HOPERF

# CMT453x VSCode GCC环境搭建应用笔记V1.0

## 简介

本文档主要介绍 CMT453x系列蓝牙芯片，使用VSCode编辑平台，ARM-GCC工具链，来搭建开发调试环境。

# 目录

## 目录

目录 ..... 2
概述 ..... 3
1 使用 VSCODE 将 KEIL/IAR 工程导入 ..... 3
1.1 VSCODE EIDE GCC 工具链安装 ..... 3
1.2 工程导入步骤及相关配置 ..... 3
1.3 工程导入 ..... 3
1.4 选择 GCC 工具链 ..... 5
1.5 修改 GCC 启动文件 ..... 7
2 工程编译问题 ..... 7
2.1 关键字__ALIGN 使用问题 ..... 7
2.2 汇编函数调用问题 ..... 8
2.3 关键字__WEAK 的使用问题 ..... 9
2.3 函数未定义问题 ..... 10
3 下载调试 ..... 12
3.1 准备 PACK 包 ..... 12
3.2 在 JLINK 中添加芯片型号 ..... 12
3.3 在 VSCODE 中添加芯片支持包 ..... 13
3.4 代码下载 ..... 14
3.5 测试结果 ..... 16
历史版本 ..... 18

# 概述

本文档硬件平台基于 CMT453x 开发板，软件开发平台基于 VSCode + EIDE + GCC Toolchain。虽然 VSCode 只是一款编辑器，只有编辑代码的功能，但是它有丰富的插件支持 MCU 的开发，且开源免费。目前比较主流的插件有 Cmake、EIDE、Clion 等。本应用笔记采用的是 EIDE（全称 Embedded IDE）。它是一款适用于 8051/Cortex-M/RISC-V 的单片机开发环境的插件，能够在 VSCode 上提供 8051/ Cortex-M/RISC-V 项目的开发, 编译, 烧录等功能。

# 1 使用 VSCode 将 Keil/IAR 工程导入

## 1.1 VSCode EIDE GCC 工具链安装

本小节主要介绍搭建 CMT453x VSCode+GCC 环境需要安装以下软件以及相关插件:

- VSCode(官网下载地址：https://code.visualstudio.com/Download)
- VSCode 中 Embedded IDE、C/C++、Cortex-Debug 插件(在 VSCode 扩展功能中安装即可，需要连网)
- Jlink(推荐 V7.66 及以下版本，下载链接： https://www.segger.com/downloads/jlink/)
- GCC工具链

  (下载链接：https://developer.arm.com/downloads/-/gnu-rm，请注意要用cortex-r/m版本，笔者目前使用的最新版本：gcc-arm-none-eabi-10.3-2021.10-win32)

## 1.2 工程导入步骤及相关配置

使用 VSCode 将现有的 CMT453x 的 Keil/IAR 工程导入分为以下几个步骤：

- 工程导入
- 选择 GCC 工具链
- 修改 gcc 启动文件

## 1.3 工程导入

使用 EIDE 的好处之一就是它可以直接导入 keil/IAR 工程，keil/IAR 的工程结构可以直接导入到 VSCode 中，用户无需再重建工程，做到无缝对接。打开安装好的 VScode，点击左侧芯片图标的 EIDE 插件，会弹出 EIDE PROJECTS 工程管理界面，这里只需导入现成的 keil/IAR 工程，如图 1 所示选择 Import Project，导入本文档所用的 rdts 工程，如图 2 所示。

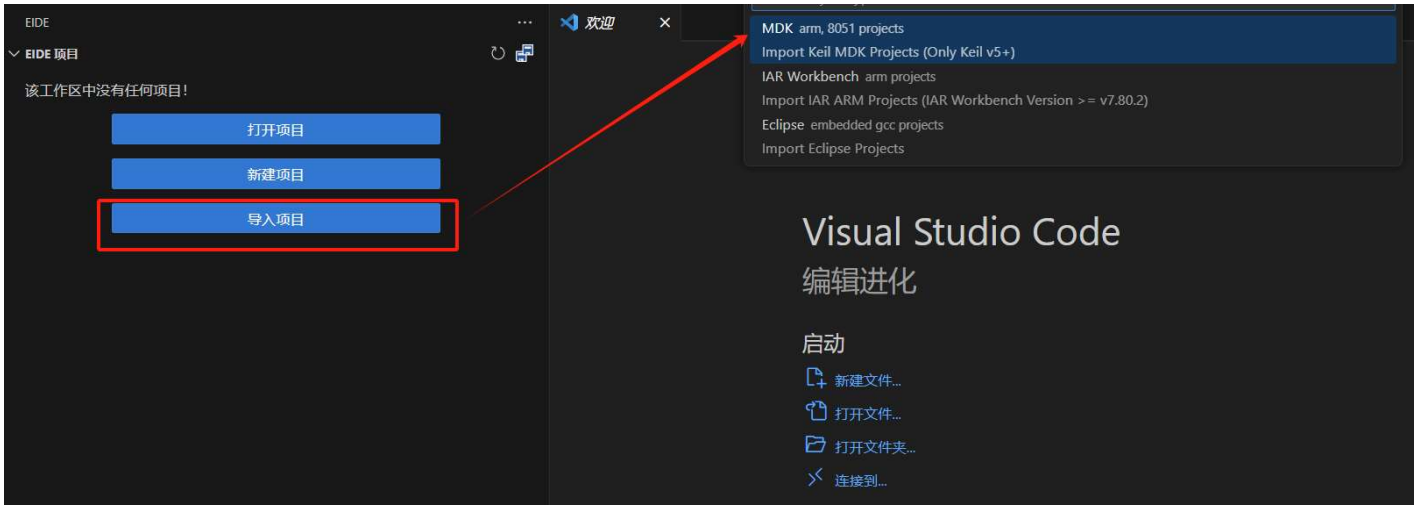



图 1 VSCode 导入 keil 工程

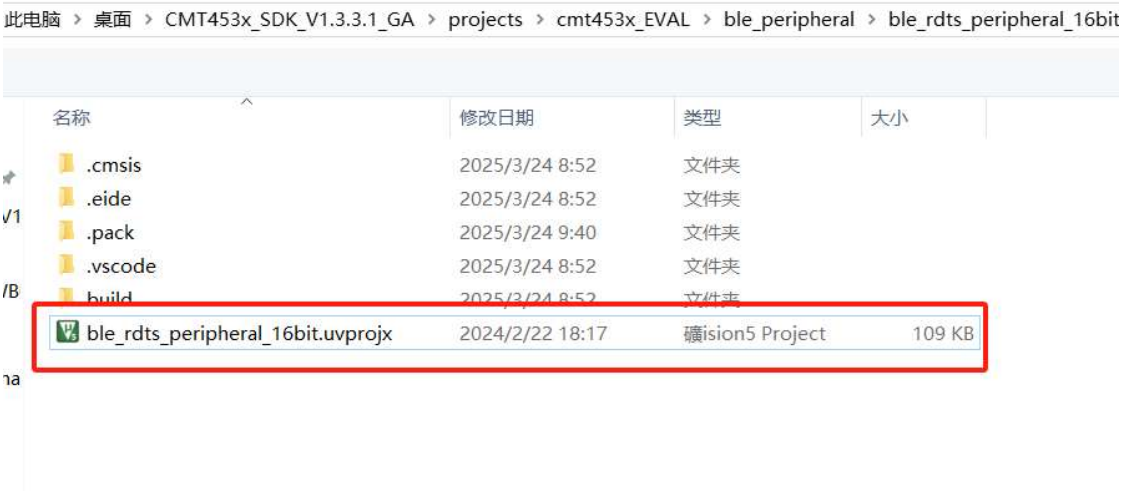
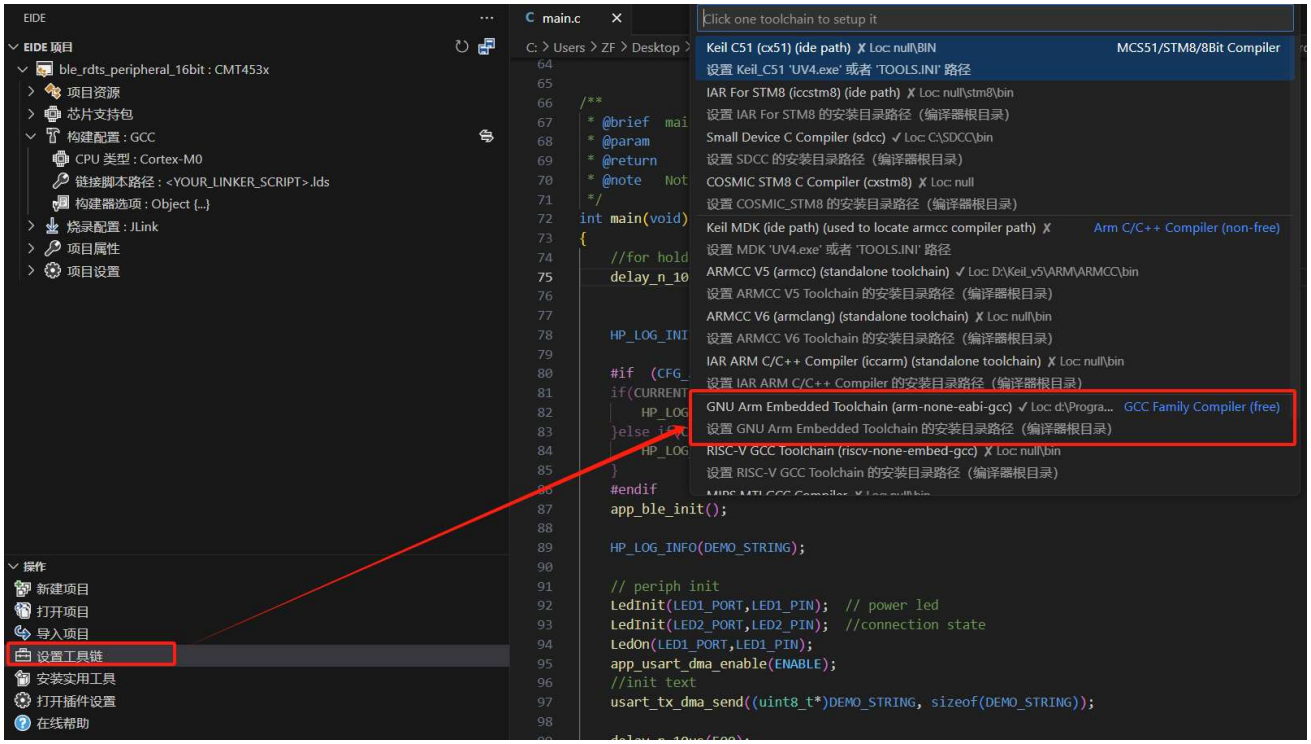


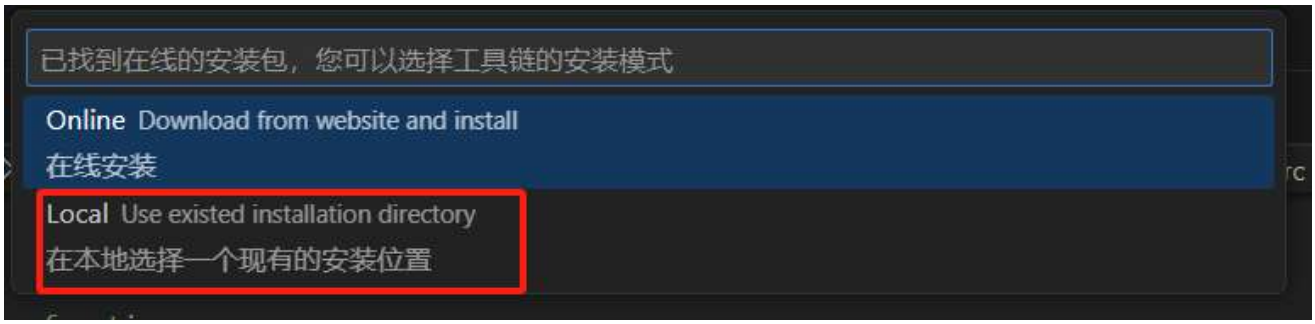

图 2 VSCode 导入 rdts 工程

## 1.4 选择 GCC 工具链

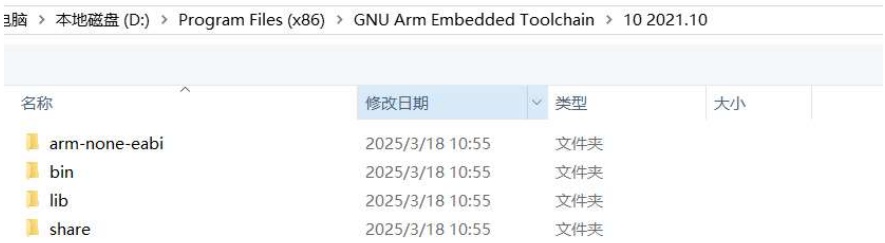
选择 GCC 工具之前，先在 VSCODE 中配置好 GCC 工具链的路径，按照下图选择配置 GCC 工具链。




然后在弹出的选框中选择本地已安装路径，




找到前面步骤安装的 GCC 路径，选择带日期版本的文件夹



设置完成后可再次打开“设置工具链”，看到“GNU Arm Embedded Toolchain”右边的‘×’变成了‘√’，就说明工具链设置完成。


因为使用的是 GCC 工具链编译工程，因此工具链配置项中要选择 GCC。EIDE 插件中默认是使用 AC5，如图 3 所示选择“Builder Configurations: AC5”，在右侧会弹出 AC5、AC6、GCC、IAR_ARM，选择 GCC 后可以看到工具链 Builder Configurations:后面变成了 GCC

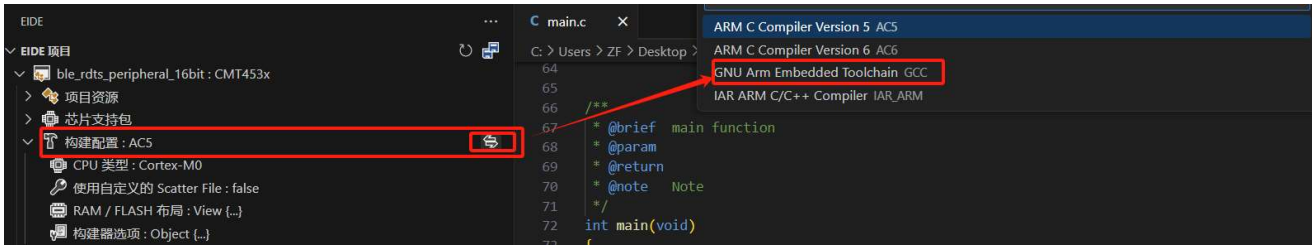



图 3 选择 GNU 工具链

构建配置项下面还有链接脚本路径，这个就是链接脚本路径配置，链接脚本已经放到了 firmware\CMSIS\device\ cmt453x_flash.ld 路径下，用户根据自己存放工程路径来选择即可。也可以输入相对路径” ../../../../../firmware/CMSIS/device/cmt453x_flash.ld” 如图 4 所示路径输入后记得要敲 Enter 键。链接脚本选择上面有个 CPU Tyte 选项，CMT453x 是基 Cortex-M0 内核，选择 Cortex-M0 即可。

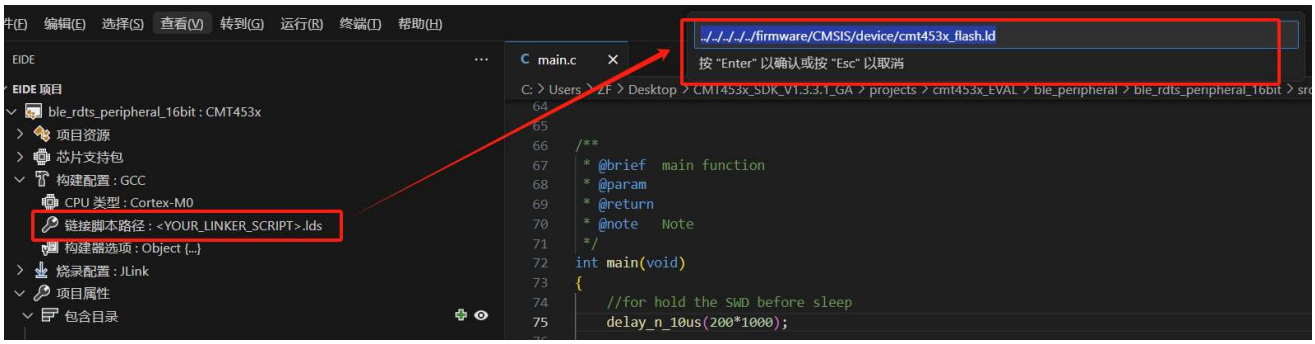



图 4 导入链接脚本

## 1.5 修改 GCC 启动文件

由于 keil/IAR 中的启动文件中带有各自特色的格式以及语法，使用 GCC 汇编器是不兼容的，因此需要给 GCC 工程单独提供一个用 GUN 汇编语法写的启动文件。此启动文件已经放在 firmware\CMSIS\device\startup 路径下，只需将工程中原有的 startup_cmt453x.s 启动文件，从工程中删掉，重新添加 startup_cmt453x_gcc.s 到工程中即可，如图 5 所示。

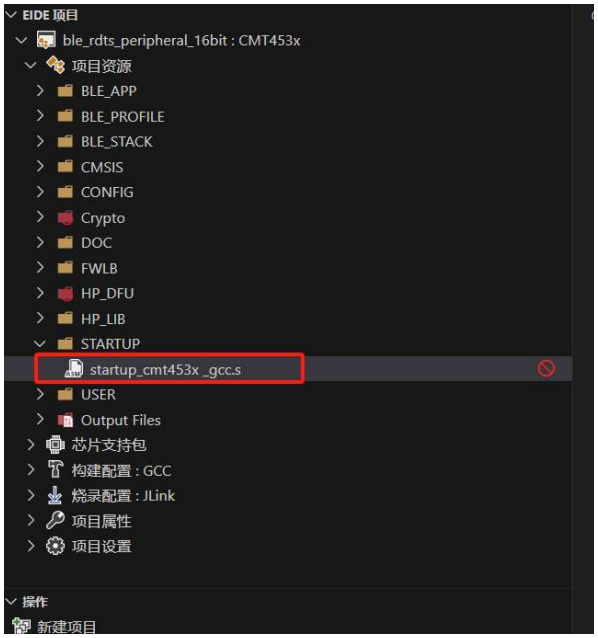



图 5 更换 gcc 启动文件

# 2 工程编译问题

按照上述章节导入、配置工程后就可以对工程进行编译了，由于 ARMCC 与 GCC 编译器在编译上有一些语法不一样，编译过程中难免有一些语法错误，下面对导入 Keil 工程过程中编译出现的错误进行描述。

## 2.1 关键字__align 使用问题

按快捷键 F7 或者点击软件界面右上角编译按钮，编译工程。如下图 6 所示，编译结果的 log 中有编译错误：

```
   80 | };
      | }
In file included from ./../../../../../middlewares/HopeRF/ble_library/hp_ble_stack/stack_common/global_func.h:100,
                 from ./../../../../../middlewares/HopeRF/ble_library/hp_ble_stack/stack_common/rwip.c:121:
./../../../../../middlewares/HopeRF/ble_library/hp_ble_stack/stack_common/ble_stack_common.h:561:16: error: expected declaration specifiers or '...' before numeric constant
  561 | static __align(64) unsigned char  patch_4_Array[0x40] =
      |                ^~
./../../../../../middlewares/HopeRF/ble_library/hp_ble_stack/stack_common/ble_stack_common.h:568:16: error: expected declaration specifiers or '...' before numeric constant
  568 | static __align(64) unsigned char  patch_5_Array[0x40] =
      |                ^~
./../../../../../middlewares/HopeRF/ble_library/hp_ble_stack/stack_common/ble_stack_common.h:577:22: error: expected declaration specifiers or '...' before numeric constant
  577 | const static __align(4) unsigned char  fun_5_Array[]  ={
```

图 6 GCC 编译关键字__align 使用问题

此问题是由于 keil 中对齐关键字__align 的用法与 GUN 环境下使用 align 不一样导致，如 keil 工程中指定某个数组以 4 字节对齐，直接写成__align(4) array[]即可，而在 GNU 环境下需要写成__attribute__((aligned(4))) array[]的形式，将__align(4)和__align(64)的定义写成如下格式：__align(4)改为__attribute__((aligned(4))) __align(64)改为__attribute__((aligned(64)))

重新编译，图 6 中的错误提示就没有了。

## 2.2 汇编函数调用问题

在解决上述字节对齐格式问题后重新编译，在显示结果的 log 中会有如下图 7 所示的错误提示。

```
./../../../../../firmware/CMSIS/device/system_cmt453x.c:273:8: error: expected '(' before 'void'
  273 | __ASM  void system_delay_cycles(uint32_t i)
      |        ^~~~
      |        (
```

图 7 GCC 下 C 语言调用汇编函数问题

从 log 的提示上看不出来是什么问题，按照提示打开 system_cmt453x.c 文件，找到第 273 行，第 273 行是汇编函数__ASM void system_delay_cycles(uint32_t i)的实现。在 keil 中直接在函数名前加上关键字__ASM 就可声明此函数是汇编函数，但是在 GUN 环境下，汇编函数的实现方式却不同，在 GUN 下需要先将汇编函数在.s 文件中实现，然后以标号的方式给函数命名，在其他 C 代码中调用标号即可实现对汇编函数的调用，至于参数 i，在 ARM 架构下默认是使用 R0-R3 传递参数，如果参数个数大于 4 就需要手动压栈操作。system_delay_cycles 函数实现方法如下：

```
.text
.syntax unified
.thumb

/*跨介质跳转调用前的完整现场保护*/
.type system_delay_cycles, %function //声明这是个函数
.global system_delay_cycles              //导出符号使得外部可见，即可导出函数也可导出全局变量

system_delay_cycles:
subs r0, #1
bne system_delay_cycles
bx lr

.end
```

已经改好的文件 system_delay.s 处于 firmware\CMSIS\device\system_delay.s, 将此文件添加工程中 CMSIS 目录下，如图 8 所示，然后将 sytem_cmt453x.c 中的 system_delay_cycles 函数注释掉。

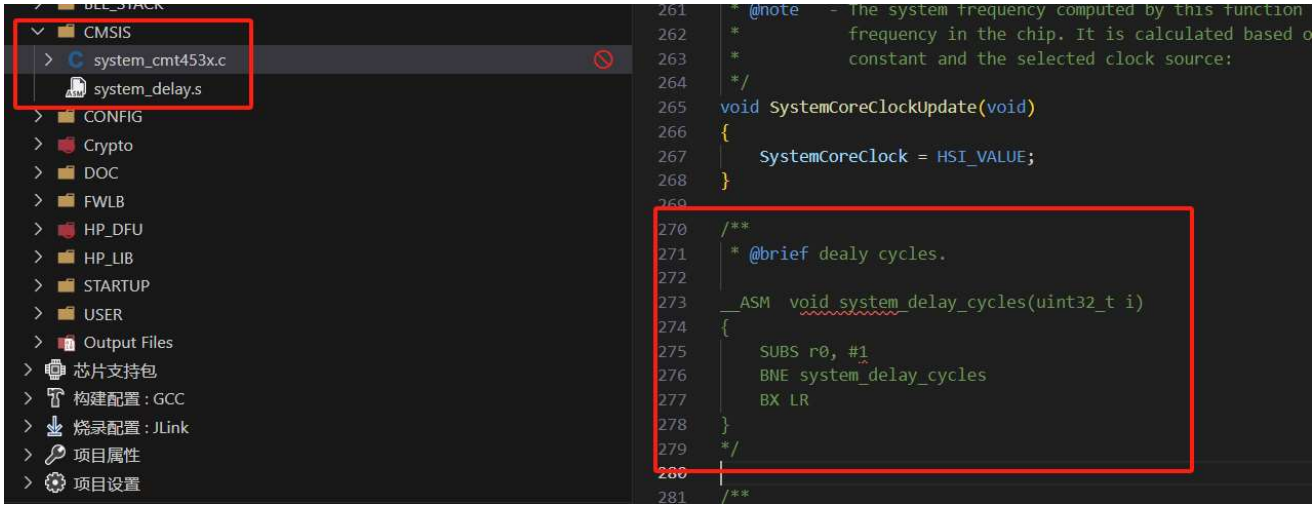



图 8 添加 system_delay.s 并注释 sytem_cmt453x.c 中的 system_delay_cycles 函数

## 2.3 关键字__weak 的使用问题

在解决上述汇编函数调用问题后重新编译，在编译 log 区会有如图 9 所示的错误提示。定位到__weak void app_sleep_prepare_proc(void)。这是 app_sleep_prepare_proc()函数的弱定义，加上了__weak 修饰符的函数，用户可以在用户文件中重新定义一个同名函数，最终编译器编译的时候，会选择用户定义的函数，如果用户没有重新定义这个函数, 那么编译器就会执行__weak 声明的函数，并且编译器不会报错。在 keil 工程中直接在函数名前加__weak 即可，但是在 GNU 下需要用 __attribute__((weak))定义函数弱输出其符号。因此要写成如下格式：

```
__attribute__((weak)) void app_sleep_prepare_proc(void);
```

同样，对 app_sleep_resume_proc 函数也采用此方式修改。

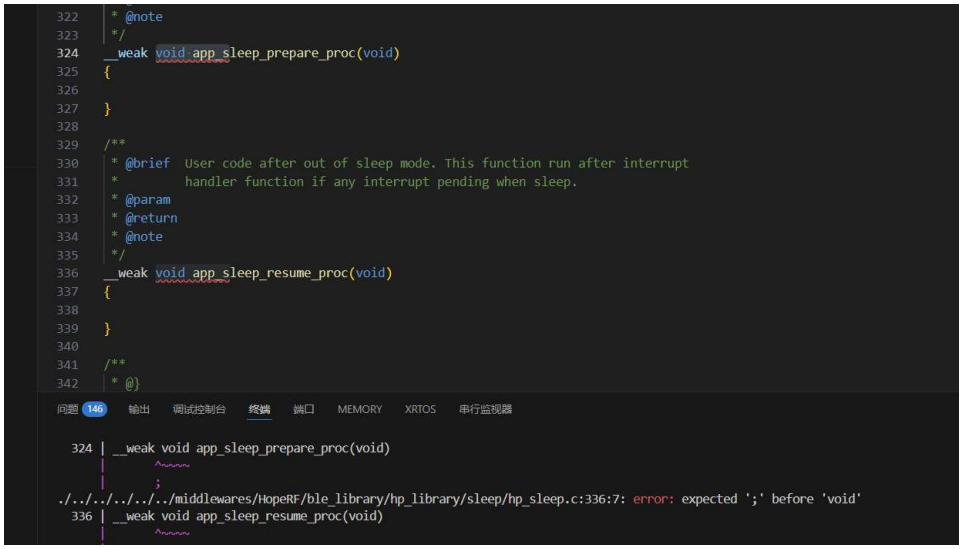



图 9 __weak 使用提示

## 2.3 函数未定义问题

当解决上述弱函数定义问题后，重新编译，还是有一堆 undefined reference 问题，如图 10 所示：

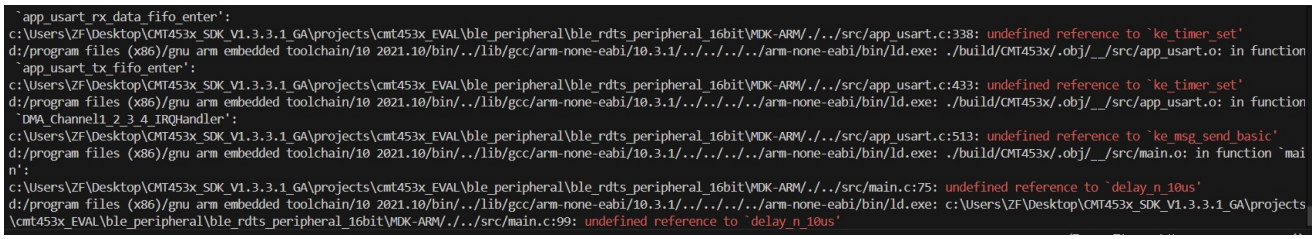



图 10 函数未定义提示

这是因为这些函数都是蓝牙库函数，已经封装起来掩膜到 ROM 中，只能通过 API 函数接口去调用，在这里编译报错是因为编译器在编译、链接的时候找不到这些函数链接的入口地址。那为什么 keil 工程可以正常编译、链接？keil 工程可以正常编译、链接是因为在建 CMT453x keil 工程时，在 Options for Target 选项卡下的 Linker/

Misc controls 一栏将这些库函数的函数符号表导入了工程中，编译链接的时候编译器会自动到选项卡中指定的路径下查找外部函数符号。如图 11 所示：

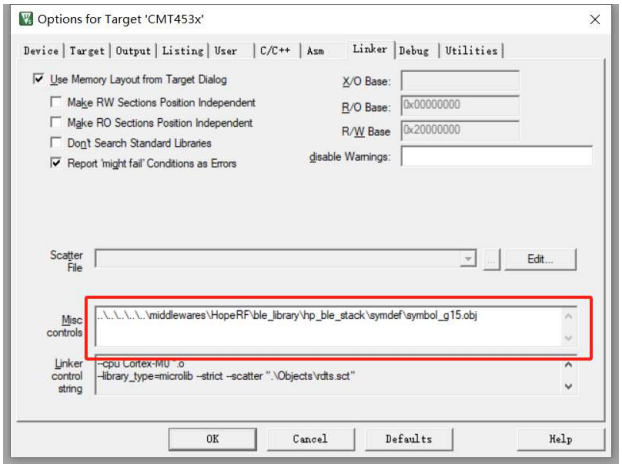



图 11 keil 下库函数符号表路径配置

关于函数符号这里不作说明，有兴趣的读者推荐参考下面这篇文章：

https://blog.csdn.net/ai5945fei/article/details/115897944

(注：本例程中的外部函数符号表位于 middlewares\HopeRF\ble_library\hp_ble_stack\symdef\symbol_g15.txt)

(注意：带主机的工程需要选择 symbol_g15_central.txt)

在 GCC 下同样需要将此外部函数符号表导入到工程中。但是在 GNU 下外部符号表达的格式与 Keil 下却不同。在 GCC 下识别不了.obj 格式，但是可以识别.txt 格式。因此需要对此符号表作下格式转换，这里已经做好转换，读者只需直接导入即可。导入步骤如图 12 所示，请注意文件的路径中父目录与子目录的分隔符要用双斜线“\\”。也可以直接使用相对路径” ../../../../../middlewares/HopeRF/ble_library/hp_ble_stack/symdef/symbol_g15.txt”

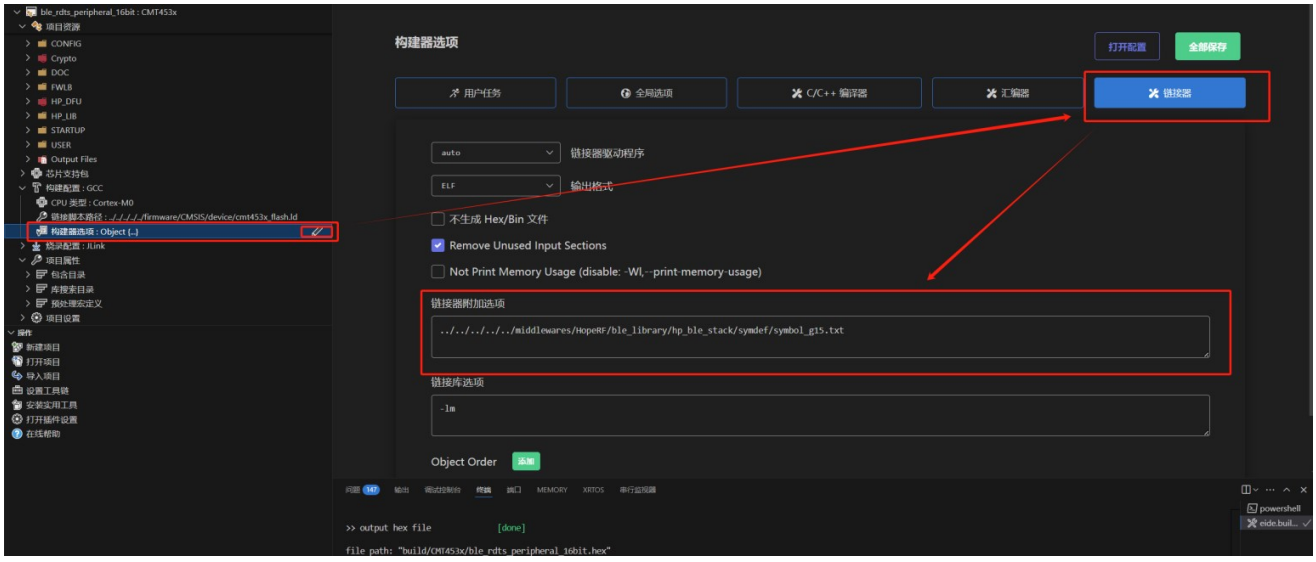



图 13 VSCode 下导入外部函数符号表

安装上述配置好外部符号表后，点击“SAVE ALL”保存配置，重新再编译工程，最终会得到编译成功的提示如图 13 所示。

```
Memory region         Used Size  Region Size  %age Used
           FLASH:       43756 B       256 KB     16.69%
             RAM:        7036 B        32 KB     21.47%

[ INFO ] start outputting files ...

>> output hex file              [done]

file path: "build/CMT453x/ble_rdts_peripheral_16bit.hex"

>> output bin file              [done]

file path: "build/CMT453x/ble_rdts_peripheral_16bit.bin"

[ DONE ] build successfully !, elapsed time 0:0:2
```

图 13 编译成功提示

# 3 下载调试

## 3.1 准备 pack 包

Pack 包在 SDK 目录里 CMT453X_DFP.1.1.1.pack

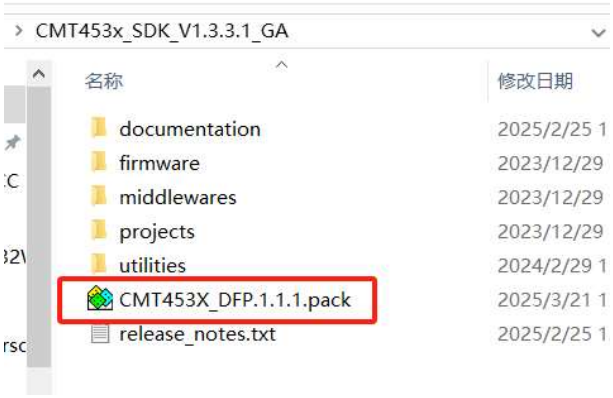



图 14 pack 包

## 3.2 在 Jlink 中添加芯片型号

将如下内容拷贝到 jlink 安装目录的 C:\Program Files\SEGGER\Jlink\JLinkDevices.xml 的末尾，如图 15 所示

```
<Device>
        <ChipInfo Vendor="HOPERF" Name="CMT453x" Core="JLINK_CORE_CORTEX_M0"
```

```
        WorkRAMAddr="0x20000000" WorkRAMSize="0x0000C000" />
        <FlashBankInfo Name="External Flash" BaseAddr="0x01000000" MaxSize="0x00040000" Loader=
        "Devices\HOPERF\CMT453x.FLM" LoaderType="FLASH_ALGO_TYPE_OPEN" AlwaysPresent="1"/>
</Device>
```

```
<Device>
  <ChipInfo Vendor="HOPERF" Name="CMT4531" Core="JLINK_CORE_CORTEX_M0"
  WorkRAMAddr="0x20000000" WorkRAMSize="0x0000C000" />
  <FlashBankInfo Name="External Flash" BaseAddr="0x01000000" MaxSize="0x00040000" Loader=
  "Devices\HOPERF\CMT453x.FLM" LoaderType="FLASH_ALGO_TYPE_OPEN" AlwaysPresent="1"/>
</Device>
```

图 15 jlinkDevices.xml 中添加 CMT453x

在 jlink 安装目录 C:\Program Files\SEGGER\JLink\Devices 下创建一个文件夹 HOPERF，将 CMT453x.FLM 文件拷贝到此文件夹中(utilities\dfu\JLink\JLink_V632\Devices\HopeRF\CMT453x.FLM)，如图 16 所示。

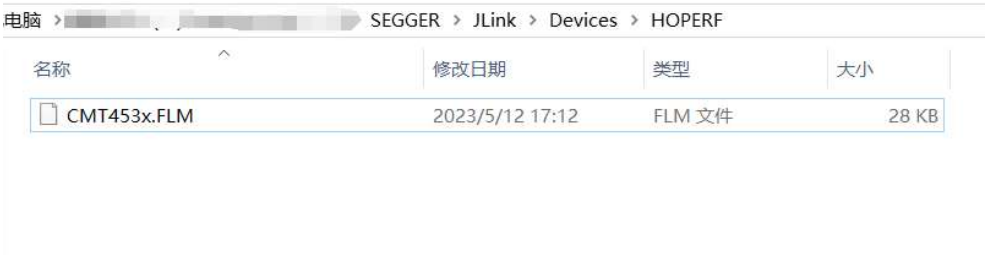



图 16 拷贝 CMT453x.FLM

## 3.3 在 VSCode 中添加芯片支持包

依次选择 EIDE->芯片支持包>From Disk，选择 3.1 章节中 PACK 包的安装路径，将 pack 包导入。如图 17，18 所示，然后再导入芯片型号。

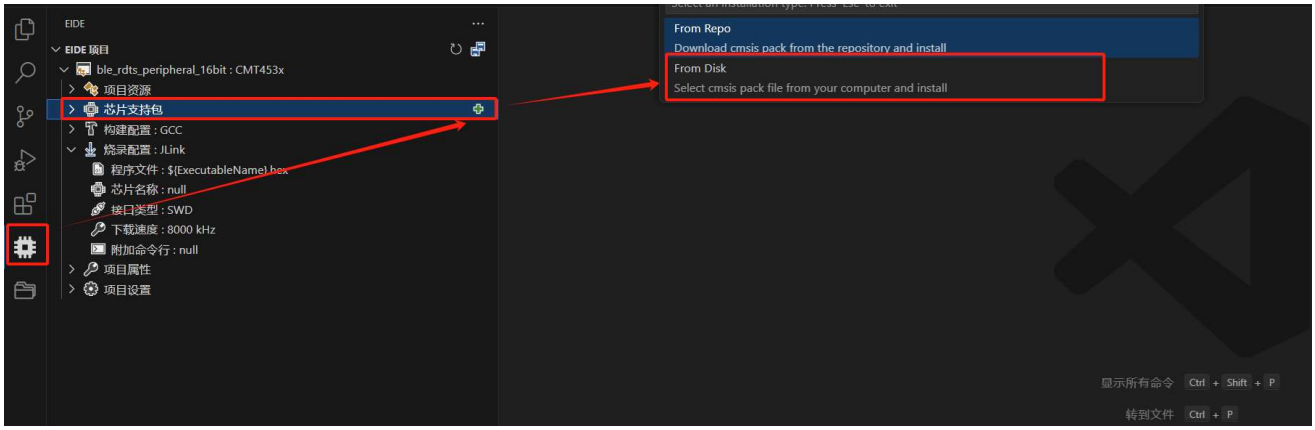



图 17 选择芯片支持包

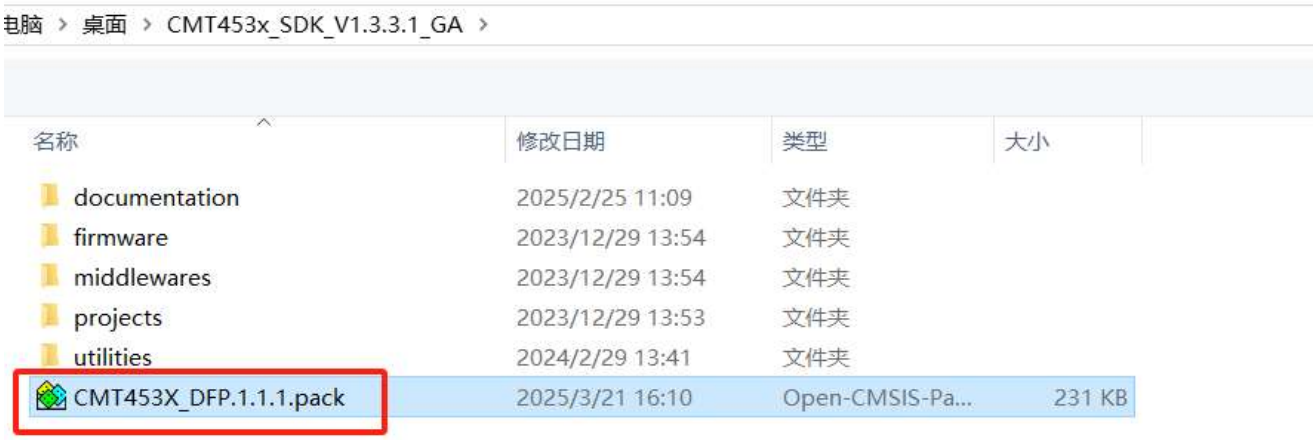



图 18 导入 pack

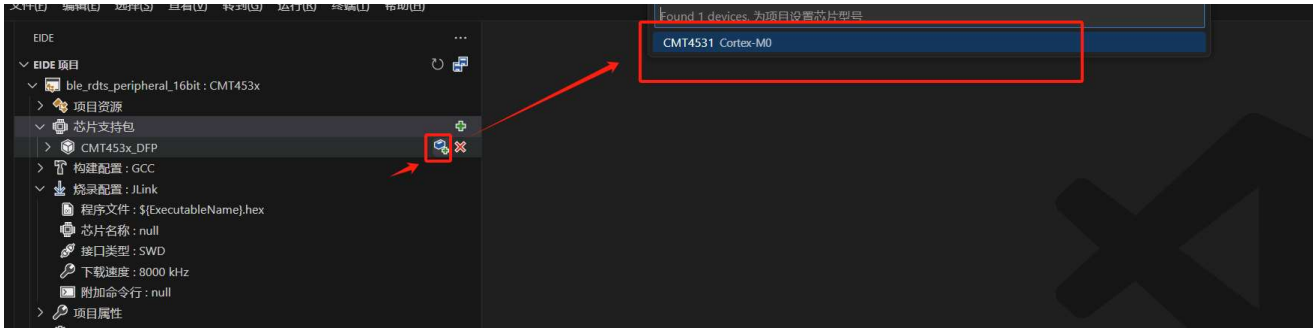



图 19 导入芯片型号

## 3.4 代码下载

代码下载这里选择 jlink，EIDE 中默认路径不一定是用户自己的安装路径，需要手动设置下，如图 20 所示，在 Vscode 搜索栏里搜索 EIDI，然后点击 EIDE 的设置->Extension Settings，找到 jlink 安装目录(Install Directory)设置项，将自己 jlink 实际的安装目录输入进去。

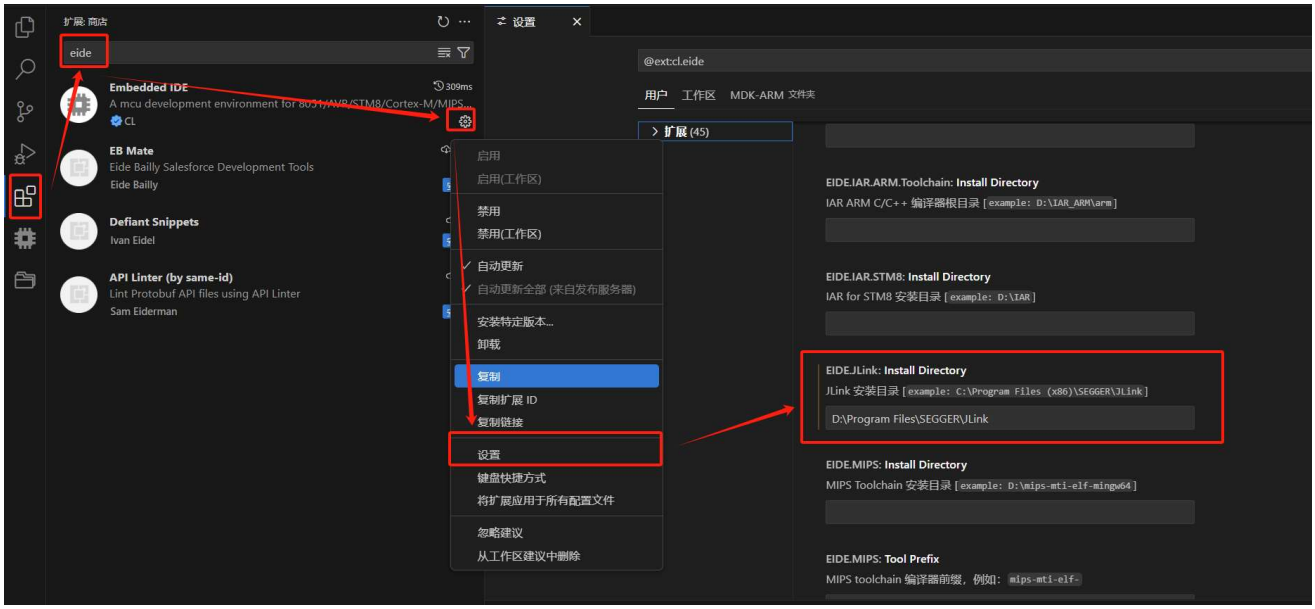
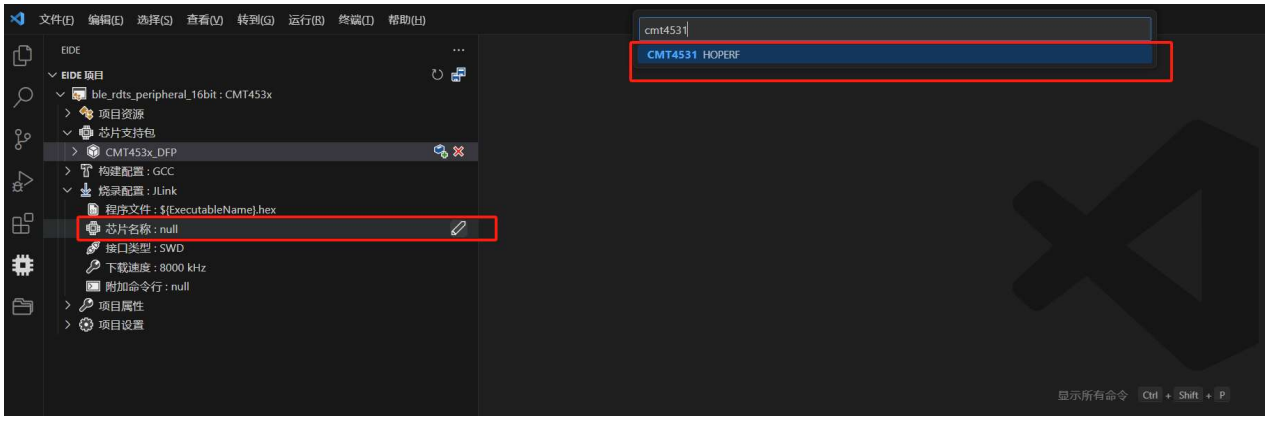



图 20 选择 jlink 安装路径

此时点击下载，会弹出芯片型号未设置的信息，需要将芯片信息设置为 CMT453x




确保以上步骤操作完成，就可以正常进行下载了，点击 VSCode 右上角下载按钮，会弹出如图 20 所示的下载界面：

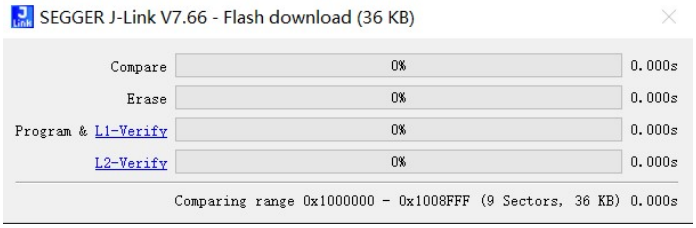



图 21 Jlink 下载界面

```
CPUID register: 0x410CC200. Implementer code: 0x41 (ARM)
Found Cortex-M0 r0p0, Little endian.
FPUnit: 4 code (BP) slots and 0 literal slots
CoreSight components:
ROMTbl[0] @ E00FF000
[0][0]: E000E000 CID B105E00D PID 000BB008 SCS
[0][1]: E0001000 CID B105E00D PID 000BB00A DWT
[0][2]: E0002000 CID B105E00D PID 000BB00B FPB
Cortex-M0 identified.
Reset delay: 0 ms
Reset type NORMAL: Resets core & peripherals via SYSRESETREQ & VECTRESET bit.
Reset: Halt core after reset via DEMCR.VC_CORERESET.
Reset: Reset device via AIRCR.SYSRESETREQ.

PC = 00000100, CycleCnt = 00000000
R0 = FFFFFFFF, R1 = FFFFFFFF, R2 = FFFFFFFF, R3 = FFFFFFFF
R4 = FFFFFFFF, R5 = FFFFFFFF, R6 = FFFFFFFF, R7 = FFFFFFFF
R8 = FFFFFFFF, R9 = FFFFFFFF, R10= FFFFFFFF, R11= FFFFFFFF
R12= FFFFFFFF
SP(R13)= 2000C000, MSP= 2000C000, PSP= FFFFFFFC, R14(LR) = FFFFFFFF
XPSR = C1000000: APSR = NZcvq, EPSR = 01000000, IPSR = 000 (NoException)
CFBP = 00000000, CONTROL = 00, FAULTMASK = 00, BASEPRI = 00, PRIMASK = 00
FPU regs: FPU not enabled / not implemented on connected CPU.

Downloading file [c:\Users\ZF\Desktop\CMT453x_SDK_V1.3.3.1_GA\projects\cmt453x_EVAL\ble_peripheral\ble_rdts_peripheral_16bit\MDK-ARM\build\CMT453x\ble_rdts_peripheral_16bit.hex]...
J-Link: Flash download: Bank 0 @ 0x01000000: 1 range affected (45056 bytes)
J-Link: Flash download: Total: 2.720s (Prepare: 0.058s, Compare: 0.155s, Erase: 0.295s, Program: 1.247s, Verify: 0.949s, Restore: 0.012s)
J-Link: Flash download: Program speed: 35 KB/s
O.K.

Reset delay: 0 ms
Reset type NORMAL: Resets core & peripherals via SYSRESETREQ & VECTRESET bit.
Reset: Halt core after reset via DEMCR.VC_CORERESET.
Reset: Reset device via AIRCR.SYSRESETREQ.
```

图 22 完成下载界面

## 3.5 测试结果

修改蓝牙设备名称为 HP_RDTS_P_20250324，打开手机搜索蓝牙，能找到名称为 HP_RDTS_P_20250324 的蓝牙设备，如图 23，图 24 所示。

```
8
9   #ifndef _APP_USER_CONFIG_H_
0   #define _APP_USER_CONFIG_H_
1
2   #include "hp_adv_data_def.h"
3
4   /* Device name */
5   #define CUSTOM_DEVICE_NAME                  "HP_RDTS_P_20250324"
6
7   /* adv configer*/
8   #define CUSTOM_ADV_FAST_INTERVAL              160
9   #define CUSTOM_ADV_SLOW_INTERVAL              3200
```

图 23

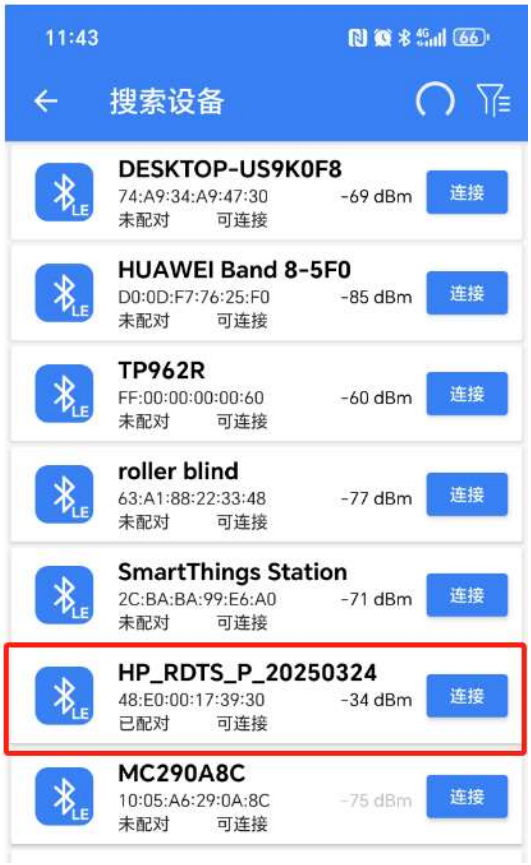

11:43
搜索设备
DESKTOP-US9K0F8
74:A9:34:A9:47:30
-69 dBm
未配对 可连接
连接
HUAWEI Band 8-5F0
D0:0D:F7:76:25:F0
-85 dBm
未配对 可连接
连接
TP962R
FF:00:00:00:00:60
-60 dBm
未配对 可连接
连接
roller blind
63:A1:88:22:33:48
-77 dBm
未配对 可连接
连接
SmartThings Station
2C:BA:BA:99:E6:A0
-71 dBm
未配对 可连接
连接
HP_RDTS_P_20250324
48:E0:00:17:39:30
-34 dBm
已配对 可连接
连接
MC290A8C
10:05:A6:29:0A:8C
-75 dBm
未配对 可连接
连接

图 24

# 历史版本

| 版本 | 日期 | 备注 |
| --- | --- | --- |
| V1.0 | 2025.3.24 | 新建文档 |