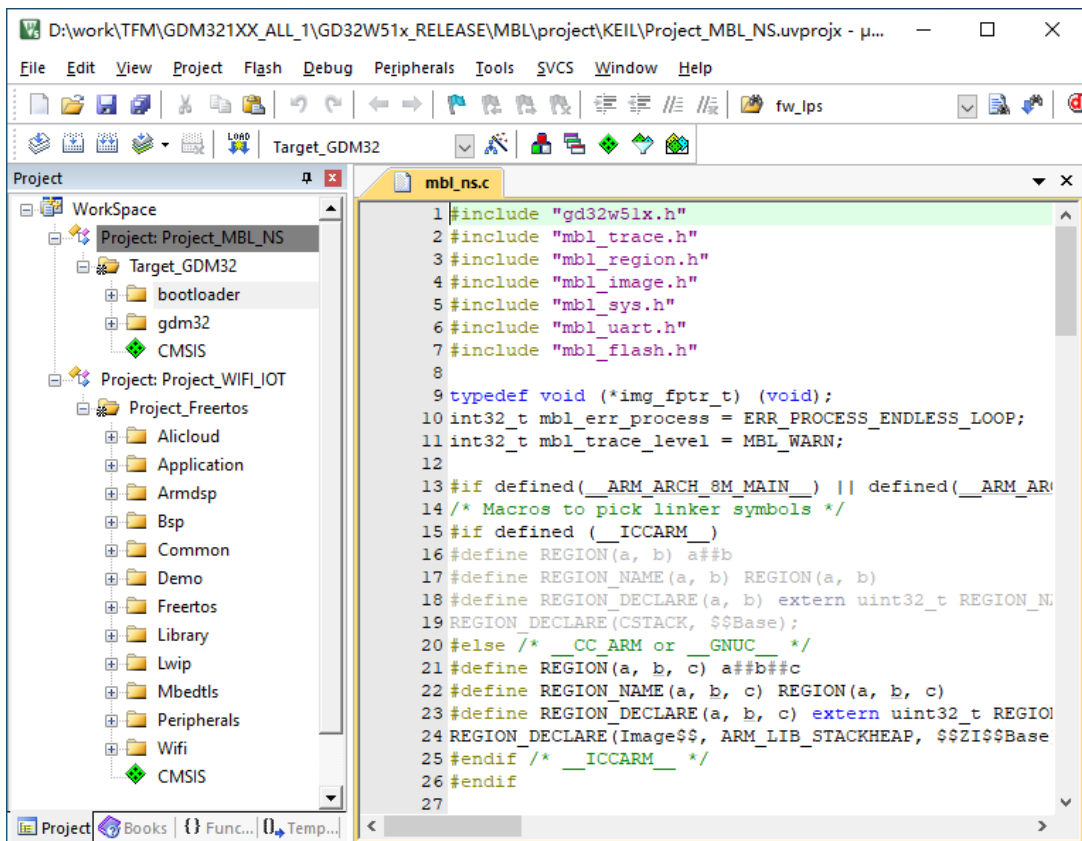
工程组由 MBL_NS / WIFI_IOT 这两个工程组成。WIFI_IOT 包含 WiFi 协议栈、外设驱动及应用程序等等，MBL_NS 主要负责从两个 WIFI_IOT 固件（一个为当前固件，一个为升级后固件）中选择一个正确的运行。

4.1. 工程组

为了使 KEIL 能够识别 GD32W51x，打开工程前请先安装目录 GD32W51x_Addon/KEIL 下 GigaDevice.GD32W51x_DFP_1.x.x.pack。

从 GD32W51x_RELEASE 根目录下找到 MultiProject_NS.uvmpw，双击使用 KEIL 打开工作空间，如[图 4-1. KEIL 工程组界面](#)所示。

图 4-1. KEIL 工程组界面



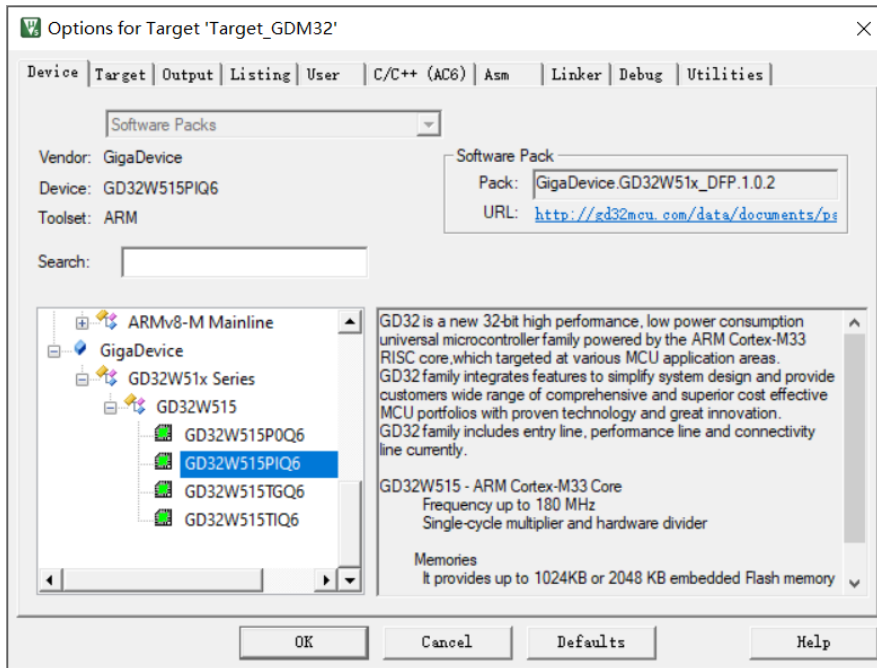
4.2. 工程配置

4.2.1. Device选择

GD32W515 芯片有多款选择，见 KEIL Options->Device 页面，安装好 PACK 之后就能看到，如 [图 4-2. KEIL 选项 Device 页面](#) 所示。我们默认选择的是 GD32W515PIQ6, 2M SIP FLASH, 448K SRAM。

如果使用的是其他款，这里需要重新选定一下。

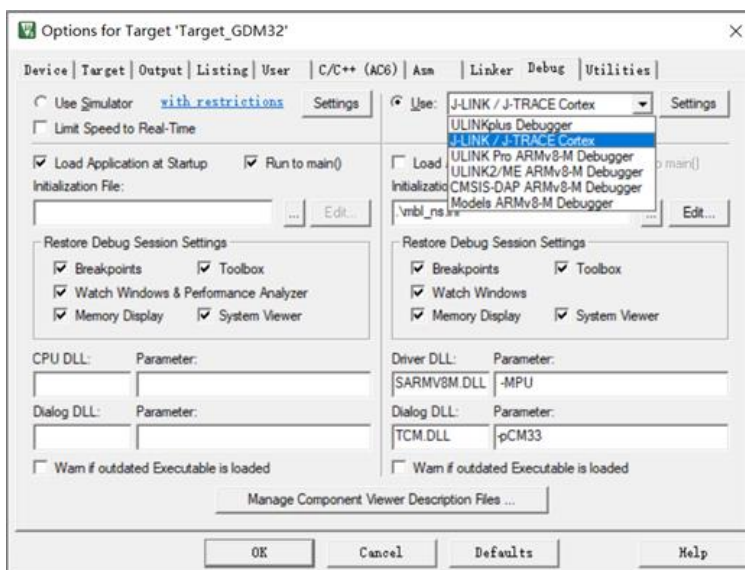
图 4-2. KEIL 选项 Device 页面



4.2.2. 调试器配置

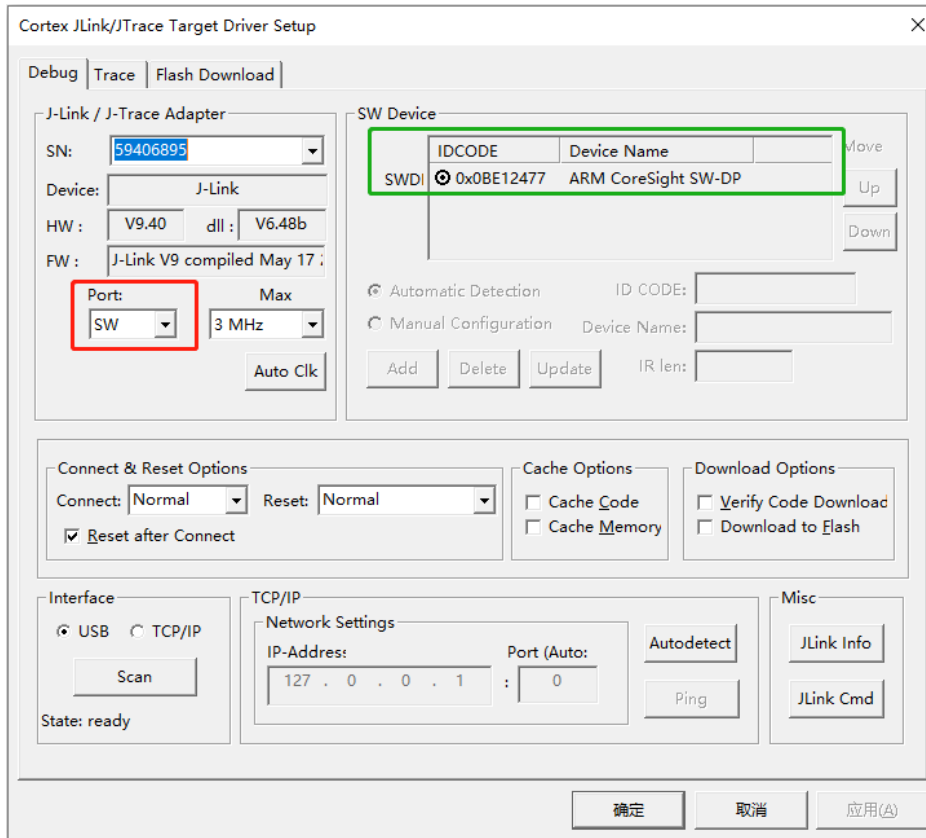
如果是 EVAL 开发板，选择板载的 GD-LINK，或者 START 开发板，选择板载的 DAPLINK，Debug 页面的调试器都是选择“CMSIS-DAP ARMv8-M Debugger”。如果是 JLINK 就选择“J-LINK / J-TRACE Cortex”。如[图 4-3. KEIL 选项 Debug 页面](#)所示。

图 4-3. KEIL 选项 Debug 页面



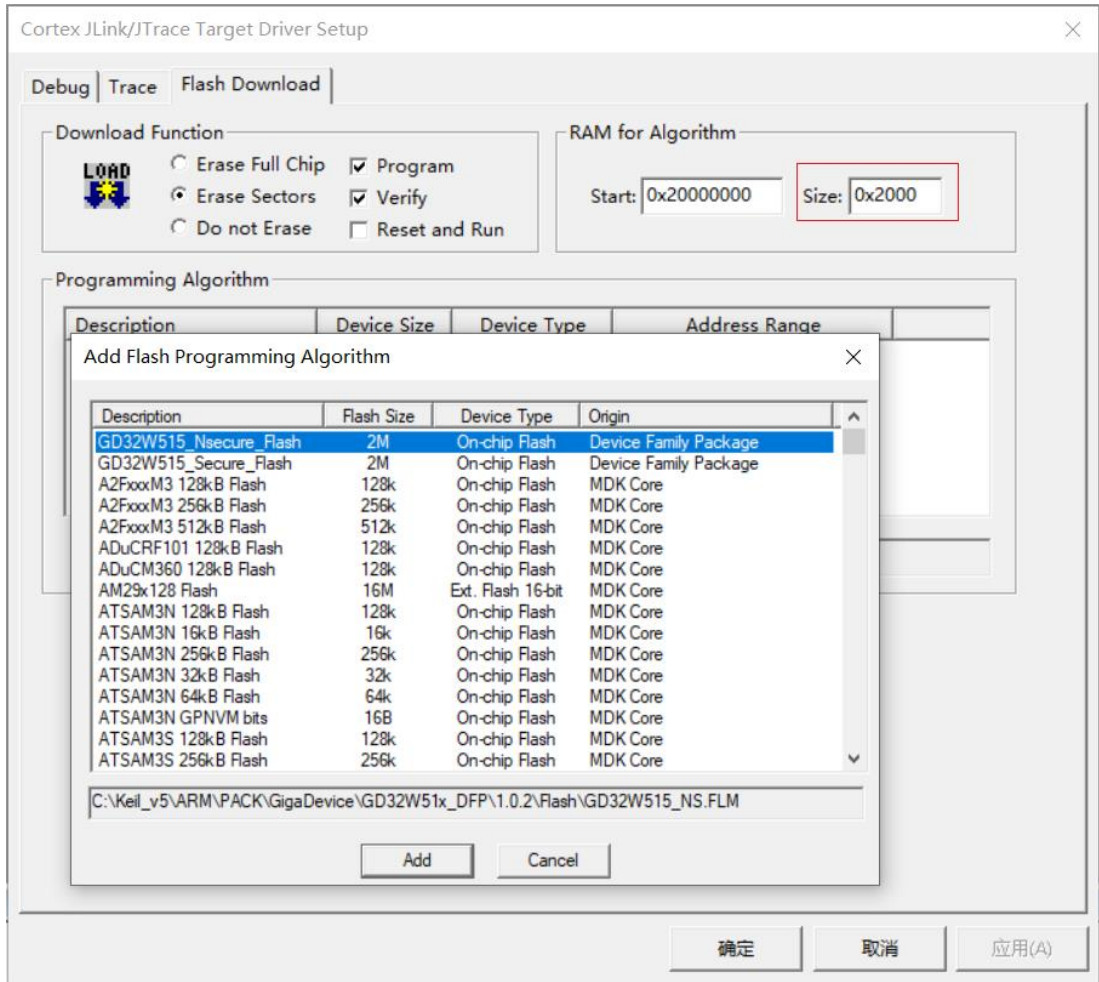
在 KEIL 工程里面默认都已经配置好了 Debugger 相关设定，需要注意的是 JLINK 端口一定要选成 SW。使用 JLINK Debugger，连接 GD32W515 芯片，能看到如[图 4-4. KEIL 选项 JLINK Settings Debug 页面](#)所示页面。

图 4-4. KEIL 选项 JLINK Settings Debug 页面



如果切换了 Device, Debugger Settings 需要重新配置。选择正确的 Debugger 之后, 还需要选择 FLASH 下载算法, 如[图 4-5. KEIL 选项 Flash Download 页面](#)所示。注意“RAM for Algorithm”这里的“Size”需要改为 0x2000。

图 4-5. KEIL 选项 Flash Download 页面



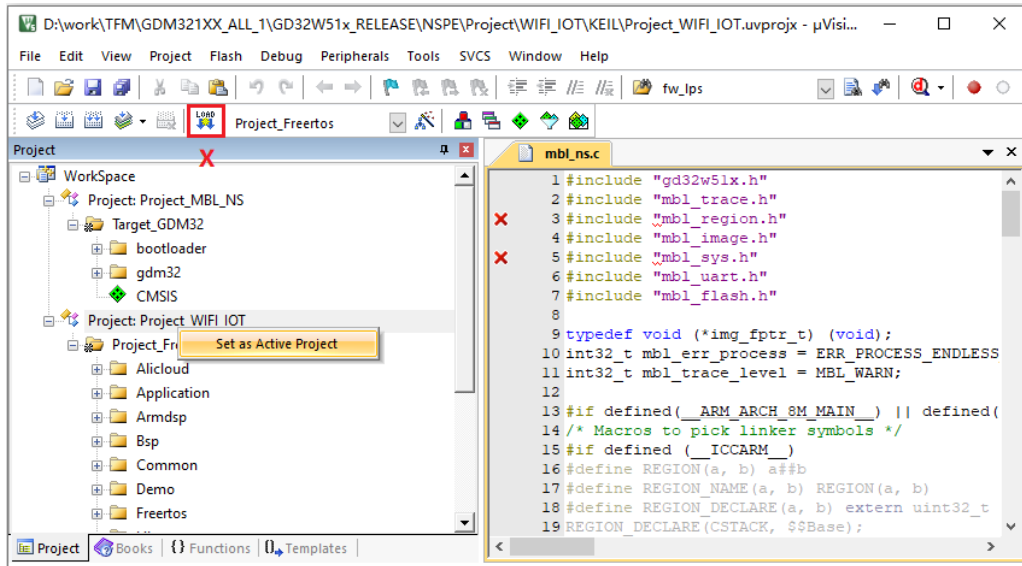
4.3. 编译

■ 编译

如果要编译某个工程，需要确认该工程是否为当前活动工程。如果不是，需要选中 Project 后点击右键，点击“Set as Active Project”，见[图 4-6. 切换当前工程](#)，然后再点

击图标   来编译工程。

图 4-6. 切换当前工程



■ 批量编译

也可以选择批量编译，见[图 4-7. 批量编译](#)，点击之后，会依次编译 Project_MBL_NS 和 Project_WIFI_IOT。点击“Batch Setup...”可以进行批量编译定制，见[图 4-8. 批量编译设置](#)。Project_WIFI_IOT 可以选择 OS 为 FreeRTOS 或 RT-Thread。

图 4-7. 批量编译

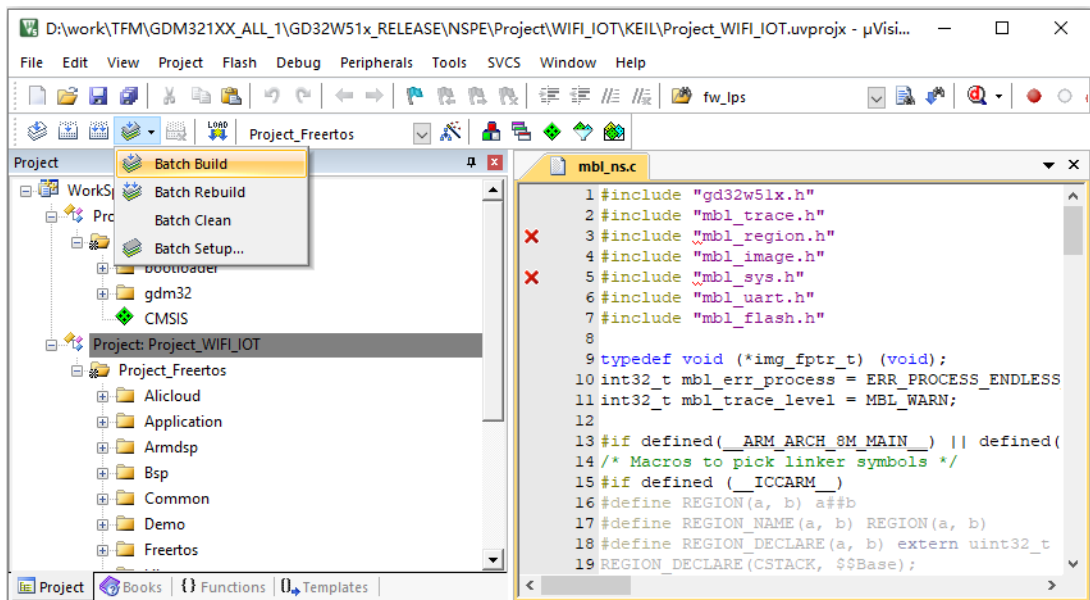
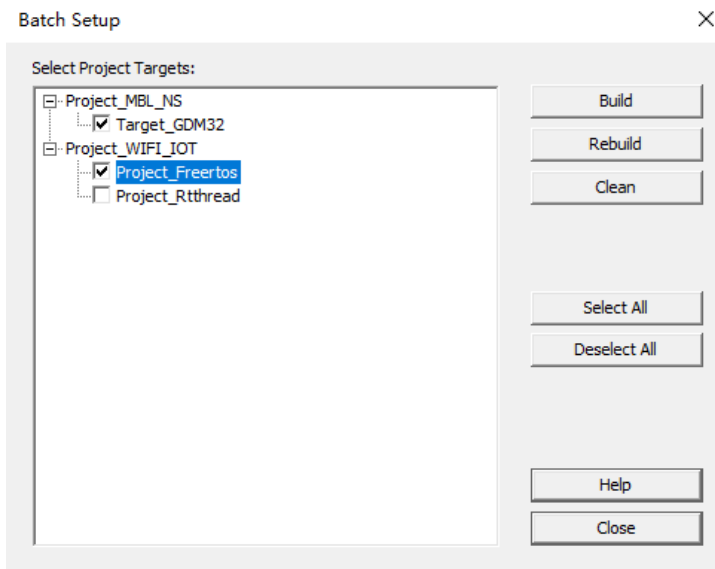


图 4-8. 批量编译设置



■ User Command

在每个 Project 上定义了 User Command，在编译链接成功后，会自动运行 xxx_afterbuild.bat。这个脚本主要是从输出 AXF 文件中提取生成 BIN 格式固件，如果是 nspe_afterbuild.bat，还会多做一件事情，就是将 mbl_ns.bin 和 nspe.bin 进行拼接得到 image_all.bin。

4.4. 下载


点击图标  进行下载。检查烧录是否成功，如 [图 4-9. KEIL 烧录结果](#) 所示。

图 4-9. KEIL 烧录结果

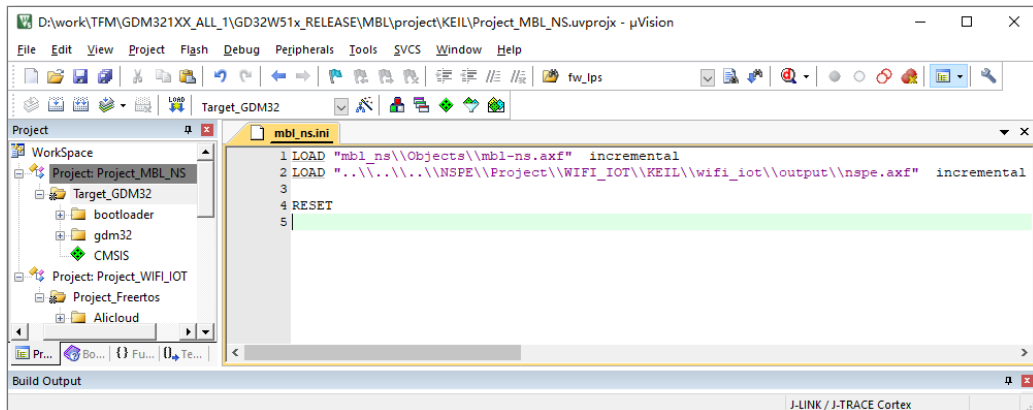
```
Erase Done.  
Programming Done.  
Verify OK.  
Flash Load finished at 16:22:56
```

4.5. 调试

■ 初始化文件

调试前还需要编辑“Initialization File”，即 xxx.ini 文件。如果要实现跨工程调试，需要将多个工程的 AXF 文件 LOAD 进来，参见 GD32W51x_RELEASE / MBL / project / KEIL / mbl_ns.ini，如 [图 4-10. 调试初始化文件](#) 所示。

图 4-10. 调试初始化文件



■ 开始调试


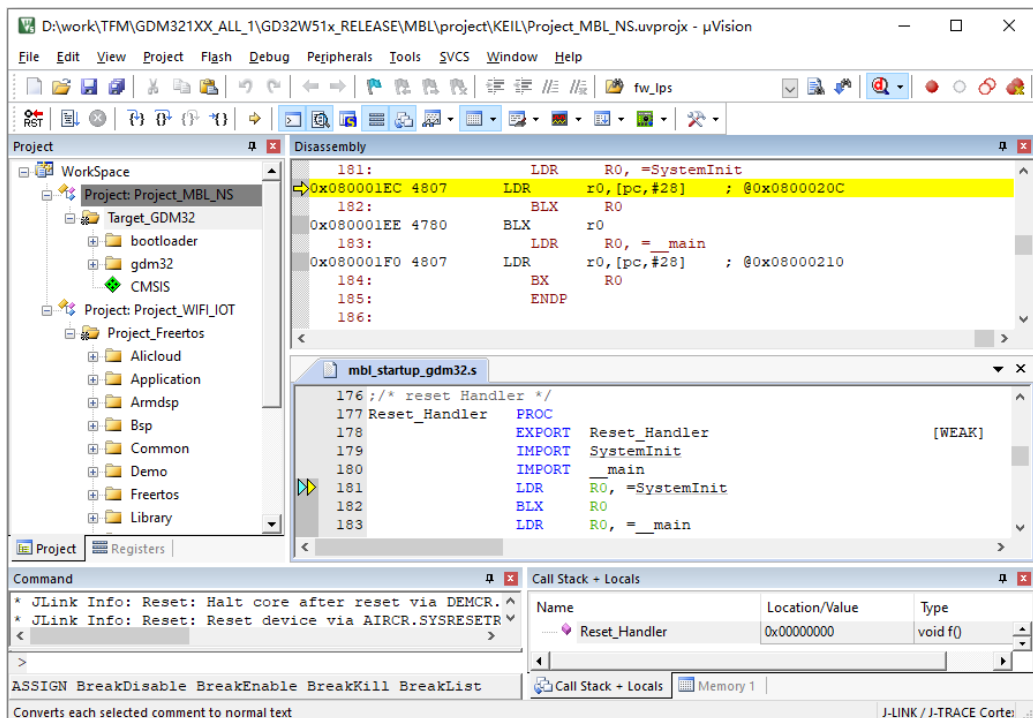
以上准备好之后，点击图标 ，或者从菜单“Debug”->“Start/ Stop Debug Session”就可以开始单步调试了，如[图 4-11. KEIL 调试界面](#)所示。

图 4-11. KEIL 调试界面



5. IAR 工程

本章将介绍如何在 IAR 下编译和调试 SDK。

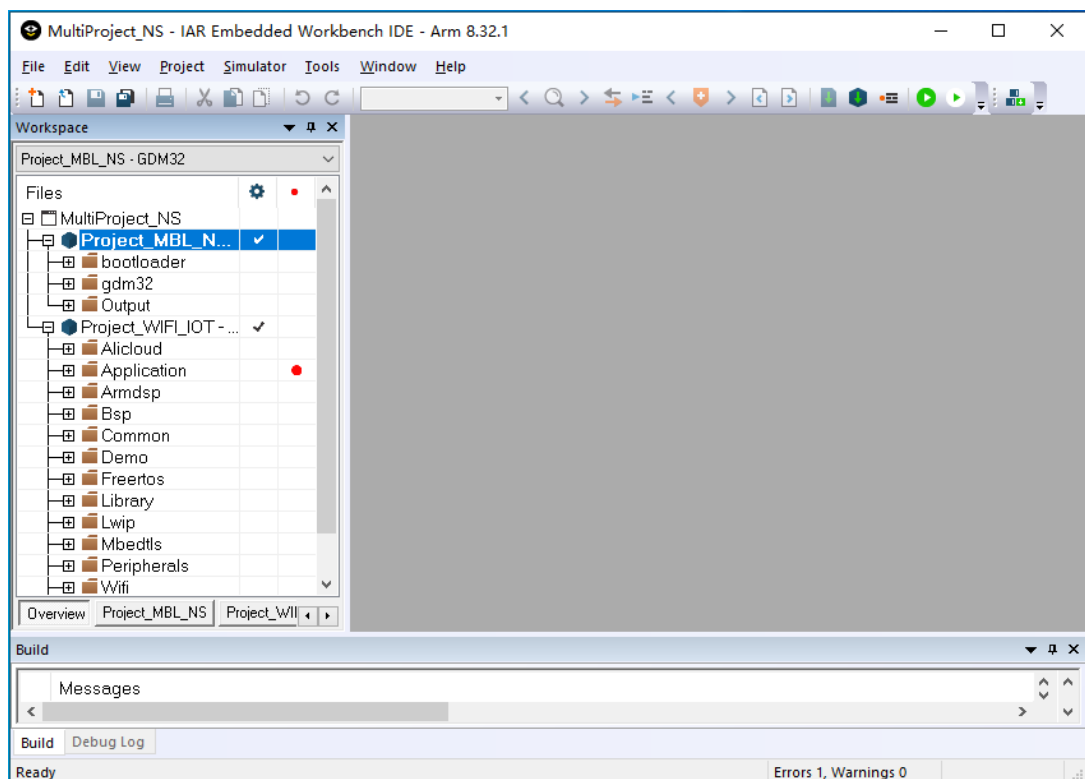
工程组由 MBL_NS 和 WIFI_IOT 这两个工程组成。WIFI_IOT 包含 WiFi 协议栈、外设驱动及应用程序等等，MBL_NS 主要负责从两个 WIFI_IOT 固件（一个为当前固件，一个为升级后固件）中选择一个正确的运行。

5.1. 工程组

从 GD32W51x_RELEASE 根目录下找到 MultiProject_NS.eww，双击打开 IAR 工作空间，如 [图 5-1. IAR 工程组](#) 所示。

请确认已经安装好 IAR_GD32W51x_ADDON_1.x.x.exe。

图 5-1. IAR 工程组



5.2. 工程配置

IAR Project 的配置，主要体现在“Options...”对话框中，也即下文提到的 IAR 选项。包括 Device 选择、输出文件选择、编译选项、链接选项以及下载和调试选项等等。

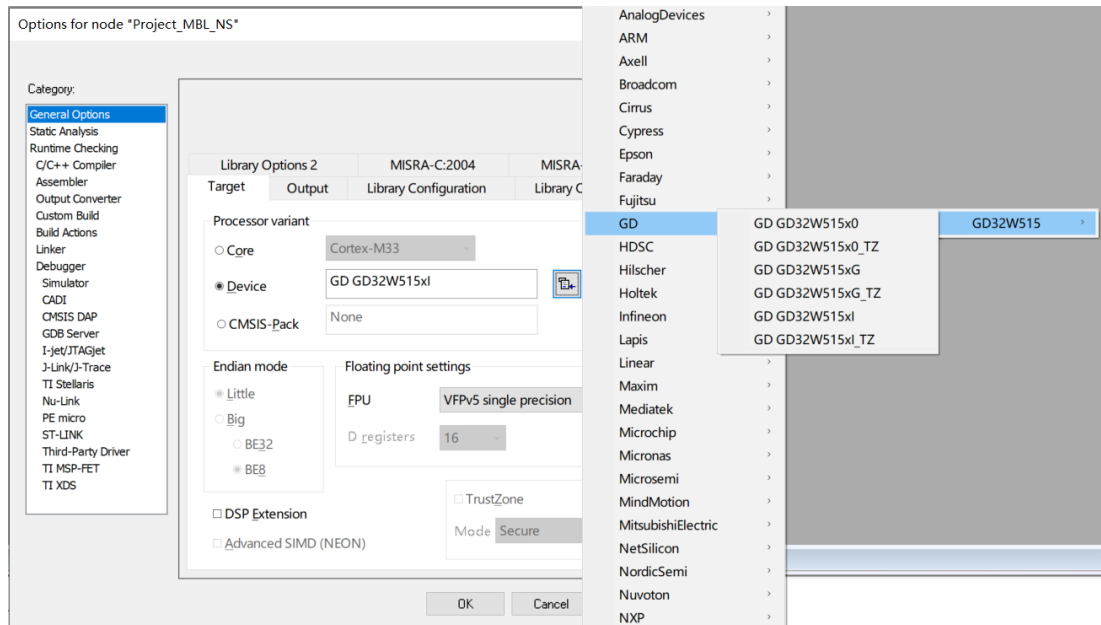
有可能需要调整的主要是 Device 选择和调试器配置。

5.2.1. Device选择

GD32W515 芯片有多款选择，见 IAR Options -> General Options -> Target 页面，安装好 Addon 之后就能看到，如[图 5-2. IAR 选项 Target 页面](#)所示。我们默认选择的是 GD32W515xl, 2M SIP FLASH, 448K SRAM。

如果使用的是其他款，这里需要重新选定一下。

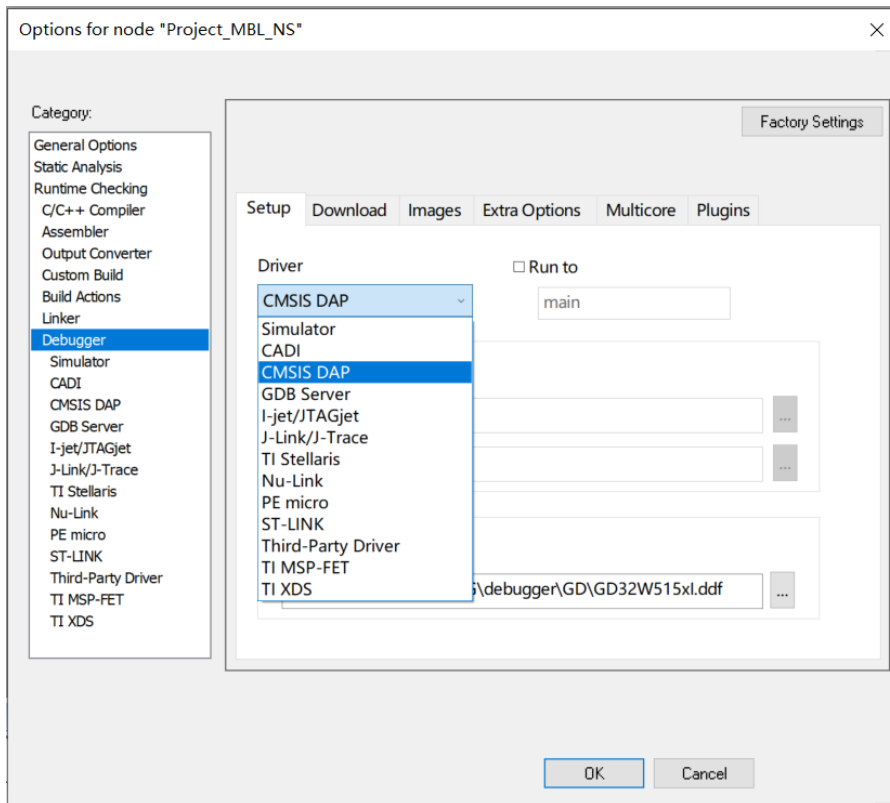
图 5-2. IAR 选项 Target 页面



5.2.2. 调试器配置

如果是 EVAL 开发板，选择板载的 GD-LINK，或者 START 开发板，选择板载的 DAPLINK，Debugger Setup 页面的 Driver 都是选择“CMSIS-DAP”。如果是 JLINK 就选择“J-LINK/ J-TRACE”。如[图 5-3. IAR 选项 Debugger 页面](#)所示。

图 5-3. IAR 选项 Debugger 页面

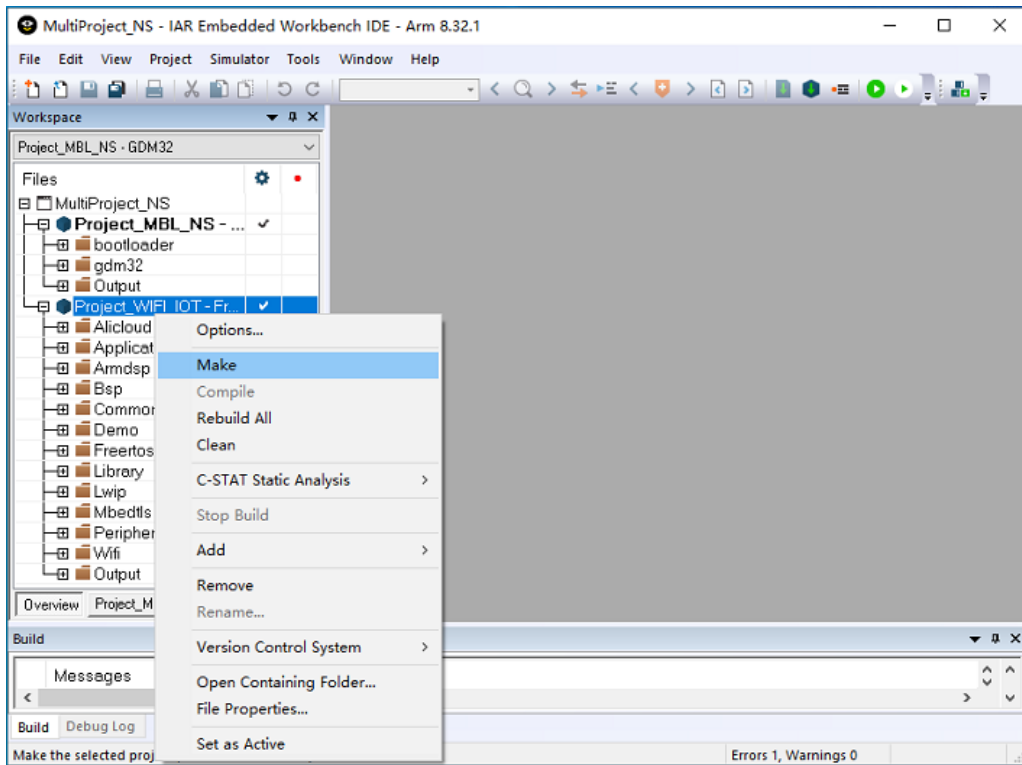


5.3. 编译

■ 编译

编译只需在相应 Project 上点击右键选择“Make”即可。如果需要全部重新编译，可以选择“Rebuild All”，如[图 5-4. IAR 编译](#)所示。

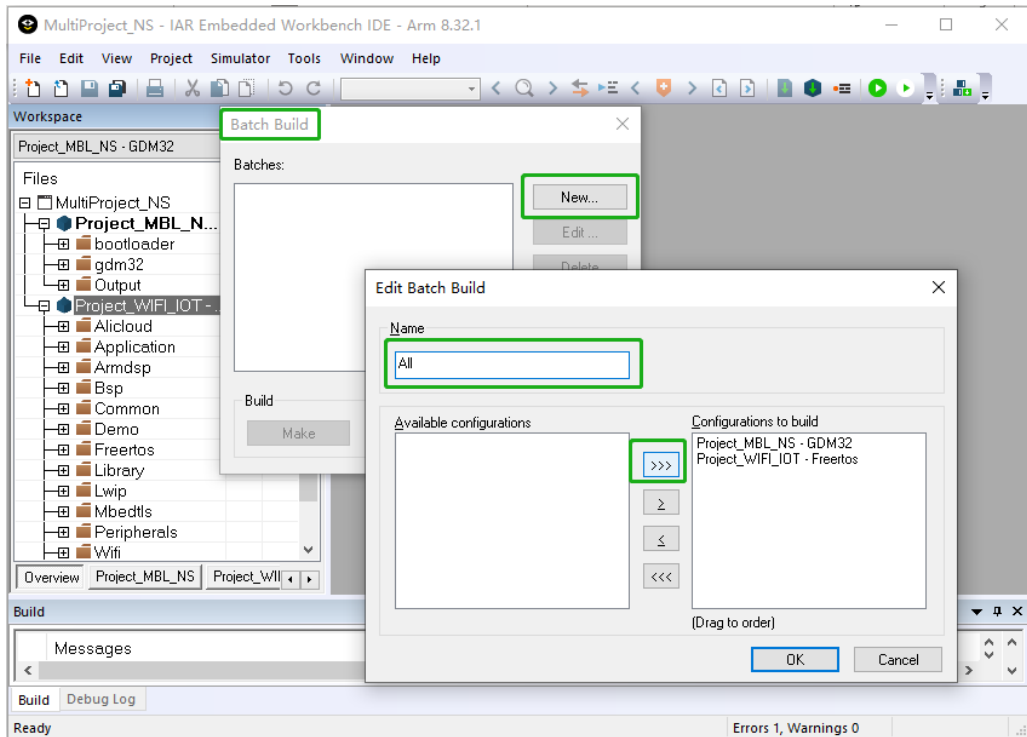
图 5-4. IAR 编译



■ 批量编译

批量编译需要先进行设置，点击菜单栏“Project”，在下拉菜单中选择“Batch build...”，会弹出“Batch Build”对话框，见[图 5-5. IAR 批量编译设置](#)，点击“New”之后，弹出“Edit Batch Build”对话框，将所有 Project 选到右框，填写上“Name”后点击“OK”。回到“Batch Build”对话框，选择“All”，点击“Make”，会依次编译 Project_MBL，Project_WIFI_IOT。

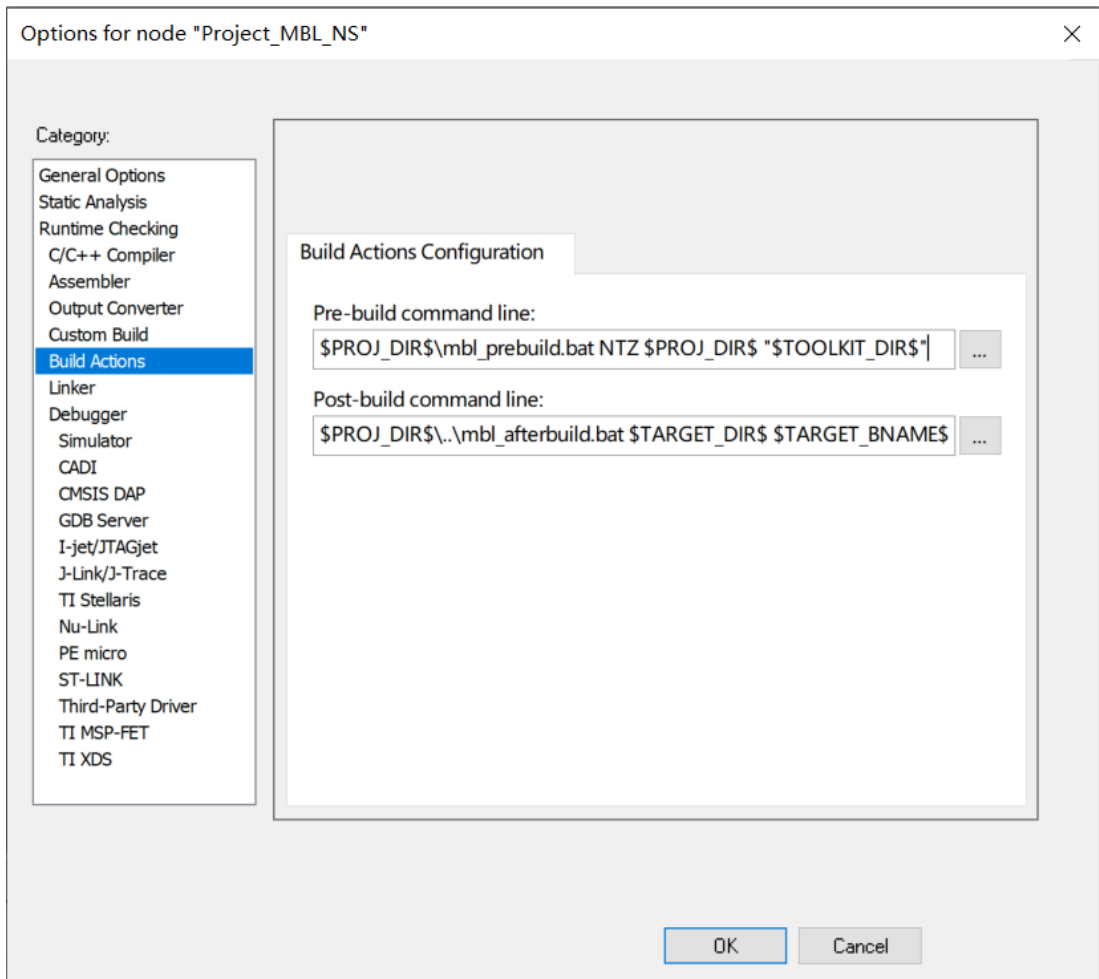
图 5-5. IAR 批量编译设置



■ Build Actions

每个 Project 用户可以自定义“Build Actions”，见[图 5-6. IAR Build Actions](#)。目前每个 Project 上都定义了“Pre-build command line”和“Post-build command line”，分别在编译前和编译后执行。Xxx_prebuild.bat 主要为了完成 ICF 文件预编译和生成编译时间戳。Xxx_afterbuild.bat 这个脚本主要是从输出 AXF 文件中提取生成 BIN 格式固件，如果是 nspe_afterbuild.bat，还会多做一件事情，就是将 mbl_ns.bin 和 nspe.bin 进行拼接得到 image_all.bin。

图 5-6. IAR Build Actions



5.4. 下载

可以使用“Project”->“Download active application”直接下载。IAR Debug Log 窗口中显示的正确烧录日志如[图 5-7. IAR 烧录结果](#)所示。

图 5-7. IAR 烧录结果

```
Reset: Halt core after reset via DEMCR.VC_CORERESET.
Reset: Reset device via AIRCR.SYSRESETREQ.
Hardware reset with strategy 0 was performed
Initial reset was performed
992 bytes downloaded (12.42 Kbytes/sec)
Loaded debuggee: C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.2\arm\config\flashloader\GD\FIashGD32W515x\IPAM448K.out
Target reset
Unloaded macro file: C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.2\arm\config\flashloader\GD\FIashGD32W515x\mac
Downloaded D:\Work\2021\GD32W51x_RELEASE_V1.0.1\MBL\project\AR\MBL-NS\Exe\mbl-ns.axf to flash memory.
12603 bytes downloaded into FLASH (7.36 Kbytes/sec)
IAR Embedded Workbench 8.32.1 (C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.2\arm\bin\armproc.dll)
```

5.5. 调试

- 调试

点击“Debug without Downloading”按钮，开始调试，见[图 5-8. IAR 调试按钮](#)和[图 5-9. IAR 开始调试](#)。之后就可以开始单步调试了。

图 5-8. IAR 调试按钮

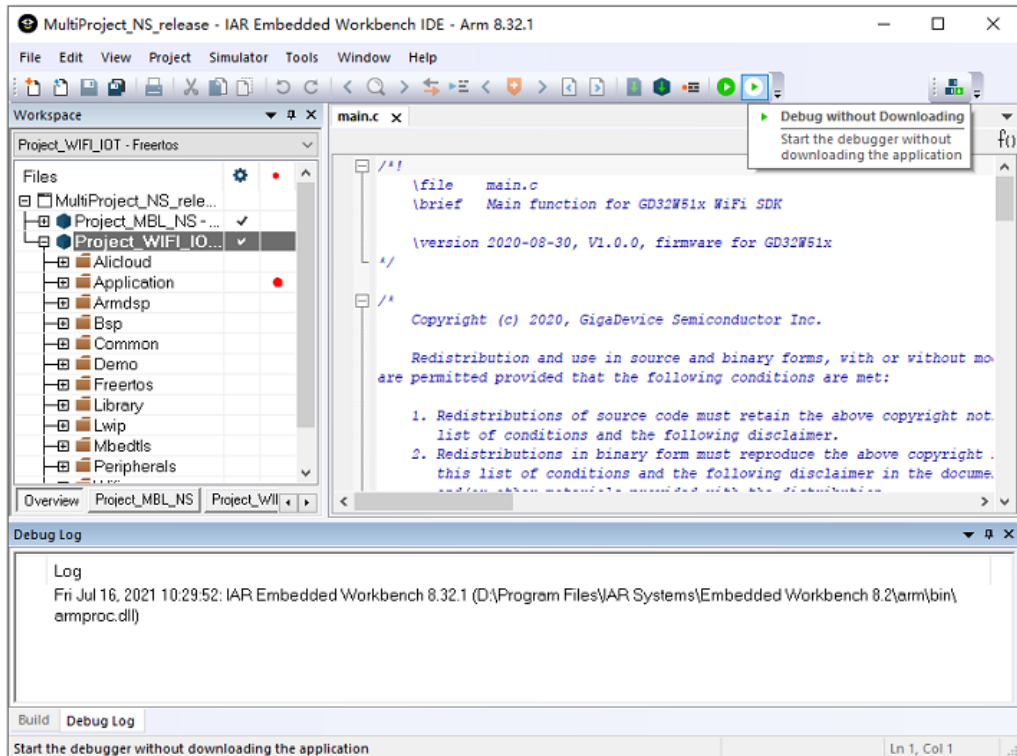
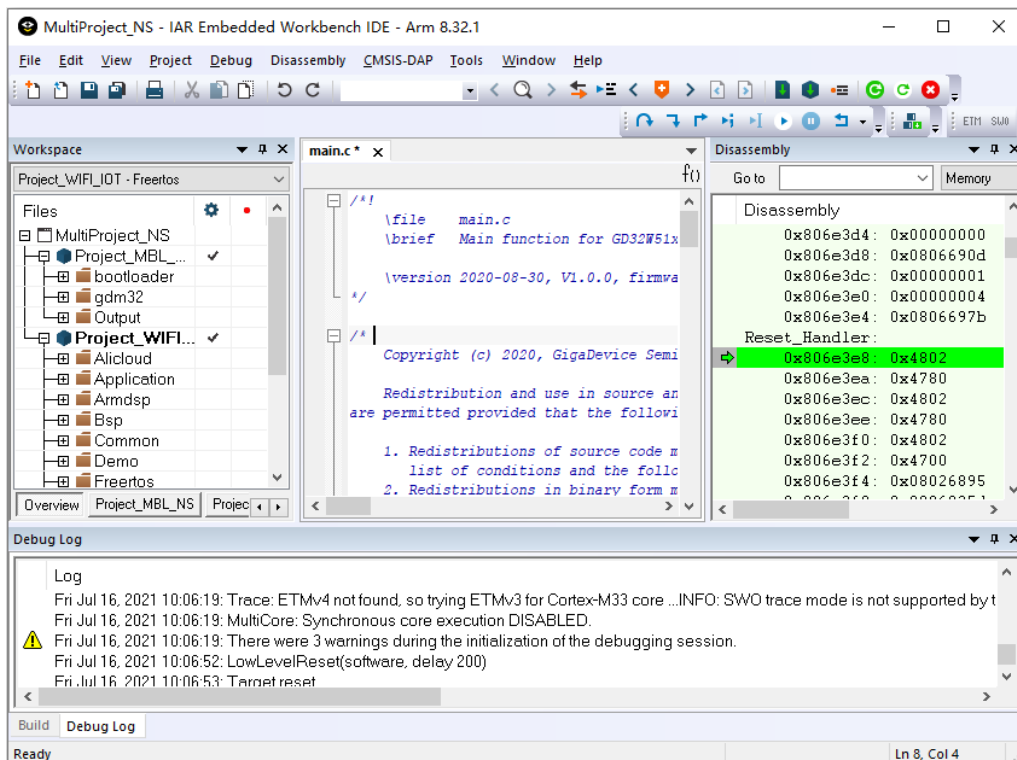


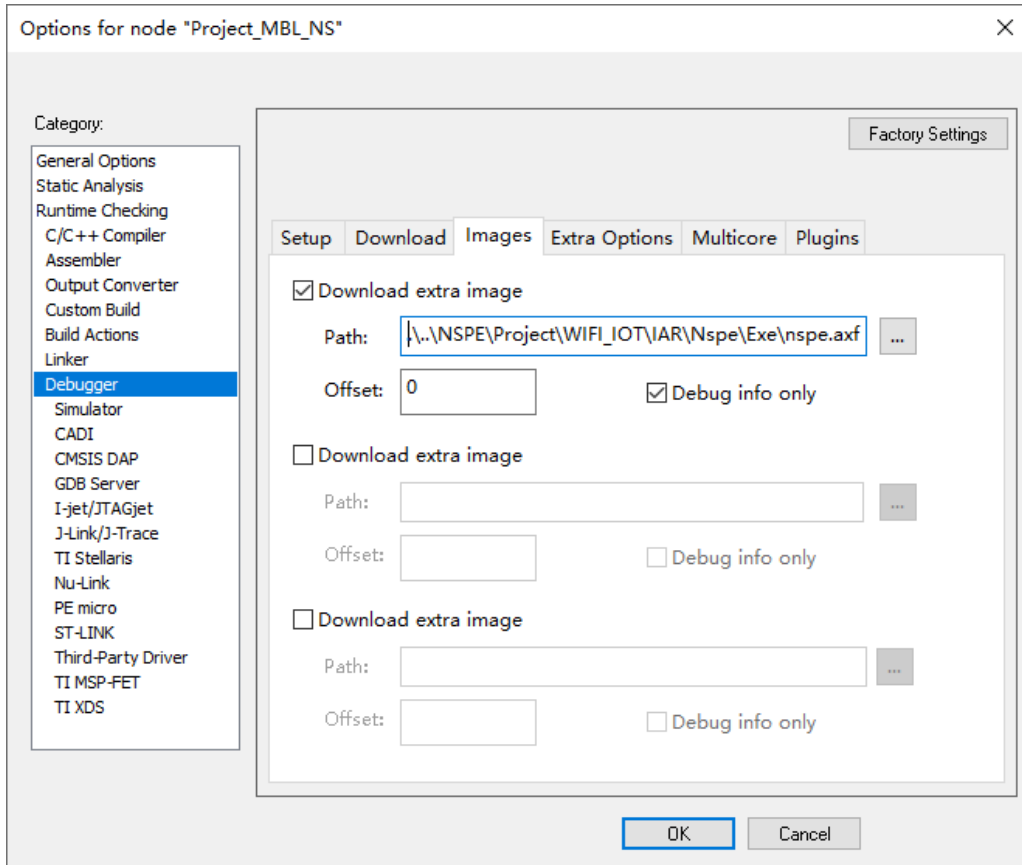
图 5-9. IAR 开始调试



■ 多镜像调试

如果需要跨 Project 调试，需要在“Debugger”->“Images”页面，将下一跳包含调试信息的输出文件，*.axf 或者*.out，填写到“Path”框中，“Offset”填 0，勾选“Debug info only”，如[图 5-10. IAR 多镜像调试](#)所示。

图 5-10. IAR 多镜像调试



6. GCC 工程

本章将介绍如何在 GCC 下编译和调试 SDK。

目前仅支持 JLINK 调试，CMSIS-DAP 功能待完善。

6.1. 工程组

使用 `cmake + gnu make` 工具进行构建 SDK，主要有两个 Target: `WIFI_IOT` 和 `MBL_NS`。`WIFI_IOT` 包含 WiFi 协议栈、外设驱动及应用程序等等，`MBL_NS` 主要负责从两个 `WIFI_IOT` 固件（一个为当前固件，一个为升级后固件）中选择一个正确的运行。

6.2. 编译

开启 Windows 命令窗口，请不要使用 PowerShell。进入 `GD32W51x_RELEASE` 目录，执行以下命令：

- 创建 `cmake` 构建目录：
 - `mkdir cmake_build`
- 进入 `cmake` 构建目录：
 - `cd cmake_build`
- 执行 `cmake` 命令，生产 Makefile：
 - `cmake -G "Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE:PATH = ./ scripts / cmake / toolchain.cmake`
- 执行 `make` 命令，编译。
 - `make -j`

以上步骤，已经写进 `gcc_build_ns.bat`，双击运行后可以完成第一次编译。

- 如果只需要编译 `nspe`，进入 `cmake_build` 目录运行：
 - `make nspe -j`
- 如果只需要编译 `mbl`，进入 `cmake_build` 目录运行：
 - `make mbl-ns -j`

6.3. 下载

开启 windows 命令窗口，请不要使用 PowerShell。进入 `GD32W51x_RELEASE` 目录，执行以下命令：

- `gcc_download_ns.bat JLINK`

会自动使用 JLINK 下载 `scripts / images / image-all.bin`。

6.4. 调试

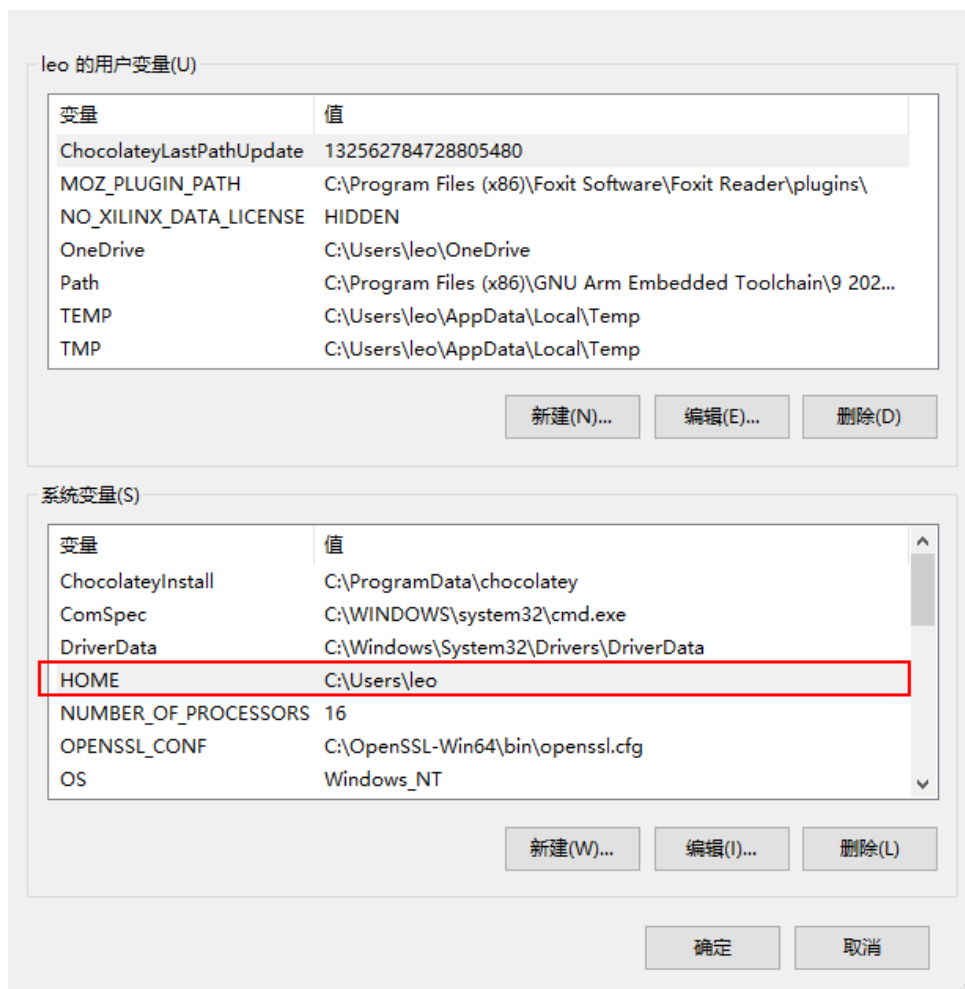
6.4.1. 启动GDB server

调试使用工具是 GDB + JLink，请确保 Jlink 连接正常。

准备条件：

- 创建 HOME 环境变量，值建议为用户目录，如 C :/ Users / leo，详见[图 6-1. 创建 HOME 环境变量](#)所示；

图 6-1. 创建 HOME 环境变量



- 在 HOME 变量所指的目录，本例中为 C :/ Users / leo，创建.gdbinit 文件，内容为：set auto-load safe-path /。

开始调试的步骤如下：

- 启动 JLink Server，双击 JLink 驱动安装目录下，配置详见[图 6-2. 启动 JLINK GDB Server](#)所示，点击 OK 按钮。

- C :/ Program Files (x86) / SEGGER / JLink / JLinkGDBServer.exe

图 6-2. 启动 JLINK GDB Server



6.4.2. 启动GDB Client进行调试

启动 GDB server 完成之后，就可以使用 GDB client 进行调试了。额外启动 windows 命令窗口，进入 GD32W51x_RELEASE / NSPE / Project / WIFI_IOT / GCC 目录，输入命令 `gdb`，详见 [图 6-3. GDB 调试窗口](#) 所示：

- `arm-none-eabi-gdb output / bin / nspe.axf`
- 遇到提示回车即可，直到程序停在 `Reset_Handler`

图 6-3. GDB 调试窗口

```

C:\Windows\System32\cmd.exe - arm-none-eabi-gdb output\bin\nspe.axf
GNU gdb (GNU Arm Embedded Toolchain 9-2020-q2-update) 8.3.1.20191211-git
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=i686-m64-mingw32 --target=arm-none-eabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from output\bin\nspe.axf...
Reset_Handler () at C:\Program Files\GNU Arm Embedded Toolchain\bin\arm-none-eabi-gdb\bin\nspe.axf:25
25      movs r1, #0
Select auto target interface speed (4000 kHz)
Target endianess set to "little endian"
Resetting target
Flash breakpoints enabled
Flash download enabled
Loading section .vectors, size 0x1d8 1ma 0x300a200
Loading section .code_to_sram, size 0x1ad8 1ma 0x300a3d8
Loading section .text, size 0x7afc0 1ma 0x300beb0
Loading section .rodata, size 0x2c8b8 1ma 0x3008e70
Loading section .ARM, size 0x3 1ma 0x30b3728
Loading section .init_array, size 0x4 1ma 0x30b3730
Loading section .fini_array, size 0x4 1ma 0x30b3734
Loading section .data, size 0x600 1ma 0x30b3738
Start address 0x30b0ff4, load size 695096
Transfer rate: 226 KB/sec, 13901 bytes/write.
Breakpoint 1 at 0x30b0ff4: file C:\Program Files\GNU Arm Embedded Toolchain\bin\arm-none-eabi-gdb\bin\nspe.axf, line 25.
Breakpoint 2 at 0x3035594: file C:\Program Files\GNU Arm Embedded Toolchain\bin\arm-none-eabi-gdb\bin\nspe.axf, line 122.

Breakpoint 1, Reset_Handler () at C:\Program Files\GNU Arm Embedded Toolchain\bin\arm-none-eabi-gdb\bin\nspe.axf:25
25      movs r1, #0
(gdb)

```

这时，就可以开始使用 gdb 命令进行调试程序。以下是常用命令参考，如图 6-4. 常用命令所示，具体请参考官方文档：<https://www.gnu.org/software/gdb/documentation/>。

图 6-4. 常用命令

```

bt          打印调用栈信息
i r        打印寄存器信息
c          (继续) 执行代码到下一断点处
si        单条指令执行
ni        按指令执行到下一个断点
b lineNumber 在源文件某行设置断点，例如 b 5
b function  在函数入口处设置断点，例如 b PendSV_Handler
b *address  在指定内存地址处设置断点，例如 b *0x08005a1c
b          不带参数，表示在下一条指令设置断点，在调用某函数前使用该命令可以使函数执行返回后立即中断
i b        查看断点信息
d id       删除断点(根据i b查看到的ID号删除)，例如 d 3 删除3号断点
display /5i $pc 查看从PC开始的5条指令
x /5i $pc  查看从PC开始的5条指令
x /4xw address 查看内存信息，以16进制32位打印 address 处的内存信息
4表示打印次数，x表示16进制，w表示双字
disassemble 查看汇编代码
disassemble /m 查看汇编代码和源码
disassemble /r 查看16进制机器码
l          查看源代码
l function 查看某函数源代码，例如 l CORE_Starting
l lineNumber 从当前文件某行开始查看源代码，例如 l 1231

```

7. 常见问题

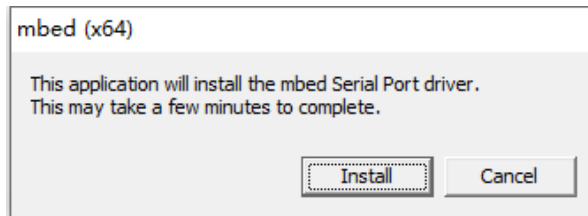
7.1. DAPLINK 盘识别

使用 START 开发板，如果串口不能被识别为“mbedSerialPort (COMx)”，如 Windows 7 及

以下版本，则需要安装 Mbed 串口驱动。

- 下载驱动
 - https://os.mbed.com/media/downloads/drivers/mbedWinSerial_16466.exe
- 安装
 - 双击运行 mbedWinSerial_16466.exe，点击“Install”，如[图 7-1. Mbed 串口驱动安装](#)，安装成功后，点击“Finish”。

图 7-1. Mbed 串口驱动安装



- 检查
 - 安装成功后，设备管理器能看到串口“mbed Serial Port (COMx)”，如[图 7-2. 设备管理器串口列表](#)。设备管理器串口列表，设备和驱动器列表能看到“DAPLINK”盘，如[图 3-6. 设备和驱动器列表](#)。

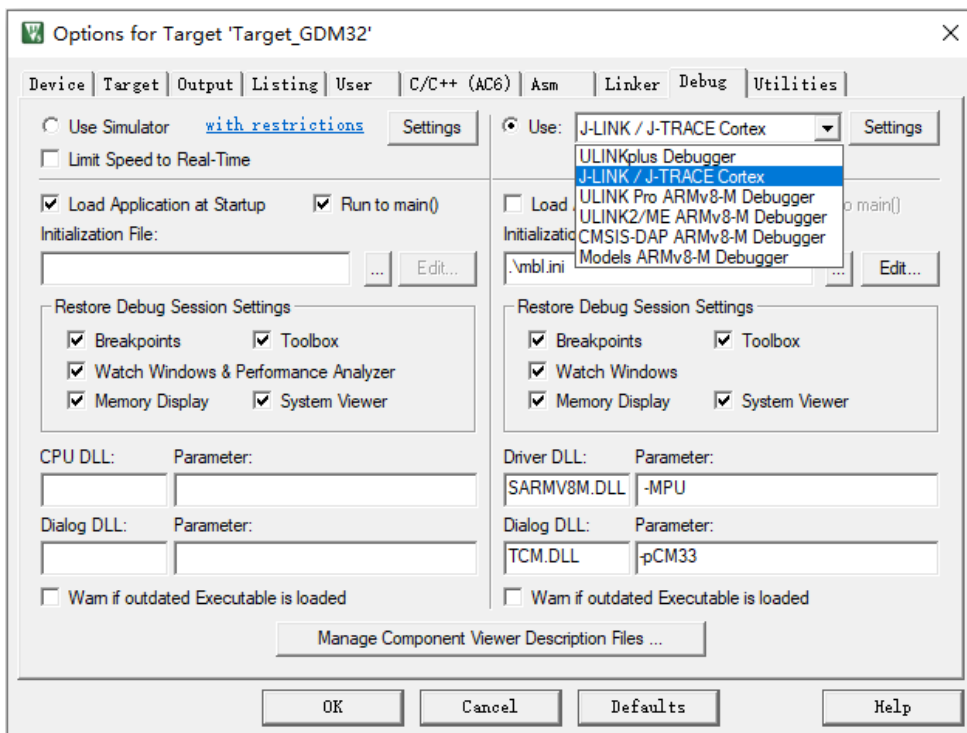
图 7-2. 设备管理器串口列表



7.2. KEIL JLINK 调试 CM33

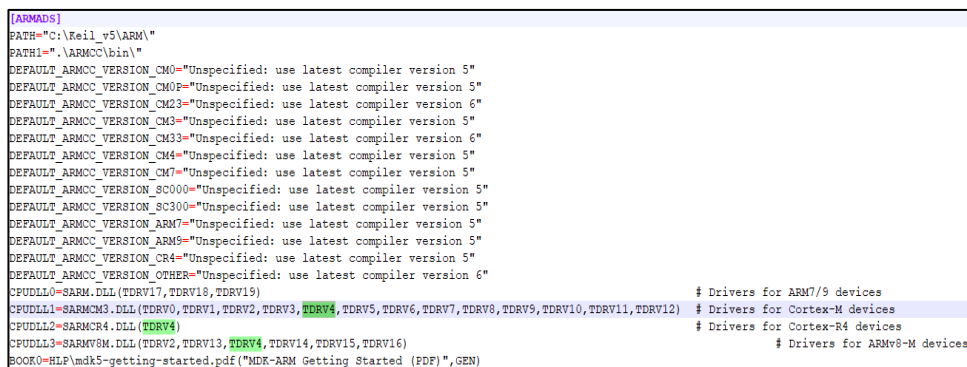
默认情况下，MDK5.25 不支持使用 JLINK 调试 ARM CM33。当需要使用 JLINK 调试 ARM CM33 时，如果在 KEIL 工具工程选项的 Debug 页面中，见[图 7-3. KEILDEBUG 选项](#)，在选择调试器一栏中，找不到选择 JLINK 调试器的选项，则需要修改 C:/Keil_v5/TOOLS.INI 文件。

图 7-3. KEILDEBUG 选项



在 TOOLS.INI 文件中[ARMADS]的 CPUDLL3 行添加对应于 J-LINK 的 TDRVx，以增加对 JLINK 的支持。采用 J-LINK 调试开发板需要添加 TDRV4，根据图 7-4. 添加 KEIL 支持的 TOOL 所示，在对应位置增加“TDRV4”即可。修改完成之后，重新打开 KEIL 工具，就可选择“J-LINK / J-TRACE Cortex”调试 ARM CM33 了。

图 7-4. 添加 KEIL 支持的 TOOL



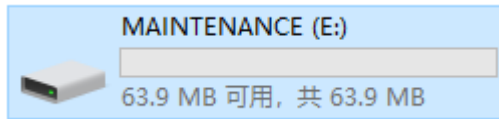
7.3. DAP 芯片固件升级

如果需要对 START 开发板上的 DAP 芯片进行固件升级，可以按住开发板上的 Reset 按钮，然后插入 USB 线上电，此时就会看到 MAINTENANCE 盘，如图 7-5. MAINTENANCE 盘。

此时将 gd32w51x_if_crc_qspi.bin（适用于芯片 GD32W51x EXT FLASH）或者 gd32w51x_if_crc_fmc.bin（适用于芯片 GD32W51x SIP FLASH）拷贝进去，成功后就会运行

新的 DAP 固件，出现 DAPLINK 盘。

图 7-5. MAINTENANCE 盘



7.4. No image 错误

打印 ERR: No image to boot (ret = -5) 。

原因：前一次引导 WIFI_IOT 出错，MBL 记录该 IMAGE 运行异常，如果另一个 IMAGE 未烧录或者也发生过引导异常，则会打印此消息。也就是说 MBL 认为没有可跳转的合法 IMAGE，引导失败。

解决方法：再烧录一遍 MBL 即可，烧录后 IMAGE 状态会清空。

7.5. 代码跑在 SRAM

如果有程序需要更快速运行，以便达到更高的性能，可以考虑将它们移动到 SRAM 里面运行。

■ 使用 KEIL

可以打开 GD32W51x_RELEASE / NSPE / Project / WIFI_IOT / KEIL / Project.sct 编辑，在 RW_IRAM2 这个 Execution Region 中，将需要添加的文件或函数放进去。例如：

```
port.o (+RO)
```

是将整个 port.c 文件放进 SRAM 运行。例如：

```
tasks.o (.text.xTaskIncrementTick)
```

是将 tasks.c 中的 xTaskIncrementTick () 函数放进 SRAM 运行。

如果发现函数未被放入，可以查看下 GD32W51x_RELEASE / NSPE / Project / WIFI_IOT / KEIL / wifi_iot / Freertos / List / nspe.map，根据函数名找到正确的 symbol 名，如 [图 7-6. NSPE MAP 文件](#) 所示。

图 7-6. NSPE MAP 文件

soc_rx_tasklet	0x20025049	Thumb Code	1770	soc_rx.o(i.soc_rx_tasklet)
soc_send_xframe	0x2002577d	Thumb Code	458	soc_tx.o(i.soc_send_xframe)
wlan_interrupt_rx_handler	0x20025a25	Thumb Code	144	soc_isr.o(i.wlan_interrupt_rx_handler)

■ 使用 IAR

如果需要将一些函数放在 SRAM 运行，可以在函数尾部加上“SECTION_RAM_CODE”，例如：


```
BaseType_t xTaskIncrementTick(void) SECTION_RAM_CODE
```

■ 使用 GCC

可以打开 GD32W51x_RELEASE/NSPE/Project/WIFI_IOT/GCC/nspe_gdm32_ns.ld, 找到 “.code_to_sram:” 这一行, 这段大括号中包含的代码都是运行在 SRAM 的。如果需要添加新内容, 可以加在最后面。格式参照已有的文件, 例如:

```
KEEP (*port.o* (.text* .rodata*))
```

是将整个 port.c 文件放进 SRAM 运行。例如:

```
KEEP (*tasks.o* (.text.xTaskIncrementTick))
```

是将 tasks.c 中的 xTaskIncrementTick () 函数放进 SRAM 运行。

8. 版本历史

表 8-1. 版本历史

版本号	说明	日期
1.0	初稿发布	2021 年 11 月 23 日
1.1	简洁化 SDK 配置和使用流程，增强功能放在 TrustZone 使能的版本中使用。	2022 年 03 月 25 日

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the "Product"), is owned by the Company under the intellectual property laws and treaties of the People's Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as its 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 its 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.