

使用指南

N32WB03x系列软件开发指南

简介

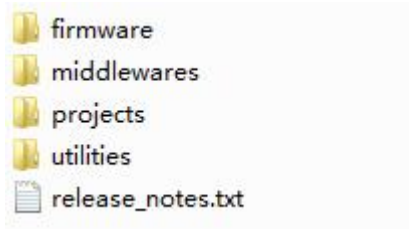
此文档的目的在于让使用者能够快速熟悉N32WB03x系列蓝牙SOC芯片的开发套件以及软件开发中的重要注意事项，旨在帮助开发者尽快熟悉该平台开发套件资料的使用及加速项目的开发进展。

目录

简介	1
1 SDK目录结构	3
1.1 FIRMWARE	3
1.2 MIDDLEWARE	3
1.3 UTILITIES	4
1.4 PROJECTS	4
2 快速启动蓝牙应用软件开发	7
2.1 编译环境安装	7
2.2 固件支持包安装	7
2.3 选择蓝牙例程	7
2.4 添加应用任务	8
2.5 编译环境配置	9
2.5.1 Flash 配置	9
2.5.2 RAM 配置	10
2.5.3 编译环境配置	10
2.6 编译与下载	11
3 外设开发注意事项	13
3.1 RCC	13
3.2 FLASH	13
3.3 PWR	13
3.4 EXTI	14
3.5 HSI	14
3.6 IIC	15
3.7 ADC	16
3.8 NVIC	16
3.9 异常处理	16
4 版本历史	19
5 声明	20

1 SDK目录结构

SDK 目录下是一个以固件库版本命名的文件，其中包含四个子文件夹：



1.1 firmware

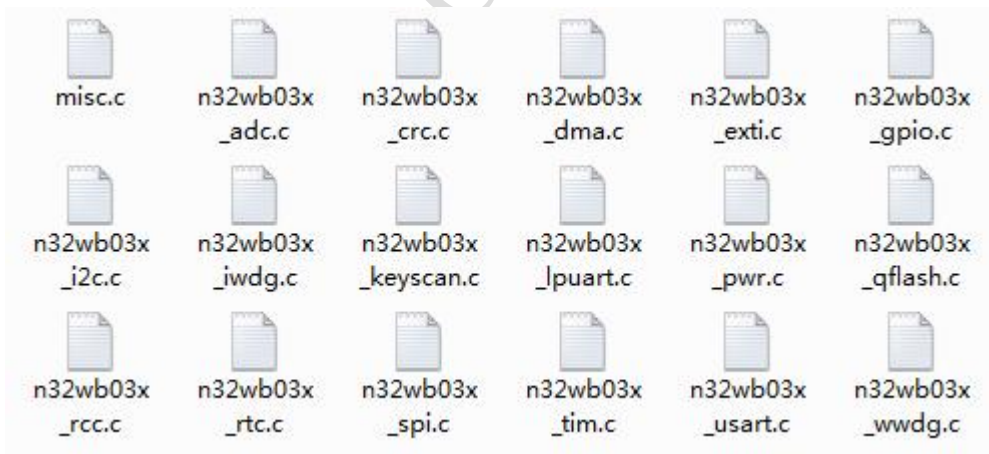
■ CMSIS:

微控制器软件接口标准，是Cortex-M处理器系列的与供应商无关的硬件抽象层，CMSIS提供了内核与外设、实时操作系统和中间设备之间的通用接口。

- 包含了用来访问内核的寄存器设备的名称定义，地址定义和配置函数。该接口包括调试通道定义。
- 提供片上所有外设的定义，包括所有外设寄存器头文件、启动文件、系统初始化模板文件。

■ n32wb03x_std_periph_driver:

芯片外设的标准驱动函数，包括.c 的源文件和.h 的头文件。用户可移植到项目中，快速完成对某个外设模块的使用。



1.2 middleware

中间层固件，主要如下两个目录：

■ Third_Party

- FreeRTOS: FreeRTOS 相关库

■ Nationstech

- ble_library: 蓝牙 BLE 相关库
 - ◆ ns_ble_stack: ble 蓝牙协议栈头文件
 - ◆ ns_ble_profile: ble profile 库源码
 - ◆ ns_library: ble 程序其他库源码, 包含 log, sleep 和 timer 等

1.3 utilities

工具软件目录, 里面主要包含:

■ dfu

dfu 固件升级相关工具软件目录和调用这些工具的 bat 脚本

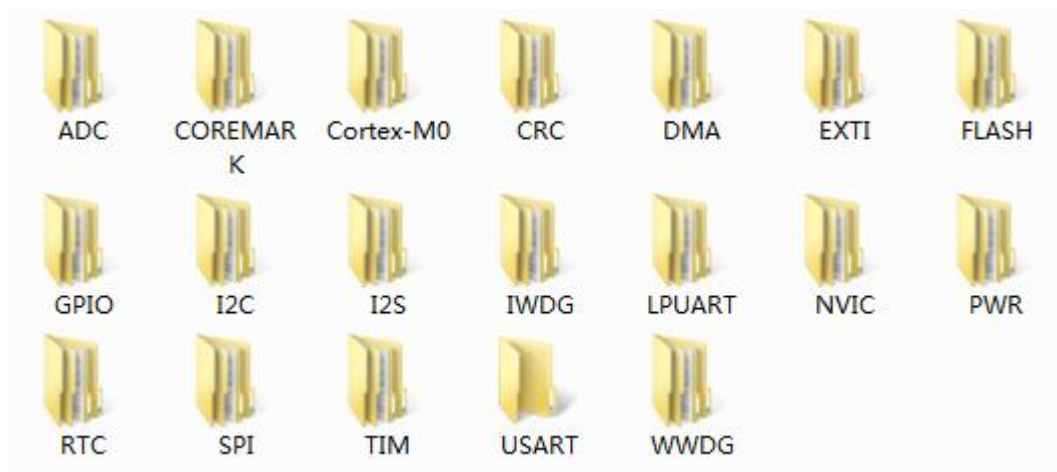
- Image: 使用 bat 脚本生成的 bin 文件和用于 DFU 演示的 bin 文件
- JLink: Jlink 烧录工具
- Keys: 用于生成 dfu 升级 bin 的密钥
- NSAndroidUtil: DFU 测试 APK
- NSpyocd: NSLink 烧录工具
- NSUtil: 脚本上位机工具软件

1.4 projects

Projects 目录包含开发板目录, 开发板目录下包含:

- bsp: 包含调试串口打印功能的 log 函数, 用于外设例程在调试过程答应各种调试信息。
- application: 应用例程, 使用了多种外设或者功能需求的综合性例程
 - peripheral_alone: 外设综合例程, 包含串口通信, TIM 的 PWM 输出, ADC 读取和转为电压, IO 输出, IO 输入中断响应和 RTC 休眠唤醒
 - FreeRTOS: Freertos 相关例程
- ble: 蓝牙从机例程, 通过对具体 profile 的蓝牙例程了解蓝牙程序开发的基本方法
 - beacon: ibeacon 信标例程
 - dis: 设备信息服务例程
 - hid_mouse: 蓝牙鼠标例程

- rdtss: 蓝牙数传例程 (128bit UUID), 包含 DFU 空中升级项目选项
- rdtss_16bit: 蓝牙数传例程 (16bit UUID), 包含 DFU 空中升级项目选项
- rdtss_3s: 蓝牙数传例程 (128bit UUID), 支持连接 3 个主机设备的多连接例程。
- heart_rate: 蓝牙心率服务例程
- blood_pressure: 蓝牙血压服务例程。
- lesc&aes: BLE 安全连接及 AES 应用例程
- ble_central: 蓝牙主机例程
 - central_rdtsc: 蓝牙数传服务主机例程。可以配合 rdtss 例程使用。
 - central_rdtsc_c&s: 蓝牙数传服务主机和从机模式切换例程。可以配合 rdtss 或 central_rdtsc 例程使用。
 - central_rdtsc_3m: 蓝牙数传服务主机例程, 支持连接 3 个从机的多连接例程。
 - central_relay_1m1s: 蓝牙数传服务中继例程, 即一主一从同时工作。可以配合 rdtss 例程使用。
 - central_relay_1m2s: 蓝牙数传服务中继例程, 即一主二从同时工作。可以配合 rdtss 例程使用。
 - central_rdtsc_16bit: 蓝牙数传服务 (16bit UUID) 主机例程。可以配合 rdtss_16bit 例程使用
 - central_hogprh: 蓝牙 HID 服务 client 端主机例程。可以配合 hid_mouse 例程使用
- dfu: 设备固件升级例程
 - common: dfu 公共库目录
 - app_ota: 蓝牙空中升级例程
 - image_update: 单 bank 模式下, image_update 例程源码
 - masterboot: 固件升级 boot, 包含串口升级, 固件校验和跳转
- prod_test: 量产测试例程
 - DTM_Test: 标准 HCI 接口 DTM 指令测试例程, 默认串口 USART1 (PB6 和 PB7), 需要连接蓝牙测试仪测试。
 - RF_Test: 支持串口指令启动 BLE TX, RX 和定频测试, 可以连接 PC 上位机发送指令。
 - DTM_Test 及 RF_Test 的测试操作步骤说明, 可找 FAE 获取对应的应用笔记。
- peripheral: 包含各个外设功能模块的例程项目, 实现每个外设模块的基本功能应用开发, 用户可以通过这些例程项目快速了解芯片外设用法。



2 快速启动蓝牙应用软件开发

2.1 编译环境安装

请安装KEIL MDK-ARM开发环境，推荐使用V5.0.0以上版本。

2.2 固件支持包安装

双击运行N32WB03x_DFP.1.4.0.pack安装Keil芯片支持包。

2.3 选择蓝牙例程

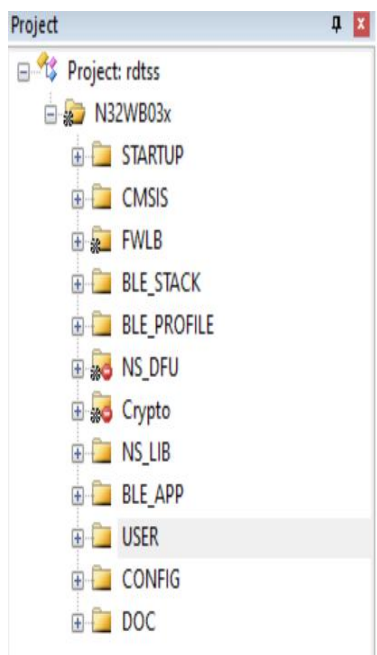
根据用户蓝牙功能需求，参考1.4章节选择相应的蓝牙demo例程。

以rdtss蓝牙串口透传项目工程为例，打开keil工程。

■ Project Target

- N32WB03x: 蓝牙工程，不带DFU配置，其他ble项目只有这个Target
- OTA_IMG_1: 带蓝牙OTA工程，配置为APP1地址0x1004000
- OTA_IMG_2: 带蓝牙OTA工程，配置为APP2地址0x1020000

■ 目录结构如下



- STARTUP: 芯片启动文件
- CMSIS: 芯片内核配置
- FWLB: 芯片外设驱动库
- BLE_STACK: 蓝牙BLE协议栈
- BLE_PROFILE: 蓝牙BLE profile
- NS_DUF (可选): 蓝牙OTA固件升级相关库
- Crypto (可选): 蓝牙OTA固件升级使用的加密相关库
- NS_LIB: 蓝牙应用相关库
- BLE_APP: 蓝牙应用代码
- USER: 用户应用代码
- CONFIG: 配置文件
- DOC: 说明文档

2.4 添加应用任务

为避免程序阻塞，蓝牙任务采用消息回调函数方式，在应用程序中先注册消息任务，然后基于中断或定时消息触发任务；尽量将蓝牙任务碎片化，采用多个消息串联任务处理，避免单个消息处理耗时过长，保证主循环中的调度函数`rwip_schedule()`运行的实时性。

(1) 基于SDK蓝牙例程代码框架，添加定时任务示例代码（**用户参考颜色高亮部分添加应用任务**）：

在SDK `app_ble_init(void)`函数中已配置回调函数`app_ble_msg_handler`与`app_user_msg_handler`。

```
void app_ble_init(void)
```

```
{  
  
    struct ns_stack_cfg_t app_handler = {0};  
  
    app_handler.ble_msg_handler = app_ble_msg_handler;  
    app_handler.user_msg_handler = app_user_msg_handler;  
  
}
```

调用`ke_timer_set(APP_LED_TEST_EVT, TASK_APP, 1000)`；ROM协议栈会产生`ke_timer`中断，上报消息给到应用层，应用层从消息列表`KE_MSG_HANDLER_TAB`里面找对应的消息ID。

```
KE_MSG_HANDLER_TAB(app)
```

```
{  
  
    // Note: first message is latest message checked by kernel so default is put on top.  
    {KE_MSG_DEFAULT_HANDLER, (ke_msg_func_t)app_entry_point_handler},  
  
}
```

```
enum user_msg_id
```

```
{  
  
    APP_USER_DEMO_EVE = APP_FREE_EVE_FOR_USER,  
    APP_CUSTS_TEST_EVT,  
    APP_LED_TEST_EVD, //添加一个LED闪烁任务ID  
  
};
```

```
void app_ble_msg_handler(struct ble_msg_t const *p_ble_msg)
```

```
{  
  
    switch (p_ble_msg->msg_id)  
    {  
  
        case APP_BLE_OS_READY:  
  
            NS_LOG_INFO("APP_BLE_OS_READY\r\n");
```

```
ke_timer_set(APP_LED_TEST_EVT, TASK_APP, 1000); //1秒后启动LED闪烁任务

break;

default:

break;

}

}

void app_user_msg_handler(ke_msg_id_t const msgid, void const *p_param)
{
switch (msgid)
{
case APP_LED_TEST_EVT: //LED闪烁任务执行
Ledlink(LED1_PORT, LED1_PIN);
ke_timer_set(APP_LED_TEST_EVT, TASK_APP, 1000); //1秒后启动LED闪烁任务
break;

default:

break;

}

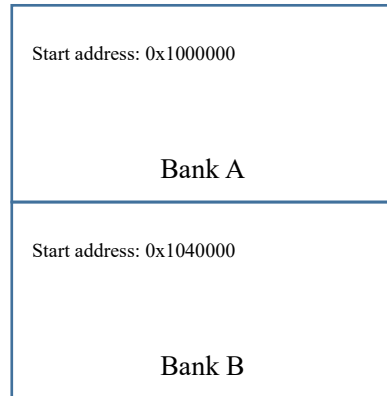
}
```

2.5 编译环境配置

2.5.1 Flash配置

N32WB031KE系列芯片FLASH地址范围是0x01000000 - 0x0107FFFF，总空间为512K字节，分为BankA、BankB两个Bank，空间大小分别是256K字节，BankA地址范围是0x01000000 - 0x0103FFFF，BankB地址范围是0x01040000 - 0x0107FFFF。用户代码运行范围只能是BankA，BankB用于数据存储。注意N32WB031KC系列FLASH空间是256K，即只有BankA区域。

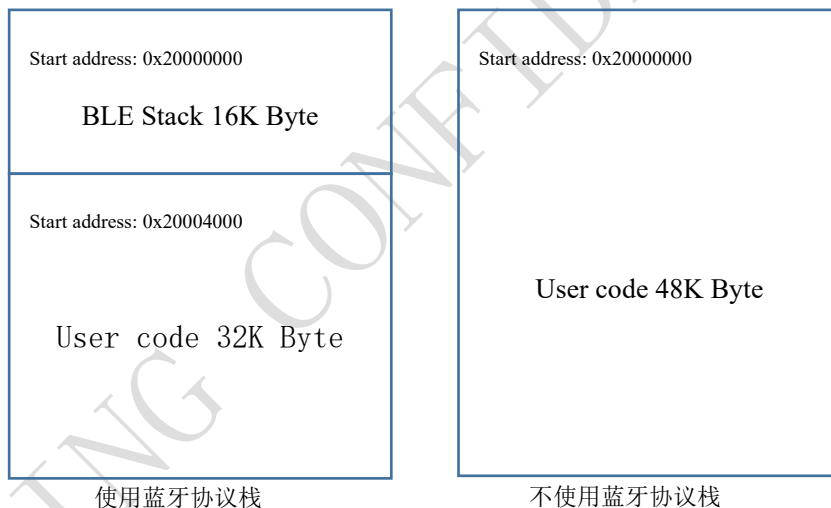
需要使用固件升级功能请参考升级指南《CN_UG_N32WB03x Firmware Update User Guide》里面的FLASH内存分布章节。



2.5.2 RAM配置

N32WB031芯片RAM地址范围是0x20000000 - 0x2000BFFF，可用空间为48K字节

- 如果使用蓝牙功能，蓝牙协议栈将占用0x20000000 - 0x20003FFF，总共16K字节RAM，用户代码可以使用0x20004000 - 0x2000BFFF，总共32K字节RAM
- 如果不用蓝牙功能，则用户代码可以使用全部48K字节RAM



2.5.3 编译环境配置

注：所有例程已经按如下配置完成，所以运行例程不需要重新配置编译环境。

- Target页的FLASH和RAM配置

- 不包含ble功能项目

on-chip			
☒	IRAM1:	0x1000000	0x40000
☐	IRAM2:		

on-chip			
☒	IRAM1:	0x20000000	0xC000
☐	IRAM2:		

- 包含ble功能项目target页



on-chip

☒ IROM1: 0x1000000 0x40000

☐ IROM2:

on-chip

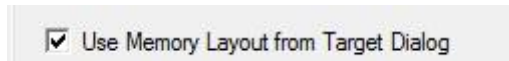
☒ IRAM1: 0x20004000 0x8000

☐ IRAM2:

- 带DFU功能项目，请参考《CN_UG_N32WB03x Firmware Update User Guide》

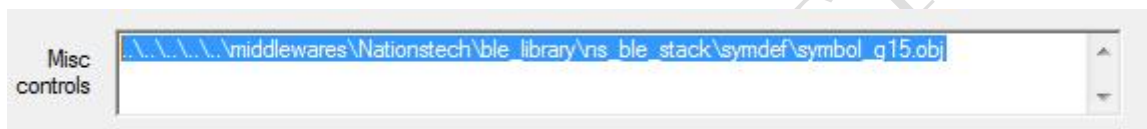
■ Linker页

- 勾选使用Target页的memory配置



☒ Use Memory Layout from Target Dialog

- 带ble项目需要加middlewares\Nationtech\ble_library\ns_ble_stack\symdef\symbol_g15.obj文件于Misc control，注意包含蓝牙主机功能的程序使用的obj文件是symbol_g15_central.obj

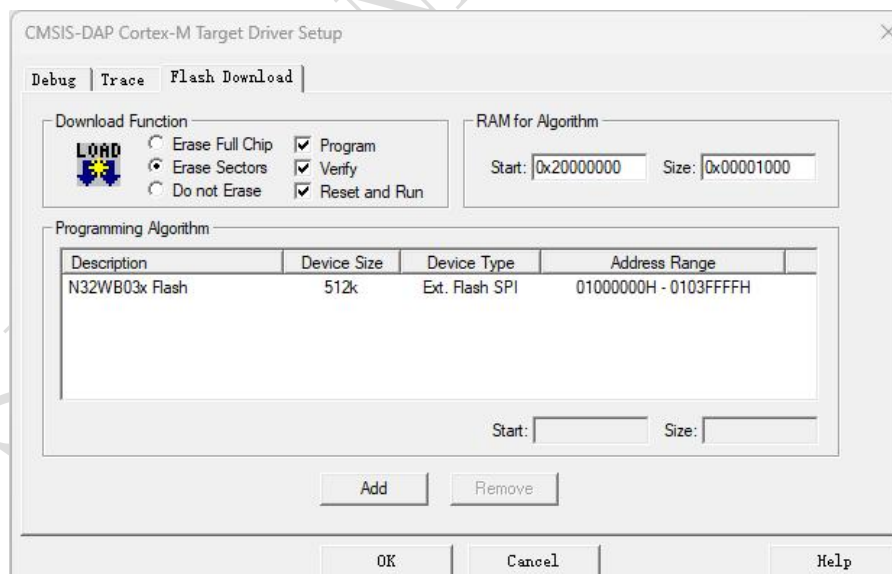


Misc controls

\\.\..\..\middlewares\Nationtech\ble_library\ns_ble_stack\symdef\symbol_g15.obj

■ Debug页

- 开发板上的NS-Link需要选择调试器CMSIS-DAP Debugger
- Setting页面如下图



CMSIS-DAP Cortex-M Target Driver Setup

Debug | Trace | Flash Download

Download Function

☒ Erase Full Chip ☒ Program

☒ Erase Sectors ☒ Verify

☐ Do not Erase ☒ Reset and Run

RAM for Algorithm

Start: 0x20000000 Size: 0x00001000

Programming Algorithm

Description	Device Size	Device Type	Address Range
N32WB03x Flash	512k	Ext. Flash SPI	01000000H - 0103FFFFH

Start: Size:

Add Remove

OK Cancel Help

2.6 编译与下载

- 编译：点击菜单的build按钮 
- 下载：点击菜单的download按钮 
- 调试：点击菜单的start/stop debug session按钮 

注意：下载程序后无法继续再下载

- 可能1： 代码中将SWDIO/SWDCLK用作其他用途，导致仿真接口失效。一般情况下不建议使用仿真引脚用作其他功能设计
- 可能2： 芯片进入低功耗sleep模式，此时仿真接口失效，需要在唤醒状态下下载。

NSING CONFIDENTIAL

3 外设开发注意事项

3.1 RCC

所有蓝牙例程默认使用HSI 64M作为系统时钟，使用蓝牙功能必须外挂32MHz晶体。高速时钟外设及IWDG在退出sleep模式后需重新初始化。

3.2 FLASH

Flash接口函数如下：

`void Qflash_Init(void);` //使用之前必须初始化，只需初始化一次

`void Qflash_Erase_Sector(uint32_t address);` //擦除一个扇区操作

`void Qflash_Write(uint32_t address, uint8_t* p_data, uint32_t len);` //写操作

`void Qflash_Read(uint32_t address, uint8_t* p_data, uint32_t len);` //读操作，地址必须是4字节对齐

Flash扇区（4KByte）擦除耗时16ms，执行时有屏蔽全局中断，会影响蓝牙中断的响应。要求在蓝牙空闲时执行Flash擦写操作。示例代码(SDK ns_sec.c中蓝牙配对信息保存)如下：

`{APP_BOND_STORE_EVT, (ke_msg_func_t)ns_sec_bond_store_evt_handler}` //注册消息任务

`ke_timer_set(APP_BOND_STORE_EVT, TASK_APP, 20)` //20ms后启动消息任务

`int ns_sec_bond_store_evt_handler(ke_msg_id_t const msgid, void *p_param, ke_task_id_t const dest_id, ke_task_id_t const src_id)` //消息任务执行函数

3.3 PWR

针对低功耗的sleep模式休眠，蓝牙工程中调用封装好的ns_sleep()函数，不需要蓝牙功能的外设工程调用PWR_EnterSLEEPMode(PWR_SLEEPENTRY_WFI)。

休眠前：SysTick和TIMER等外设（不支持低功耗唤醒）启用中断后，进sleep前需关闭它们的中断。

休眠前相关处理建议放在void app_sleep_prepare_proc(void)中执行。

唤醒后：高速时钟外设及IWDG在sleep唤醒后需重新初始化。

唤醒后相关处理建议放在void app_sleep_resume_proc(void)中执行，但IWDG的重新初始化需在entry_sleep(void)中执行，参考NS_WAKEUP_CONFIG()调用。

蓝牙广播，扫描和连接等蓝牙任务期间会周期性动态执行休眠唤醒，如需长时间休眠，需停止所有的蓝牙任务。

3.4 EXTI

针对EXTI唤醒，须将ns_sleep.c运行在ram中，确保sleep流程时序固定，并在__WFI()前增加如下代码：

```
RCC->CFG &= ~RCC_CFG_APB1PRES;  
RCC->CFG |= RCC_HCLK_DIV16; //APB1时钟16分频  
RCC->CFG &= ~RCC_CFG_APB2PRES;  
RCC->CFG |= RCC_HCLK_DIV16; //APB2时钟16分频  
EXTI->PEND = 0x3fff; //清除已产生的EXTI中断  
__nop(); __nop(); __nop();.....//连续执行42个__nop();
```

3.5 HSI

HSI支持校准功能，在系统时钟初始化函数SystemInit()后面，可以添加HSI校准函数RCC_HsiCalib_64M()，示例代码如下：

```
uint32_t RCC_HsiCalib_64M(void)  
{  
    uint32_t tmp=0,trim_value = 0;  
    uint32_t hsi_timeoutcnt = 0;  
    uint32_t g_timeoutcnt = 1000;  
    uint16_t count_value = 0;  
    uint32_t delta =0;  
    delay_n_10us(5);  
    RCC->OSCFCCR |= 0x02;  
    while(1)  
    {  
        delay_n_10us(1);  
        if((RCC->OSCFCSR & 0x02))  
        {  
            break;  
        }  
        if(hsi_timeoutcnt++ > g_timeoutcnt)  
        {
```

```
        break;
    }
}

count_value = RCC->OSCFCHSICNT;    //ready cnt value
if(count_value > 2048)
{
    delta = (count_value - 2048)/16;
    if((delta != 0)  && (delta <6))
    {
        tmp = RCC->CTRL & ~(0x7F << 8);    // TRIM 8-14 bit
        trim_value = ((RCC->CTRL>>8)&0x7F);
        RCC->CTRL = tmp|((trim_value-delta) << 8); // clear and set TRIM value
    }
    return delta;
}
else
{
    delta = (2048 - count_value)/16;
    if((delta != 0)  && (delta <6))
    {
        tmp = RCC->CTRL & ~(0x7F << 8);    // TRIM 8-14 bit
        trim_value = ((RCC->CTRL>>8)&0x7F);
        RCC->CTRL = tmp|((trim_value+delta) << 8); // clear and set TRIM value
    }
    return delta;
}
}
```

3.6 IIC

I2C在工作过程,SCL和SDA在通讯过程中可能受到毛刺干扰,造成通讯异常,可以使用IO软件模拟I2C。

SDK参考例程...peripheral\I2C\Software_I2C_Master

3.7 ADC

ADC参考电压是芯片内核电压，默认配置为1V，如果用户程序将内核电压配置为1.1V，ADC转换后的电压，需要做相应的调整，

如果用户采用ADC_ConverValueToVoltage函数，实际电压是转换后的电压值的1.1倍。

如果用户采用ADC16Bit_ConverValueToVoltage函数，校准参数配置调整如下：

//调整前，对应内核电压配置为1V

```
value_850mv = value_3400mv = p_trim->rc_gpadc_value_3400mv;
```

```
value_150mv = value_600mv = p_trim->rc_gpadc_value_600mv;
```

//调整后，对应内核电压配置为1.1V

```
value_850mv = value_3400mv = p_trim->rc_gpadc_value_3400mv*66.512-5301;
```

```
value_150mv = value_600mv = p_trim->rc_gpadc_value_600mv*65.875-2420;
```

3.8 NVIC

防止外设中断打断蓝牙的传输，推荐设置外设中断优先级低于蓝牙中断。蓝牙中断优先级默认配置的是0和1，用户外设的中断优先级要求配置为2或3。

中断函数中触发消息任务需使用ke_msg_send_basic(), 不能使用ke_timer_set()定时触发，避免定时嵌套异常。

Reset PIN在ACTIVE状态下是硬件复位功能，在SLEEP状态只有唤醒功能没有复位功能,须在唤醒中断中调用NVIC_SystemReset()函数来复位芯片，参考SDK void ns_EXTI4_12_IRQHandler(void)函数。

EXTI4_12、HardFault的中断处理函数，在SDK中已定义为ns_EXTI4_12_IRQHandler和ns_HardFault_Handler，请勿再重新定义。

蓝牙初始化app_ble_init()中会对中断向量初始化，参考ns_ble_stack_vtor_init(void)。

3.9 异常处理

(1) 32MHz晶体受到外部强干扰，可能会导致蓝牙工作异常（如搜索不到蓝牙广播）。可以通过软件检测机制来判断蓝牙工作状态，在蓝牙异常时进行蓝牙协议栈复位。

示例代码：

```
static rwip_time_t last_target_time = {0}; // 保存的当前蓝牙任务时间
```

```
void app_ble_target_time_check(void)
```

```
{
```

```
    rwip_time_t target_time = {0}; // 新的蓝牙任务时间，广播或连接期间蓝牙任务时间周期性更新
```

```
target_time.hs = ip_clkngt1_getf();
target_time.hus = ip_hmicrosectgt1_getf();
if ((last_target_time.hs == target_time.hs) && ((app_env.adv_state == APP_ADV_STATE_STARTED)
    || (ke_state_get(TASK_APP) == APP_CONNECTED)))
{
    last_target_time.hs = 0;
    last_target_time.hus = 0;
    ns_ble_reset(); // 蓝牙异常时执行蓝牙协议栈复位
}
else
{
    last_target_time.hs = target_time.hs;
    last_target_time.hus = target_time.hus;
}
}

void ns_ble_reset(void)
{
    /// Initialize Task state
    ke_state_set(TASK_APP, APP_INIT);
    app_env.adv_actv_idx = GAP_INVALID_ACTV_IDX;
    app_env.scan_actv_idx = GAP_INVALID_ACTV_IDX;
    app_env.init_actv_idx = GAP_INVALID_ACTV_IDX;
    app_env.conidx = GAP_INVALID_CONIDX;
    app_env.adv_mode = APP_ADV_MODE_ENABLE;
    app_env.adv_state = APP_ADV_STATE_IDLE;
    for (uint8_t i=0; i<APP_CON_IDX_MAX; i++)
    {
        app_env.conn_env[i].conidx = GAP_INVALID_CONIDX;
    }
    app_env.next_svc = 0;
    ble_prf_evn.prf_num = 0;
    prf_init(RWIP_RST);
```

```
/// Reset ble stack  
  
rwip_reset();  
  
app_send_gapm_reset_cmd();  
  
}
```

(2) 电源系统受到外部强干扰，可以通过配置芯片内核电压1.1V，提高抗干扰能力。

示例代码：

```
*(uint32_t*)0x40007014 = 0x0000080F; // 内核电压配置 1.0V, SDK默认配置
```

```
*(uint32_t*)0x40007014 = 0x00000838; // 内核电压配置 1.1V
```

内核电压配置1.1V后，ADC的参考电压也变为1.1V，可参考ADC章节做调整；HSI的频率会变快，需要对HSI重新校准，可参考HSI章节。

4 版本历史

日期	版本	修改
2021.08.05	V1.0	初始版本
2021.12.22	V1.1	添加主机例程部分描述。
2022.03.31	V1.2	章节1.5添加蓝牙血压服务和主从切换例程。 章节2.2替换FLM文件描述为安装Pack包。 章节2.3增加主机OBJ文件描述。 章节3.1增加256K Flash版本芯片描述。
2022.12.29	V1.3	章节1.1移除doc章节。 章节1.4增加例程描述： rdtss_3s, rdtsc_3m, DTM_Test和RF_Test。
2025.08.20	V1.4	章节1.4增加例程描述： beacon, lesc&aes , central_rdtsc_16bit, central_relay_1m2s 和central_hogprh； 删除例程app_usart。 章节2.2更新pack包为N32WB03x_DFP.1.4.0。 章节2.3的Setting页面更新图示。 章节3.1更新flash Bank说明和固件升级文档名称。
2026.03.05	V1.5	增加章节2.3选择蓝牙例程与章节2.4添加应用任务 把原章节3内容移到章节2.51， 章节2.52 增加章节3外设开发注意事项内容

5 声明

国民技术股份有限公司（下称“国民技术”）对此文档拥有专属产权。依据中华人民共和国的法律、条约以及世界其他法域相适用的管辖，此文档及其中描述的国民技术产品（下称“产品”）为公司所有。

国民技术在此并未授予专利权、著作权、商标权或其他任何知识产权许可。所提到或引用的第三方名称或品牌（如有）仅用作区别之目的。

国民技术保留随时变更、订正、增强、修改和改良此文档的权利，恕不另行通知。请使用人在下单购买前联系国民技术获取此文档的最新版本。

国民技术竭力提供准确可信的资讯，但即便如此，并不推定国民技术对此文档准确性和可靠性承担责任。

使用此文档信息以及生成产品时，使用者应当进行合理的设计、编程并测试其功能性和安全性，国民技术不对任何因使用此文档或本产品而产生的任何直接、间接、意外、特殊、惩罚性或衍生性损害结果承担责任。

国民技术对于产品在系统或设备中的应用效果没有任何故意或保证，如有任何应用在其发生操作不当或故障情况下，有可能致使人员伤亡、人身伤害或严重财产损失，则此类应用被视为“不安全使用”。

不安全使用包括但不限于：外科手术设备、原子能控制仪器、飞机或宇宙飞船仪器、所有类型的安全装置以及其他旨在支持或维持生命的应用。

所有不安全使用的风险应由使用人承担，同时使用人应使国民技术免于因为这类不安全使用而导致被诉、支付费用、发生损害或承担责任时的赔偿。

对于此文档和产品的任何明示、默示之保证，包括但不限于适销性、特定用途适用性和不侵权的保证责任，国民技术可在法律允许范围内进行免责。

未经明确许可，任何人不得以任何理由对此文档的全部或部分进行使用、复制、修改、抄录和传播。