GR551x APP Log应用说明

版本： **1.0**

发布日期： **2022-03-31**

深圳市汇顶科技股份有限公司

版权所有 © 2022 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2层、13层

电话：+86-755-33338828　　传真：+86-755-33338099

网址：www.goodix.com

前言

编写目的

本文档介绍GR551x SDK中的APP Log模块的作用、原理和使用方法，旨在帮助开发者快速使用该模块进行二次开发。

读者对象

本文适用于以下读者：

- 芯片用户
- 开发人员
- 测试人员
- 开发爱好者

版本说明

本文档为首次发布，对应的产品系列为GR551x。

修订记录

版本	日期	修订内容
1.0	2022-03-31	首次发布

版权所有 © 2022 深圳市汇顶科技股份有限公司

目录

前言 ..... **I**

1 简介 **1**

2 环境搭建 **2**

2.1 准备工作 2

3 使用APP Log模块 **3**

3.1 导入App Log模块 3

3.1.1 添加源文件 3

3.1.2 配置模式与功能 4

3.2 模块初始化与调度 5

3.2.1 日志输出功能 5

3.2.2 日志存储与导出功能 7

3.3 输出日志 8

3.4 获取日志 9

3.4.1 实时获取日志 9

3.4.2 导出存储日志 11

4 模块详解 **15**

4.1 日志发送和存储接口 15

4.2 日志调度接口 17

5 常见问题 **18**

5.1 使用GRToolbox导出的日志有缺失 18

5.2 使用GRToolbox无法导出历史日志 18

版权所有 © 2022 深圳市汇顶科技股份有限公司

1 简介

GR551x APP Log模块是GR551x SDK提供的用于辅助开发者进行开发和调试的工具模块，支持以下功能：

- 日志实时输出。支持开发者自定义调试日志的输出方式（硬件端口UART或J-Link RTT）。
- 日志存储及导出。可将日志存储在GR551x芯片的Flash中，开发者可在需要时在手机APP GRToolbox（Android）上通过蓝牙连接获取日志。
- 日志级别裁定和过滤。支持输出及过滤多级别日志（DEBUG、INFO、WARNING、ERROR），可记录日志级别、时间、来源等信息。

进行操作前，可参考以下文档。

表 1-1 文档参考

名称	描述
GR551x开发者指南	介绍GR551x SDK以及基于SDK的应用开发和调试
J-Link用户指南	J-Link使用说明：www.segger.com/downloads/jlink/UM08001_JLink.pdf
Keil用户指南	Keil详细操作说明：www.keil.com/support/man/docs/uv4/

版权所有 © 2022 深圳市汇顶科技股份有限公司

2 环境搭建

本章介绍如何快速搭建GR551x APP Log模块应用的运行环境。

2.1 准备工作

应用GR551x APP Log模块之前，请完成以下准备工作。

- 硬件准备

表 2-1 硬件准备

名称	描述
开发板	GR5515 Starter Kit开发板（以下简称“开发板”），作为运行GR551x APP Log模块的目标设备
连接线	Micro USB 2.0串口连接线
Android Phone	操作系统Android 5.0（KitKat）及以上版本

- 软件准备

表 2-2 软件准备

名称	描述
Windows	Windows 7/Windows 10操作系统
J-Link Driver	J-Link驱动程序，下载网址：www.segger.com/downloads/jlink/
Keil MDK-ARM IDE（Keil）	集成开发环境（IDE），GR551x SDK支持Keil V5.20及以上版本。 下载网址：https://www.keil.com/demo/eval/arm.htm
J-Link RTT Viewer（Windows）	J-Link日志输出工具，下载网址：https://www.segger.com/products/debug-probes/j-link/tools/rtt-viewer/
GRUart（Windows）	GR551x串口调试工具，位于`SDK_Folder\tools\GRUart`
GRToolbox（Android）	GR551x BLE调试工具，位于`SDK_Folder\tools\GRToolbox`

说明:

SDK_Folder为GR551x SDK的根目录。

版权所有 © 2022 深圳市汇顶科技股份有限公司

3 使用APP Log模块

本章以GR551x SDK的ble_app_pcs工程为例，主要介绍GR551x APP Log模块的导入以及使用方式。

3.1 导入App Log模块

App Log模块是一个独立的功能模块，在使用前需要在相应的工程中添加APP Log模块的文件，并打开相关模块功能的宏开关。

3.1.1 添加源文件

GR551x SDK示例工程中，ble_app_rscs和ble_app_template_freertos工程使能了APP Log日志功能并实现日志存储和导出功能，可参考这两个工程进行移植和开发。

APP Log模块相关源文件描述如下表：

表 3-1 APP Log模块相关源文件

文件	描述
SDK_Folder\components\libraries\app_log\app_log.c	APP Log日志模块的源文件。如需使用APP Log模块，则必须添加该文件
SDK_Folder\components\libraries\app_log\app_log_store.c	APP Log日志存储功能源文件。如需使用APP Log日志存储与导出功能，则需要添加该文件
SDK_Folder\components\libraries\app_log\app_log_dump_port.c	支持蓝牙导出已存储日志功能的源文件。如需使用日志存储与导出功能，则需要添加该文件
SDK_Folder\components\profile\ls\ls.c	日志导出所使用的蓝牙服务对应源文件。如需使用日志存储与导出功能，则需要添加该文件

以GR551x SDK的ble_app_pcs工程为例，添加APP Log模块相关源文件的步骤如下：

1. 打开ble_app_pcs示例工程。

   ble_app_pcs示例工程的源代码和工程文件位于`SDK_Folder\projects\ble\ble_peripheral\ble_app_pcs`，其中工程文件位于Keil_5文件夹。

2. 在ble_app_pcs工程目录添加APP Log的源文件。

   (1) 选中gr_libraries目录点击鼠标右键，选择“Add Existing Files to Group ‘gr_libraries’”，将*app_log.c*、*app_log_store.c*以及*app_log_dump_port.c*文件手动添加至gr_libraries目录下。

   (2) 选中gr_profiles目录点击鼠标右键，选择“Add Existing Files to Group ‘gr_profiles’”，将*ls.c*文件手动添加至gr_profiles目录下。

添加成功后如图 3-1所示：

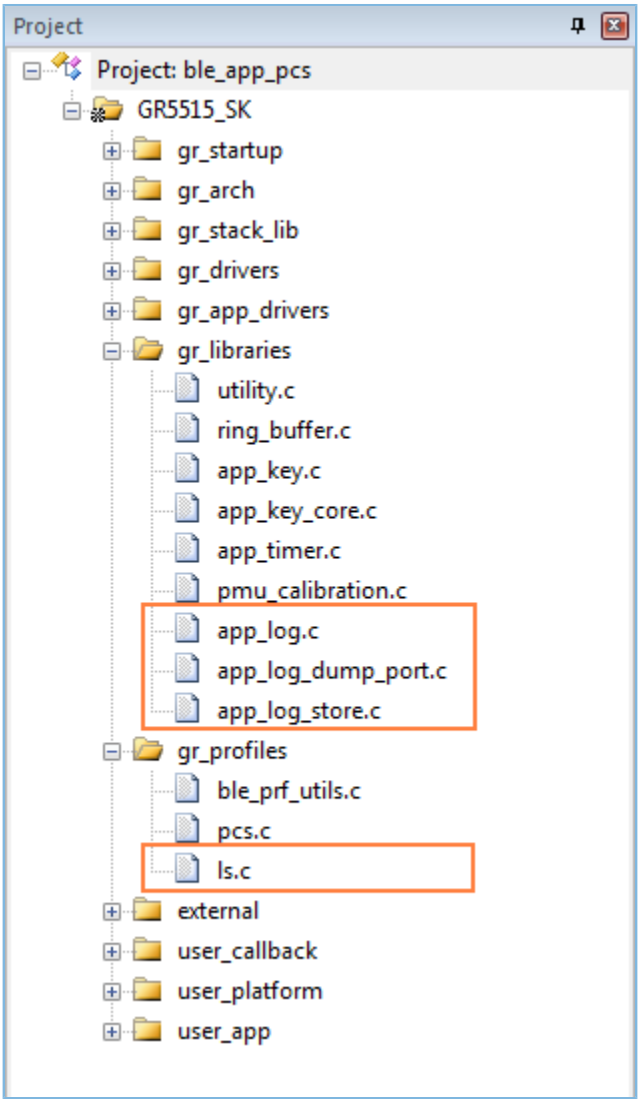

图 3-1 在工程中添加源文件

此外，APP Log模块根据所使用的输出端口，还可能会依赖UART驱动文件和SEGGER RTT相关文件，由开发者选择的输出方式决定。添加方式与添加APP Log的源文件方式类似。

目前，GR551x SDK中的所有工程示例默认已添加了这两个驱动文件，无需手动添加。

- UART驱动源文件位于`SDK_Folder\components\app_drivers\src`和`SDK_Folder\drivers\src`。
- SEGGER RTT源文件位于`SDK_Folder\external\segger_rtt`。

3.1.2 配置模式与功能

APP Log模块相关的宏定义在*custom_config.h*文件中，如下所示。开发者应根据自身工程需求和硬件环境来进行配置。

```
// <o> Eanble APP log module
// <0=> DISABLE
// <1=> ENABLE
#ifndef APP_LOG_ENABLE
```

```
#define APP_LOG_ENABLE          1
#endif

// <o> APP log port type
// <0=> UART
// <1=> RTT
// <2=> ITM
#ifndef APP_LOG_PORT
#define APP_LOG_PORT            0
#endif

// <o> Eanble APP log store module
// <0=> DISABLE
// <1=> ENABLE
#ifndef APP_LOG_STORE_ENABLE
#define APP_LOG_STORE_ENABLE    0
#endif
```

表 3-2 APP Log相关宏说明

宏	定义
APP_LOG_ENABLE	启用或禁用APP Log模块。 • 0：禁用APP Log模块 • 1：启用APP Log模块
APP_LOG_PORT	设置APP Log输出方式。 • 0：UART • 1：J-Link RTT
APP_LOG_STORE_ENABLE	启用或禁用APP Log存储功能。 • 0：禁用APP Log存储功能 • 1：启用APP Log存储功能

3.2 模块初始化与调度

完成配置后，开发者还需要在外设初始化阶段，调用相关初始化函数接口完成初始化，并适时地调用调度函数。根据所需的APP Log日志功能不同，需调用的初始化和调度函数不同，下文将介绍相关接口的使用方式和场景。

3.2.1 日志输出功能

若仅需要APP Log日志输出功能，则只需调用APP Log模块的初始化函数app_log_init()来完成模块初始化。

该函数入参包括Log初始化参数、Log输出接口和Flush接口，可不注册Flush接口。根据开发者所设置的输出端口不同，调用相应接口的初始化函数并注册相应的发送和Flush函数。

- 当使用串口输出调试Log时，需调用串口相关的初始化函数。以*bsp.c*文件为例，需实现串口初始化函数bsp_uart_init、串口发送函数bsp_uart_send和串口发送清空函数bsp_uart_flush。代码片段如下：

说明:

*bsp.c*文件在SDK中的路径为：`SDK_Folder\app\components\libraries\bsp\bsp.c`。

```
    bsp_uart_init();
    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);
```

相关参数的描述如下：

- bsp_uart_send用于实现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。开发者可根据具体的应用需求选择合适的Log输出方式。
- bsp_uart_flush为uart_flush接口，用于中断模式下输出缓存在GR551x芯片的RAM中的未发完的数据。

以上两个接口中的内容开发者均可重写。

- 当使用J-Link RTT输出调试Log时，已实现的Log输出接口为bsp_segger_rtt_send()。此模式下无需实现Flush接口。

  在*bsp.c*中已实现不同输出方式的初始化，开发者若直接使用该文件，只需根据需要配置宏APP_LOG_PORT来选择日志输出方式，也可以参考该文件来进行开发。

若使用异步输出方式（如串口中断模式异步输出），在需要清除数据缓存的场景下，需要先调用app_log_flush()函数输出缓存中的全部日志，避免清除缓存导致日志丢失。例如，在系统进入睡眠前调用app_log_flush()函数，相关代码如下：

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

    // Initialize ble stack.
    ble_stack_init(&s_app_ble_callback, &heaps_table);/*< init ble stack*/

    // loop
    while (1)
    {
        app_log_flush();
        pwr_mgmt_schedule();
    }
```

```
}
```

其中，app_log_flash()会调用初始化时用户挂载的Flush接口来实现全部输出功能。

3.2.2 日志存储与导出功能

若使用APP Log模块的日志存储与导出功能，开发者还需要调用app_log_store_init()接口来完成日志存储相关配置，如下所示：

```
    app_log_store_info_t store_info;
    app_log_store_op_t   op_func;

    store_info.nv_tag   = 0x40ff;
    store_info.db_addr  = 0x01080000;
    store_info.db_size  = 0x20000;
    store_info.blk_size = 0x1000;

    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    = rtc_time_get;
    op_func.sem_give    = app_log_sem_give;
    op_func.sem_take    = app_log_sem_take;

app_log_store_init(&store_info, &op_func);
```

该函数的两个入参结构体分别为：

1. app_log_store_info_t：包含日志存储区域相关信息，涉及参数依次为NVDS Tag、存储起始地址、存储区域大小以及存储区域块大小（即最小擦除单位）。

2. app_log_store_op_t：包含日志存储Flash的操作函数和其他功能函数。操作函数必须全部实现，包括初始化、擦、读、写函数。而其他功能函数则根据情况确定是否需要实现。
   - 若需要在存储的日志中添加实时时间，则需要实现时间获取函数op_func.time_get。
   - 若在搭载操作系统的环境下使用APP Log模块，则需要实现信号量相关的两个函数（op_func.sem_give和op_func.sem_take）。

开发者可根据自身系统Flash布局和类别，来确定该模块的初始化参数。

日志的存储和导出需在调度函数app_log_store_schedule()中进行，因此需要开发者适时地调用该函数。

- 若在裸机环境中使用该模块，则建议在main()函数循环中调用app_log_store_schedule()：

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

```
    // Initialize ble stack.
    ble_stack_init(&s_app_ble_callback, &heaps_table);

    //Loop
    while (1)
    {
        app_log_flush();
        app_log_store_schedule();
        pwr_mgmt_schedule();
    }
}
```

- 若在搭载操作系统的环境下使用该模块，则建议单独使用一个低优先级的任务调度app_log_store_schedule()，且必须要在初始化时注册信号量相关接口，调度方式如下：

```
static void log_store_dump_task(void *p_arg)
{
    while (1)
    {
        app_log_store_schedule();
    }
}
```

此外，APP Log日志导出功能基于蓝牙传输实现，需要对使用的蓝牙服务进行初始化，建议在BLE协议栈初始化完成的回调函数中调用初始化函数app_log_dump_service_init()：

```
void ble_init_cmp_callback(void)
{
    ......
    app_log_dump_service_init();
    ......
}
```

完成对工程的修改（添加、使能、初始化模块）后，可参考《GR551x开发者指南》将编译完成的程序烧录至开发板中。

3.3 输出日志

APP Log模块支持使用标准C库函数printf()和APP Log模块提供的API输出调试日志。

- 若使用printf()输出调试日志，则可将app_log_init()函数中的app_log_init_t *p_log_init设置为NULL。但此时无法利用APP Log模块的Log级别、格式、过滤方式来优化Log，且此方式输出日志不能进行存储和导出。
- 若使用APP Log API输出调试日志，需要在APP Log模块初始化完成之后，调用以下四个API输出调试Log：
  - APP_LOG_ERROR()
  - APP_LOG_WARNING()
  - APP_LOG_INFO()

- APP_LOG_DEBUG()

  在该模式下，开发者还可以利用日志级别、格式、过滤方式等参数设置来优化输出的Log信息，进一步简化应用调试。

说明:

可通过配置`SDK_Folder\components\libraries\app_log\app_log.h`中宏APP_LOG_TAG、APP_LOG_SEVERITY_LEVEL分别设置日志格式和日志输出过滤等级。

3.4 获取日志

获取日志的方式有两种：实时获取和通过GRToolbox工具导出获取。

3.4.1 实时获取日志

开发者可在PC端获取调试日志，可根据配置的输出方式，选择对应的PC端工具。

- 若使用串口输出，可使用GR551x SDK中的GRUart工具，实时获取日志。

  将PC连接至需要读取调试日志的开发板后，运行GRUart。完成相关配置后，可获取开发板日志，如下图所示。

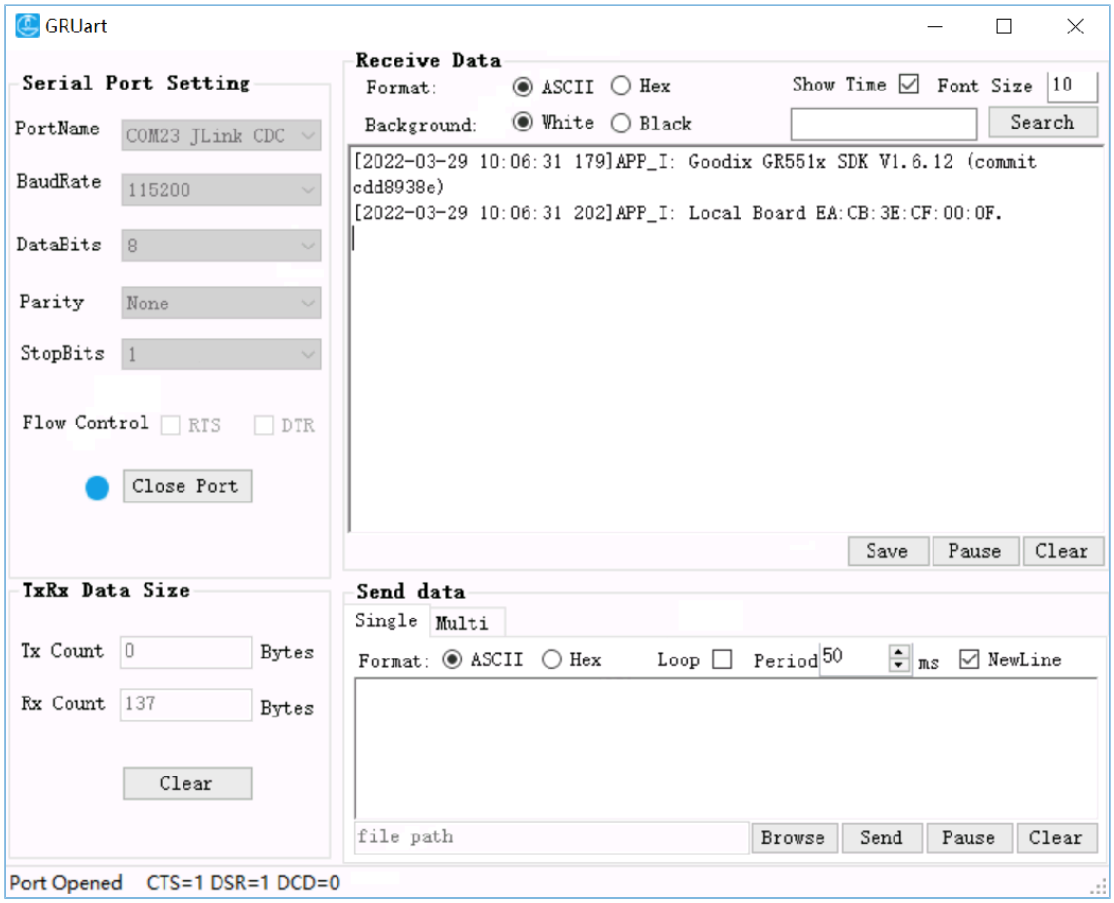

图 3-2 GRUart界面

- 若使用J-Link RTT输出日志，可使用J-Link RTT Viewer软件，实时获取日志。

  将PC连接至需要读取调试日志的开发板后，运行J-Link RTT Viewer软件，进入配置界面，按下图中的各项进行配置。

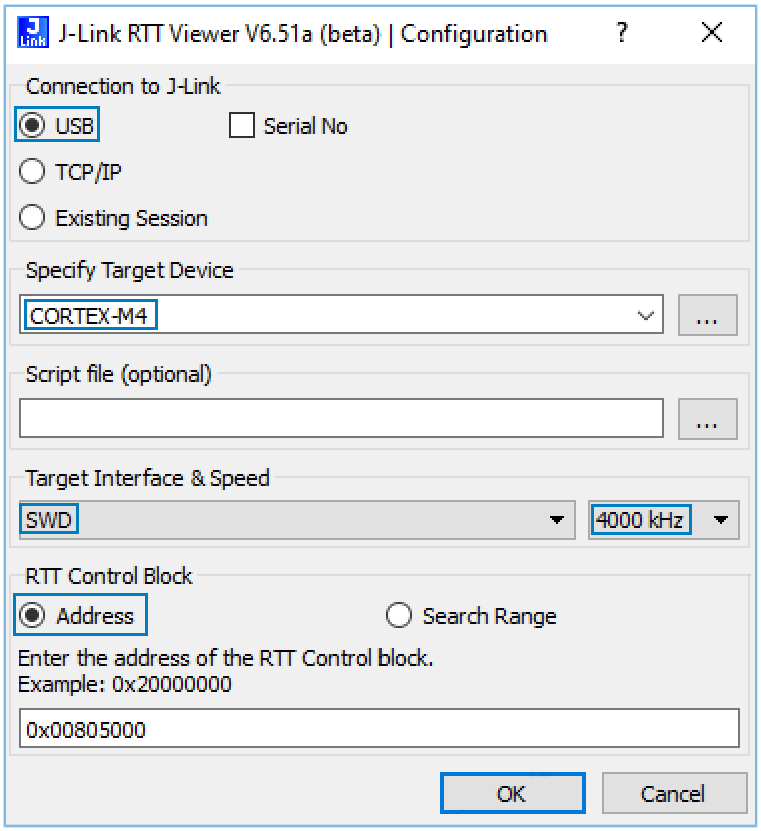

图 3-3 J-Link RTT Viewer配置界面

配置“RTT Control Block”前，需要首先查询到“RTT Control Block”（即变量_SEGGER_RTT）的地址位置。

- 可在J-Link RTT Viewer配置界面中选中“Search Range”，手动指定整个RAM地址搜索范围，然后通过该配置工具自动搜索“RTT Control Block”地址（速度较慢，不推荐）
- 也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，然后直接在配置界面上选中“Address”后指定“RTT Control Block”地址。

推荐在*SEGGER_RTT.c*中做如下修改，直接将RTT Control Block固定定义到指定地址，以提升效率。此处以“RTT Control Block”配置为“0x00805000”为例。

```
// RTT Control Block and allocate buffers for channel 0
//
__attribute__((section(".ARM.__at_0x00805000"))) SEGGER_RTT_CB _SEGGER_RTT
//SEGGER_RTT_PUT_CB_SECTION(SEGGER_RTT_CB_ALIGN(SEGGER_RTT_CB _SEGGER_RTT));
```

版权所有 © 2022 深圳市汇顶科技股份有限公司

说明:

*SEGGER_RTT.c*文件在SDK中的路径为：`SDK_Folder\external\segger_rtt\SEGGER_RTT.c`。

配置完成后，点击“OK”。开发板与J-Link连接后将进入J-Link RTT Viewer日志界面，如下图所示。若界面上显示出固件中的日志，说明配置成功。

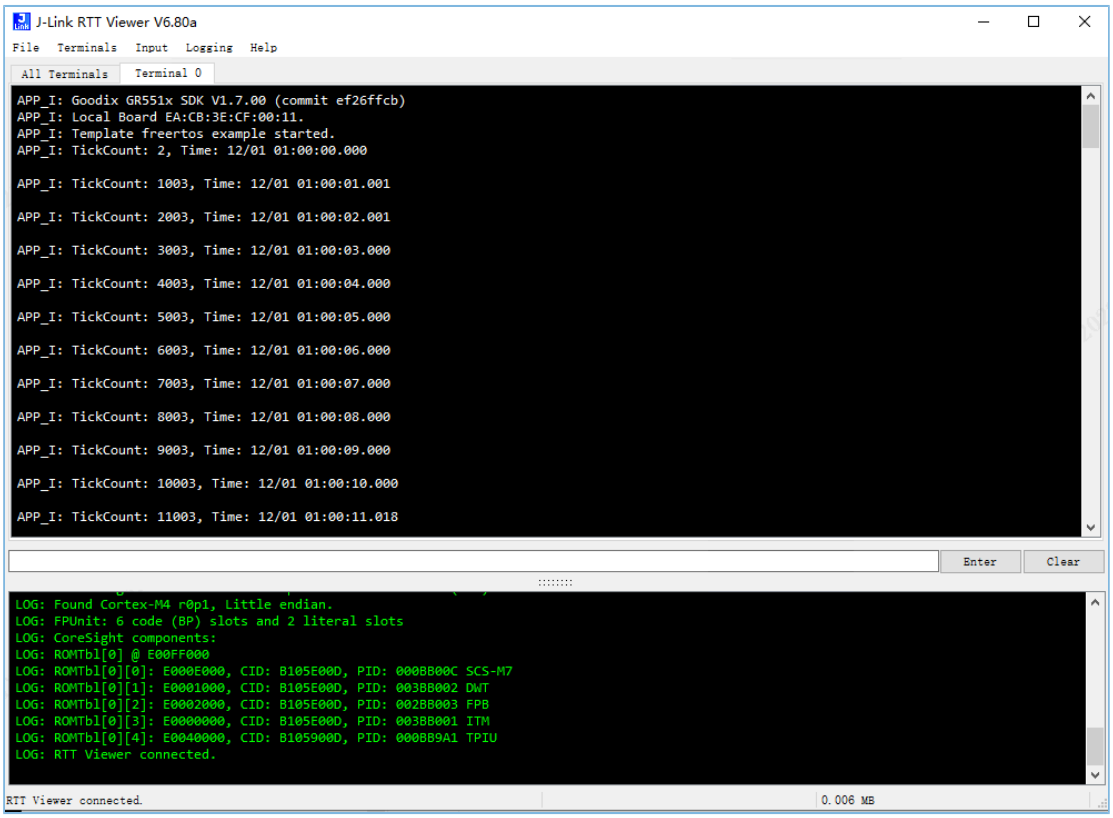

图 **3-4 J-Link RTT Viewer**输出日志界面

3.4.2 导出存储日志

GR551x SDK包中提供的配套工具GRToolbox（Android），支持日志导出功能，可用于导出APP Log模块存储的日志。

完成工程配置后，使用开发板运行该工程生成的固件。

1. 在安卓手机端使用GRToolbox连接该开发板后，可以发现GLS服务“Goodix Log Service”，如下图所示。

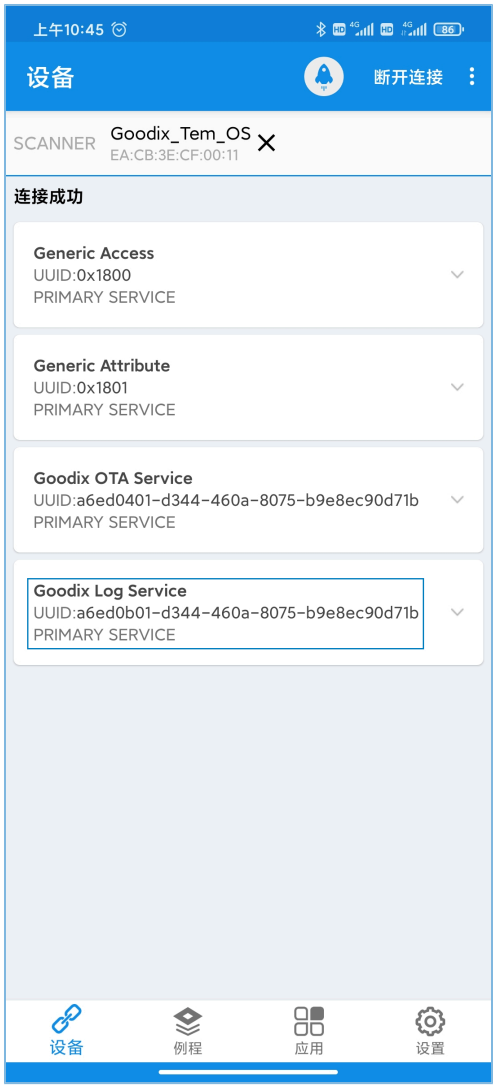

图 3-5 GRToolbox连接开发板后发现GLS服务

说明:

本文中GRToolbox的截图仅供开发者了解操作步骤，实际界面请参考最新版本GRToolbox。

2. 点击右上角的⋮按钮，在下拉菜单中选中“导出日志”：

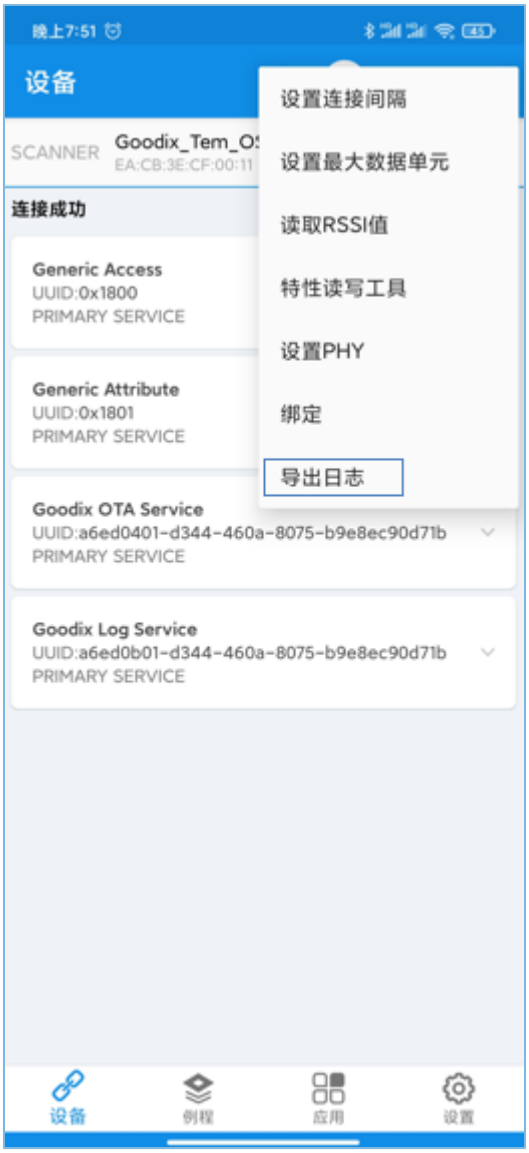

图 3-6 导出日志

3. 在弹出的“导出日志”对话框里，开发者可进行日志删除、保存和读取操作，如下图所示。

图 **3-7 GRToolbox**导出日志界面

4 模块详解

APP Log模块提供多个级别的日志接口，调用这些接口时，会根据接口级别在原始日志前拼接日志级别、时间、来源等信息，并根据初始化时设置的过滤级别进行过滤，最终调用发送函数进行数据传输。下图说明日志输出函数调用关系。

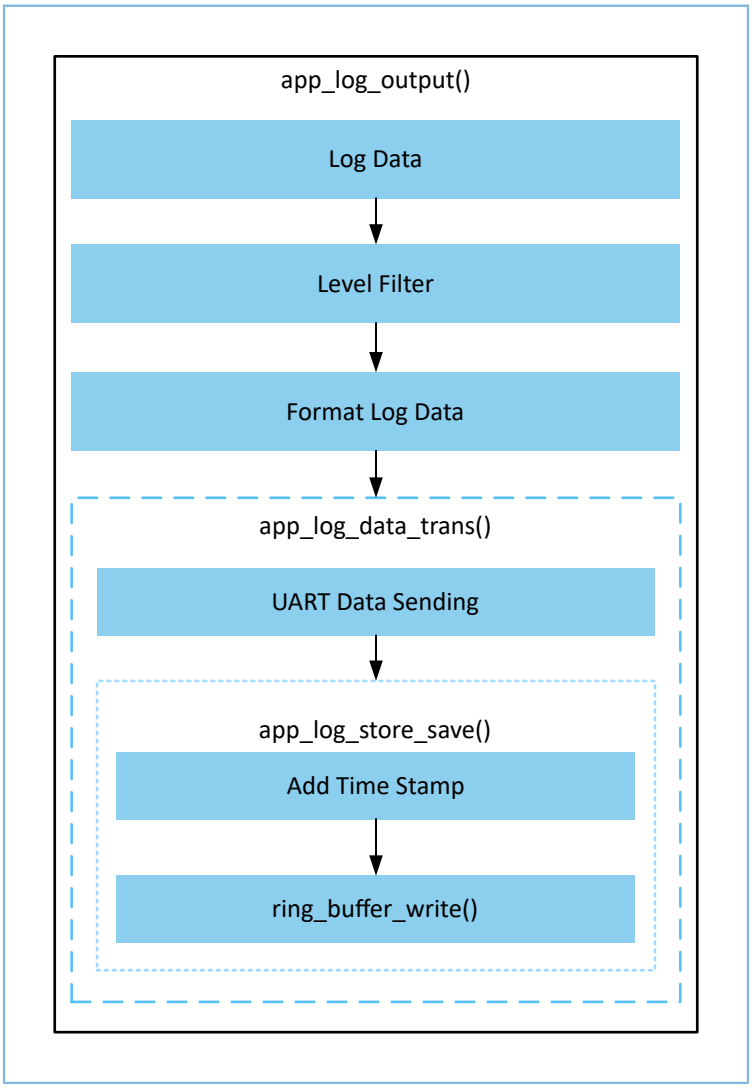

图 **4-1 APP Log**日志输出函数调用关系

说明:

APP Log模块的逻辑代码位于源文件*app_log.c*中。

4.1 日志发送和存储接口

路径：工程目录下的`gr_libraries\app_log.c`

名称：app_log_data_trans()

```
static void app_log_data_trans(uint8_t *p_data, uint16_t length)
{
```

```
    if (NULL == p_data || 0 == length)
    {
        return;
    }

    if (s_app_log_env.trans_func)
    {
        s_app_log_env.trans_func(p_data, length);
    }

#if APP_LOG_STORE_ENABLE
    app_log_store_save(p_data, length);
#endif
}
```

在发送函数中，会调用模块初始化时挂载的端口输出函数s_app_log_env.trans_func（例如串口发送函数），并根据是否使能APP Log日志存储功能宏APP_LOG_STORE_ENABLE，确认是否调用日志存储函数app_log_store_save()。

路径：工程目录下的`gr_libraries\app_log_store.c`

名称：app_log_store_save()

```
uint16_t app_log_store_save(const uint8_t *p_data, const uint16_t length)
{
    ......
    ring_buffer_write(&s_log_store_rbuf, time_encode, APP_LOG_STORE_TIME_SIZE);
    ring_buffer_write(&s_log_store_rbuf, p_data, length);
    if ((APP_LOG_STORE_ONECE_OP_SIZE <= ring_buffer_items_count_get(&s_log_store_rbuf)) &&
        !(s_log_store_env.store_status & APP_LOG_STORE_DUMP_BIT))
    {
        s_log_store_env.store_status |= APP_LOG_STORE_SAVE_BIT;
        if (s_log_store_ops.sem_give)
        {
            s_log_store_ops.sem_give();
        }
    }
    ......
}
```

app_log_store_save()函数会将日志缓存到环形Buffer中，并添加时间戳。当Buffer中数据达到水位线时，会置位准备写入Flash的标志位并发送信号量。

说明:

开发者可根据工程实际情况调整环形Buffer大小和水位线值，在节省RAM空间和避免Buffer溢出之间折中考虑。可使用接口ring_buffer_init配置ring buffer大小，修改`SDK_Folder\toolchain\gr551x\source\arm\flash_scatter_config.h`中宏RAM_RESERVE_SECTION_SIZE来调整日志存储使用RAM大小。

4.2 日志调度接口

实际涉及的Flash操作均在调度函数app_log_store_schedule()中进行，包括日志写入、日志导出以及日志清空。进行这些操作时将调用模块初始化时挂载的Flash操作函数。日志存储和导出逻辑代码位于源文件*app_log_store.c*中。

日志导出时，会调用导出成功的回调函数s_log_dump_cbs->dump_process_cb，以传递出导出的数据。

路径：工程目录下的`gr_libraries\app_log_store.c`

名称：log_dump_from_flash()

```
static void log_dump_from_flash(void)
{
    ......
    if (s_log_store_ops.flash_read && need_dump_size)
    {
    ......
        if (s_log_dump_cbs->dump_process_cb)
        {
            s_log_dump_cbs->dump_process_cb(dump_buffer, dump_len);
        }

    }
    ......
}
```

本模块的实现中，该回调函数中调用了BLE Log Service服务中的数据发送接口，将从Flash读取出的日志数据，通过BLE从设备端传输到手机端。数据发送和对端指令处理相关逻辑在源文件*app_log_store_dump_port.c*中实现，Log Service在源文件*ls.c*中实现。

版权所有 © 2022 深圳市汇顶科技股份有限公司

5 常见问题

本章描述在使用APP Log模块时，可能出现的问题、原因及处理方法。

5.1 使用GRToolbox导出的日志有缺失

- 问题描述

  使用GRToolbox导出日志时，发现日志有缺漏。

- 问题分析

  用于临时存储日志的环形Buffer溢出。

- 处理方法

  增大用于临时存储日志的环形Buffer。若处于搭载操作系统的环境下，可尝试适当提高调度app_log_store_schedule()函数的任务优先级。

5.2 使用GRToolbox无法导出历史日志

- 问题描述

  使用GRToolbox导出日志时，发现只有最近的日志，无法获取更早的日志。

- 问题分析

  存储日志的RAM存储区域空间不足，或者打印日志过于频繁，导致存储区域溢出，覆盖了更早的日志。

- 处理方法
  - 增大RAM存储区域大小。
  - 删除不必要的日志打印任务。

版权所有 © 2022 深圳市汇顶科技股份有限公司