

2. 4G_SDK 使用指南

BOLING 公司保留对以下所有产品在可靠性、功能和设计方面的改进作进一步说明的权利。BOLING 不承担由本手册所涉及的产品或电路的运用和使用所引起的任何责任，BOLING 的产品不是专门设计来应用于外科植入、生命维持和任何 BOLING 产品产生的故障会对个体造成伤害甚至死亡的领域。如果将 BOLING 的产品用于上述领域，即使这些是由 BOLING 在产品设计和制造上的疏忽引起的，用户应赔偿所有费用、损失、合理的人身伤害或死亡所直接或间接所产生的律师费用，并且用户保证 BOLING 及其雇员、子公司、分支机构和销售商与上述事宜无关。

波领科技

2024 年 9 月

目录

2. 4G_SDK	1
1、SDK 概述.....	3
1.1 SDK 文件构成.....	3
1.2 开发环境及配置选项.....	4
2、SDK 代码说明.....	5
2.1 初始化函数介绍.....	5
3、SDK 配置文件说明.....	7
3.1 射频参数和频点配置.....	7

1、SDK 概述

1.1 SDK 文件构成

SDK 文件包括源码文件、各模块程序文件及应用说明文档：

app: 保存 SDK 中的源代码应用文件（.c 文件，.h 文件） Modules: 保存 SDK 中各个模块程序文件（.c 文件，.h 文件） Hal: 保存 SDK 中的 2.4G 应用函数（.h 函数）

各文件夹内包含文件如下表所示：

Table 1-1 SDK 文件夹内容列表

文件夹		包含文件	内容
SDK	app	main.c	主控程序
	modules	core	CPU 及系统设备相关的实现
		drv	外设驱动，如 GPIO、UART 等
		misc	杂项，实现如 printf 用于 log 打印的 API
	hal	Include	2.4G 相关控制，调用函数声明

1.2 开发环境及配置选项

使用本 SDK 开发应用项目时，需使用以下开发环境（IDE）、编译器，并遵循推荐的配置选项。

Table 1-2 SDK 文件夹内容列表

项目		配置选项设定
综合开发环境		剑池 CDK-V2.22.3
配置选项	Flash	
	Download Function	Erase Sectors
		<input type="checkbox"/> Program
		<input type="checkbox"/> Verify
		<input type="checkbox"/> Reset and Run Soft Reset
	Path	路径中包含 .elf 文件, 确定其路径
	Debug	
	Connector Configurations	⊙Use : ICE
	Load Configurations	<input checked="" type="checkbox"/> Load Application to Target
		<input checked="" type="checkbox"/> Perform Reset after Load
		<input checked="" type="checkbox"/> Auto Run <input checked="" type="checkbox"/> Stop at: main
	Misc Configurations	Reset CPU Type: Soft Reset
	ICE Configuration	
	ICE Adaptor	ICE Clock:12000 KHz NReset Delay: 100 x10us
		<input checked="" type="checkbox"/> Use DDC <input checked="" type="checkbox"/> Enable TRST <input checked="" type="checkbox"/> Enable Pre-Reset
	Debug	<input checked="" type="checkbox"/> Reset After Connect: Hard Reset
		RTOS Type: Bare Metal <input checked="" type="checkbox"/> Download To Flash

2、SDK 代码说明

介绍 2.4G 初始化、TX 与 RX 配置以及休眠处理

2.1 初始化函数介绍

```
omw_rf_init();           //初始化 RF 配置
```

函数初始使用，无参数调用。

```
omw_svc_2g4_init(hc_2g4_param_t* param);    //2.4G 配置初始化
```

omw_2g4_param_t 结构体设置 2.4G 配置参数

```
omw_rf_set_tx_power(int dbm);               //发射功率配置
```

dBm 范围为-25~12dBm。超过 10dBm 存在性能不稳定的可能。

```
RF_2G4_PrepareStart(void);                 //Prepare 使能
```

```
RF_2G4_UpdateDesc_TxPkt(void);              //TX_Pkt 更新
```

```
RF_2G4_UpdateDesc_RxPkt(void);              //RX_Pkt 更新
```

RF 与 2.4G 配置完后调用。

```
omw_svc_2g4_en_whiten_tx(uint8_t en);
```

```
omw_svc_2g4_en_whiten_rx(uint8_t en);
```

白噪使能与失能。en 可设置为 0/1。

```
omw_svc_2g4_tx_data(uint8_t * data, uint16_t len, uint16_t rf_ch1,uint8_t  
whiten_ch1);           //数据发送函数
```

```
omw_svc_2g4_rx_data(uint8_t * data, uint16_t len, uint16_t rf_ch1,uint8_t  
whiten_ch1);           //数据接收函数
```

2.4G 收发函数。data 对应 TX/RX 的数据 buf，len 对应 buf 长度，rf_ch1 对应频点设置与带宽选择，whiten_ch1 对应信道选择。TX/RX 不能同一时间配置使用。

```
omw_sleep_set_sleep_len(SLEEP_ELAPSE);      //休眠唤醒时间配置
```

```
omw_sleep_goto_sleep();                     //使用后进入休眠
```

SLEEP_ELAPSE 配置为 0xFFFF FFFF 时为完全休眠。

休眠唤醒可以分为大醒与小醒。当 SLEEP_ELAPSE 设置为非 0xFFFF FFFF 时，其时间为唤醒时间。定时唤醒称为小醒，Demo 例程中小醒会在信道 37、38、39 各发送一次数据，然后再进入休眠。

大醒为 GPIO 高低电平唤醒，唤醒后直接进入主函数运行。大醒后部分使能失效，因此需要做休眠保护。

```
void omw_svc_24g_rx_end(uint8_t is_crc_err)    //RX结束，数据处理函数
```

在接收到数据后，程序会通过底层跳转至该函数，判断CRC，并计算CRC通过次数。默认将8字节通过串口打印出，打印还包括SYNC与CRC通过次数。不需要再次调用。

```
void omw_svc_24g_tx_end()    //TX结束
```

发送数据后，会跳转至该函数，保留无需操作。

```
#define OMW_GPIO_WAKEUP_MASK    0XFFFFFF    //使能置1，对应GPIO高电平唤醒
```

```
#define OMW_GPIO_nWAKEUP_MASK 0XFFFFFF    //使能置1，对应GPIO低电平唤醒
```

其bit位对应相应GPIO口，置1使能。

```
uint8_t omw_sleep_wkup_worker(uint32_t wkup_type)    //唤醒处理函数
```

函数内部会针对大醒与小醒作出判断。大醒直接返回主程序运行。小醒会在该程序做出对应发送与接收操作（根据用户需求修改），完成操作后，会再次进入休眠，等待下次定时唤醒进入小醒。该函数底层会调用，不需要用户在主程序调用。

```
#define OMW_USE_STATIC_IRQ_TBL    //定义时，使用的为动态向量表，反之为静态
```

```
#define OMW_2G4_RX_ON_SLEEP    //定义时，开始RX，不进入休眠
```

```
#define OMW_2G4_ROLE_SLAVE_NO_SLP    //定义时，开始TX，不进入休眠
```

```
#define OMW_2G4_ROLE_SLAVE_ON_SLP    //定义时，开始TX，进入休眠
```

Demo 中通过定义与屏蔽对应宏，完成相应功能。

3、SDK 配置文件说明

3.1 射频参数和频点配置

```
.channel = 38,  
.phy_mode = RF_RATE_125K,  
.preamble_len = 1,  
.access_code = { 0xD6, 0xBE, 0x89, 0x8E },  
.access_len = 4,
```

默认频点为 38、带宽为 1M，修改 channel 可以改变频点与带宽，收发函数中 rf_ch1 也可修改。

rf_ch1: bits[15:11] channel width, 0=default 1M, 1=100K, 5=500k, 10=1M, 20=2M

rf_ch1: bits[10:0] channel idx

默认发射模式为 1Mbps，修改 phy_mode 可以改变，共 7 种选择。

```
#define RF_RATE_1M      0  
#define RF_RATE_2M      1  
#define RF_RATE_S2      2  
#define RF_RATE_S8      3  
#define RF_RATE_500K    4  
#define RF_RATE_250K    5  
#define RF_RATE_125K    6
```

默认前导码长度为 1Byte，在 2Mbps 配置下设为 2Byte，其余速率使用 1Byte，修改 preamble_len 来改变。

Accesscode: 默认为{ 0xD6, 0xBE, 0x89, 0x8E }，通过修改 access_code[] 与 access_len 来配置。

3.2 GPIO 口的配置

```
#define OMW_UNUSED_GPIO_MASK    (0xCC83FC) //对应 GPIO 置 1，则不使用  
#define OMW_WHEN_SLEEP_GPIO_MASK (0x237000) //对应 GPIO 置 1，则 sleep  
时不使用
```

3.3 调试环境的配置

```
#define CONFIG_OTP_PROGRAM    0          //OTP 环境下调试，不可擦除  
#define CONFIG_FLASH_PROGRAM 1          //Flash 环境下调试，可下载与擦除
```

调试环境的更改除上述宏定义外，还需修改 .ld 文件的 ROM_BASE 数据。 .ld 文件通过 CDK 中 Project Settings 的 Linker 项，Link File 点击开启。

```
__ROM_BASE = 0x10000000;  
__ROM_SIZE = 0x00040000;
```

Flash 环境对应地址为 0X10000000，OTP 环境对应地址为 0X1F800000。

修改至对应环境后，编译会生成附带起始地址的 Bin 文件，供烧录使用。

3.4 休眠保护配置

```
omw_sleep_save_reg_info_ext();      //休眠前保存
omw_sleep_restore_reg_info_ext();    //大醒后恢复
```

休眠保存与休眠唤醒函数内部只配置了 UART0 保护以及 GPIO 状态保护。

```
void t1001_sleep_save_uart_reg_info(uint32_t * uart, uint32_t * save_buf);
void t1001_sleep_restore_uart_reg_info(uint32_t * uart, uint32_t * save_buf);
```

UART0 的保存以及恢复函数。

```
void hc_Timer_sleep_save_reg();
void hc_Timer_sleep_restore_reg();
```

Timer0 的保存以及恢复函数

```
static uint32_t addr_list[]
```

寄存器与中断休眠保护结构体，直接内部填入寄存器地址与中断即可。默认配置了全 GPIO 保护以及 UART0 中断。