GigaDevice Semiconductor Inc.

GD32VW553 快速开发指南

应用笔记

AN154

1.3a 版本

（2025 年 09 月）

目录

图索引

表索引

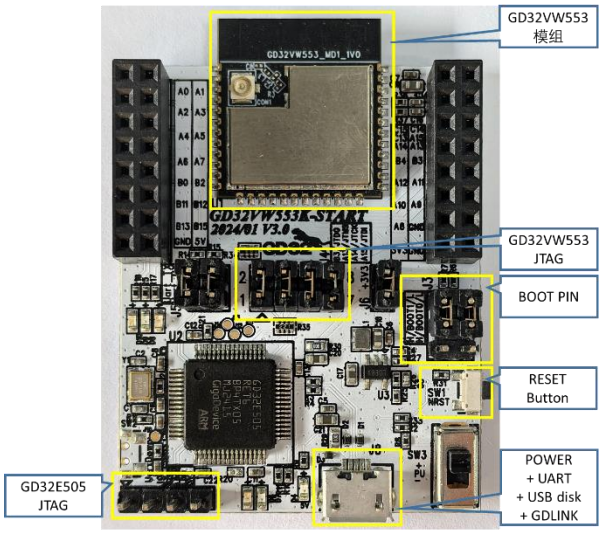
1. 认识开发板

1.1. 开发板实物图

1.1.1. START开发板

START 开发板由底板和模组组成，模组搭载了 GD32VW55x WiFi+BLE 芯片。

图 1-1. START 开发板实物图

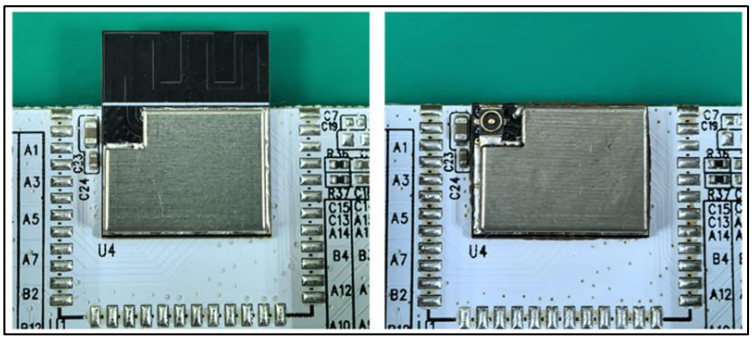


主要关注开发板的以下几个部分，已在*图 1-1. START 开发板实物图*中标注出来。

- 启动模式（Boot PIN）；
- 供电口（POWER）；
- 查看日志（UART）；
- 调试器接口（JLink 或 GDLink）；
- 重启（Reset Button）。

START V4.0、V4.1 开发板模组支持 GD32VW553_MD1、GD32VW553_MD2、GD32VW553-MINI-I 与 GD32VW553-MINI-E 四款无线模组，上述四款模组中，除 GD32VW553_MD1 模组无线主芯片为 QFN40 封装外，其余模组无线主芯片均为 QFN32 封装。*图 1-1. START 开发板实物图*中所示的模组型号为 GD32VW553_MD1，其与 GD32VW553_MD2 的封装、尺寸完全一致。GD32VW553-MINI-I 模组与 GD32VW553-MINI-E 模组如*图 1-2. GD32VW553-MINI-I 模组（左）与 GD32VW553-MINI-E 模组（右）*所示。

图 1-2. GD32VW553-MINI-I 模组（左）与 GD32VW553-MINI-E 模组（右）



START 开发板连接件与开关功能见*表 1-1. START 开发板连接件与开关功能*。

表 1-1. START 开发板连接件与开关功能

接口	描述
J1	连接至主芯片 PA0~PA7 / PB0 / PB15 等 GPIO 管脚的接口，以及+5V 接口、GND 接口。
J2	连接至主芯片 PA12~PA15 / PB3 / PB4 / PC13~PC15 等 GPIO 管脚的接口，以及模组 +3.3V 供电测试接口、GND 接口。
J3	连接至主芯片 PC8(BOOT0)与 PB1(BOOT1)管脚，Boot 模式选择时需进行相应配置。默认 Boot0 / 1 使用短路跳帽下拉，即芯片默认从 Sip flash 启动。
J4	连接至主芯片 PB3(JTDO) / PA13(JTMS) / PA14(JTCK) / PA15(JTDI) JTAG 管脚的接口，以及与 GDLINK 芯片 L_TDO / L_TMS / L_TCK / L_TDI 等 JTAG 管脚相连的接口。默认主芯片和 GDLINK 芯片的上述管脚通过跳帽相连，可通过 GDLINK 芯片给主芯片烧录固件。
J5	连接至主芯片 PA6(UART2_TX)与 PA7(UART2_RX) UART T / RX 管脚的接口，以及与 GDLINK 芯片 L_UART_RX 与 L_UART_TX UART R / TX 管脚相连的接口。默认主芯片和 GDLINK 芯片的上述管脚通过跳帽相连，可由 GDLINK 芯片和主芯片做串口通信。
J6	GD32VW553 模组 3.3V 供电连接口，默认使用短路帽连接。功耗测试时可直接向 J6.2 外接 3.3V 电源。
J8	USB Type-C 接口，默认的串口通信与+5V 供电接口。
J9	连接至 GDLINK 芯片 L_SWDIO / L_SWDCK / L_NRST 等 SWD 管脚的接口，以及 GDLINK +3.3V 供电测试接口与 GND 接口
SW1	连接模组经 1KR 上拉至 3.3V 的 NRST 管脚与 GND。按下再松开此开关，可 Reset 主芯片。
SW2	连接 GDLINK 芯片经 1KR 上拉至 3.3V 的 UartDownload 管脚与 GND，按住此开关，经 USB 数据线连接 START 开发板与 PC，再松开开关，可复制 / 粘贴烧录 GDLINK 芯片固件。
SW3	连接模组经 1KR 串联的 PU 管脚与+3.3V 供电（或 GND）。向上拨码，主芯片使能；向下拨码，主芯片掉电。 对于在使用 GD32VW553-MINI-I / E 模组的 START 开发板上无此开关。

START 开发板主要接口说明见*表 1-2. START 开发板主要接口说明*。对于 V3.0 以及更早版本的 START 开发板，连接开发板红绿蓝三色 LED 的 GPIO 分别是 PB11/12/13，因此使用 MD2 模组的 START 开发板无法点亮这三个 LED。但新版 V4.0、V4.1START 开发板已经将连接红

绿蓝三色 LED 的 GPIO 改为了 PB0/PA12/PB4，此时 MD2、MINI 模组均可以点亮这三个 LED。

更多关于 START 开发板的说明，可以参考 GD32 官网 ***GD32VW553K-START Demo Suites*** 页面下载附件中的内容。

表 1-2. START 开发板主要接口说明

接口	描述
PA0	IO 口，可由用户自行配置。
PA1	IO 口，可由用户自行配置。
PA2	IO 口，可由用户自行配置。
PA3	IO 口，可由用户自行配置。
PA4	IO 口，可由用户自行配置。
PA5	IO 口，可由用户自行配置。
PA6 / UART2_TX	IO 口；UART TX。
PA7 / UART2_RX	IO 口；UART RX。
PB0	IO 口，可由用户自行配置。
PB1 / BOOT1	IO 口，Boot 模式选择。
PB2	IO 口，可由用户自行配置。 **对于采用 MD2 或 MINI-I / E 模组的 START 开发板，此接口无效。**
GND	参考地
PB11	IO 口，可由用户自行配置。 **对于采用 MD2 或 MINI-I / E 模组的 START 开发板，此接口无效。**
PB12	IO 口，可由用户自行配置。 **对于采用 MD2 或 MINI-I / E 模组的 START 开发板，此接口无效。**
PB13	IO 口，可由用户自行配置。 **对于采用 MD2 或 MINI-I / E 模组的 START 开发板，此接口无效。**
PB15	IO 口，可由用户自行配置。
PA8	IO 口，可由用户自行配置。
PA9	IO 口，可由用户自行配置。 **对于采用 MD2 或 MINI-I / E 模组的 START 开发板，此接口无效。**
PA10	IO 口，可由用户自行配置。 **对于采用 MD2 或 MINI-I / E 模组的 START 开发板，此接口无效。**
PA11	IO 口，可由用户自行配置。 **对于采用 MD2 或 MINI-I / E 模组的 START 开发板，此接口无效。**
PA12	IO 口，可由用户自行配置。
PB3 / JTDO	IO 口；JTDO 脚。
PB4 / JNTRST	IO 口；JNTRST 脚。
PA13 / JTMS	IO 口；JTMS 脚。
PA14 / JTCK	IO 口；JTCK 脚。
PC8 / BOOT0	IO 口；Boot 模式选择。
PA15 / JTDI	IO 口；JTDI 脚。

接口	描述
PC13	IO 口，可由用户自行配置。 **对于采用 MINI-I / E 模组的 START 开发板，此接口无效。**
PC14	IO 口，可由用户自行配置。
PC15	IO 口，可由用户自行配置。
NRST	模块使能脚，接 3.3V 后模块方可正常使用。
PU	模块复位脚，接 3.3V 后模块方可正常使用。 **对于采用 MINI-I / E 模组的 START 开发板，此接口无效。**
3V3	3.3V 供电电源
GND	参考地

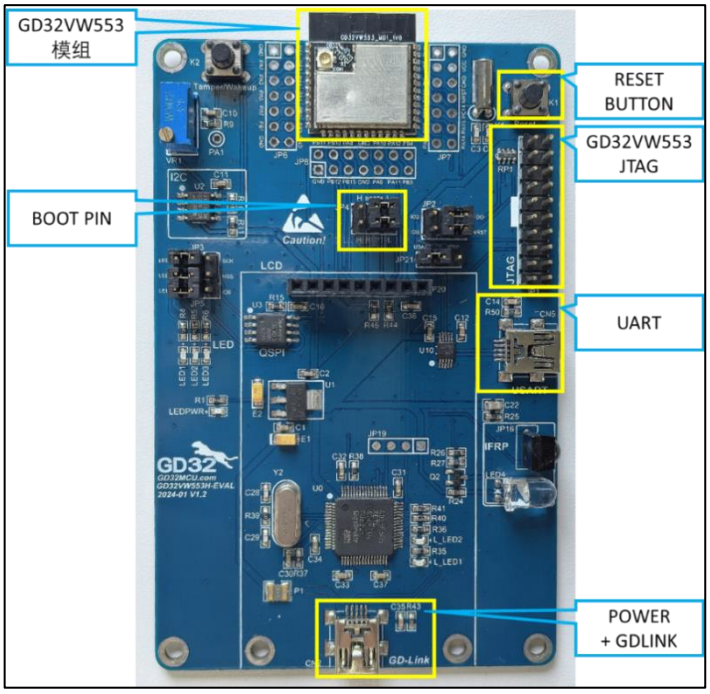
1.1.2. EVAL开发板

EVAL 开发板由底板和模组组成，模组搭载了 GD32VW55x WiFi+BLE 芯片，底板引出了众多外设测试口，例如：I2C，IFRP，ADC 等。

对开发者来说，主要关注开发板的以下几个部分，已在*图 1-3. EVAL 开发板实物图*中标注出来。

- 启动模式（Boot PIN）；
- 供电口（POWER）；
- 查看日志（UART）；
- 调试器接口（JLink 或 GDLink）；
- 重启（Reset Button）。

图 1-3. EVAL 开发板实物图



更多关于 EVAL 开发板的说明，可以参考 GD32 官网 *GD32VW553 Demo Suites* 页面下载中

附件内容。

对于 START 和 EVAL 开发板，SDK 配置不同，需要选择不同的宏使能。如*图 1-4. 开发板类型配置*所示，SDK 默认选择 START 开发板配置。配置文件为 GD32VW55x_RELEASE/config/platform_def.h。

图 1-4. 开发板类型配置

```
// board type
#define PLATFORM_BOARD_32VW55X_START    0
#define PLATFORM_BOARD_32VW55X_EVAL     1
#define PLATFORM_BOARD_32VW55X_F527     2
#ifdef CONFIG_PLATFORM_ASIC
#define CONFIG_BOARD                    PLATFORM_BOARD_32VW55X_START
#endif
```

1.2. 启动模式

GD32VW55x 可以选择从 ROM 启动，FLASH 启动或者 SRAM 启动。

开发板 BOOT SWD 框内的 BOOT0 和 BOOT1 两根引脚的高低选择决定了启动模式，见*表 1-3. 启动模式*。更多关于启动模式的说明请参考文档《GD32VW55x_User_Manual》。

表 1-3. 启动模式

EFBOOTLK	BOOT0	BOOT1	EFSB	启动地址	启动区域
0	0	-	0	0x08000000	SIP Flash
0	0	-	1	0x0BF46000	secure boot
0	1	0	-	0x0BF40000	Bootloader / ROM
0	1	1	-	0x20000000	SRAM
1	0	-	0	0x08000000	SIP Flash
1	0	-	1	0x0BF46000	Secure boot
1	1	-	-	0x0BF40000	Bootloader / ROM

1.3. 调试器接口

对于 START 开发板，开发板自带 GDLink（GD32E505），可以搭配 OpenOCD 使用，也可外接 GDLink 调试器或者 JLink 调试器进行调试和下载。在 GD32E505 芯片内还集成了 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。将引脚 JTCK、JTMS、JTDO 和 JTDI 与开发板中部四引脚通过跳线帽连接，即可通过 GDLink 下载和调试代码。*图 1-1. START 开发板实物图*中展示的是透过 GDLink 进行调试。

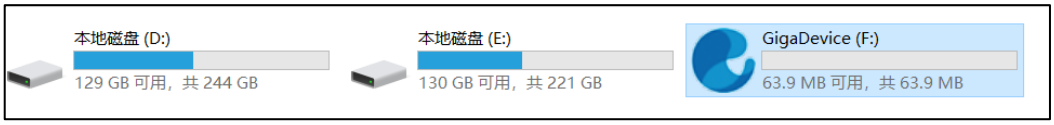
对于 EVAL 开发板，也可使用 GDLink 调试器或者 JLink 调试器进行调试和下载。

此外需要注意的是，GD32VW55x 支持 cJTAG 和 JTAG，不支持 SWD 的调试接口。

1.4. 下载接口

对于 START 开发板，除了上一节提到的通过 GDLink 调试器或者 JLink 调试器进行固件下载外，如果不需要调试功能，仅需要下载固件，还可以使用 U 盘拷贝的方式下载。将开发板通过 USB 线插入电脑，如*图 1-5. 设备和驱动器列表*中所示的 GigaDevice 盘。将 image-all.bin 文件（见后续章节）拷贝进 GigaDevice 盘，就能完成对 GD32VW55x 芯片的 FLASH 烧写。

图 1-5. 设备和驱动器列表

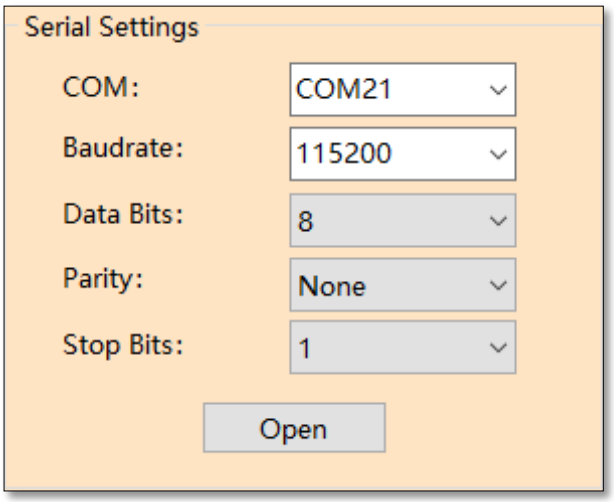


对于 EVAL 开发板，支持使用 GDLink 调试器或者 JLink 调试器进行下载，不支持 U 盘拷贝。

1.5. 查看日志

使用 MicroUSB 线连接 START 开发板，PC 端使用串口工具并根据*图 1-6. 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 1-6. 串口配置



2. 搭建开发环境

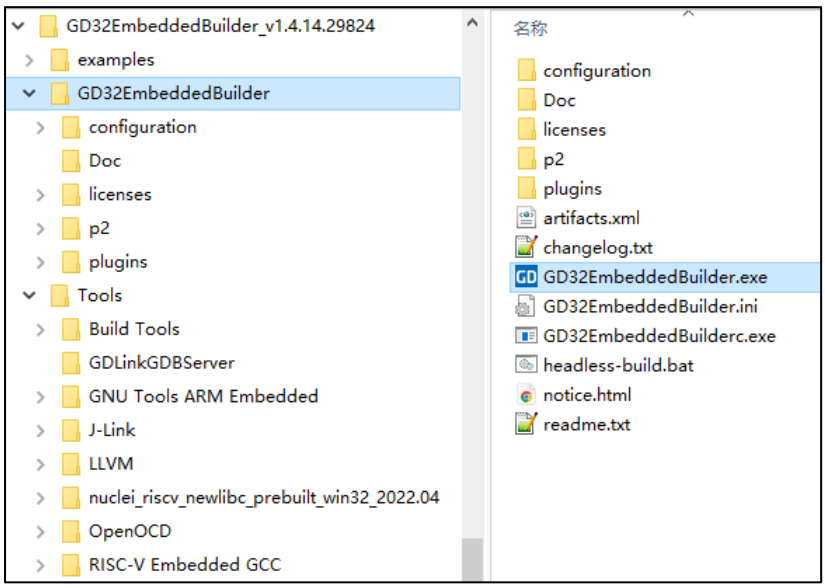
在编译和烧录固件之前，需要搭建开发环境。

目前可使用的开发工具是 GD32 Embedded Builder 和 SEGGER Embedded Studio IDE。

2.1. GD32 Embedded Builder 安装

GD32 Embedded Builder 可在地址 https://gd32mcu.com/cn/download 选择 GD32VW5 下载，解压后的路径如*图 2-1 GD32 Embedded Builder 目录结构*所示，其中 build tool、tool chain、openocd、jlink 等都已经放在 Tools 目录下。

图 2-1 GD32 Embedded Builder 目录结构



2.2. SEGGER Embedded Studio IDE 安装

请访问 https://wiki.segger.com/GD32V 获取 SEGGER Embedded Studio IDE 和 License Activation Key。

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，以及如何正确配置 SDK。

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（Main Bootloader），一个是 MSDK（Main SDK）。它们最终都将被烧写到 FLASH 运行。上电之后，程序将从 MBL 的 Reset_Handler 启动，然后跳转到 MSDK 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. 无线模块配置

配置文件为 GD32VW55x_RELEASE/config/platform_def.h，主要内容见*图 3-2. 无线模块配置*。

图 3-2. 无线模块配置

```
#define CFG_WLAN_SUPPORT
#define CFG_BLE_SUPPORT
#if defined(CFG_WLAN_SUPPORT) && defined(CFG_BLE_SUPPORT)
  #define CFG_COEX
#endif
```

- 如果是 BLE/WiFi combo 模式，请打开：
  - #define CFG_WLAN_SUPPORT
  - #define CFG_BLE_SUPPORT
- 如果是 BLE only，请只打开：
  - #define CFG_BLE_SUPPORT
- 如果是 WiFi only，请只打开：
  - #define CFG_WLAN_SUPPORT
- 如果关闭无线模块，请全部关闭。

3.2.2. SRAM布局

SRAM layout 配置文件为 GD32VW55x_RELEASE\config\config_gdm32.h。修改*图 3-3. SRAM 布局*中宏定义值，可以对可执行程序段 MBL 及 IMG 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。

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       0x300           /* !Keep unchanged! */
#define RE_IMG_DATA_START       0x200           /* !Keep unchanged! */
```

每个可执行程序段内部的 SRAM 空间规划可以参考对应工程下的.ld 文件，如 MBL\project\eclipse\mbl.ld 和 MSDK\plf\riscv\env\gd32vw55x.ld。

3.2.3. FLASH布局

Flash layout 配置文件为 GD32VW55x_RELEASE\config\config_gdm32.h。修改*图 3-4. FLASH 布局*中宏定义值，可以对可执行程序段 MBL 及 MSDK 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。

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_SYS_SET_OFFSET       0x0             /* !Keep unchanged! */
#define RE_MBL_OFFSET           0x0             /* 0x0: Boot from MBL, 0x1000: Boot from ROM */
#define RE_SYS_STATUS_OFFSET    0x8000          /* !Keep unchanged! */
#define RE_IMG_0_OFFSET         0xA000          /* !Keep unchanged! */
#define RE_IMG_1_OFFSET         0x1EA000
#define RE_IMG_1_END            0x3CA000        /* reserved 196KB for user data */
#define RE_NVDS_DATA_OFFSET     0x3FB000        /* reserved 20KB for nvds data */
#define RE_END_OFFSET           0x400000        /* equal to flash total size */
```

每个可执行程序段内部的 FLASH 空间规划可以参考对应工程下的.ld 文件，如 MBL\project\eclipse\mbl.ld 和 MSDK\plf\riscv\env\gd32vw55x.ld。

3.2.4. 固件版本号

配置文件为 GD32VW55x_RELEASE\config\config_gdm32.h。修改*图 3-5. 固件版本号*中宏定义值，可以指定版本号。此外 RE_IMG_VERSION 在 Secure Boot 中会用于固件版本的判定。

MBL 只能本地升级，IMG 可以支持在线升级，SDK 发布的版本号与 RE_IMG_VERSION 保持一致。

图 3-5. 固件版本号

```
/* FW_VERSION */
#define RE_MBL_VERSION          0x01000003
#define RE_IMG_VERSION          0x01000003
```

3.2.5. APP配置

配置文件为 GD32VW55x_RELEASE\MSDK\app\app_cfg.h。可以选择是否打开一些应用，例如：ATCMD，阿里云，MQTT，COAP 等。

通过修改 app_cfg.h 内的宏 CONFIG_BLE_LIB 可切换 BLE library。将 CONFIG_BLE_LIB 配置为 BLE_LIB_MIN（如*图 3-6 BLE library 选择*所示），工程编译时会选择 libble.a，同时头文件会 include ble_config_min.h;将 CONFIG_BLE_LIB 配置为 BLE_LIB_MAX，工程编译时会选择 libble_max.a，同时头文件会 include ble_config_max.h。

图 3-6 BLE library 选择

```
#define BLE_LIB_MIN                     0       //only peripharal and server
#define BLE_LIB_MAX                     1       //add central and client usage

#define CONFIG_BLE_LIB                  BLE_LIB_MIN
```

libble.a 支持的功能如下：

1. 支持 peripheral
2. 支持一条链路连接
3. 支持 server
4. 支持 host 和 controller
5. 支持 EATT
6. 支持微信小程序 wifi 配网

libble_max.a 在 libble.a 的基础上，额外支持的功能如下：

1. 支持 central
2. 支持四条链路
3. 支持 client
4. 支持周期广播
5. 支持 phy update
6. 支持 power control
7. 支持 ble ping
8. 支持 secure connection

3.2.6. 工程配置

主工程-MSDK，支持多种 Configurations，默认选择 msdk，还有 msdk_ffd、msdk_mbedtls_2.17.0、msdk_rtthread、msdk_threadx 可选。

msdk_ffd 与 msdk 的差别主要是工程包含的 WiFi 连接管理库不同。msdk 包含 libwpas，它更精炼、更少占用内存资源。而 msdk_ffd 包含 wpa_supplicant，它功能更全更通用，但代码比较大，而且占用内存资源比较多。另外，msdk_ffd 会默认包含 libble_max.a，开启更多 BLE 功能。当然 msdk 可以通过修改配置来实现 libble.a 和 libble_max.a 的切换。

msdk_mbedtls_2.17.0 与 msdk 的差别主要是工程包含的 MbedTLS 库版本不同。msdk 包含 MbedTLS 3.6.2，而 msdk_mbedtls_2.17.0 包含 MbedTLS 2.17.0。MbedTLS 3.6.2 运行在 FLASH 中，而 MbedTLS 2.17.0 大部分内容运行在 ROM 中。如果是对安全有严格要求，建议选择 msdk。如果 FLASH 空间紧张，建议选择 msdk_mbedtls_2.17.0。

msdk_rtthread 与 msdk 的主要差别是工程使用的 RTOS 不同。msdk 使用 FreeRTOS，msdk_rtthread 使用 rt-thread.

msdk_threadx 与 msdk 的主要差别也是工程使用的 RTOS 不同。msdk 使用 FreeRTOS，msdk_threadx 使用 threadx.

- msdk

  FreeRTOS + Libwpas.a + libble.a + MbesTLS 3.6.2

- msdk_ffd

  FreeRTOS + wpa_supplicant + libble_max.a + MbesTLS 3.6.2

- msdk_mbedtls_2.17.0

  FreeRTOS + Libwpas.a + libble.a + ROM MbesTLS 2.17.0

- msdk_rtthread

  RTThread + Libwpas.a + libble.a + MbesTLS 3.6.2

- msdk_threadx

  Threadx + Libwpas.a + libble.a + MbesTLS 3.6.2

在实际使用时，如何进行 Configurations 选择详见 *4.2 编译*小节中编译 MSDK 工程部分。

3.3. 正确日志示例

在固件组（MBL+MSDK）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 *6 常见问题*，看能否找到帮助。

图 3-7. 工程启动信息

```
ALW: MBL: First print.
ALW: MBL: Boot from Image 0.
ALW: MBL: Validate Image 0 OK.
ALW: MBL: Jump to Main Image (0x0800a000).
=== RF initialization finished ===
SDK Version: v1.0.3a-86d78d058d779fad
Build date: 2025/05/14 16:29:11
=== WiFi calibration done ===
=== PHY initialization finished ===
BLE local addr: AB:89:67:45:23:01, type 0x0
=== BLE Adapter enable complete ===
```

4. GD32 Embedded Builder IDE 工程

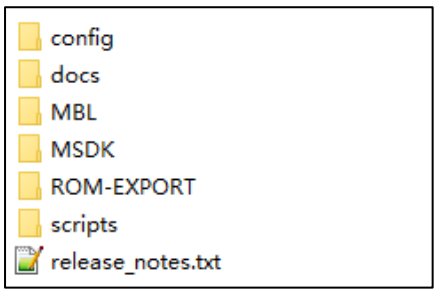
本章将介绍如何在 Embedded Builder IDE 下编译和调试 SDK。

工程组由 MBL 和 MSDK 两个工程组成。MSDK 包含 WiFi 和 BLE 协议栈、外设驱动及应用程序等，MBL 主要负责从两个 MSDK 固件（当前固件和 OTA 固件）中选择正确的去运行。

4.1. 打开工程组

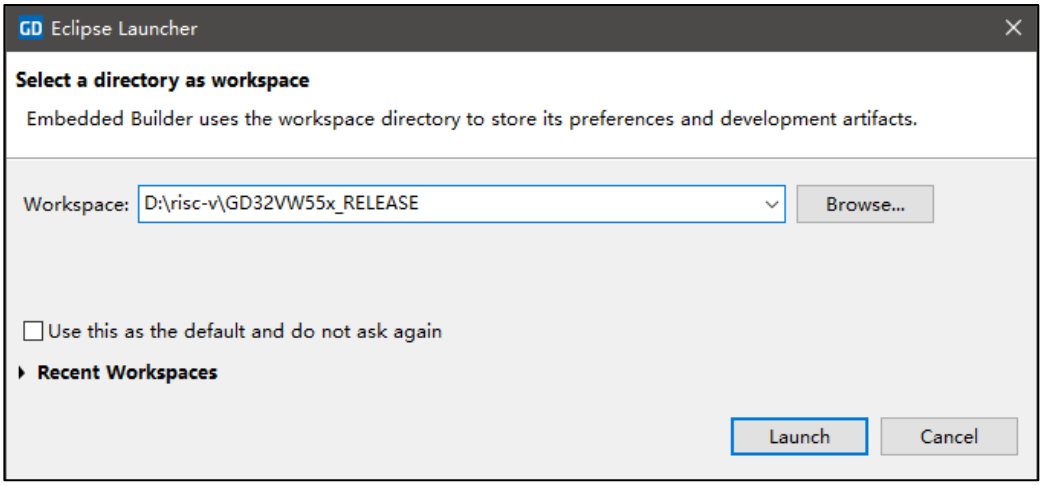
- 检查 SDK 目录 GD32VW55x_RELEASE，如*图 4-1. SDK 目录*所示。

图 4-1. SDK 目录



启动 IDE，双击 EmbeddedBuilder 目录下的 Embedded Builder.exe,并选择 SDK 目录 GD32VW55x_RELEASE 为 workspace，点击 launch 按钮，如*图 4-2. 启动 Embedded Builder*所示。

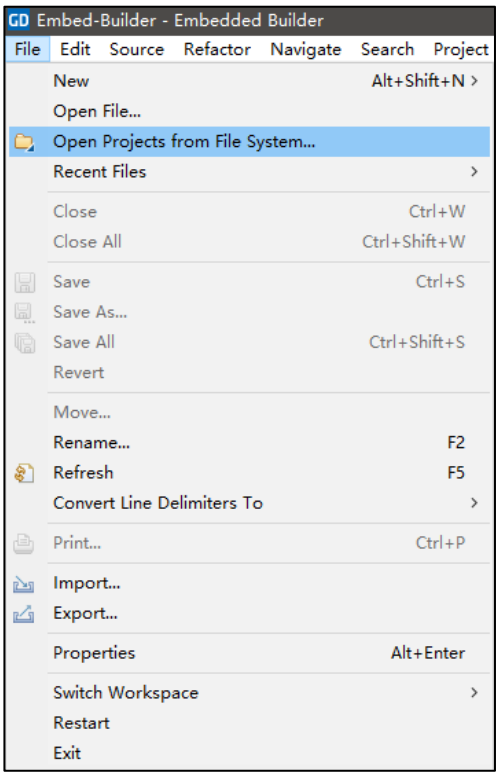
图 4-2. 启动 Embedded Builder



- 导入 MBL 工程

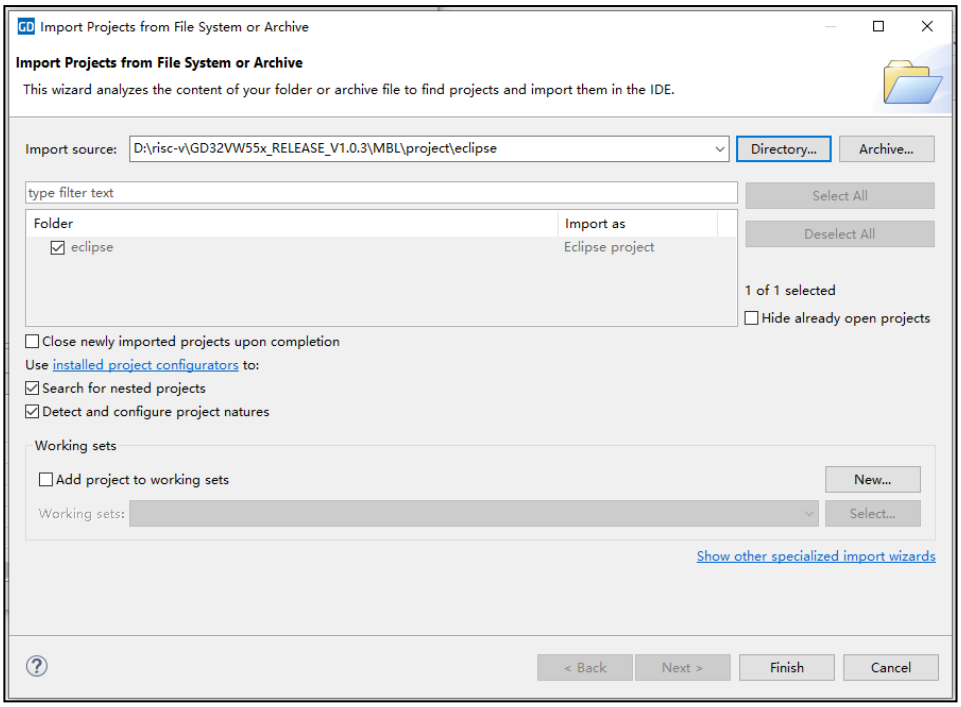
File 菜单点击 Open Projects from file System，如*图 4-3. Open Projects from file System*所示。

图 4-3. Open Projects from file System



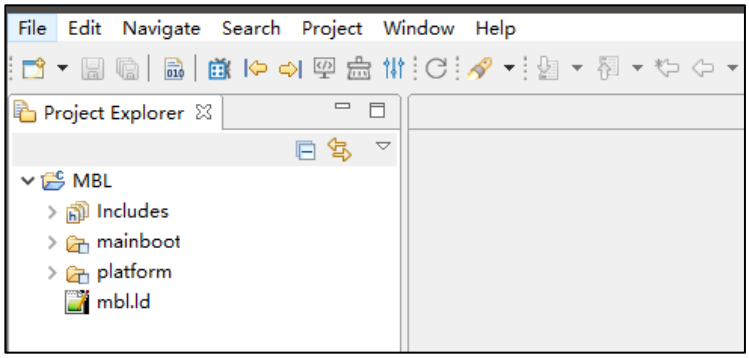
选择工程路径 GD32VW55x_RELEASE\MBL\project\eclipse，如*图 4-4. 选择 MBL 工程路径*所示，并点击 finish。

图 4-4. 选择 MBL 工程路径



关闭 welcome 界面就可以看到 MBL 工程，如*图 4-5. MBL 工程界面*所示。

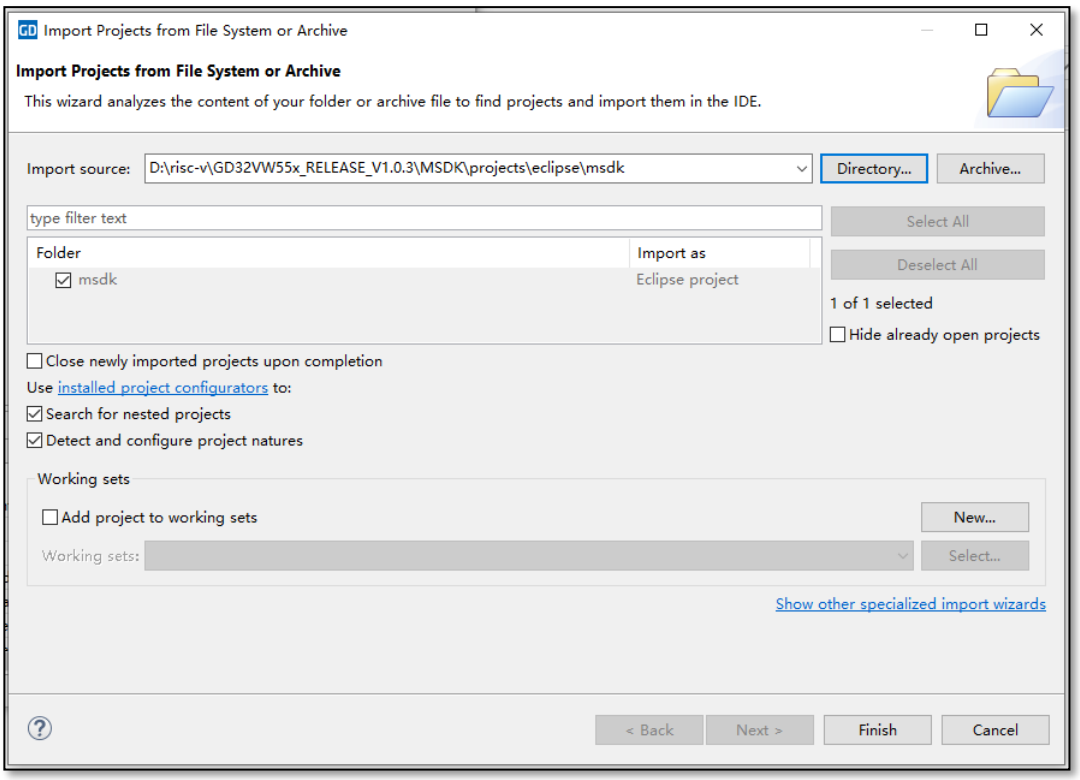
图 4-5. MBL 工程界面



■ 导入 MSDK 工程

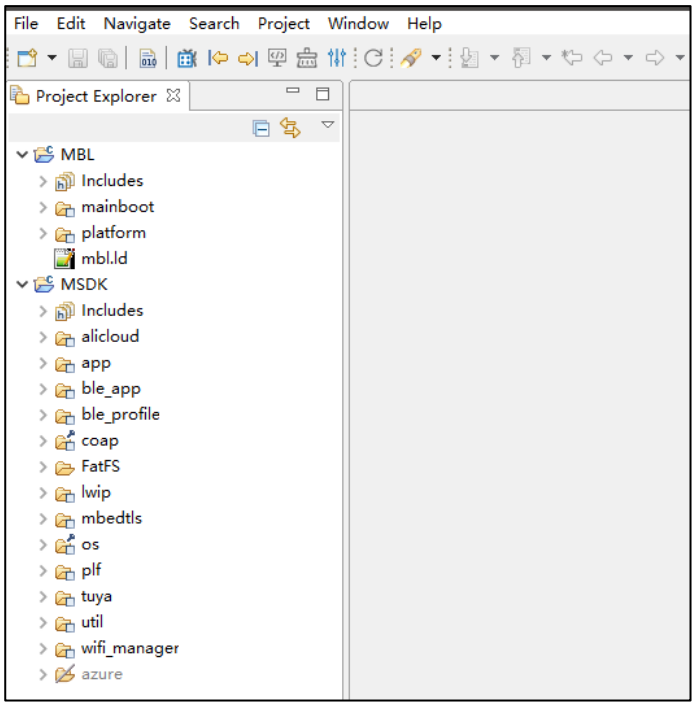
File 菜单点击 Open Projects from file System，工程路径选择 GD32VW55x_RELEASE\MSDK\projects\eclipse\msdk，如*图 4-6. 选择 MSDK 工程路径*所示，并点击 finish。

图 4-6. 选择 MSDK 工程路径



查看 MSDK 和 MBL 工程界面，如*图 4-7. MSDK 和 MBL 工程界面*所示。

图 4-7. MSDK 和 MBL 工程界面

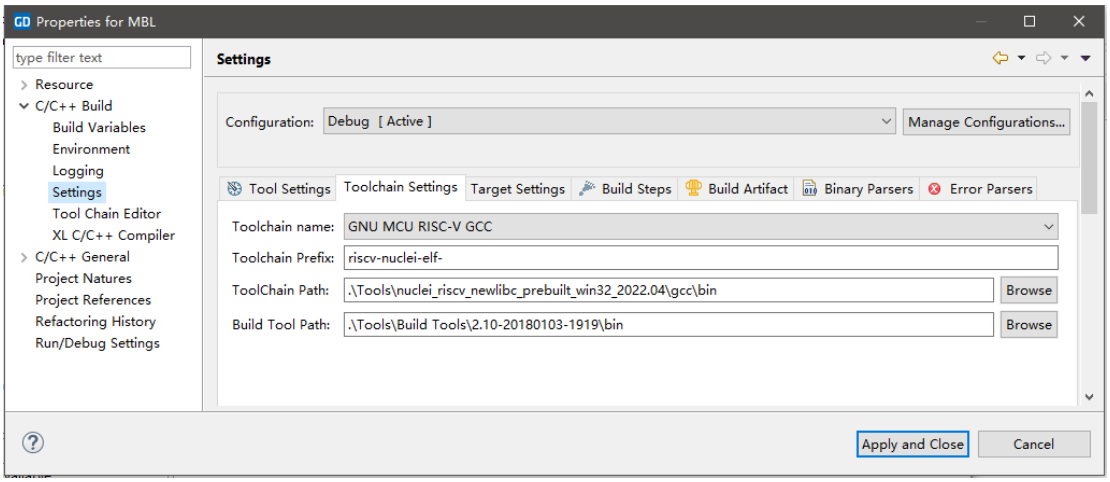


4.2. 编译

■ 检查工程编译工具配置

右击工程，点击 properties，依次选择 C/C++ Build →Settings，选项卡点击 toolchain settings，如*图 4-8. 工具链配置*所示。

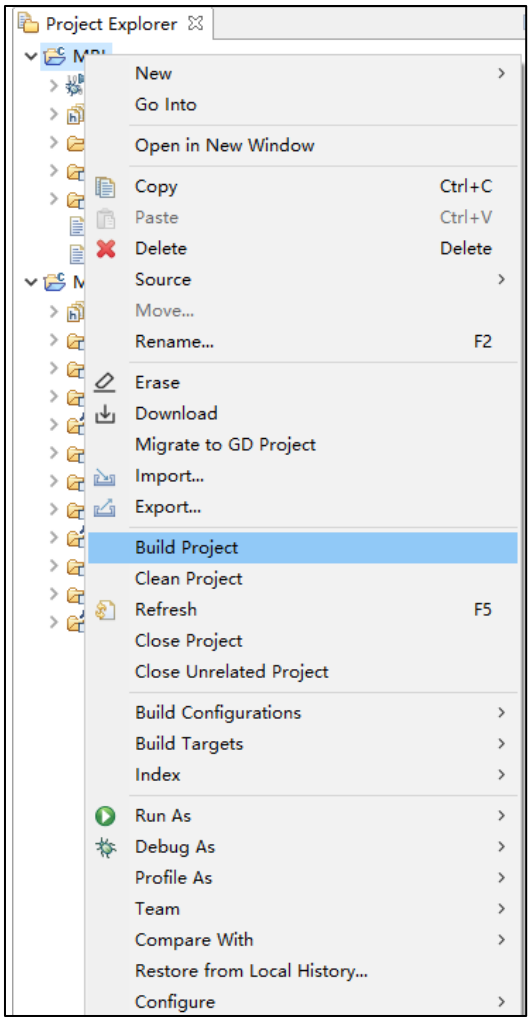
图 4-8. 工具链配置



■ 编译 MBL 工程

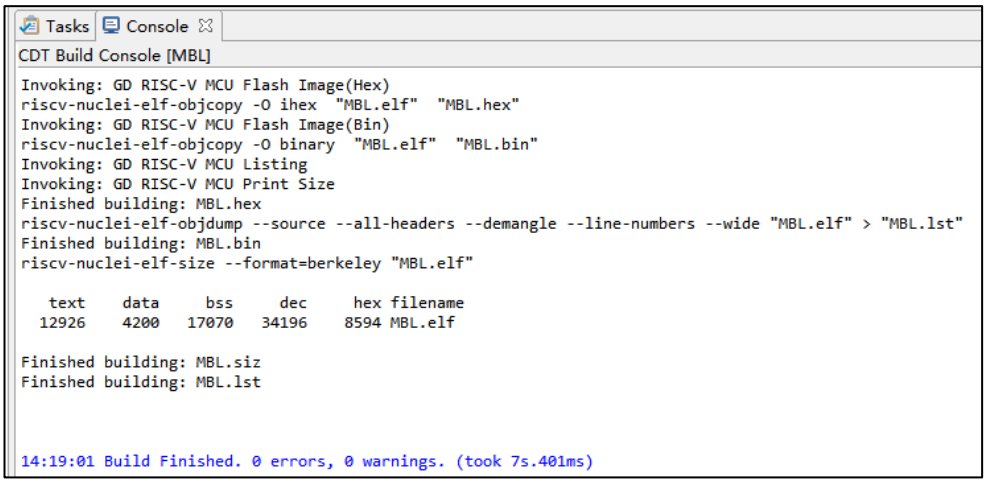
右击工程，点击 build project，如*图 4-9. 编译 MBL 工程*所示。

图 4-9. 编译 MBL 工程



编译结果，如*图 4-10. 编译 MBL 结果*所示。

图 4-10. 编译 MBL 结果

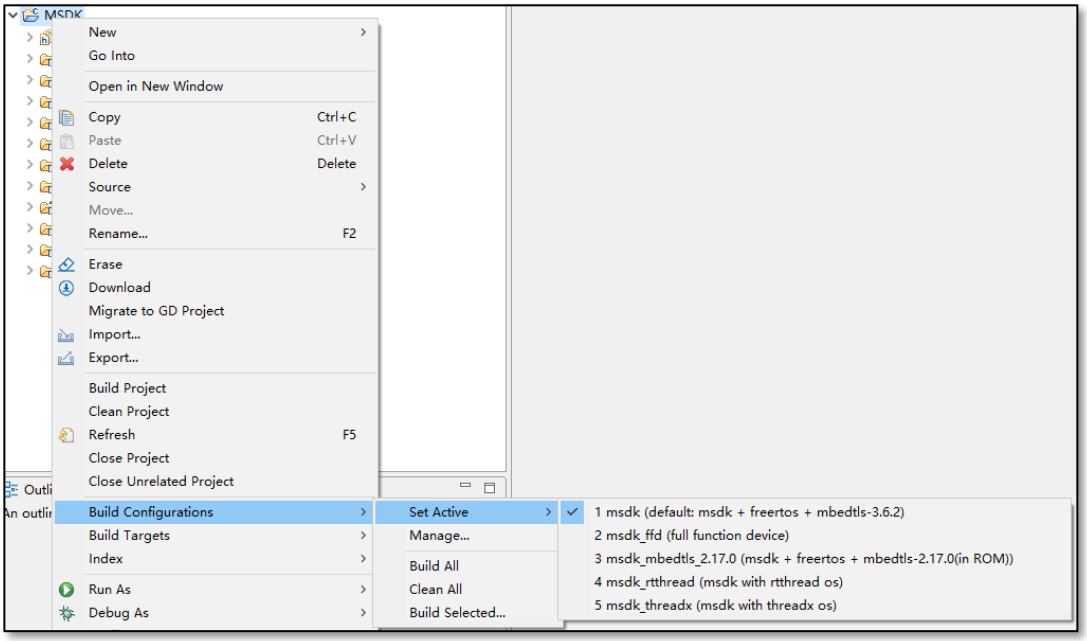


编译完成后会自动调用脚本 MBL\project\mbl_afterbuild.bat 生成 mbl.bin，并拷贝至目录 scripts\images。

■ 编译 MSDK 工程

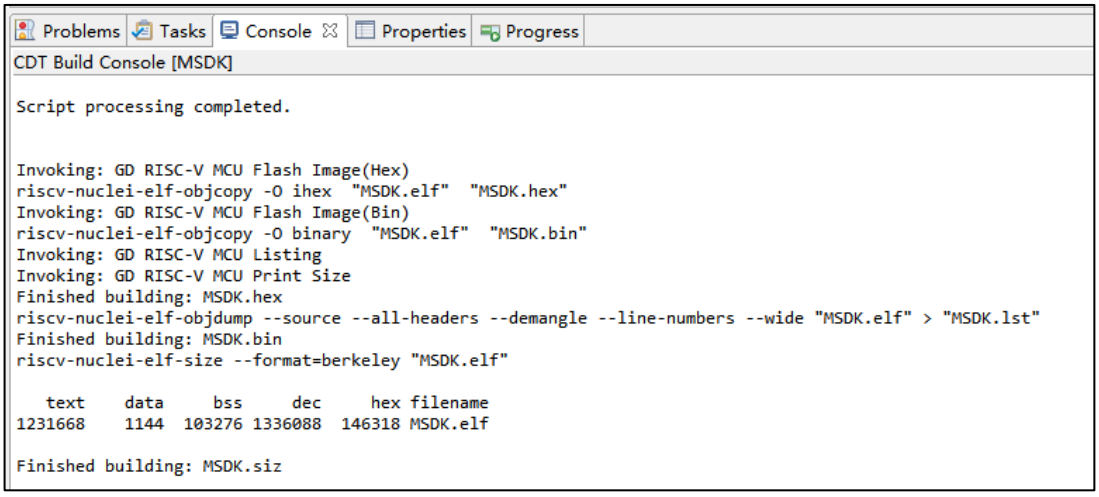
右击工程，依次点击 Build Configurations—>Set Active—>选择需要的配置，如*图 4-11. Configurations 选择*所示，MSDK 默认 configuration 为 msdk。

图 4-11. Configurations 选择



再次右击工程，点击 Build project，编译结果如*图 4-12. MSDK 编译结果*所示。

图 4-12. MSDK 编译结果



■ SDK 生成的 image

MSDK 编译完成之后，会调用 MSDK\projects\image_afterbuild.bat 生成 image-ota.bin 和 image-all.bin。并将生成的 bin 文件拷贝至\scripts\images 内，如*图 4-13. images 输出*所示。

image-ota.bin 是 MSDK 工程生成的 bin 文件，可用于 OTA 升级，image-all.bin 是 MBL(mbl.bin)和 MSDK(image-ota.bin)的组合，该固件可用于生产，烧录到 FLASH 中运行。

图 4-13. images 输出

名称	修改日期	类型	大小
image-all.bin	2024/7/11 14:26	BIN 文件	788 KB
image-ota.bin	2024/7/11 14:26	BIN 文件	748 KB
mbl.bin	2024/7/11 14:19	BIN 文件	17 KB

4.3. 固件下载

4.3.1. U盘拷贝

如 *1.4 下载接口*章节展示，拷贝 GD32VW55x_RELEASE\scripts\images\image-all.bin 至 Gigadevice 盘即可烧写，该功能仅在 START 开发板且使用板载 GDLink 连接时支持。

4.3.2. 使用afterbuild.bat下载

工程通过 afterbuild 调用脚本支持编译完成后自动完成 image 的下载，脚本文件为：MSDK\projects\image_afterbuild.bat。

在 image_afterbuild.bat 的末尾有如*图 4-14 配置 image 自动下载*所示的一段代码。

图 4-14 配置 image 自动下载

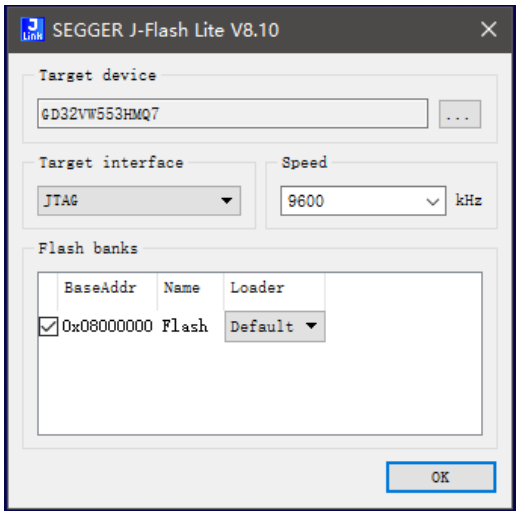
```
:download
set OPENOCD="%OPENOCD_PATH%\\openocd.exe"
::set LINKCFG="..\\openocd_gdlink.cfg"
set LINKCFG="..\\openocd_jlink.cfg"
@echo on
::%OPENOCD% -f %LINKCFG% -c "program %DOWNLOAD_BIN% 0x0800A000 verify reset exit;"
:end
```

该段代码是运用 openocd+Jlink/GDLink 来实现下载，通过配置 openocd 使用不同的 cfg 文件来选择是使用 Jlink 或 GDLink 来下载，当使用 GDLink 连接电脑和开发板时，需将图中 LINKCFG 设置成 openocd_gdlink.cfg，若使用 Jlink 连接时，将 LINKCFG 设置成 openocd_jlink.cfg。在编译完成后，afterbuild 会调用该脚本执行配置的内容完成对应 image 的下载（请将图中红框内的取消注释，该功能默认不启用）。

4.3.3. 使用J-Flash Lite下载

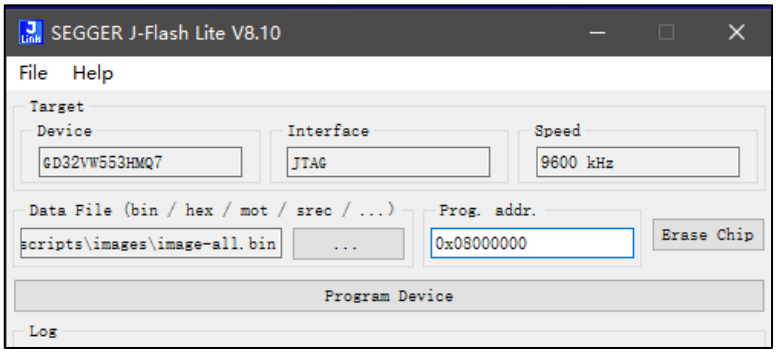
当使用 Jlink 进行调试和固件下载时，在目录：GD32EmbeddedBuilder_v1.4.14.29824\Tools\J-Link 下有 JFlashLite.exe，双击打开后按*图 4-15 JFlashLite 配置*所示配置 JFlashLite，选择 Target device 为 GD32VW553xxxx，选择 Target interface 为 JTAG（也可选 cJTAG，但是速度慢），Speed 选择 9600Khz；然后点击 OK。

图 4-15 JFlashLite 配置



在打开界面中，如*图 4-16 J-Flash 烧录界面*所示，选择 Data File 为编译产生的 image-all.bin 或 image-ota.bin（编译完成后存放在目录 GD32VW55x_RELEASE_V1.0.xx\scripts\images 下）；当选择的是 image-all.bin 文件时，右侧的 Prog. Addr.设置为 0x08000000；当选择的是 image-ota.bin 时，右侧的 Prog. Addr.设置为 0x0800A000。设置完成后点击 Program Device，等待进度条完成即完成烧录。

图 4-16 J-Flash 烧录界面



4.4. 调试

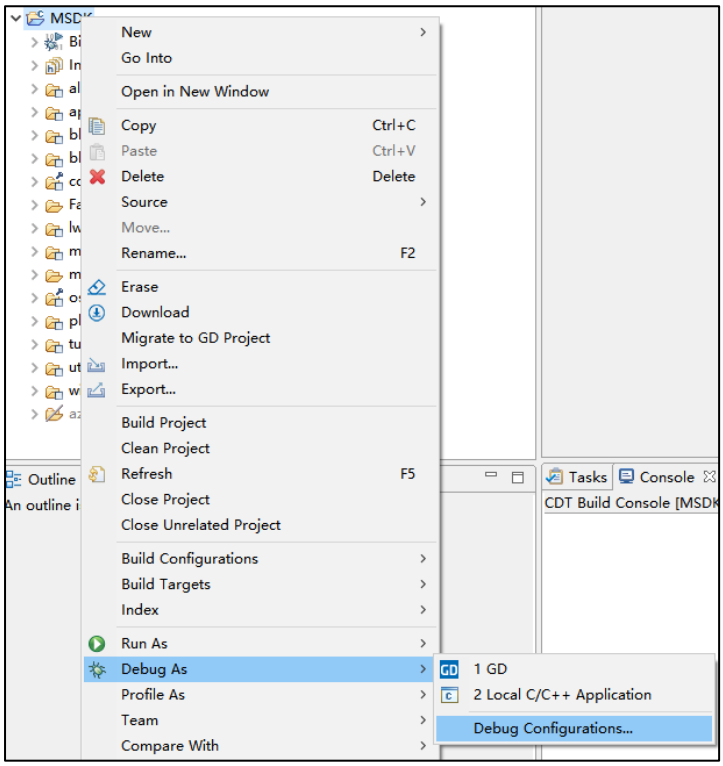
目前 START 开发板和 EVAL 开发板都有板载 GDLink，也可外接 Jlink 用于调试。

调试过程如下所述，默认使用 msdk 的工程配置，如需切换其他工程配置，请参考 *6.3 调试时选择不同的工程配置*。

4.4.1. 配置调试配置

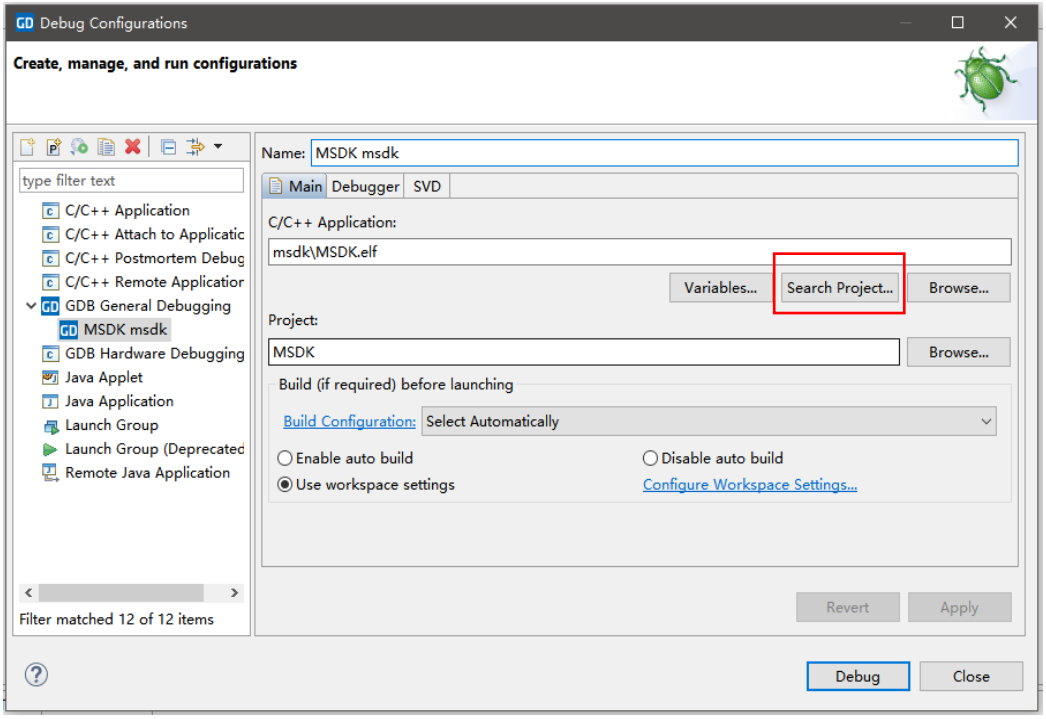
右击 MSDK 工程，依次点击 Debug As->Debug Configurations，如*图 4-17. 打开调试配置选项*所示。

图 4-17. 打开调试配置选项



双击左侧的 GDB General Debugging，会自动新建一个 Debug configuration，如*图 4-18. MSDK 调试配置*所示。其中 c/c++ application 已经自动选择 msdk\MSDK.elf，它指向需要调试的 configuration 编译产生的 elf 文件。可勾选 Enable/Disable auto build 来选择是否在调试前编译工程。

图 4-18. MSDK 调试配置



4.4.2. 使用GDLink进行调试

按*图 4-19 MSDK 使用 openocd 调试配置界面*所示在 Debugger 界面将 GDB Server 切换到 openocd，同时将红框内的 Config Options 按图中所示进行指定后点击 Debug 即开始调试。调试界面如*图 4-20. MSDK 调试界面*所示。

图 4-19 MSDK 使用 openocd 调试配置界面

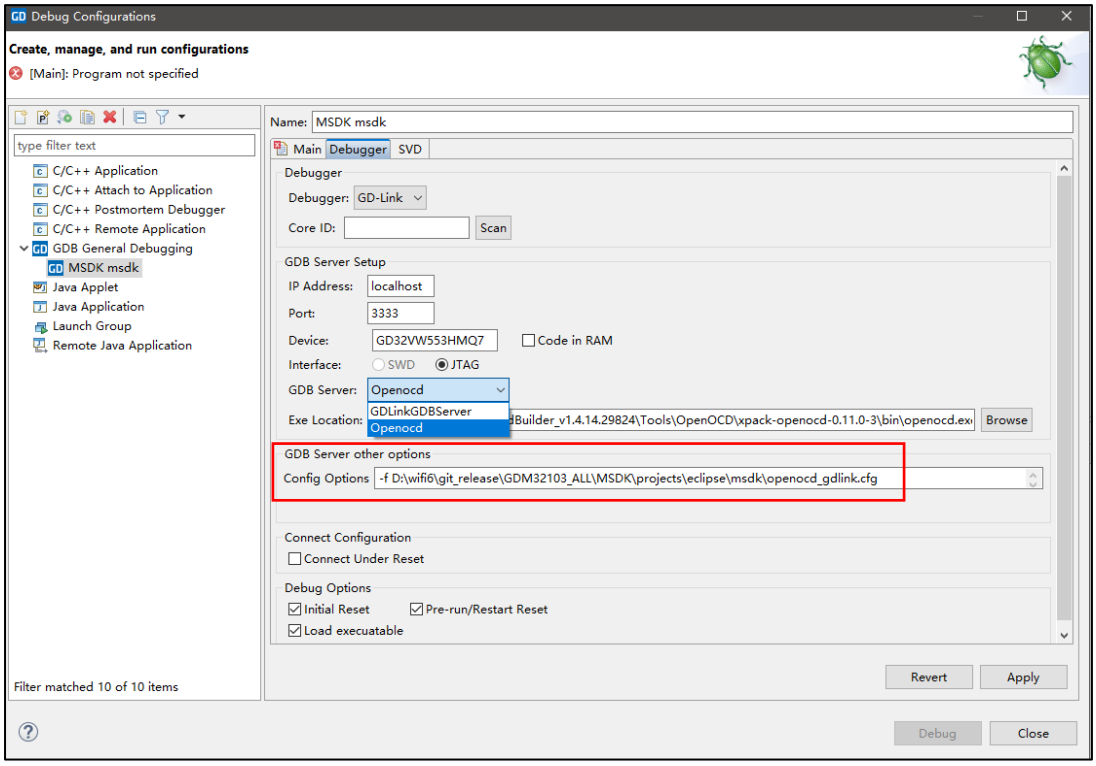
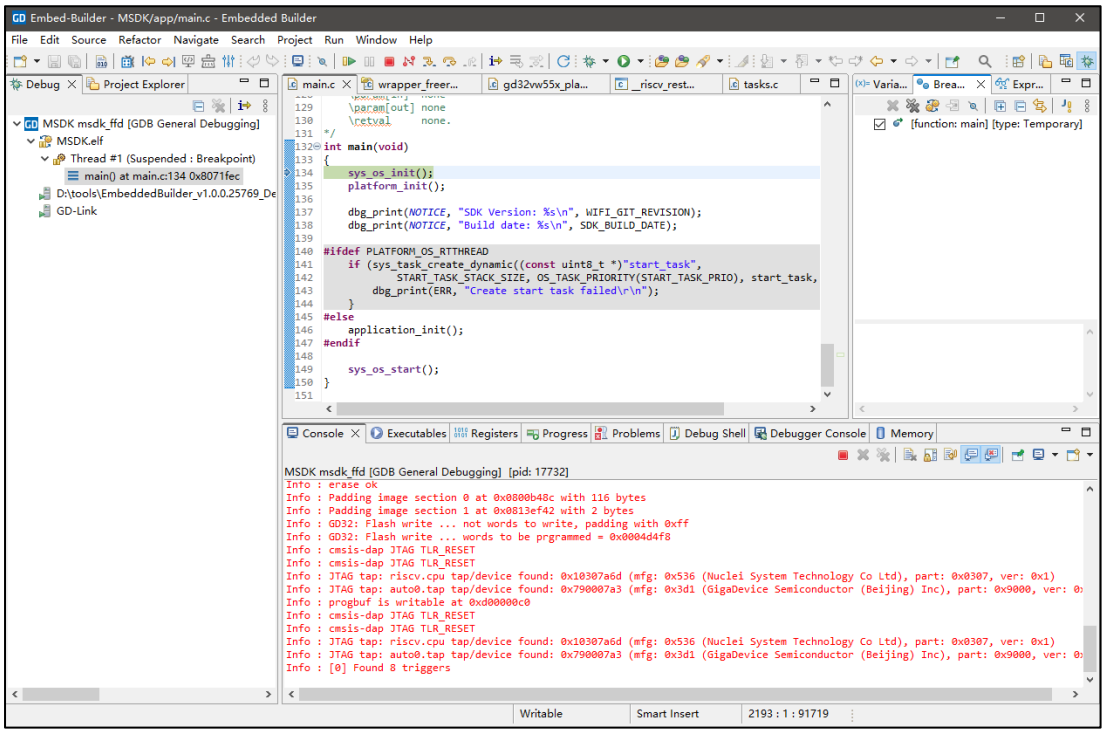


*图 4-19 MSDK 使用 openocd 调试配置界面*中的 Debug Options 中 Initial Reset 和 Pre-run/Restart Reset 选中后会在调试开始时 reset 芯片；Load executable 会在调试前烧录一次固件。

4.4.3. 使用JLink进行调试

GD32VW553 支持 JTAG 和 cJTAG 调试，首先将 Jlink 调试器的引脚和 GD32VW553 JTAG 引脚连接。其次将*图 4-19 MSDK 使用 openocd 调试配置界面*中红框内的 cfg 文件换成 openocd_jlink.cfg（该文件在目录： MSDK\projects\eclipse\msdk 下），然后点击 debug 即可进入调试。在使用 JLink 调试过程中若遇到驱动问题，请参考 *6.4JLink 驱动更换*。

图 4-20. MSDK 调试界面



5. SEGGER Embedded Studio IDE 工程

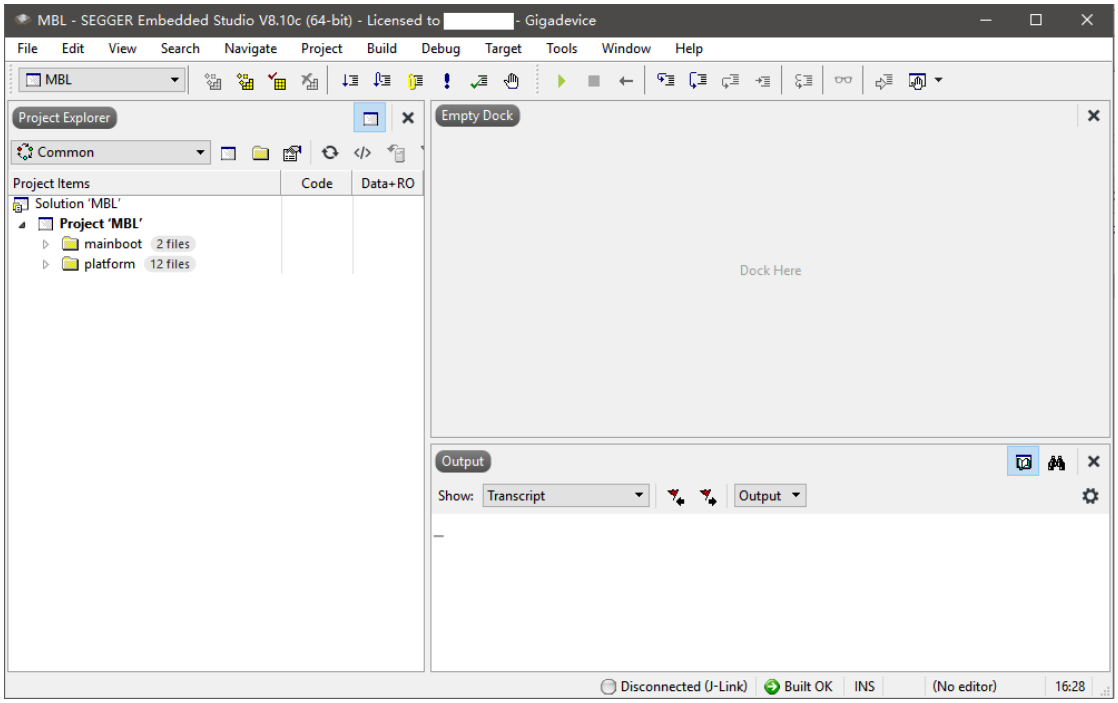
本章将介绍如何在 SEGGER Embedded Studio IDE 下编译和调试 SDK。

5.1. 打开工程组

- 打开 MBL 工程

打开目录：GD32VW55x_RELEASE\MBL\project\segger，双击 MBL.emProject 打开 MBL SES 工程，打开后的工程如*图 5-1. MBL SES Project 工程界面*所示。

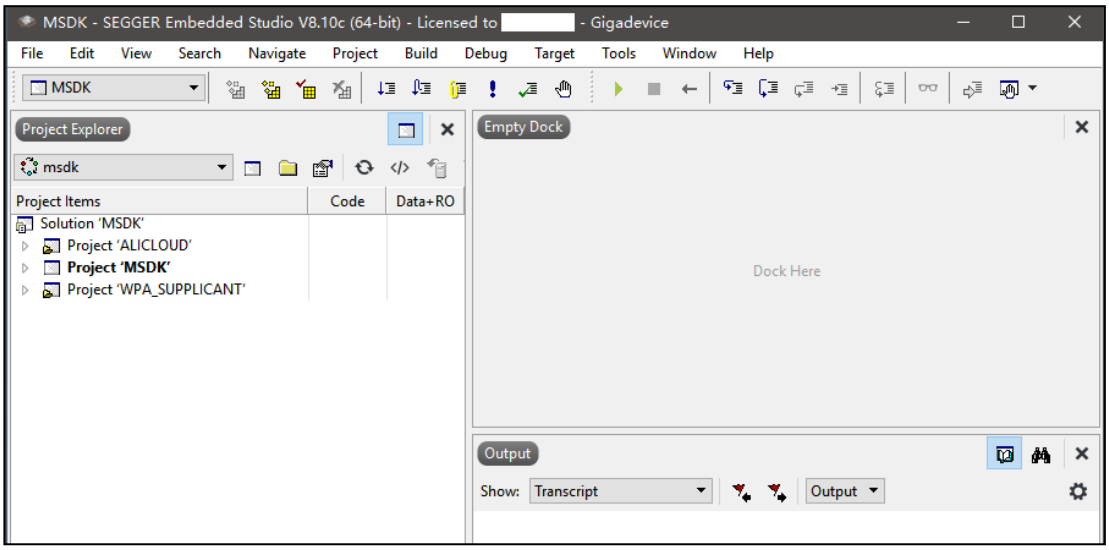
图 5-1. MBL SES Project 工程界面



- 打开 MSDK 工程

打开目录：GD32VW55x_RELEASE\MSDK\projects\segger，双击 MSDK.emProject 打开 MSDK 工程，打开后的工程如*图 5-2. MSDK SES 工程界面*所示。

图 5-2. MSDK SES 工程界面

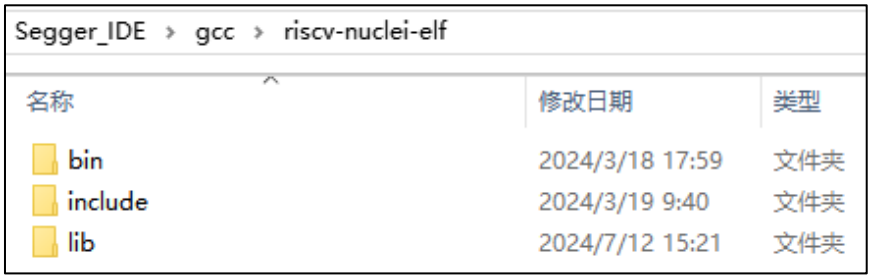


5.2. 编译

- SES build tool 配置

SES 默认使用 riscv32-none-elf 的工具链编译 GD32VW55x 的工程，为了更好地支持 riscv 的拓展指令集，需要使用 nuclei 的工具链即 riscv-nuclei-elf 编译。编译工具可联系销售或 FAE 获取，工具链具体内容如*图 5-3. nuclei 工具链内容*所示。图中 Segger_IDE 为 SES IDE 安装目录。

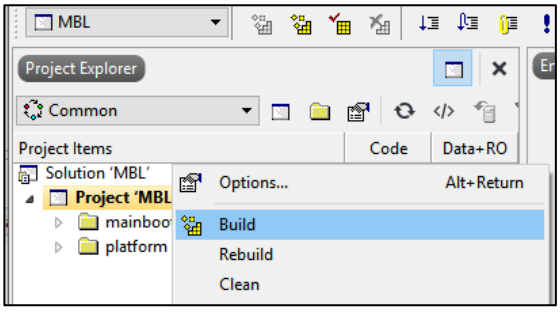
图 5-3. nuclei 工具链内容



- 编译 MBL 工程

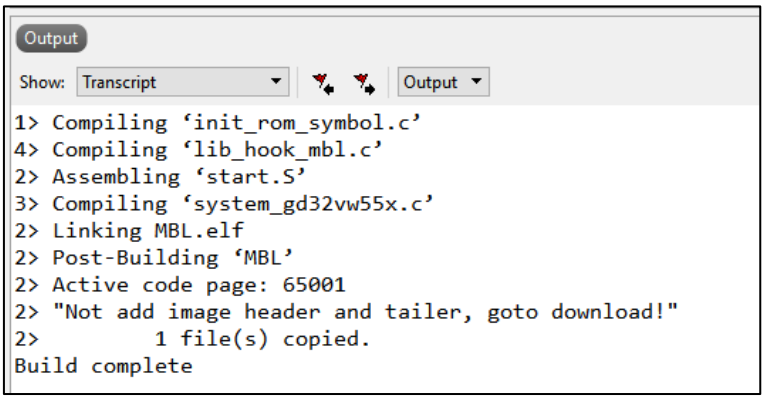
右击工程，点击 build，如*图 5-4. 编译 MBL 工程*所示；或者点击菜单栏 Build->Build MBL。

图 5-4. 编译 MBL 工程



编译结果，如*图 5-5. MBL 编译结果*所示。

图 5-5. MBL 编译结果

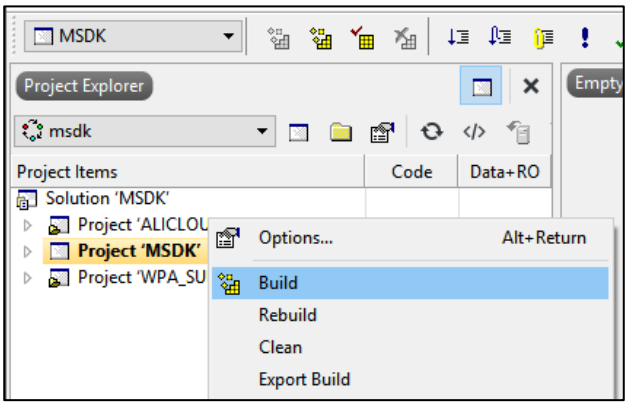


编译完成后会自动调用脚本 MBL\project\mbl_afterbuild.bat 生成 mbl.bin，并拷贝至目录 scripts\images。

- 编译 MSDK 工程

右击 MSDK 工程内 Project 'MSDK'，点击 Build，如*图 5-6. 编译 MSDK 工程*所示。

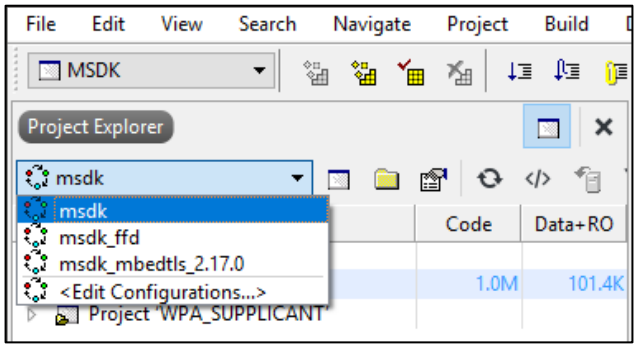
图 5-6. 编译 MSDK 工程



- MSDK configuration 选择

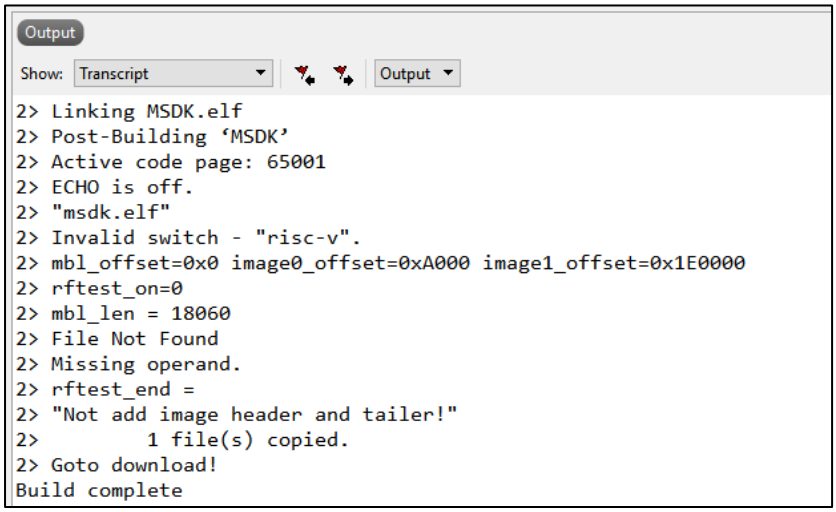
MSDK 的 configuration 切换如*图 5-7. MSDK 工程配置选择*所示，当前 SES 的工程仅支持了 msdk、msdk_ffd、msdk_mbedtls_2.17.0；若需使用配置 msdk_threadx 和 msdk_rtthread 还请使用 GD32 EmbeddedBuilder IDE 工程或者等待后续更新。

图 5-7. MSDK 工程配置选择



选择对应的配置后，右击工程，点击 Build，编译结果如*图 5-8. MSDK 编译结果*所示。

图 5-8. MSDK 编译结果



- SDK 生成的 image

MSDK 编译完成之后，会调用 MSDK\projects\image_afterbuild.bat 生成 image-ota.bin 和 image-all.bin。并将生成的 bin 文件拷贝至\scripts\images 内，如*图 5-9. images 输出*所示。

image-ota.bin 是 MSDK 工程生成的 bin 文件，可用于 OTA 升级，image-all.bin 是 MBL(mbl.bin) 和 MSDK(image-ota.bin)的组合，该固件可用于生产，烧录到 flash 中运行。

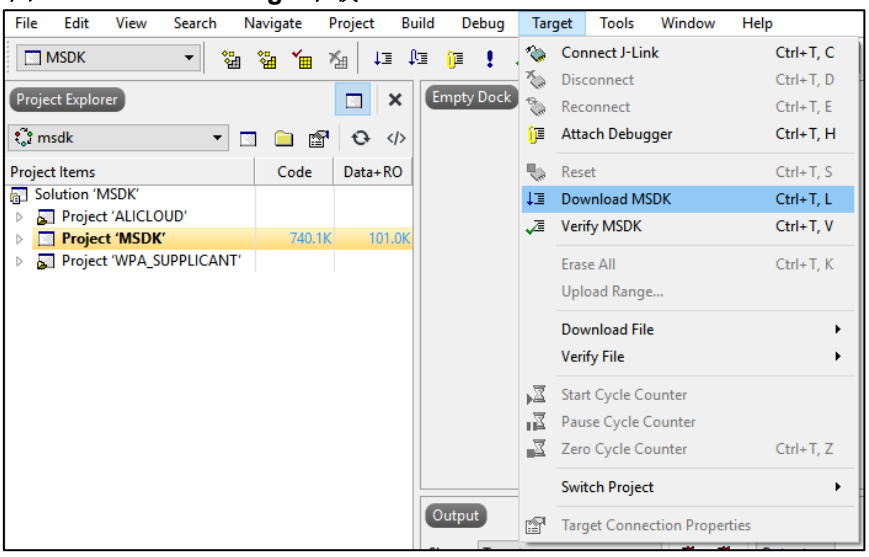
图 5-9. images 输出

5.3. 固件下载

如 *1.4 下载接口* 章节，拷贝 GD32VW55x_RELEASE\scripts\images\image-all.bin 至 Gigadevice 盘即可烧写。

或者通过点击菜单栏 Target->Download MSDK 下载，如*图 5-10 SES IDE image 下载*所示。

图 5-10 SES IDE image 下载



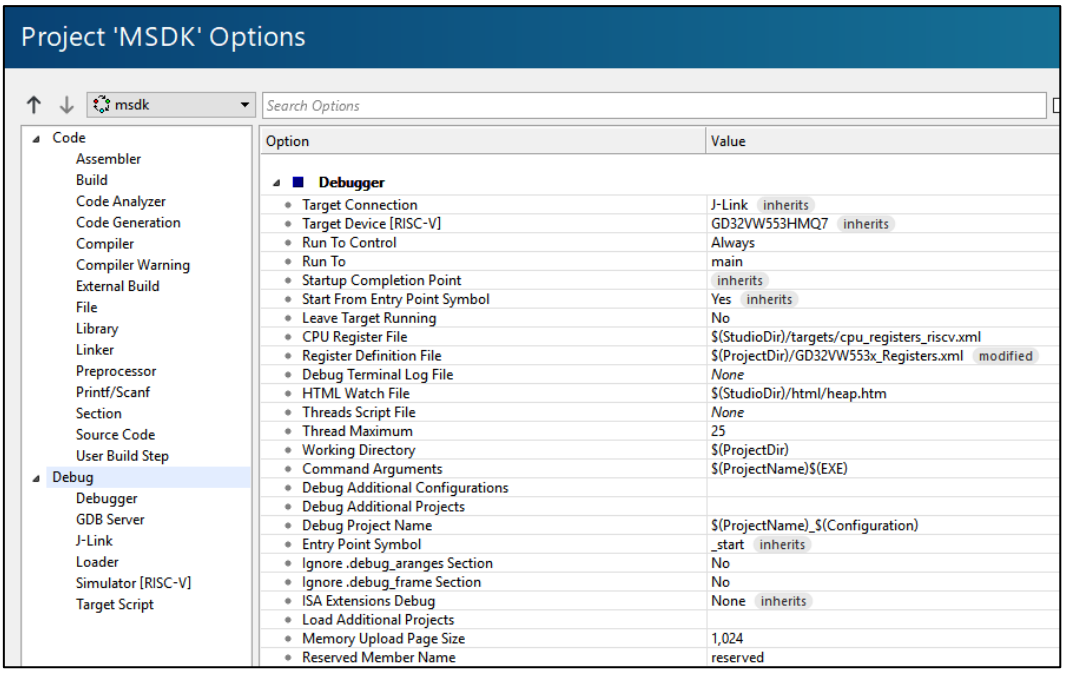
5.4. 调试

■ 调试配置

SES IDE 建议使用 jlink 调试，且 jlink driver 版本至少为 V7.92o，该版本 jlink driver 支持 GD32VW55x 芯片。

工程默认已经配置好 Debug 信息，如需更改，右击 MSDK 工程，点击 Options 打开配置界面，可在 Debug 选项下修改 Debugger 和 JLink，如*图 5-11 MSDK SES 工程配置界面*所示。

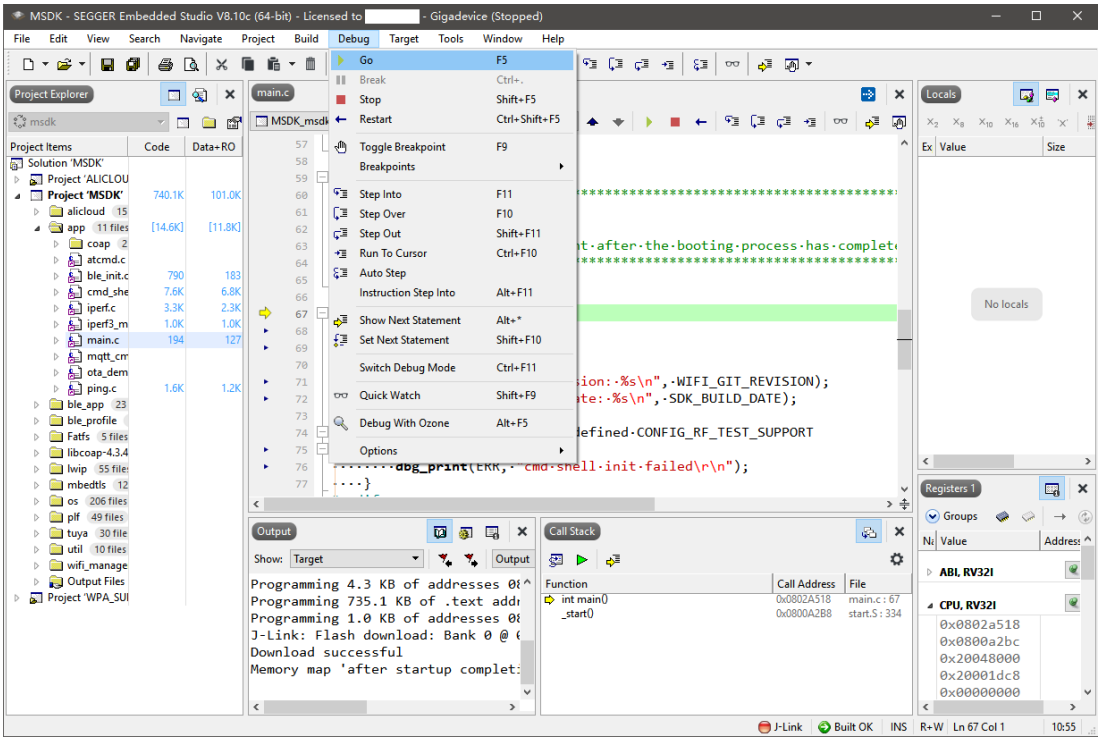
图 5-11 MSDK SES 工程配置界面



■ 开始调试

点击菜单栏的 Debug->GO 即可调试，点击后等待烧录 image 完成后进入如*图 5-12 SES IDE Debug 界面*所示界面。

图 5-12 SES IDE Debug 界面



6. 常见问题

6.1. No image 错误

打印 ERR: No image to boot (ret = -5) 。

原因：前一次引导 MSDK 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。

6.2. 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。

可以打开 GD32VW55x_RELEASE\MSDK\plf\riscv\env\gd32vw55x.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：

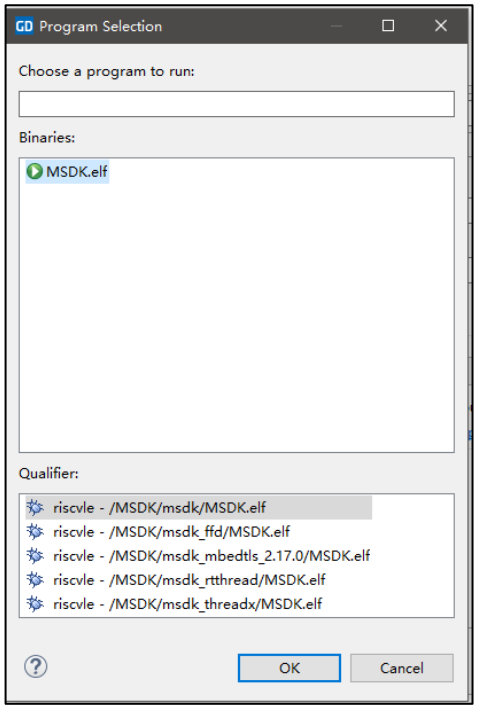
KEEP (*port.o* (.text* .rodata*))是将整个 port.c 文件放进 SRAM 运行。

例如：KEEP (*tasks.o* (.text.xTaskIncrementTick))是将 tasks.c 中的 xTaskIncrementTick () 函数放进 SRAM 运行。

6.3. 调试时选择不同的工程配置

MSDK 工程对应多个 configuration（详见 ***3.2.6 工程配置***），调试时需选择对应的 configuration，具体操作为，点击***图 4-18. MSDK 调试配置***中的红框的 Search Project，打开如***图 6-1 选择工程配置调试***所示界面，双击下方 Qualifier 框内要调试的 configuration 即可（需要先编译对应的 configuration 才会出现对应的选项）。

图 6-1 选择工程配置调试

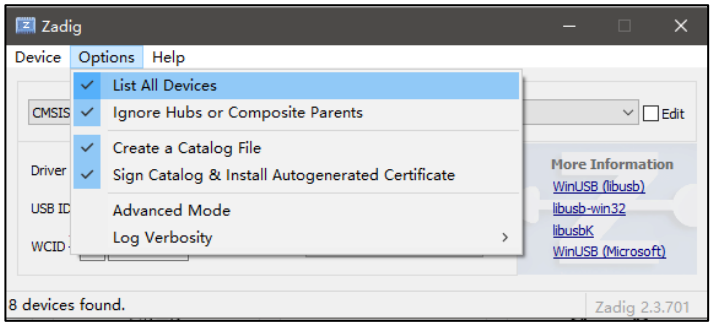


6.4. JLink 驱动更换

在 Embedded Builder 工程中使用 openocd+JLink 调试时若遇到“Error: No J-Link device found”的问题，则需要更改 JLink 驱动，更改步骤如下：

1. 使用管理员权限打开 zadig.exe 文件（官网：https://zadig.akeo.ie），点击 options，勾选 List All Devices，如*图 6-2 Zadig 选项勾选*所示。

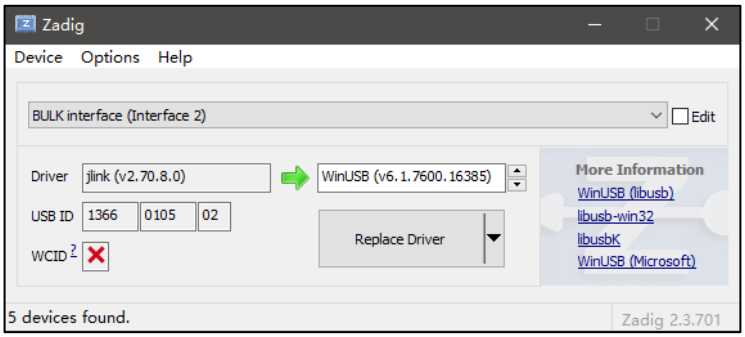
图 6-2 Zadig 选项勾选



2. 选择下拉框内的 JLink devices，如*图 6-3 替换 JLink 驱动*

3. 中所示是 BULK interface，点击 Replace Driver，将 JLink 驱动替换成 WinUSB。

图 6-3 替换 JLink 驱动



4. 等待替换完成之后插拔一下 JLink 设备，之后再用 JLink debug 即无驱动问题。

7. 版本历史

表 7-1. 版本历史

版本号.	说明	日期
1.0	首次发布	2023 年 10 月 17 日
1.1	第二章开发环境内容修订	2024 年 01 月 26 日
1.2	新增 SES IDE 工程，GD32 Eclipse IDE 更新为 GD32 Embedded Builder	2024 年 7 月 17 日
1.3	更新部分图示与 SDK1.0.3 相匹配	2025 年 4 月 8 日
1.3a	更新开发板图片，新增固件下载等	2025 年 5 月 15 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 according to the laws of the People’s Republic of China and other applicable laws. The Company reserves all rights under such laws and no Intellectual Property Rights are transferred (either wholly or partially) or licensed by the Company (either expressly or impliedly) herein. 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.

To the maximum extent permitted by applicable law, the Company makes no representations or warranties of any kind, express or implied, with regard to the merchantability and the fitness for a particular purpose of the Product, nor does the Company assume any liability arising out of the application or use of any Product. Any information provided in this document is provided only for reference purposes. It is the sole responsibility of the user of this document to determine whether the Product is suitable and fit for its applications and products planned, and properly design, program, and test the functionality and safety of its applications and products planned using the Product. The Product is designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and the Product is not designed or intended for use in (i) safety critical applications such as weapons systems, nuclear facilities, atomic energy controller, combustion controller, aeronautic or aerospace applications, traffic signal instruments, pollution control or hazardous substance management; (ii) life-support systems, other medical equipment or systems (including life support equipment and surgical implants); (iii) automotive applications or environments, including but not limited to applications for active and passive safety of automobiles (regardless of front market or aftermarket), for example, EPS, braking, ADAS (camera/fusion), EMS, TCU, BMS, BSG, TPMS, Airbag, Suspension, DMS, ICMS, Domain, ESC, DCDC, e-clutch, advanced-lighting, etc.. Automobile herein means a vehicle propelled by a self-contained motor, engine or the like, such as, without limitation, cars, trucks, motorcycles, electric cars, and other transportation devices; and/or (iv) other uses where the failure of the device or the Product can reasonably be expected to result in personal injury, death, or severe property or environmental damage (collectively "Unintended Uses"). Customers shall take any and all actions to ensure the Product meets the applicable laws and regulations. The Company is not liable for, in whole or in part, and customers shall hereby 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 Product. Customers shall indemnify and hold the Company, and its officers, employees, subsidiaries, affiliates 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 Product.

Information in this document is provided solely in connection with the Product. The Company reserves the right to make changes, corrections, modifications or improvements to this document and the Product described herein at any time without notice. The Company shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. Information in this document supersedes and replaces information previously supplied in any prior versions of this document.