GR5526开发者指南

版本： **1.0**

发布日期： **2023-01-10**

深圳市汇顶科技股份有限公司

版权所有 © 2023 深圳市汇顶科技股份有限公司。保留一切权利。

非经本公司书面许可，任何单位和个人不得对本手册内的任何部分擅自摘抄、复制、修改、翻译、传播，或将其全部或部分用于商业用途。

商标声明

GOODiX 和其他汇顶商标均为深圳市汇顶科技股份有限公司的商标。本文档提及的其他所有商标或注册商标，由各自的所有人持有。

免责声明

本文档中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。

深圳市汇顶科技股份有限公司（以下简称“GOODIX”）对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。GOODIX对因这些信息及使用这些信息而引起的后果不承担任何责任。

未经GOODIX书面批准，不得将GOODIX的产品用作生命维持系统中的关键组件。在GOODIX知识产权保护下，不得暗中或以其他方式转让任何许可证。

深圳市汇顶科技股份有限公司

总部地址：深圳市福田保税区腾飞工业大厦B座12-13层

电话：+86-755-33338828　　邮编：518000

网址：www.goodix.com

前言

编写目的

本文档主要介绍Goodix GR5526低功耗蓝牙片上系统（SoC）的软件开发工具包（SDK），以及使用Keil开发和调试程序的方法，以帮助开发者开发低功耗蓝牙（Bluetooth Low Energy，BLE）应用。

读者对象

本文适用于以下读者：

- GR5526用户
- GR5526开发人员
- GR5526测试人员
- 文档工程师

版本说明

本手册为第1次发布，对应的产品系列为GR5526。

修订记录

版本	日期	修订内容
1.0	2023-01-10	首次发布

版权所有 © 2023 深圳市汇顶科技股份有限公司

目录

前言 I

1 简介 1

1.1 GR5526 SDK 1

1.2 低功耗蓝牙协议栈 1

2 GR5526低功耗蓝牙软件平台 4

2.1 硬件架构 4

2.2 软件架构 5

2.3 存储器映射 6

2.4 Flash存储映射 8

2.4.1 SCA 9

2.4.2 NVDS 11

2.5 RAM存储映射 13

2.5.1 XIP模式的典型RAM布局 14

2.5.2 Mirror模式的典型RAM布局 15

2.5.3 RAM电源管理 16

2.6 PSRAM介绍 17

2.7 GR5526 SDK目录结构 18

3 启动流程（Bootloader） 21

4 使用SDK开发调试 23

4.1 安装Keil 23

4.2 安装GR5526 SDK 24

4.3 创建BLE Application 24

4.3.1 准备ble_app_example 24

4.3.2 配置工程 27

4.3.2.1 配置custom_config.h与ble_basic_config.h 27

4.3.2.2 配置存储器布局 32

4.3.2.3 配置After Build 33

4.3.3 添加用户代码 33

4.3.3.1 修改主函数 33

4.3.3.2 实现BLE业务逻辑 34

4.3.3.3 BLE_Stack_IRQ、BLE_SDK_IRQ与Application的调度机制 39

4.4 生成固件 39

4.5 下载.hex文件到Flash 40

4.6 调试 42

4.6.1 配置调试器 42

4.6.2 启动调试 44

版权所有 © 2023 深圳市汇顶科技股份有限公司

4.6.3 输出调试Log.....44
4.6.3.1 模块初始化.....45
4.6.3.2 使用方法.....46
4.6.4 使用GRToolbox调试.....47
5 术语和缩略语.....48

版权所有 © 2023 深圳市汇顶科技股份有限公司

1 简介

GR5526系列芯片是Goodix推出的支持Bluetooth 5.3的单模低功耗蓝牙片上系统（SoC）芯片，可以配置为广播者（Broadcaster）、观察者（Observer）、外围设备（Peripheral）和中央设备（Central），并支持上述各种角色的组合应用，以及支持Bluetooth LE（Bluetooth Low Energy，低功耗蓝牙）定向（AoA/AoD）、等时通道（音频）等功能，可广泛应用于物联网（IoT）、低功耗音频（LE Audio）和智能穿戴设备领域。

GR5526系列芯片架构以ARM® Cortex®-M4F CPU为核心，集成Bluetooth 5.3协议栈、2.4 GHz RF收发器、片上可编程存储器Flash、RAM以及多种外设，提供更丰富的I2C/UART接口数量与I/O功能。GR5526 部分型号还提供图形化处理单元+显示控制器（GPU + DC）解决方案，并支持内/外部 SiP PSRAM空间，为用户提供更丰富的数据空间与强大的图形化表现能力，并为可穿戴设备方案提供丰富的片上资源。

GR5526系列支持BGA83和QFN68两种封装，如下表所示。用户可根据实际应用场景，选择对应芯片型号。

表 1-1 GR5526系列芯片配置

特性	GR5526VGBIP	GR5526VGBI	GR5526RGNIP	GR5526RGNI
CPU	Cortex® -M4F	Cortex® -M4F	Cortex® -M4F	Cortex® -M4F
RAM	512 KB	512 KB	512 KB	512 KB
SiP Flash	1 MB	1 MB	1 MB	1 MB
SiP PSRAM	8 MB	N/A	8 MB	N/A
GPU + DC	Yes	N/A	Yes	N/A
I/O数量	50	50	48	48
封装 （mm）	BGA83 (4.3 x 4.3 x 0.96)	BGA83 (4.3 x 4.3 x 0.96)	QFN68 (7.0 x 7.0 x 0.85)	QFN68 (7.0 x 7.0 x 0.85)

1.1 GR5526 SDK

GR5526软件开发工具包（Software Development Kit，SDK）为GR5526系列SoC提供全面的软件开发支持。该工具包中包含BLE（Bluetooth Low Energy）Stack API、LE Audio API、System API、RTLS（Real Time Location Service）API、外设驱动程序、hex文件生成和下载工具、工程示例代码以及相关的用户文档等。

本文档支持的GR5526 SDK版本，适用于GR5526系列的所有芯片。

1.2 低功耗蓝牙协议栈

低功耗蓝牙（BLE）协议栈的架构如图 1-1所示。

版权所有 © 2023 深圳市汇顶科技股份有限公司

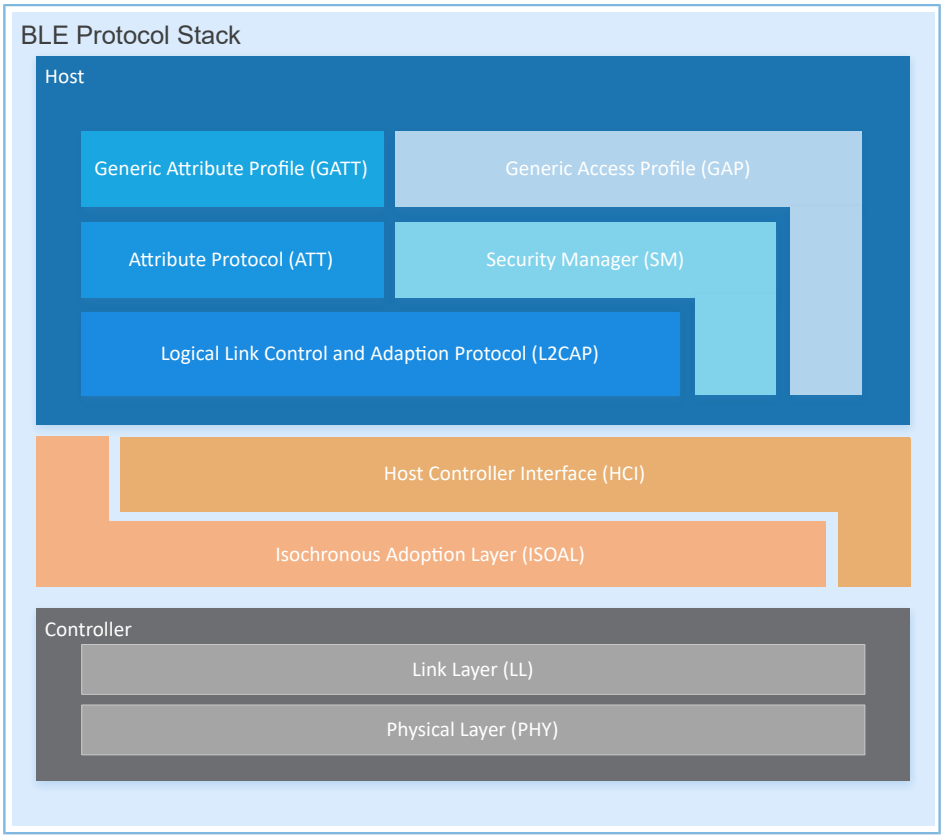

图 1-1 BLE协议栈架构

BLE协议栈由控制器（Controller）、同步适配层（ISOAL）、主机控制接口（HCI）和主机（Host）组成。

控制器（Controller）

- 物理层（Physical Layer，PHY）支持1 Mbps和2 Mbps的自适应跳频GFSK（高斯频移键控）射频（RF）操作。
- 链路层（Link Layer，LL），控制设备的射频状态，设备可以处于如下的五种状态，根据应用需求相互切换：Standby、Advertising、Scanning、Initiating或者Connection。

同步适配层（ISOAL）

- 同步适配层（Isochronous Adaptation Layer， ISOAL）提供同步数据在Host和Controller之间的适配功能。该层将分段的数据帧组装成可供应用层使用的数据流，或是将应用层提供的数据流分段成数据帧通过空口传输。

主机控制接口（HCI）

- 主机控制接口（Host-Controller Interface，HCI）提供Host与Controller之间的通信。该接口层的实现可以是软件接口，也可以是标准硬件接口，比如UART，Secure Digital（SD）或USB。HCI commands和events通过这个接口层在Host与Controller之间传递。

主机（Host）

- 逻辑链路控制和适配协议（Logical Link Control and Adaption Protocol，L2CAP）为上层提供多路复用、数据分段与重组服务，并且支持逻辑端对端的数据通信。

版权所有 © 2023 深圳市汇顶科技股份有限公司

- 安全管理层（Security Manager，SM）定义配对和密钥分发的方法，为上层协议栈和应用程序提供端到端的安全连接和数据交换的功能。
- 通用访问规范层（Generic Access Profile，GAP）为上层应用和Profiles提供和协议栈通信交互的接口，主要包括广播、扫描、连接发起、服务发现、连接参数更新、安全过程发起和响应的相关功能。
- 属性协议层（Attribute Protocol，ATT）定义服务端和客户端之间的服务数据交互协议。
- 通用属性规范层（Generic Attribute Profile，GATT）基于ATT协议之上，定义一系列用于GATT Client和GATT Server之间服务数据交互的通信过程，供上层应用、Profile和Service进行使用。

更多BLE技术及其协议的资料请访问Bluetooth SIG的官方网站www.bluetooth.com获取。

GAP、SM、L2CAP和GATT的规范包含在Bluetooth Core Spec中。其他BLE应用层Profiles/Services规范可以在GATT Specs页面下载。BLE应用可能会用到的Assigned Numbers，IDs和Codes都列在Assigned Numbers页面。

版权所有 © 2023 深圳市汇顶科技股份有限公司

2 GR5526低功耗蓝牙软件平台

GR5526 SDK是一套基于GR5526芯片定义的低功耗蓝牙应用开发的软件套件，包括BLE 5.3 API、System API和外设驱动API接口，并提供丰富的蓝牙和外设应用示例工程和使用说明文档。应用开发者可以基于GR5526 SDK的示例工程进行快速产品开发和迭代。

2.1 硬件架构

GR5526的硬件框图如下。本节将简要介绍芯片内各模块，更多的详细资料请参考《GR5526 Datasheet》。

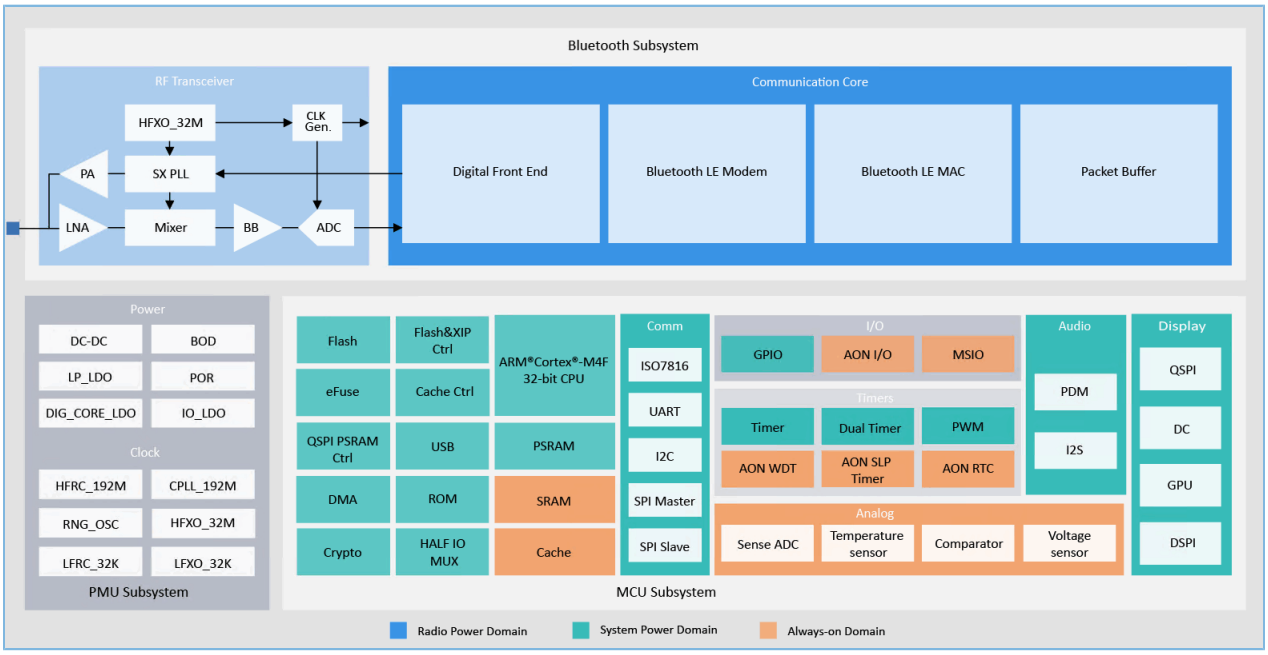

图 2-1 GR5526硬件架构

- ARM® Cortex®-M4F：GR5526 SoC芯片的系统核心处理器（CPU）。BLE协议栈和Application代码都运行在该处理器上。
- RAM：随机存取存储器，提供程序执行时需要的内存空间。
- ROM：只读存储器，固化Bootloader，BLE协议栈的软件部分。
- Security Cores：安全计算引擎单元，主要包括TRNG、AES、SHA和PKC等模块，提供对加密的用户应用Firmware进行校验的功能。对加密Firmware的校验是通过ROM中的安全启动流程完成的（Bluetooth SPEC中与安全相关的计算单元是包含于Communication Core中的独立模块，与Security Cores无关）。
- Peripherals：GPIO、DMA、I2C、I2S、SPI、QSPI、DSPI、OSPI、UART、PWM、Timer、GPU、DC等硬件。
- RF Transceiver：2.4 GHz射频信号收发器。
- Communication Core：Bluetooth 5.3协议栈控制器的物理层。它也是软件协议栈与2.4 GHz射频硬件之间的接口。

版权所有 © 2023 深圳市汇顶科技股份有限公司

- PMU（Power Management Unit）：电源管理单元，为各系统模块提供电源供应，根据配置参数和当前的运行状态，设定合理的DC/DC、IO-LDO、Dig-LDO、RF Subsystem等模块参数。
- Flash：封装在芯片内部的Flash存储单元，用于用户代码和数据存储，支持用户代码片上运行模式（Execute in Place，XIP）。

2.2 软件架构

图 2-2展示了GR5526 SDK的软件架构。

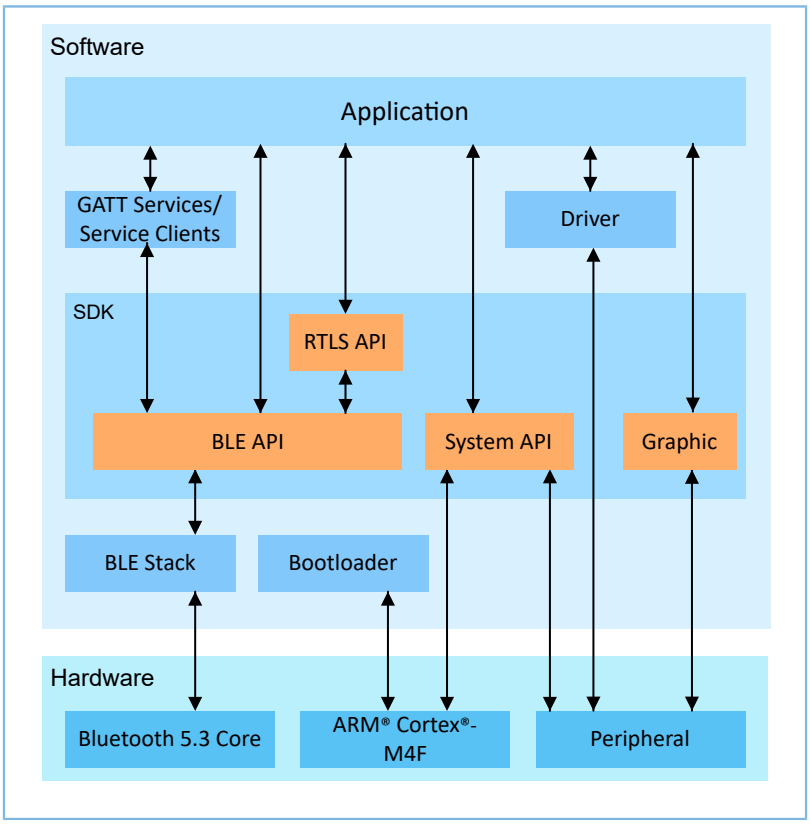

图 2-2 GR5526软件架构

- Bootloader

  固化在芯片中的引导程序，负责初始化芯片的软硬件环境，校验并启动应用程序。

- BLE Stack

  低功耗蓝牙协议栈实现核心，由控制器（Controller）、同步适配层（ISOAL）、主机控制接口（HCI）和主机（Host）协议组成（包括ATT、L2CAP、GAP、SM、GATT），支持Broadcaster、Observer、Peripheral和Central角色。

- BLE SDK

  软件开发工具包，提供易于使用的SDK BLE API、SDK System API和SDK RTLS API。

  - SDK BLE API包括L2CAP、GAP、SM和GATT API，LE Audio API也包含在其中。
  - SDK System API提供了对非易失性数据存储系统（NVDS）、Firmware升级（Device Firmware Update，DFU）、系统电源管理以及通用系统级访问接口的API定义。

版权所有 © 2023 深圳市汇顶科技股份有限公司

  - SDK RTLS API用于支持AoA（Angle of Arrival）和AoD（Angle of Departure）功能。RTLS独立封装lib库，向上与应用层进行数据交互，如接受定位参数配置、接收定位控制命令、上报定位原始数据和上报定位计算结果等；向下依赖SDK BLE API，获取Iq数据上报事件。此外还包含Music、DBSCAN等算法处理内容。

- Application

  在SDK包中，提供了丰富的蓝牙及外设示例工程，每个示例工程中都包含编译后的二进制文件，用户可以下载到芯片中运行和测试。对于大部分的蓝牙应用，SDK包中的安卓APP也提供了对应的功能，方便用户测试。

- Drivers

  外设驱动部分的API定义及说明。

- Graphic

  GPU显示驱动模块SDK库。

2.3 存储器映射

图 2-3展示GR5526系列SoC的存储器映射。

版权所有 © 2023 深圳市汇顶科技股份有限公司

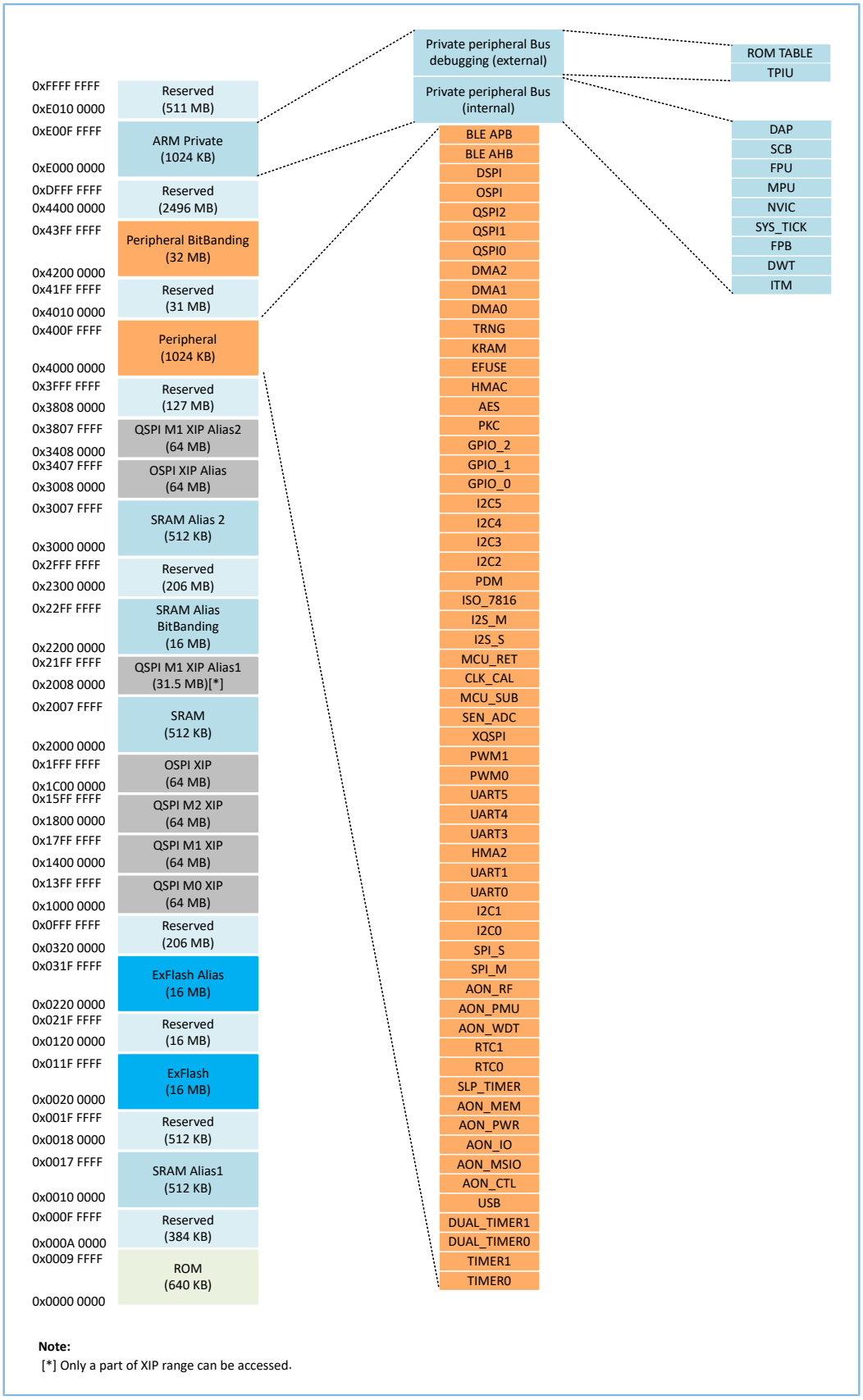

图 2-3 GR5526存储器映射

版权所有 © 2023 深圳市汇顶科技股份有限公司

- RAM存储：0x0010_0000 ～ 0x0017_FFFF或0x2000_0000 ～ 0x2007_FFFF或0x3000_0000 ～ 0x3007_FFFF，共512 KB。
  - 0x2000_0000 ～ 0x2007_FFFF：该区域支持位段操作，它对应的位段地址为0x2200_0000 ～ 0x2207_FFFF，该区域可进行数据原子操作。SDK中RW、ZI、HEAP、STACK等变量位于该区域。
  - 0x0010_0000 ～ 0x0017_FFFF：由于 Cortex-M4F 总线架构的特点该区域的访问效率高于其他区域，故SDK中RAM_CODE可执行代码位于该区域。

说明:

QSPI0/QSPI1/QSPI2/OSPI均支持XIP模式。在这种模式下，可以将Flash或PSRAM的数据空间映射到内存中，方便直接对内存地址进行操作。

- 在外置PSRAM方案中，QSPI1上挂载的外置PSRAM可以与0x2000 0000 ～ 0x2007 FFFF的内存结合，形成一个连续的SRAM地址空间。
- 在内部PSRAM方案中，OSPI上挂载的内部PSRAM可以与0x3000 0000 ～ 0x3007 FFFF存储区结合，形成一个连续的SRAM地址空间。

- Flash存储：0x0020_0000 ～ 0x011F_FFFF或0x0220_0000 ～ 0x031F_FFFF，共16 MB。
  - 0x0020_0000 ～ 0x011F_FFFF区域存放代码以及非加密模式下的数据。
  - 0x0220_0000 ～ 0x031F_FFFF区域存放加密模式下的数据。

说明:

GR5526芯片内部Flash封装为1 MB，地址为0x0020_0000 ～ 0x002F_FFFF。

2.4 Flash存储映射

GR5526封装了一个采用XQSPI总线接口的可擦除外部Flash存储器。该Flash物理上由若干个4 KB大小的Flash Sector组成；逻辑上可根据不同的应用场景，分成不同用途的存储区域。

图 2-4为GR5526典型应用场景的Flash存储布局。

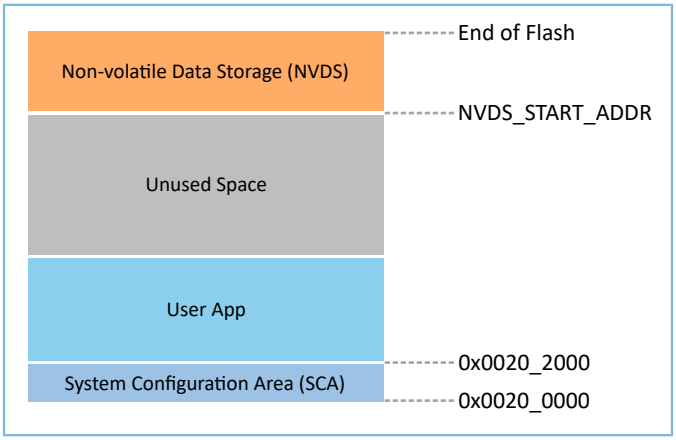

图 2-4 Flash存储布局

版权所有 © 2023 深圳市汇顶科技股份有限公司

- System Configuration Area（SCA）：系统配置区，主要用于存储系统启动参数配置信息。
- User App：Application Firmware存储区域。
- Unused Space：空闲区域。开发者可以自行使用该区域。比如，在DFU升级过程中，用Unused Space临时存储新的Application Firmware。
- Non-volatile Data Storage（NVDS）：非易失性数据存储区域。

说明:

NVDS缺省占用Flash的最后两个扇区（Sector）。开发者也可以根据产品的Flash使用布局，合理安排NVDS的起始地址以及所占用的扇区数量，具体配置方法参考4.3.2.1 配置custom_config.h与ble_basic_config.h。

注意：NVDS起始地址需要和Flash扇区的起始地址对齐。

2.4.1 SCA

System Configuration Area（SCA）位于Flash的前两个Sector（共8 KB，0x0020_0000 ～ 0x0020_2000）。SCA中存储了系统启动过程使用的标志以及其他系统配置参数。SDK toolchain链接工具会根据Application Firmware中的BUILD_IN_APP_INFO结构体（路径：`Project_Folder\platform\soc\common\gr_platform.c`）生成SCA Image，并将其烧写到SCA，Bootloader会根据SCA区域中的启动信息，校验和跳转到Firmware的入口地址。BUILD_IN_APP_INFO的定义和配置如下：

说明:

Project_Folder为工程的根目录。

```
const APP_INFO_t BUILD_IN_APP_INFO __attribute__((section(".app_info"))) =
#endif
{
    .app_pattern      = APP_INFO_PATTERN_VALUE,
    .app_info_version = APP_INFO_VERSION,
    .chip_ver         = CHIP_VER,
    .load_addr        = APP_CODE_LOAD_ADDR,
    .run_addr         = APP_CODE_RUN_ADDR,
    .app_info_sum     = CHECK_SUM,
    .check_img        = BOOT_CHECK_IMAGE,
    .boot_delay       = BOOT_LONG_TIME,
    .sec_cfg          = SECURITY_CFG_VAL,
#ifdef APP_INFO_COMMENTS
    .comments         = APP_INFO_COMMENTS,
#endif
};
```

- app_pattern：固定值0x47525858。
- app_info_version：标识固件信息版本，与APP_INFO_VERSION对应，当前版本为1。
- chip_ver：标识此固件使用的芯片版本，与*custom_config.h*中的CHIP_VER对应。

版权所有 © 2023 深圳市汇顶科技股份有限公司

- load_addr：Firmware存储地址，与*custom_config.h*中的APP_CODE_LOAD_ADDR对应。
- run_addr：Firmware运行地址，与*custom_config.h*中的APP_CODE_RUN_ADDR对应。
- app_info_sum：Firmware信息的校验和，由CHECK_SUM宏自动计算。
- check_img：启动配置参数，与*custom_config.h*中的BOOT_CHECK_IMAGE对应，当此参数配置为1时，启动时将会对Firmware进行校验。
- boot_delay：启动配置参数，与*custom_config.h*中的BOOT_LONG_TIME对应，当此参数配置为1时，冷启动时将会加入1 秒延时。
- sec_cfg：安全配置参数，暂未使用。
- reserved0：保留字段。
- comments：固件描述信息，最长支持12 字节信息。
- reserved1：保留字段。

图 2-5为System Configuration Area布局。

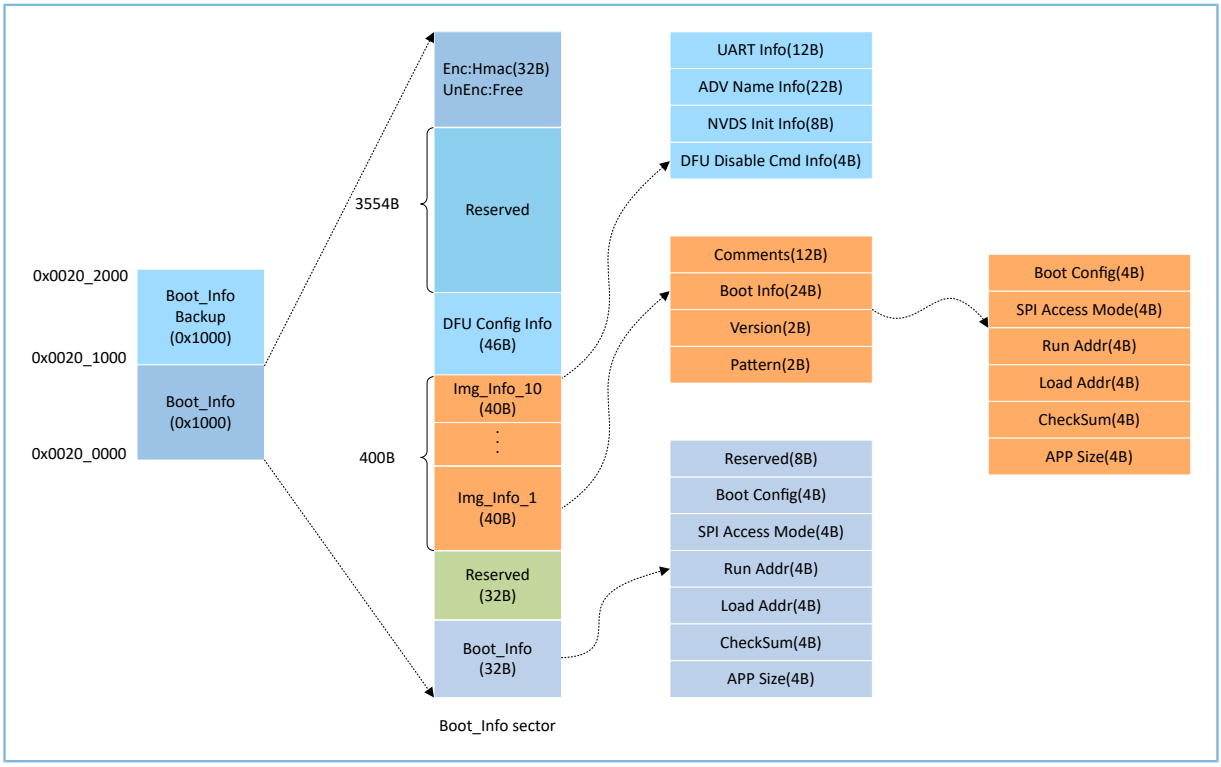

图 2-5 System Configuration Area布局

- Boot_Info与Boot_Info Backup存储相同的信息，Boot_Info Backup是Boot_Info的备份。
  - 在非安全模式下，Bootloader会默认从Boot_Info中获取启动信息。
  - 在安全模式下，Bootloader会先校验Boot_Info，如果Boot_Info校验不过，则会校验Boot_Info Backup，并从Boot_Info Backup中获取启动信息。
- Boot_Info（32B）区域中存储Firmware启动信息，Bootloader会根据启动信息，校验和跳转到Firmware的入口地址。

版权所有 © 2023 深圳市汇顶科技股份有限公司

  - Boot Config为系统启动配置信息。
  - SPI Access Mode为SPI访问方式配置。为系统固定配置，用户无法修改。
  - Run Addr为Firmware运行地址，与BUILD_IN_APP_INFO中的run_addr对应。
  - Load Addr为Firmware存储地址，与BUILD_IN_APP_INFO中的load_addr对应。
  - CheckSum为Firmware校验和，生成Firmware后，由SDK toolchain自动算出。
  - APP Size为Firmware的Size信息，生成Firmware后，由SDK toolchain自动算出。
- Img_Info区域存储最多10个Firmware的信息。当使用GProgrammer下载Firmware或使用DFU升级Firmware时，Firmware信息会被存储到Img_Info区域。
  - Comments为Firmware描述信息，支持最多12个字符。生成Firmware后，SDK toolchain使用Firmware文件名作为Comments信息。
  - Boot Info（24B）为Firmware启动信息，与上述Boot_Info（32B）的低24 Byte相同。
  - Version为Firmware版本信息，与*custom_config.h*中VERSION对应。
  - Pattern为固定值0x4744。
- DFU Config Info区域存储ROM中DFU模块的配置信息。
  - UART Info为DFU模块的UART串口相关配置，包括状态位、波特率、GPIO配置等。
  - ADV Name Info为DFU模块的广播相关配置，包括状态位、广播名、广播长度。
  - NVDS Init Info为DFU模块的NVDS系统的初始化配置，包括状态位、NVDS区域大小、起始地址。
  - DFU Disable Cmd Info为DFU模块的DFU禁用命令配置，包括状态位和Disable DFU Cmd（2B，设置格式为Bitmask），可通过设置Disable DFU Cmd值来禁用某些DFU命令。
- Hmac区域存储Hmac校验值。该区域仅在安全模式下有效。

2.4.2 NVDS

NVDS是一个轻量级逻辑数据存储系统，它依赖于Flash硬件抽象层（Flash HAL）。其存储于Flash中，掉电时数据不会丢失。NVDS默认使用Flash最后两个扇区（Sector）。在NVDS区域中，最后一个扇区用于碎片整理，其余扇区用于数据存储。

NVDS系统适合存储小块数据，例如应用程序的配置参数、校准数据、状态和用户信息等。BLE协议栈也会使用NVDS存储设备绑定等参数。

NVDS系统具有以下特性：

- 每个存储项（TAG）都有唯一的TAG ID用于标识，用户程序可以根据TAG ID对数据内容进行读取和更改操作，不需要关心存储的物理地址。
- 针对Flash存储介质的特性进行了优化，支持数据校验、Word对齐、碎片整理和擦写平衡。
- 存储区域的大小和起始地址可配置，Flash存储区以Sector为单位，一个Sector的大小为4 KB，NVDS存储区域可配置为若干个Sector；配置的起始地址须按4 KB对齐。

版权所有 © 2023 深圳市汇顶科技股份有限公司

NVDS提供了以下8个简单的API用于操作Flash中的非易失性数据。

表 2-1 NVDS API

函数原型	描述
uint8_t nvds_init(uint32_t start_addr, uint8_t sectors)	初始化NVDS使用的Flash Sectors。
uint8_t nvds_get(NvdsTag_t tag, uint16_t *p_len, uint8_t *p_buf)	从NVDS中读出tag标识对应的数据。
uint8_t nvds_put(NvdsTag_t tag, uint16_t len, const uint8_t *p_buf)	将数据写入到NVDS并用tag标识。如果这个tag不存在，则创建一个新的。
uint8_t nvds_del(NvdsTag_t tag)	删除NVDS中tag标识的数据。
uint16_t nvds_tag_length(NvdsTag_t tag)	获取指定Tag的数据长度。
uint8_t nvds_drv_func_replace(nvds_drv_func_t *p_nvds_drv_func)	替代直接操作Flash的相关API。
uint8_t nvds_func_replace(nvds_func_t *p_nvds_func)	替代NVDS操作相关API。
void nvds_retention_size(uint8_t bond_dev_num)	为设备绑定信息预留空间，预留空间大小与设备绑定数有关。

NVDS API的详细说明请直接查看NVDS头文件（位于`SDK_Folder\components\sdk\gr55xx_nvds.h`）。

NVDS存储数据的格式如图 2-6所示：

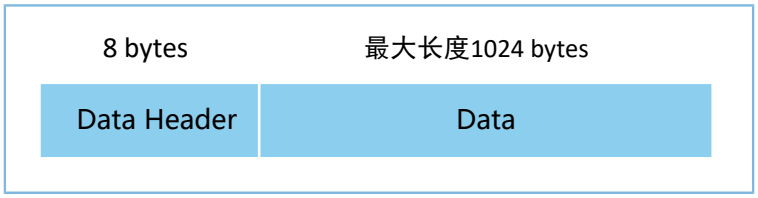

图 2-6 NVDS存储数据格式

数据头（Data Header）格式如表 2-2 所示：

表 2-2 Data Header格式

Byte	Name	Description
0-1	tag	数据的tag标识
2-3	len	数据的长度
4-4	checksum	数据头的校验和
5-5	value_cs	数据的校验和
6-7	reserved	保留字段

版权所有 © 2023 深圳市汇顶科技股份有限公司

说明:

BLE协议栈也会使用NVDS存储一些参数，因此必须为NVDS分配Flash存储区域。GR5526 SDK会缺省使用Flash Memory倒数第二个Sector作为NVDS使用的存储区域，最后一个Sector用作NVDS碎片整理，可通过更改*custom_config.h*文件中的宏NVDS_START_ADDR和NVDS_NUM_SECTOR来配置NVDS区域的起始地址和占用大小。BLE协议栈和Application共享相同的NVDS存储区域，但是TAG ID命名空间被划分为不同的类别，开发者只能使用分配给Application的TAG ID命名类别。

- Application必须使用NV_TAG_APP(idx)来获取应用程序数据的TAG ID。该TAG ID被用作NVDS API的参数。
- Application不能将idx直接作为NVDS API的参数。idx取值范围是0x4000 ～ 0x7FFF。

在Application第一次运行前，开发者可使用工具GProgrammar将BLE协议栈和Application所使用的TAG初始值写入到NVDS。如果开发者不使用GR5526 SDK缺省的NVDS区域而要自行指定NVDS区域起始地址，请确保GProgrammar中的NVDS区域起始地址配置与4 KB对齐。

2.5 RAM存储映射

GR5526的RAM为512 KB，起始地址为0x3000_0000，由11个内存块（RAM Block）组成（前2个内存块的大小均为16 KB，第3个和第4个内存块大小为32 KB，其后的6个内存块大小为64 KB，最后的内存块大小为32 KB）。每个RAM内存块都可由软件独立打开/关闭电源。

说明:

GR5526为起始地址0x3000_0000的RAM提供一个起始地址为0x0010_0000和0x2000_0000的Aliasing Memory，见图 2-3。

- 起始地址为0x2000_0000区域支持位段操作，它对应的位段起始地址是0x2200_0000。
- 由于 Cortex®-M4F总线架构的特点起始地址为0x0010_0000区域的访问效率高于其他区域，故代码的运行地址在0x0010_0000区域地址范围，可以加快运行速度。
- GR5526 SDK中RW、ZI、HEAP、STACK使用0x2000_0000区域的RAM，RAM_CODE可执行代码使用0x0010_0000区域的RAM。

512 KB RAM存储布局如图 2-7所示：

版权所有 © 2023 深圳市汇顶科技股份有限公司

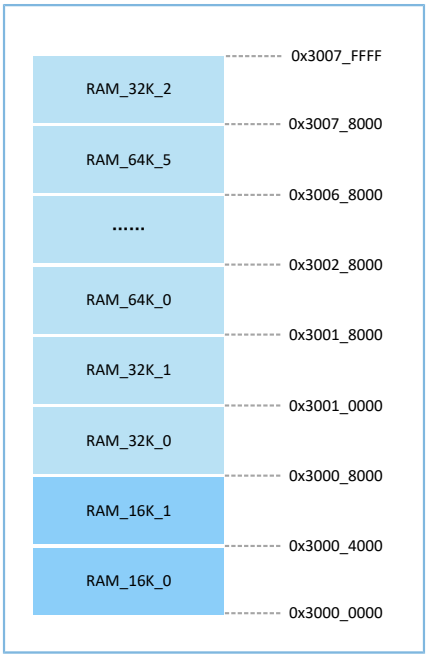

图 2-7 512 KB RAM存储布局

程序运行模式可配置为Execute in Place（XIP）模式或Mirror模式。详细的配置方法参考4.3.2.1 配置custom_config.h与ble_basic_config.h中的“APP_CODE_RUN_ADDR”。这两种运行模式有不同的RAM布局。

表 2-3 程序运行模式

运行模式	描述
XIP模式	片上运行模式，用户应用程序存储在片上Flash空间，程序运行空间和加载空间相同。系统完成上电配置后，通过Cache Controller直接从Flash空间取指运行。
Mirror模式	镜像运行模式，用户应用程序存储在片上Flash空间，程序的运行空间定义在RAM空间。在程序启动阶段，会在校验完成后，将程序从外部Flash空间加载到RAM空间，并跳转到RAM中进行运行。

说明:

由于XIP模式运行时需要持续访问Flash，因此该模式下的运行功耗会略高于Mirror模式。

2.5.1 XIP模式的典型RAM布局

图 2-8为XIP模式的典型RAM布局，开发者可以根据产品需要对其进行修改。

版权所有 © 2023 深圳市汇顶科技股份有限公司

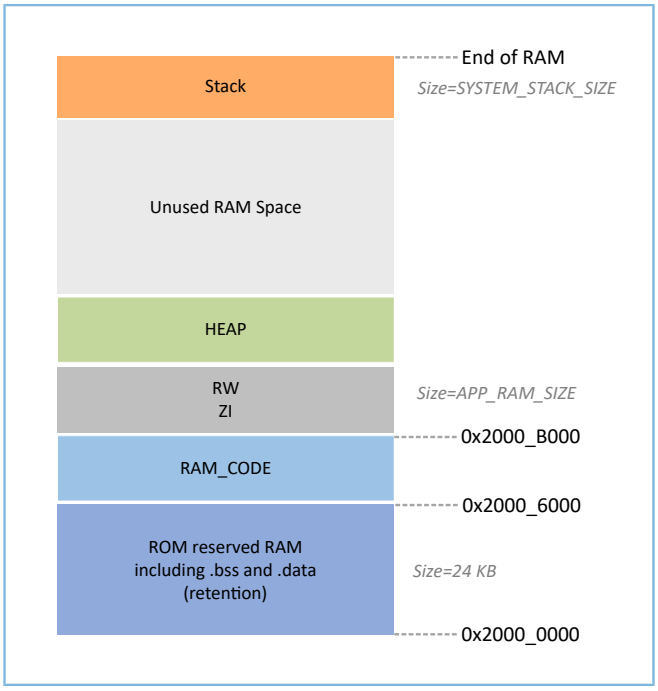

图 2-8 XIP模式的RAM布局

RAM CODE为在RAM中执行的代码，为提高执行效率，建议将其定位到同物理地址的0x00100 Aliasing Memory区域 。

XIP模式布局允许在代码加载处直接执行Application的固件，从而让Application能使用更多RAM内存。在对Flash存储内容进行更新时，会关闭XIP模式，在擦写期间优先级低于FLASH_PROTECT_PRIORITY的中断无法生成。

说明:

- QSPI0/QSPI1/QSPI2/OSPI均支持XIP模式。在这种模式下，可以将Flash或PSRAM的数据空间映射到内存中，方便直接对内存地址进行操作。
  - 在外置PSRAM方案中，QSPI1上挂载的外置PSRAM可以与0x2000 0000 ～ 0x2007 FFFF的内存结合，形成一个连续的SRAM地址空间。
  - 在内部PSRAM方案中，OSPI上挂载的内部PSRAM可以与0x3000 0000 ～ 0x3007 FFFF存储区结合，形成一个连续的SRAM地址空间 。
- 开发者可以根据实际需求合理地添加自定义section，但不要轻易删改SDK默认的Scatter文件布局，如从Scatter文件中删除“RAM_CODE”段。关于Scatter文件描述，请参考4.3.2.2 配置存储器布局。

2.5.2 Mirror模式的典型RAM布局

图 2-9为Mirror模式的典型RAM布局，开发者可以根据产品需求对其进行修改。

版权所有 © 2023 深圳市汇顶科技股份有限公司

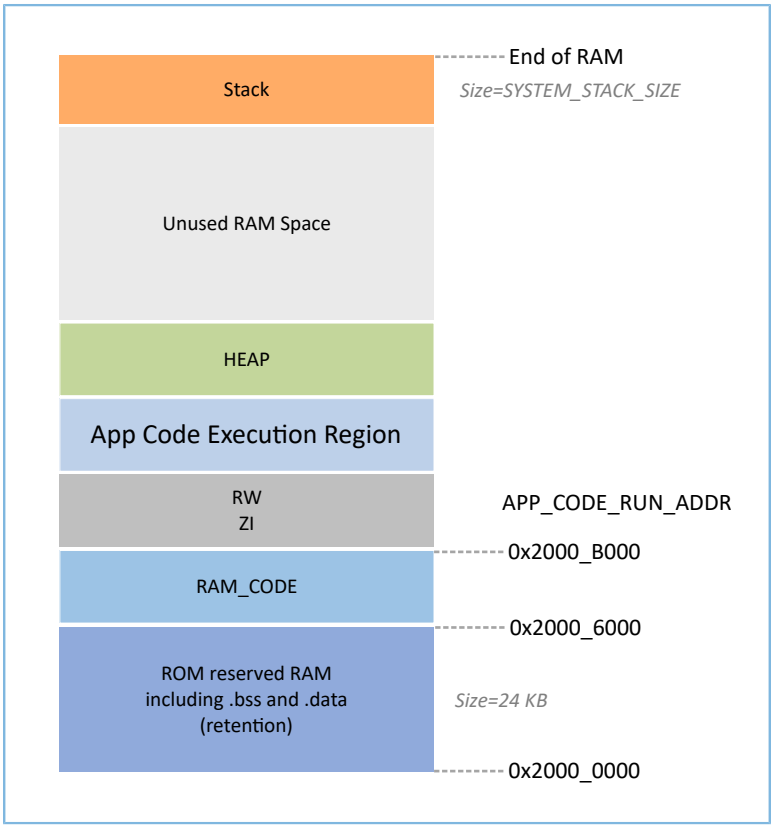

图 2-9 Mirror模式的RAM布局

Mirror模式布局允许在RAM中执行Application的固件。芯片上电之后，会进入冷启动流程。Bootloader会将Application的固件从Flash中复制到名为“App Code Execution Region”的RAM段。睡眠模式的芯片被唤醒后进入热启动流程。为减少热启动时间，Bootloader不会重新复制Application的固件到名为“App Code Execution Region”的RAM段中。

“App Code Execution Region”段的起始位置由*custom_config.h*中的宏APP_CODE_RUN_ADDR决定。开发者需要根据Application的.data和.bss实际使用情况，来确定代码运行地址APP_CODE_RUN_ADDR的值，避免与低地址处的.bss段或高地址处的Call Stack段地址重叠。开发者可根据*.map*文件来获得RAM各段的分布情况。

建议开发者使用RAM Aliasing Memory地址（0x0010_0000 ～ 0x0017_FFFF）来设置APP_CODE_RUN_ADDR。若出现RAM段重叠，在工程构建时会出现error并提示重叠位置，帮助开发者确认并快速定位RAM段重叠情况。

2.5.3 RAM电源管理

每一个RAM Block可以处于三种不同的电源状态：POWER OFF、RETENTION或FULL。

- FULL对应于系统Active状态，MCU可以进行RAM Block读写。
- RETENTION主要用于系统Sleep状态，处于该电源状态的RAM Block中的数据将不会丢失，供系统从Sleep状态恢复到Active态使用。
- 处于POWER OFF状态的RAM Block会掉电，存储于其中的数据会丢失，需用户提前进行保存。

在GR5526中，电源管理单元（PMU）在系统启动时默认开启了全部RAM电源。GR5526 SDK中也提供了完备的RAM电源管理API，开发者可以根据应用需要，合理配置RAM Block的电源。

版权所有 © 2023 深圳市汇顶科技股份有限公司

系统启动时，默认启用了自动RAM功耗管理模式，系统会根据Application的RAM使用情况，自动进行各RAM Block的电源控制。配置规则如下：

- 在系统Active状态，不使用的RAM Block会设置为POWER OFF状态；需要使用的RAM Block则设置为FULL状态。
- 当系统进入Sleep状态时，会将不使用的RAM Block保持为POWER OFF状态，而需要使用的RAM Block设置为RETENTION状态。

在实际应用中，应配置如下：

- 在BLE应用中，RAM_16K_0和RAM_16K_1的前 8 KB是保留给Bootloader和BLE协议栈使用，Application不可使用。在系统Active时，RAM_16K_0和RAM_16K_1应都处于FULL状态；在系统Sleep期间，它们应处于RETENTION状态。非BLE类MCU应用可以使用这两个RAM Block。
- RAM_32K_0及以后的RAM Block的用途可由Application进行规划定义。通常，将用户数据和需要在RAM中执行的代码段定义在从RAM_32K_0开始的连续区间；将函数调用栈（Call Stack）的栈顶定义在RAM的高端地址。这些RAM Block的电源状态可以全部开启，也可以由Application自行控制。

说明:

- 只有当RAM Block处于FULL状态时，MCU才能对其进行访问。
- 更多RAM电源管理API的详细说明在`SDK_Folder\components\sdk\platform_sdk.h`中。

  SDK_Folder为GR5526 SDK的根目录。

2.6 PSRAM介绍

GR5526VGBIP 和 GR5526RGNIP SoC 配有使用OSPI 接口进行数据访问的8 MB PSRAM，其地址映射到0x30080000区域，与0x3000 0000 ～ 0x3007 FFFF存储区结合，形成一个连续的SRAM地址空间，用于拓展用户可用SRAM区域。其具备如下特性功能：

- 低功耗特性
  - 部分阵列自刷新
  - 内置温度传感器的自动温度补偿自刷新（ATCSR）
  - 用户可配置的刷新率
  - 半睡眠模式下的超低功耗（ULP），保留数据
- 软件复位
- 具有可编程驱动强度的输出驱动器LVCMOS
- 用于写入数据的数据掩码
- 数据选通支持高速读取操作
- 注册可配置的写入和读取初始延迟
- 最大1024字节和最小 2 字节的写入突发长度

版权所有 © 2023 深圳市汇顶科技股份有限公司

- 16 B/32 B/64 B/1 KB 长度的环绕和混合突发
- 线性突发命令
- 行边界交叉（RBX）
  - 读取操作可以通过模式寄存器启用
  - 不支持RBX 写入

说明:

- GR5526中已经嵌入了PSRAM，但默认情况下，PSRAM被禁用，需要在使用OSPI控制器之前启用它。
- PSRAM功耗优化可以通过调整PSRAM的驱动强度来修改OSPI控制器和PSRAM之间的阻抗匹配。
  - 驱动强度越低，功耗越低。波形趋于三角波，波形质量较差。
  - 驱动强度越大，功耗越高。波形趋于方波，波形质量较好。
  - 过大的驱动强度可能会导致系统崩溃，应根据应用场景选择合适的驱动强度。
- MCU读访问OSPI的效率较低，故建议使用DMA对齐方式访问OSPI。

2.7 GR5526 SDK目录结构

图 2-10为GR5526 SDK的文件夹目录结构。

版权所有 © 2023 深圳市汇顶科技股份有限公司

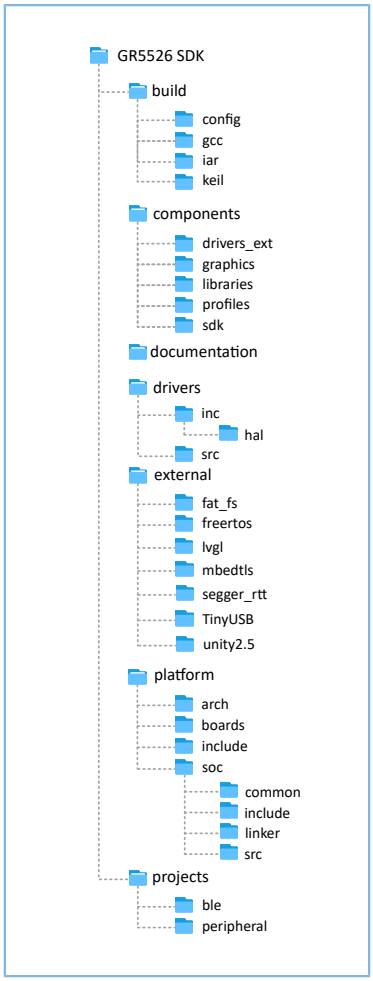

图 2-10 GR5526 SDK目录

表 2-4 为GR5526 SDK中各文件夹的详细描述。

表 2-4 GR5526 SDK文件夹

文件夹	描述
build\config	工程配置目录，用于存放*custom_config.h*模板文件。该文件的内容用于配置工程和提供SDK toolchain所需的相关输入参数。
build\gcc	运行GCC开发环境所需要使用的工具。
build\iar	运行IAR开发环境所需要使用的工具。
build\keil	运行Keil开发环境所需使用的工具。
components\drivers_ext	开发板上第三方元器件的驱动。
components\graphics	GPU显示相关目录。
components\libraries	GR5526 SDK提供的libraries。
components\profiles	GR5526 SDK提供的GATT Services/Service Clients实现示例的源文件。
components\sdk	GR5526 SDK提供的API头文件。
documentation	GR5526 API Reference。
drivers\inc\hal	GR5526外设驱动的HAL和LL的头文件。

版权所有 © 2023 深圳市汇顶科技股份有限公司

文件夹	描述
drivers\inc	易于Application开发者使用的驱动API头文件。
drivers\src	易于Application开发者使用的驱动API源代码。
external\fat_fs	第三方程序，FatFS源代码。
external\freertos	第三方程序，FreeRTOS源代码。
external\lvgl	第三方程序，Lvgl源代码。
external\mbedtls	第三方程序，mbedtls源代码。
external\segger_rtt	第三方程序，SEGGER RTT源代码。
external\TinyUSB	第三方程序，TinyUSB源代码。
external\unity2.5	第三方程序，unity2.5源代码。
platform\arch	CMSIS的Toolchain文件。
platform\boards	存放GR5526 Starter Kit开发板的板级初始化源文件，主要实现对板级基础外设的初始化。
platform\include	存放与平台相关公共头文件。
platform\soc\common	存放Goodix全系BLE SoC兼容的公共源文件，如*gr_interrupt.c*、*gr_platform.c*、*gr_system.c*。
platform\soc\linker	GR5526 SDK提供给链接器使用的符号表文件和库文件。
platform\soc\include	存放SoC寄存器、时钟配置等驱动底层强相关的公共头文件。
platform\soc\src	存放*gr_soc.c*，主要实现SoC芯片强相关的一些初始化流程实现，如Flash、NVDS初始化，晶振配置和PMU校准等。
projects\ble	BLE Application工程示例，比如Heart Rate Sensor，Proximity Reporter。
projects\peripheral	芯片外设APP工程示例。

版权所有 © 2023 深圳市汇顶科技股份有限公司

3 启动流程（Bootloader）

GR5526支持两种固件运行模式：XIP模式和Mirror模式。系统上电后，启动引导程序从系统配置区（SCA）读取系统启动配置信息，并据此进行应用固件完整性校验、系统初始化配置后，跳转到代码运行空间执行固件。不同的运行模式，启动流程略有差异。

- XIP模式下，启动引导程序在完成应用固件校验后，初始化Cache和XIP控制器，然后跳转到Flash空间的代码运行地址进行代码执行。
- Mirror模式下，启动引导程序在完成应用固件校验后，根据系统配置信息，将Flash空间的固件加载到对应的RAM运行空间后，跳转到RAM空间进行代码执行。

图 3-1为GR5526 SDK上Application的启动流程。

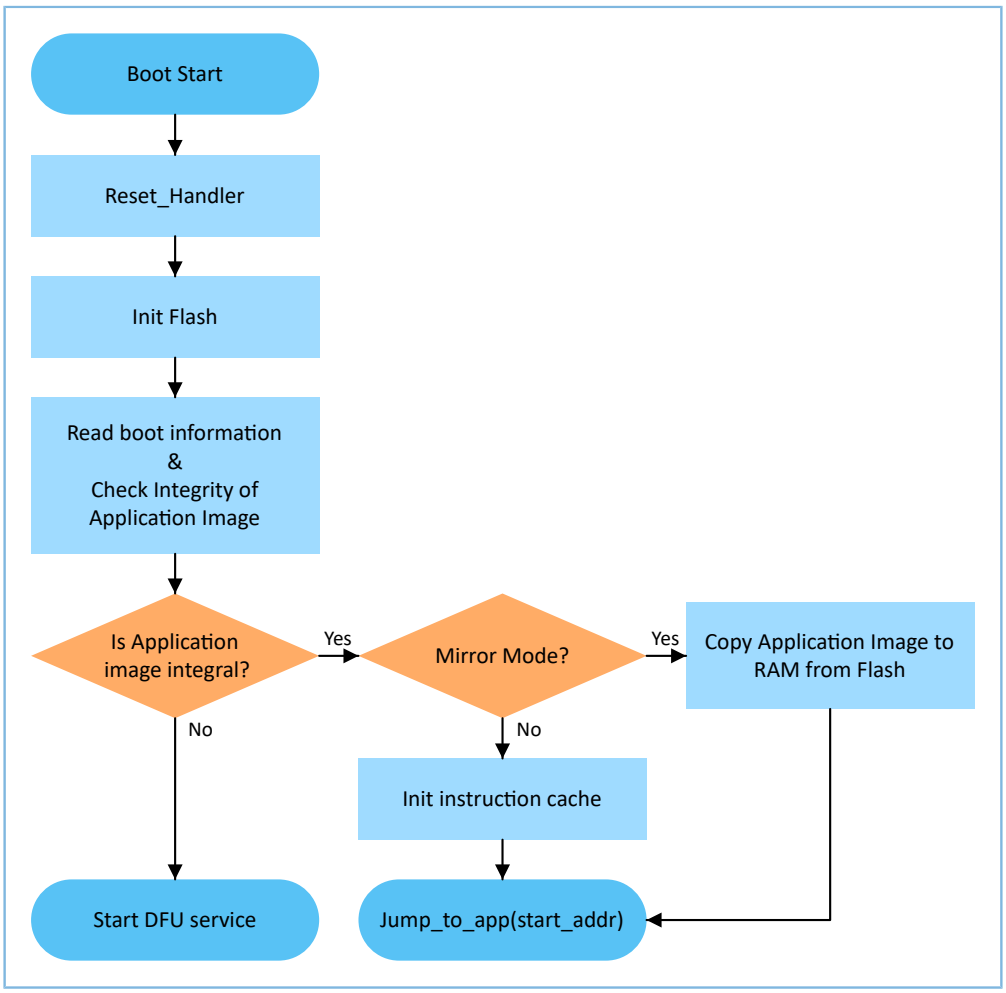

图 3-1 GR5526 SDK上的Application启动流程

1. 设备上电后，CPU将跳转到0x0000_0000处，从该地址处取出C-STACK栈顶指针并赋值给MSP，然后PC指向0x0000_004处，执行ROM中的reset_handler，进入Bootloader。
2. Bootloader执行Flash初始化。
3. Bootloader从Flash中的SCA读取启动信息，并执行Application Firmware的完整性检查。

版权所有 © 2023 深圳市汇顶科技股份有限公司

说明:

GR5526提供安全功能对Application Firmware进行加密和签名。

- 安全模式：若使能安全模式，启动程序从SCA读取启动信息并进行HMAC校验；校验成功后，会解密SCA启动信息，后续执行安全启动流程的验证签名流程，用于保证Firmware完整性和防止篡改，防止伪装等；若验证签名成功，则会使能自动解密功能。
- 非安全模式：若未使能安全模式，启动程序使用SCA启动信息对Application Firmware进行CRC完整性校验。

4. 如果完整性检查失败，则会进入J-Link DFU模式。开发者可通过J-Link配合GProgrammer工具更新Flash中的Application Firmware。
5. 如果通过完整性检查，Bootloader判断运行模式。
   - 对于XIP模式，Bootloader会在完成XIP配置后跳转到Flash中的Application Firmware开始执行。
   - 对于Mirror模式，Bootloader会把Application Firmware从Flash中拷贝到RAM中的指定区段，然后执行RAM中的Application Firmware。

版权所有 © 2023 深圳市汇顶科技股份有限公司

4 使用SDK开发调试

本章将以Keil为例，介绍如何使用SDK来完成BLE Application的创建、编译、下载和调试。

4.1 安装Keil

Keil MDK-ARM IDE（Keil）是ARM®公司提供的用于Cortex®和ARM设备的集成开发环境（IDE）。开发者可从官方网站https://www.keil.com/demo/eval/arm.htm下载Keil安装包并进行安装。GR5526 SDK必须运行在Keil V5.20及以上的版本。

说明:

关于Keil MDK-ARM IDE的使用，可查看ARM提供的在线用户手册：http://www.keil.com/support/man_arm.htm。

图 4-1为Keil启动后的主界面。

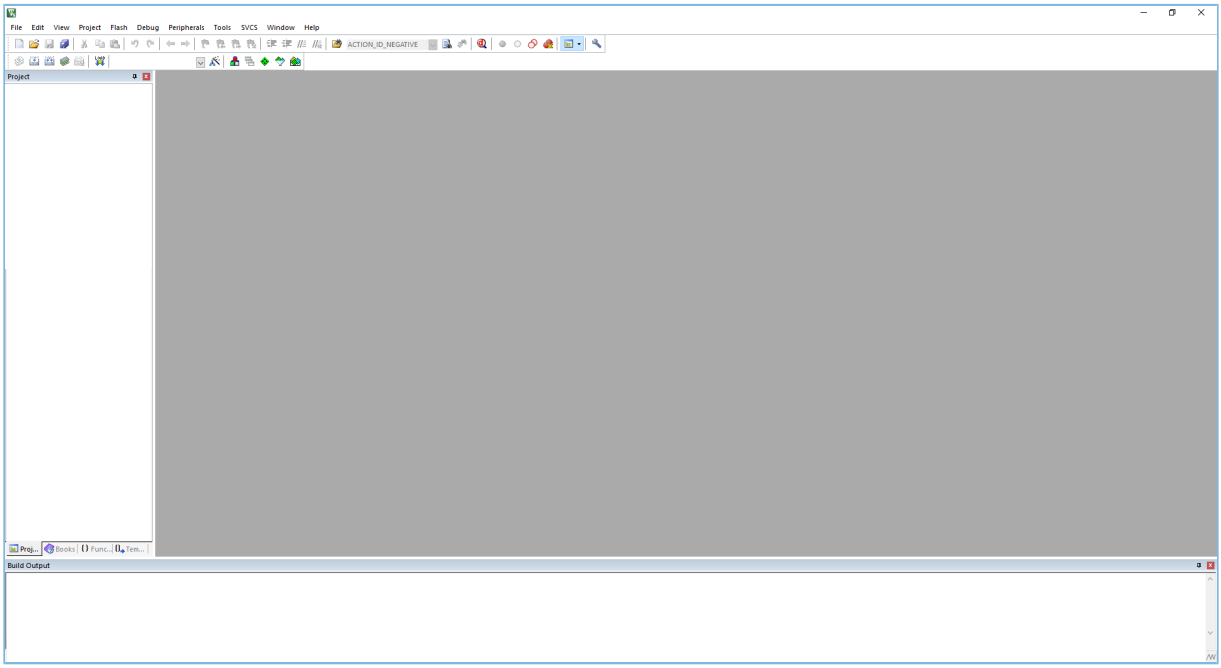

图 4-1 Keil软件界面

表 4-1 为Keil的常用功能按钮。

表 4-1 Keil常用功能按钮

Keil Icon	Description
	Options for Target
	Start/Stop debug session
	Download
	Build

4.2 安装GR5526 SDK

开发者解压GR5526 SDK zip软件包后即可使用，无需手动安装。

说明:

- SDK_Folder为GR5526 SDK的根目录。
- Keil_Folder为Keil的根目录。

4.3 创建BLE Application

本节介绍如何创建一个BLE Application。

4.3.1 准备ble_app_example

进入`SDK_Folder\projects\ble\ble_peripheral\`，拷贝ble_app_template到当前目录，并重命名为ble_app_example。将`ble_app_example\Keil_5`中的*.uvoptx*和*.uvprojx*的主文件名修改为ble_app_example。

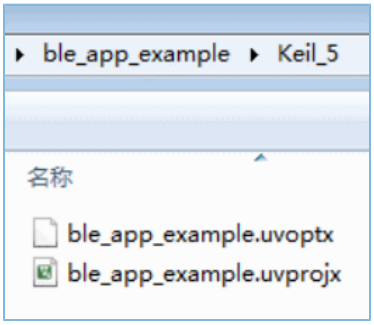

图 4-2 ble_app_example文件夹

双击*ble_app_example.uvprojx*，Keil打开该工程示例。点击 ，在“Options for Target 'GRxx_Soc'”中选择“Output”，在“Name of Executable”输入“ble_app_example”。

版权所有 © 2023 深圳市汇顶科技股份有限公司

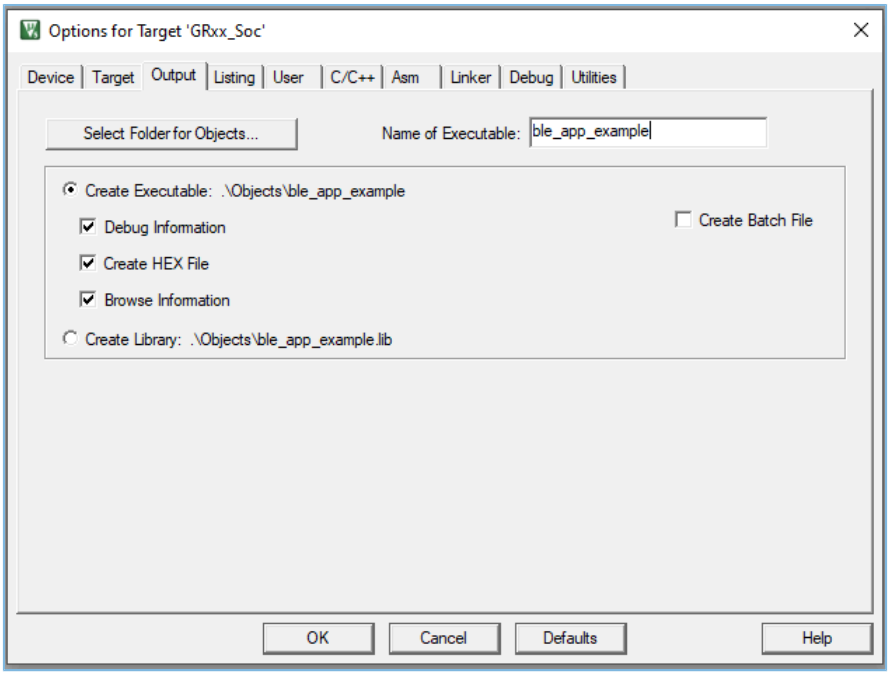

图 4-3 修改Name of Executable

在Keil Project Window中，可查看到ble_app_example工程下的所有groups。

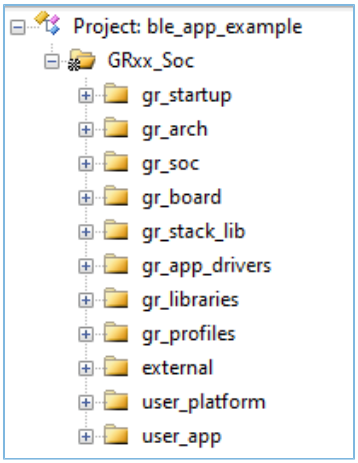

图 4-4 ble_app_example项目

ble_app_example工程下的groups主要分为两类：SDK groups和User groups。

- SDK groups

  SDK groups包括gr_startup、gr_arch、gr_soc、gr_board、gr_stack_lib、gr_app_drivers、gr_libraries、gr_profiles和external。

版权所有 © 2023 深圳市汇顶科技股份有限公司

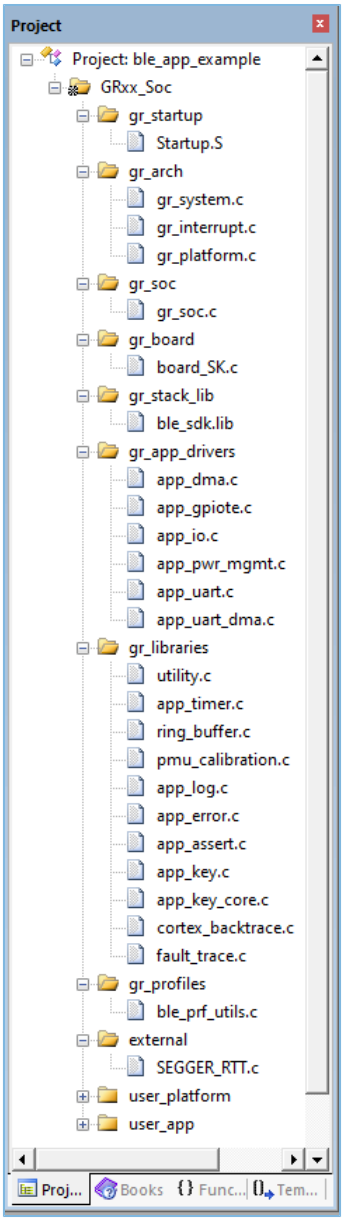

图 4-5 SDK groups

SDK groups下的源文件无需修改，各group的具体描述如下：

表 4-2 SDK groups

SDK group名称	描述
gr_startup	系统启动文件。
gr_arch	System Core、PMU的初始化配置文件和系统中断的接口实现
gr_soc	和SoC相关的处理文件。
gr_board	板级描述文件。
gr_stack_lib	GR5526 SDK lib文件。
gr_app_drivers	易于Application开发者使用的驱动API源文件。开发者可自行添加项目所需要的相关app drivers。
gr_libraries	SDK提供的常用辅助软件模块、外设驱动的开源文件。

版权所有 © 2023 深圳市汇顶科技股份有限公司

SDK group名称	描述
gr_profiles	GATT Services/Service Clients源文件。开发者可自行添加项目所需的相关GATT源文件。
external	第三方程序的源文件，例如freertos，segger rtt。开发者可自行添加项目所需第三方程序。

- User groups

  User groups包括user_platform和user_app。

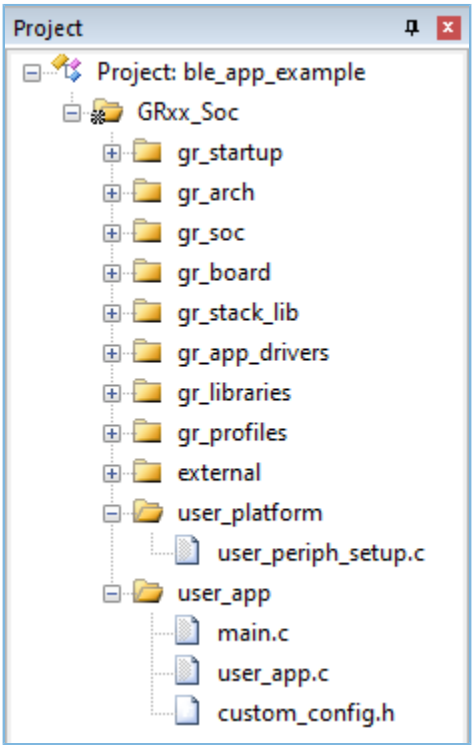

图 4-6 user_groups

User groups下的源文件需要开发者来实现，各group的具体描述如下：

表 4-3 User groups

User group名称	描述
user_platform	软硬件资源的设置和应用程序的初始化，开发者需要根据自己项目需要实现相应的接口。
user_app	主函数入口以及开发者创建的其他源文件，配置BLE协议栈运行时参数和实现GATT Service/ Service Client的事件处理函数。

4.3.2 配置工程

开发者需要根据自己产品的特性配置相应的工程选项，包括NVDS，代码运行模式，存储器布局，After Build以及其他配置项。

4.3.2.1 配置custom_config.h与ble_basic_config.h

*custom_config.h*用于配置Application工程的参数。`SDK_Folder\build\config\`提供一个*custom_config.h*的模板。各Application示例工程的*custom_config.h*位于其工程目录下的`Src\config`。

表 4-4 custom_config.h中的参数

宏	描述
SOC_GR5526	定义芯片版本号。
SYS_FAULT_TRACE_ENABLE	打印Callstack Trace Info的开关。打开该开关后，发生HardFault时，会将Callstack Trace Info打印出来。 0：关闭Trace打印 1：打开Trace打印
APP_DRIVER_USE_ENABLE	App Drivers模块的开关。 0：不使用App Drivers模块 1：使用App Drivers模块
APP_LOG_ENABLE	APP LOG模块的开关。 0：关闭Application中的Logs 1：打开Application中的Logs
APP_LOG_STORE_ENABLE	APP LOG STORE模块的开关。 0：不使用APP LOG STORE模块 1：使用APP LOG STORE模块
APP_LOG_PORT	设置APP LOG输出方式。 0：UART 1：J-Link RTT 2：ARM ITM
SK_GUI_ENABLE	GR5526 Start Kit Board的GUI模块的开关。 0：不使用GUI模块 1：使用GUI模块
DTM_TEST_ENABLE	DTM Test功能的开关。 0：关闭DTM Test功能 1：打开DTM Test功能
DFU_ENABLE	DFU功能的开关。 0：关闭DFU功能 1：打开DFU功能
PMU_CALIBRATION_ENABLE	PMU Calibration功能开关，打开后，系统会自动监控温度和电压并做自适应调整，高低温场景必须开启，建议默认开启。 0：关闭PMU CALIBRATION功能 1：打开PMU Calibration功能
FLASH_PROTECT_PRIORITY	Application在Flash擦写过程中，可阻止异常事件响应的优先级等级。

版权所有 © 2023 深圳市汇顶科技股份有限公司

宏	描述
	当FLASH_PROTECT_PRIORITY配置为N时，中断优先级小于或等于N的中断请求会被挂起，待Flash擦写完毕后，可响应挂起的中断请求。默认情况下，Flash擦写过程中不响应任何中断请求，开发者可以根据实际需要设置其值。
NVDS_START_ADDR	NVDS占用区域的起始地址，默认状态下该宏在*cutom_config.h*中已被注释，开发者若需要更改NVDS占用区域的配置，可启用该宏并更改为所需的值（必须按4 KB对齐），该起始地址不能设置在存储器的SCA或User App等已使用的区域内。
NVDS_NUM_SECTOR	NVDS使用的Flash Sector的数量。
SYSTEM_STACK_SIZE	Application所需要的Call Stack的大小，开发者可以根据Application的实际使用情况，调整其值，但不能少于6 KB。当前默认大小为12 KB。 ble_app_example示例工程编译后其目录下的`Keil_5\Objects\ble_app_example.htm`提供可供参考的Maximum Stack Usage。
SYSTEM_HEAP_SIZE	Application所需要的Heap的大小，开发者可以根据Application的实际使用情况，调整其值。当前默认大小为16 KB。
APP_CODE_LOAD_ADDR *	程序存储空间的起始地址。该地址应在Flash地址范围内。
APP_CODE_RUN_ADDR *	程序运行空间的起始地址。 如果该地址的值与APP_CODE_LOAD_ADDR相等，则Application会采用XIP模式运行。 如果该地址的值在RAM地址范围内，则Application会采用Mirror模式运行。
SYSTEM_CLOCK *	系统时钟频率。其取值如下： 0：96 MHz 1：64 MHz 2：16 MHz（XO） 3：48 MHz 4：24 MHz 5：16 MHz 6：32 MHz（PLL）
CFG_LPCLK_INTERNAL_EN	是否使用芯片内部的OSC时钟作为BLE低频睡眠时钟。若使用，则CFG_LF_ACCURARY_PPM被强制设置为500 PPM。 0：不使用 1：使用
CFG_LF_ACCURARY_PPM	BLE低频睡眠时钟精度，其取值范围1 ～ 500，单位PPM。
BOOT_LONG_TIME *	设置芯片启动时是否需要延迟1s再执行后半段启动代码。 0：不延迟 1：延迟1秒
BOOT_CHECK_IMAGE	在XIP模式中，冷启动时是否要对image进行校验。 0：不进行校验

版权所有 © 2023 深圳市汇顶科技股份有限公司

宏	描述
	1：进行校验
VERSION *	Application Firmware版本号，长度2 Bytes，采用十六进制存储。
CHIP_VER	固件使用的芯片版本，当前值为0x5526。

*：结构体BUILD_IN_APP_INFO被定义在固件的0x200地址处，并使用*custom_config.h*中的宏进行初始化。系统启动时，Bootloader程序会从该地址读取固件的配置信息，作为启动参数。

*ble_basic_config.h*用于配置Application工程BLE相关的参数。各Application示例工程的*ble_basic_config.h*位于其工程目录下的`Src\config`。APP会根据这些宏定义的值来分配BLE所需的资源。

表 4-5 ble_basic_config.h中的参数

宏	描述
CFG_CONTROLLER_ONLY	是否仅使用BLE Controller。 • 0：使用BLE Controller和Host • 1：仅使用BLE Controller
CFG_MAX_PRFS	Application所能够包含的GATT Profile/Service的最大数量。开发者根据实际需求设置该值，其值越大占用的RAM空间就越大。
CFG_MAX_BOND_DEVS	Application支持的最大绑定设备数，最大值为4。
CFG_MAX_CONNECTIONS	Application支持的最大连接数量，最大值为10。开发者可以根据实际需要设置其值。该值越大，BLE Stack Heaps需要占用的RAM空间就越大。BLE Stack Heaps具体大小由*flash_scatter_config.h*中的以下四个宏定义，开发者不可修改这四个宏： • ENV_HEAP_SIZE • ATT_DB_HEAP_SIZE • KE_MSG_HEAP_SIZE • NON_RET_HEAP_SIZE
CFG_MAX_ADVS	Application支持的BLE传统广播和扩展广播总量的最大值。
CFG_MAX_PER_ADVS	Application支持的最大BLE周期性广播数量。 **说明：** 配置的传统广播和扩展广播总量值（CFG_MAX_LEG_EXT_ADVS）与周期性广播数量值（CFG_MAX_PER_ADVS）的总和不能超过5。
CFG_MAX_SYNCS	周期性广播同步数量，用于预留协议栈所需的RAM资源。开发者可根据实际使用的周期性广播同步数量设置该值，最大值为5。
CFG_MAX_SCAN	Application支持的最大BLE扫描数量，最大值为1。

版权所有 © 2023 深圳市汇顶科技股份有限公司

宏	描述
CFG_MAX_EATT_CHANNELS	Application支持的最大BLE EATT通道数，最大值为10。
CFG_ISO_SUPPORT	是否支持ISO模块。 • 0：不支持 • 1：支持

*custom_config.h*与*ble_basic_config.h*的注释符合Keil的Configuration Wizard Annotations 规范，因此开发者可以使用图形化的Keil Configuration Wizard来配置Application工程参数。强烈推荐开发者使用Wizard以避免出现非法参数值。

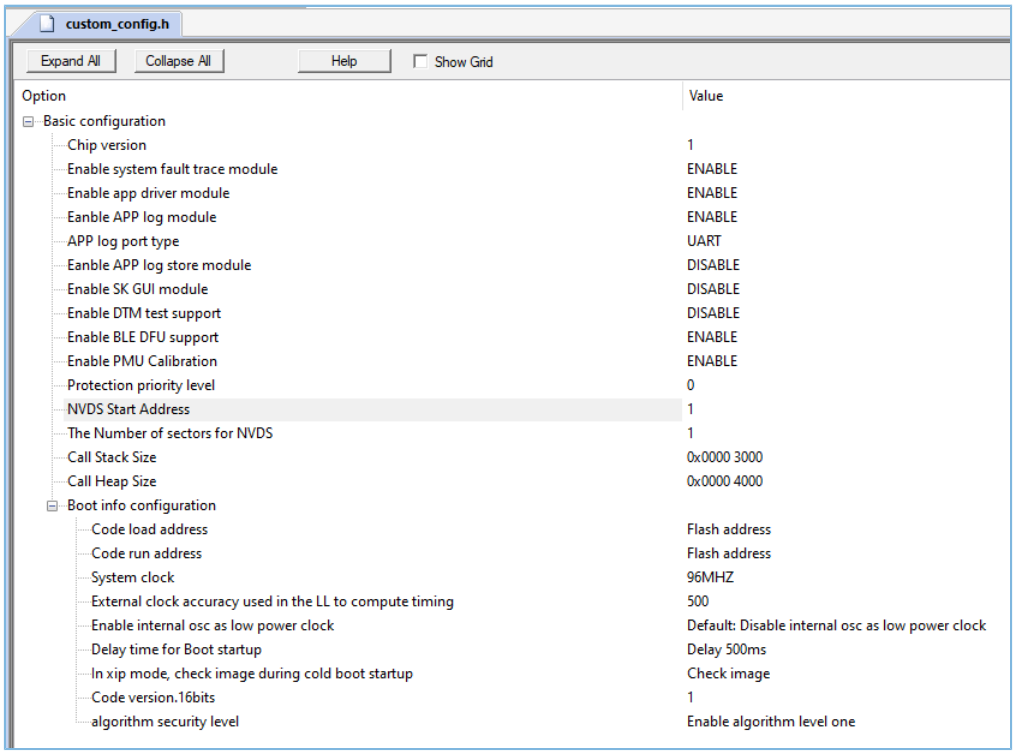

图 4-7 Configuration Wizard for custom_config.h

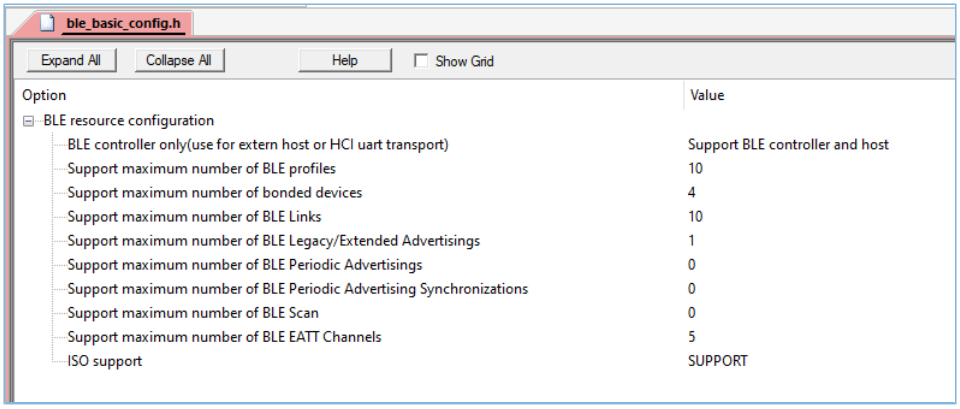

图 4-8 Configuration Wizard for ble_basic_config.h

版权所有 © 2023 深圳市汇顶科技股份有限公司

4.3.2.2 配置存储器布局

Keil用*.sct*文件定义链接器用到的存储段。GR5526 SDK为Application开发者提供一个示例*flash_scatter_common.sct*。该*.sct*文件使用的宏被定义在*flash_scatter_config.h*中。

说明:

在Keil中，`__attribute__((section("name")))`可用来将一个函数或变量放在单独的内存段中，其中“name”取决于开发者的选择。Scatter(*.sct*)文件可用来将被命名的段放在特定的位置。例如，将应用程序的ZI（零初始化）数据放在名为`__attribute__((section(".bss.app")))`段中。

开发者可按照以下步骤配置存储器布局：

1. 点击Keil Toolbar的“Options for Target”按钮，打开“Options for Target ‘GRxx_Soc’”对话框。选中“Linker”标签页。
2. 在“Scatter File”栏，点击按钮“...”浏览选择`SDK_Folder\platform\soc\linker\keil`下的*flash_scatter_common.sct*文件，或者将scatter文件（*.sct*）及其包含的*.h*文件复制到ble_app_example工程目录再选择。

说明:

*flash_scatter_common.sct*中的`#! armcc -E -I ..\Src\user\ -I .\..\Src\config\ --cpu Cortex-M4`指定两个Include路径，其中一个指向Application工程的user目录，另一个指向Application工程的*custom_config.h*所在的目录。如果路径错误，则会产生Linker Error。

---

3. 点击“Edit...”按钮，打开*.sct*文件，然后根据自己产品的存储器布局修改相应的代码。

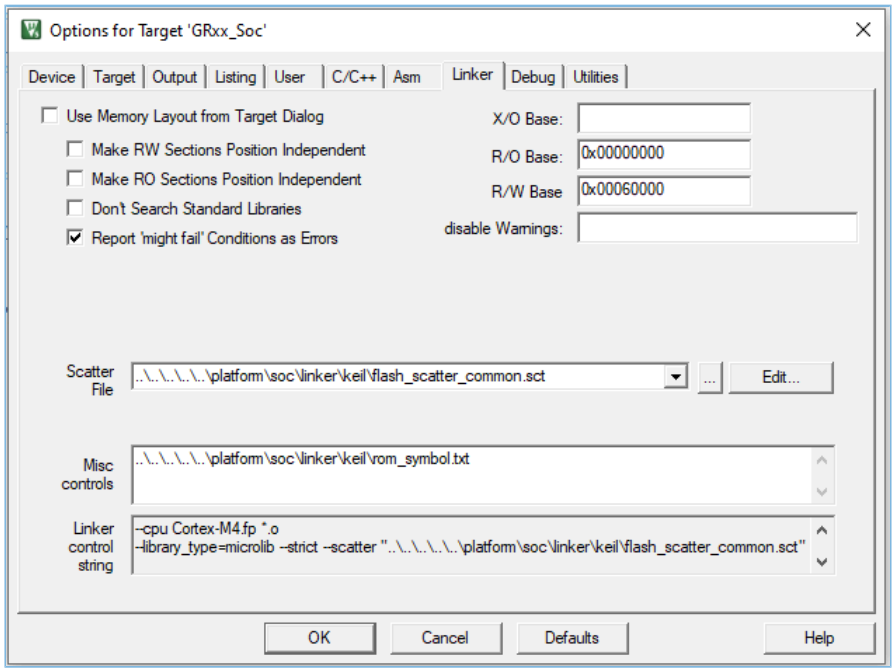

图 4-9 配置Scatter File

4. 点击“OK”按钮，保存设置。

版权所有 © 2023 深圳市汇顶科技股份有限公司

4.3.2.3 配置After Build

Keil的“After Build”可以指定在工程build完成后执行命令行语句。ble_app_template工程默认已配置after build命令，开发者无需再给基于ble_app_template的ble_app_example工程手动配置“After Build”。

如果开发者新建一个工程，则需要按照以下步骤配置“After Build”：

1. 点击Keil Toolbar的“Options for Target”按钮，打开“Options for Target ‘GRxx_Soc’”对话框。选中“User”标签页。
2. 在“After Build/Rebuild”展开的选项中勾选“Run #1”，在对应的“User Command”栏输入“fromelf.exe --text -c –output Listings\@L.s Objects\@L.axf”。其作用是调用Keil fromelf工具，基于axf文件生成汇编文件。
3. 在“After Build/Rebuild”展开的选项中勾选“Run #2”，在对应的“User Command”栏输入“fromelf.exe --bin --output Listings\@L.bin Objects\@L.axf”。其作用是调用Keil fromelf工具，基于axf文件生成bin格式固件。
4. 点击“OK”，保存设置。

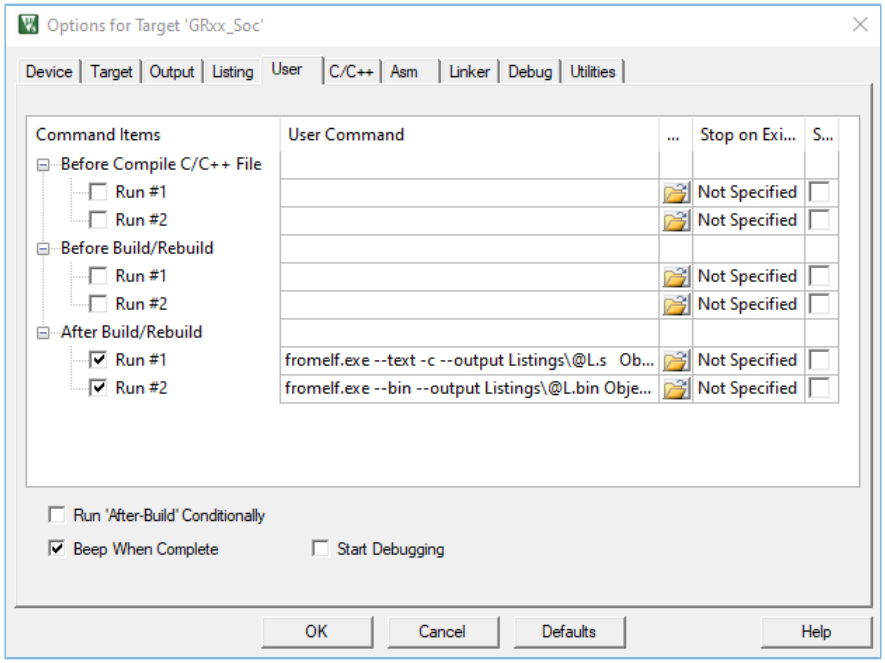

图 4-10 配置After Build

4.3.3 添加用户代码

开发者需要根据自己的需求，修改ble_app_example中相应的代码。

4.3.3.1 修改主函数

以下为典型的*main.c*文件的内容。

```
/**@brief Stack global variables for Bluetooth protocol stack. */
STACK_HEAP_INIT(heaps_table);
```

版权所有 © 2023 深圳市汇顶科技股份有限公司

```
...
int main (void)
{
    /** Initialize user peripherals. */
    app_periph_init();

    /** Initialize BLE Stack. */
    ble_stack_init(&&m_app_ble_callback, &heaps_table);

    // Main Loop
    while (1)
    {
       /*
        * Add Application code here, e.g. GUI Update.
        */
        app_log_flush();
        pwr_mgmt_schedule();
    }
}
```

- `STACK_HEAP_INIT(heaps_table)`定义四个全局数组，供BLE协议栈作为Heap使用。开发者不可修改此定义，否则BLE协议栈无法正常运行。关于Heap的大小，参考4.3.2.1 配置custom_config.h与ble_basic_config.h的CFG_MAX_CONNECTIONS。
- 开发者可以在`app_periph_init()`中初始化外设。在开发调试阶段，该函数中的SYS_SET_BD_ADDR可用于设置临时的Public Address。该函数所在的*user_periph_setup.c*包含以下主要代码：

```
/**@brief Bluetooth device address. */
static const uint8_t s_bd_addr[SYS_BD_ADDR_LEN] = {0x11, 0x11, 0x11, 0x11,0x11, 0x11};
...
void app_periph_init(void)
{
   SYS_SET_BD_ADDR(s_bd_addr);
   bsp_log_init();
   pwr_mgmt_mode_set(PMR_MGMT_SLEEP_MODE);
}
```

- 开发者应该在`while(1) { }`中添加Application的Main Loop代码，比如处理外部输入、更新GUI。
- 如果开发者使用App Log模块，则需要在Main Loop中调用`app_log_flush()`。这可以保证在系统进入Sleep状态之前，Logs被完整输出。关于App Log模块使用，参考4.6.3 输出调试Log。
- 需调用`pwr_mgmt_shcedule()`以实现自动功耗管理，降低系统功耗。

4.3.3.2 实现BLE业务逻辑

Application的BLE相关业务逻辑是通过GR5526 SDK中定义的若干BLE Events来驱动的。Application需要实现相应的BLE Event Handler，以获得BLE Stack的运行结果或者状态改变通知。由于BLE Event Handler是在BLE SDK IRQ的中断上下文（Interrupt Context）中被调用的，因此开发者不能在handler中执行比较耗时的操作，比如阻

版权所有 © 2023 深圳市汇顶科技股份有限公司

塞式函数调用、无限循环等；否则，将阻塞整个系统运行，导致BLE Stack和SDK BLE模块无法按照正常的时序运行。

BLE Events按照Common、GAP Management 、GAP Connection Control、Security Manager、L2CAP、GATT Common、GATT Server和GATT Client来分类，下表列出GR5526 SDK支持的所有BLE Events。

表 4-6 BLE Events

Event Type	Event	Note
Common	BLE_COMMON_EVT_STACK_INIT	BLE Stack init complete event.
GAP Management	BLE_GAPM_EVT_CH_MAP_SET	Channel Map Set complete event.
	BLE_GAPM_EVT_WHITELIST_SET	Whitelist Set complete event.
	BLE_GAPM_EVT_PER_ADV_LIST_SET	Periodic Advertising List Set complete event.
	BLE_GAPM_EVT_PRIVACY_MODE_SET	Privacy Mode for Peer Device Set complete event.
	BLE_GAPM_EVT_LEPSM_REGISTER	LEPSM Register complete event.
	BLE_GAPM_EVT_LEPSM_UNREGISTER	LEPSM Unregister complete event.
	BLE_GAPM_EVT_DEV_INFO_GOT	Device Info Get event.
	BLE_GAPM_EVT_ADV_START	Advertising Start complete event.
	BLE_GAPM_EVT_ADV_STOP	Advertising Stop complete event.
	BLE_GAPM_EVT_SCAN_REQUEST	Scan Request event.
	BLE_GAPM_EVT_ADV_DATA_UPDATE	Advertising Data update event.
	BLE_GAPM_EVT_SCAN_START	Scan Start complete event.
	BLE_GAPM_EVT_SCAN_STOP	Scan Stop complete event.
	BLE_GAPM_EVT_ADV_REPORT	Advertising Report event.
	BLE_GAPM_EVT_SYNC_ESTABLISH	Periodic Advertising Synchronization Establish event.
	BLE_GAPM_EVT_SYNC_STOP	Periodic Advertising Synchronization Stop event.
	BLE_GAPM_EVT_SYNC_LOST	Periodic Advertising Synchronization Lost event.
	BLE_GAPM_EVT_READ_RSLV_ADDR	Read Resolvable Address event.
GAP Connection Control	BLE_GAPC_EVT_PHY_UPDATED	PHY Update event.
	BLE_GAPC_EVT_CONNECTED	Connected event.
	BLE_GAPC_EVT_DISCONNECTED	Disconnected event.
	BLE_GAPC_EVT_CONNECT_CANCEL	Connect Cancel event.
	BLE_GAPC_EVT_AUTO_CONN_TIMEOUT	Auto Connect Timeout event.

版权所有 © 2023 深圳市汇顶科技股份有限公司

Event Type	Event	Note
	BLE_GAPC_EVT_CONN_PARAM_UPDATED	Connect Parameter Updated event.
	BLE_GAPC_EVT_CONN_PARAM_UPDATE_REQ	Connect Parameter Request event.
	BLE_GAPC_EVT_PEER_NAME_GOT	Peer Name Get event.
	BLE_GAPC_EVT_CONN_INFO_GOT	Connect Info Get event.
	BLE_GAPC_EVT_PEER_INFO_GOT	Peer Info Get event.
	BLE_GAPC_EVT_DATA_LENGTH_UPDATED	Data Length Updated event.
	BLE_GAPC_EVT_DEV_INFO_SET	Device Info Set event.
	BLE_GAPC_EVT_CONNECT_IQ_REPORT	Connection IQ Report info event.
	BLE_GAPC_EVT_CONNECTLESS_IQ_REPORT	Connectionless IQ Report info event.
	BLE_GAPC_EVT_LOCAL_TX_POWER_READ	Local transmit power read indication info event.
	BLE_GAPC_EVT_REMOTE_TX_POWER_READ	Remote transmit power read indication info event.
	BLE_GAPC_EVT_TX_POWER_CHANGE_REPORT	Transmit power change reporting info event.
	BLE_GAPC_EVT_PATH_LOSS_THRESHOLD_REPORT	Path loss threshold reporting info event.
	BLE_GAPC_EVT_RANGING_IND	Ranging indication event.
	BLE_GAPC_EVT_RANGING_SAMPLE_REPORT	Ranging sample report event.
	BLE_GAPC_EVT_RANGING_CMP_IND	Ranging complete indication event.
	BLE_GAPC_EVT_DFT_SUBRATE_SET	Default subrate param set complete event.
	BLE_GAPC_EVT_SUBRATE_CHANGE_IND	Subrate change indication event.
GATT Common	BLE_GATT_COMMON_EVT_MTU_EXCHANGE	MTU Exchange event.
	BLE_GATT_COMMON_EVT_PRF_REGISTER	Service Register event.
GATT Server	BLE_GATTS_EVT_READ_REQUEST	GATTS Read Request event.
	BLE_GATTS_EVT_WRITE_REQUEST	GATTS Write Request event.
	BLE_GATTS_EVT_PREP_WRITE_REQUEST	GATTS Prepare Write Request event.
	BLE_GATTS_EVT_NTF_IND	GATTS Notify or Indicate Complete event.
	BLE_GATTS_EVT_CCCD_RECOVERY	GATTS CCCD Recovery event.

版权所有 © 2023 深圳市汇顶科技股份有限公司

Event Type	Event	Note
	BLE_GATTS_EVT_MULT_NTF	GATTS Multiple Notifications event.
	BLE_GATTS_EVT_ENH_READ_REQUEST	GATTS Enhanced Read Request event.
	BLE_GATTS_EVT_ENH_WRITE_REQUEST	GATTS Enhanced Write Request event.
	BLE_GATTS_EVT_ENH_PREP_WRITE_REQUEST	GATTS Enhanced Prepare Write Request event.
	BLE_GATTS_EVT_ENH_NTF_IND	GATTS Enhanced Notify or Indicate Complete event.
	BLE_GATTS_EVT_ENH_CCCD_RECOVERY	GATTS Enhanced CCCD Recovery event.
	BLE_GATTS_EVT_ENH_MULT_NTF	GATTS Enhanced Multiple Notifications event.
GATT Client	BLE_GATTC_EVT_SRVC_BROWSE	GATTC Service Browse event.
	BLE_GATTC_EVT_PRIMARY_SRVC_DISC	GATTC Primary Service Discovery event.
	BLE_GATTC_EVT_INCLUDE_SRVC_DISC	GATTC Include Service Discovery event.
	BLE_GATTC_EVT_CHAR_DISC	GATTC Characteristic Discovery event.
	BLE_GATTC_EVT_CHAR_DESC_DISC	GATTC Characteristic Descriptor Discovery event.
	BLE_GATTC_EVT_READ_RSP	GATTC Read Response event.
	BLE_GATTC_EVT_WRITE_RSP	GATTC Write Response event.
	BLE_GATTC_EVT_NTF_IND	GATTC Notify or Indicate Receive event.
	BLE_GATTC_EVT_CACHE_UPDATE	GATTC Cache Update event.
	BLE_GATTC_EVT_ENH_SRVC_BROWSE	GATTC Enhanced Service Browse event.
	BLE_GATTC_EVT_ENH_PRIMARY_SRVC_DISC	GATTC Enhanced Primary Service Discovery event.
	BLE_GATTC_EVT_ENH_INCLUDE_SRVC_DISC	GATTC Enhanced Include Service Discovery event.
	BLE_GATTC_EVT_ENH_CHAR_DISC	GATTC Enhanced Characteristic Discovery event.

版权所有 © 2023 深圳市汇顶科技股份有限公司

Event Type	Event	Note
	BLE_GATTC_EVT_ENH_CHAR_DESC_DISC	GATTC Enhanced Characteristic Descriptor Discovery event.
	BLE_GATTC_EVT_ENH_READ_RSP	GATTC Enhanced Read Response event.
	BLE_GATTC_EVT_ENH_WRITE_RSP	GATTC Enhanced Write Response event.
	BLE_GATTC_EVT_ENH_NTF_IND	GATTC Enhanced Notify or Indicate Receive event.
Security Manager	BLE_SEC_EVT_LINK_ENC_REQUEST	Link Encrypted Request event.
	BLE_SEC_EVT_LINK_ENCRYPTED	Link Encrypted event.
	BLE_SEC_EVT_KEY_PRESS_NTF	Key Press event.
	BLE_SEC_EVT_KEY_MISSING	Key Missing event.
L2CAP	BLE_L2CAP_EVT_CONN_REQ	L2cap Connect Request event.
	BLE_L2CAP_EVT_CONN_IND	L2cap Connected Indicate event.
	BLE_L2CAP_EVT_ADD_CREDITS_IND	L2cap Credits Add Indicate event.
	BLE_L2CAP_EVT_DISCONNECTED	L2cap Disconnected event.
	BLE_L2CAP_EVT_SDU_RECV	L2cap SDU Receive event.
	BLE_L2CAP_EVT_SDU_SEND	L2cap SDU Send event.
	BLE_L2CAP_EVT_ADD_CREDITS_CPLT	L2cap Credits Add Completed event.
	BLE_L2CAP_EVT_ENH_CONN_REQ	L2cap Enhanced Connect Request event.
	BLE_L2CAP_EVT_ENH_CONN_IND	L2cap Enhanced Connected Indicate event.
	BLE_L2CAP_EVT_ENH_RECONFIG_CPLT	L2cap Enhanced Reconfig Completed event.
	BLE_L2CAP_EVT_ENH_RECONFIG_IND	L2cap Enhanced Reconfig Indicate event.

开发者需要根据产品的功能需求实现所需的BLE Event Handler。例如，如果产品不支持Security Manager，则可以不实现对应的Event；如果产品只支持GATT Server而不支持GATT Client，则可以不实现GATT Client对应的Event。每一类Event里的Event Handler也不一定需要全部实现，开发者只需要实现产品所必须的那些Event Handler。

关于BLE API和Event API的用法，请参考`SDK_Folder\documentation\GR5526_API_Reference`以及`SDK_Folder\projects\ble`中的BLE示例源代码。

版权所有 © 2023 深圳市汇顶科技股份有限公司

4.3.3.3 BLE_Stack_IRQ、BLE_SDK_IRQ与Application的调度机制

BLE Stack是低功耗蓝牙协议栈实现核心，它直接操作Bluetooth 5.3 Core硬件（参考2.2 软件架构）。因此，BLE_Stack_IRQ具有整个系统中次高的优先级（SVCall IRQ具有最高优先级），以保证BLE Stack严格按照Bluetooth Core Spec规定的时序运行。

BLE Stack的状态改变会触发优先级较低的BLE_SDK_IRQ中断。在该中断处理函数中会调用Application实现的BLE Event Handler，将BLE Stack的状态改变通知以及相关的业务数据发送到Application。开发者在这些Event Handler中应避免操作耗时的业务，应当将耗时业务转移到Main Loop或者用户级线程中处理。开发者可使用SDK_Folder\components\libraries\app_queue模块从BLE Event Handler向Main Loop传递事件，或者使用自己的Application Framework。

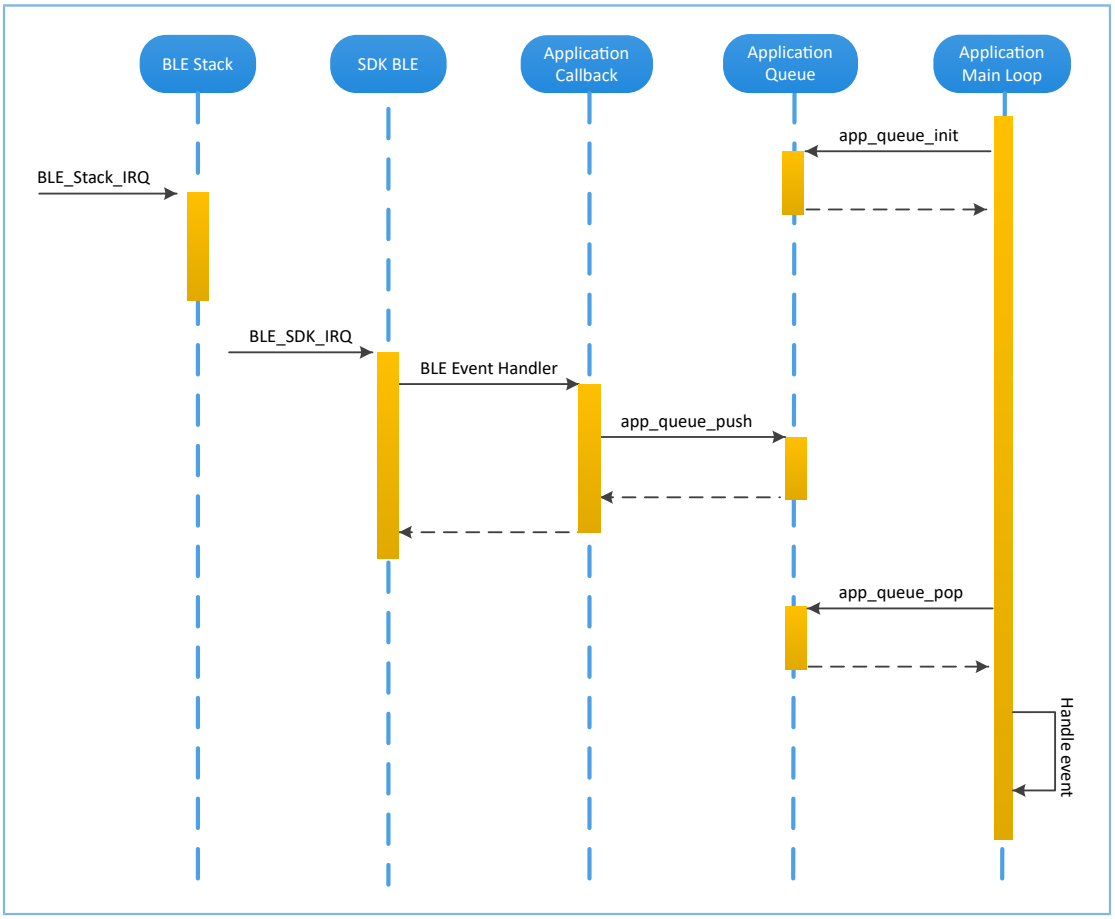

图 4-11 non-OS system schedule

4.4 生成固件

BLE Application创建完成以后，开发者可以直接点击Keil Toolbar中的“Build”按钮构建工程。工程构建完成后，在工程目录下的Keil_5\Listings文件夹和Keil_5\Objects文件夹中会分别生成以下固件。这两种类型的固件都可以通过GProgrammer下载到芯片中运行，具体可参考《GProgrammer用户手册》。

表 4-7 生成的固件

名称	描述
ble_app_example.bin	二进制应用固件，可通过GProgrammer工具下载到芯片中运行。

版权所有 © 2023 深圳市汇顶科技股份有限公司

名称	描述
ble_app_example.hex	二进制应用固件，可通过Keil或GProgrammer工具下载到芯片中运行。

4.5 下载.hex文件到Flash

构建成功以后，开发者需将构建生成的*.hex*文件下载到Flash中，具体操作步骤如下：

1. 配置Keil flash编程算法。

   (1) 复制`SDK_Folder\build\Keil\GR5xxx_16MB_Flash.FLM`文件到`Keil_Folder\ARM\Flash`目录。

   (2) 点击Keil Toolbar中的“Options for Target”按钮，打开“Options for Target ‘GRxx_Soc’”对话框，选择“Debug”标签页。点击“Use: J-LINK/J-TRACE Cortex”右侧的“Settings”按钮。

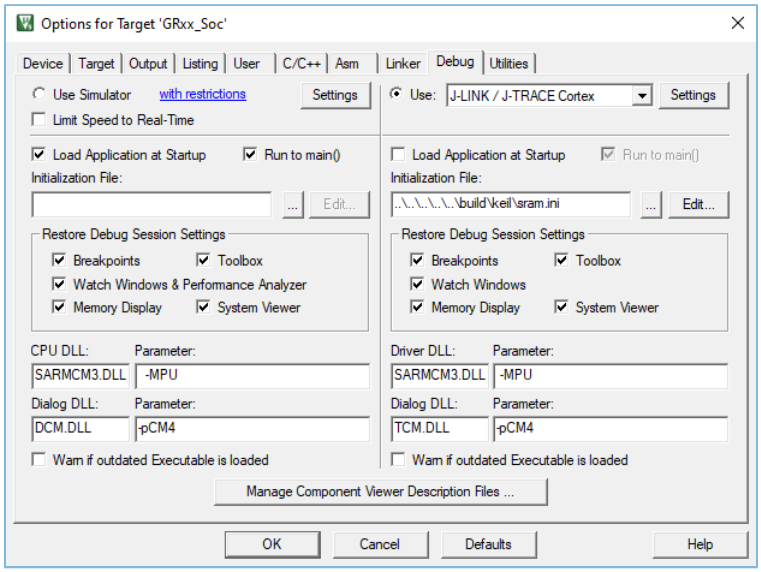

图 4-12 Debug标签页

   (3) 在打开的“Cortex JLink/JTrace Target Driver Setup”窗口中，选中“Flash Download”项。在“Download Function”区域，开发者可以设置Erase方式、选择是否“Program”、“Verify”、“Reset and Run”。Keil默认配置如下：

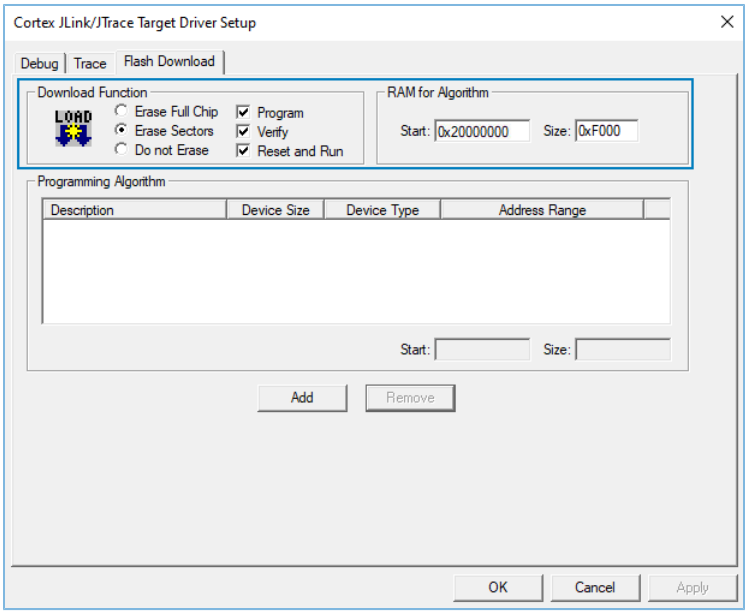

图 4-13 选择“Download Function”

(4) 点击“Add”按钮，在“Programming Algorithm”中添加*GR5xxx_ 16MB_Flash.FLM*。

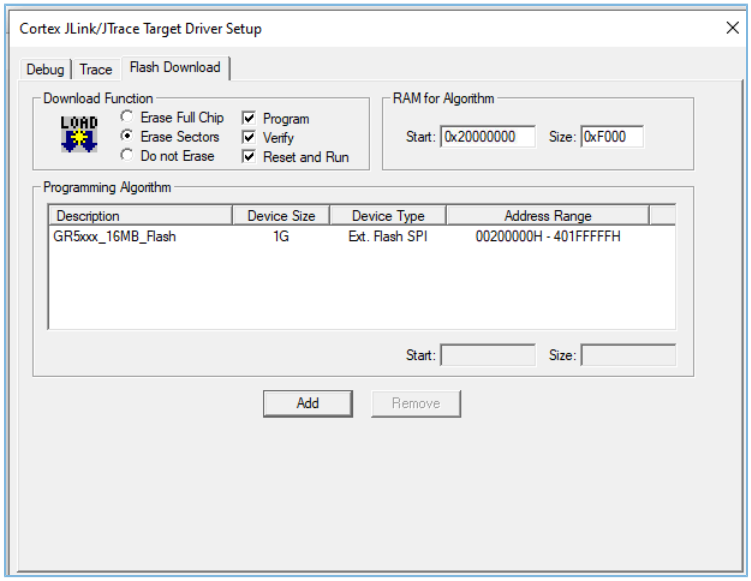

图 4-14 添加GR5xxx_16MB_Flash编程算法

(5) 配置“RAM for Algorithm”，它定义了加载和执行编程算法的地址空间。“Start”输入框中的值必须为GR5526中RAM的起始地址“0x20000000”，“Size”输入框中的值为“0xF000”。

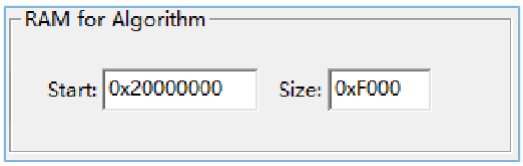

图 4-15 RAM for Algorithm设置

(6) 点击“确定”，保存设置。

版权所有 © 2023 深圳市汇顶科技股份有限公司

2. 下载固件。

   配置完成以后，点击Keil Toolbar上的“Downlod”按钮 将*ble_app_example.axf*文件下载到Flash中。如果下载成功，Keil的“Build Output”窗口将显示如下结果。

说明:

下载过程中，若界面提示“No Cortex-M SW Device Found”，说明芯片当前可能处于睡眠状态（即开启了睡眠模式的工程正在运行），无法直接下载.hex文件到Flash中。开发者需要先按下GR5526 SK板的“RESET”键，并间隔1秒左右点击“Downlod”按钮，重新下载文件。

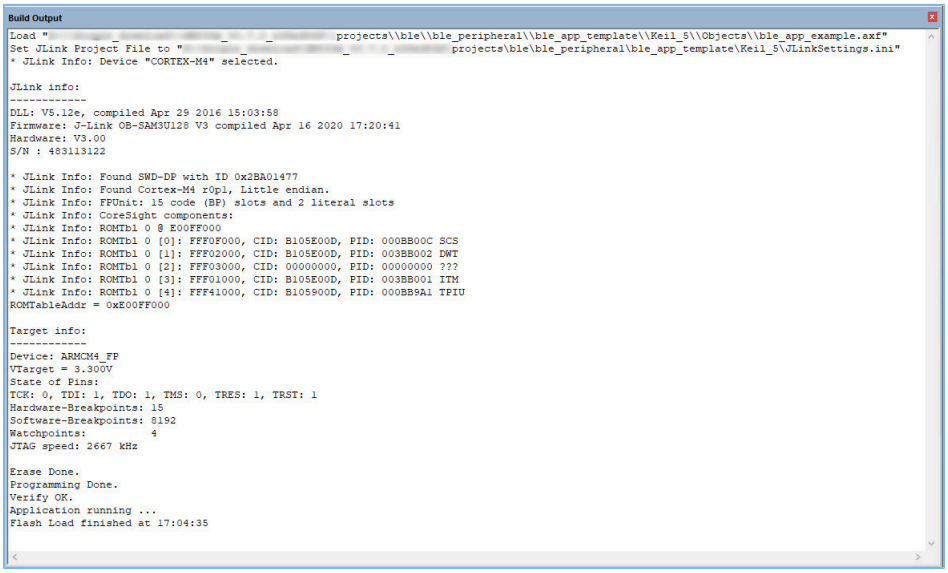

图 4-16 下载结果

4.6 调试

Keil提供调试器，支持代码在线调试。该调试器支持设置6个硬件断点和多个软件断点。开发者还可以用多种方式设置调试命令。

4.6.1 配置调试器

在启动调试之前，需要配置调试器。点击Keil Toolbar中的“Options for Target”按钮，打开“Options for Target ‘GRxx_Soc’ ”对话框，选择“Debug”标签页。窗口左边为使用软件仿真调试，右边为使用硬件在线调试。BLE Examples工程使用硬件在线调试，相关的调试器默认配置如下：

版权所有 © 2023 深圳市汇顶科技股份有限公司

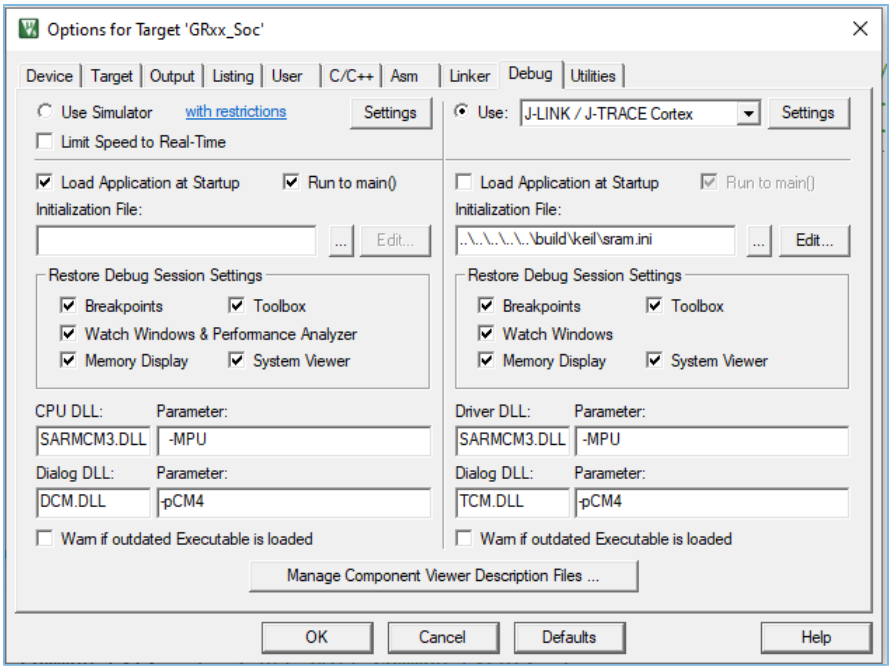

图 4-17 配置调试器

默认使用的Initialization File：*sram.ini*在SDK_Folder\build\keil目录中。开发者可以直接使用该文件，也可以将该文件复制到自己的工程目录下使用。

初始化文件*sram.ini*中包含一组调试命令，调试过程将执行这些命令。在“Initialization File”栏，点击右侧的“Edit...”按钮打开*sram.ini*文件。*sram.ini*的代码示例如下：

```
/**
*************************************************************
* GR55xx object loading script through debugger interface
* (e.g.Jlink,  *etc).
* The goal of this script is to load the Keils's object file to the
* GR55xx RAM
* assuring that the GR55xx has been previously cleaned up.
*************************************************************
*/
// Debugger reset(check Keil debugger settings)
// Preselected reset type(found in Options->Debug->Settings)is
// Normal(0);
// -Normal:Reset core & peripherals via SYSRESETREQ & VECTRESET bit
// RESET
// Load object file
LOAD %L
// Load stack pointer
SP = _RDWORD(0x00000000)
// Load program counter
$ = _RDWORD(0x00000004)
// Write 0 to vector table register,  remap vector
_WDWORD(0xE000ED08,  0x00000000)
```

版权所有 © 2023 深圳市汇顶科技股份有限公司

说明:

Keil支持按照以下顺序执行开发者设置的调试器命令：

1. 当“Options for Target ‘GRxx_Soc’ > Debug > Load Application at Startup”被使能，调试器会首先载入“Options for Target ‘GRxx_Soc’ > Output > Name of Executable”中的文件。
2. 执行“Options for Target ‘GRxx_Soc’ > Debug > Initialization File”所指定文件中的命令。
3. 当“Options for Target ‘GRxx_Soc’ > Debug > Restore Debug Session Settings”包含的选项被选中，恢复相应的Breakpoints，Watch Windows，Memory Display等。
4. 当“Options for Target ‘GRxx_Soc’ > Debug > Run to main()”被选中或者命令`g,main`被发现位于Initialization File中，调试器就会自动开始执行CPU指令，直到遇到main()才会停下来。

4.6.2 启动调试

完成调试器配置后，点击Keil Toolbar的“Start/Stop Debug Session”按钮 即可开始调试。

说明:

需要确保“Connect & Reset Options”均为“Normal”，如图 4-18所示。这样可以保证在“Start Debug Session”之后，点击了Keil Toolbar的“Reset”按钮，程序还能正常运行。

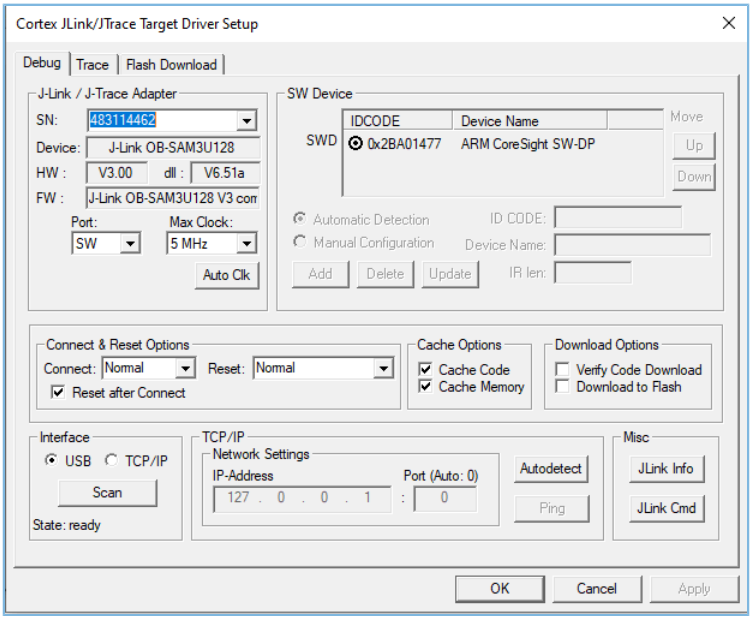

图 4-18 配置“Connect & Reset Options”均为“Normal”

4.6.3 输出调试Log

GR5526 SDK支持自定义输出方式从硬件端口（UART、J-Link RTT或ARM ITM（Instrumentation Trace Macrocell））输出Application的调试Log。为方便开发者使用，GR5526 SDK提供了APP LOG模块。使用该模块时需在*custom_config.h*中打开宏APP_LOG_ENABLE，并根据开发者所需输出方式配置宏APP_LOG_PORT。

版权所有 © 2023 深圳市汇顶科技股份有限公司

4.6.3.1 模块初始化

完成配置后，开发者还需要在外设初始化阶段，通过调用app_log_init()设置Log参数，注册Log输出接口、注册Flush接口完成APP LOG模块的初始化。APP LOG模块支持使用标准C库函数printf()和APP LOG API输出调试Log，若使用APP LOG API，还可以利用Log级别、格式、过滤方式等参数设置来优化Log；而若使用printf()，则可将Log参数设置为NULL。

根据开发者所设置的输出方式不同，调用相应模块的初始化函数（具体可见SDK_Folder\components\libraries\bsp\bsp.h），并注册相应的发送和Flush函数，可参考函数user_log_debug_init()。以UART输出方式为例，接口如下：

```
static void user_log_debug_init(void)
{
    app_log_init_t log_init;

    log_init.filter.level = APP_LOG_LVL_DEBUG;
    log_init.fmt_set[APP_LOG_LVL_ERROR] = APP_LOG_FMT_ALL & (~APP_LOG_FMT_TAG);
    log_init.fmt_set[APP_LOG_LVL_WARNING] = APP_LOG_FMT_LVL;
    log_init.fmt_set[APP_LOG_LVL_INFO] = APP_LOG_FMT_LVL;
    log_init.fmt_set[APP_LOG_LVL_DEBUG] = APP_LOG_FMT_LVL;

    app_log_init(&log_init, bsp_uart_send, bsp_uart_flush);

#if APP_LOG_STORE_ENABLE
    app_log_store_info_t store_info;
    app_log_store_op_t   op_func;

    store_info.nv_tag   = APP_LOG_NVDS_TAG;
    store_info.db_addr  = APP_LOG_DB_START_ADDR;
    store_info.db_size  = APP_LOG_DB_SIZE;
    store_info.blk_size = APP_LOG_ERASE_BLK_SIZE;

    op_func.flash_init  = hal_flash_init;
    op_func.flash_erase = hal_flash_erase;
    op_func.flash_write = hal_flash_write;
    op_func.flash_read  = hal_flash_read;
    op_func.time_get    = NULL;

    app_log_store_init(&store_info, &op_func);
#endif

}
```

版权所有 © 2023 深圳市汇顶科技股份有限公司

说明:

- app_log_init()接口的入参包括Log初始化参数、Log输出接口和Flush接口。其中Flush接口可以不注册。
- GR5526 SDK提供APP LOG STORE模块，该模块支持将调试Log存入Flash中以及从Flash中导出。使用该模块时需在*custom_config.h*中打开宏APP_LOG_STORE_ENABLE。`SDK_Folder\projects\ble\ble_peripheral\ble_app_rscs`工程中配置了该功能，用户可参照该工程配置使用APP LOG STORE模块。
- 使用`printf()`输出的Application Log无法使用APP LOG STORE模块进行存储。

当使用UART输出调试Log时，已实现的Log输出接口和Flush接口分别为bsp_uart_send和bsp_uart_flush。前者实现了app_uart异步（app_uart_transmit_async接口）和hal_uart同步（hal_uart_transmit接口）两种方式的输出接口，用户可根据具体的应用选择合适的Log输出方式；后者为uart_flush接口，用于中断模式下输出缓存在内存中的未发完的数据。这两个接口中的内容用户都可重写。

当使用J-Link RTT或ARM ITM输出调试Log时，已实现的Log输出接口分别为bsp_segger_rtt_send() 和bsp_itm_send()。这两种模式下没有实现Flush接口。

4.6.3.2 使用方法

APP LOG模块初始化完成之后，开发者可以使用以下四个API输出调试Log：

- APP_LOG_ERROR()
- APP_LOG_WARNING()
- APP_LOG_INFO()
- APP_LOG_DEBUG()

如果使用了中断输出模式，可调用app_log_flush()函数将缓存中的全部调试Log输出，从而保证在芯片复位或系统睡眠前将调试Log全部输出。

当选择ARMCC编译并使用J-Link RTT方式输出Log时，推荐在*SEGGER_RTT.c*中做如下修改：

SEGGER_RTT.c

```
238 *
239 *       Static data
240 *
241 **********************************************************************
242 */
243 //
244 // RTT Control Block and allocate buffers for channel 0
245 //
246 __attribute__((section (".ARM.__at_0x20005000"))) SEGGER_RTT_CB _SEGGER_RTT;
247 //SEGGER_RTT_PUT_CB_SECTION(SEGGER_RTT_CB_ALIGN(SEGGER_RTT_CB _SEGGER_RTT));
```

图 4-19 创建RTT Control block并置于地址0x20005000处

此时还需对J-Link RTT Viewer进行配置，配置方式如图 4-20所示。

版权所有 © 2023 深圳市汇顶科技股份有限公司

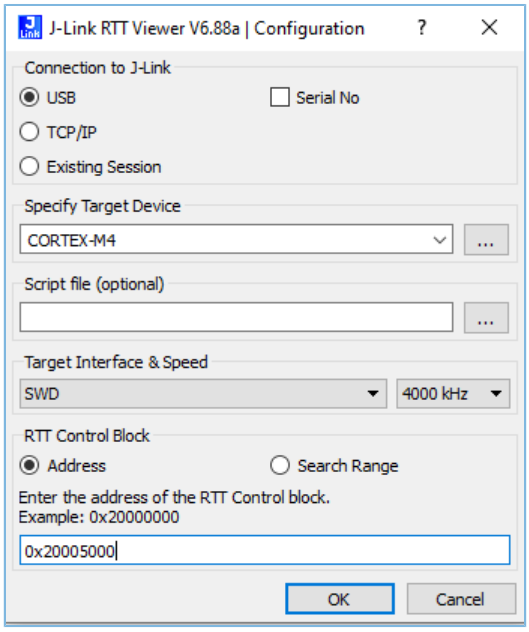

图 4-20 配置J-Link RTT Viewer

其中“RTT Control Block”的位置通过“Address”方式指定，输入值可通过查询编译工程生成的.map映射文件中“_SEGGER_RTT”结构体的地址来获取，如下图所示。若按图 4-19推荐方式创建了RTT Control block并置于地址0x20005000处，则上图中可直接填入地址“0x20005000”。

```
ultra_wfi_or_wfe          0x200037ec   Data          0  rom_symbol.txt ABSOLUTE
sdk_gap_env               0x200038ec   Data          0  rom_symbol.txt ABSOLUTE
_SEGGER_RTT               0x20005000   Data        120  segger_rtt.o(.ARM.__at_0x20005000)
jlink_opt_info            0x20006000   Data          0  rom_symbol.txt ABSOLUTE
SystemCoreClock           0x2000b000   Data          4  system_gr55xx.o(.data)
__stdout                  0x2000b044   Data          4  app_log.o(.data)
```

图 4-21 获取RTT Control Block地址

说明:

当使用GCC编译时，无需按照图 4-19修改，J-Link RTT Viewer中RTT Control block地址为编译工程生成的.map映射文件中“_SEGGER_RTT”结构体的地址。

4.6.4 使用GRToolbox调试

GR5526 SDK提供了用于调试GR5526 BLE应用的Android App：GRToolbox，位于`SDK_folder\tools\GRToolbox\GRToolbox-Version.apk`。该App提供了如下功能：

- 通用的BLE扫描和连接，对characteristics的读写。
- 标准Profile的Demo展示，包括Heart Rate，Blood Pressure等。
- Goodix自定义应用程序。

版权所有 © 2023 深圳市汇顶科技股份有限公司

5 术语和缩略语

表 5-1 术语和缩略语

名称	描述
AoA/AoD	Angle of Arrival/Angle of Departure，到达角/出发角
ATT	Attribute Protocol，属性协议层
BLE	Bluetooth Low Energy，低功耗蓝牙
DFU	Device Firmware Update，设备固件更新
GAP	Generic Access Profile，通用访问规范
GATT	Generic Attribute Profile，通用属性规范
GFSK	Gauss Frequency Shift Keying，高斯频移键控
HAL	Hardware Abstract Layer，硬件抽象层
HCI	Host-Controller Interface，主机-控制器接口
IoT	Internet of Things，物联网
ISOAL	Isochronous Adoption Layer，同步适配层
L2CAP	Logical Link Control and Adaption Protocol，逻辑链路控制与适配协议
LL	Link Layer，链路层
NVDS	Non-volatile Data Storage，非易失性数据存储
OTA	Over The Air，用无线传输
PMU	Power Management Unit，电源管理单元
PHY	Physical Layer，物理层
RF	Radio Frequency，射频
SCA	System Configuration Area，系统配置区
SDK	Software Development Kit，软件开发工具包
SM	Security Manager，安全管理器
SoC	System on Chip，片上系统
UART	Universal Asynchronous Receiver/Transmitter，异步收发传输器
XIP	Execute in Place，片上执行

版权所有 © 2023 深圳市汇顶科技股份有限公司