

## 8 蓝牙协议栈

到目前为止，大家可能感觉 nRF51822 只是一个不错的 Cortex-M0 单片机和 2.4Ghz 收发器的集成芯片而已。但是，它的杀手锏应用是蓝牙 4.0（BLE）。根据 NORDIC 的财报数据显示，该公司 2013 年二季度蓝牙芯片的销售比例从 2012 年的 1% 增长到 12%，财报中同时承认这是因为 BLE 强劲的销售导致的。原文为：offset by higher sales of Bluetooth Smart (low energy) solutions

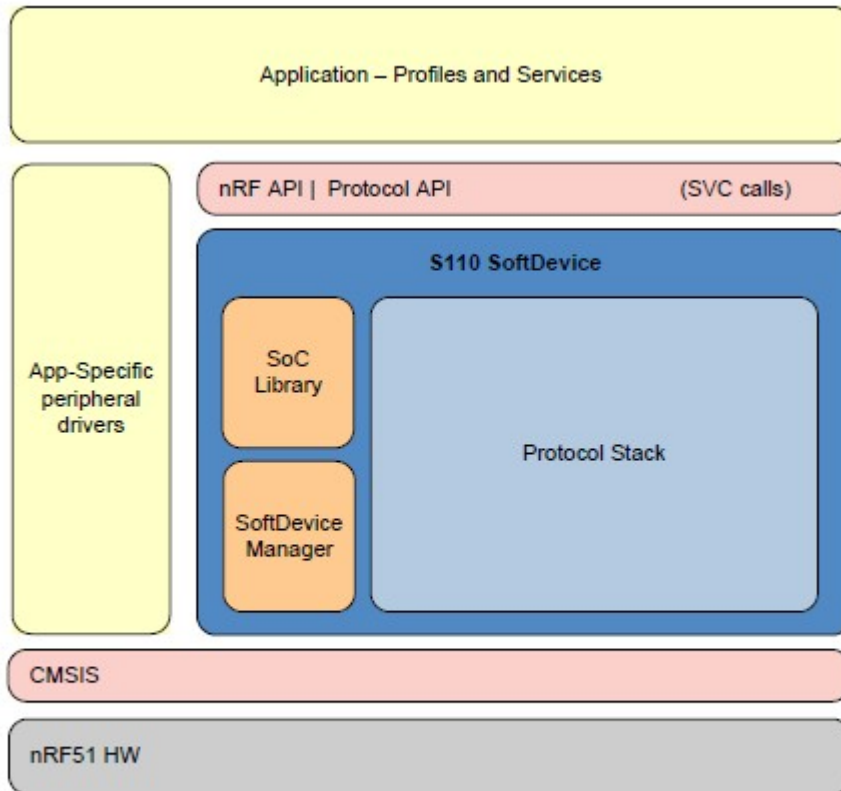
如前言所诉，BLE 是蓝牙 4.0 的一个低功耗子集，再加上 NORDIC 出众的功耗控制，可以让一个纽扣电池（coin-cell batteries）供电的 Bluetooth Smart 设备工作几个月甚至几年。

传感器、移动智能设备、大数据与云计算，这种三位一体的发展模式已经越来越清晰。对于用户而言移动智能设备是一切连接的中心，对于穿戴设备开发者来说能够与移动智能设备无缝连接的传感器，就能够通过这个桥梁顺利的与云端连接。

### 8.1 S110 SoftDevice

S110\_SoftDevice 是一个 BLE 协议栈，集成了低功耗控制器和 host，并且提供了全功能的 API 供系统调用。NORDIC 提供了预编译好的 HEX 文件，该文件和应用程序是分开下载的。从应用程序开发的角度看，使用 SoftDevice 蓝牙协议栈的应用程序和普通的 ARM® Cortex™-M0 工程没有区别。这也意味着兼容 ARM® Cortex™-M0 的编译工具都可以开发 Bluetooth low energy 应用程序。

#### 8.1.1 系统框图



**Figure 1** System on Chip application with the SoftDevice

注意：S110 SoftDevice 支持多协议非同时共存。

### 8.1.2 S110 SoftDevice 支持的 Profile

为了更容易的保持 Bluetooth 设备之间的兼容，Bluetooth 规范中定义了 Profile。Profile 定义了设备如何实现某种特定的应用（或者连接），比如心率计。Bluetooth 的一个很重要特性，就是所有的 Bluetooth 产品都无须实现全部的 Bluetooth 规范，你可根据所需要的产品实现需要的 Profile，不必给开发带来更大的开销。

目前发布的 S110 SoftDevice 支持如下的 Profile：

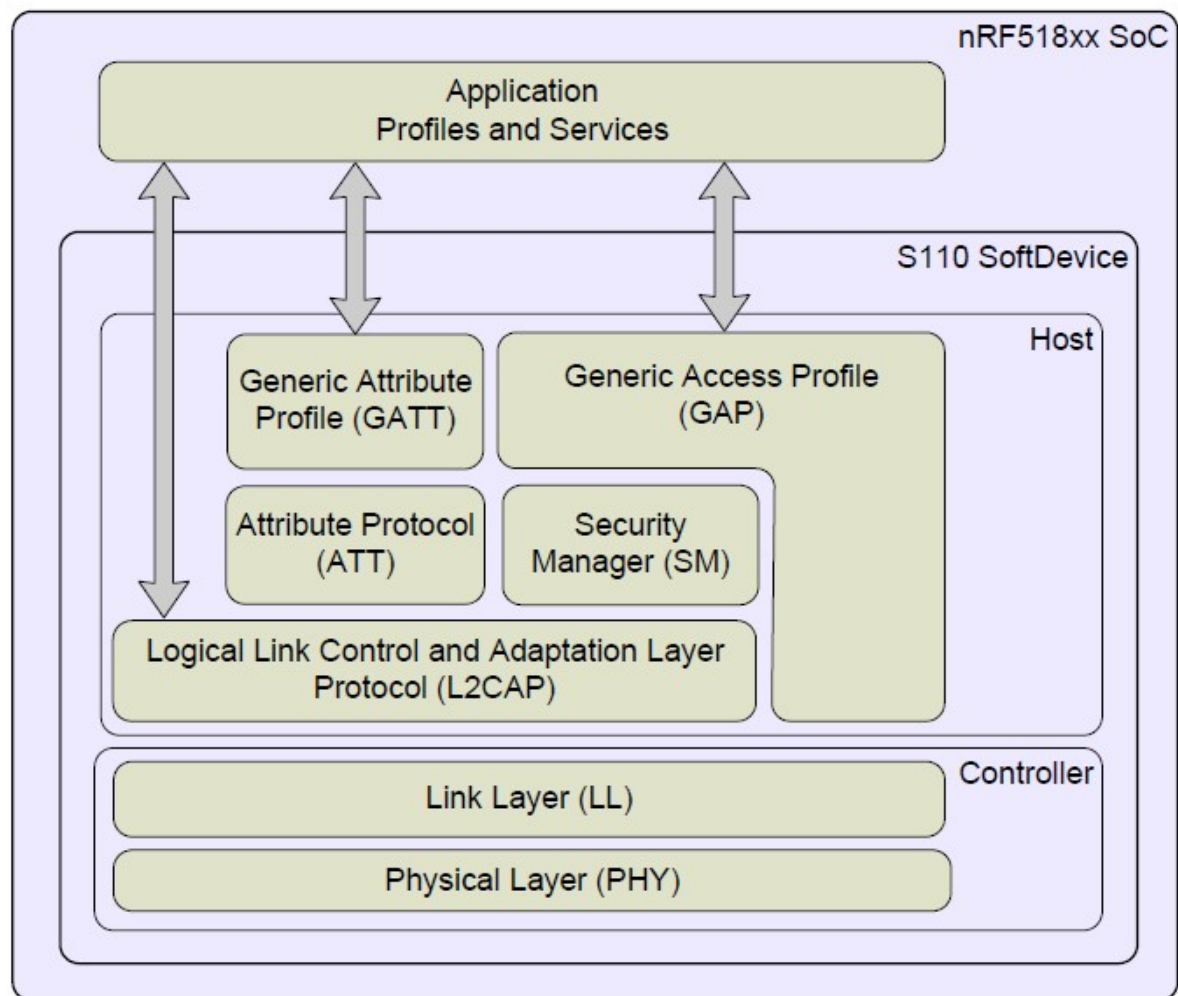
Adopted Profile	Adopted Services	Supported
HID over GATT	HID Battery Device Information	YES
Heart Rate Monitor	Heart Rate Device Information	YES
Proximity	Link Loss Immediate Alert TX Power	YES
Blood Pressure	Blood pressure	YES
Health Thermometer	Health Thermometer	YES
Glucose	Glucose	YES
Phone Alert Status	Phone Alert Status	YES
Alert Notification	Alert Notification	YES
Time	Current Time Next DST Change Reference Time Update	YES
Find Me	Immediate Alert	YES
Cycling speed and cadence	Cycling speed and cadence Device information	YES
Running speed and cadence	Running speed and cadence Device information	YES

*Table 1 Adopted Profile and Service support*

### 8.1.3 S110 SoftDevice 的 API

S110 SoftDevice 内建了 Bluetooth 4.0 compliant low energy (BLE) Host 和 Controller。  
API 建立在 Generic Attribute Protocol (GATT), Generic Access Profile (GAP), and Logical

Link Control and Adaptation Protocol (L2CAP)之上。



**Figure 2** LE stack architecture

## 8.2 使用例程

本节描述了如何在 SoftDevice（蓝牙协议栈）基础上，编写一个接近器（proximity）。它会通过 BLE（Bluetooth low energy）4.0 发送信号，应用程序可以通过判断信号强度来确定位置。注意：该例程使用 AK board 或者 USB dongle 均可

### 8.2.1 下载 SoftDevice 固件

下载 S110 nRF51822 SoftDevice

Follow these steps to program your device:

1. 打开 nRFgo Studio.

2. 在 Device Manager 中选择 nRF51 Programming



3. 选择 Program SoftDevice 标签栏.



4. 点击 Browse 并找到需要下载的 SoftDevice 文件（在 s110\_nrf51822\_xxxxxx.zip 内）.
5. 点击 Program.
6. 至此蓝牙协议栈下载完毕，下面演示下载上层应用程序。

*注意：当下载非蓝牙程序时，如简单的“跑马灯”等，需要使用 nRFgo Studio 将芯片上的蓝牙协议栈擦除，才能正常下载。*

## 8.2.2 编译, 下载, 并运行 ble\_app\_proximity 演示程序

1. 找到 ble\_app\_proximity 工程，路径为  
ARM\Device\Nordic\nrf51822\board\PCA10001\ble\ble\_app\_proximity\arm
2. 双击 ble\_app\_proximity.uvproj 文件，打开 Keil μVision
3. 单击 Build 图标，或者按 F7 编译工程.

4. 打开 **Flash** 菜单，并单击 **Download** 下载程序(或者点击 **Load** 图标).
5. **LED 0** 将会闪烁，表示它正在广播。

Peripheral: **Nordic\_Prox**

Clone Peripheral

## Services Found

<b>0x1800</b> UUID: 1800	>
<b>0x1801</b> UUID: 1801	>
<b>Tx Power</b> UUID: 1804	>
<b>Immediate Alert</b> UUID: 1802	>
<b>Link Loss</b> UUID: 1803	>
<b>Battery Service</b> UUID: 180F	>

ble\_app\_proximity

## 8.3 代码分析

### 8.3.1 打开 ble\_app\_proximity 例程

(安装 SDK 后, 源代码位于<keil  
path>\ARM\Device\Nordic\RF51822\Board\PCA10001\ble\ble\_app\_proximity\)

### 8.3.2 main()函数

```
/**@brief Application main function.
 */
int main(void)
{
    // Initialize
    leds_init();
    timers_init();
    gpiote_init();
    buttons_init();

    bond_manager_init();
    ble_stack_init();
    gap_params_init();
    advertising_init(BLE_GAP_ADV_FLAGS_LE_ONLY_LIMITED_DISC_MODE);
    services_init();
    conn_params_init();
    sec_params_init();
    radio_notification_init();

    // Start execution
    advertising_start();

    // Enter main loop
    for (;;)
    {
        power_manage();
    }
}
```

### 8.3.3 BLE 通讯的基本流程

上面的 main()代码中, 前面 4 行是板载硬件初始化:

```
leds_init();    // LED
timers_init();  // 定时器
gpiote_init();  // GPIO task&event
buttons_init(); // 按键
```

之后进入标准的蓝牙初始化:

```
bond_manager_init(); // Bond 初始化
ble_stack_init();    // 协议栈初始化
gap_params_init();   // Generic Access Profile
```



```

advertising_init(BLE_GAP_ADV_FLAGS_LE_ONLY_LIMITED_DISC_MODE);
// 广播地址
services_init(); // 需要开启的服务初始化
conn_params_init(); // 连接参数设置
sec_params_init(); // 参数设置
radio_notification_init(); // radio 通知初始化

```

初始化完成后，就是真正执行阶段：

```

advertising_start(); // 开广播

// Enter main loop
for (;;)
{
    power_manage();
}

```

开始广播之后，AK board 处于等待连接的状态。手机端 App 主动连接，并且绑定后，可以与 AK board 通讯。具体过程可以参考蓝牙连接的其他讲解，步骤是完全相同的。