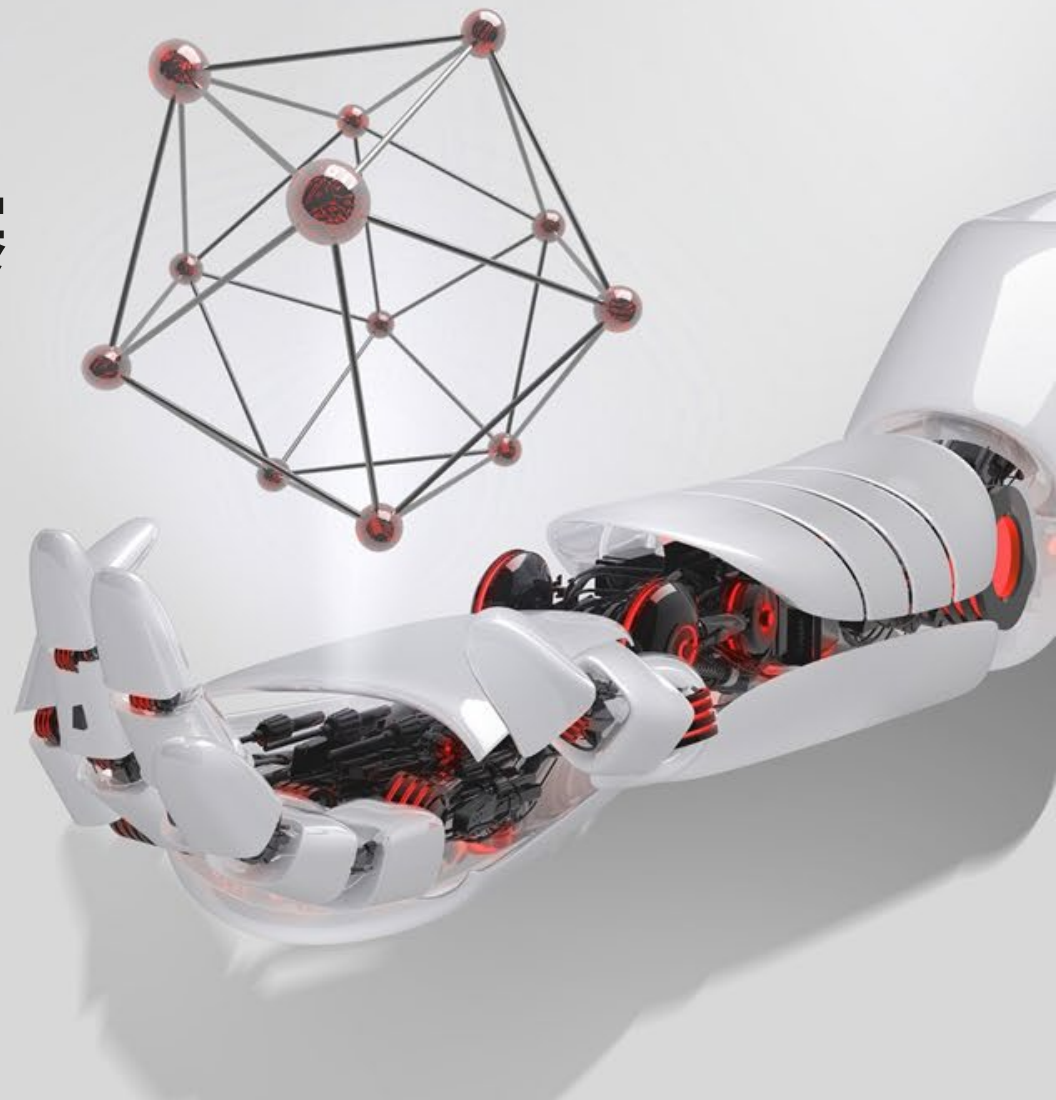


全国大学生嵌入式暨智能互联大赛 海思赛道赋能课件

WiFi-IoT芯片Hi3861 SDK使用介绍

主讲人：梁晓楠





**每一位开发者
都是海思要汇聚的星星之火**

目录

Hi3861规格及优势

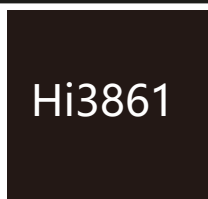
Hi3861 软件架构

Hi3861 SDK目录及文档介绍

Hi3861 SDK使用介绍

Hi3861产品详细规格

WiFi IoT SoC



UART



可选

常电智能家居

白电, 小家电, 电工类

◆ 主要规格

- IEEE 802.11b/g/n, 1×1 2.4GHz频段 (ch1 ~ ch14)
- 支持IEEE802.11 d/e/h/i/k/v/w
- 内置PA和LNA, 集成TX/RX Switch
- 支持与BT/BLE芯片共存、Mesh组网
- 支持RF自校准方案
- 功耗: Deep Sleep模式: 5μA@3.3V
- DTIM1: 1.27mA@3.3V
- DTIM3: 0.495mA@3.3V

◆ PHY特性

- 支持IEEE802.11b/g/n单流所有的数据速率
- 最大速率: 72.2Mbps@HT20 MCS7
- 支持标准20MHz带宽和5M/10M窄带宽
- 支持STBC

◆ MAC特性

- 支持AMPDU、Block-ACK、AMSDU
- 支持Short-GI
- 支持QoS, 满足不同业务服务质量需求

◆ CPU子系统

- 高性能32bit CPU, 最大工作频率160MHz
- 内嵌SRAM 352KB, ROM 288KB, 用户可用RAM 160KB, 支持对接双云

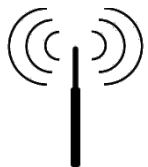
◆ 外围接口

- 1个SDIO Slave接口、2个SPI Master/Slave接口、2个I2C接口、3个UART接口、15个GPIO接口、7路ADC输入、5路PWM、一路I2S

◆ 其他信息

- 封装: QFN32 5×5mm
- 工作温度: -40°C ~ +85°C
- 内置2M flash
- 电源电压输入范围: 2.3V ~ 3.6V
- IO电源电压支持1.8V和3.3V

海思SoC WiFi 芯片，6大优势协同切入市场



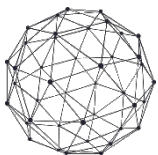
● 射频性能及抗干扰能力

- ✓ 发射功率及接收灵敏度普遍优于友商2~3dB，多穿一堵木质隔断
- ✓ 中强信号(RSSI>-75dBm)邻频干扰下，吞吐率不受影响



● 低功耗性能

- ✓ 同等条件下测试，常电功耗优于友商
- ✓ 外加32K晶体，DTIM模式再节电30%



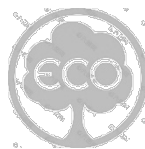
● 网状Mesh组网能力

- ✓ 针对IoT场景设计
- ✓ 连接节点数250+
- ✓ 路径备份及实时优化



● 集成多种安全能力

- ✓ 硬件加密，不消耗RAM资源，支持多种加密协议
- ✓ 支持安全启动，防止数据篡改



● 生态优势

- ✓ 预集成华为HiLink，苏宁智能，小京鱼等多种生态，缩短二次开发周期及认证周期
- ✓ 提供160k客户可用RAM，支持双云同时在线



● 一碰/靠近配网 (HiLink特性)

- ✓ 与SoftAP配网模式相比，进一步简化操作步骤，提升用户配网体验

注：此特性具体以hilink发布为准

Hilink配网集成，配网更快



- 1. 通过NFC/WiFi 靠近，设备自动发现
 - 2. 点击“确认”，自动完成配网及连接
- 说明：具体配网条件请依据hilink发布特性为准

常见配网方式对比一览表
(注：海思WiFi-IoT芯片同时支持SmartConfig及SoftAP等常规配网模式)

配网方式	方法	优势	局限性
Hilink配网	采用 NAN 协议或者私有协议，近距离接触传输SSID和密码	缩减配网步骤，成功率高，配网快	• 具体配网条件请依据hilink发布特性为准
SoftAP	设备启动为SoftAP模式，手机做STA连接SoftAP，通过socket将SSID和密码发送给设备	6步配网，当前最常见配网方式	• 需要手动连接智能设备WiFi网络 • 手动输入要连入连接的WiFi网络的SSID和密码；
SmartConfig	设备进入 混杂模式 接收空中的包，手机通过一组广播/组播包将SSID和密码进行编码，设备接收到后解码获得SSID和密码	操作简单，直接用手机配网	• 配网速度慢，且成功率低

目录

Hi3861规格及优势

Hi3861 软件架构

Hi3861 SDK目录及文档介绍

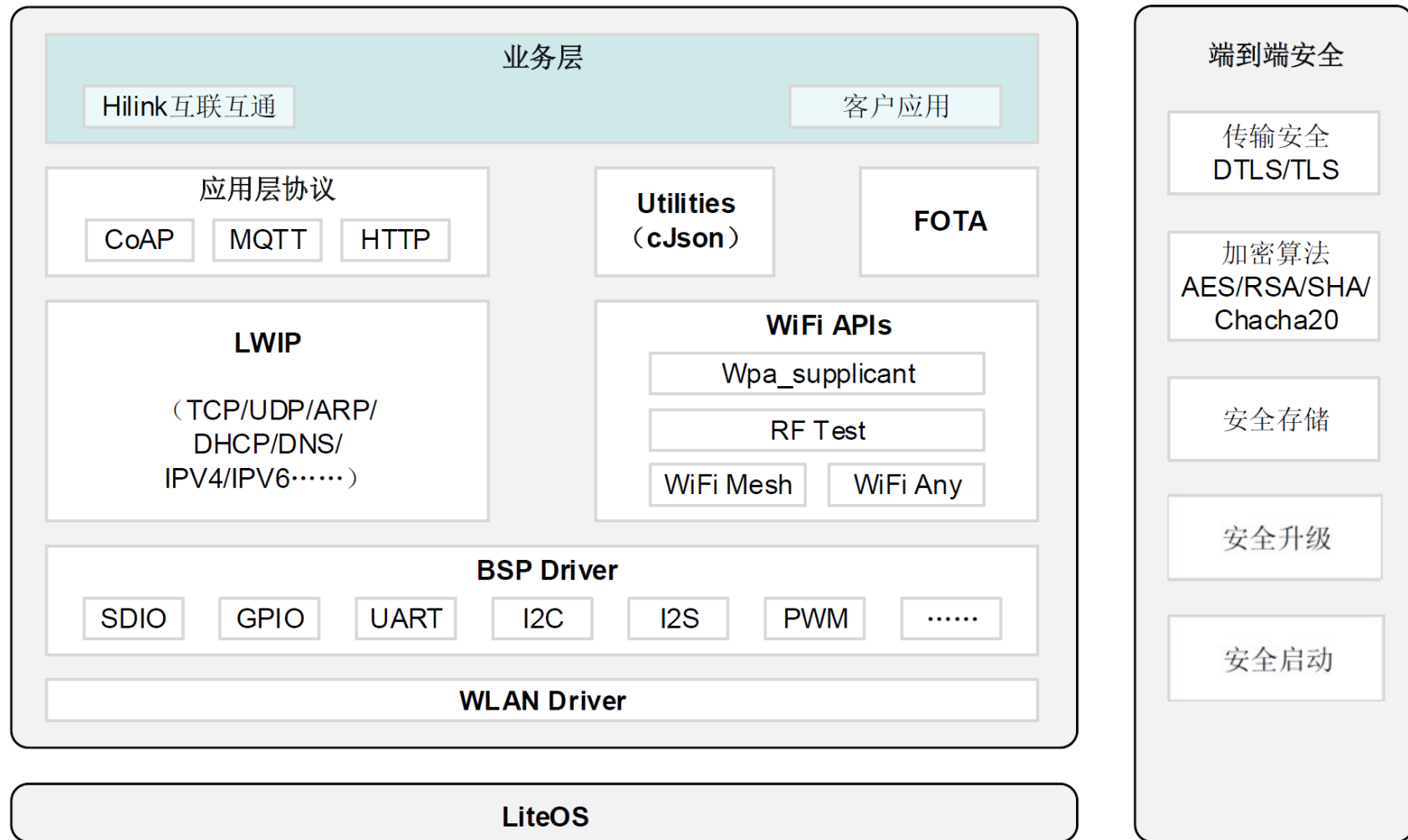
Hi3861 SDK使用介绍

Hi3861 软件架构

从下往上分为五层：

- 系统层
- 驱动层
- 网络服务层
- 组件层
- 业务层

安全作为公共组件分布在相关层中。



目录

Hi3861规格及优势

Hi3861 软件架构

Hi3861 SDK目录及文档介绍

Hi3861 SDK使用介绍

Hi3861 SDK 代码目录




























解压sdk，目录如下图所示：

└─ app	2020/9/24 21:25
└─ boot	2020/9/24 21:25
└─ build	2020/9/24 21:49
└─ components	2020/9/24 21:25
└─ config	2020/9/24 21:25
└─ include	2020/9/24 21:25
└─ output	2020/9/24 21:48
└─ platform	2020/9/24 21:25
└─ third_party	2020/9/24 21:25
└─ tools	2020/9/24 21:25
└─ build.sh	2020/9/24 21:25
└─ build_patch.sh	2020/9/24 21:25
└─ factory.mk	2020/9/24 21:25
└─ Makefile	2020/9/24 21:25
└─ non_factory.mk	2020/9/24 21:25
└─ SConstruct	2020/9/24 21:25

目录	说明
app	应用层代码（其中demo程序为参考示例）
boot	Flash bootloader代码
build	SDK构建所需的库文件、链接文件、配置文件
components	SDK组件目录
config	SDK系统配置文件
include	API头文件存放目录
output	编译时生成的目标文件和中间文件，包括库文件、打印log、生成的二进制文件等
platform	SDK平台相关的文件，包括镜像、内核驱动模块等
third_party	开源第三方软件目录
tools	SDK提供的linux和Windows系统上的使用工具，包括effuse 工具、NV工具、签名工具、Menuconfig等
build.sh	启动编译脚本，同时支持“sh build.sh menuconfig”进行客制化配置
build_patch.sh	Uboot补丁脚本，不用关注
factory.mk	可以生成产测固件的Makefile编译脚本
Makefile	Makefile编译脚本
non_factory.mk	不带产测固件的Makefile编译脚本
SConstruct	Scons编译脚本

Hi3861 参考示例

参考示例在app\demo\src目录下:

 app_demo_adc.c	 app_demo_uart.c
 app_demo_efuse.c	 app_demo_upg_verify.c
 app_demo_flash.c	 app_demo_upg_verify.h
 app_demo_i2c.c	 app_http_client.c
 app_demo_i2s.c	 app_main.c
 app_demo_io_gpio.c	 app_promis.c
 app_demo_nv.c	 app_promis.h
 app_demo_pwm.c	 es8311_codec.c
 app_demo_quick_send.c	 es8311_codec.h
 app_demo_sdio_device.c	 network_config_sample.c
 app_demo_sdio_device.h	 SConscript
 app_demo_spi.c	 wifi_softap.c
 app_demo_timer_systick.c	 wifi_sta.c
 app_demo_tsensor.c	

Hi3861 API

API接口头文件，大部分位于/include/base目录下，详细API接口参考文档《API开发参考.chm》

常用API:

printf(): 输出格式化字符串打印到串口，头文件: **hi_early_debug.h**

hi_task_xx: 任务创建，锁任务，sleep等，头文件: **hi_task.h**

hi_sem_xx: 信号量创建与等待，头文件: **hi_sem.h**

hi_mux_xx: 互斥锁创建于PEND/POST，头文件: **hi_mux.h**

memcpy_s: memcpy_s等标准库，头文件: **hi_stdlib.h**

hi_timer_start: 系统定时器，10ms精度，头文件: **hi_timer.h**

hi_hrtimer_start: 高精度定时器，us精度，头文件: **hi_hrtimer.h**

hi_get_tick: 系统tick获取等，头文件: **hi_time.h**

hi_uart_write: 串口读写操作，头文件: **hi_uart.h**

hi_event_wait: 事件创建与等待，头文件: **hi_event.h**

hi_flash_read: flash读写操作，头文件: **hi_flash.h**

hi_nv_read: NV读写操作，头文件: **hi_nv.h**

hi_malloc: 内存申请与释放，头文件: **hi_mem.h**

其它重要头文件:

hi_wifi_api.h: WIFI驱动接口函数

hi_types.h: 基础数据类型定义

hi_cipher.h: AES、RSA、TRNG等硬件安全算法

hi_errno.h: 错误码定义

Hi3861 编译生成文件说明

SDK根目录下执行 **sh build.sh**命令进行编译（默认压缩升级），编译生成的文件在output/bin目录下：






- Hi3861_boot_signed.bin
- Hi3861_boot_signed_B.bin
- Hi3861_demo.asm
- Hi3861_demo.map
- Hi3861_demo.out
- Hi3861_demo_allinone.bin
- Hi3861_demo_burn.bin
- Hi3861_demo_flash_boot_ota.bin
- Hi3861_demo_ota.bin
- Hi3861_demo_vercfg.bin
- Hi3861_loader_signed.bin

其中Hi3861_demo_allinone.bin就是烧写到底板中的程序。
















目录	说明	备注
Hi3861_boot_signed.bin	签名的bootloader文件	Flash boot
Hi3861_boot_signed_B.bin	签名的bootloader备份文件	Flash boot备份
Hi3861_demo.asm	Kernel asm文件	汇编程序源文件
Hi3861_demo.map	Kernel map文件	程序的全局符号，函数的地址、占用的空间等，用于调试，例如程序崩溃就可以查这个文件
Hi3861_demo.out	Kernel 输出文件	
Hi3861_demo_allinone.bin	产线工装烧写文件（已经包含独立烧写程序和loader程序）	包括2个bin: Hi3861_loader_signed.bin和Hi3861_demo_burn.bin
Hi3861_demo_burn.bin	Kernel烧写文件，建议直接使用Hi3861_demo_allinone.bin	默认包含boot、NV、可执行程序镜像
Hi3861_demo_flash_boot_ota.bin	Flash Boot升级文件	
Hi3861_demo_ota.bin	Kernel升级文件	
Hi3861_demo_vercfg.bin	Kernel和Boot的版本号文件	
Hi3861_loader_signed.bin	烧写工具使用的加载文件	只用在烧写，位于内存中。烧写最少需要Hi3861_loader_signed.bin和Hi3861_demo_burn.bin，这2个文件，allinone包括这2个，推荐使用allinone






Hi3861 文档

硬件相关文档:

-  Hi3861LV100 产品简介.pdf
-  Hi3861V100 产品简介.pdf
-  Hi3861V100 / Hi3861LV100 产线工装 用户指南.pdf
-  Hi3861V100 / Hi3861LV100 单板硬件关键器件 兼容性列表.pdf
-  Hi3861V100 / Hi3861LV100 开发板 使用指南.pdf

软件相关文档:

-  Hi3861V100 / Hi3861LV100 ANY软件 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 API 开发参考.chm
-  Hi3861V100 / Hi3861LV100 AT命令 使用指南.pdf
-  Hi3861V100 / Hi3861LV100 Boot API 开发参考.chm
-  Hi3861V100 / Hi3861LV100 Boot移植应用 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 cJSON 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 CoAP 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 EFUSE 使用指南.pdf
-  Hi3861V100 / Hi3861LV100 HiBurn工具 使用指南.pdf
-  Hi3861V100 / Hi3861LV100 HTTP 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 lwIP 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 Mesh AT命令 使用指南.pdf
-  Hi3861V100 / Hi3861LV100 Mesh软件 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 MQTT 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 NV 使用指南.pdf

-  Hi3861V100 / Hi3861LV100 开发板功耗 测试指南.pdf
-  Hi3861V100 / Hi3861LV100 射频 测试指南.pdf
-  Hi3861V100 / Hi3861LV100 硬件设计 Checklist.pdf
-  Hi3861V100 / Hi3861LV100 / Hi3881V100 WiFi芯片 硬件用户指南.pdf
-  Hi3861V100 / Hi3861LV100 / Hi3881V100 WiFi芯片 用户指南.pdf

-  Hi3861V100 / Hi3861LV100 OSA&FreeRTOS API适配指南.pdf
-  Hi3861V100 / Hi3861LV100 RPL 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 SDK 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 SDK开发环境搭建 用户指南.pdf
-  Hi3861V100 / Hi3861LV100 TLS&DTLS 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 Wi-Fi软件 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 安全模块 使用指南.pdf
-  Hi3861V100 / Hi3861LV100 常见问题 FAQ.pdf
-  Hi3861V100 / Hi3861LV100 单板冒烟 测试指南.pdf
-  Hi3861V100 / Hi3861LV100 低功耗 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 第三方软件 移植指南.pdf
-  Hi3861V100 / Hi3861LV100 二次开发网络安全 注意事项.pdf
-  Hi3861V100 / Hi3861LV100 设备驱动 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 升级 开发指南.pdf
-  Hi3861V100 / Hi3861LV100 文件系统 使用指南.pdf

目录

Hi3861规格及优势

Hi3861 软件架构

Hi3861 SDK目录及文档介绍

Hi3861 SDK使用介绍

Hi3861 程序入口

app/demo/src/app_main.c: app_main()

可在该函数末尾添加应用程序处理，如果需要添加新的app，参考SDK开发指南。

```
#ifndef CONFIG_QUICK_SEND_MODE
hi_void app_main(hi_void)
{
    (hi_void)hi_event_init(APP_INIT_EVENT_NUM, HI_NULL);
#ifdef CONFIG_FACTORY_TEST_MODE
    printf("factory test mode!\r\n");
#endif

    const hi_char* sdk_ver = hi_get_sdk_version();
    printf("sdk ver:%s\r\n", sdk_ver);

    hi_flash_partition_table *ptable = HI_NULL;

    peripheral_init();
    peripheral_init_no_sleep();

    ret = hi_wifi_init(APP_INIT_VAP_NUM, APP_INIT_USR_NUM);
    if (ret != HISI_OK) {
        printf("wifi init failed!\n");
    } else {
        printf("wifi init success!\n");
    }
    app_demo_task_release_mem(); /* Task used to release the sys
    printf("Hello World!\n");
```

程序运行效果示例：

```
ready to OS start
sdk ver:Hi3861V100R001C00SPC025 2020-09-24 21:10:00
wifi init success!
Hello World!
```

Flash分区、OTA升级及固件分离方案

Hi3861芯片内置2M Flash，默认分区结构如下：

Flash boot	32K
NV区	20K
Kernel A	912K
Kernel B	968K
.....	84K
Flash boot 备份区	32K

Hi3861 有两种固件 (bin)：

业务固件：

正常使用的固件，客户业务代码都在此固件中；从Kernel A起始地址开始存放。

产测固件：

用于产测测试的固件，例如设置Mac地址，设置校准参数等；存放于Kernel B的后600K。

产线流程：

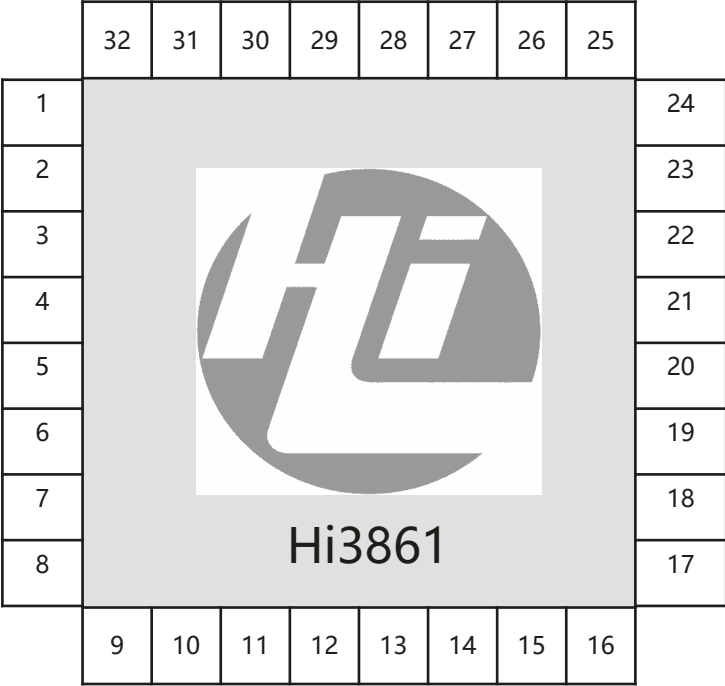
Menuconfig中配置产测宏，默认程序跳转到产测固件运行，等Mac地址写入，产线校准等参数调整之后，再通过AT+FTM=0指令跳转到业务固件，重启生效，然后在业务固件中执行AT+FTMERASE指令将产测固件删除，释放Flash空间。

OTA升级方案：

双分区升级：Kernel A和Kernel B都存放业务固件，哪个区固件版本新，使用哪个区固件。

压缩升级：Kernel A + B 做为一个整体分区，头部存放正常使用业务固件，压缩业务固件存放于尾部；升级时，压缩固件先下载到Flash，再解压缩搬到Kernel A起始地址。

Hi3861 管脚定义



封装：QFN32 5mm×5mm

共32个引脚，包括：15个GPIO接口 (复用)

PIN_NUMBER	PIN_NET	PIN_NUMBER	PIN_NET
1	VDDIO1	17	GPIO 05
2	GPIO 00	18	GPIO 06
3	GPIO 01	19	GPIO 07
4	GPIO 02	20	GPIO 08
5	GPIO 03	21	VDDIO2
6	GPIO 04	22	PMU PWRON
7	VDD WL RF LNA 1P2	23	VDD PMU CLDO
8	WL RF RFIO 2G	24	VDD BUCK 1P3
9	VDD WL RF PA2G 3P3	25	PMU BUCK LX
10	VDD WL RF TRX 1P2	26	VDD PMU VBAT1
11	VDD WL RF VCO 1P2	27	GPIO 09
12	VDD PMU RFLDO1	28	GPIO 10
13	VDD PMU 1P3	29	GPIO 11
14	VDD PMU VBAT2	30	GPIO 12
15	XOUT	31	GPIO 13
16	XIN	32	GPIO 14

管脚定义

Hi3861 IO功能定义

1. IO复用枚举定义在include路径下的hi_io.h接口文件。
2. IO复用接口调用在app/demo/init/app_io_init.c文件中的app_io_init函数中实现，如图：

```
hi_void app_io_init(hi_void)
{
    /* 用户需根据应用场景，合理选择各外设的IO复用配置，此处仅列出示例 */
    /* uart0 调试串口 */
    hi_io_set_func(HI_IO_NAME_GPIO_3, HI_IO_FUNC_GPIO_3_UART0_TXD); /* uart0 tx */
    hi_io_set_func(HI_IO_NAME_GPIO_4, HI_IO_FUNC_GPIO_4_UART0_RXD); /* uart0 rx */

    /* uart1 AT命令串口 */
    hi_io_set_func(HI_IO_NAME_GPIO_5, HI_IO_FUNC_GPIO_5_UART1_RXD); /* uart1 rx */
    hi_io_set_func(HI_IO_NAME_GPIO_6, HI_IO_FUNC_GPIO_6_UART1_TXD); /* uart1 tx */

    /* uart2 sigma认证使用串口 */
    hi_io_set_func(HI_IO_NAME_GPIO_11, HI_IO_FUNC_GPIO_11_UART2_TXD); /* uart2 tx */
    hi_io_set_func(HI_IO_NAME_GPIO_12, HI_IO_FUNC_GPIO_12_UART2_RXD); /* uart2 rx */

    /* SPI MUX: */
    #ifdef CONFIG_SPI_SUPPORT
    /* SPI IO复用也可以选择5/6/7/8;0/1/2/3，根据产品设计选择 */
    hi_io_set_func(HI_IO_NAME_GPIO_9, HI_IO_FUNC_GPIO_9_SPI0_TXD);
    hi_io_set_func(HI_IO_NAME_GPIO_10, HI_IO_FUNC_GPIO_10_SPI0_CK);
    hi_io_set_func(HI_IO_NAME_GPIO_11, HI_IO_FUNC_GPIO_11_SPI0_RXD);
    #endif
}
```
3. 修改完成之后，重新编译生成新的bin文件，然后烧写到板子上验证。
- 通过串口工具命令窗口输入AT指令查看修改是否生效：
- AT+GETIOMODE=pin (GPIO引脚编号)
- 返回结果第二位代表IO工作模式，如图：

```
# AT+GETIOMODE=9
+GETIOMODE:9,0,0,3
OK
```

IO工作模式表

模式 IO号	0	1	2	3	4	5	6	7
0	GPIO0	Reserved	UART1_TXD	SPI1_CK	JTAG_TDO	PWM3_OUT	I2C1_SDA	Reserved
1	GPIO1	Reserved	UART1_RXD	SPI1_RXD	JTAG_TCK	PWM4_OUT	I2C1_SCL	BT_FREQ
2	GPIO2	Reserved	UART1_RSTN	SPI1_TXD	JTAG_TRSTN	PWM2_OUT	Reserved	SSI_CLK
3	GPIO3	UART0_TXD	UART1_CTS_N	SPI1_CSN	JTAG_TDI	PWM5_OUT	I2C1_SDA	SSI_DATA
4	GPIO4	Reserved	UART0_RXD	Reserved	JTAG_TMS	PWM1_OUT	I2C1_SCL	Reserved
5	GPIO5	Reserved	UART0_RXD	SPI0_CSN	Reserved	PWM2_OUT	I2S0_MCLK	BT_STATUS
6	GPIO6	Reserved	UART1_TXD	SPI0_CK	Reserved	PWM3_OUT	I2S0_TX	COEX_SWITCH
7	GPIO7	Reserved	UART1_CTS_N	SPI0_RXD	Reserved	PWM0_OUT	I2S0_BCLK	BT_ACTIVE
8	GPIO8	Reserved	UART1_RSTN	SPI0_TXD	Reserved	PWM1_OUT	I2S0_WS	WLAN_ACTIVE
9	GPIO9	I2C0_SCL	UART2_RSTN	SDIO_D2	SPI0_TXD	PWM0_OUT	Reserved	I2S0_MCLK
10	GPIO10	I2C0_SDA	UART2_CTS_N	SDIO_D3	SPI0_CK	PWM1_OUT	Reserved	I2S0_TX
11	GPIO11	Reserved	UART2_TXD	SDIO_CMD	SPI0_RXD	PWM2_OUT	RF_TX_EN_EXT	I2S0_RX
12	GPIO12	Reserved	UART2_RXD	SDIO_CLK	SPI0_CS_N	PWM3_OUT	RF_RX_EN_EXT	I2S0_BCLK
13	SSI_DATA	UART0_TXD	UART2_RSTN	SDIO_D0	GPIO13	PWM4_OUT	I2C0_SDA	I2S0_WS
14	SSI_CLK	UART0_RXD	UART2_CTS_N	SDIO_D1	GPIO14	PWM5_OUT	I2C0_SCL	Reserved

Hi3861 串口

Hi3861芯片有3个UART接口，默认情况下：

- **UART0** 是调试串口（复用3/4引脚）：烧写程序，打印日志信息
- **UART1** 是业务串口（复用5/6引脚）：AT以及客户业务串口
- **UART2** 用于sigma认证（复用11/12引脚）

```
/* 用户需根据应用场景，合理选择各外设的io复用配置，此处仅列出示例 */
#ifdef CONFIG_UART0_SUPPORT
/* uart0 调试串口 */
hi_io_set_func(HI_IO_NAME_GPIO_3, HI_IO_FUNC_GPIO_3_UART0_TXD); /* uart0 tx */
hi_io_set_func(HI_IO_NAME_GPIO_4, HI_IO_FUNC_GPIO_4_UART0_RXD); /* uart0 rx */
#endif

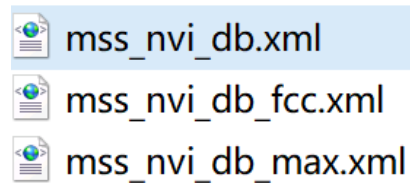
#ifdef CONFIG_UART1_SUPPORT
/* uart1 AT命令串口 */
hi_io_set_func(HI_IO_NAME_GPIO_5, HI_IO_FUNC_GPIO_5_UART1_RXD); /* uart1 rx */
hi_io_set_func(HI_IO_NAME_GPIO_6, HI_IO_FUNC_GPIO_6_UART1_TXD); /* uart1 tx */
#endif

#ifdef CONFIG_UART2_SUPPORT
/* uart2 sigma认证使用串口 */
hi_io_set_func(HI_IO_NAME_GPIO_11, HI_IO_FUNC_GPIO_11_UART2_TXD); /* uart2 tx */
hi_io_set_func(HI_IO_NAME_GPIO_12, HI_IO_FUNC_GPIO_12_UART2_RXD); /* uart2 rx */
#endif
```

两种方式修改默认串口：

第一种：修改NV配置，重新编译应用程序

1、NV配置文件位于SDK目录下：tools/nvtool/xml_file 3个文件



2、找到ID= “0x42” 项中PARAM_VALUE="{1,0,2,0}"

3、PARAM_VALUE 前3个参数分别对应AT命令串口、调试串口、sigma认证使用串口，默认值为1,0,2

4、修改相应值重新编译固件即可

5、例如AT命令使用UART0，就需要配置成0,0,2

第二种：程序运行过程中，通过AT命令修改配置，单板重启后生效

AT+SETUART=1,0,2

Hi3861 MAC地址

MAC地址相关接口:

1、int hi_wifi_get_macaddr(char *mac_addr, unsigned char mac_len);

接口说明: mac地址获取接口, 优先使用内存中的mac地址(系统启动读取), 如果没有使用Flash中的mac地址, 如果还是没有就使用efuse中的mac地址, 如果还是没有会随机生成一个mac地址。

对应AT命令: AT+MAC?

3、unsigned int wal_set_customer_mac(const char *mac_addr, unsigned char type)

接口说明: 设置mac地址到efuse或者Flash NV中, 参数type: 写入类型, 0: efuse 1: Flash NV区; **注意: 每块单板有3次写入eFuse的机会, 写入nvram的次数建议不超过20次。**

对应AT命令: AT+EFUSEMAC=90:2B:D2:E4:CE:28,0

2、int hi_wifi_set_macaddr(const char *mac_addr, unsigned char mac_len)

接口说明: 设置mac地址, 调用之后存到内存中, 模组重启后此接口设置的mac失效, 需重新调用接口设置。

对应AT命令: AT+MAC=90:2B:D2:E4:CE:28

4、unsigned int wal_get_customer_mac(void);

接口说明: 优先从nvram读取, 如果无效, 则从eFuse读取MAC地址返回。

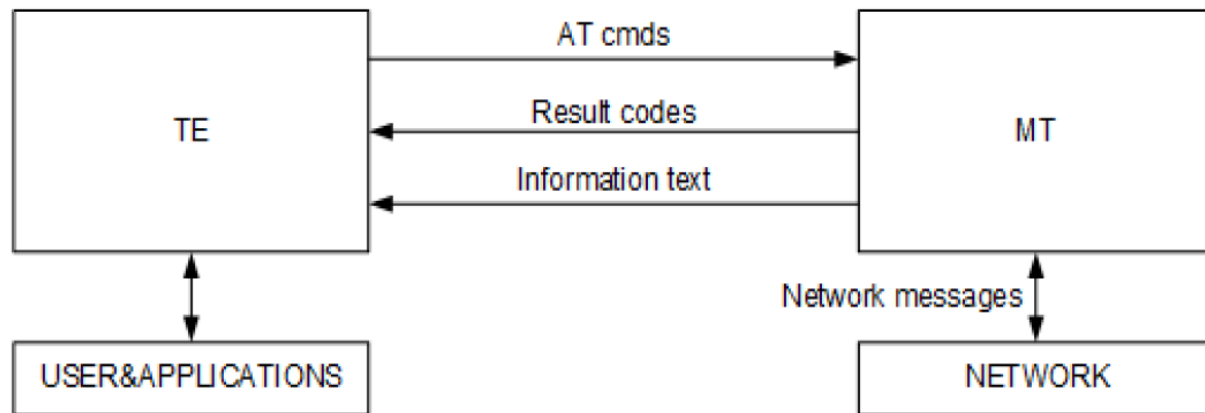
对应AT命令: AT+EFUSEMAC?

建议:

如果拿到的板子上没有设置过mac地址, 每次启动会随机生成, 就需要固定mac, 而写mac地址到Flash或者efuse的AT命令在产测固件, 所以勾选产测选项, 编译生成包含产测固件烧录文件, 进入产测模式, 设置mac地址到efuse, 再切换到业务模式

Hi3861 AT命令

AT 命令用于控制TE（例如：PC 等用户终端）和MT（例如：移动台等移动终端）之间交互的规则，如下图所示：



Hi3861的AT命令实现了STA、AP相关wifi功能、TCP/IP相关功能、测试调试相关功能、IO相关功能及基本信息的查询设置功能，AT命令详细使用参考文档《Hi3861V100 / Hi3861LV100 AT命令 使用指南.pdf》

注意事项：

- 1、AT命令必须大写，且没有任何空格
- 2、AT命令必须以回车换行符作为结尾（CR LF），部分串口工具在用户敲击键盘回车键时只有回车符（CR）没有换行符（LF），导致AT指令无法识别，如需使用串口工具手动输入AT指令，需在串口工具中将回车键设置为回车符（CR）+换行符（LF）
- 3、AT串口默认使用UART1口
- 4、AT命令业务固件和产测固件不完全一致，产测固件包括校准参数、mac地址写入等，请使用AT+HELP查看支持命令

Hi3861 AT命令使用示例

SoftAP:

启动SoftAP示例:

AT+MAC=90:2B:D2:E4:CE:28

AT+STARTAP="hisilicon",0,6,2,"123456789"

AT+IFCFG=ap0,192.168.3.1,netmask,255.255.255.0,gateway,192.168.3.2

AT+DHCP=ap0,1

注意: 设置MAC地址命令可选, 如果不设置则使用随机MAC; 设置的MAC地址为STA的地址, SoftAP的地址为STA的地址+1。

关闭SoftAP示例:

AT+DHCP=ap0,0

AT+STOPAP

STA:

启动STA示例:

AT+MAC=90:2B:D2:E4:CE:28

AT+STARTSTA

AT+SCAN

AT+SCANRESULT

AT+CONN="hisilicon",,"123456789"

AT+STASTAT

AT+DHCP=wlan0,1

关闭STA示例:

AT+DHCP=wlan0,0

AT+STOPSTA

Hi3861 新增AT命令

1. AT指令初始化在app_main.c的入口函数里:

hi_at_sys_cmd_register();

2. 在components\at\src\目录下创建自定义的AT指令.c和.h文件, 例如: at_test.c/at_test.h

3. 举例: 要增加**AT+MYAT**指令, at_test.c实现代码:

```
hi_u32 at_hi_test(void)
{
    printf("at cmd test!!\n");
    return HI_ERR_SUCCESS;
}

at_cmd_func g_at_test_func_tbl[] = {
    {"+MYAT", 5, HI_NULL, HI_NULL, HI_NULL, (at_call_back_func)at_hi_test},
};

#define AT_TEST_FUNC_NUM (sizeof(g_at_test_func_tbl) / sizeof(g_at_test_func_tbl[0]))

void hi_at_test_cmd_register(void)
{
    hi_at_register_cmd(g_at_test_func_tbl, AT_TEST_FUNC_NUM);
}
```

4. 在components\at\src\hi_at.c 文件hi_at_sys_cmd_register函数中调用at_test.c中定义的注册方法:

```
hi_void hi_at_sys_cmd_register(hi_void)
{
    hi_at_general_cmd_register();
    hi_at_sta_cmd_register();
    hi_at_softap_cmd_register();
    hi_at_hipriv_cmd_register();
    hi_at_is_cmd_register();
    hi_at_test_cmd_register();
#ifdef LOSCFG_APP_MESH
    hi_at_mesh_cmd_register();
#endif
}
```

5. 重新编译, 烧写新的bin文件。

6. 测试AT+MYAT:

```
# AT+HELP
+HELPP
AT
AT+CSV          AT+HELP          AT+MYATT          AT+SYSINFO
AT+RST          AT+WREG          AT+RREG
AT+DHCP         AT+DHCP          AT+IFCFG          AT+MAC
AT+NETSTAT      AT+IPERF         AT+PING           AT+PING6
AT+DNS          AT+STARTSTA      AT+STOPSTA        AT+SCAN
AT+SCANCHN      AT+SCANSSID      AT+SCANPRSSID     AT+SCANRESULT
AT+CONN         AT+FCONN         AT+DISCONN        AT+STASTAT
AT+RECONN       AT+PBC           AT+PIN            AA+PIISHOW
AT+CALCPSK      AT+STARTAP       AT+SETAPADV       AT+STOPAP
AT+SHOWSTA      AT+DEAUTHSTA     AT+APSCAN         AT+ALTX
AT+ALRX         AT+RXINFO        AT+CC             AT+TPC
AT+SETIOMODE    AT+GETIOMODE     AT+GPIODIR        AT+WTGPIO
AT+RDGPIO       AT+MYAT
OK

# AT+MYAT
at cmd test!!
```



筑就智能时代基石

Copyright©2021 Shanghai HiSilicon Technologies Co., Ltd.
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Shanghai HiSilicon may change the information at any time without notice.