FB401 SDK 使用手册

目录

一、版本历史

日期	描述
2019 年 1 月 1 日	初版

二、关于本文档

这个文档基于此 SDK 的模块组织章节结构。

每个模块的信息分为如下几个方面：

- 这个模块的功能介绍
- 模块的 API
- 常见的使用方式和设计模式

（一）命名规范

每个模块具有一个全部大写的独一无二的名字,例如 GPIO。模块的 API 参考如下风格：MODULE_FunctionName，例如 GPIO_SetPinMode。许多模块会产生事件，处理事件的函数叫做 MODULE_EventHandler，例如 BLE_EventHandler。

三、系统架构

整个系统分为三层，其中

硬件抽象层包含有对 SoC 的功能封装，例如 GPIO。
应用框架层构建了一个基于 BLE 的应用平台，内部封装了系统的主事件循环及底层事件的处理。
应用层是由用户实现的，通过响应上层事件和调用 API，定义应用的业务逻辑。

（一）对比无 RTOS 的 MCU 典型软件架构

一个典型的不基于 RTOS 的 MCU 软件，由 main 和 IRQ 函数组成，其中 main 负责系统的初始化，并实现一个主循环；IRQ 负责响应硬件事件。
在 FB401 平台上，main()隐藏于应用框架层内部，对用户不可见，但 FB401 会提供高层事件供用户实现系统初始化与运行过程中的业务逻辑。底层的 IRQ 由硬件抽象层管理，但高层事件会提交给应用层处理。
基于 FB401 开发，用户无需关心最底层的细节，仅需考虑上层业务逻辑。

（二）对比有 RTOS 的 MCU 典型软件架构

通常基于 RTOS 的软件会有多个任务，各个任务之间通过线程间通信相互协作。FB401 的核心是一个事件主循环，应用层只考虑对各种事件如何响应，因此不存在任务概念。

四、SYS 模块

SYS 是对整个系统的抽象，它主要用于两个目的：

- 通过 SYS 事件响应整个系统的状态变化
- 通过 API 查看和改变系统状态

（一）事件

```
void SYS_EventHandler(sys_event_t event, void *params);
```

1．SYS_EVENT_READY

系统就绪。仅当启动完成，系统进入就绪状态时发生。此时可以开始应用层初始化动作。

（二）接口

1．SYS_Reset

```
void SYS_Reset(void);
```

调用此函数后，系统将重启。

（三）例子

1．应用初始化

```
void SYS_EventHandler(sys_event_t event, void *params)
{
    switch (event)
    {
    case SYS_EVENT_READY:
        printf("Application Started\n");
        // 应用初始化代码从此开始

        // 应用初始化代码到此结束
```

```
            break;
        }
}
```

2．重启系统

```
SYS_Reset();
```

五、BLE 模块

BLE 模块封装了蓝牙通信功能（含 Mesh 组网），此模块屏蔽了蓝牙的复杂细节，用户只需要关注对数据的处理。

（一）事件

```
void BLE_EventHandler(ble_event_t event, void *params);
```

1．BLE_EVENT_DATA_RX

收到数据。当蓝牙模块接收到来自 Mesh 网络或者上位机的新的数据时产生。

事件参数：ble_data_rx_event_params_t

项目	类型	含义
data	uint8_t*	数据指针
len	uint32_t	数据长度

（二）接口

六、GPIO 模块

GPIO 封装了通用输入输出功能，具体包括：

- 引脚初始化
- 输出高低电平
- 读取电平

（一）事件

（二）接口

1．GPIO_SetPinMode

```
void GPIO_SetPinMode(uint8_t port, uint8_t pin,
                     gpio_pin_mode_t mode)
```

初始化引脚并配置引脚的工作模式

函数参数：

参数	含义	取值
port	端口号	GPIO_PA GPIO_PB GPIO_PC GPIO_PD
pin	引脚编号	GPIO_PIN_0 GPIO_PIN_1 … GPIO_PIN_7
mode	模式代码	GPIO_PIN_MODE_NONE 不使用（高阻态） GPIO_PIN_MODE_INPUT 输入 GPIO_PIN_MODE_INPUT_PULL_UP 输入上拉 GPIO_PIN_MODE_OUTPUT 输出

此 API 内部会设置引脚复用为 GPIO 模式。若引脚之前被用作其他 IO（例如 UART/I2C 等），则之前的功能会被断开。

注意：若将调试串口，JLink 烧录口配置成 GPIO，则调试或者烧录的功能会失去。

2．GPIO_WritePin

```
void GPIO_WritePin(uint8_t port, uint8_t pin, uint8_t value);
```

向输出模式的引脚写值。引脚之前应被配置成 GPIO_PIN_MODE_OUTPUT，否则不会得到期望的效果。

函数参数：

参数	含义	取值
port	端口号	GPIO_PA GPIO_PB GPIO_PC GPIO_PD
pin	引脚编号	GPIO_PIN_0 GPIO_PIN_1 … GPIO_PIN_7
value	值	1：高电平 0：低电平

3．GPIO_ReadPin

```
uint8_t GPIO_ReadPin(uint8_t port, uint8_t pin);
```

读取引脚当前值。

若引脚是输入模式，则可读取外部输出的电平。

若引脚是输出模式，则会读取当前实际的输出电平。

若引脚为其他模式，则结果没有确定含义。

函数参数：

参数	含义	取值
port	端口号	GPIO_PA GPIO_PB GPIO_PC GPIO_PD
pin	引脚编号	GPIO_PIN_0 GPIO_PIN_1 … GPIO_PIN_7
返回	值	1：高电平 0：低电平