软件开发说明

HL9-SDK 模块

（V1.1）

南京芮捷电子科技有限公司

地址：南京市浦口高新区星火路 20 号

电话：156 5102 8736

邮箱：sales@rejeee.com

网址：www.rejeee.com

 2019-07-25

修订历史

日期	版本	描述	作者/修改者	审核
2019-02-20	V1.0	文档创建，初稿	Felix	
2019-07-25	V1.1	完善说明，更新版本	Felix	

目　　录

导　言

1.1. 概述

HL9-SDK 开发包是在 HL9 软硬件基础上，进行优化完善，集成了 HL9 模组的所有功能，并对硬件进行了扩展，提供了更多的外部接口，满足不同的用户使用和开发需求。对成本控制严苛的用户而言，更便于其简单、快速的进行 LoRa 通信技术开发与评估。

1.2. 读者对象

本文档适用于：

▲ 研发工程师

▲ 技术支持工程师

▲ 客户

如果您是第一次本产品，建议您从第一章开始，阅读本文档全部内容，以便更好的了解产品功能，熟悉使用方式，防止造成操作不当等人为原因带来的不必要损失。

2. 硬件介绍

参考对应 M-HL9 相关数据手册。

3. 系统说明

3.1. 关于 HL9 版本

HL9 标准版本由于不用考虑二次开发需求，因此将剩余的空间做了一个缓冲区，用于接收大数据量情况，内部分配了 5 个包，每包 228 个字节缓冲池。串口按照字节流间隔延时或长度进行分隔。一次性突发大数据会自动分成多个大数据包发送出去。

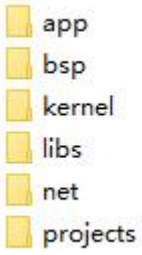
如果串口推送数据大于无线发送能力，则会导致串口数据无法及时取出而与后续数据组合在一起发送。

HL9-SDK 版本去除了缓冲池功能，空出的 RAM 和 Flash 资源用于开发者构建自己的业务逻辑。

HL9-SDK-Lite 版本基于更高开发资源需求，只保留了主任务操作和收发示例操作。

4. 源码说明

4.1. 目录结构



app: 公用程序文件

bsp：主板驱动支持文件

kernel：内核相关文件

libs：通用库文件

net：网络 MAC 协议相关

projects：工程相关文件

4.2. 开发环境

最新版本 SDK 支持 IAR 和 Keil 两种编译环境。使用之前请阅读 MCU 对应的开发软件包，主要先为 IDE 配置 MCU 支持软件包。

IAR 的 IDE 支持包目录结构如下图所示，使用时将其对应目录下的 HDSC 文件夹下的文件一一对应拷贝到 IAR 的安装路径（例如： ~\IAR Systems\Embedded Workbench7.5\arm\config）下，即可使用。

Keil 对应软件包为： HDSC.HC32L13X.1.0.0.pack ， 也可以通过 http://www.keil.com/dd2/pack/看是否有相关 MCU 的最新版本支持包。

SDK 的 IAR 的开发环境用 7.7 构建，更高版本请自行移植。相关软件请自行在官网：https://www.iar.com/iar-embedded-workbench 下载和安装。

Keil 采用 5.25 构建，软件请自行在官网：http://www.keil.com/下载和安装。

仿真器可采用 J-Link 仿真或 Keil，IAR 支持的相关仿真器，采用 SWD 接口。

4.3. 调式说明

本评估板使用 MCU 仿真的话，需要注意两点，相关说明参考《HC32L13 / HC32F03 系列的 MCU 开发工具用户手册》，如下所示截图。

1. MCU 深度休眠时无法使用 SWD 调式，需要复位芯片以恢复 SWD 调式口功能进行程序仿真。
2. MCU 启动支持 BOOT 选择开关，对应的端口是 PD03，PD03 低电平则为运行模式（可仿真调试），高电平为 ISP 烧录模式。

8.1 SWD 端口作为 GPIO 功能程序调试

在应用程序中如果需要将 SWD 端口配置为 IO 使用，程序将无法进行调试。

如果程序中需要使用该功能，建议在调试开发阶段，在程序一开始添加几秒钟的延时程序，或者添加外部 IO 控制程序等方法来决定是否执行该段程序，以便在二次调试开发时 SWD 功能能够正常使用。

8.2 低功耗模式程序调试

在应用程序中，如果使用的芯片具备低功耗模式并需要进入低功耗模式，此时因为 SWD 功能关闭，程序将无法使用调试功能。

如果程序中需要使用该功能，建议在调试开发阶段，在程序一开始添加几秒钟的延时程序，或者添加外部 IO 控制程序等方法来决定是否执行该段程序，或者增加外部唤醒机制，以便在二次调试开发时 SWD 功能能够正常使用。

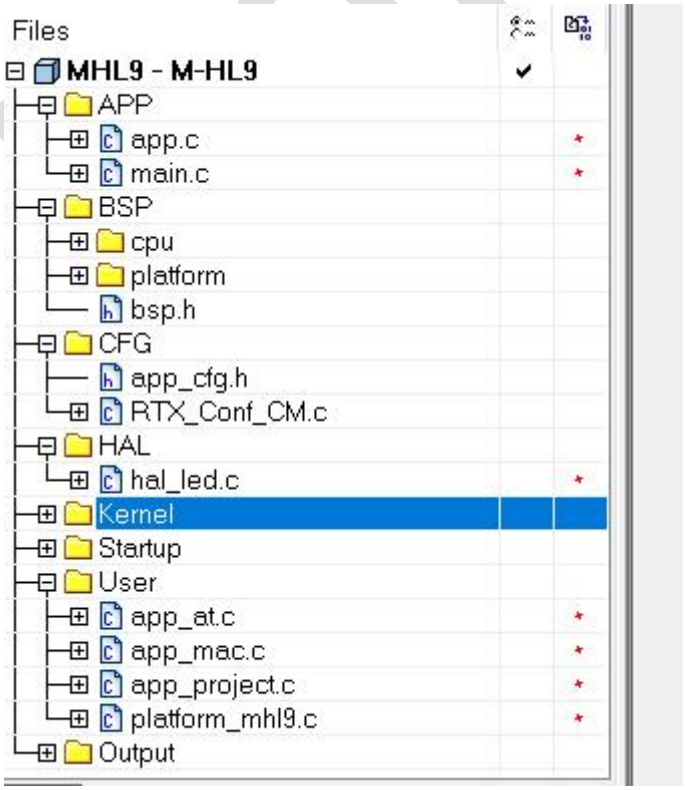
4.4. 工程说明

例程经过很好的代码封装，模块化耦合度低，main 文件为主程序入口。

采用 RTX 系统多任务处理，除主任务外，分别 AT Task（app_at.c）和 Mac Task(app_mac.c)分别处理 AT 指令和无线收发。

User 为 HL9 用户二次开发和可修改代码目录。

上述代码结构目录如下所示。



4.5. 二次开发参考

SDK 包中集成了 AT 指令集，AT 模式软硬件切换方式、LoRa 无线自动收发操作，休眠唤醒、低功耗串口等，提供 RTOS 系统接口方便进行任务管理，用户可以根据需要增删功能。下面简要介绍一下相关简单操作。

4.5.1. 工程文件入口

app_project.c 文件为工程文件入口，相关函数：AppTaskCreate 用于初始化硬件平台和任务构建。为了满足不同用户需求，将 LPTimer 使用开放出来，默认 LPTimer 用于低功耗休眠。

低功耗休眠的初始化操作为：BSP_LPowerInit，配合 PlatformSleep 和 PlatformSleepMs，分别对应按秒计数休眠和毫秒级别休眠。如果不用 SDK 的休眠动作，可以不用 BSP_LPowerInit 初始化操作，可以将 LPTimer 用于自己业务需求。

4.5.2. 创建任务

为了节约资源，SDK 任务定义中只定义了 SDK 使用到的任务数量，如果需要自行增加任务，需要在 RTX 的主配置文件 RTX_Conf_CM.C 中修改任务数量限制。如下所示，OS_TASKCNT 最大可运行任务数，OS_STKSIZE 任务堆栈大小，OS_MAINSTKSIZE 主任务堆栈大小。

```
//
// <h>Thread Configuration
// ======================
//
//   <o>Number of concurrent running threads <0-250>
//   <i> Defines max. number of threads that will run at the same time.
//   <i> Default: 6
#ifndef OS_TASKCNT
 #define OS_TASKCNT     3
#endif

//   <o>Default Thread stack size [bytes] <64-4096:8><#/4>
//   <i> Defines default stack size for threads with osThreadDef stacksz = 0
//   <i> Default: 200
#ifndef OS_STKSIZE
 #define OS_STKSIZE     200
#endif

//   <o>Main Thread stack size [bytes] <64-4096:8><#/4>
//   <i> Defines stack size for main thread.
//   <i> Default: 200
#ifndef OS_MAINSTKSIZE
 #define OS_MAINSTKSIZE 100
#endif
```

参考 app_mac.c 文件，创建任务主要有三段：

声明任务处理函数

static void MacTaskHandler(void const *p_arg);

定义任务结构体

osThreadDef(MacTaskHandler, osPriorityNormal, 1, 0);

#define APP_MAC_NAME　　osThread(MacTaskHandler)

创建任务

BSP_OS_TaskCreate(&gParam.macid, APP_MAC_NAME, NULL);

此处使用 SDK 封装好的函数执行，也可以自己按照 RTX 标准方式创建。

4.5.3. 串口 FIFO 实现

SDK 中主要关键外设代码在文件 platform_mhl9.c 中。主要提供给二次开发者进行外设自定义操作，默认其中主要实现串口、中断、AT 或 WakeUp 相关 IO 的操作，开发者可以根据自己业务需要修改或删除相关外设功能。

这个文件主要实现了串口收发的初始化和回调，实现了串口 FIFO 功能，配合 AT 任务进行串口数据读取。

UART 串口相关代码：

UserDebugInit 串口初始化动作，最新 SDK 为满足客户需要，开放串口分包延时和串口上下拉配置，默认分包延时为 5ms。

FIFO 定义

```
static uint8_t sDebugBuffer[DBG_UART_SIZE] = {0};
static Ringfifo sDebugFIFO = {0};
struct sp_uart_t gDebugUart = {
    .rx_sem = {0},
    .rx_fifo = &sDebugFIFO,
    .timeout = DBG_UART_TIMEOUT,
    .num = DBG_UART_NUM
};
```

UART GPIO 和中断定义

```
BSP_UART_TypeDef uart = {
    .cb = DebugCallback,
    .gpio = DBG_GPIO,
    .tx_pin = DBG_TX_PIN,
    .rx_pin = DBG_RX_PIN,
    .af = DBG_AF,
    .pd = GpioPu,
    .num = DBG_UART_NUM,
    .bdtype = baudrateType,
    .pri = pariType
};
```

UART 分包超时时间 timeout 设置，单位 ms

```
/* Note: you can set timeout you need */
gDebugUart.timeout = BSP_UartSplitTime(baudrateType);
gDebugUart.num = uart.num;
```

注意：

PA 端口用于 SPI 操作 Radio 相关动作，SDK 内部自动使能 PA 和中断功能，不能在外部被禁用或禁止 PA 中断功能，如果禁用需及时恢复，否则射频收发无法正常工作。

4.5.4. 无线收发操作

通过 Mac_Init 初始化射频参数和无线状态控制。只要 PlatformInit 和 Mac_Init 初始化成功后，就可以选择如下函数实现无线收发功能，更简单示例可参考 SDK-Lite 版本代码。MAC 收发数据自动通过 gMacParam 进行中转，相关数据输出可参考示例函数 RadioPrintRecv。

主要函数如下

MacRadio_RxProcess 为持续接收函数

MacRadio_CadProcess 为侦听接收函数

MacRadio_TxProcess 为发送函数

4.5.5. ADC 功能

最新版本 SDK 开放 ADC 接口，用户可直接调用采集相关 IO 口，如下所示采集 MCU 内部 VCC 电压。

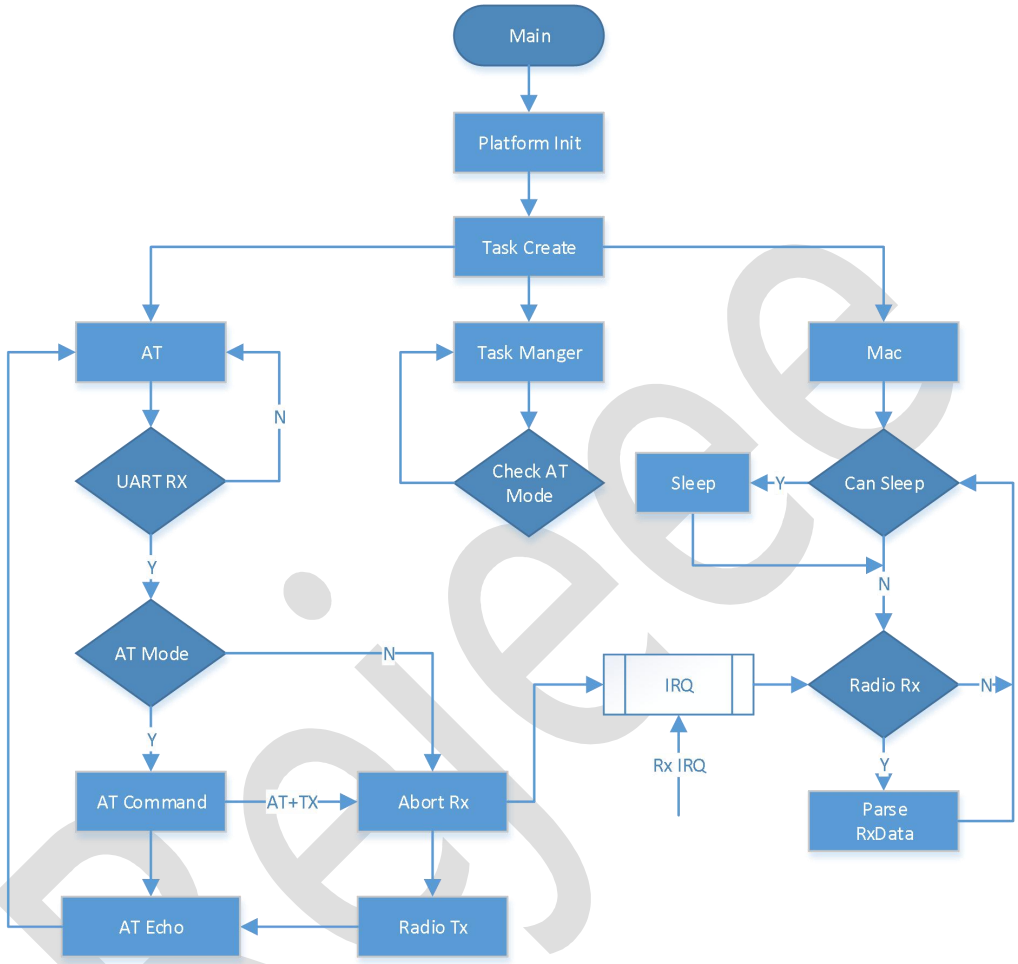
```
void Dev_GetVol(void)
{
    /**
     * NOTE:  you can wait a moment for measure VCC after execute TX, but you
     *        should not be too long and must be less than TX Air Time.
     *
     * for example: Delay for a short time for radio TX work,
     *              you can measure voltage at maximum power consumption
     *        osDelayMs(5);
     */

    /* voltage value */
    BSP_ADC_Enable();
    gParam.dev.vol = (3 * BSP_ADC_Sample(AdcAVccDiV3Input));
    BSP_ADC_Disable();
}
```

4.5.6. 更多参考

具体函数调用参考：HL9-API.chm

具体 AT 操作，请参考 Rejeee AT 指令手册。

SDK 工程代码参考流程如下所示。



HL9 整体流程框图