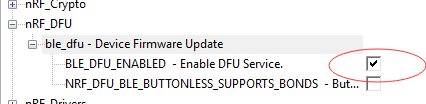
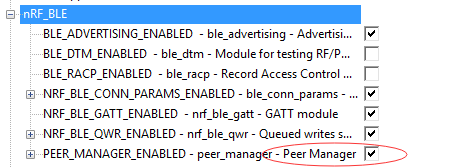
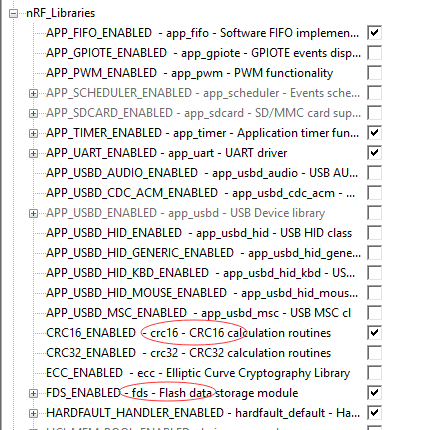
## DFU设置

### sdk\_config.h配置

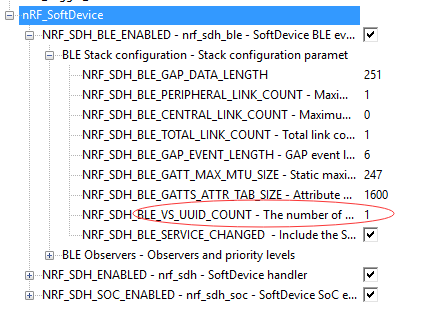
使用DFU需要使能的选项



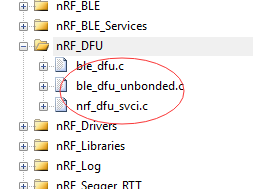


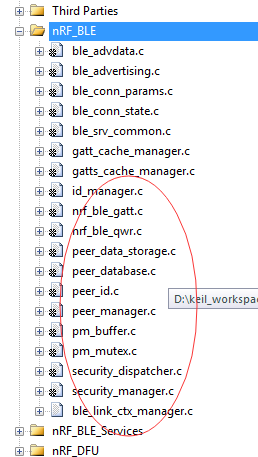


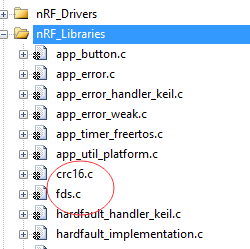
由于DFU服务采用私有UUID，需要配置协议栈，注意每增加一个VS\_UUID，应用RAM起始地址增加0x10

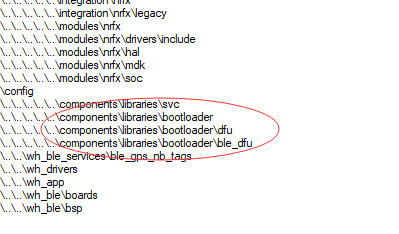


### 工程文件添加









### 工程服务代码添加

采用全局宏定义DFU\_SUPPORT作为应用层DFU功能的编译总开光

所需头文件

#if defined (DFU\_SUPPORT)

#include "nrf\_dfu\_ble\_svci\_bond\_sharing.h"

#include "nrf\_svci\_async\_function.h"

#include "nrf\_svci\_async\_handler.h"

#include "ble\_dfu.h"

#include "nrf\_power.h"

#include "nrf\_bootloader\_info.h"

#endif

应用层添加服务相关代码：

#ifdef DFU\_SUPPORT

// 进入DFU，应用关机之前的一些操作，注册到power manager电源管理中

static bool app\_shutdown\_handler(nrf\_pwr\_mgmt\_evt\_t event)

{

switch (event)

{

case NRF\_PWR\_MGMT\_EVT\_PREPARE\_DFU:

NRF\_LOG\_INFO("Power management wants to reset to DFU mode.");

// YOUR\_JOB: Get ready to reset into DFU mode

break;

default:

// YOUR\_JOB: Implement any of the other events available from the power management module:

// -NRF\_PWR\_MGMT\_EVT\_PREPARE\_SYSOFF

// -NRF\_PWR\_MGMT\_EVT\_PREPARE\_WAKEUP

// -NRF\_PWR\_MGMT\_EVT\_PREPARE\_RESET

return true;

}

NRF\_LOG\_INFO("Power management allowed to reset to DFU mode.");

return true;

}

// 注册应用关机事件处理函数（常用于关机前需要进行的一些操作：如flash操作，控制模块的关闭等）

NRF\_PWR\_MGMT\_HANDLER\_REGISTER(app\_shutdown\_handler, 0);

static void buttonless\_dfu\_sdh\_state\_observer(nrf\_sdh\_state\_evt\_t state, void \* p\_context)

{

if (state == NRF\_SDH\_EVT\_STATE\_DISABLED)

{

NRF\_LOG\_INFO("NRF\_SDH\_EVT\_STATE\_DISABLED to DFU mode.");

// Softdevice was disabled before going into reset. Inform bootloader to skip CRC on next boot.

nrf\_power\_gpregret2\_set(BOOTLOADER\_DFU\_SKIP\_CRC);

//Go to system off.

nrf\_pwr\_mgmt\_shutdown(NRF\_PWR\_MGMT\_SHUTDOWN\_GOTO\_SYSOFF);

}

}

/\* nrf\_sdh state observer. \*/

NRF\_SDH\_STATE\_OBSERVER(m\_buttonless\_dfu\_state\_obs, 0) =

{

.handler = buttonless\_dfu\_sdh\_state\_observer,

};

#endif

#ifdef DFU\_SUPPORT

/\*\*@brief Function for handling dfu events from the Buttonless Secure DFU service

\*

\* @param[in] event Event from the Buttonless Secure DFU service.

\*/

static void ble\_dfu\_evt\_handler(ble\_dfu\_buttonless\_evt\_type\_t event)

{

switch (event)

{

case BLE\_DFU\_EVT\_BOOTLOADER\_ENTER\_PREPARE:

NRF\_LOG\_INFO("Device is preparing to enter bootloader mode.");

// YOUR\_JOB: Disconnect all bonded devices that currently are connected.

// This is required to receive a service changed indication

// on bootup after a successful (or aborted) Device Firmware Update.

break;

case BLE\_DFU\_EVT\_BOOTLOADER\_ENTER:

// YOUR\_JOB: Write app-specific unwritten data to FLASH, control finalization of this

// by delaying reset by reporting false in app\_shutdown\_handler

NRF\_LOG\_INFO("Device will enter bootloader mode.");

break;

case BLE\_DFU\_EVT\_BOOTLOADER\_ENTER\_FAILED:

NRF\_LOG\_ERROR("Request to enter bootloader mode failed asynchroneously.");

// YOUR\_JOB: Take corrective measures to resolve the issue

// like calling APP\_ERROR\_CHECK to reset the device.

break;

case BLE\_DFU\_EVT\_RESPONSE\_SEND\_ERROR:

NRF\_LOG\_ERROR("Request to send a response to client failed.");

// YOUR\_JOB: Take corrective measures to resolve the issue

// like calling APP\_ERROR\_CHECK to reset the device.

APP\_ERROR\_CHECK(false);

break;

default:

NRF\_LOG\_ERROR("Unknown event from ble\_dfu\_buttonless.");

break;

}

}

#endif

/\*\*@brief Function for initializing services that will be used by the application.

\*

\* @details Initialize the Heart Rate, Battery and Device Information services.

\*/

static void services\_init(void)

{

ret\_code\_t err\_code;

nrf\_ble\_qwr\_init\_t qwr\_init = {0};

// Initialize Queued Write Module.

qwr\_init.error\_handler = nrf\_qwr\_error\_handler;

err\_code = nrf\_ble\_qwr\_init(&m\_qwr, &qwr\_init);

APP\_ERROR\_CHECK(err\_code);

// 添加GPS\_NB\_TAG服务

ble\_gnts\_init\_t gnts\_init;

memset(&gnts\_init, 0, sizeof(gnts\_init));

gnts\_init.data\_handler = gnts\_data\_handler;

err\_code = ble\_gnts\_init(&m\_gnts, &gnts\_init);

APP\_ERROR\_CHECK(err\_code);

#ifdef DFU\_SUPPORT

ble\_dfu\_buttonless\_init\_t dfus\_init = {0};

// Initialize the async SVCI interface to bootloader.

err\_code = ble\_dfu\_buttonless\_async\_svci\_init();

APP\_ERROR\_CHECK(err\_code);

dfus\_init.evt\_handler = ble\_dfu\_evt\_handler;

err\_code = ble\_dfu\_buttonless\_init(&dfus\_init);

APP\_ERROR\_CHECK(err\_code);

#endif

}

## 调试小笔记

1.nRF52832 — NFC脚（P9/P10）配置为普通GPIO口

需添加宏定义 CONFIG\_NFCT\_PINS\_AS\_GPIOS （可在system\_nrf52.h文件中#define CONFIG\_NFCT\_PINS\_AS\_GPIOS

或全局宏定义添加CONFIG\_NFCT\_PINS\_AS\_GPIOS）

2.使用NRF\_DRV驱动之前，需要在nrf\_drv\_config.h中进行配置，同时有些参数也需要设置

如：使用nrf\_drv\_saadc驱动，需要#define SAADC\_ENABLED 0改为#define SAADC\_ENABLED 1

同时驱动层同时添加nrf\_drv\_saadc.c（驱动）和nrf\_saadc.c（频道配置）

3.按键中断标志位主要PUSH和RELEASE分别设置，否则会发生按键触发两次事件；

4.SDK15的FreeRTOS例程功耗无法下降的问题？

原因：空闲任务没有调用协议栈低功耗管理，采用协议栈时最好不要直接采用操作寄存器方式；

解决：空闲任务休眠函数修改如下：

#endif // (NRFFOSDK-11174)

{

/\* No SD - we would just block interrupts globally.

\* BASEPRI cannot be used for that because it would prevent WFE from wake up.

\*/

do{

// \_\_SEV(); //发送事件

// \_\_WFE(); //休眠并且在发生事件时被唤醒

sd\_app\_evt\_wait();

} while (0 == (NVIC->ISPR[0] | NVIC->ISPR[1])); //等待中断

}

5.串口功耗问题？

原因：由于SDK15默认采用了UARTE使用了DMA功能，使得功耗为2.2mA左右。

解决：取消DMA，功耗下降至0.6mA

文件nrf52832\_peripherals.h,注释掉222行的//#define UARTE\_PRESENT 同时sdk\_config.h配置#define NRFX\_UARTE\_ENABLED 0

6.按键初次不触发问题？

原因：按键使能状态没有设置，针对不同按按键需要不同设置。当外部有分压/下拉时，不使用上拉下拉，以免影响功能

解决：{BSP\_BUTTON\_0, APP\_BUTTON\_ACTIVE\_LOW, BUTTON\_PULL, bsp\_button\_event\_handler},

{BSP\_BUTTON\_1, APP\_BUTTON\_ACTIVE\_HIGH, NRF\_GPIO\_PIN\_PULLDOWN, bsp\_button\_event\_handler},

{BSP\_BUTTON\_3, APP\_BUTTON\_ACTIVE\_HIGH, NRF\_GPIO\_PIN\_NOPULL, bsp\_button\_event\_handler},