life.augmented

STM32WB Offline Training 2021_Technical

主题

- STM32WB 硬件介绍
- STM32WB 软件介绍
- BLE 基础
- BLE低功耗设计
- BLE速率优化
- OTA应用介绍

life.augmented

STM32WB硬件简介

1 系统和存储

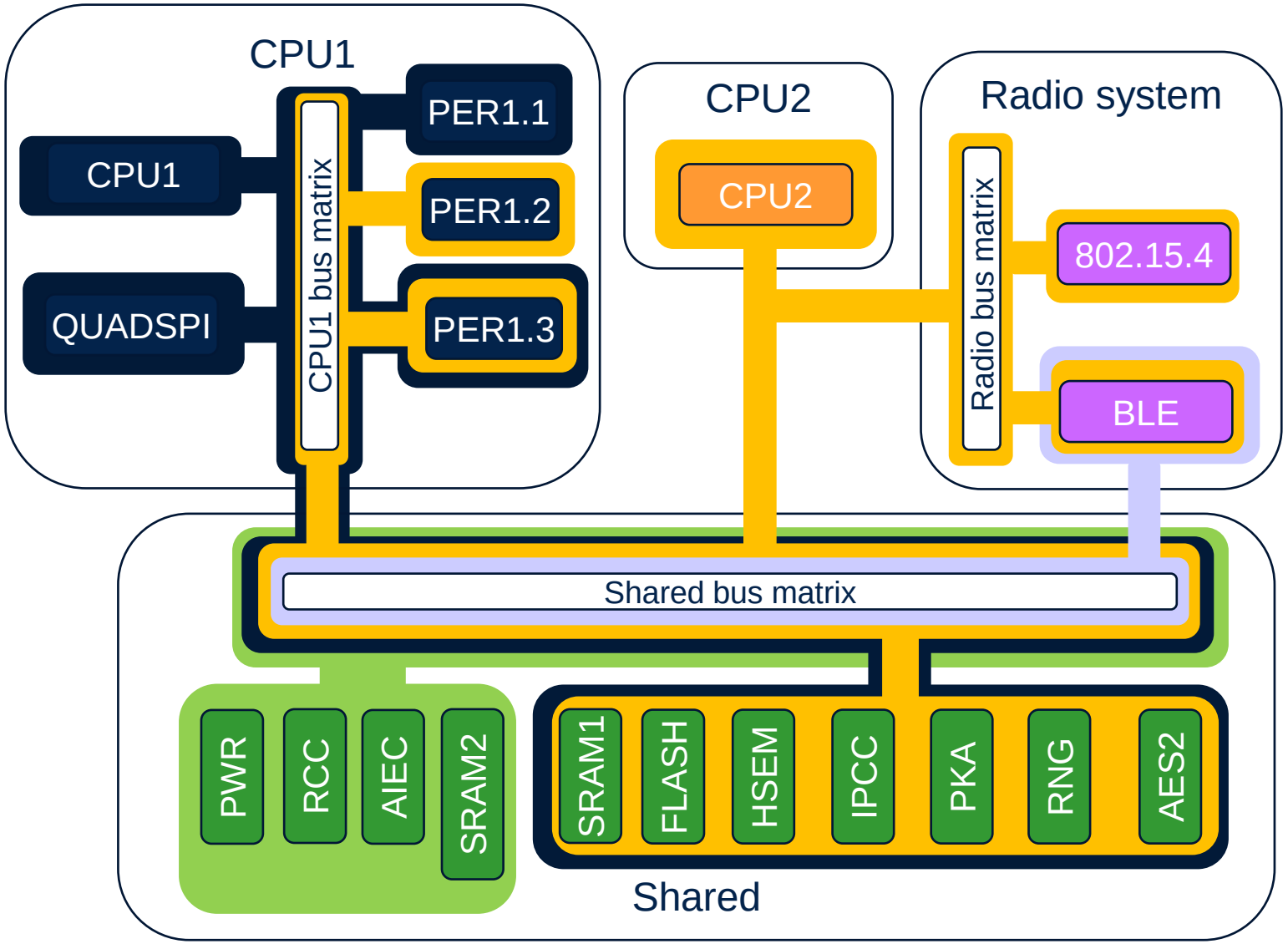
2 RF射频简介

3 电源管理

life.augmented

1.系统和存储

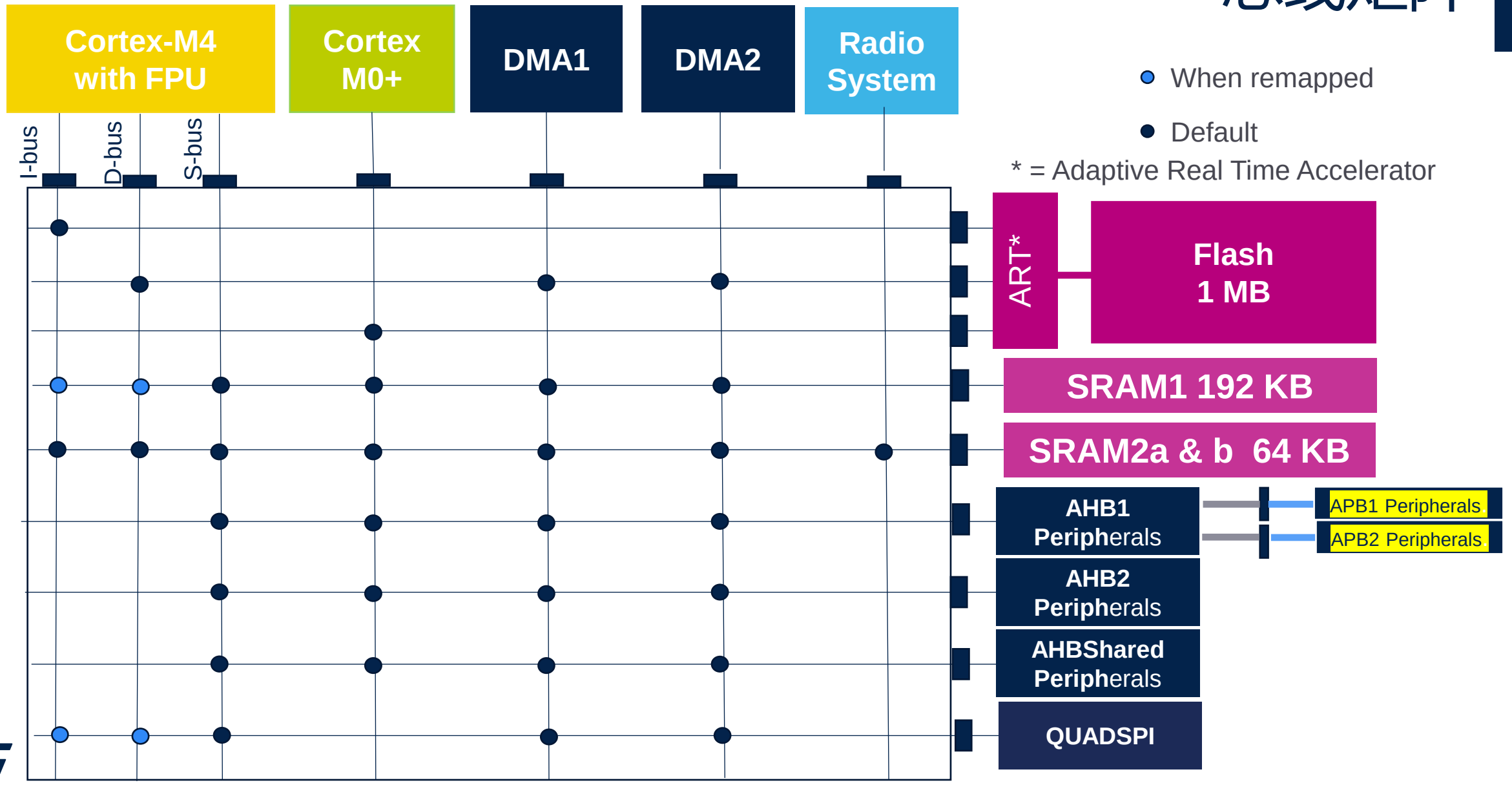
系统总览



- 3 个子系统
  - CPU1 Corex-M4 (深蓝色)
  - CPU2 Cortex-M0+ (橙色)
  - Radio system (紫色)
- 始终有时钟(Green)
- 独立的操作模式
  - CRun
  - CSleep
  - CStop
- 通过RCC中的使能位分配CPU外设

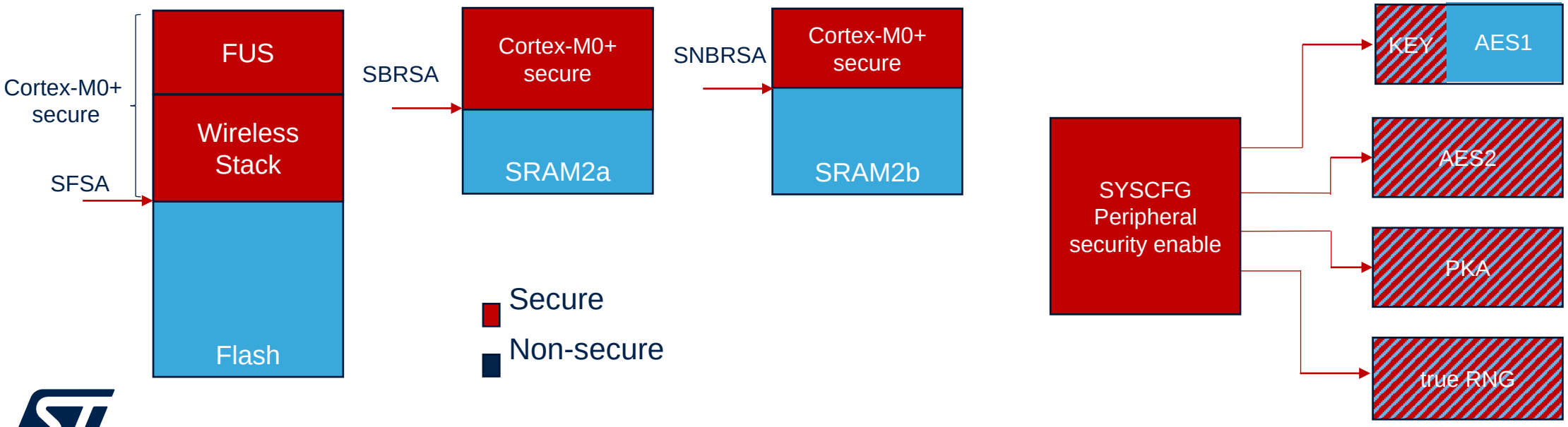
life.augmented

总线矩阵


Cortex-M4 with FPU
Cortex M0+
DMA1
DMA2
Radio System
I-bus
D-bus
S-bus
When remapped
Default
* = Adaptive Real Time Accelerator
ART*
Flash 1 MB
SRAM1 192 KB
SRAM2a & b 64 KB
AHB1 Peripherals
AHB2 Peripherals
AHBShared Peripherals
QUADSPI
APB1 Peripherals
APB2 Peripherals

Cortex-M0+ 安全性

- 闪存的上半部分只能由Cortex-M0+访问.
  - 由安全选项字SFD和SFSA定义.
- 全局安全使能.
  - 允许通过安全选项字SBRD和SBRSA来添加SRAM2a上部分安全性
  - 允许通过安全选项字SNBRD和SNBRSA来添加SRAM2b上部分安全性
  - 允许通过SYSCFG使能外设的安全性.



life.augmented

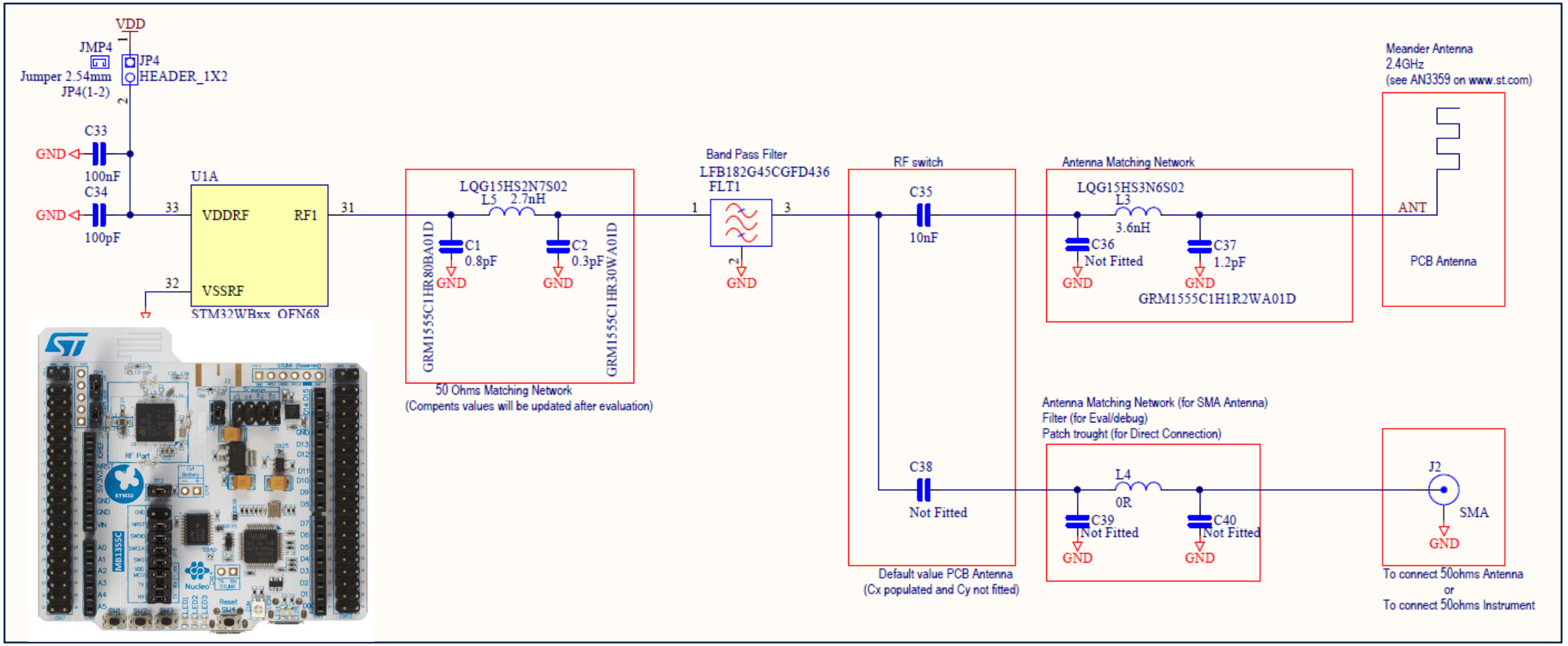
2.RF射频简介

STM32WB射频规范

- 模拟前端:
  - 最大输出功率:
    - 集成巴伦，6 dBm发射功率，具有1 dBm步进调节
    - 专用引脚驱动外部PA，可获得更好的输出功率。
  - 接收灵敏度:
    - BLE: -96 dBm @ 1Mbps, -94 dBm @ 2Mbps
    - 802.15.4: -100 dBm @ 250Kbps
  - 功耗@ 3.3V:
    - TX @ 0 dBm: 5.2mA
    - RX: 4.5mA
    - Stop2 with radio in standby (accurate clock LSI2): 1.8µA
- 调制解调器:
  - 通过硬件格式化BLE包 (对比软件完全开放射频) 支持 1和2 Mbps速率
  - IEEE 802.15.4: 硬件模式支持250Kbps速率

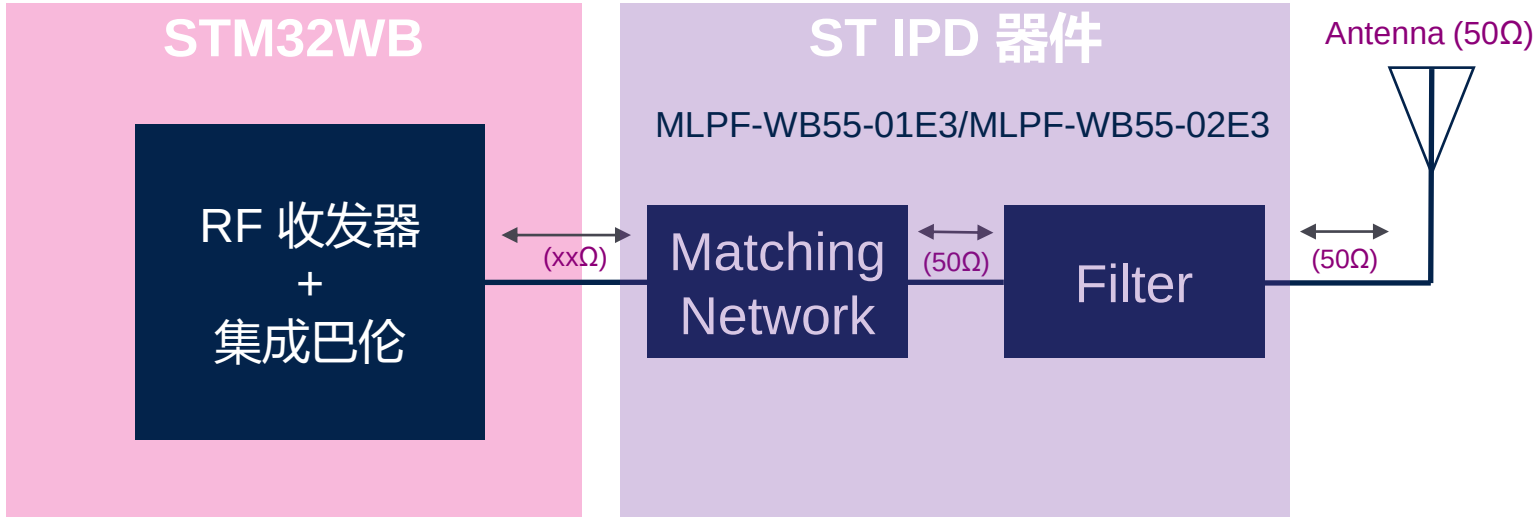
STM32WB RF输入/输出匹配网络

AN 5165



IPD器件

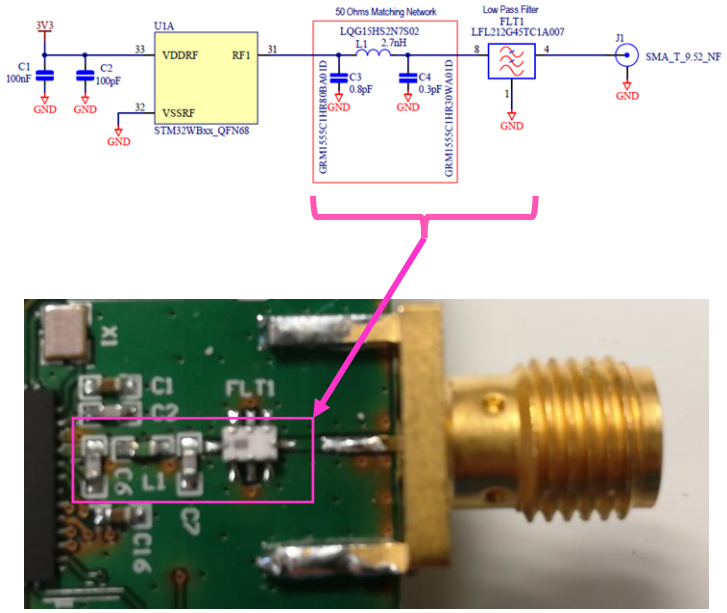
- **两部分**
  - 匹配网络– 转换为50 Ω阻抗
  - 谐波滤波器– 减少带外TX谐波发射，提高RX灵敏度



*IPD: Integrated Passive Device，集成被动器件

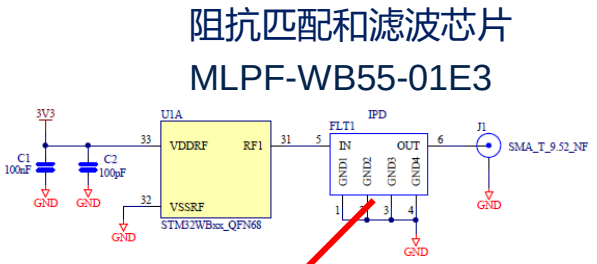
IPD器件

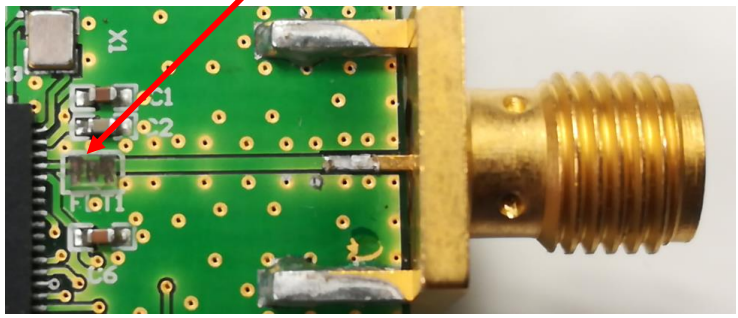
目标：替换分立的匹配网络和滤波器，并达到相同的TX/RX性能。



阻抗匹配和滤波芯片

MLPF-WB55-01E3





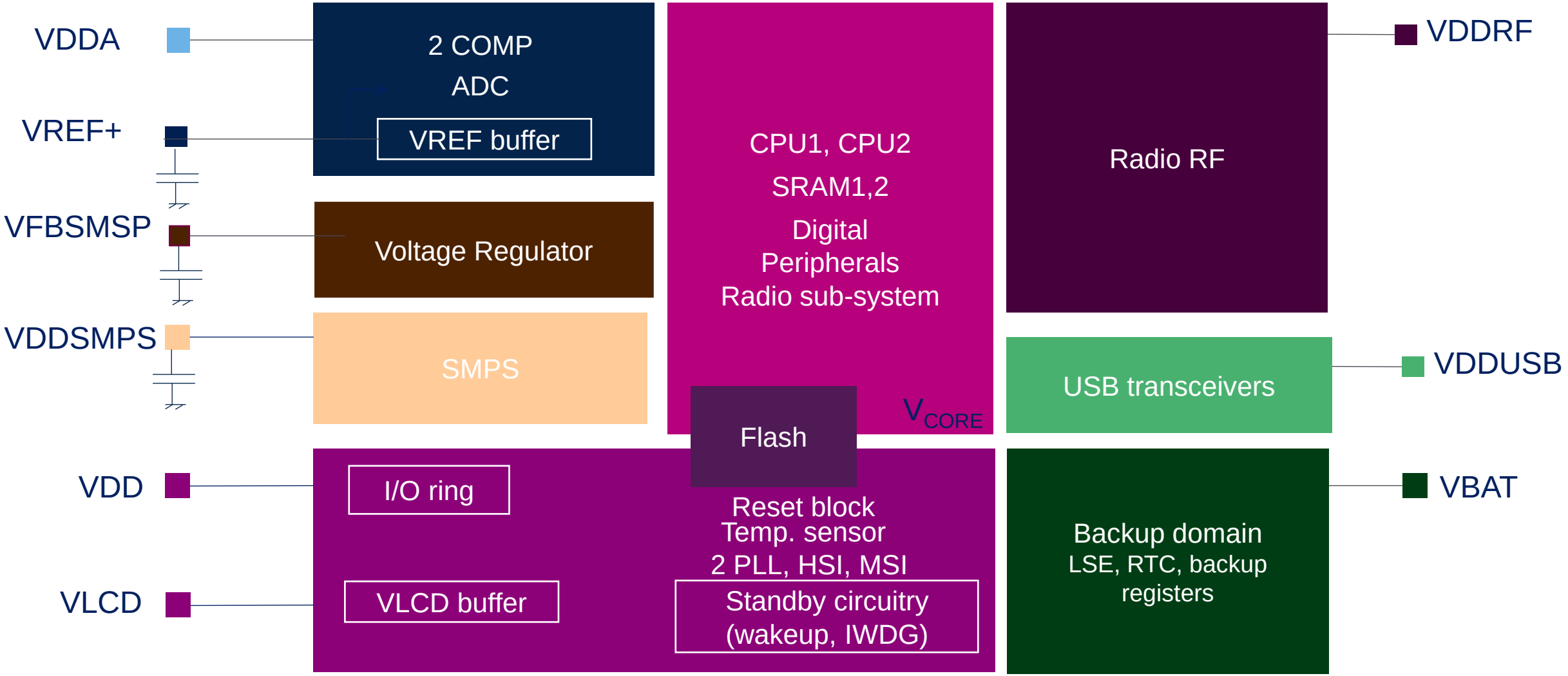
AN 5165

life.augmented

3.电源管理

life.augmented

供电方案



life.augmented

内部SMPS

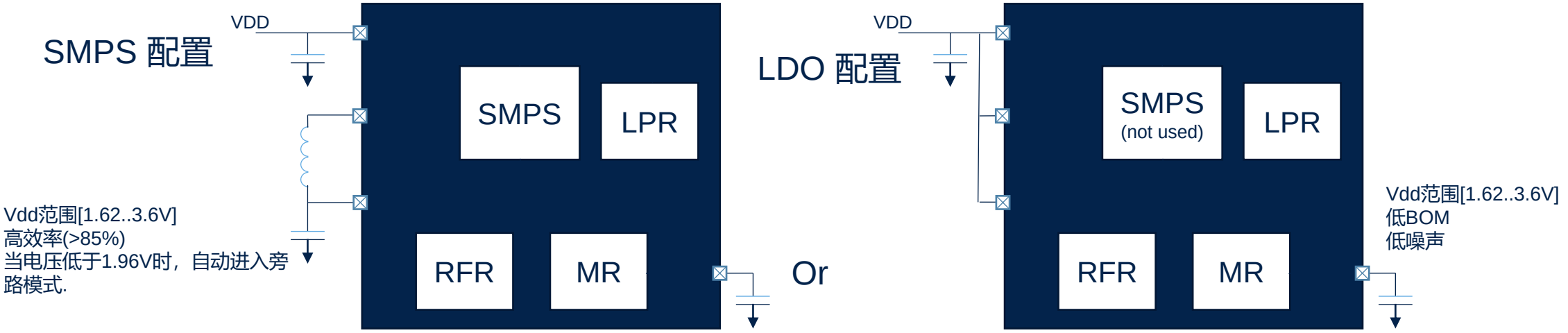
- **提高功耗效率**

- SMPS用于降低VDD电源.
- SMPS 为数字核心和射频LDO供电.
- 当VDD电源高于BOR [1..4]阈值时，使用SMPS模式.
  - 低于此阈值使用旁路模式.支持即时切换.
  - 通过HW机制执行关闭, 通过SW重新开启.
- SMPS将遵循设备操作模式
  - 在Run和Stop0模式下为ON.
  - 在Stop1,2，待机Standby和关机Shutdown模式下，SMPS自动处于开放模式. 唤醒时会自动恢复进入前使用的模式.

电源配置

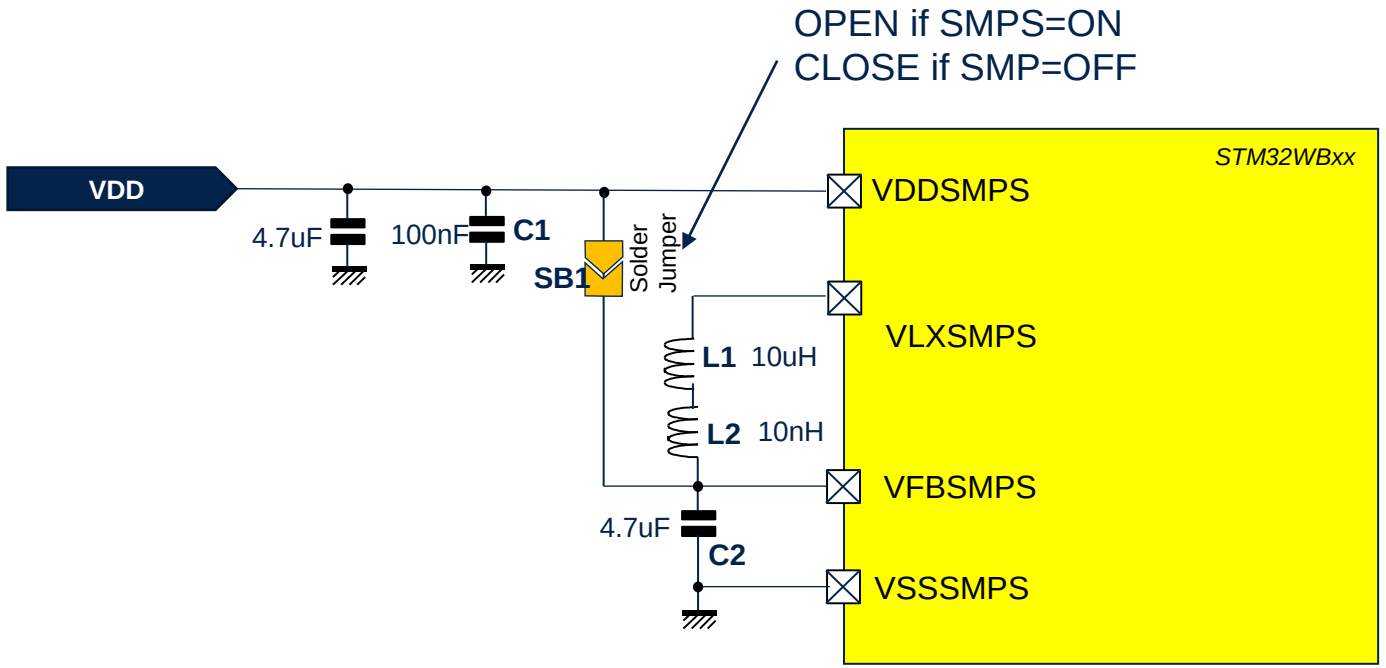
- **灵活选择性能和成本**

- 高性能使用SMPS
  - 通过添加外部电容和电感，降低功耗.
- 低成本仅适用LDO.
  - 通过短接SMPS输入，LDO直接由VDD提供.节省电容和电感的成本，但以增加总功耗为代价.



life.augmented

SMPS 原理图



- 为了获得最佳的功率性能，应选择4MHz，由一个10μH的电感，10nH的电感，4.7μF的电容组成。
- 为了获得更小的封装，可以选择8MHz，由一个2.2μH的电感，10nH的电感，4.7μF的电容组成。

STM32WB软件简介

1 WB软件包

2 调度器

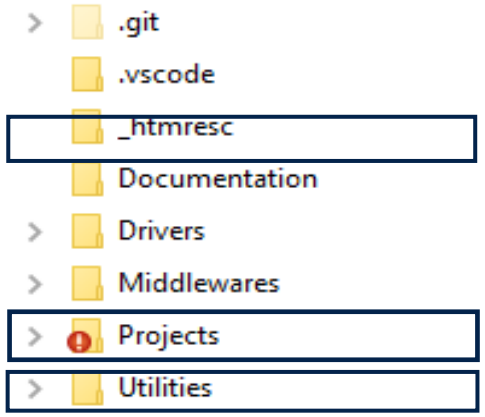
3 低功耗管理

4 定时器管理

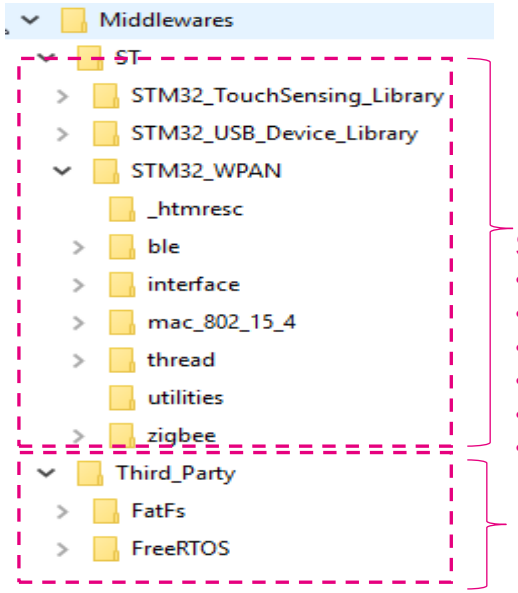
life.augmented

1.软件包

软件包



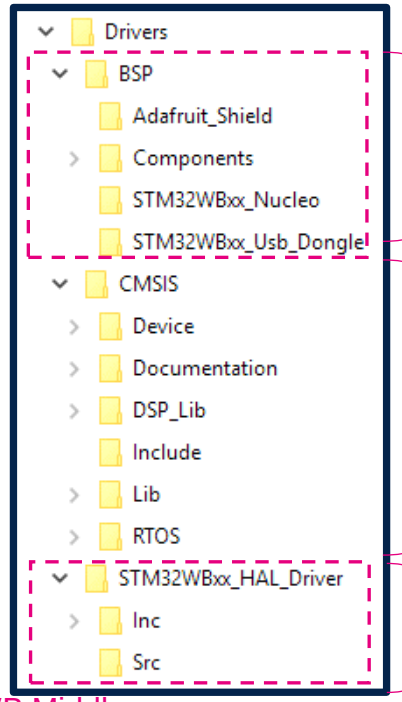
Middleware



STM32WB Middlewares
- USB
- BLE
- Thread
- Zigbee
- TouchSensing
- Application Utilities

Third party
- FreeRTOS
- FatFs

Driver

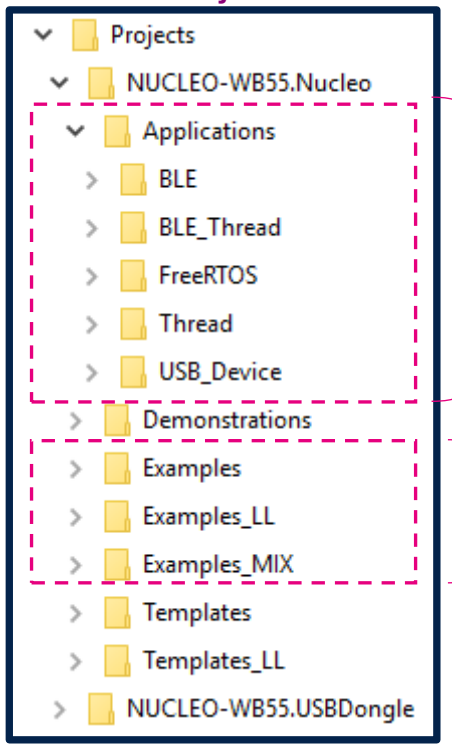


BSP

CMSIS

HAL Driver

Projects



STM32WB Application
- USB
- BLE
- Thread
- Zigbee

Peripheral Examples

只针对STM32WB M4

下载链接

https://www.st.com/content/st_com/en/products/embedded-software/mcu-mpu-embedded-software/stm32-embedded-software/stm32cube-mcu-mpu-packages/stm32cubewb.html

life.augmented

协议栈目录

- 在CPU2上运行的所有代码都以加密二进制的形式发布。

STM32Cube › Repository › STM32Cube_FW_WB_V1.11.0 › Projects › STM32WB_Copro_Wireless_Binaries

STM32WB1x

Name	Size
Release_Notes.html	9 KB
stm32wb1x_BLE_HCI_AdvScan_fw.bin	39 KB
stm32wb1x_BLE_HCILayer_fw.bin	77 KB
stm32wb1x_BLE_LLD_fw.bin	29 KB
stm32wb1x_BLE_Stack_full_fw.bin	168 KB
stm32wb1x_BLE_Stack_light_fw.bin	120 KB

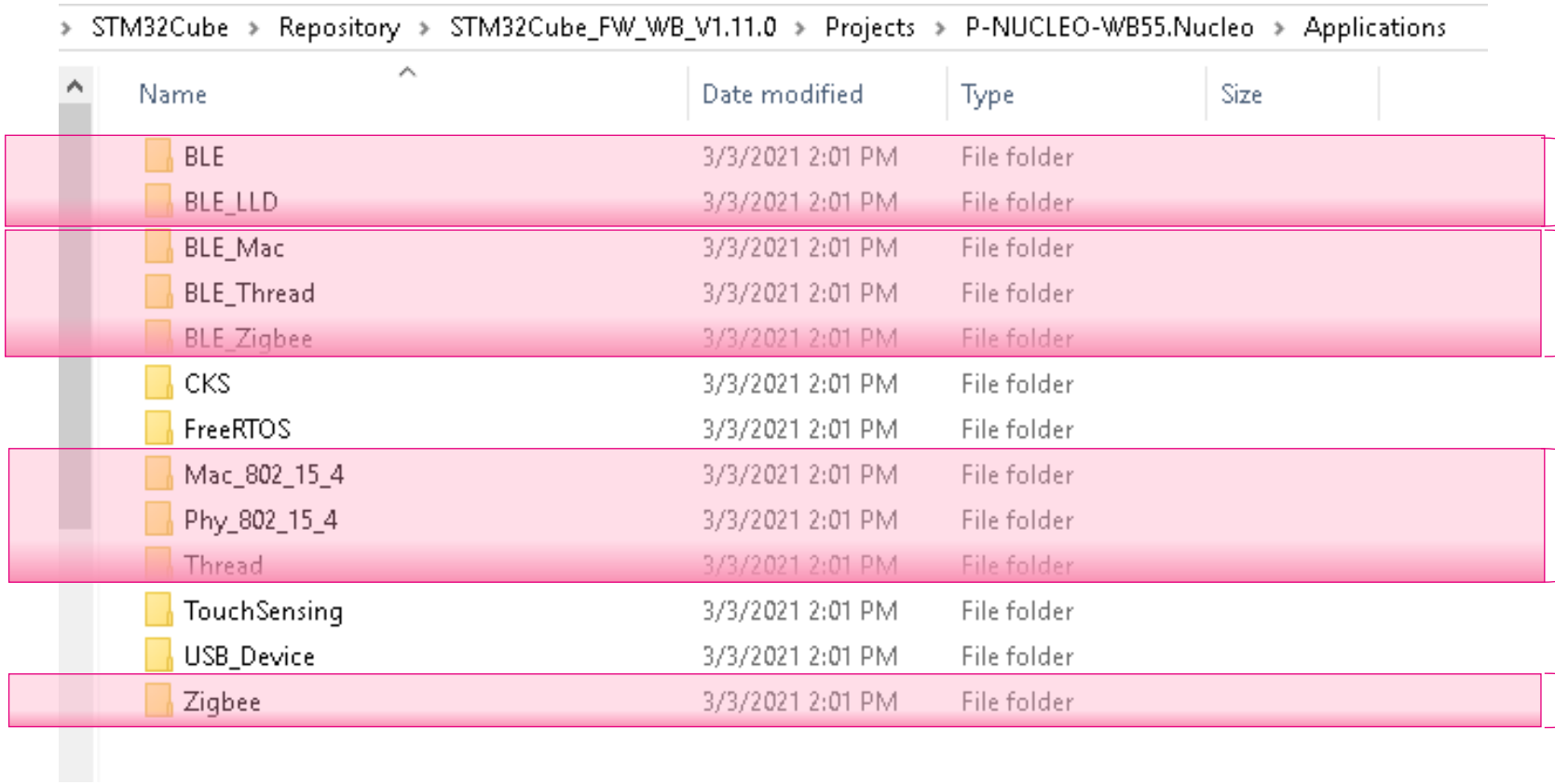
STM32WB3x

Name	Size
Release_Notes.html	42 KB
stm32wb3x_BLE_HCI_AdvScan_fw.bin	39 KB
stm32wb3x_BLE_HCILayer_fw.bin	77 KB
stm32wb3x_BLE_LLD_fw.bin	29 KB
stm32wb3x_BLE_Mac_802_15_4_fw.bin	252 KB
stm32wb3x_BLE_Stack_full_fw.bin	167 KB
stm32wb3x_BLE_Stack_light_fw.bin	120 KB
stm32wb3x_FUS_fw.bin	24 KB
stm32wb3x_FUS_fw_1_0_2.bin	24 KB
stm32wb3x_Mac_802_15_4_fw.bin	66 KB
stm32wb3x_Phy_802_15_4_fw.bin	86 KB
stm32wb3x_Thread_FTD_fw.bin	367 KB
stm32wb3x_Thread_MTD_fw.bin	291 KB
stm32wb3x_Zigbee_FFD_fw.bin	308 KB
stm32wb3x_Zigbee_RFD_fw.bin	250 KB

STM32WB5x

Name	Size
Release_Notes.html	82 KB
stm32wb5x_BLE_HCI_AdvScan_fw.bin	39 KB
stm32wb5x_BLE_HCILayer_fw.bin	77 KB
stm32wb5x_BLE_LLD_fw.bin	29 KB
stm32wb5x_BLE_Mac_802_15_4_fw.bin	252 KB
stm32wb5x_BLE_Stack_full_fw.bin	167 KB
stm32wb5x_BLE_Stack_light_fw.bin	120 KB
stm32wb5x_BLE_Thread_dynamic_fw.bin	527 KB
stm32wb5x_BLE_Thread_static_fw.bin	517 KB
stm32wb5x_BLE_Zigbee_FFD_dynamic_fw.bin	498 KB
stm32wb5x_BLE_Zigbee_FFD_static_fw.bin	491 KB
stm32wb5x_BLE_Zigbee_RFD_dynamic_fw.bin	440 KB
stm32wb5x_BLE_Zigbee_RFD_static_fw.bin	433 KB
stm32wb5x_FUS_fw.bin	24 KB
stm32wb5x_FUS_fw_1_0_2.bin	24 KB
stm32wb5x_Mac_802_15_4_fw.bin	66 KB
stm32wb5x_Phy_802_15_4_fw.bin	86 KB
stm32wb5x_Thread_FTD_fw.bin	367 KB
stm32wb5x_Thread_MTD_fw.bin	291 KB
stm32wb5x_Zigbee_FFD_fw.bin	308 KB
stm32wb5x_Zigbee_RFD_fw.bin	250 KB

应用目录



BLE应用：Beacon，血压，心率，数据透传，Mesh等

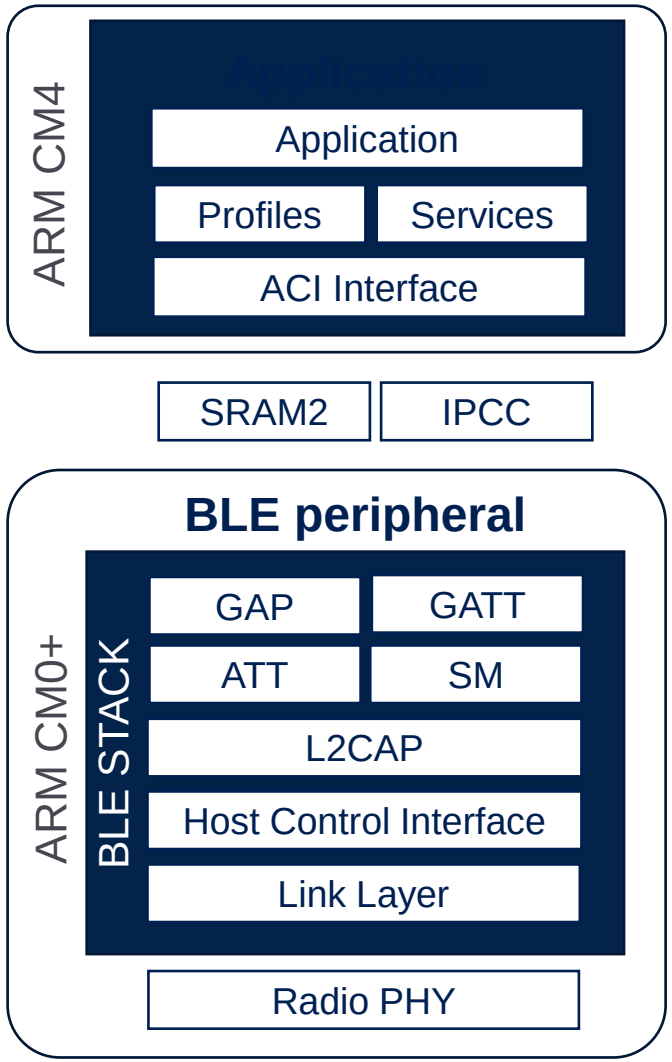
BLE与其他协议共存应用

802.15.4与thread应用

Zigbee应用

life.augmented

BLE软件架构概览



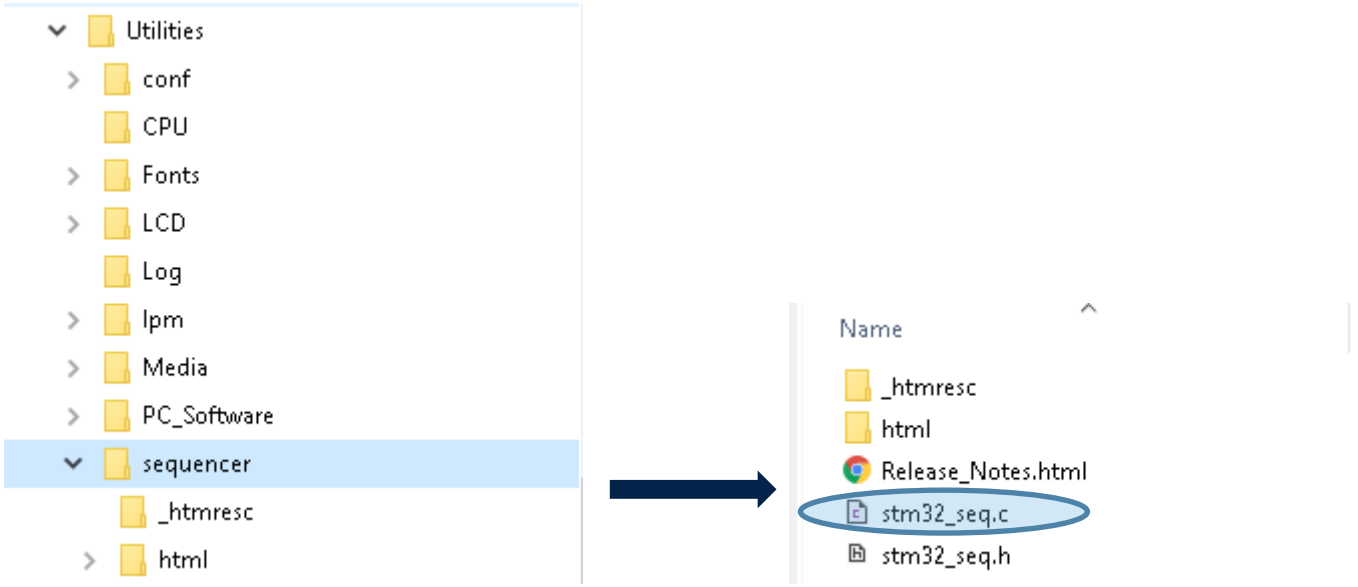
- Cortex-M4 应用
  - 收集和计算要通过BLE发送的数据.
  - 受益于标准或私有的应用profile，应用使用BLE栈的服务和特征能力来发送数据.
- 通过通用属性配置文件进行通信 **(GATT).**
  - 基于HCI协议
  - 使用共享SRAM2和IPCC
- BLE外设Cortex-M0+和射频PHY
  - 包含LE控制器和LE主机
  - 所有的实时连接层和射频PHY层交互

2.调度器

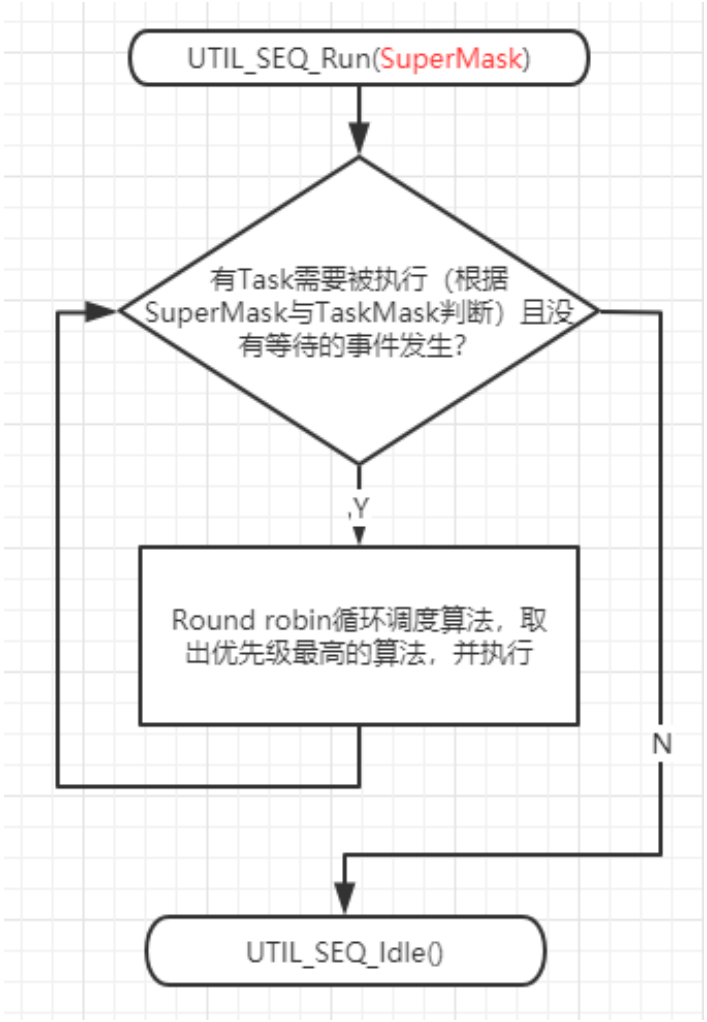
调度器

调度器提供以下特性:

- ✓ 最高32个任务
- ✓ 请求要执行的任务
- ✓ 暂停和恢复任务
- ✓ Round Robin调度
- ✓ 2个任务优先级（优先级:0 > 1）
- ✓ 动态优先级



UTIL_SEQ_Run() / UTIL_SEQ _Idle()

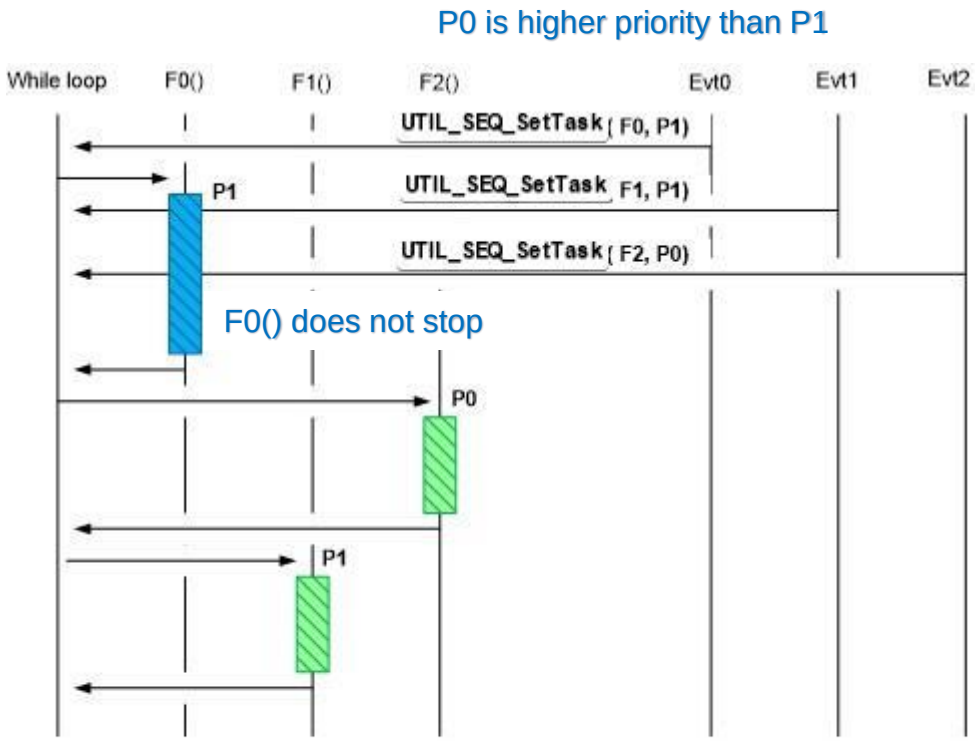


- UTIL_SEQ_Run() 执行Round robin任务调度算法
- UTIL_SEQ_Idle()中可以让MCU进入低功耗状态。

- Round Robin调度策略：
  - 优先级0的Task先被调度
  - 同一优先级，Task ID小的优先被调度

life.augmented

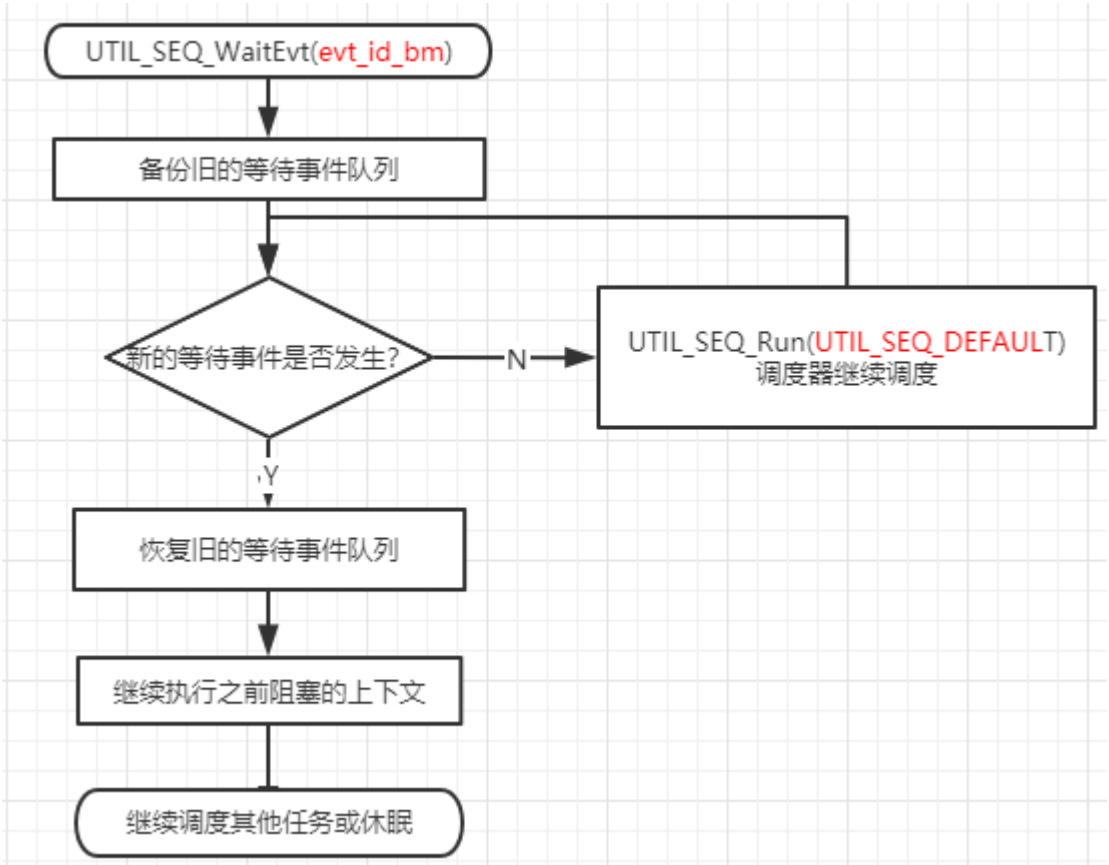
调度器与优先级



- 特征：
  - 任务不会被抢占。
  - 基于优先级调度。
  - 动态优先级。

life.augmented

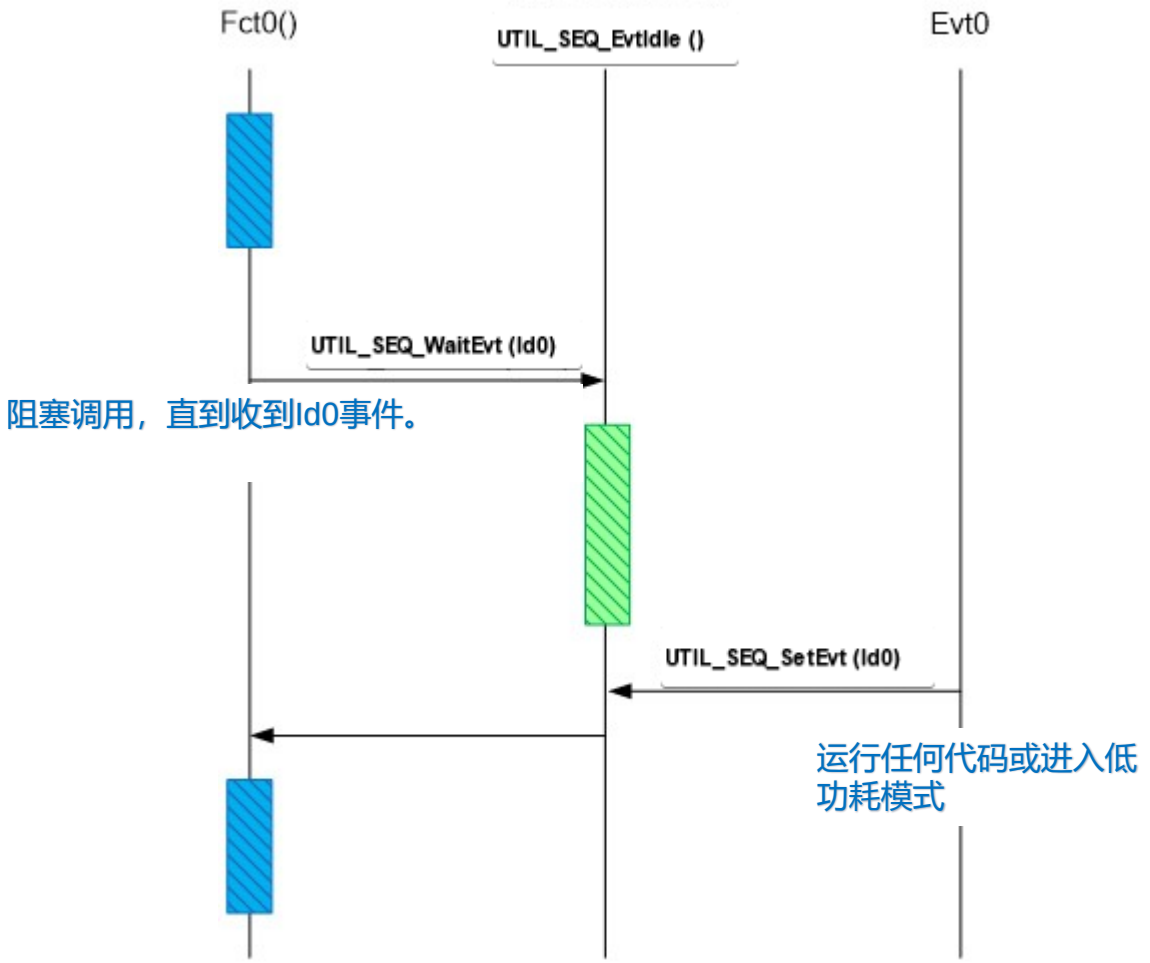
UTIL_SEQ_WaitEvt()/ UTIL_SEQ_SetEvt()




```
/**
 *  this function can be nested
 */
void UTIL_SEQ_SetEvt( UTIL_SEQ_bm_t evt_id_bm )
{
  UTIL_SEQ_ENTER_CRITICAL_SECTION( );

  EvtSet |= evt_id_bm;

  UTIL_SEQ_EXIT_CRITICAL_SECTION( );

  return;
}
```

life.augmented

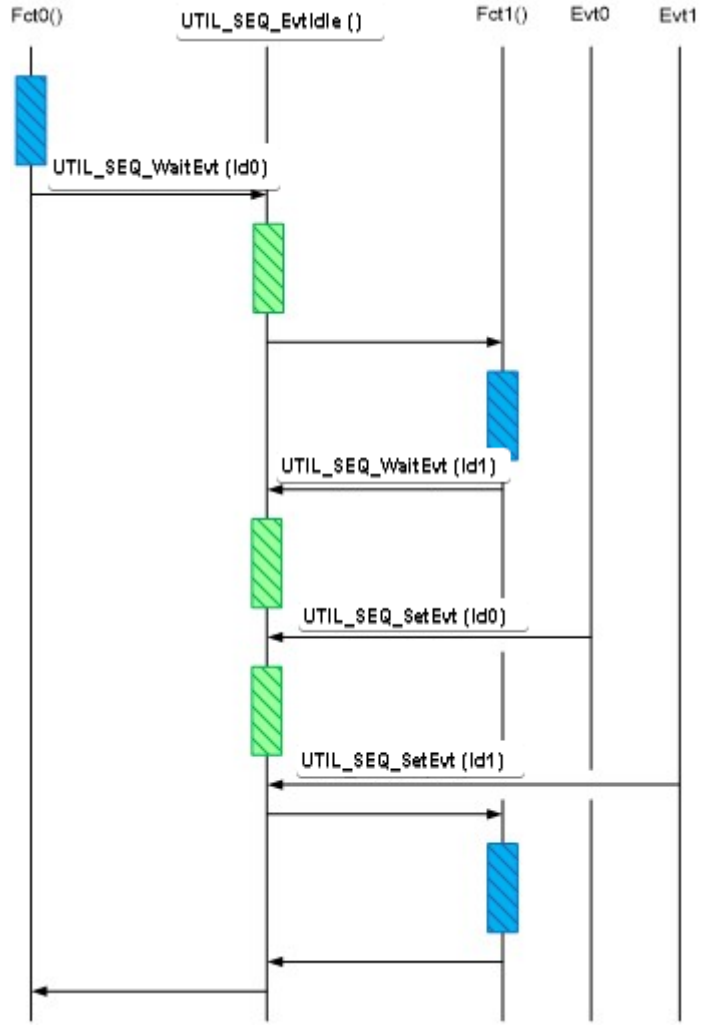
事件 (1/2)
增强阻塞特性



- Fct0()等待事件Id0。
- 阻塞调用。
- CPU在UTIL_SEQ_EvtIdle()中循环，直到收到Id0事件。
- 应用可以在UTIL_SEQ_EvtIdle() 中运行任何代码，或进入低功耗模式。

life.augmented

事件 (2/2)
增强阻塞特性



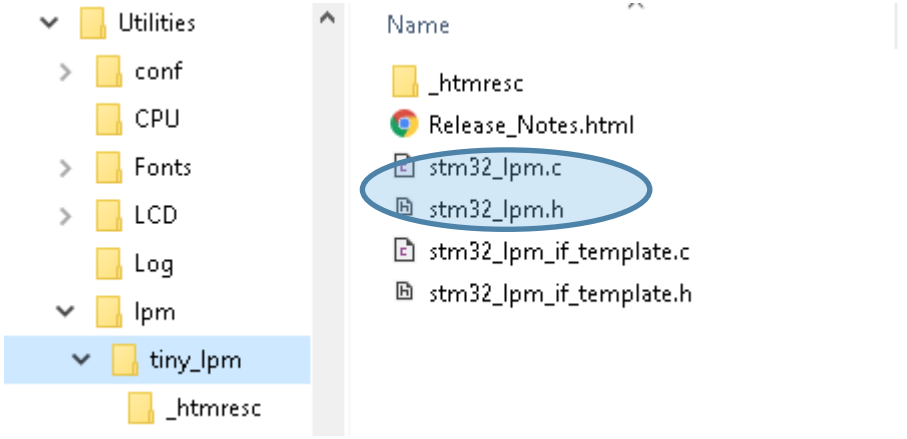
- 等待事件
- UTIL_SEQ_EvtIdle（）循环执行
- 先处理最新等待的事件
- 后处理旧的事件

3.低功耗管理

低功耗管理器

低功耗管理器提供以下特性:

- ✓低功耗模式选择(Sleep/Stop/Off 模式)
- ✓低功耗模式执行
- ✓进入/退出低功耗模式时回调
- ✓32个对象共享

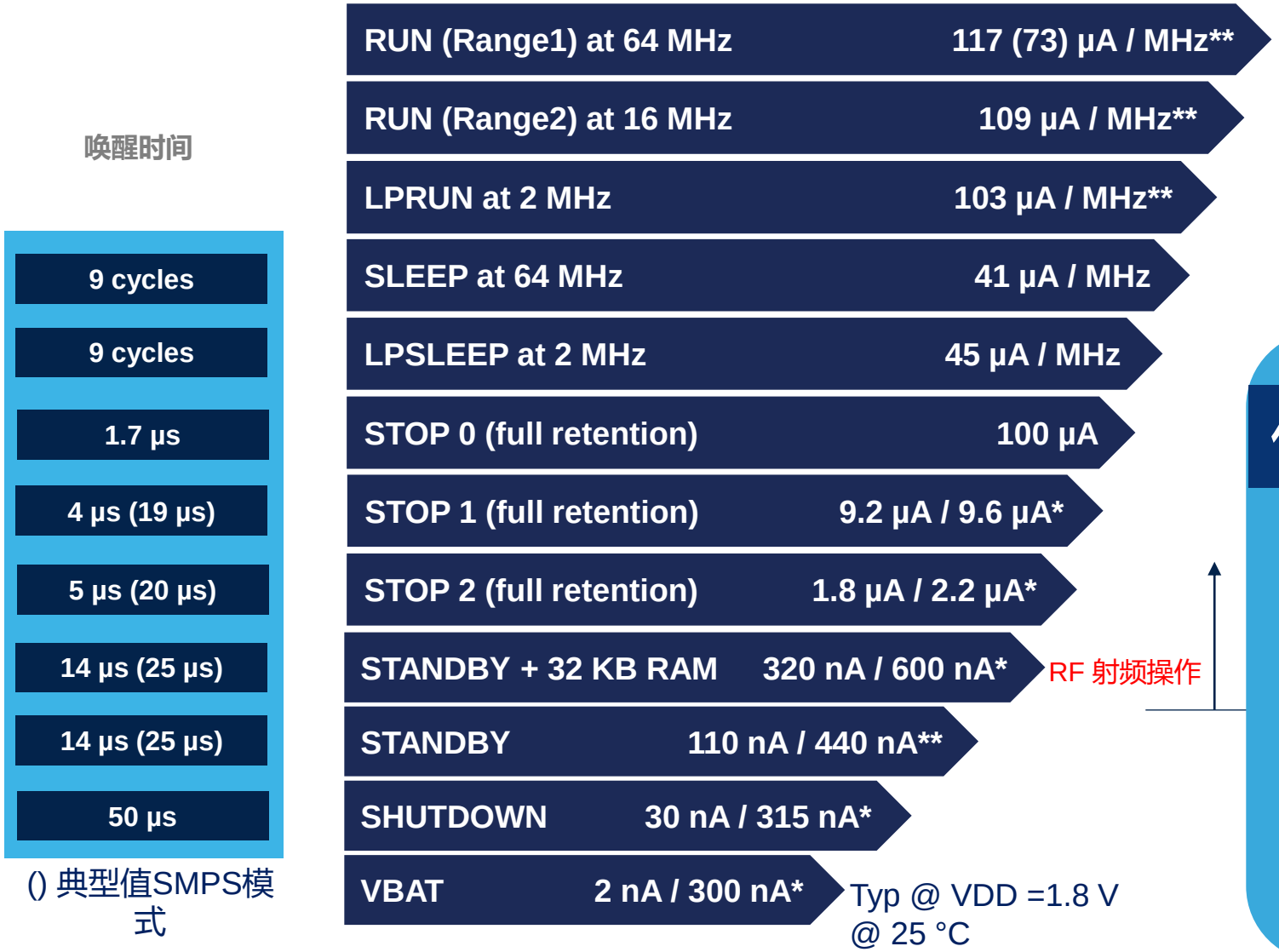


life.augmented

多种低功耗模式

灵活的功耗控制

- 高效运行
- 8 个低功耗模式, 几个子模式
- 高灵活性



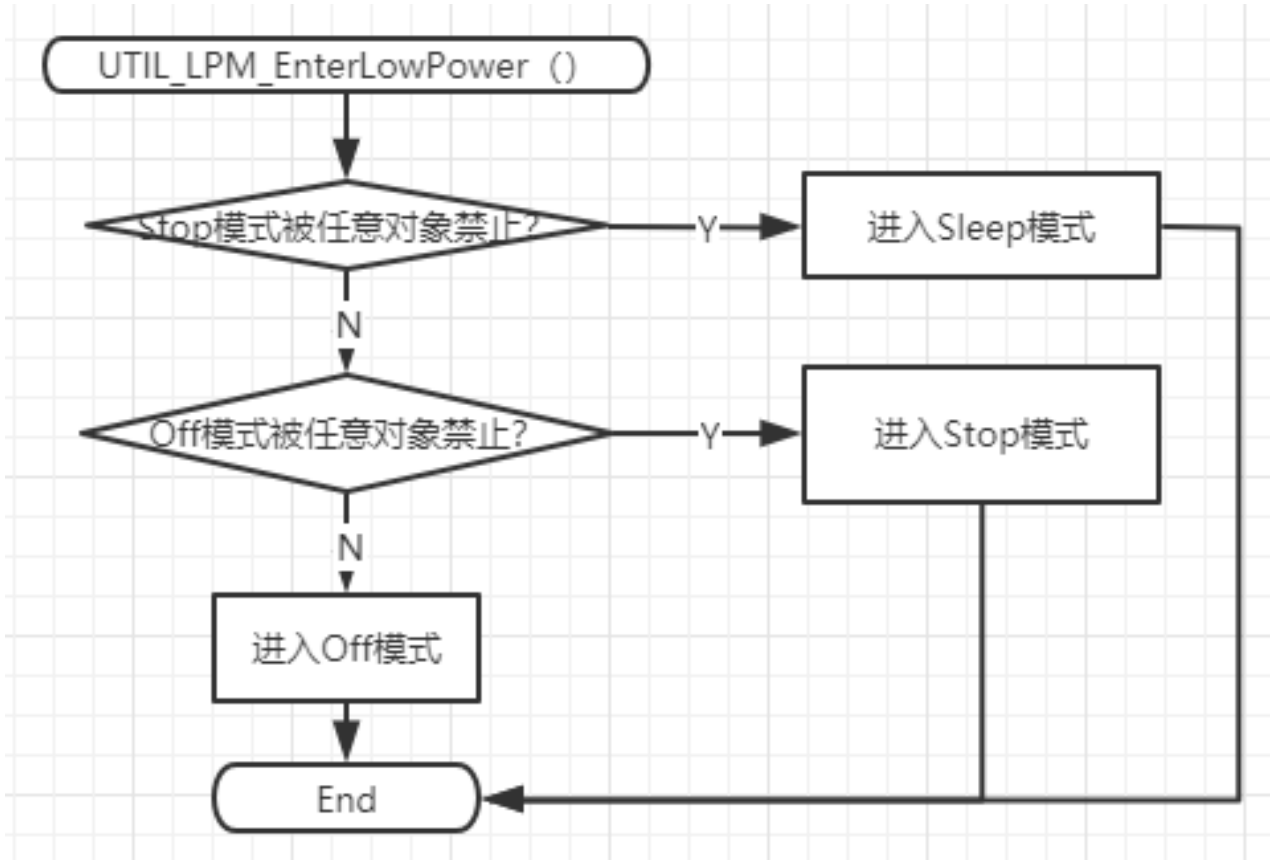
优势

CoreMark® An EEMBC Benchmark

- 高性能
  - ➔ CoreMark 分数 = 219
- 出色的电源效率
  - ➔ ULPBbench 分数 = 303
- 当RF和SMPS同时打开时
  - ➔ M4 核心消耗 = 53µA/Mhz

life.augmented

低功耗管理器


UTIL_LPM_EnterLowPower（）
Stop模式被任意对象禁止?
Y
进入Sleep模式
N
Off模式被任意对象禁止?
Y
进入Stop模式
N
进入Off模式
End

4.定时器管理

定时器服务器

定时器服务器提供以下特性:

- ✓ 多达255个虚拟定时器 (或由于RAM限制减少)
- ✓ 单次和重复模式
- ✓ 停止一个虚拟定时器和使用一个不同的超时值重启定时器
- ✓ 动态添加和删除
- ✓ 超时从1到(2^32)-1 滴答

警告:

- ✓ 运行中的虚拟计时器应在重启前停止。

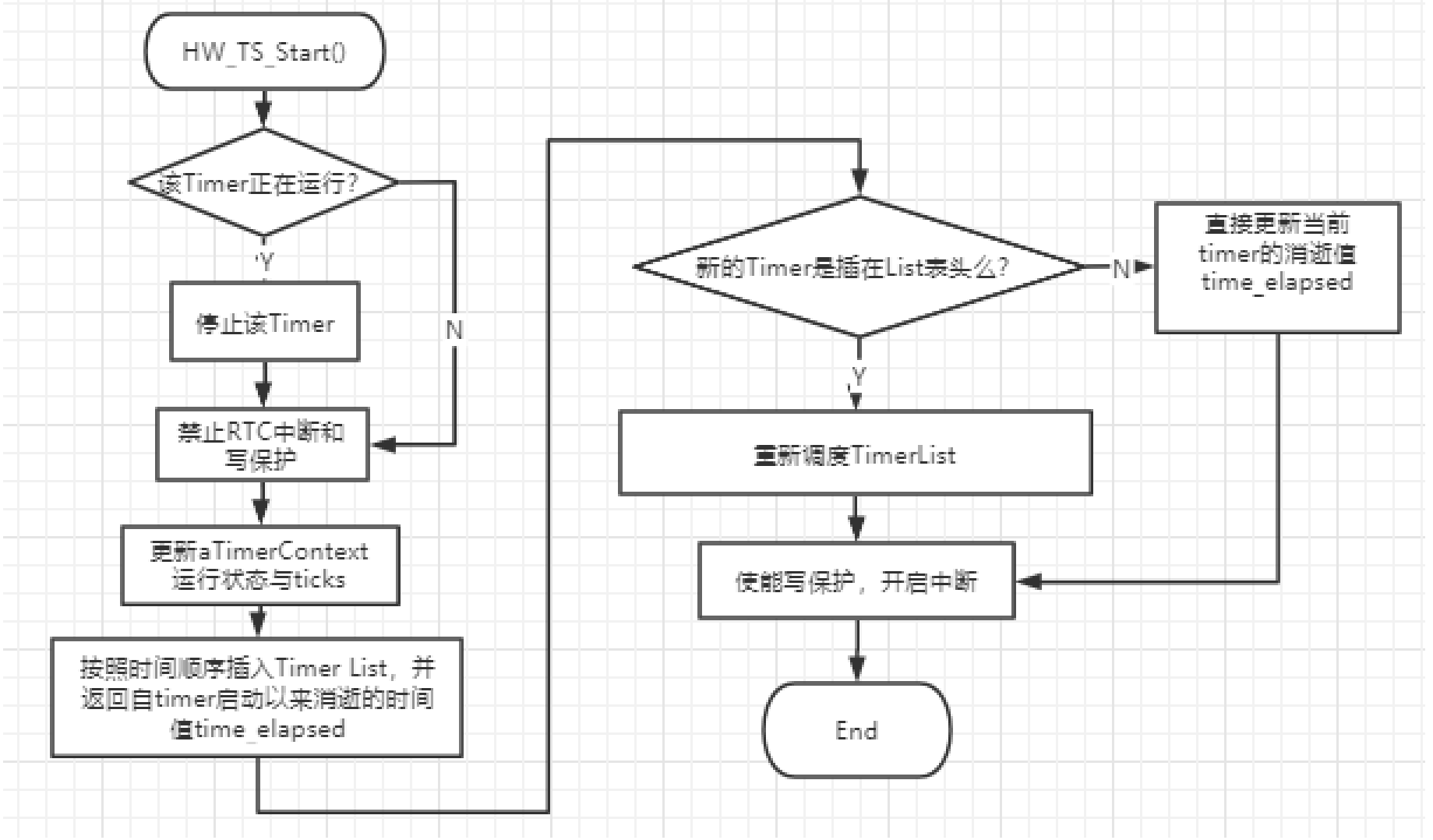
HW_TS_Init()

- 使用RTC作为计时器
- 两种初始化模式：
  - hw_ts_InitMode_Full
    - 从standby模式中重启时，会将定时器所有的内容进行初始化
  - hw_ts_InitMode_Limited
    - 从standby模式中重启时，定时器不会初始化它的上下文，只会初始化丢失的部分寄存器内容
- 中断优先级配置

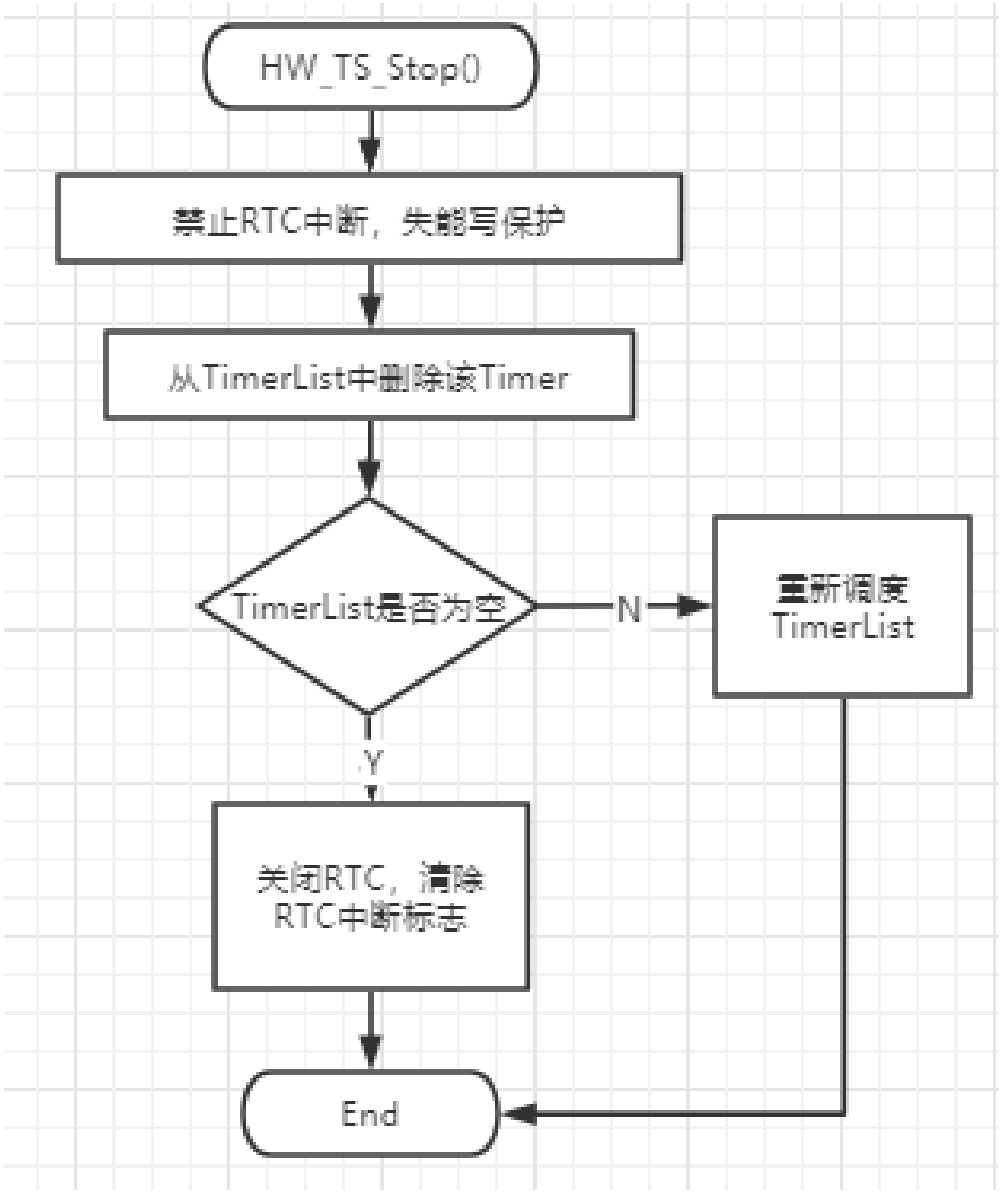
HW_TS_Create()/HW_TS_Delete()

- HW_TS_Create()
  - 创建时保存以下信息：
    - Task ID
    - Timer模式
    - 回调函数
  - 返回以下信息：
    - 是否创建成功
    - Timer ID
- HW_TS_Delete()
  - Timer的删除：
    - 停止Timer
    - 从Timer上下文中删除

HW_TS_Start()


HW_TS_Start()
该Timer正在运行?
Y
N
停止该Timer
禁止RTC中断和
写保护
更新aTimerContext
运行状态与ticks
按照时间顺序插入Timer List，并
返回自timer启动以来消逝的时间
值time_elapsed
新的Timer是插在List表头么?
N
直接更新当前
timer的消逝值
time_elapsed
Y
重新调度TimerList
使能写保护，开启中断
End

HW_TS_Stop()


HW_TS_Stop()
禁止RTC中断，失能写保护
从TimerList中删除该Timer
TimerList是否为空
N
重新调度
TimerList
Y
关闭RTC，清除
RTC中断标志
End

BLE基础介绍

1 LL(Link Layer)基本介绍

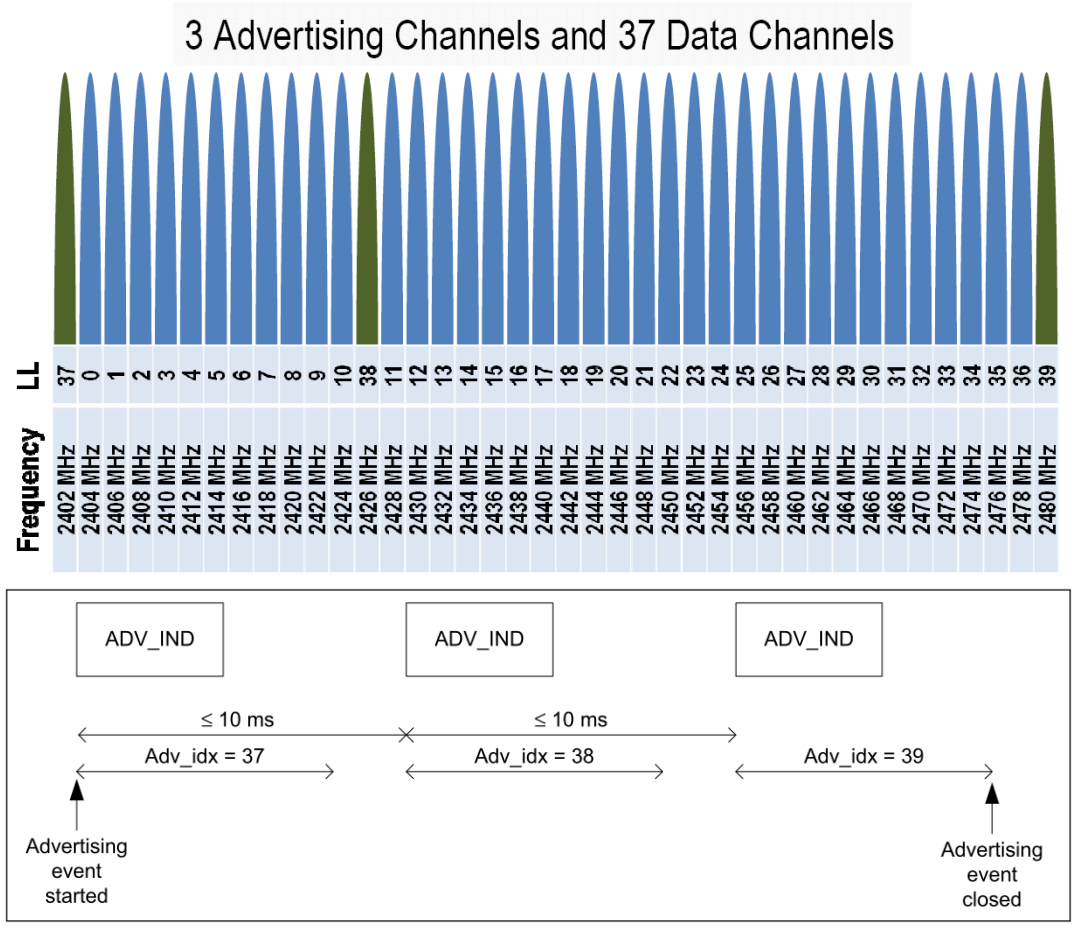
2 GAP 基本介绍

3 GATT 简介

life.augmented

1. LL(Link Layer)链路层基本介绍

BLE链路层信道

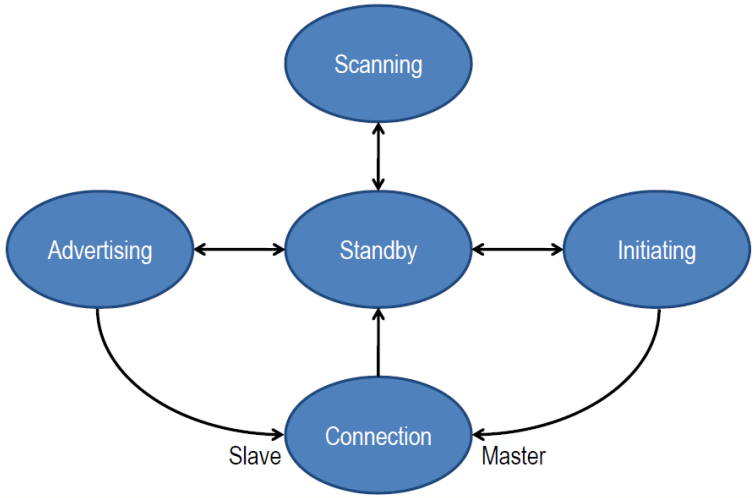


- 频段：
  - 2402~2480MHz
- 信道：
  - 40个，每个2MHz
- 广播信道：
  - 37，38，39
- 跳频算法
  - $f_{n+1} = (f_n + hop) \bmod 37$

life.augmented

链路层状态

- 链路层定义了两个设备如何使用射频在彼此之间传输信息, 其状态有:
  - Standby
    - 不传输或接收数据包
  - Advertising
    - 在广播信道广播广告(Broadcast advertisements in advertising channels)
  - Scanning
    - 被动扫描(Passive scanning)
    - 主动扫描(Active scanning)
  - Initiating
    - 发起与advertiser的连接(Initiate connection to advertiser)
  - Connection
    - 发起设备将变成主机(Initiator device will be in Master Role)
    - 广播设备将变成从机(Advertiser device will be in Slave Role)



life.augmented

链路层角色

- BLE蓝牙射频以五种不同的角色运行:
  - Advertiser -- 定期在广播信道播放广告
  - Scanner -- 寻找其他BLE蓝牙射频广播
  - Initiator -- 请求与Advertiser建立LL连接
  - Master -- 确定LL连接的操作时间表.
  - Slave -- 仅在从LL连接的主设备接收到BLE蓝牙数据包时才能进行传输.

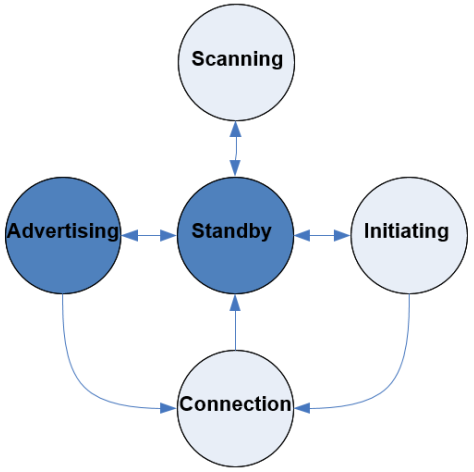
2.GAP 基本介绍

GAP 角色(1/2)

- GAP Roles
  - 广播者Broadcaster
  - 观察者Observer

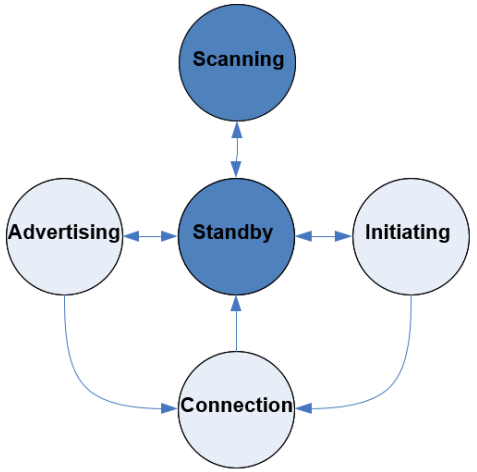
Broadcaster

Sends advertising events
Can include characteristics and service data
Doesn't need receiver
Can be discoverable if it does have receiver



Observer

Receives advertising events
Listens for characteristics and service data
Doesn't need transmitter
Can discover devices if it does have transmitter



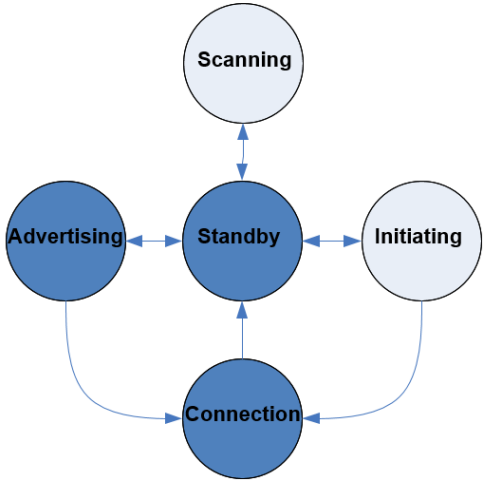
life.augmented

GAP 角色 (2/2)

- GAP Roles
  - 外设Peripheral (Salve)
  - 中央设备Central (Master)

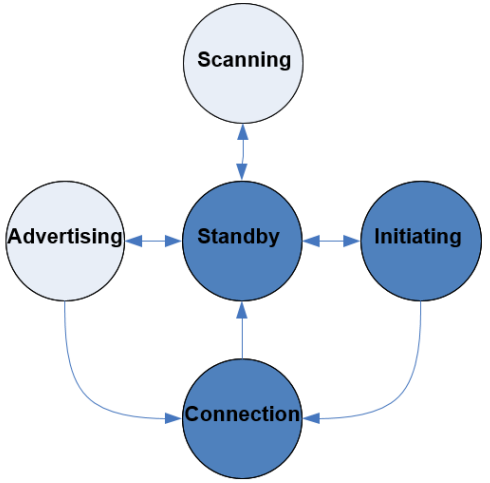
Peripheral

Has transmitter and receiver
Always slave
Connectable advertising



Central

Has transmitter and receiver
Always master
Never advertises

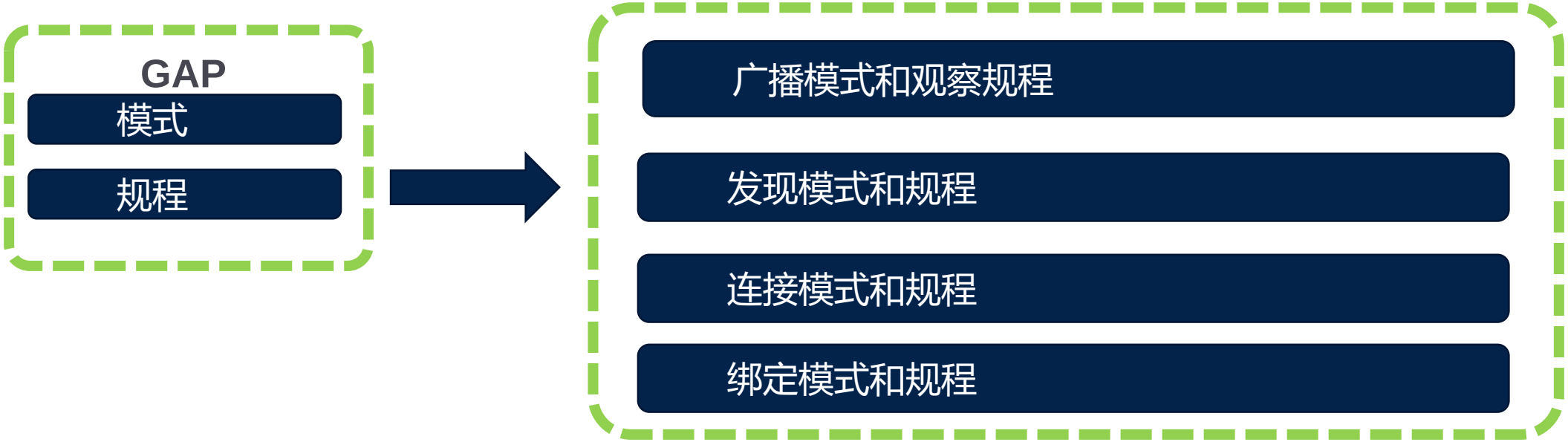


- 管理连接、安全性，并允许LE设备充当主设备或/和从设备

life.augmented

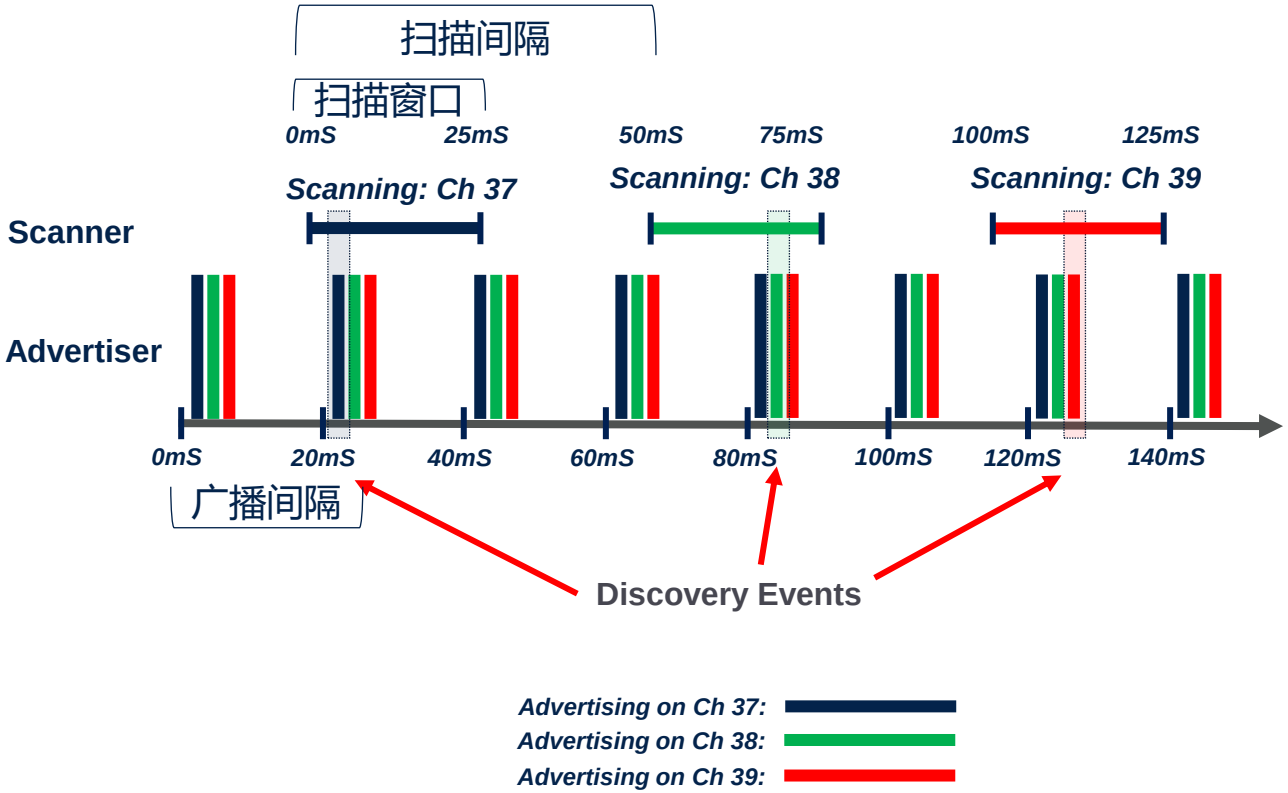
GAP 模式和规程

- 设备在一段时间内以某种状态运行
  - 模式（Mode）　　强调设备在一段时间内被设置为某种状态
  - 规程（Procedure）强调设备在某种模式下执行某些操作
- 两个基本概念用于描述设备的行为:



广播(Advertising) & 扫描(Scanning)参数设置

- 广播和扫描参数的设置是发现BLE设备的关键。
- 必须考虑时间间隔和扫描窗口设置
  - Advertising Interval (20mS – 10.24 seconds for most packet types)
  - Scanning Interval (小于10.24 秒)
  - Scanning Window (小于 10.24 秒 以及小于扫描间隔)



广播参数设置:

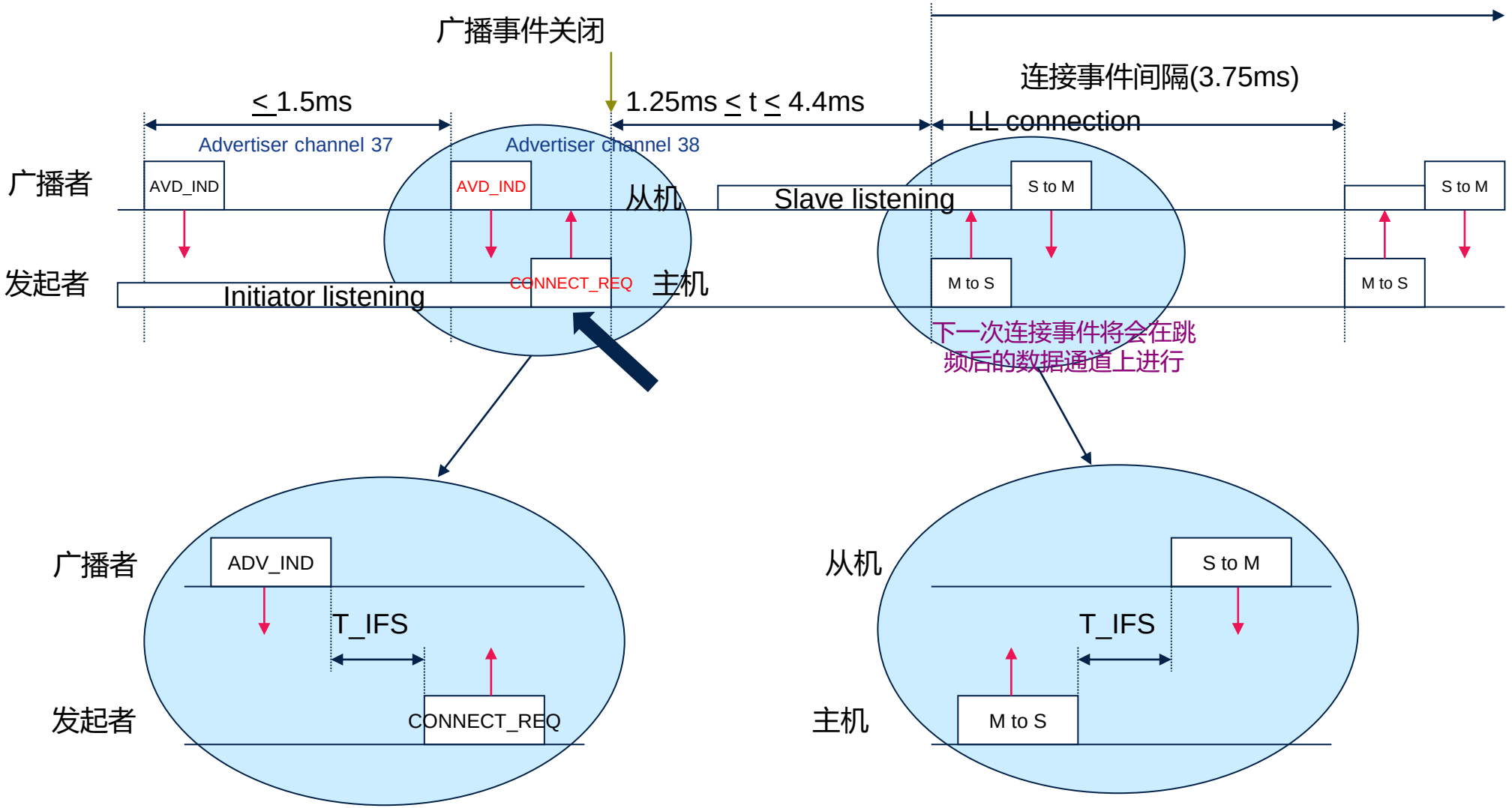
- **Advertising Interval: 20mS**

扫描参数设置:

- **Scan Interval: 50mS**
- **Scan Window: 25mS**

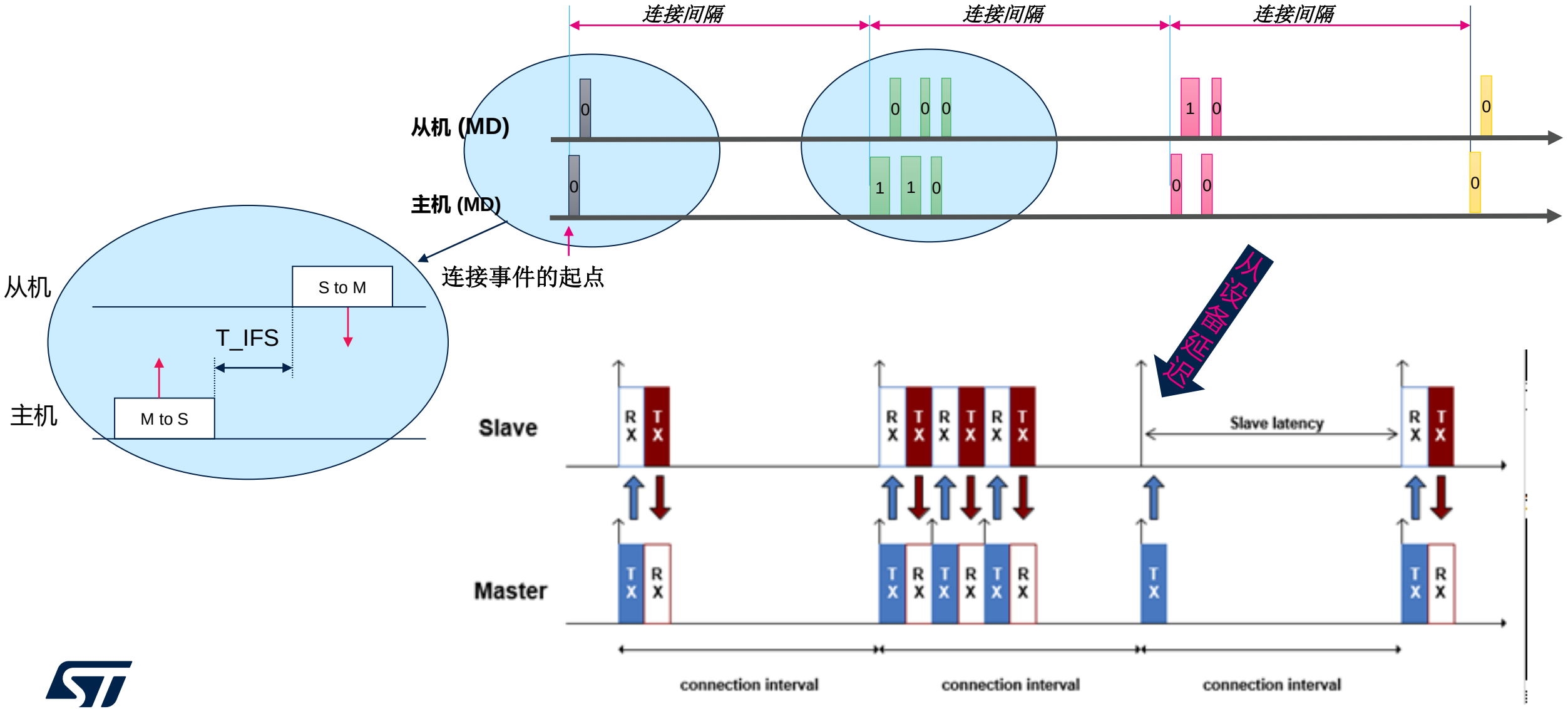
life.augmented

连接过程



life.augmented

连接参数设置

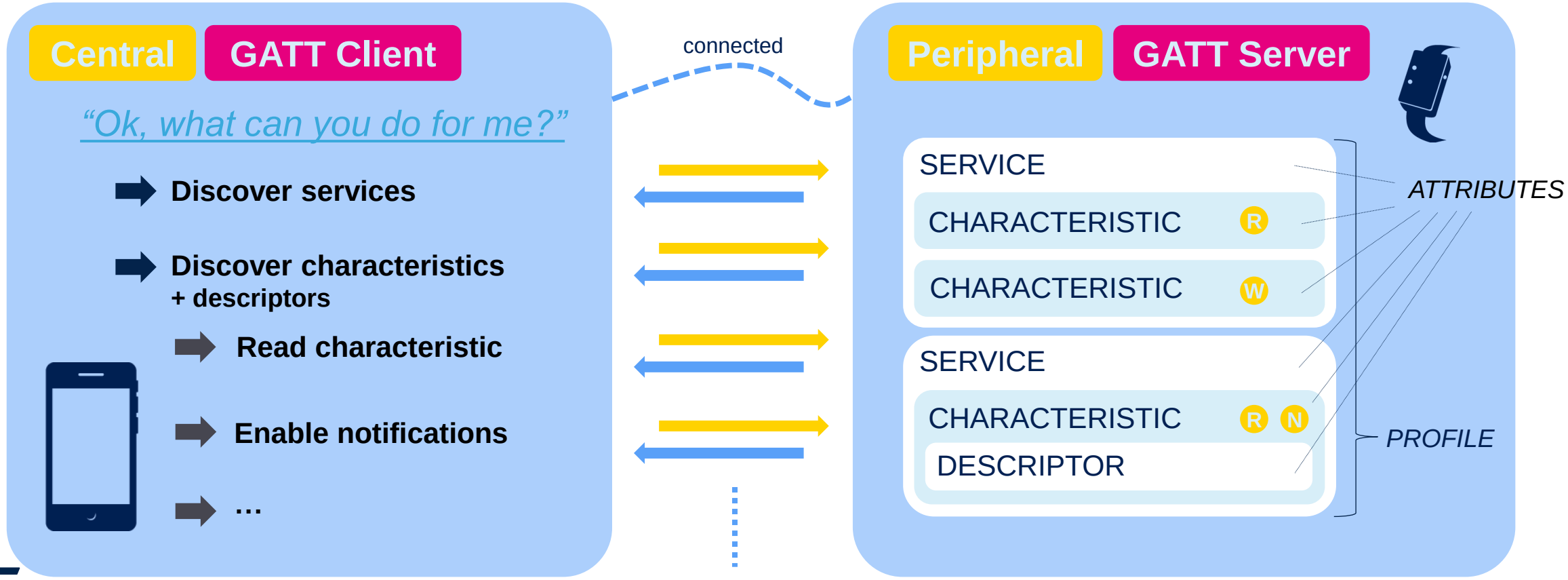

连接间隔
连接间隔
连接间隔
从机 (MD)
主机 (MD)
连接事件的起点
从机
主机
S to M
T_IFS
M to S
从设备延迟
Slave
Master
Slave latency
connection interval
connection interval
connection interval

2.GATT基本介绍

GATT角色介绍

通用属性配置文件GATT – Generic Attribute Profile

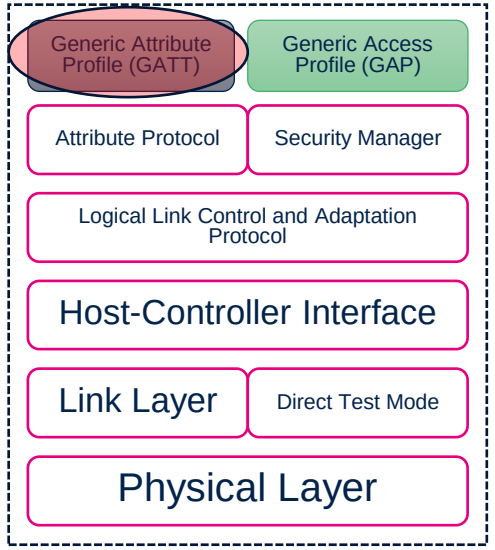
- 定义和管理输入/输出数据交换

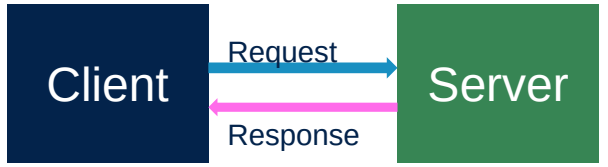


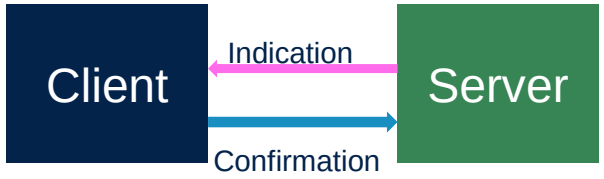
根据中心设备定义(外设建议)的连接间隔重复通信尝试

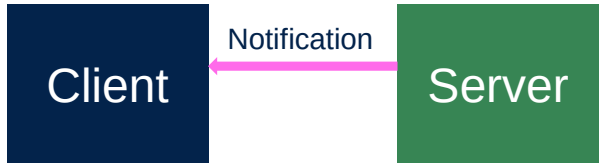
GATT Client Server架构

- 配置和角色
  - 客户端(Client)
    - 向Server发起命令(Command)、确认(Confirmation)和请求(Request)
  - 服务端(Server)
    - 向Client发送回复(Response)、指示(Indication)和通知(Notification)





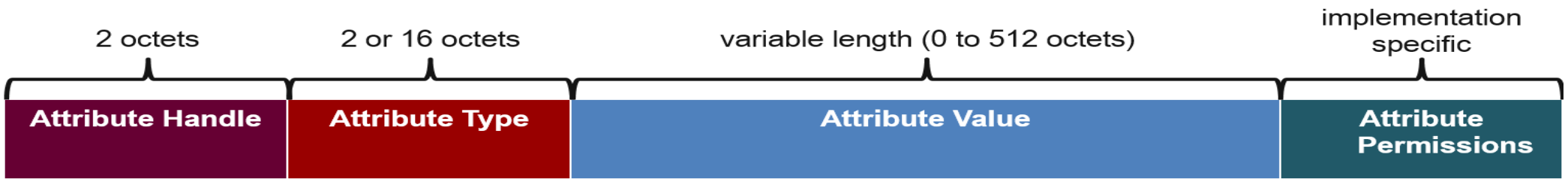




life.augmented

ATT基本概念

- 属性
  - 带有标签，可被寻址的数据。
  - 句柄：用于寻址
  - 属性类型：可被公开的各种各样的数据类型
  - 属性值：设备公开的状态信息
  - 属性许可：包括使用许可，认证许可，授权许可。

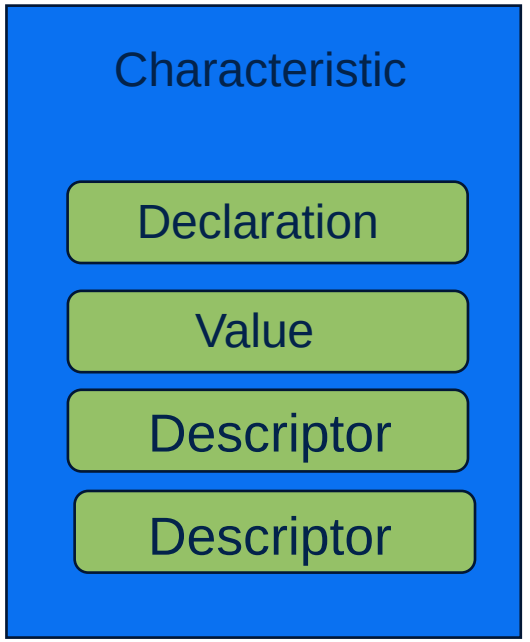


Characteristic基本概念

- 特性Characteristic由三个基本要素组成:
  - 声明Declaration：声明是一个特性的开始;
    - 性质（读取/写入/通知/指示/广播/命令/签名认证）
    - 数值句柄：特性值的存储位置
    - 特性UUID
  - 值Value

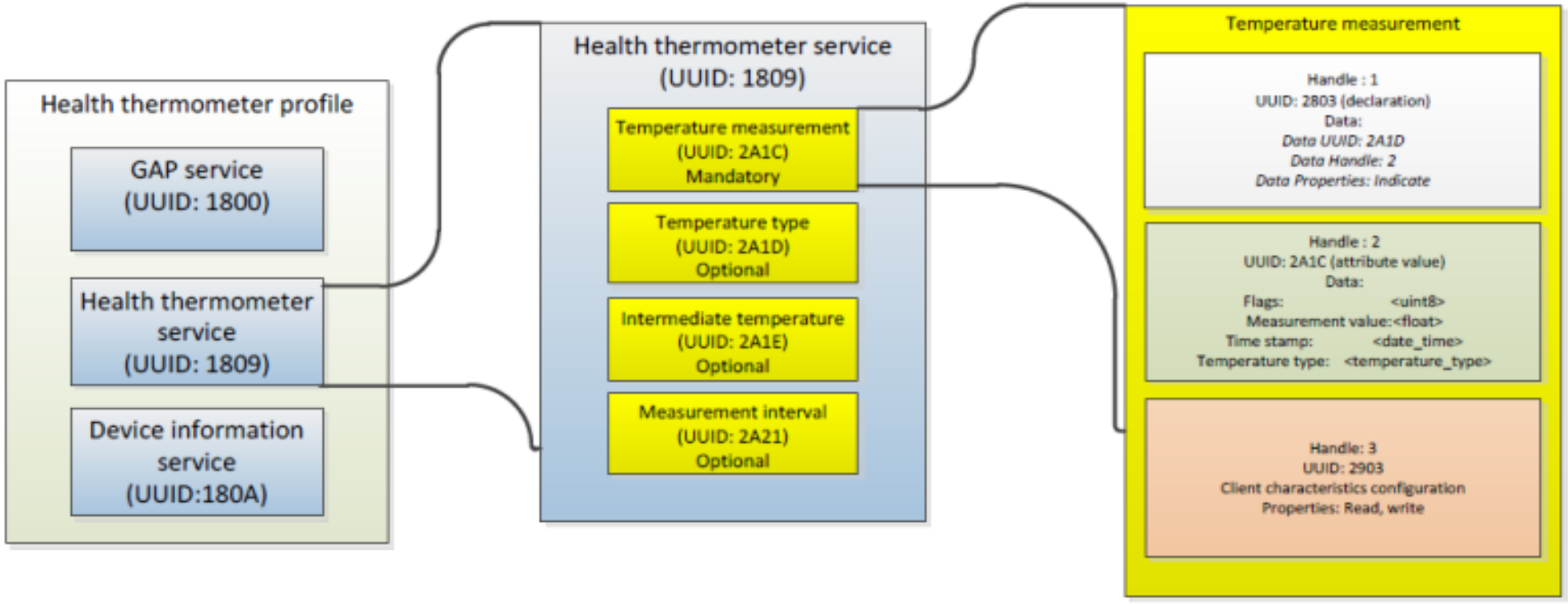
    包含此特性的实际值
  - 描述符Descriptors

    描述符包含此特性的附加信息或配置.
    - 附加信息
    - 任何数量
    - 任何顺序
    - 可以是供应商特定的



协议结构

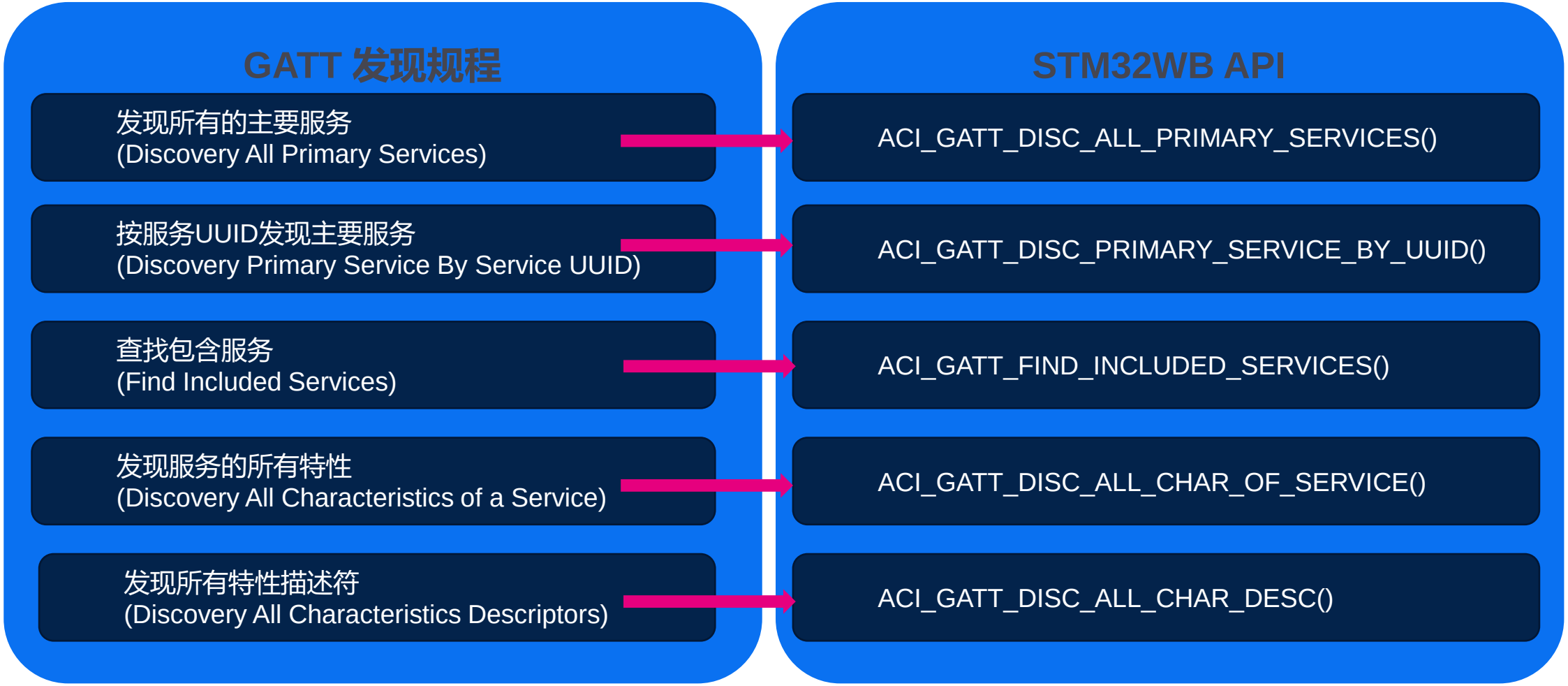
- 协议/服务和特性之间的关系



life.augmented

GATT发现规程

- GATT程序定义了服务，特征及其描述符发现然后使用的标准方法



参考文档

更多关于GATT API相关的内容可以参考AN5270文档：

AN5270

Application note

STM32WBx5 Bluetooth® Low Energy (BLE) wireless interface

BLE低功耗设计

1 功耗测量工具

2 低功耗模式

3 SMPS

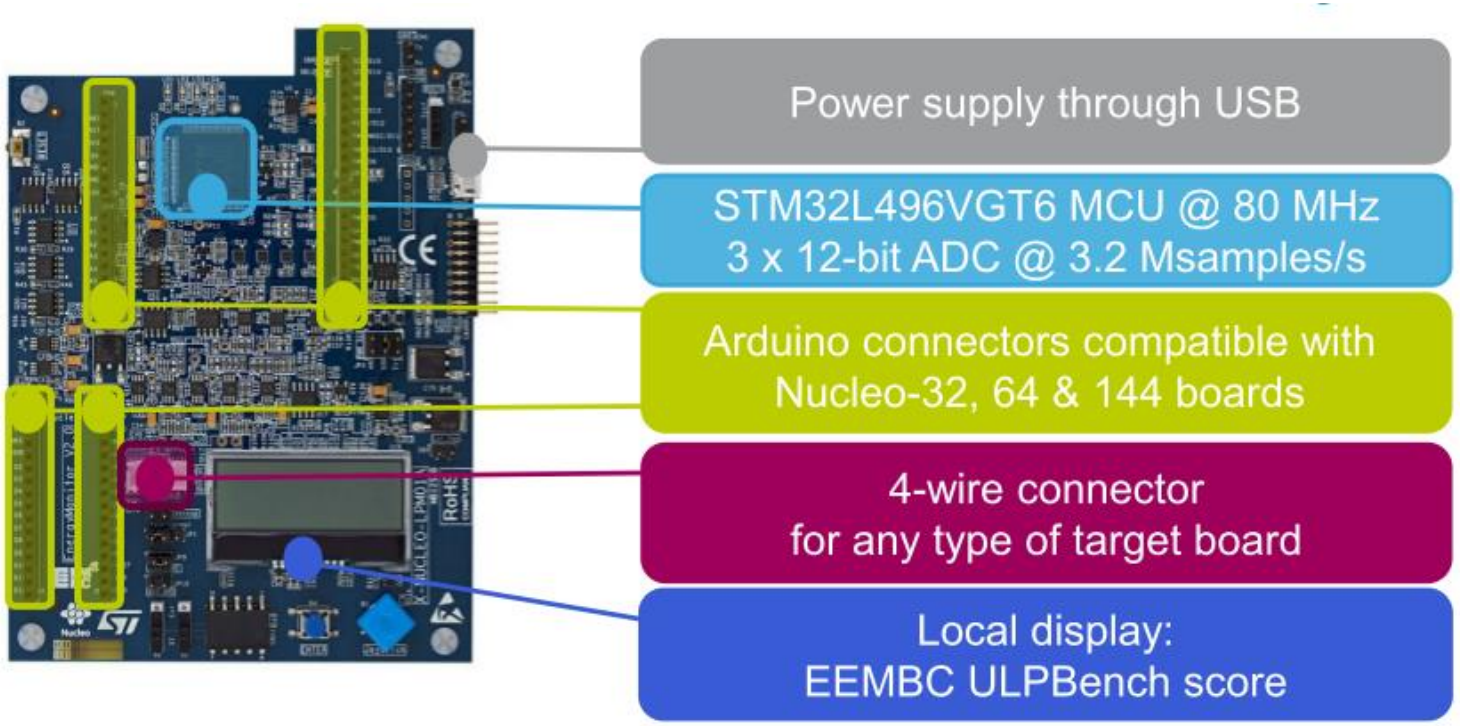
4 广播和连接参数

life.augmented

1.功耗测量工具

X-NUCLEO-LPM01A

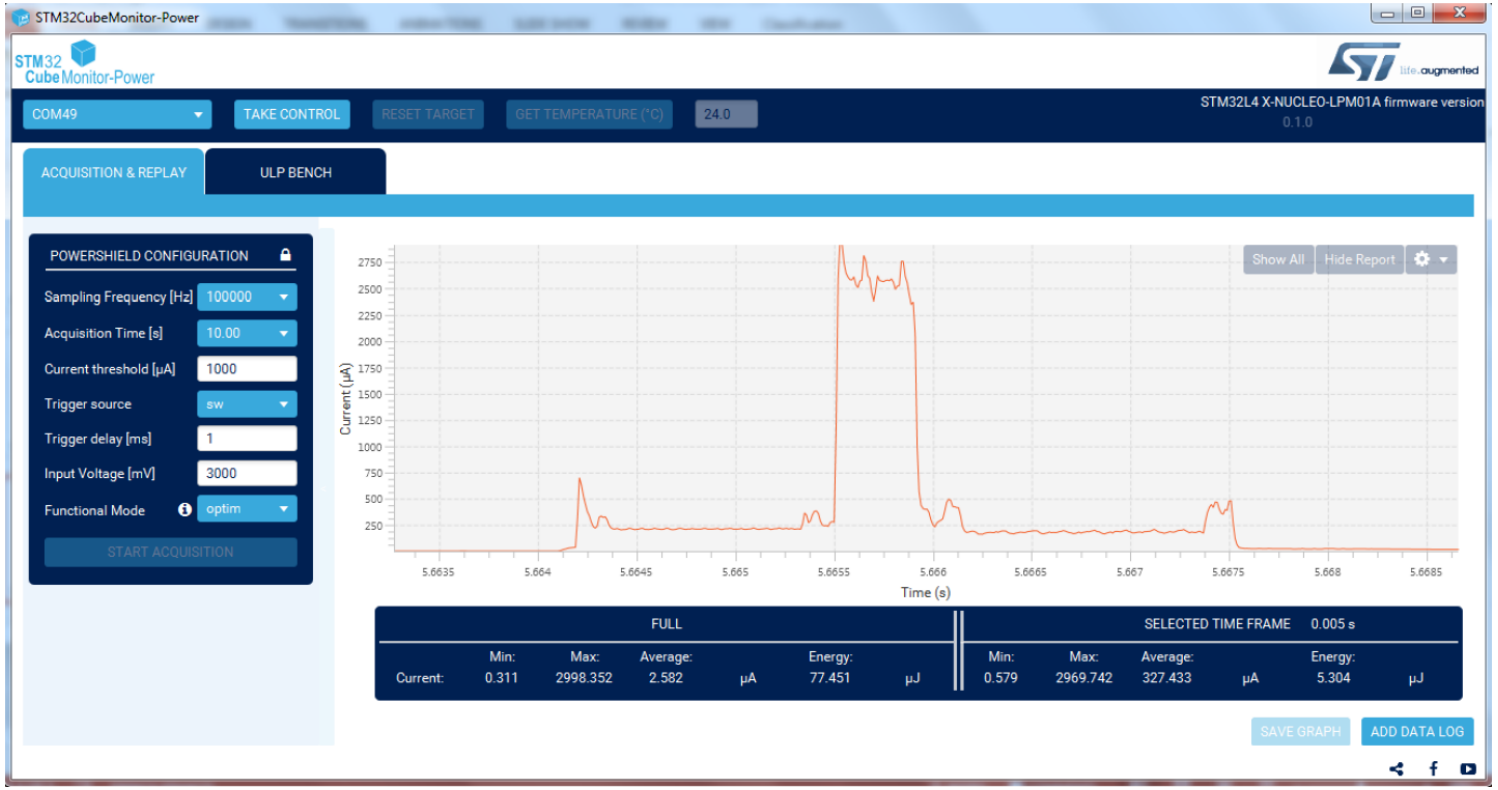
- 精确功耗测量，支持任意目标板
- 可编程电压源范围:1.8v~3.3v
- 静态测量
  - 电流范围：1nA~200mA
- 动态测量：
  - 电流范围：100nA~50mA
  - 100KHz带宽，3.2Msps采样率
  - 功耗测量范围：180nW~165mW
- Arduino或4线连接接口
- 图形可视化
- EEMBC能量监测v2.0
- 直接计算ULPBench



life.augmented

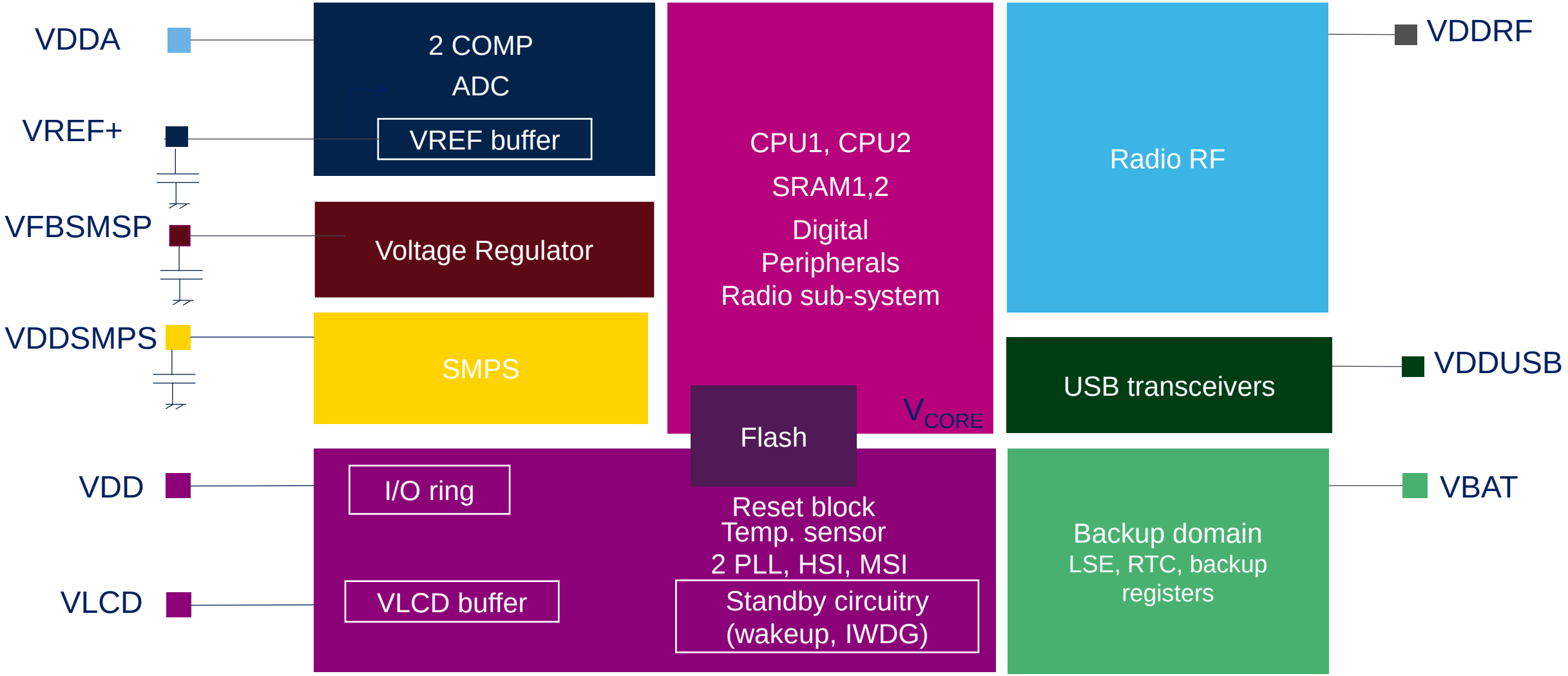
CubeMonitor-Power

- 上位机远程控制STM32功耗板
- 可视化功耗测量
- 支持命令行自定义测试案例



2.低功耗模式

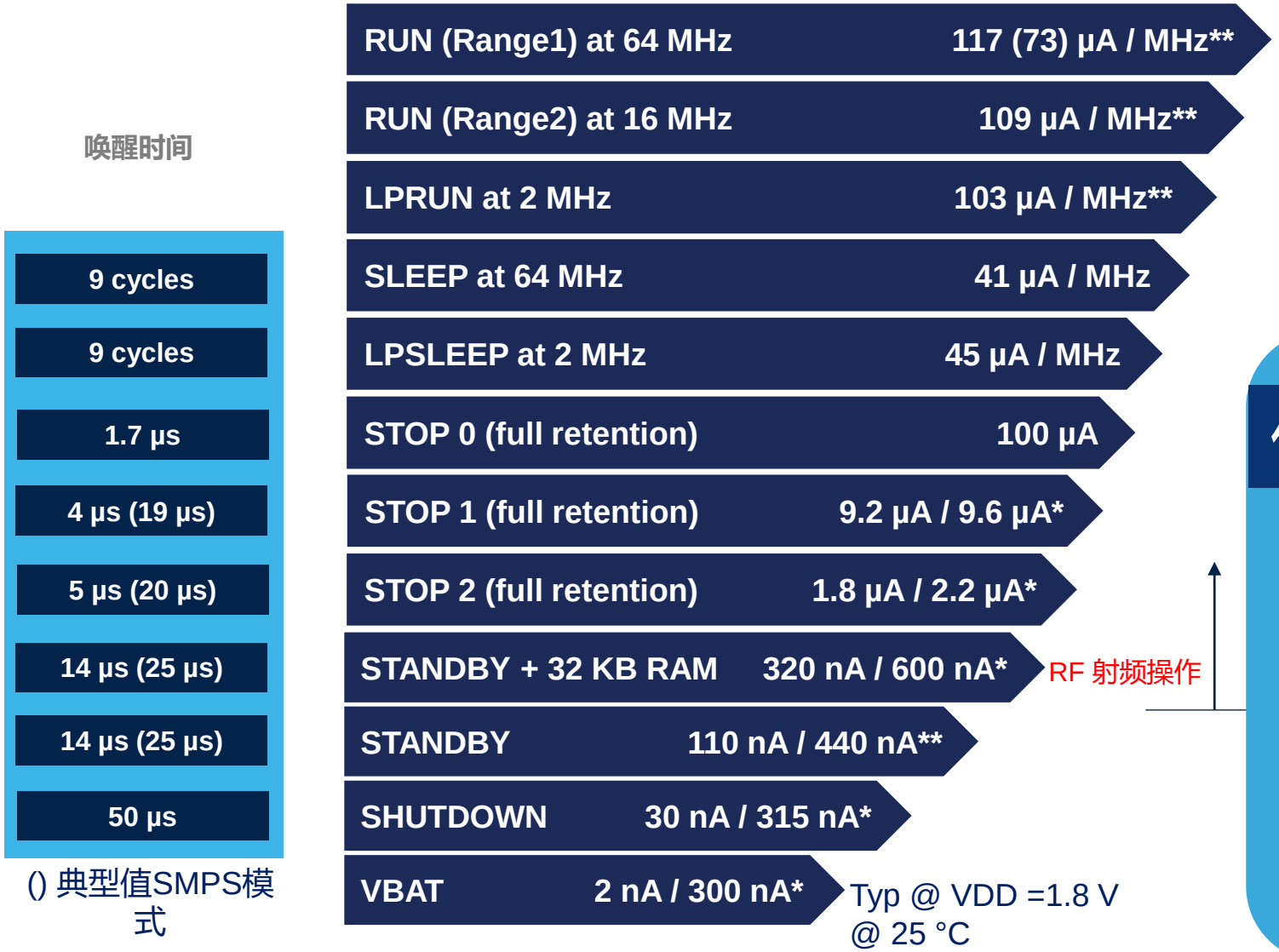
电源配置概览


VDDA
VREF+
VFBSMSP
VDDSMPS
VDD
VLCD
2 COMP
ADC
VREF buffer
Voltage Regulator
SMPS
I/O ring
VLCD buffer
CPU1, CPU2
SRAM1,2
Digital
Peripherals
Radio sub-system
VCORE
Flash
Reset block
Temp. sensor
2 PLL, HSI, MSI
Standby circuitry
(wakeup, IWDG)
Radio RF
USB transceivers
Backup domain
LSE, RTC, backup
registers
VDDRF
VDDUSB
VBAT

多种低功耗模式

灵活的功耗控制

- 高效运行
- 8 个低功耗模式, 几个子模式
- 高灵活性



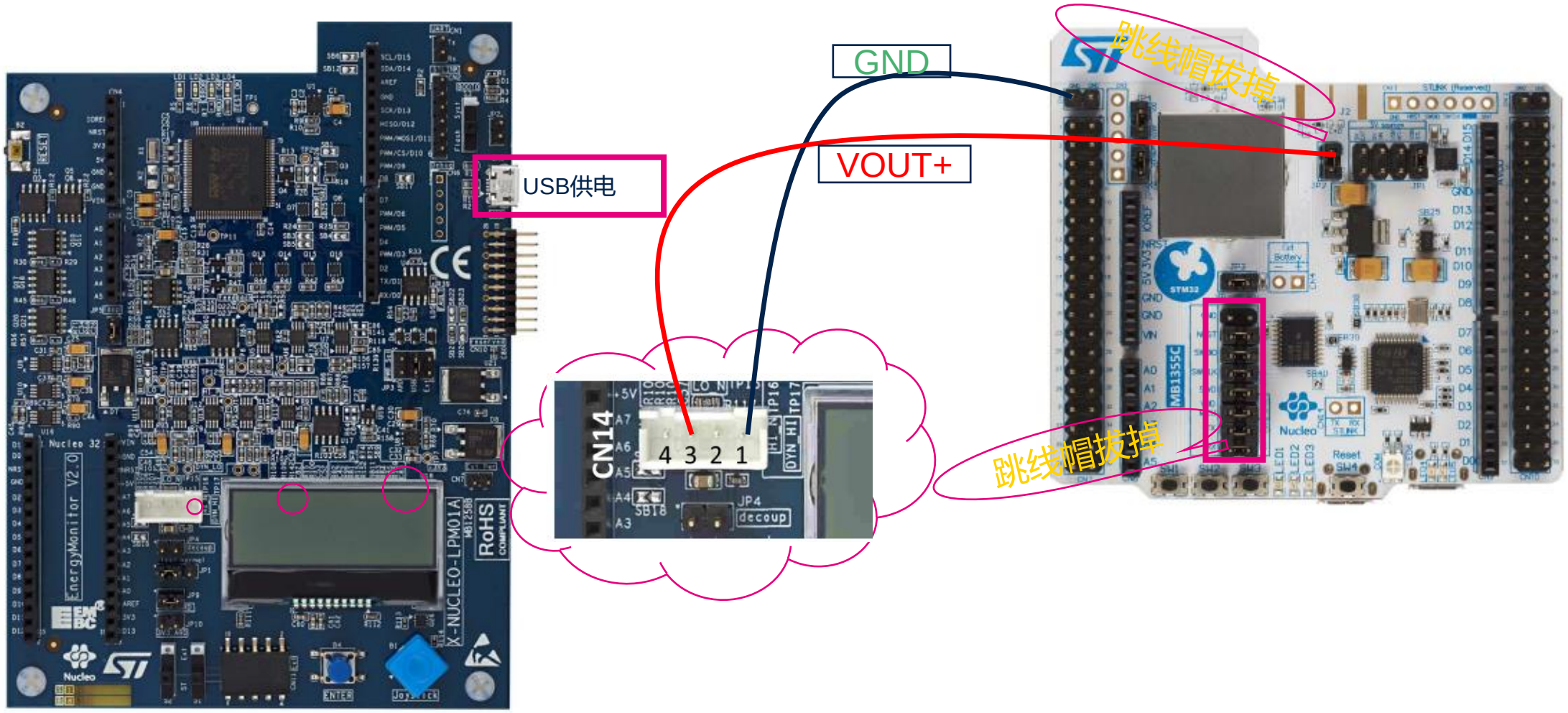
优势

CoreMark® An EEMBC Benchmark

- 高性能
  - ➔ CoreMark 分数 = 219
- 出色的电源效率
  - ➔ ULPBbench 分数 = 303
- 当RF和SMPS同时打开时
  - ➔ M4 核心消耗 = 53µA/Mhz

life.augmented

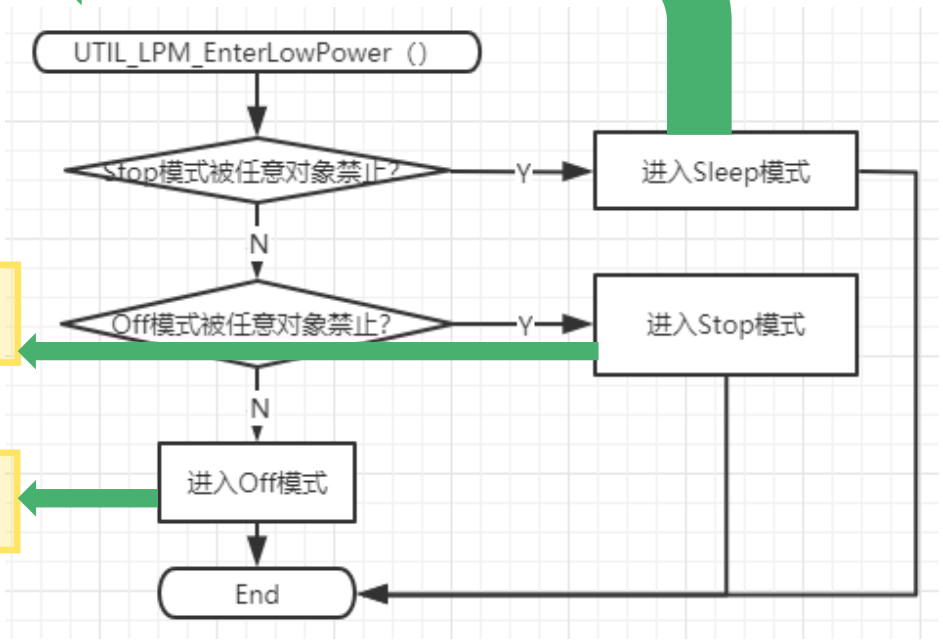
硬件连接


USB供电
GND
VOUT+
跳线帽拔掉
跳线帽拔掉
CN14
4 3 2 1
JP4
decoup
SB18
X-NUCLEO-LPM01A
EnergyMonitor V2.0
RoHS
Nucleo
MB1355C
Reset
SW1

低功耗管理器

唤醒时间		
	RUN (Range1) at 64 MHz	117 (73) µA / MHz**
	RUN (Range2) at 16 MHz	109 µA / MHz**
	LPRUN at 2 MHz	103 µA / MHz**
9 cycles	SLEEP at 64 MHz	41 µA / MHz
9 cycles	LPSLEEP at 2 MHz	45 µA / MHz
1.7 µs	STOP 0 (full retention)	100 µA
4 µs (19 µs)	STOP 1 (full retention)	9.2 µA / 9.6 µA*
5 µs (20 µs)	STOP 2 (full retention)	1.8 µA / 2.1 µA*
14 µs (25 µs)	STANDBY + 32 KB RAM	320 nA / 650 nA*
14 µs (25 µs)	STANDBY	110 nA / 440 nA*
50 µs	SHUTDOWN	30 nA / 315 nA*
	VBAT	2 nA / 300 nA*

RF Operation



() Typ. value for SMPS mode

Typ @ V_{DD} = 1.8 V @ 25 °C

** : with RTC*

*** : from SRAM1*

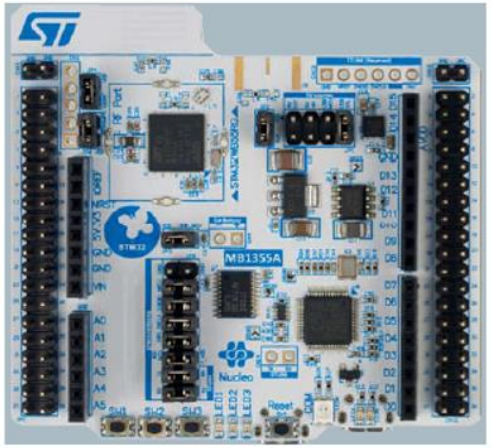
life.augmented

低功耗模式

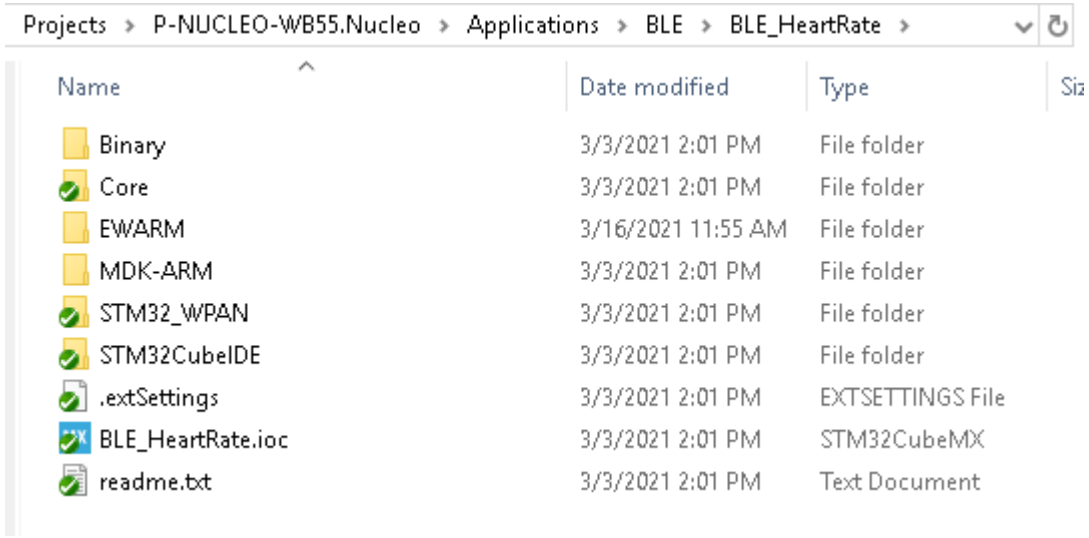
低功耗模式	唤醒源	唤醒后系统时钟	时钟影响
Sleep	中断/事件	与进入Sleep模式之前一样	CPU时钟关闭 对其他时钟或模拟时钟源无影响
LowPower-Run	清除LPR位	与进入LP-Run模式之前一样	无
LowPower-Sleep	中断/事件	与进入LP-Sleep模式之前一样	CPU时钟关闭 对其他时钟或模拟时钟源无影响
Stop0/Stop1/Stop2	任意EXTI， 特定外设事件	STOPWUCK=1，HSI16 STOPWUCK=0，MSI	所有时钟关闭，除了LSI和LSE
Standby	WKUP引脚,RTC事件,LSECSS，NRST引脚复位,IWDG重置	HSI16	
Shutdown	WKUP引脚,RTC事件,NRST引脚复位	MSI 4 MHz	所有时钟关闭，除了LSE

life.augmented

HeartRate Demo：测试不同的功耗模式 --准备工作



编译和烧录



HeartRate Demo：测试不同的功耗模式 --代码修改

app_ble.c | app_ble.c (8a753d4) ↔ app_ble.c

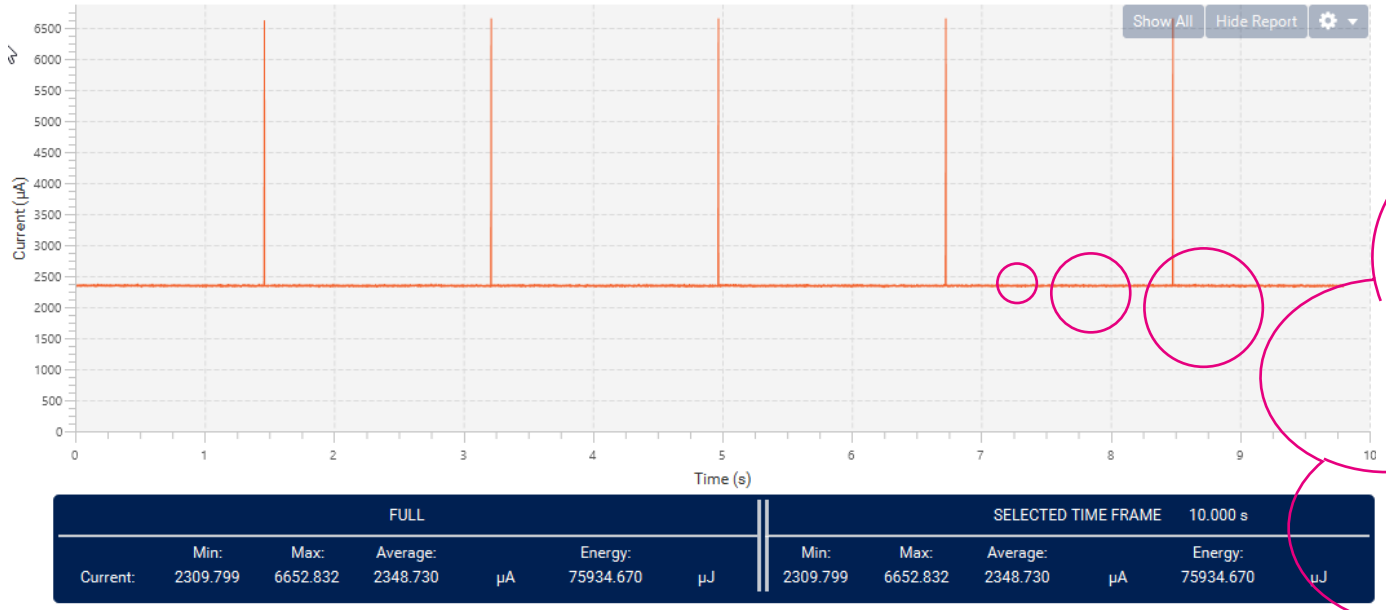
Projects > P-NUCLEO-WB55.Nucleo > Applications > BLE > BLE_HeartRate > STM32_WPAN > App > app_ble.c

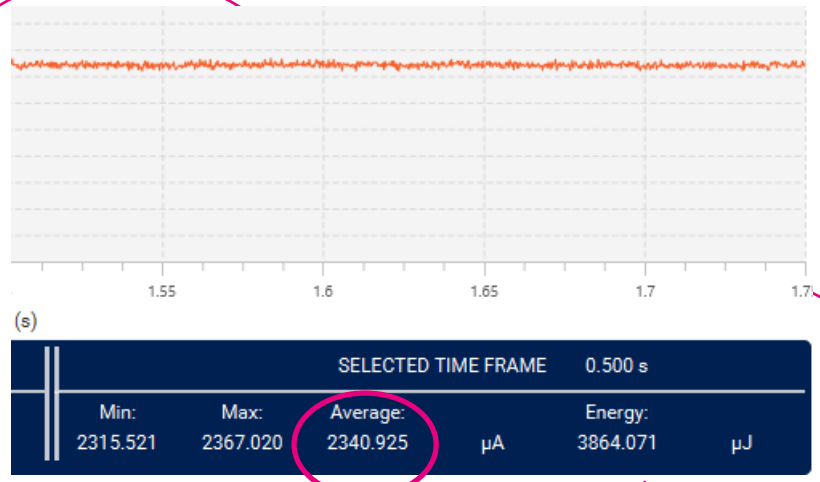
```
601  /* USER CODE BEGIN FD*/
602  void APP_BLE_Key_Button1_Action(void)
603  {
604    tBleStatus ret = BLE_STATUS_INVALID_PARAMS;
605    ret = aci_gap_clear_security_db();
606        if (ret == BLE_STATUS_SUCCESS)
607        {
608          APP_DBG_MSG("Successfully aci_gap_clear_security_db()\n");
609        }
610        else
611        {
612          APP_DBG_MSG("aci_gap_clear_security_db() Failed , result: %d \n", ret);
613        }
614  }
```

```
601  /* USER CODE BEGIN FD*/
602  void APP_BLE_Key_Button1_Action(void)
603  {
604    tBleStatus ret = BLE_STATUS_INVALID_PARAMS;
605    ret = aci_gap_clear_security_db();
606        if (ret == BLE_STATUS_SUCCESS)
607        {
608          APP_DBG_MSG("Successfully aci_gap_clear_security_db()\n");
609        }
610        else
611        {
612          APP_DBG_MSG("aci_gap_clear_security_db() Failed , result: %d \n", ret);
613        }
614+
615+   static enum{
616+     STOP_MODE=0,      You, a few seconds ago • Uncommitted changes
617+     SLEEP_MODE=1
618+   }lpm_flag;
619+   if(lpm_flag==STOP_MODE){
620+     UTIL_LPM_SetStopMode(1 << CFG_LPM_APP, UTIL_LPM_ENABLE);
621+     UTIL_LPM_SetOffMode(1 << CFG_LPM_APP, UTIL_LPM_DISABLE);
622+     lpm_flag = SLEEP_MODE;
623+   }else{
624+     UTIL_LPM_SetStopMode(1 << CFG_LPM_APP, UTIL_LPM_DISABLE);
625+     UTIL_LPM_SetOffMode(1 << CFG_LPM_APP, UTIL_LPM_DISABLE);
626+     lpm_flag = STOP_MODE;
627+   }
628  }
```

life.augmented

测量结果：SLEEP模式





SLEEP at 64 MHz **41 µA / MHz** 41 x 64MHz = 2624uA

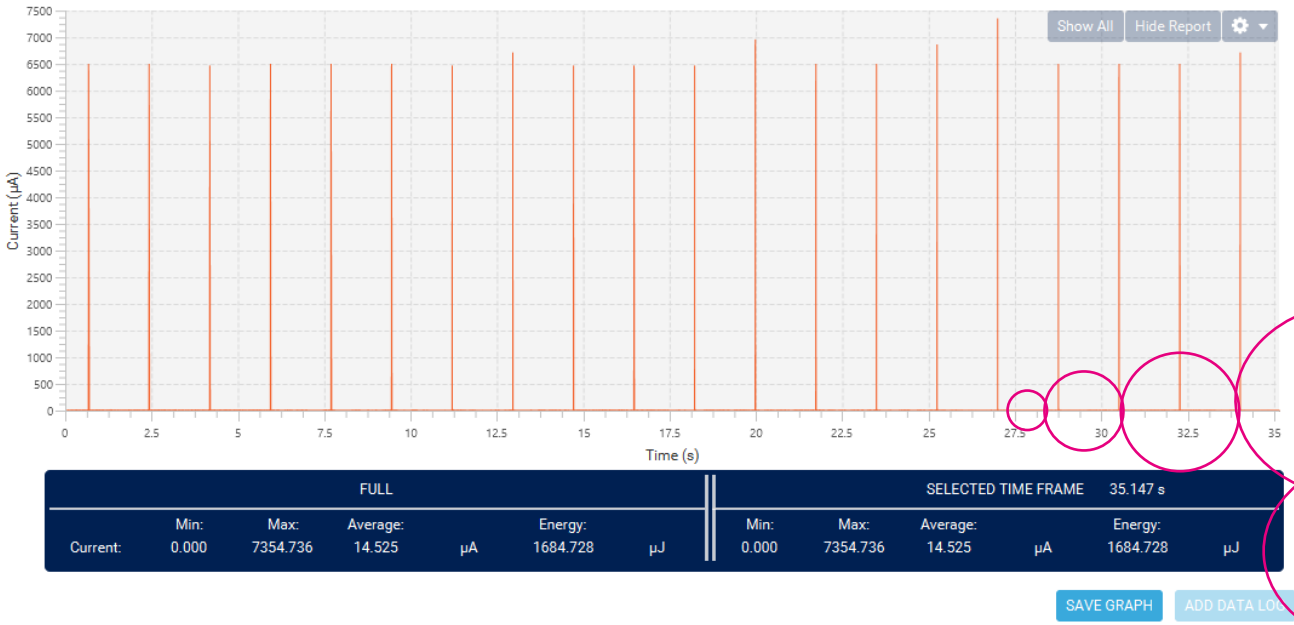
Typ @ V_{DD} = 1.8 V @ 25 °C

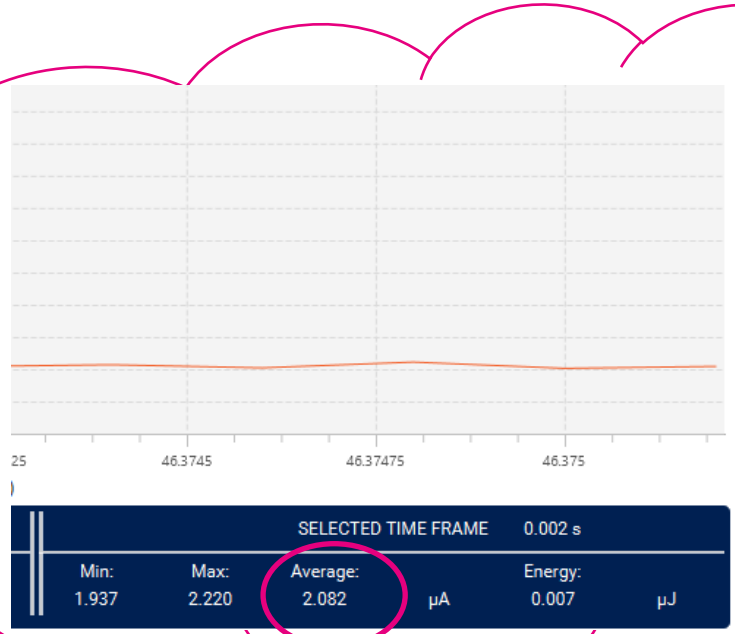
** : with RTC*

*** : from SRAM1*

life.augmented

测量结果：STOP2模式





STOP 2 (full retention) **1.8 µA / 2.1 µA***

Typ @ V_{DD} = 1.8 V @ 25 °C

** : with RTC*

*** : from SRAM1*

life.augmented

3.SMPS

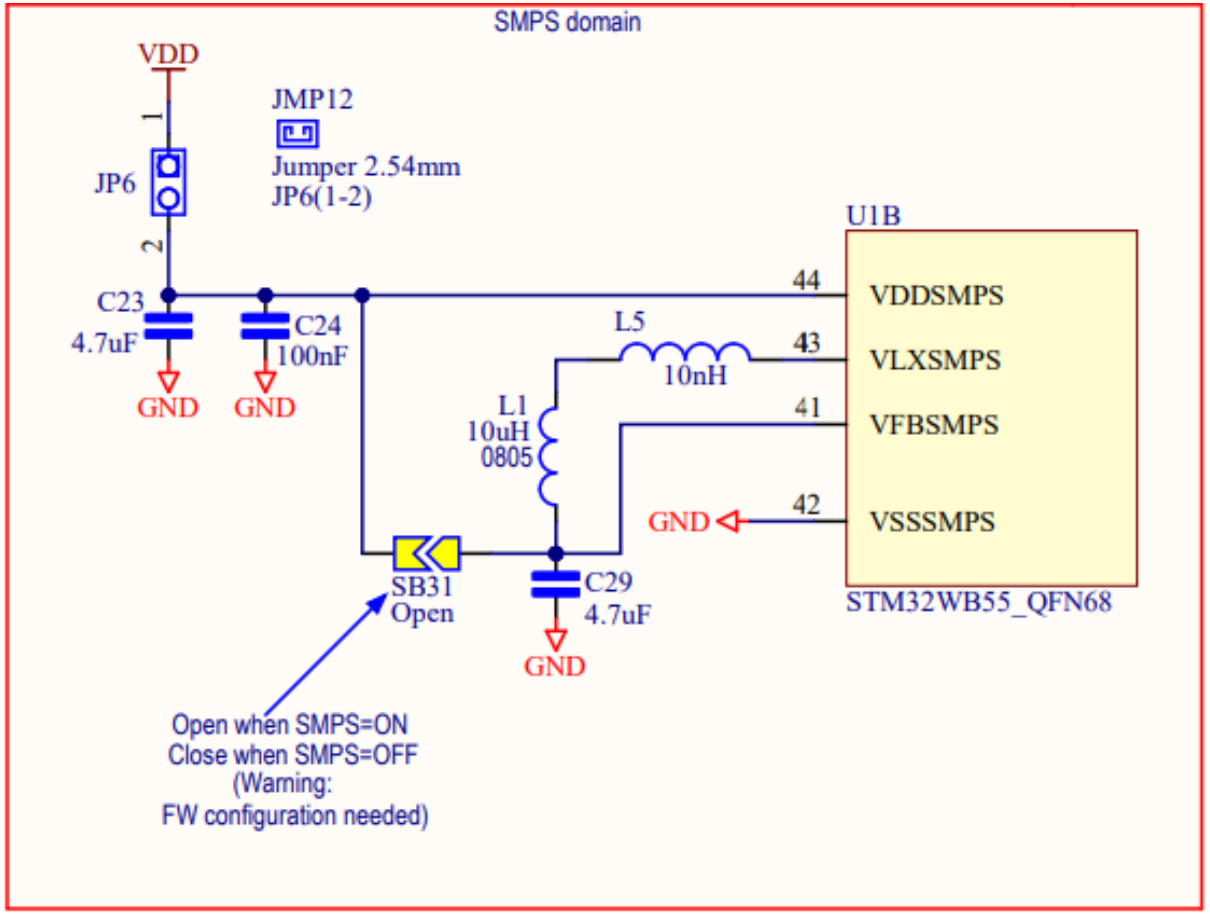
life.augmented

BLE_HeartRate Demo：对比测试SMPS和LDO --硬件修改

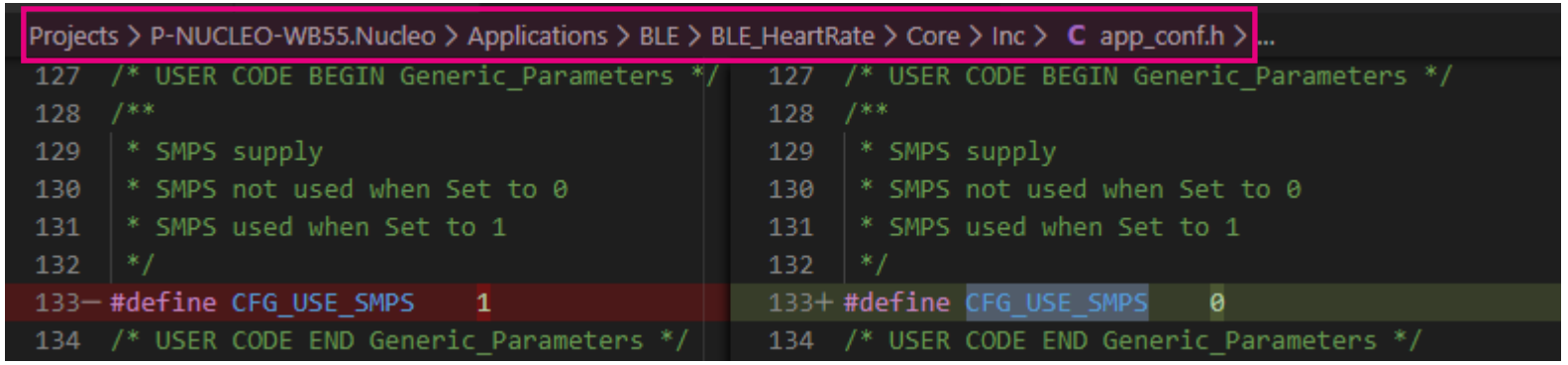
UM2435：

修改硬件打开或关闭SMPS

SB31打开：SMPS打开
SB31关闭：SMPS关闭



BLE_HeartRate Demo：对比测试SMPS和LDO --代码修改



life.augmented

使用SMPS：STOP2模式

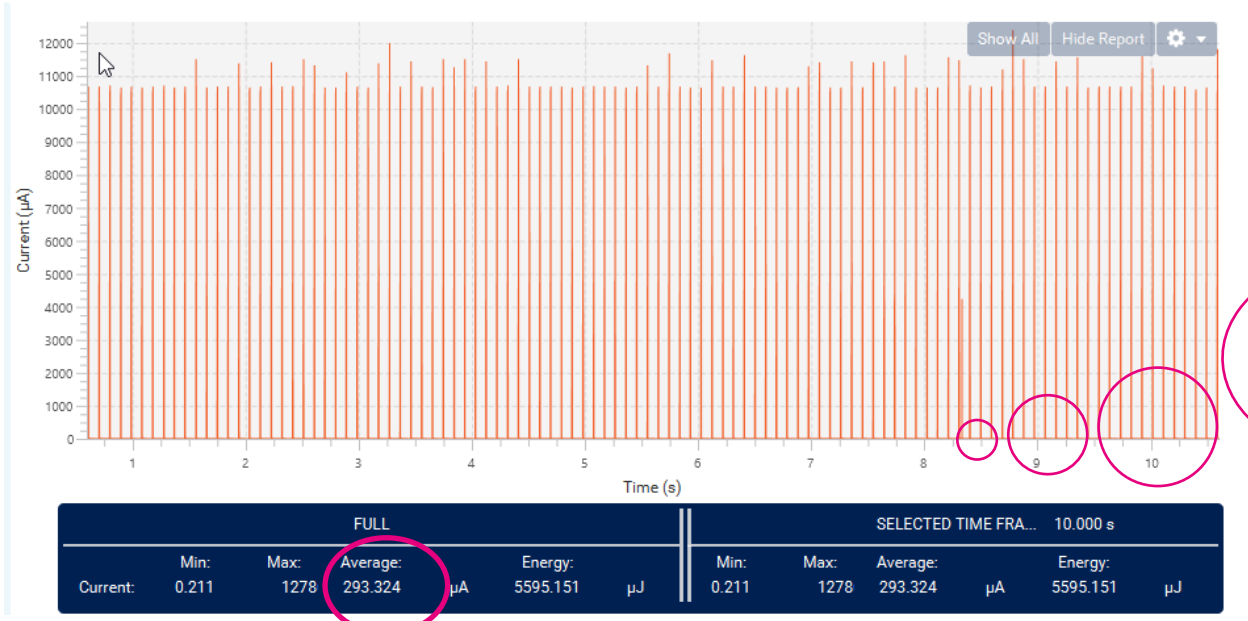
STOP 2 (full retention) **1.8 µA / 2.1 µA***

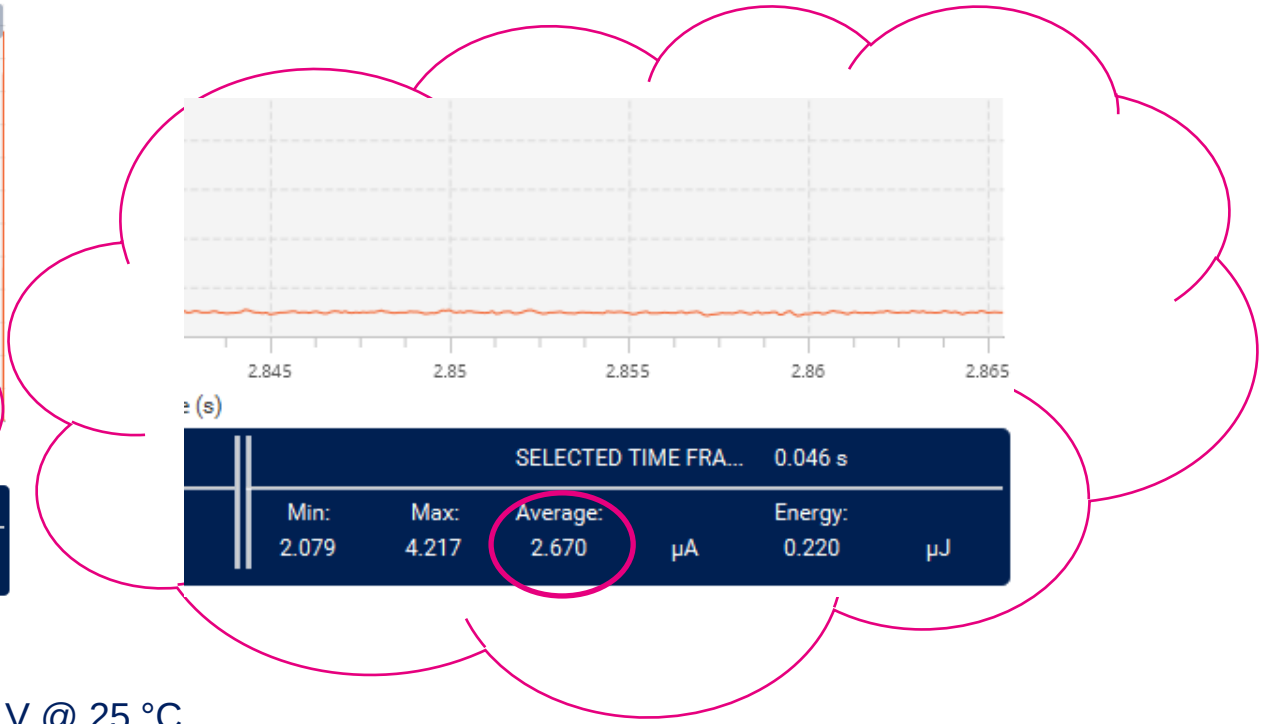
Typ @ V_{DD} = 1.8 V @ 25 °C

** : with RTC*

*** : from SRAM1*

使用LDO：STOP2模式





STOP 2 (full retention) **1.8 µA / 2.1 µA***

Typ @ V_{DD} = 1.8 V @ 25 °C

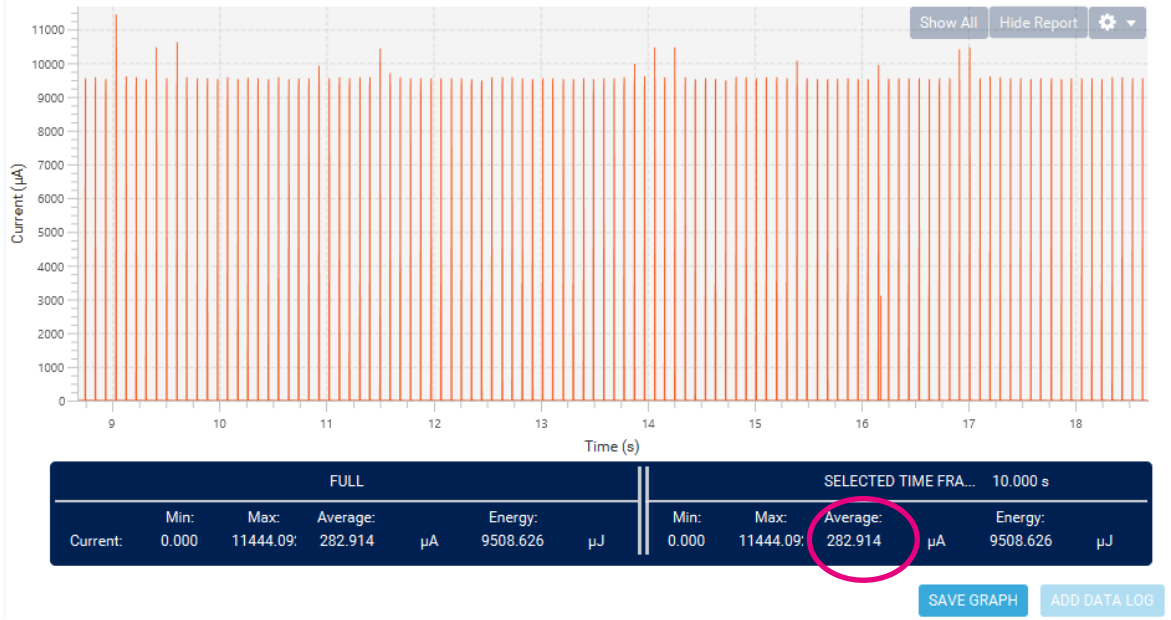
** : with RTC*

*** : from SRAM1*

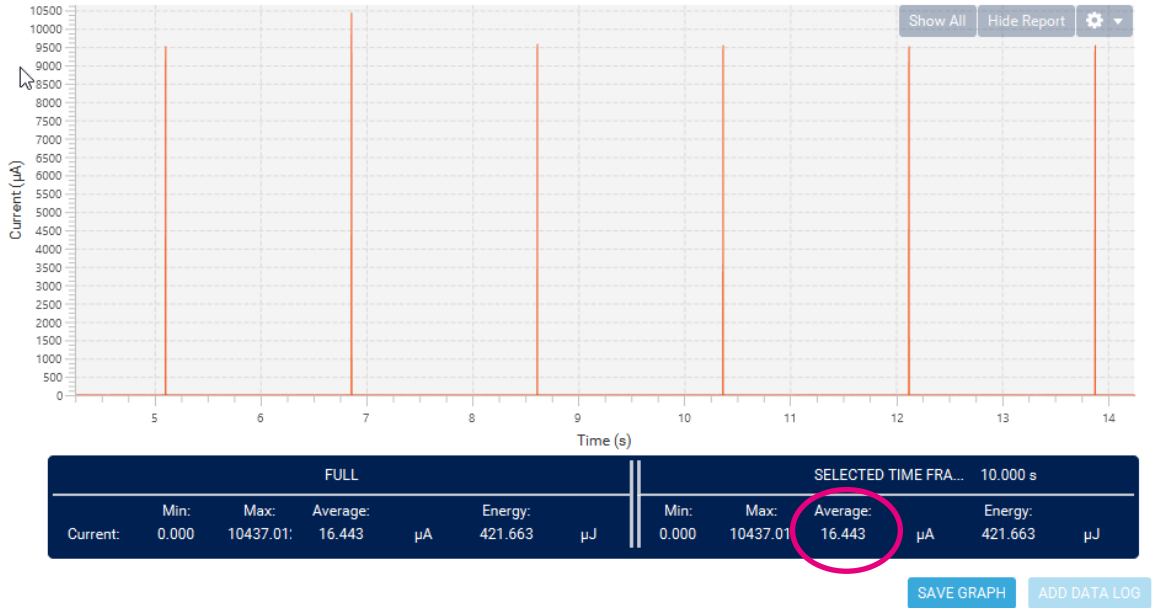
life.augmented

4.广播和连接参数

HeartRate Demo:测试不同的广播参数 --测量结果



广播间隔:
80ms~100ms



广播间隔:
1s~2.5s

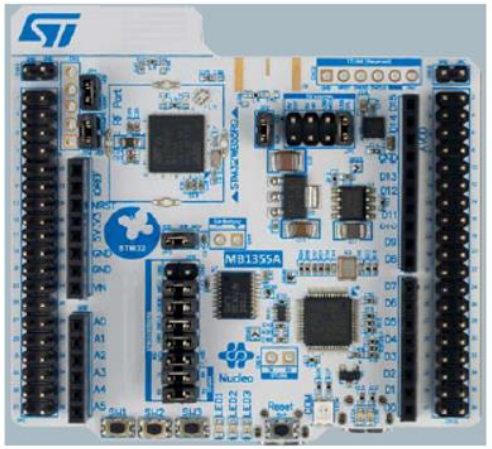
life.augmented

P2P Demo：测试不同连接参数 --准备工作

编译和烧录

Projects › P-NUCLEO-WB55.USBDongle › Applications › BLE › BLE_p2pServer ›

Name	Date modified	Type	Size
Binary	3/3/2021 2:02 PM	File folder	
Core	3/3/2021 2:02 PM	File folder	
EWARM	3/3/2021 2:02 PM	File folder	
MDK-ARM	3/3/2021 2:02 PM	File folder	
STM32_WPAN	3/3/2021 2:02 PM	File folder	
STM32CubeIDE	3/3/2021 2:02 PM	File folder	
readme.txt	3/3/2021 2:02 PM	Text Document	8 KB



编译和烧录

Projects › P-NUCLEO-WB55.Nucleo › Applications › BLE › BLE_p2pClient ›

Name	Date modified	Type	Si
Binary	3/3/2021 2:01 PM	File folder	
Core	3/3/2021 2:01 PM	File folder	
EWARM	3/3/2021 2:01 PM	File folder	
MDK-ARM	3/3/2021 2:01 PM	File folder	
STM32_WPAN	3/3/2021 2:01 PM	File folder	
STM32CubeIDE	3/3/2021 2:01 PM	File folder	
.extSettings	3/3/2021 2:01 PM	EXTSETTINGS File	
BLE_p2pClient.ioc	3/3/2021 2:01 PM	STM32CubeMX	
readme.txt	3/3/2021 2:01 PM	Text Document	

life.augmented

P2P Demo：测试不同连接参数 --修改代码

Projects > P-NUCLEO-WB55.Nucleo > Applications > BLE > BLE_p2pClient > Core > Inc > app_conf.h > PUSH_BUTTON_SW3_EXTI_IRQHandler

```
462  #define CFG_BUTTON_SUPPORTED      1
463
464- #define PUSH_BUTTON_SW1_EXTI_IRQHandler      EXTI4_IRQHandler
```

```
462  #define CFG_BUTTON_SUPPORTED      1
463
464+ #define PUSH_BUTTON_SW1_EXTI_IRQHandler      EXTI4_IRQHandler
465+ #define PUSH_BUTTON_SW2_EXTI_IRQHandler      EXTI0_IRQHandler
466+ #define PUSH_BUTTON_SW3_EXTI_IRQHandler      EXTI1_IRQHandler
```

Projects > P-NUCLEO-WB55.Nucleo > Applications > BLE > BLE_p2pClient > Core > Src > stm32wbxx_it.c > RTC_WKUP_IRQHandler(void)

```
279   * @retval None
280   */
281  void PUSH_BUTTON_SW2_EXTI_IRQHandler(void)
282  {
283-
284  }
```

```
279   * @retval None
280   */
281  void PUSH_BUTTON_SW2_EXTI_IRQHandler(void)
282  {
283+   HAL_GPIO_EXTI_IRQHandler(BUTTON_SW2_PIN);
284  }
```

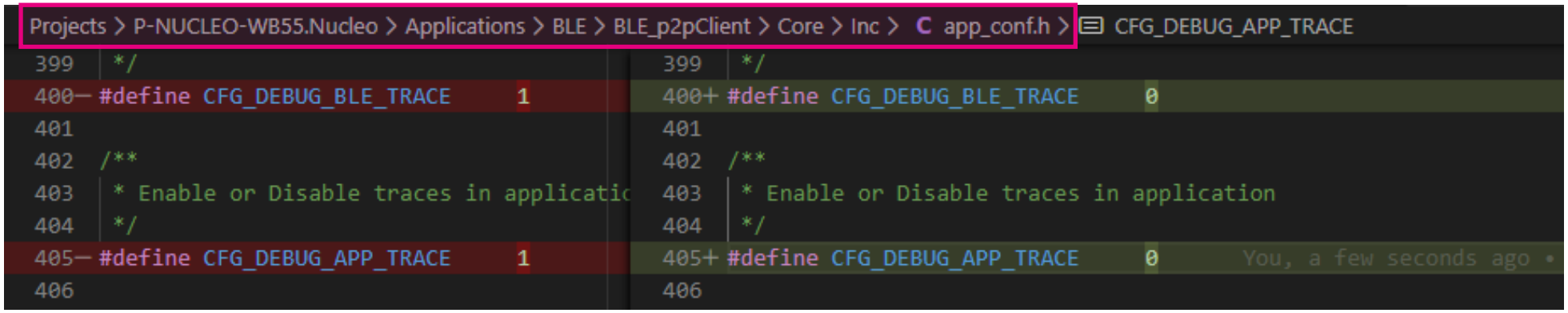
Projects > P-NUCLEO-WB55.Nucleo > Applications > BLE > BLE_p2pClient > Core > Src > app_entry.c > ...

```
225  static void Button_Init( void )
226  {
227  #if (CFG_BUTTON_SUPPORTED == 1)
228    /**
229     * Button Initialization
230     */
231
232    BSP_PB_Init(BUTTON_SW1, BUTTON_MODE_EXTI);
233  #endif
```

```
225  static void Button_Init( void )
226  {
227  #if (CFG_BUTTON_SUPPORTED == 1)
228    /**
229     * Button Initialization
230     */
231
232    BSP_PB_Init(BUTTON_SW1, BUTTON_MODE_EXTI);
233+   BSP_PB_Init(BUTTON_SW2, BUTTON_MODE_EXTI);
234+   BSP_PB_Init(BUTTON_SW3, BUTTON_MODE_EXTI);
235  #endif
```

life.augmented

P2P Demo：测试不同连接参数 --修改代码



P2P Demo：测试不同连接参数 --修改代码

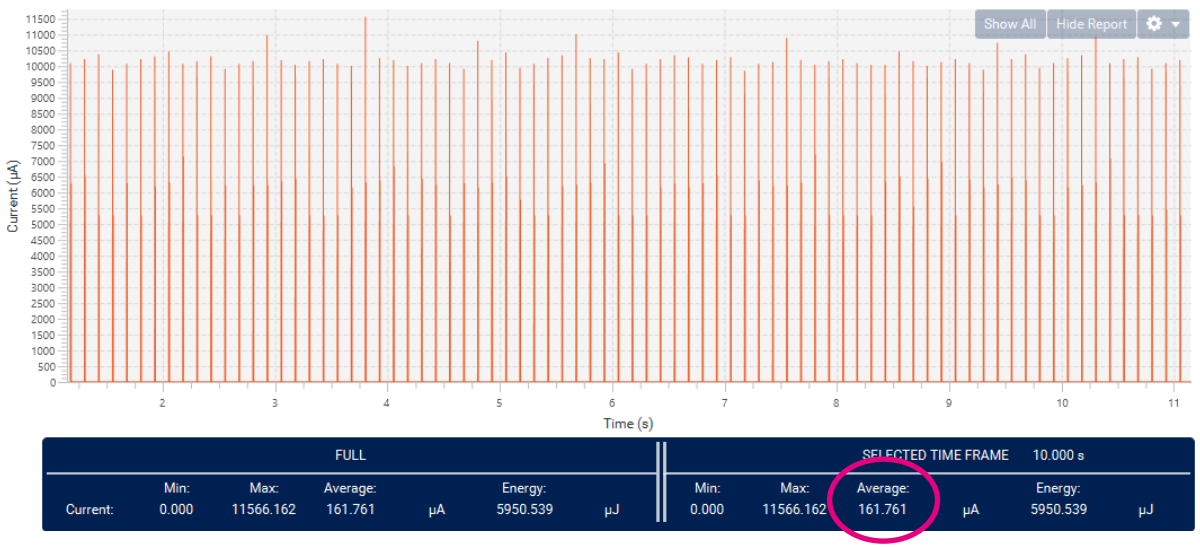
Projects > P-NUCLEO-WB55.Nucleo > Applications > BLE > BLE_p2pClient > STM32_WPAN > App > C app_ble.c > ...

```
657  void APP_BLE_Key_Button2_Action(void)
658  {
659  }
```

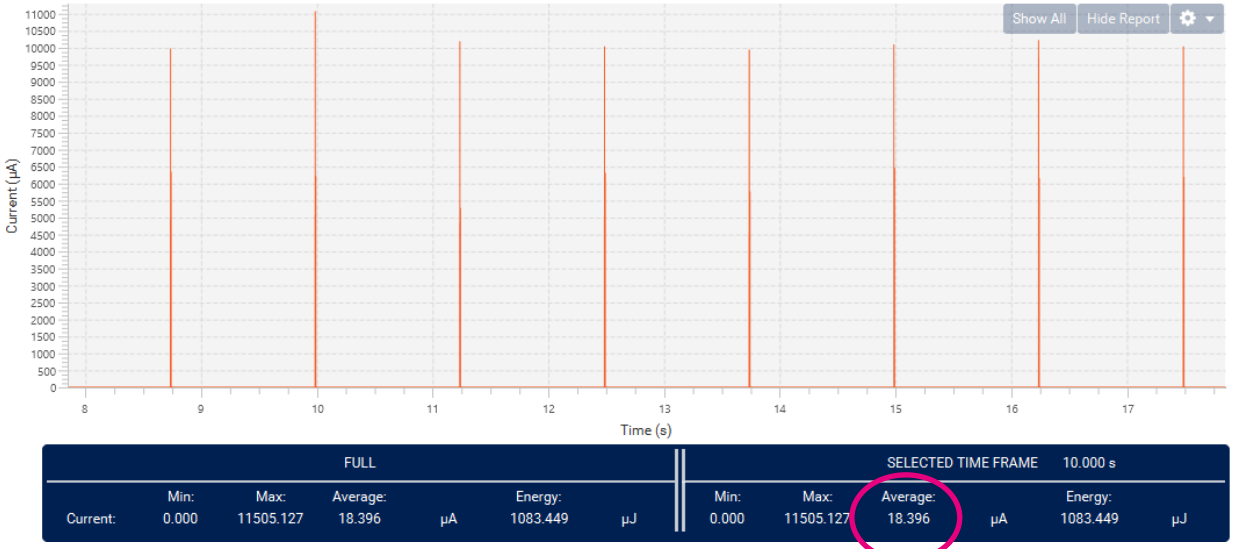
```
657  void APP_BLE_Key_Button2_Action(void)
658  {
659+   tBleStatus ret = BLE_STATUS_INVALID_PARAMS;
660+   static enum{
661+     CONN_FAST=0,
662+     CONN_SLOW=1
663+   }conn_mode;
664+   if(conn_mode == CONN_FAST){
665+     ret = aci_gap_start_connection_update(BleApplicationContext.BleApplicationContext_legacy.connectionHandle,
666+                                           0x64, 0x64, //Min&Max connect interval:125ms
667+                                           0x00, 0x0c80, //Connect latency:0, Supervision Timeout:32000ms
668+                                           0x00,0x64); //Min&Max CE Length:0~100*0.625ms
669+     conn_mode = CONN_SLOW;
670+   }else{
671+     ret = aci_gap_start_connection_update(BleApplicationContext.BleApplicationContext_legacy.connectionHandle,
672+                                           0x3E8, 0x3E8, //Min&Max connect interval:1250ms
673+                                           0x00, 0x0c80, //Connect latency:0, Supervision Timeout:32000ms
674+                                           0x00,0x64); //Min&Max CE Length:0~100*0.625ms
675+     conn_mode = CONN_FAST;
676+   }
677+   if (ret == BLE_STATUS_SUCCESS)
678+   {
679+     APP_DBG_MSG("Successfully aci_gap_start_connection_update()\n");
680+   }
681+   else
682+   {
683+     APP_DBG_MSG("aci_gap_start_connection_update() Failed , result: %d \n", ret);
684+   }
685  }
```

life.augmented

P2P Demo：测试不同连接参数 --测量结果



连接间隔：
125ms



连接间隔：
1250ms

life.augmented

如何优化BLE通讯速率

1 BLE理论通讯速率

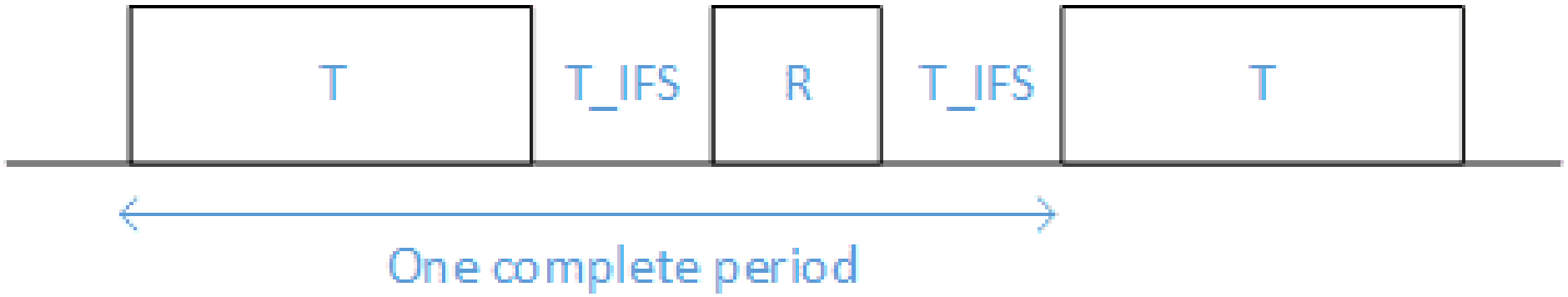
2 BLE通讯速率优化要点

3 测试工程举例

life.augmented

1.BLE理论通讯速率

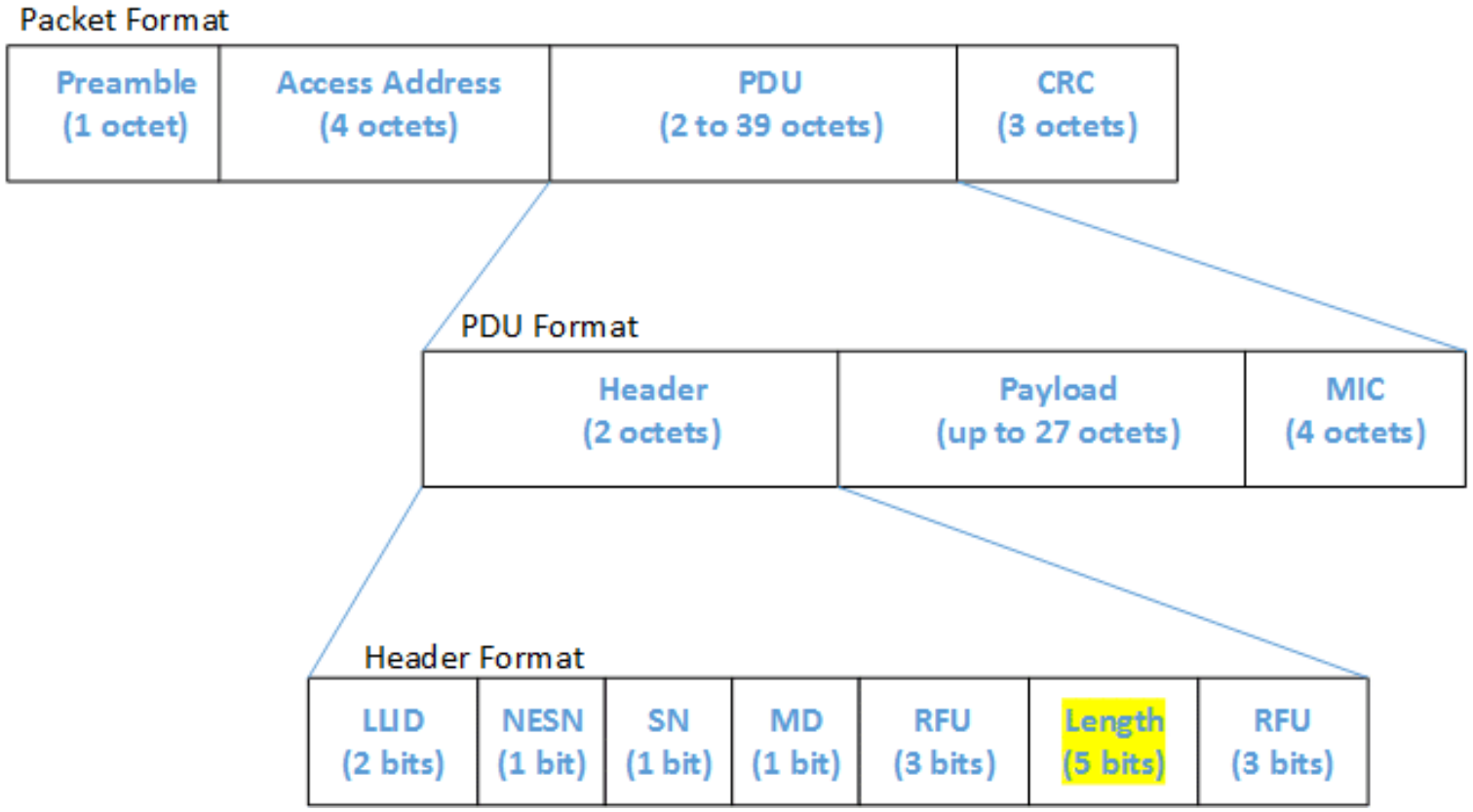
BLE通讯速率



$$\text{吞吐率公式} = \frac{\text{有用的负载}}{\text{一个完整周期}}$$

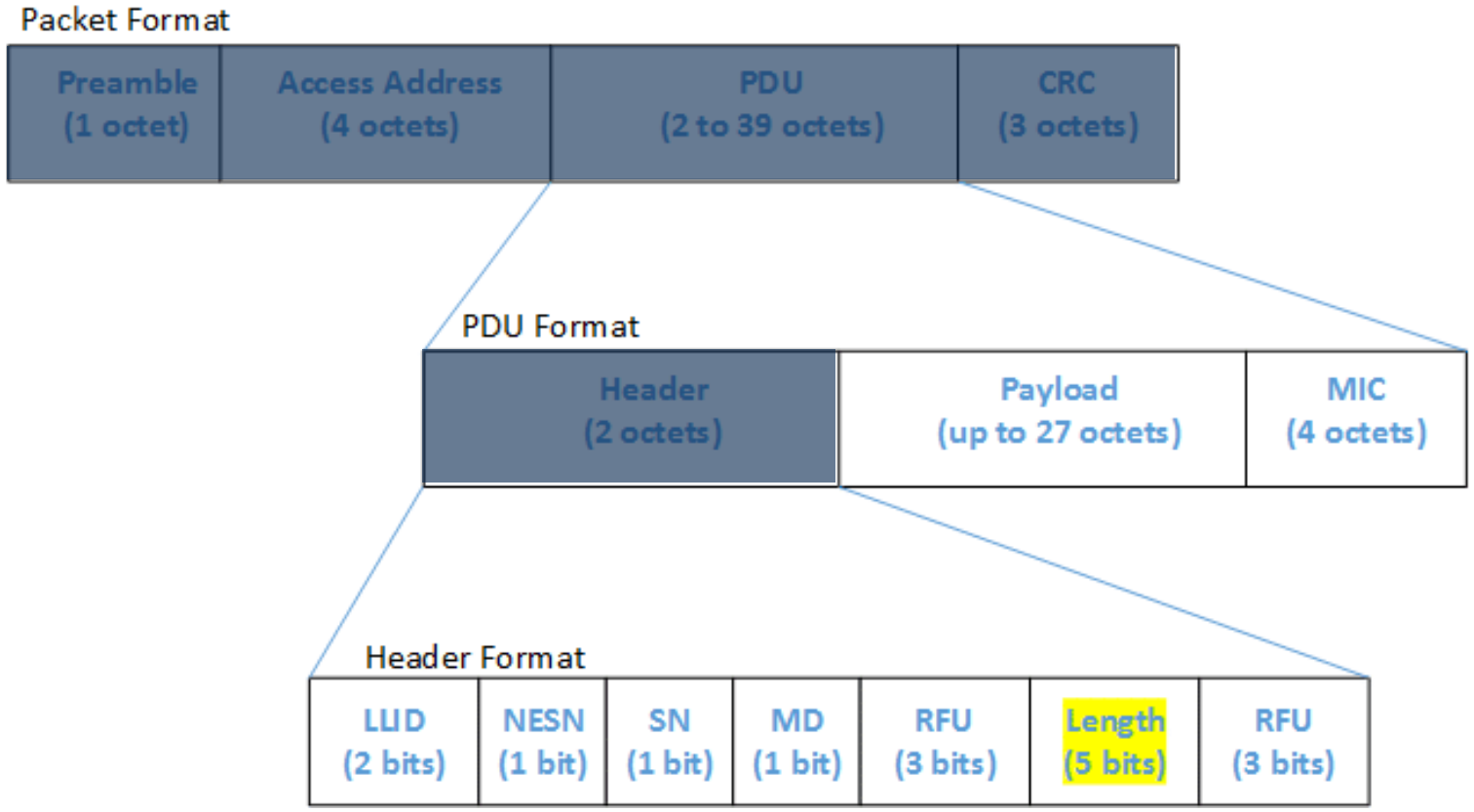
BLE通讯速率 ---BLE4.0/4.1

- 调制速率：
  - 1Mb/s
- Length:
  - 5bit，最大31字节
- 有效载荷：
  - 31 – 4（MIC长度）=27字节



BLE通讯速率 ---BLE4.0/4.1

- R包的空中时间：
  - 空包长度 = 1(前导码)+4（接入地址）+2（PDU头）+3（CRC）= 10字节

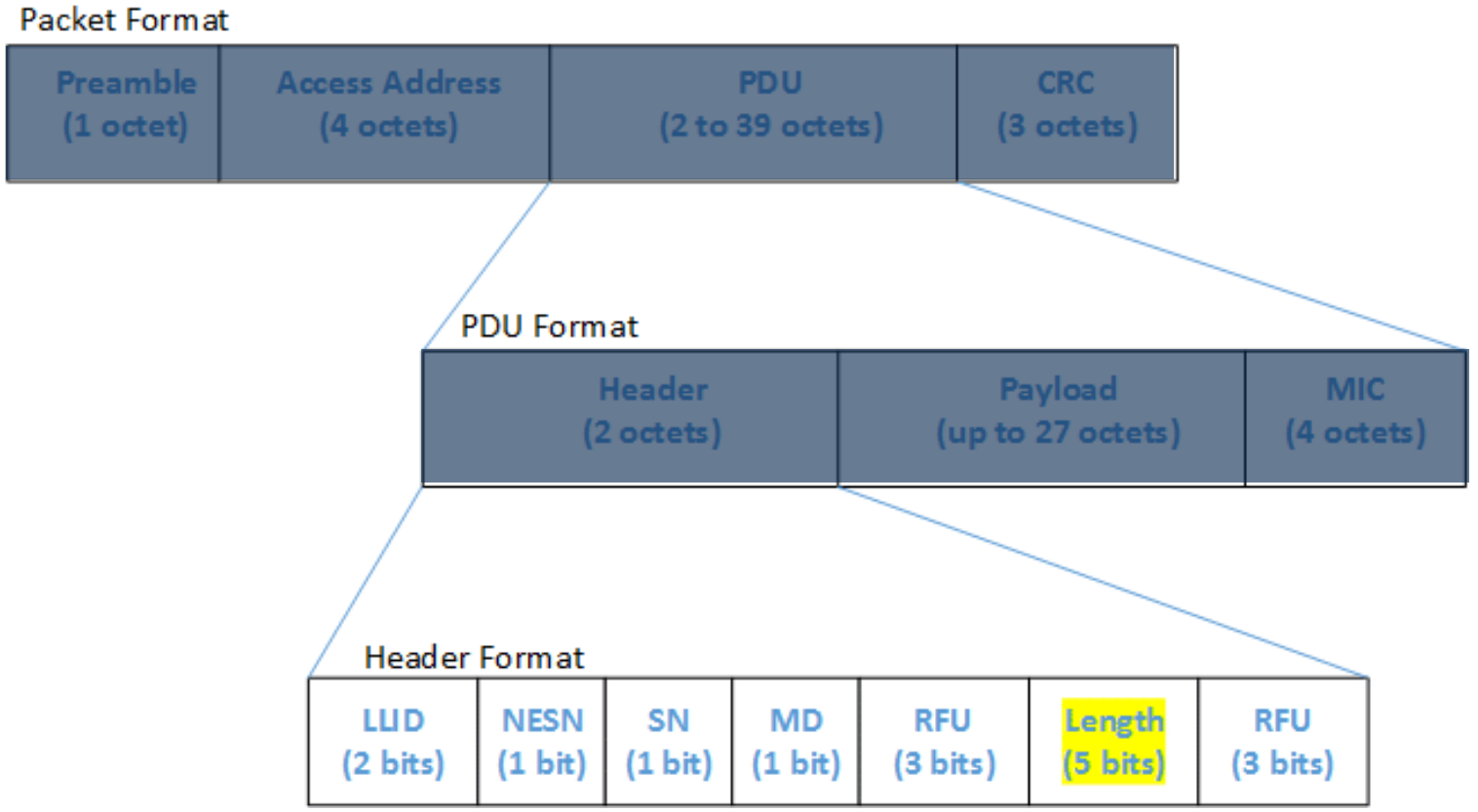


  - 空中时间：

$$\frac{10Bytes \times 8bit}{1M\ b/s} = 80us$$

BLE通讯速率 ---BLE4.0/4.1

- T包的空中时间：
  - 最大长度 = 1（前导码）+4（接入地址）+2（PDU头）+27（载荷）+4（MIC）+3（CRC） = 41字节

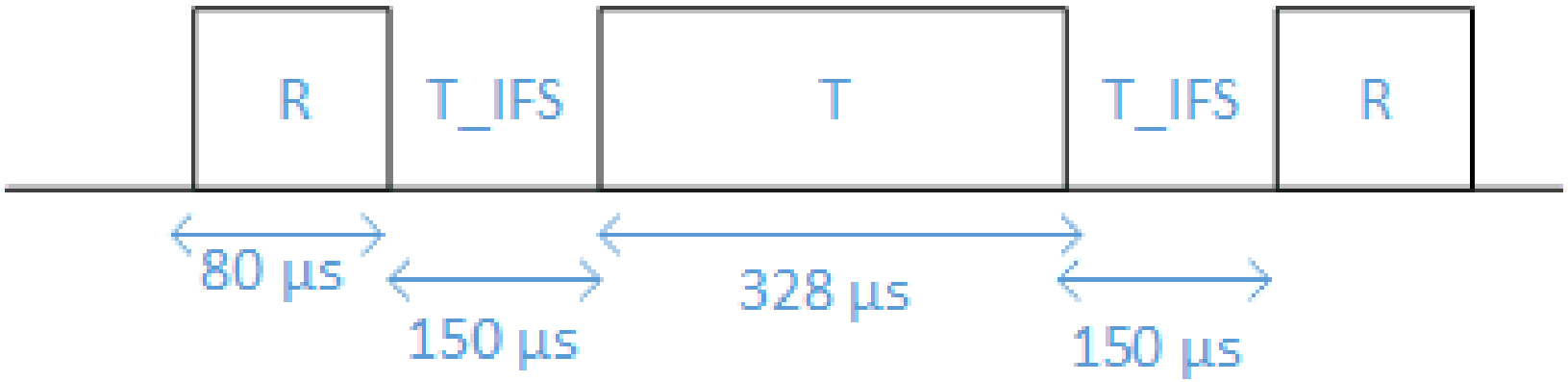


- 空中时间：

$$\frac{41Bytes \times 8bit}{1M\ b/s} = 328us$$

BLE通讯速率 ---BLE4.0/4.1

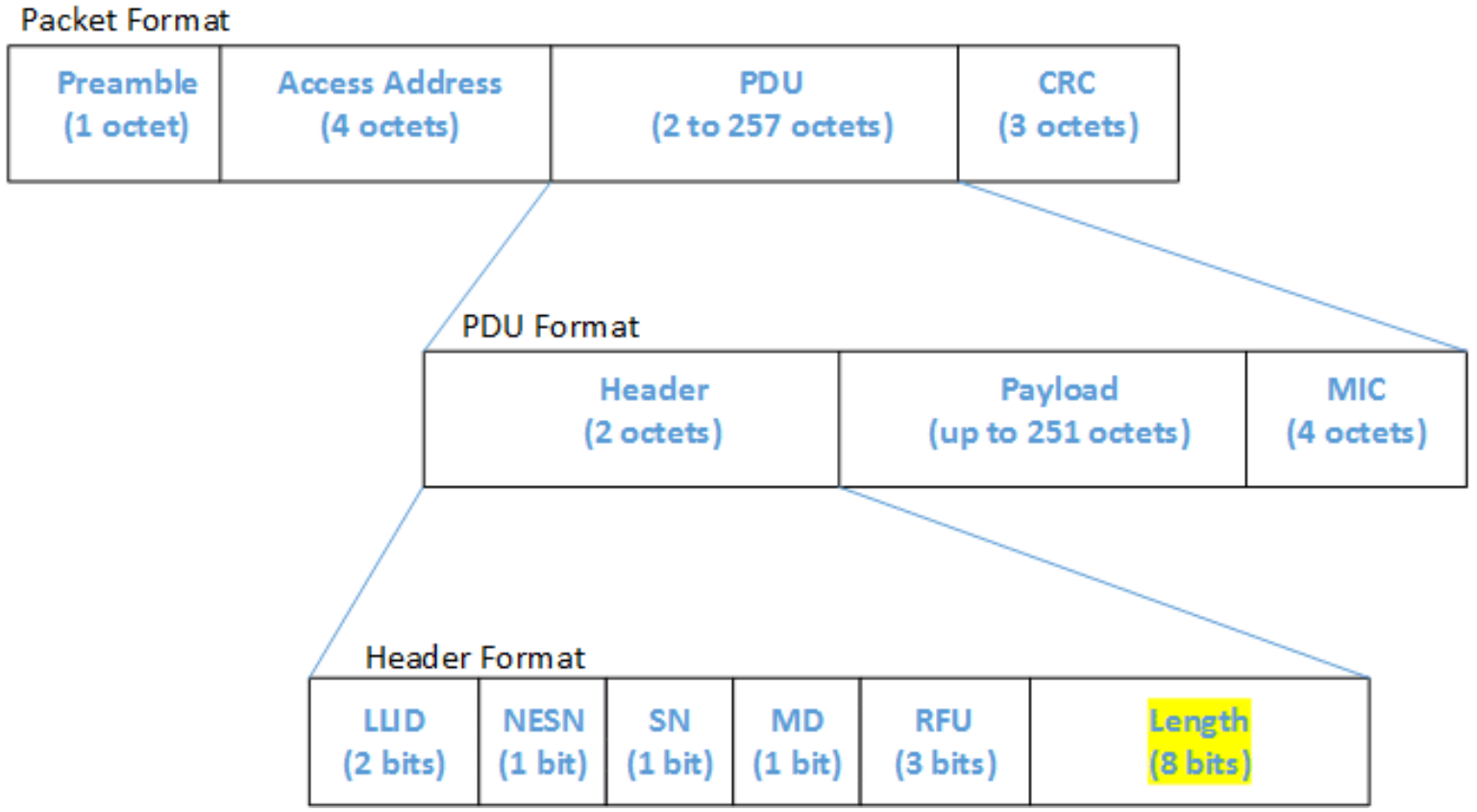
4.0/4.1



吞吐率公式 = $\frac{\text{有用的负载}}{\text{一个完整周期}} = \frac{(27 \times 8)bits}{(80 + 150 + 328 + 150)us} = \frac{216bits}{708us} \approx$ 0.305b/us = 305Kb/s

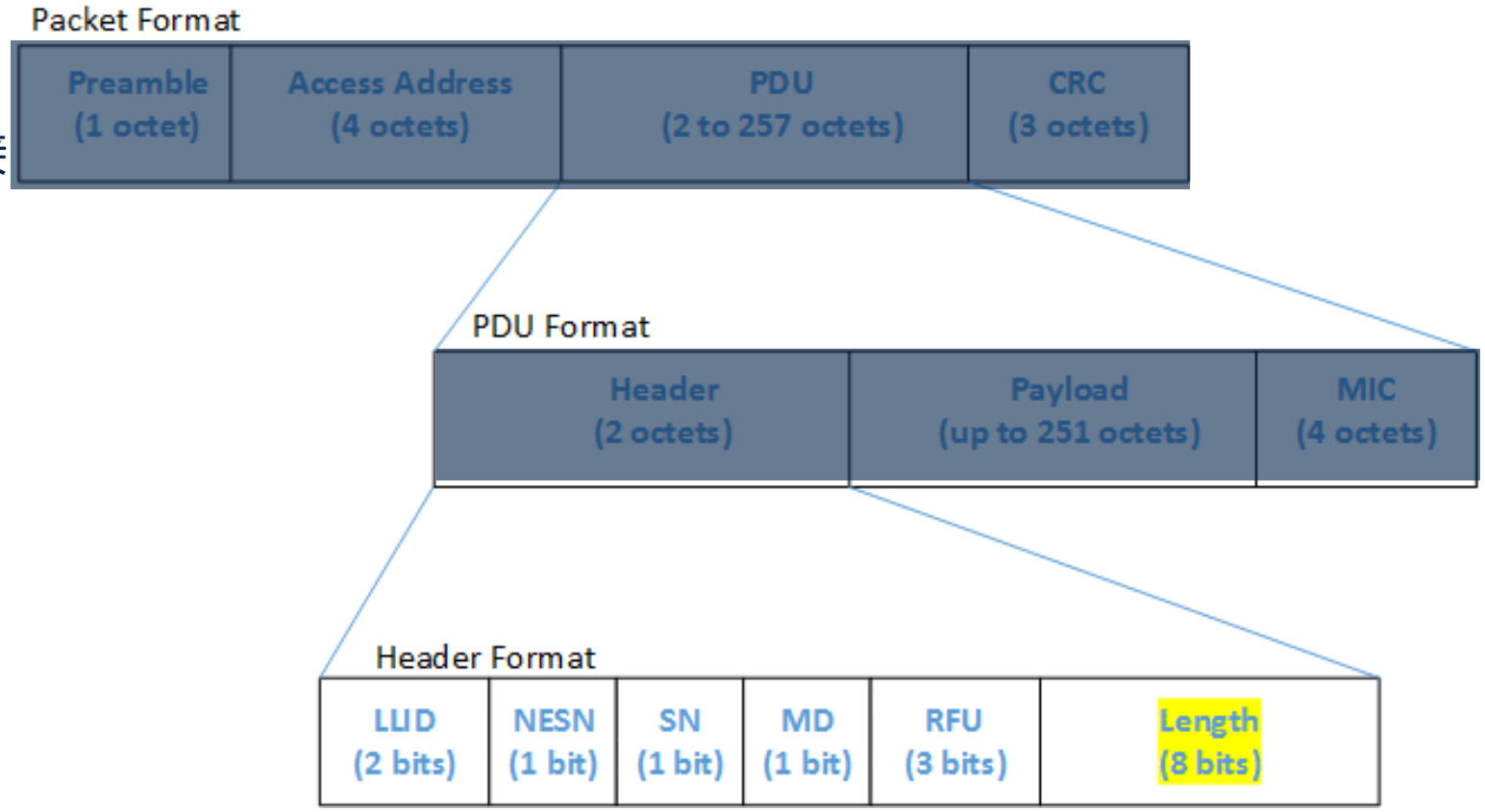
BLE通讯速率 ---BLE4.2

- 调制速率：
  - 1Mb/s
- Length:
  - 8bit，最大255字节
- 有效载荷：
  - 255 – 4（MIC长度）=251字节



BLE通讯速率 ---BLE4.2

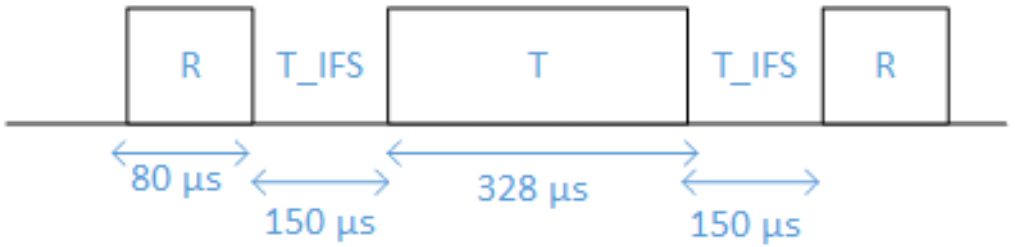
- T包的空中时间：
  - 最大长度 = 1（前导码）+4（接入地址）+2（PDU头）+251（载荷）+4（MIC）+3（CRC）= 265字节



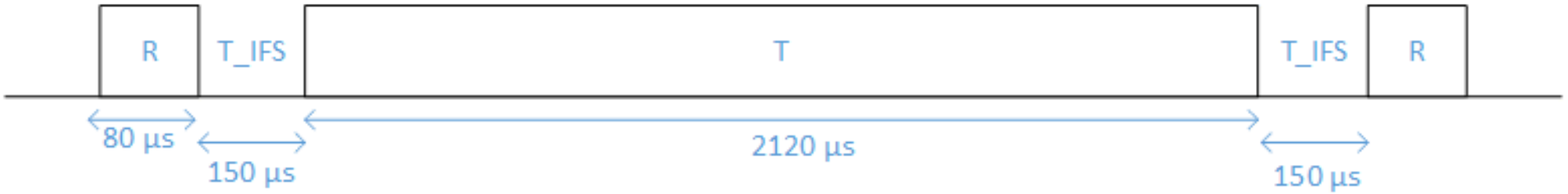
  - 空中时间：$\frac{265Bytes \times 8bit}{1M\ b/s}$ = 2020us

BLE通讯速率 ---BLE4.2

4.0/4.1



4.2

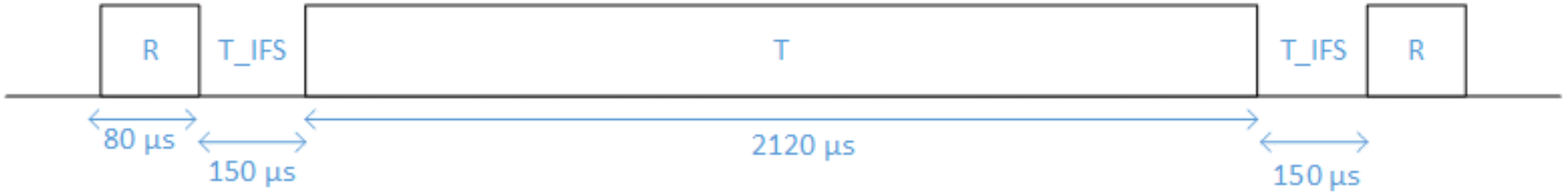


吞吐率公式 = $\dfrac{\text{有用的负载}}{\text{一个完整周期}} = \dfrac{(251 \times 8)bits}{(80 + 150 + 2120 + 150)us} = \dfrac{2008bits}{2500us} \approx$ 0.803b/us = 803Kb/s

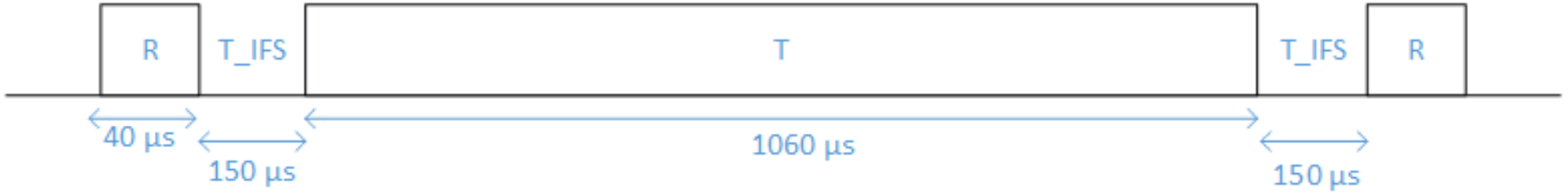
life.augmented

BLE通讯速率
---BLE5.2(2M PHY)

4.2



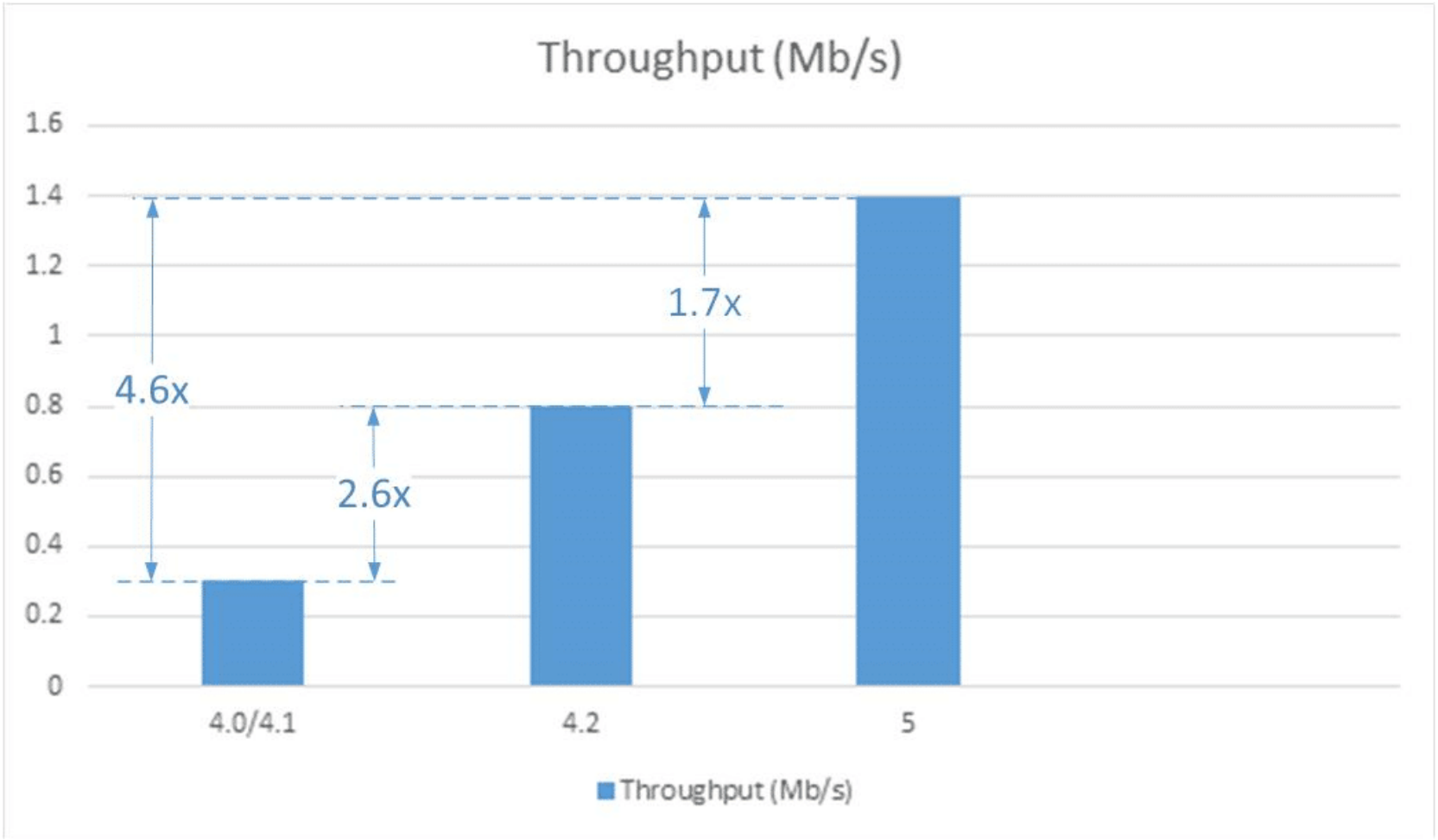
5



吞吐率公式 = $\dfrac{\text{有用的负载}}{\text{一个完整周期}} = \dfrac{(251 \times 8)bits}{(40 + 150 + 1060 + 150)us} = \dfrac{2008bits}{1400us} \approx$ 1.4b/us = 1.4Mb/s

life.augmented

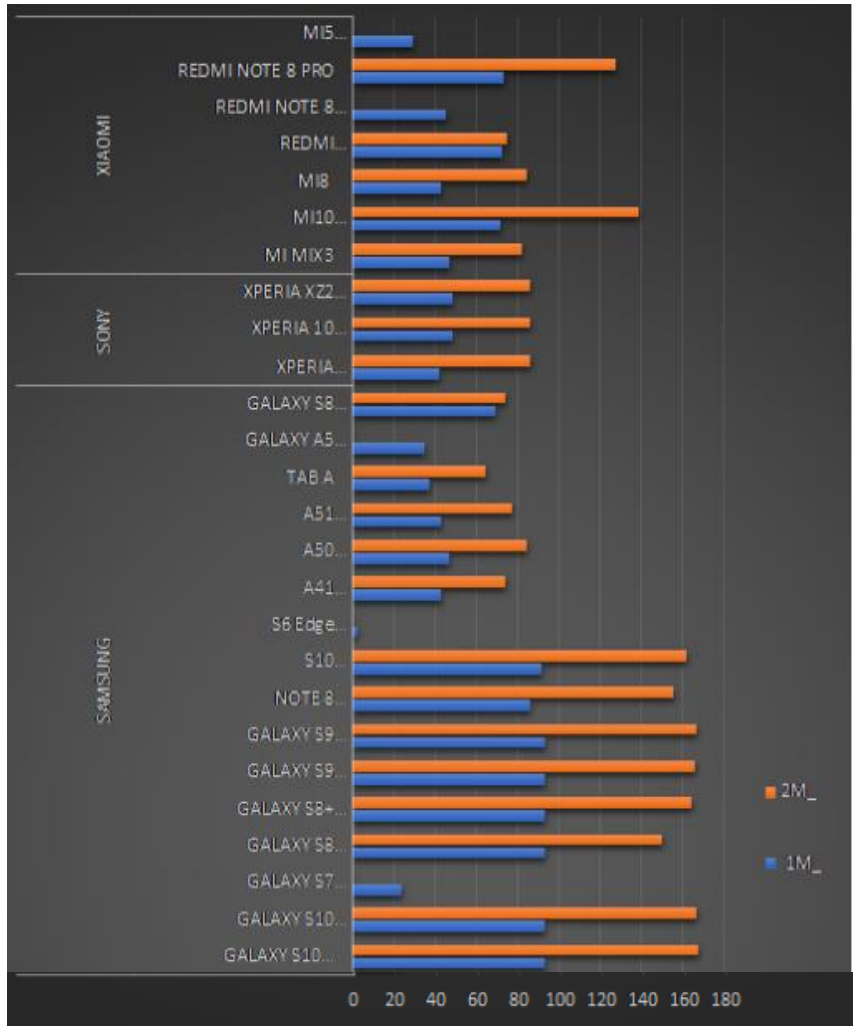
BLE通讯速率总结

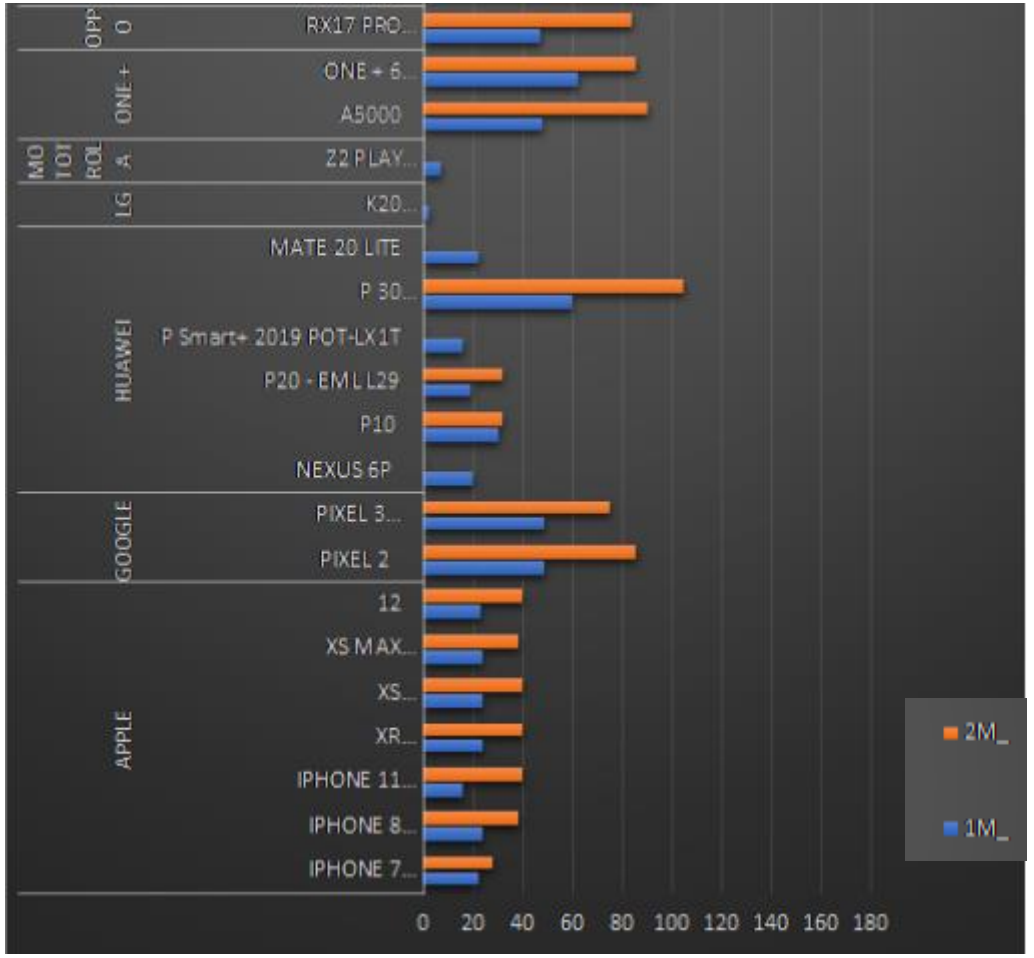


life.augmented

WB与手机通信速率报告

AN5604:STM32WB Series BLE interoperability report





2.BLE通讯速率优化要点

速率优化 ---BLE5.0

- 关键指标：
  - 调制速率：1M/2M
  - PDU有效载荷最大长度：251字节
  - 最长包的一个完整周期：1400us（2M PHY）
- 策略：
  - 使用2M PHY
  - hci_le_set_data_length(con_handle, 251, 265*8)

速率优化
--hci_le_set_data_length()说明

Table 68. **HCI_LE_SET_DATA_LENGTH input parameters**

Parameter	Size	Description	Possible values
Connection_Handle	2	Connection handle for which the command is given.	0x0000 ... 0x0EFF
TxOctets	2	Preferred maximum number of payload octets that the local controller includes in a single link layer packet on this connection	0x001B ... 0x00FB
TxTime	2	Preferred maximum number of microseconds that the local controller must use to transmit a single link layer packet on this connection	0x0148 ... 0x4290

- T包的空中时间：
  - 最大长度 = 1（前导码）+4（接入地址）+2（PDU头）+251（载荷）+4（MIC）+3（CRC） = 265字节
  - 空中时间：

$$\frac{265Bytes \times 8bit}{1M\ b/s} = 2020us$$

  - 连接成功后调用：hci_le_set_data_length(con_handle, 251, 2020)

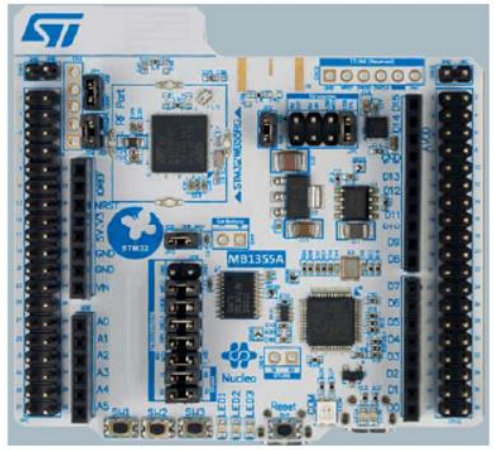
life.augmented

速率优化
--ATT层：特性

- 关键问题：
  - 使用何种方式传输数据
  - 如何确定特性值最大长度?
- 策略：
  - Notify或Write without response方式
  - 不能分段：特性值最大长度=251（空中PDU载荷最大值）- 4（L2CAP层包头） – 3（ATT层包头）= 244
  - 设置MAX_ATT_MTU=251

3. 测试工程举例

测试WB最大吞吐率 --准备工作

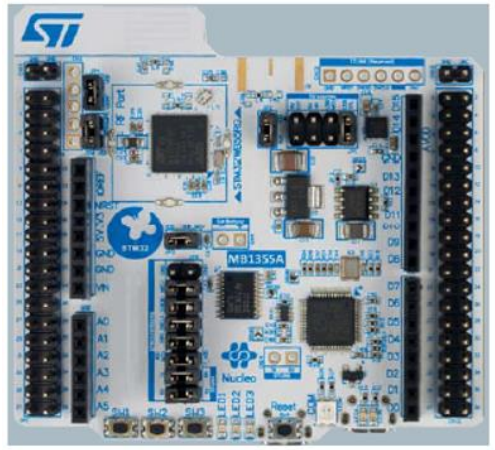


编译和烧录

Projects › P-NUCLEO-WB55.Nucleo › Applications › BLE › BLE_DataThroughput ›

Name	Date modified	Type	Size
Core	3/3/2021 2:01 PM	File folder	
EWARM	3/25/2021 11:48 AM	File folder	
MDK-ARM	3/3/2021 2:01 PM	File folder	
STM32_WPAN	3/3/2021 2:01 PM	File folder	
STM32CubeIDE	3/3/2021 2:01 PM	File folder	
readme.txt	3/3/2021 2:01 PM	Text Document	8 KB

```
#define CFG_BLE_CENTRAL    1
```



编译和烧录

Projects › P-NUCLEO-WB55.Nucleo › Applications › BLE › BLE_DataThroughput ›

Name	Date modified	Type	Size
Core	3/3/2021 2:01 PM	File folder	
EWARM	3/25/2021 11:48 AM	File folder	
MDK-ARM	3/3/2021 2:01 PM	File folder	
STM32_WPAN	3/3/2021 2:01 PM	File folder	
STM32CubeIDE	3/3/2021 2:01 PM	File folder	
readme.txt	3/3/2021 2:01 PM	Text Document	8 KB

```
#define CFG_BLE_CENTRAL    0
```

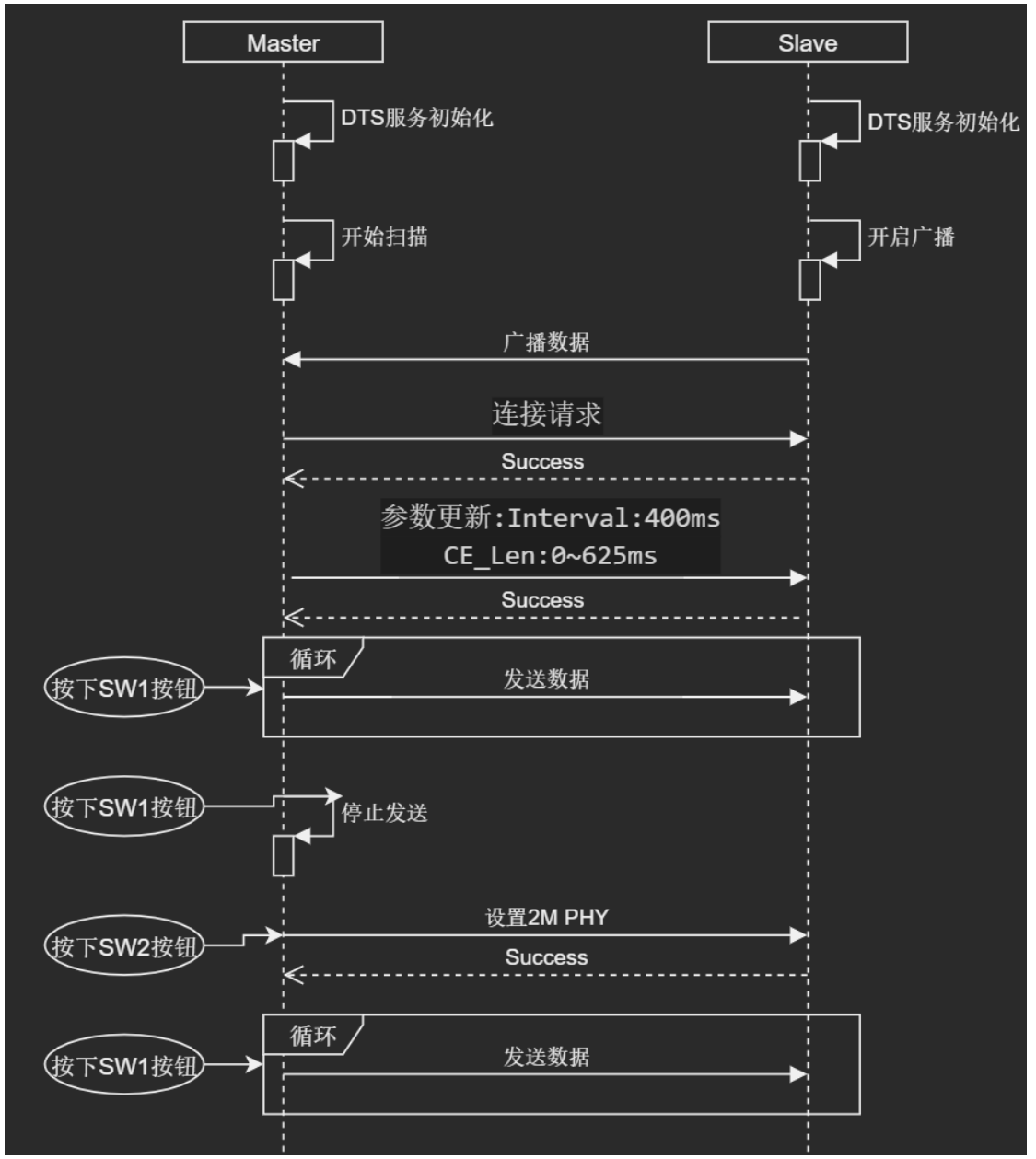
life.augmented

DTS服务

特性	长度	Property（性质）	Permission（许可）	作用
CHAR1	255	Notify	None	发送数据：Tx
CHAR2	255	Write	None	接收数据：Rx
CHAR3	255	Notify	None	每秒更新吞吐率（仅从机）

时序图

- 主机：
  - DTS服务初始化
  - 开启扫描
- 从机:
  - DTS服务初始化
  - 开启广播
- 按键:
  - SW1：发送数据开关
  - SW2：设置2M PHY



结果对比

条件一	条件二	理论值 (bits/s)	实测最小值 (bits/s)	实测最大值 (bits/s)	最大百分比
BLE 5.0	1M PHY	≈803000	775680	781440	≈97.3%
BLE 5.0	2M PHY	≈1400000	1368960	1378570	≈98.5%

BLE的安全特性

1 安全和隐私

2 如何理解蓝牙地址

3 配对

4 测试工程

life.augmented

1.安全和隐私

安全和隐私的原则

安全的原则：防止‘攻击者’获取或改变信息

安全三要素：C.I.A

- Confidentiality（保密性）
- Integrity（完整性）
- Availability（有效性）

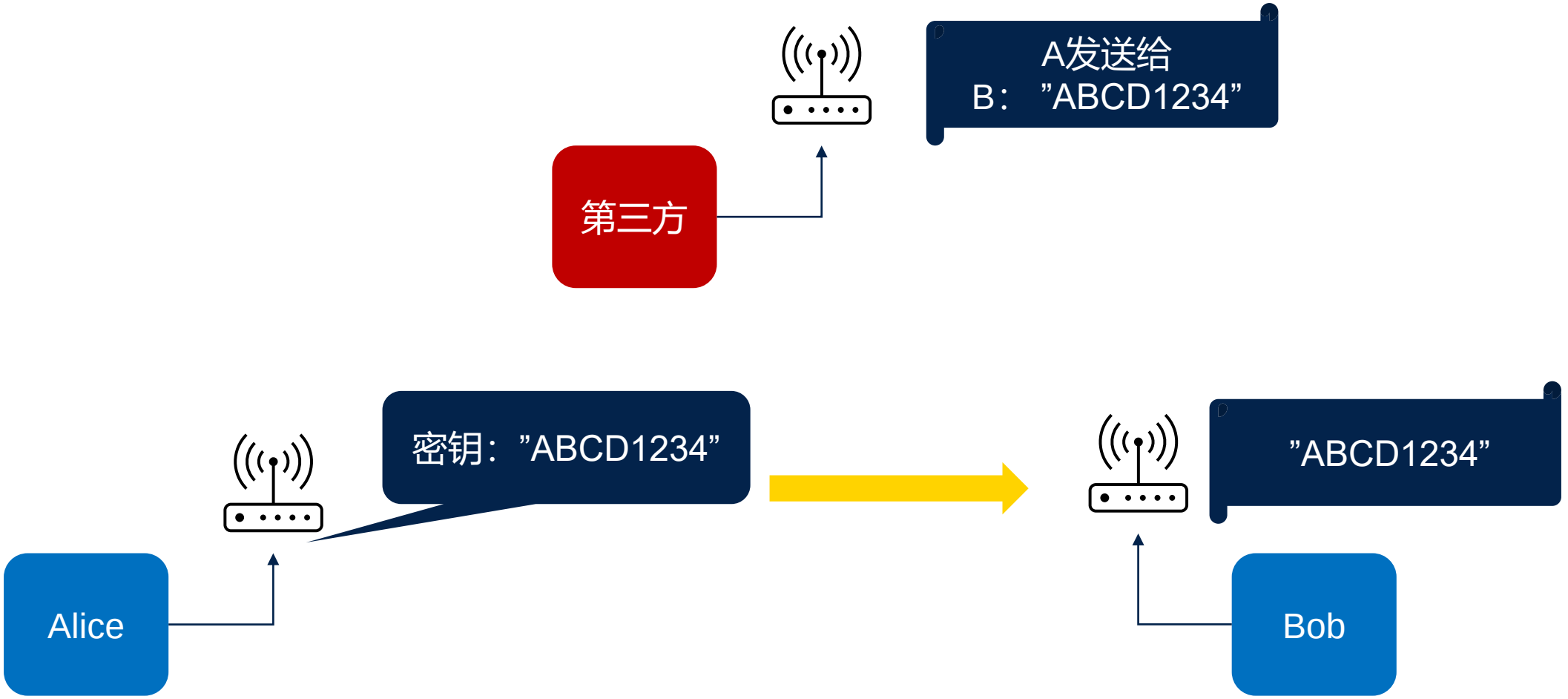
隐私(Privacy)：防止追踪

认证(Authentication)：

- 身份的验证，确保不被第三方冒充

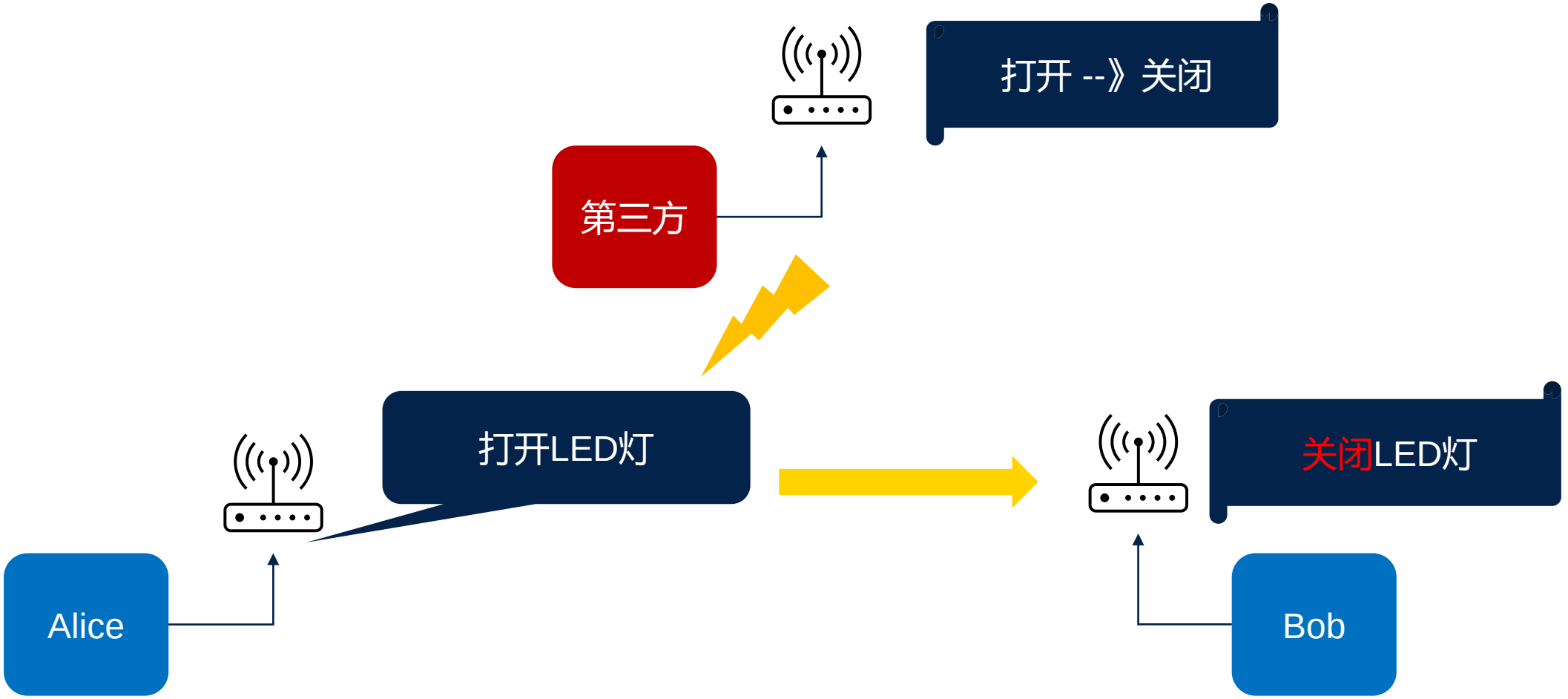
life.augmented

安全三要素和隐私
-- 保密性（Confidentiality）



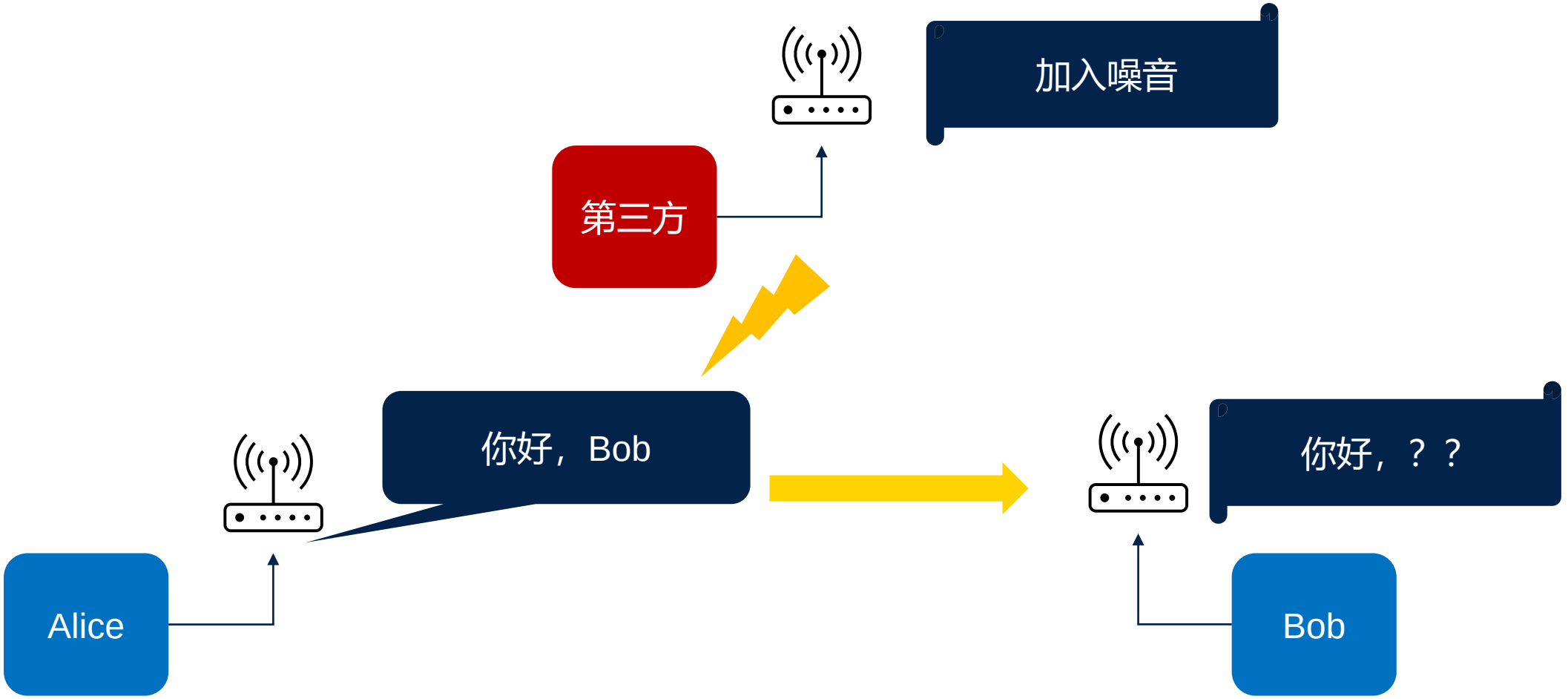
life.augmented

安全三要素和隐私
--完整性（ Integrity ）



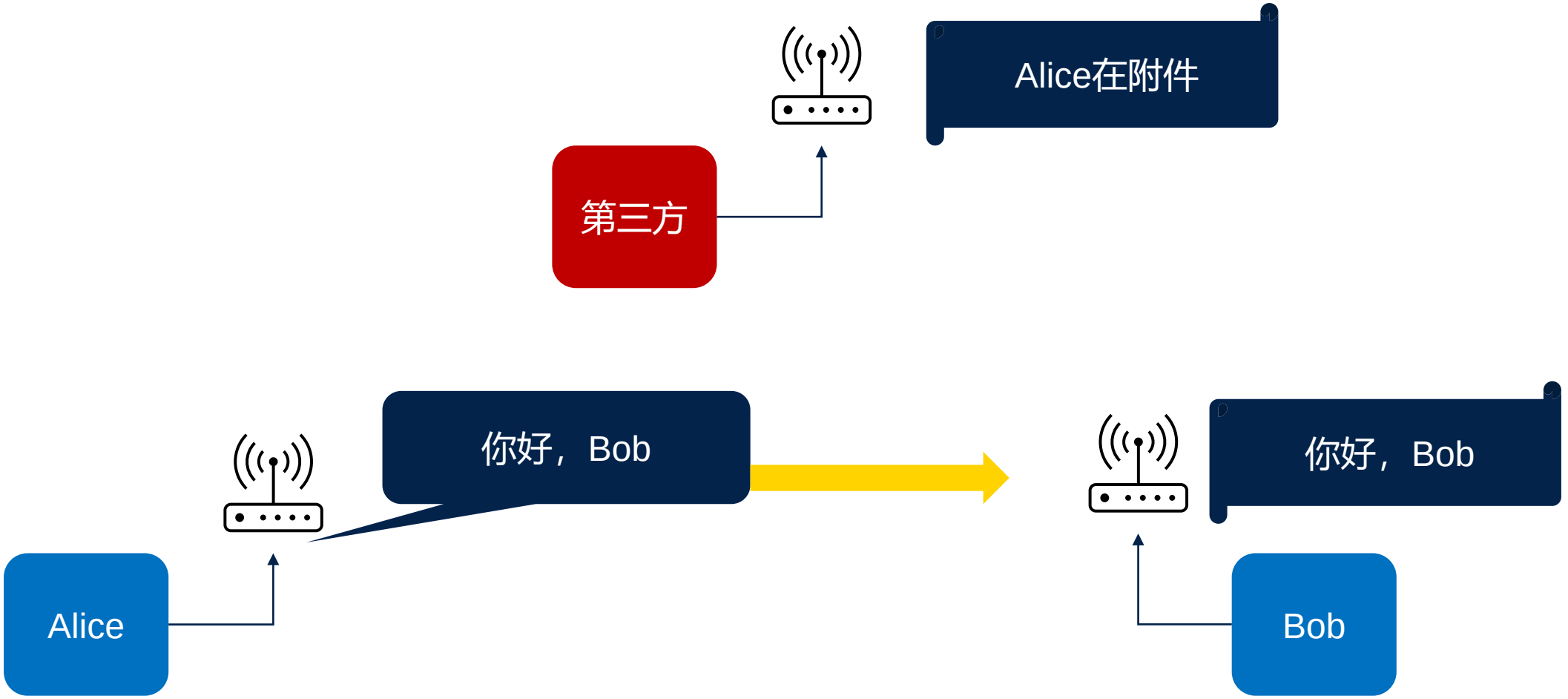
life.augmented

安全三要素和隐私
--有效性（Availability）



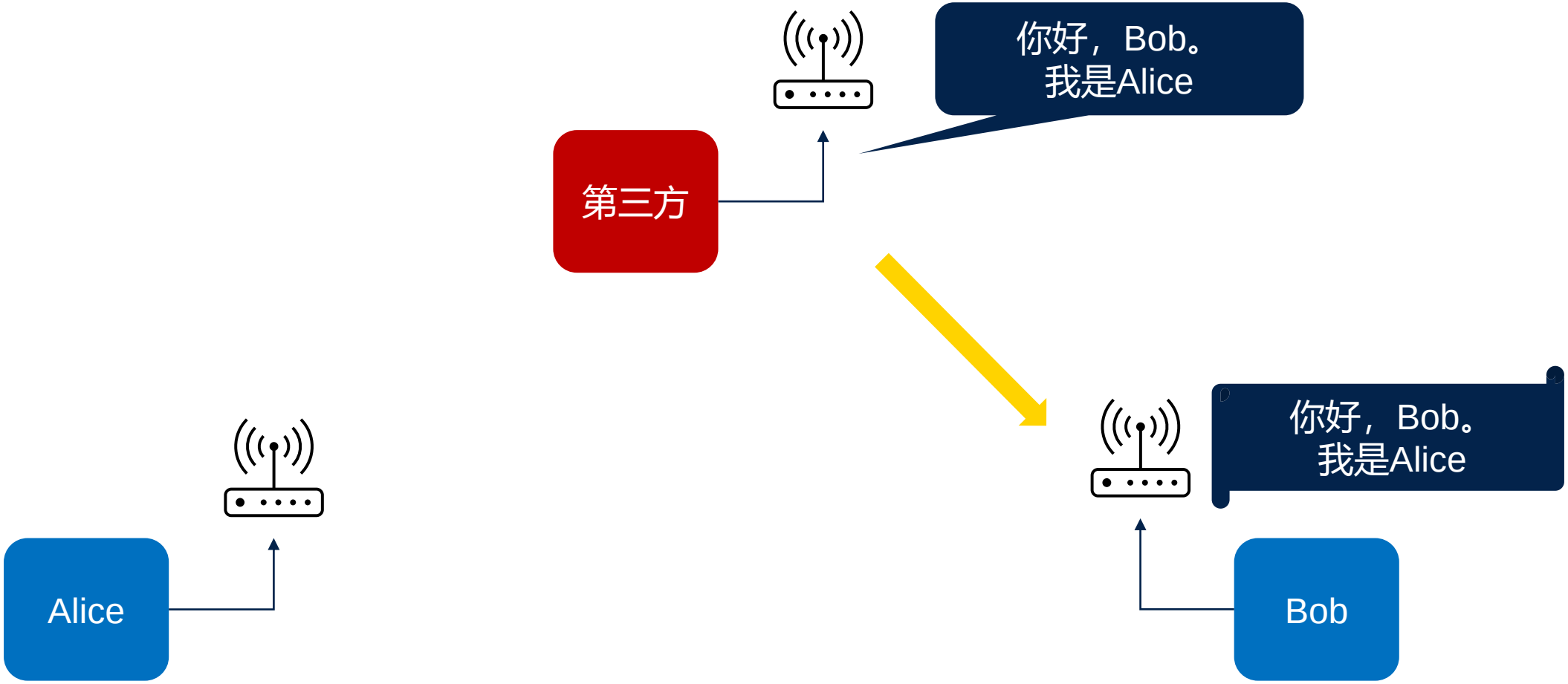
life.augmented

安全三要素和隐私
--隐私（Privacy）

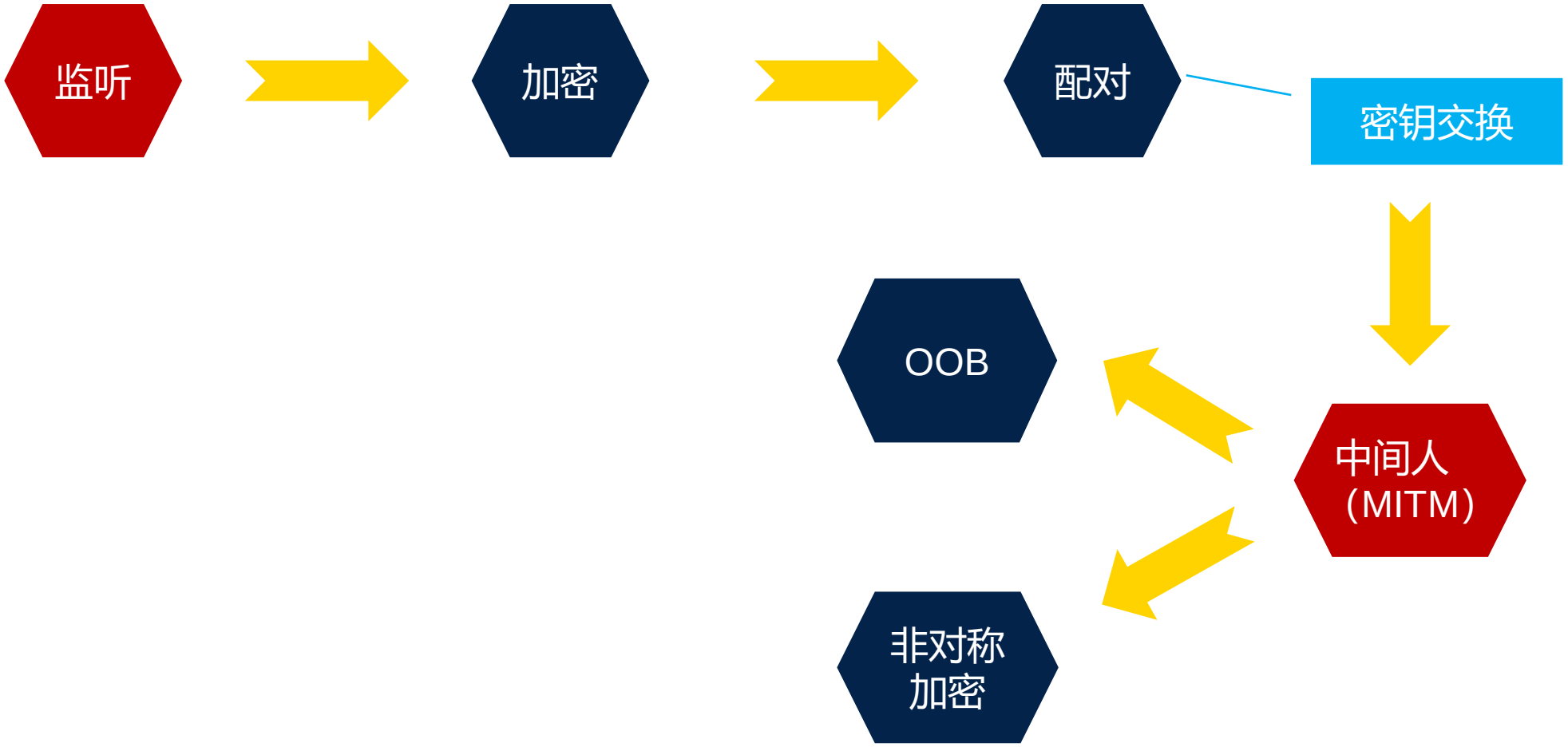


life.augmented

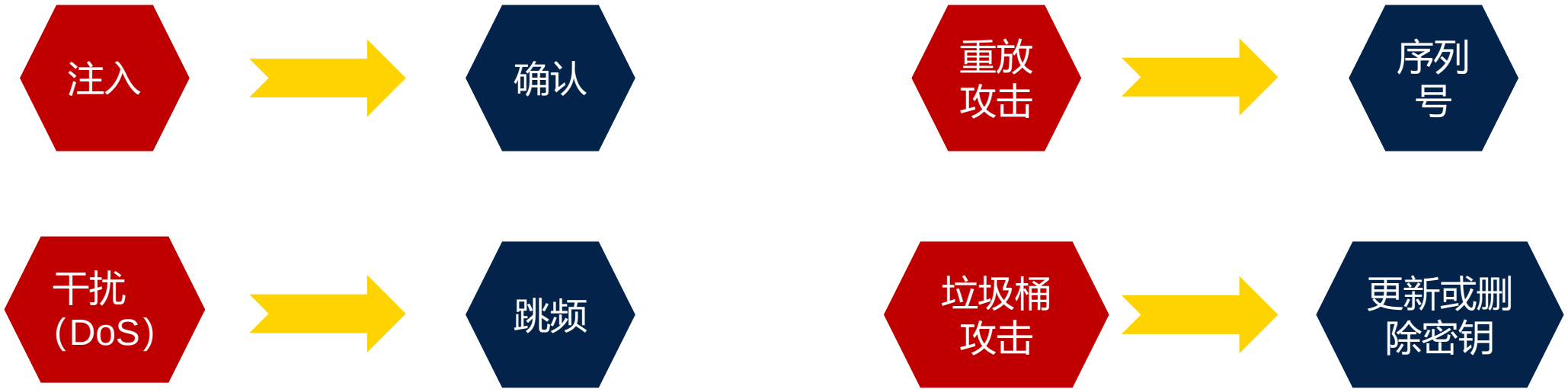
安全三要素和隐私
--认证（Authentication）



攻击手段与保护措施

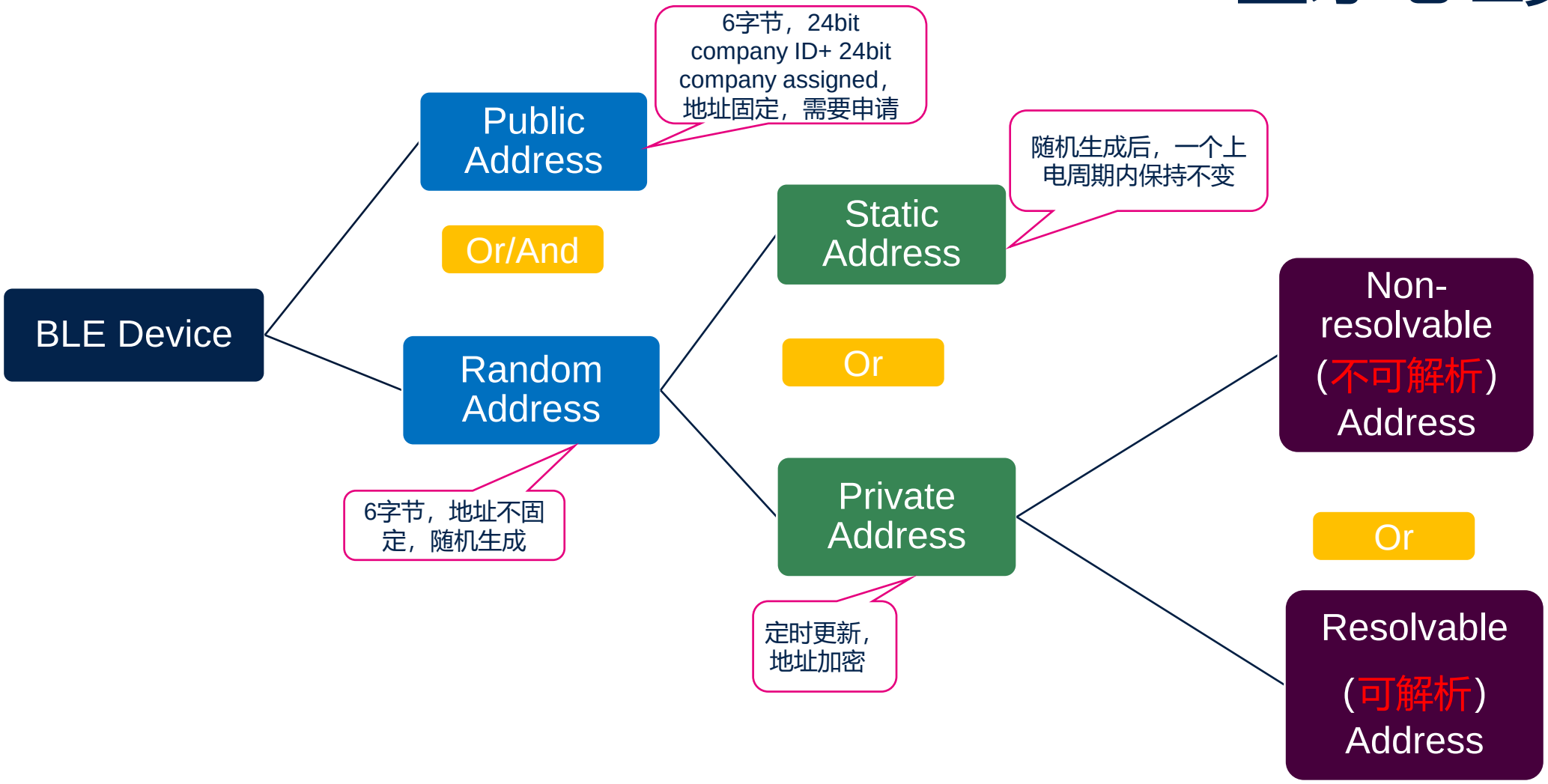


攻击手段与保护措施



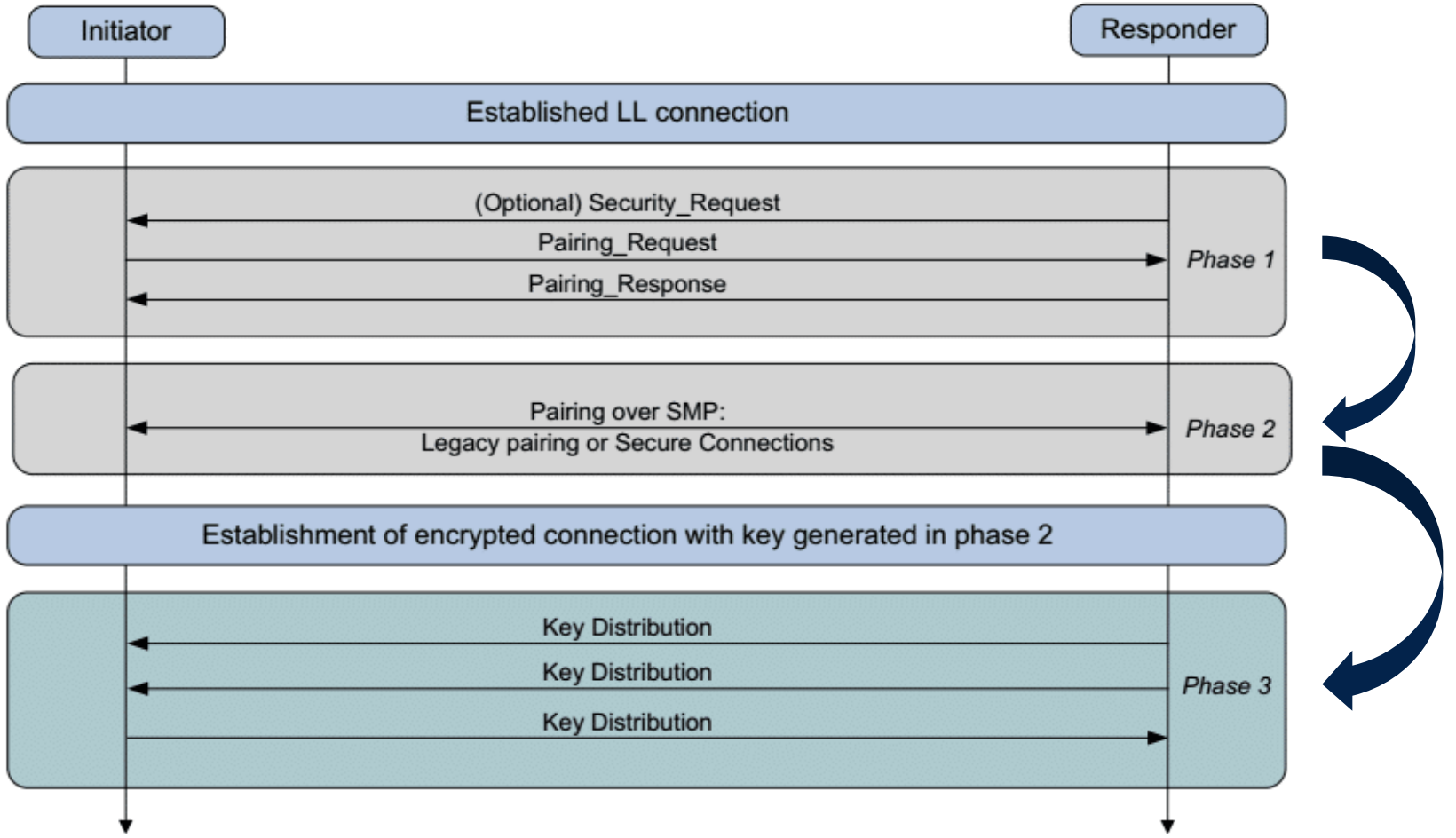
2.如何理解蓝牙地址

蓝牙地址类型


BLE Device
Public Address
6字节，24bit company ID+ 24bit company assigned，地址固定，需要申请
Or/And
Random Address
6字节，地址不固定，随机生成
Static Address
随机生成后，一个上电周期内保持不变
Or
Private Address
定时更新，地址加密
Non-resolvable (不可解析) Address
Or
Resolvable (可解析) Address

3.配对

配对
--三个阶段


Initiator
Responder
Established LL connection
(Optional) Security_Request
Pairing_Request
Pairing_Response
Phase 1
Pairing over SMP:
Legacy pairing or Secure Connections
Phase 2
Establishment of encrypted connection with key generated in phase 2
Key Distribution
Key Distribution
Key Distribution
Phase 3

配对
--特性交换

Field	Code	IO	OOB	AuthReq (1 Byte)					Maximum	Initiator	Responder
Sub-define	(1 Byte)	Cap (1 Byte)	DF (1 Byte)	BF	MITM	SC	KP	Reserved	Encryption Key Size (1 Byte)	Key Distribution (1 Byte)	Key Distribution (1 Byte)
Bits*	8	8	8	2	1	1	1	3	8	8	8

Table 1 Pairing Request/Response

*Bit order is LSB to MSB.

配对请求/响应包格式

		Local output capacity	
		No output	Numeric output
Local input capacity	No input	NoInputNoOutput	DisplayOnly
	Yes/No	NoInputNoOutput[1]	DisplayYesNo
	Keyboard	KeyboardOnly	KeyboardDisplay

设备的输入输出能力

life.augmented

配对方法选择

- 配对方法
  - 传统配对方法：
    - 仅能工作：Just Works
    - 输入密钥：Passkey
    - 带外：Out-of-Band(OOB)

查询

		Initiator			
		OOB Set	**OOB Not Set**	**MITM Set**	**MITM Not Set**
Responder	**OOB Set**	Use OOB	Use OOB		
	OOB Not Set	Use OOB	Check MITM		
	MITM Set			Use IO Capabilities	Use IO Capabilities
	MITM Not Set			Use IO Capabilities	Use Just Works

  - 安全连接：
    - 仅能工作：Just Works
    - 输入密钥：Passkey
    - 带外：Out-of-Band(OOB)
    - 字符比较：Numeric Comparison

查询

		Initiator			
		OOB Set	**OOB Not Set**	**MITM Set**	**MITM Not Set**
Responder	**OOB Set**	Use OOB	Check MITM		
	OOB Not Set	Check MITM	Check MITM		
	MITM Set			Use IO Capabilities	Use IO Capabilities
	MITM Not Set			Use IO Capabilities	Use Just Works

life.augmented

配对方法选择

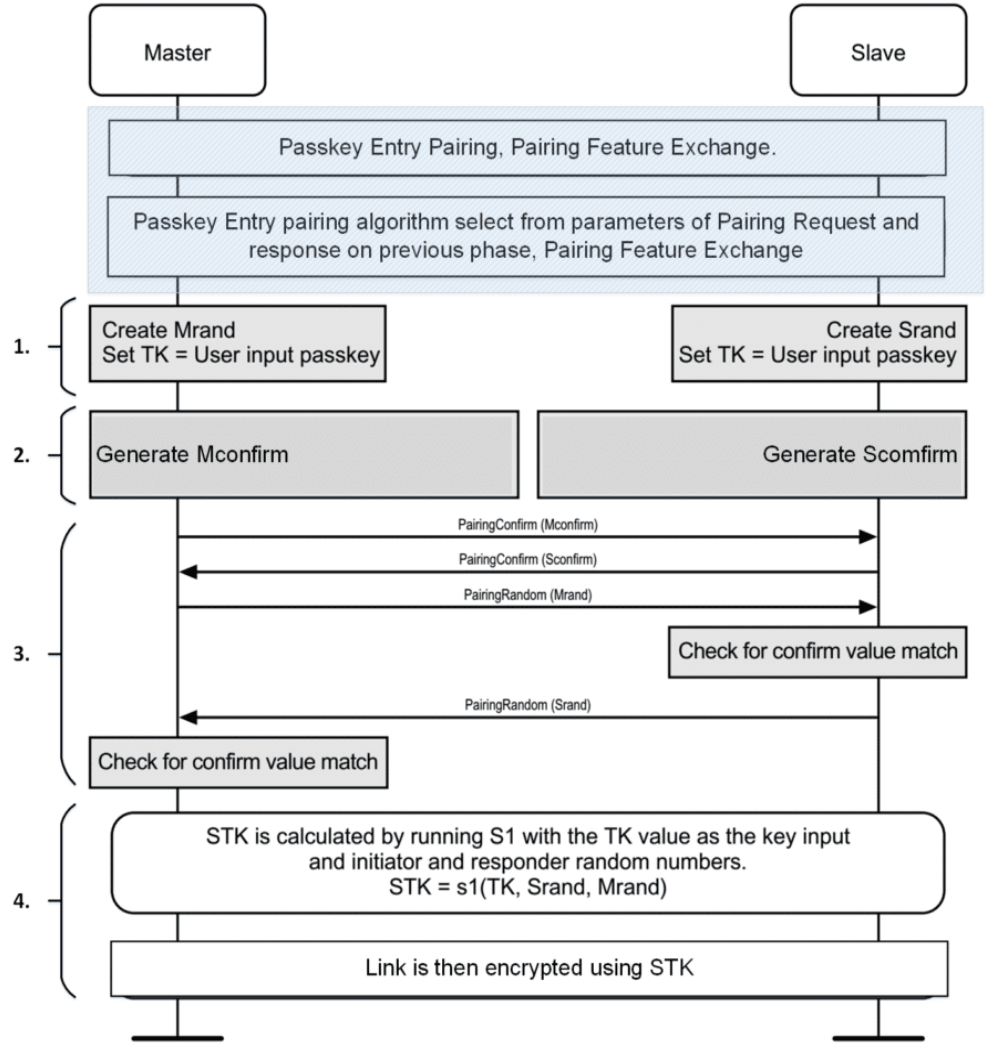
Responder	Initiator				
	DisplayOnly	**Display YesNo**	**Keyboard Only**	**NoInput NoOutput**	**Keyboard Display**
Keyboard Only	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry: initiator and responder inputs Authenticated	Just Works Unauthenticated	Passkey Entry: initiator displays, responder inputs Authenticated
NoInput NoOutput	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated
Keyboard Display	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry (For LE Legacy Pairing): initiator displays, responder inputs Authenticated	Passkey Entry: responder displays, initiator inputs Authenticated	Just Works Unauthenticated	Passkey Entry (For LE Legacy Pairing): initiator displays, responder inputs Authenticated
		Numeric Comparison (For LE Secure Connections) Authenticated			Numeric Comparison (For LE Secure Connections) Authenticated

Responder	Initiator				
	DisplayOnly	**Display YesNo**	**Keyboard Only**	**NoInput NoOutput**	**Keyboard Display**
Display Only	Just Works Unauthenticated	Just Works Unauthenticated	Passkey Entry: responder displays, initiator inputs Authenticated	Just Works Unauthenticated	Passkey Entry: responder displays, initiator inputs Authenticated
Display YesNo	Just Works Unauthenticated	Just Works (For LE Legacy Pairing) Unauthenticated	Passkey Entry: responder displays, initiator inputs Authenticated	Just Works Unauthenticated	Passkey Entry (For LE Legacy Pairing): responder displays, initiator inputs Authenticated
		Numeric Comparison (For LE Secure Connections) Authenticated			Numeric Comparison (For LE Secure Connections) Authenticated

life.augmented

配对方法举例

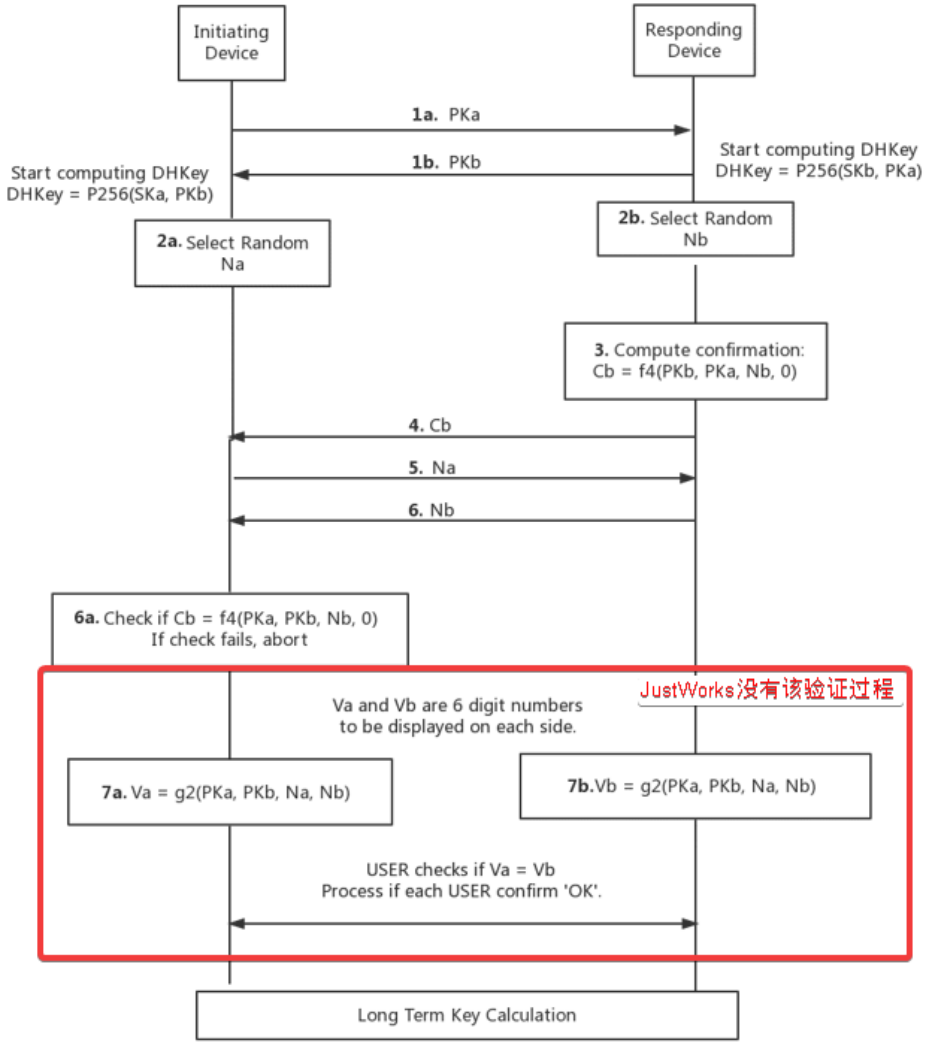
--传统配对：密钥输入（Passkey Entry）



- 1.TK=用户输入Passkey（当TK=0时，为Just Works，当TK由OOB决定时，为OOB配对）
  - 产生128bit随机数：Mrand，Srand
- 2.产生128bit确认值：Mconfirm，Sconfirm（产生函数：由TK，双方特性和蓝牙地址决定）
- 3.验证：使用Mrand确认Mconfirm，使用Srand确认Sconfirm。如果确认失败，则返回配对失败。
- 4.STK：如果验证都成功，那么将计算STK，并告知控制器使能加密。
- STK由以下信息产生：
  - TK
  - Srand
  - Mrand

life.augmented

配对方法举例
--安全连接：Numeric Comparison和Just Works



- 安全连接使用ECDH椭圆曲率加密算法，解决了在有人监听的条件下如何完成密钥协商的问题。

1. 首先双方交换公共密钥PKa，PKb,根据ECDH函数生成DHKey
2. 双方生成随机数
3. 响应的设备生成确认信息Cb，
4. Cb发送给发起设备
5. 发起设备发送随机数Na
6. 响应设备发送随机数Nb

6a. 发起设备进行验证，如果失败则终止流程

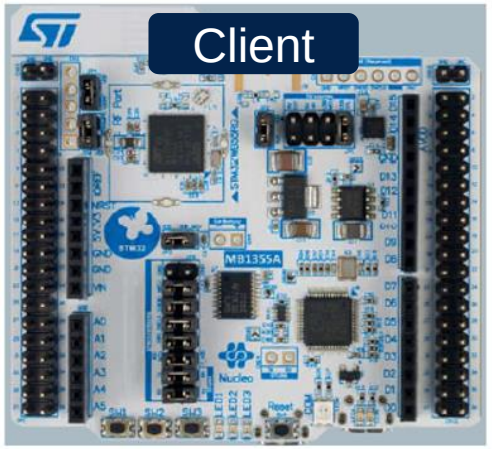
7. 用户双方判断Va 是否等于 Vb

- 安全连接Just Works：无验证

life.augmented

4.测试工程：配对

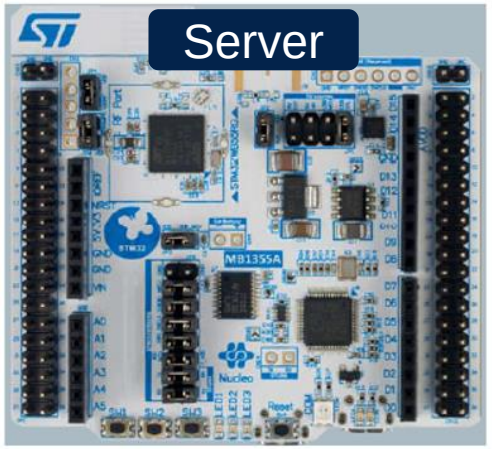
测试BLE配对 --准备工作



编译和烧录

Projects › P-NUCLEO-WB55.Nucleo › Applications › BLE › BLE_p2pClient ›

Name	Date modified	Type	Si
Binary	3/3/2021 2:01 PM	File folder	
Core	3/3/2021 2:01 PM	File folder	
EWARM	3/3/2021 2:01 PM	File folder	
MDK-ARM	3/3/2021 2:01 PM	File folder	
STM32_WPAN	3/3/2021 2:01 PM	File folder	
STM32CubeIDE	3/3/2021 2:01 PM	File folder	
.extSettings	3/3/2021 2:01 PM	EXTSETTINGS File	
BLE_p2pClient.ioc	3/3/2021 2:01 PM	STM32CubeMX	
readme.txt	3/3/2021 2:01 PM	Text Document	



编译和烧录

Projects › P-NUCLEO-WB55.Nucleo › Applications › BLE › BLE_p2pServer ›

Name	Date modified	Type	Siz
Binary	3/3/2021 2:01 PM	File folder	
Core	3/3/2021 2:01 PM	File folder	
EWARM	3/3/2021 2:01 PM	File folder	
MDK-ARM	3/3/2021 2:01 PM	File folder	
STM32_WPAN	3/3/2021 2:01 PM	File folder	
STM32CubeIDE	3/3/2021 2:01 PM	File folder	
.extSettings	3/3/2021 2:01 PM	EXTSETTINGS File	
BLE_p2pServer.ioc	3/3/2021 2:01 PM	STM32CubeMX	
readme.txt	3/3/2021 2:01 PM	Text Document	

life.augmented

配对相关API

AN5270

Table 144. ACI_GAP_SET_IO_CAPABILITY input parameters

Parameter	Size	Description	Possible values
IO_Capability	1	IO capability of the device.	• 0x00: IO_CAP_DISPLAY_ONLY • 0x01: IO_CAP_DISPLAY_YES_NO • 0x02: IO_CAP_KEYBOARD_ONLY • 0x03: IO_CAP_NO_INPUT_NO_OUTPUT • 0x04: IO_CAP_KEYBOARD_DISPLAY

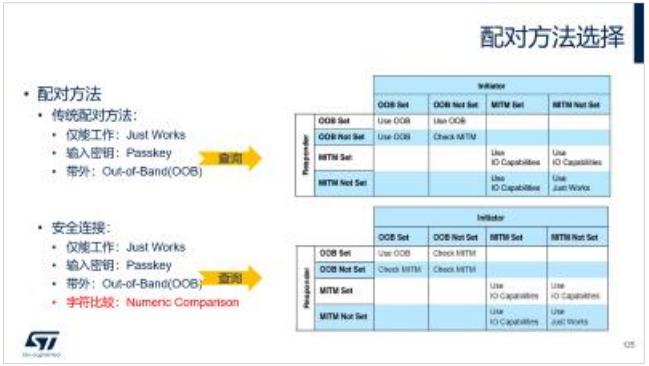
- 设备输入能力如何?
- 设备输出能力如何?

配对相关API

Table 146. **ACI_GAP_SET_AUTHENTICATION_REQUIREMENT input parameters**

Parameter	Size	Description	Possible values
Bonding_Mode	1	Bonding mode. Only if bonding is enabled (0x01), the bonding information is stored in flash	• 0x00: No-bonding mode • 0x01: Bonding mode
MITM_Mode	1	MITM mode	• 0x00: MITM protection not required • 0x01: MITM protection required
SC_Support	1	LE secure connections support	• 0x00: Secure connections pairing not supported • 0x01: Secure connections pairing supported but optional • 0x02: Secure connections pairing supported and mandatory (SC only mode)
KeyPress_Notification_Support	1	Keypress notification support	• 0x00: Keypress notification not supported • 0x01: Keypress notification supported
Min_Encryption_Key_Size	1	Minimum encryption key size to be used during pairing	-
Max_Encryption_Key_Size	1	Maximum encryption key size to be used during pairing	-
Use_Fixed_Pin	1	Use or not fixed pin. If set to 0x00, then during the pairing process the application is not requested for a pin (Fixed_Pin is used). If set to 0x01, then during pairing process if a passkey is required the application is notified	• 0x00: use a fixed pin • 0x01: do not use a fixed pin
Fixed_Pin	4	Fixed pin to be used during pairing if MIMT protection is enabled. Any random value between 0 to 999999	0 ... 999999
Identity_Address_Type	1	Identity address type	• 0x00: Public identity address • 0x01: Random (static) identity address

- 是否需要绑定?
- 是否需要防止中间人?
- 是否使用安全连接?
- 输入Pin码的时候是否通知对方?
- 密钥长度?
- 是否使用固定的pin码?
- 地址类型?



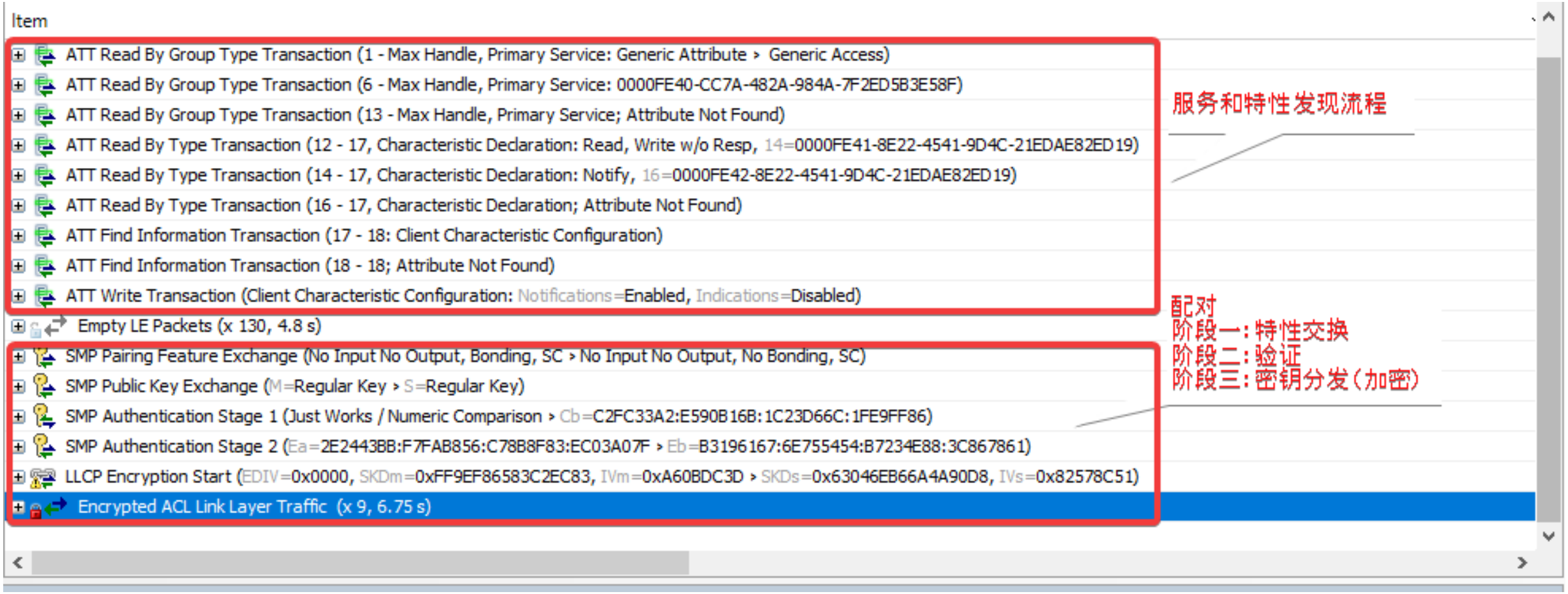
配对相关API

Table 194. **ACI_GAP_SEND_PAIRING_REQ input parameters**

Parameter	Size	Description	Possible values
Connection_Handle	2	Connection handle for which the command is given.	0x0000 ... 0x0EFF
Force_Rebond	1	If 1, pairing request is sent even if the device was previously bonded, otherwise pairing request is not sent.	• 0x00: NO • 0x01: YES

在连接成功之后，可以发起配对流程。

配对结果 --空中包


Item
ATT Read By Group Type Transaction (1 - Max Handle, Primary Service: Generic Attribute › Generic Access)
ATT Read By Group Type Transaction (6 - Max Handle, Primary Service: 0000FE40-CC7A-482A-984A-7F2ED5B3E58F)
ATT Read By Group Type Transaction (13 - Max Handle, Primary Service; Attribute Not Found)
ATT Read By Type Transaction (12 - 17, Characteristic Declaration: Read, Write w/o Resp, 14=0000FE41-8E22-4541-9D4C-21EDAE82ED19)
ATT Read By Type Transaction (14 - 17, Characteristic Declaration: Notify, 16=0000FE42-8E22-4541-9D4C-21EDAE82ED19)
ATT Read By Type Transaction (16 - 17, Characteristic Declaration; Attribute Not Found)
ATT Find Information Transaction (17 - 18: Client Characteristic Configuration)
ATT Find Information Transaction (18 - 18; Attribute Not Found)
ATT Write Transaction (Client Characteristic Configuration: Notifications=Enabled, Indications=Disabled)
Empty LE Packets (x 130, 4.8 s)
SMP Pairing Feature Exchange (No Input No Output, Bonding, SC › No Input No Output, No Bonding, SC)
SMP Public Key Exchange (M=Regular Key › S=Regular Key)
SMP Authentication Stage 1 (Just Works / Numeric Comparison › Cb=C2FC33A2:E590B16B:1C23D66C:1FE9FF86)
SMP Authentication Stage 2 (Ea=2E2443BB:F7FAB856:C78B8F83:EC03A07F › Eb=B3196167:6E755454:B7234E88:3C867861)
LLCP Encryption Start (EDIV=0x0000, SKDm=0xFF9EF86583C2EC83, IVm=0xA60BDC3D › SKDs=0x63046EB66A4A90D8, IVs=0x82578C51)
Encrypted ACL Link Layer Traffic (x 9, 6.75 s)
服务和特性发现流程
配对
阶段一：特性交换
阶段二：验证
阶段三：密钥分发（加密）

OTA应用介绍

1 OTA简介

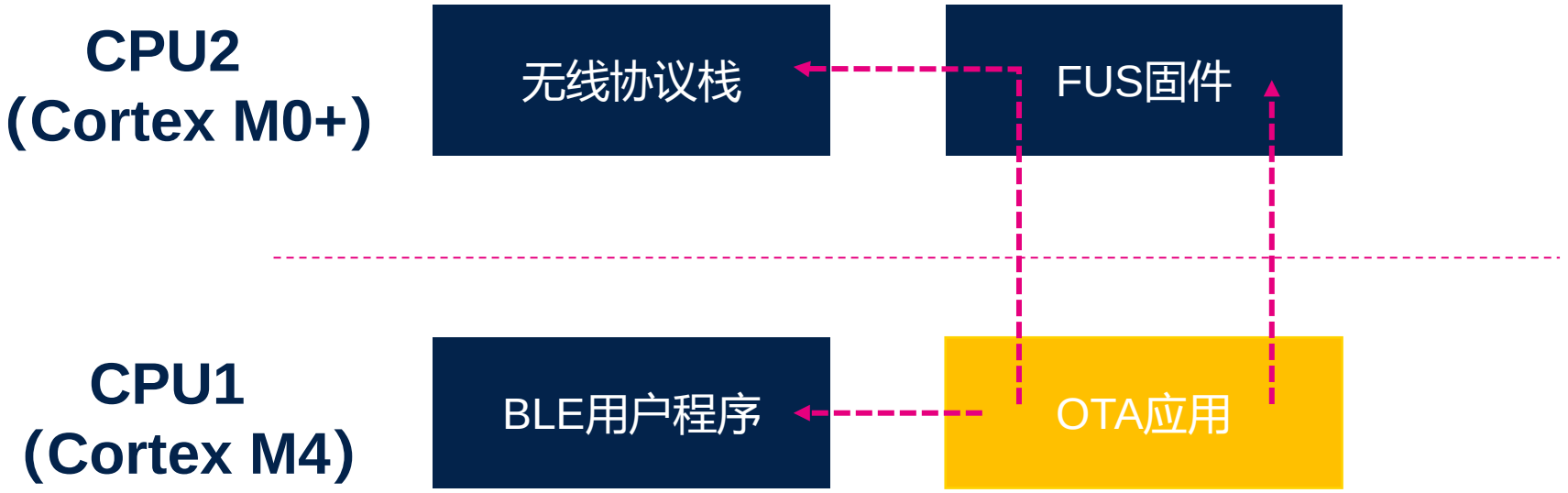
2 如何使BLE应用具备OTA功能?

3 OTA：Step By Step

life.augmented

1.OTA简介

STM32WB双核固件架构



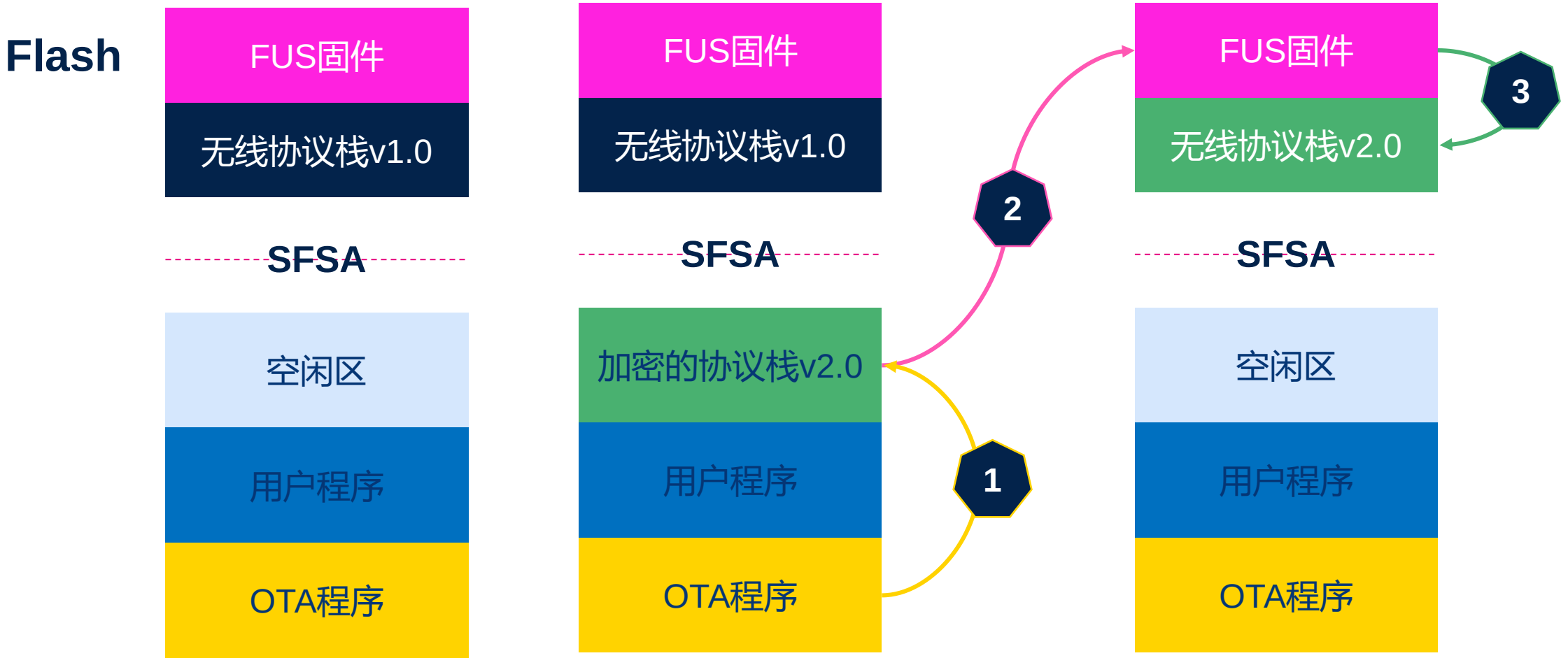
可以由OTA更新

无法由OTA更新

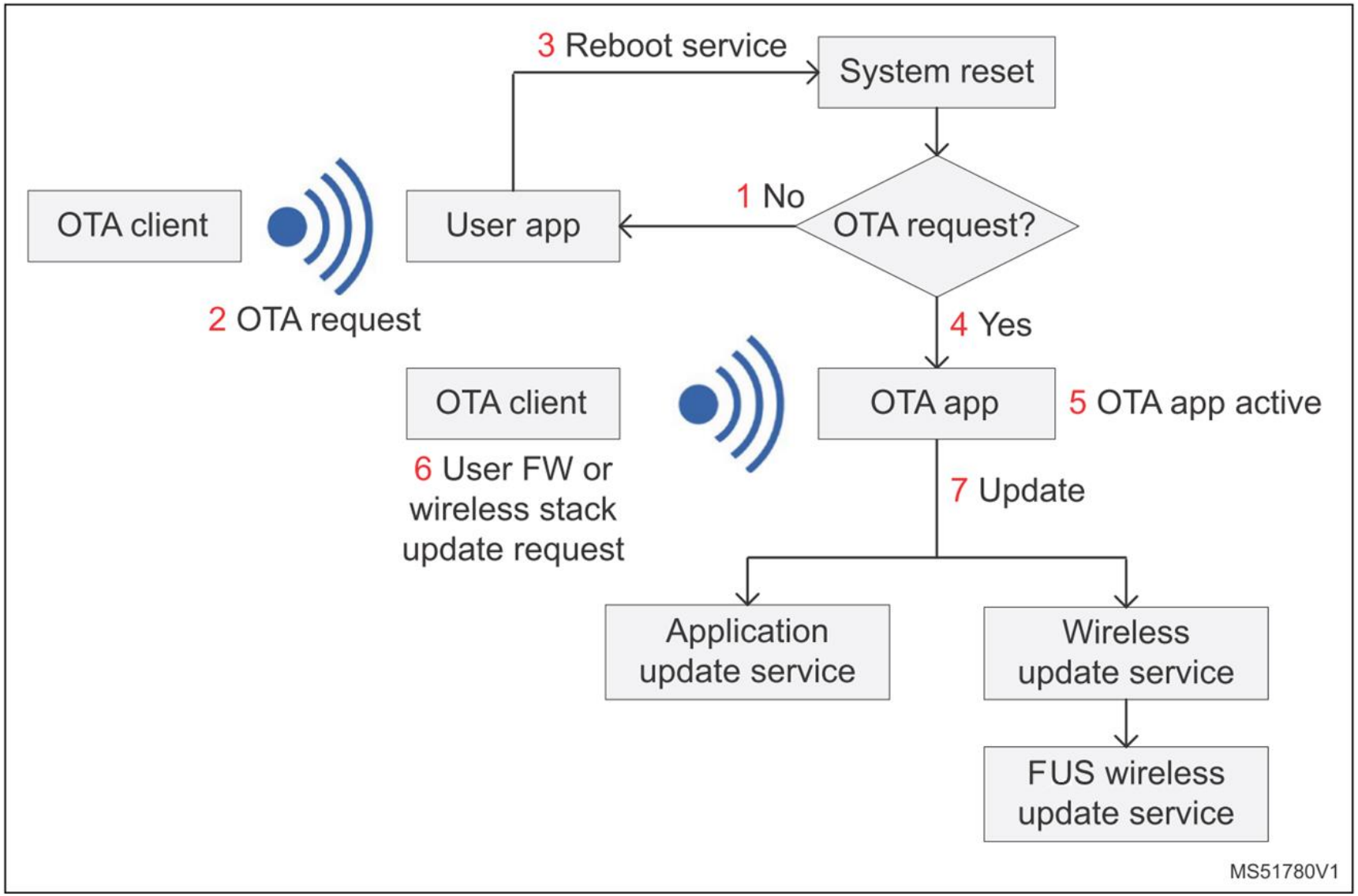
OTA
--用户程序更新


Flash
无线协议栈+FUS
SFSA
空闲区
用户程序v1.0
OTA程序
无线协议栈+FUS
SFSA
空闲区
OTA程序
无线协议栈+FUS
SFSA
空闲区
用户程序v2.0
OTA程序
初始状态
删除当前应用
下载新的应用

OTA
--协议栈更新


Flash
FUS固件
无线协议栈v1.0
SFSA
空闲区
用户程序
OTA程序
初始状态
FUS固件
无线协议栈v1.0
SFSA
加密的协议栈v2.0
用户程序
OTA程序
下载新的加密协议栈
2
1
FUS固件
无线协议栈v2.0
SFSA
空闲区
用户程序
OTA程序
解密和安装新的协议栈
3

OTA应用流程


3 Reboot service
System reset
OTA client
User app
1 No
OTA request?
2 OTA request
4 Yes
OTA client
OTA app
5 OTA app active
6 User FW or
wireless stack
update request
7 Update
Application
update service
Wireless
update service
FUS wireless
update service
MS51780V1

OTA重启特性

Table 1. Reboot request characteristics

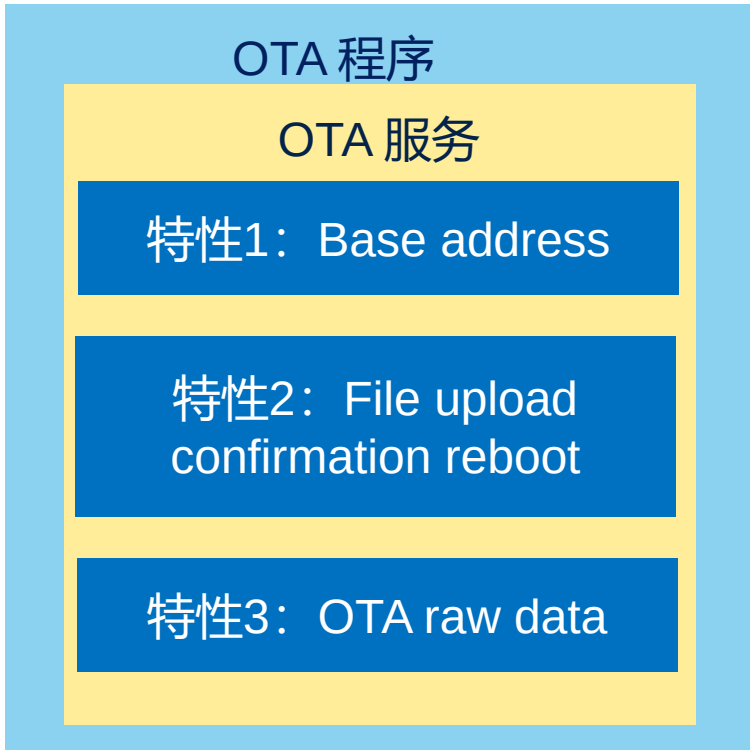
Service				
One of the BLE user application	UUID	-		
Characteristics				
Base address	Function	Request device reboot for OTA application		
	Size	3		
	Mode	Write		
	UUID	0000FE11-8e22-4541-9d4c-21edae82ed19		
	Fields	0	Boot Mode:	0x00 Application 0x01 OTA application
		1	Sector Index	0xXX -> 0x080XX000
		3	Number of sectors to erase	0x00 – 0xFF

life.augmented

OTA服务

Table 2. OTA Service and characteristics declaration

Service				
OTA FW Update	UUID	0000FE20-cc7a-482a-984a-7f2ed5b3e58f		
Characteristics				
Base address	Function	Address to store the file		
	Size	4		
	Mode	Write		
	UUID	000FE22-8e22-4541-9d4c-21edae82ed19		
	Fields	0	Actions	0x00: STOP all upload 0x01: START wireless file upload 0x02: START user application file upload 0x07: File upload finished 0x08: Cancel upload
		1-3	Address	0x007000
File upload confirmation reboot	Function	Confirm the reboot of the application after file uploaded		
	Size	1		
	Mode	Indicate		
	UUID	0000FE23-8e22-4541-9d4c-21edae82ed19		
	Fields	0	0x01	Reboot
OTA raw data	Function	Data to transfer file (split by 20 bytes)		
	Size	20		
	Mode	Write without response		
	UUID	0000FE24-8e22-4541-9d4c-21edae82ed19		
	Fields	0-19	Raw data	



life.augmented

OTA特性的广播

Table 3. AD structure - Main

AD Field Name	AD Type	AD Len	Record size
TX_POWER_LEVEL	0x0A	2	3
COMPLETE_NAME	0x09	8	9
MANUF_SPECIFIC	0xFF	13	14
FLAGS	0x01	2	3

Table 4. AD structure - Manufacturer specific field

Byte	Name	Value	Comment
0	Length	8	-
1	Type	0xFF	Manufacturer specific
2	Version	0x01	-
3	ID	0x00 - 0xFF	0x86 for OTA loader
4	Feature group A	Bit field	Reserved
5			
6	Feature group B		0x20 for OTA enabled device
7			
8-13	Public device address	Bytes	Optional

Table 5. AD structure – Group B features

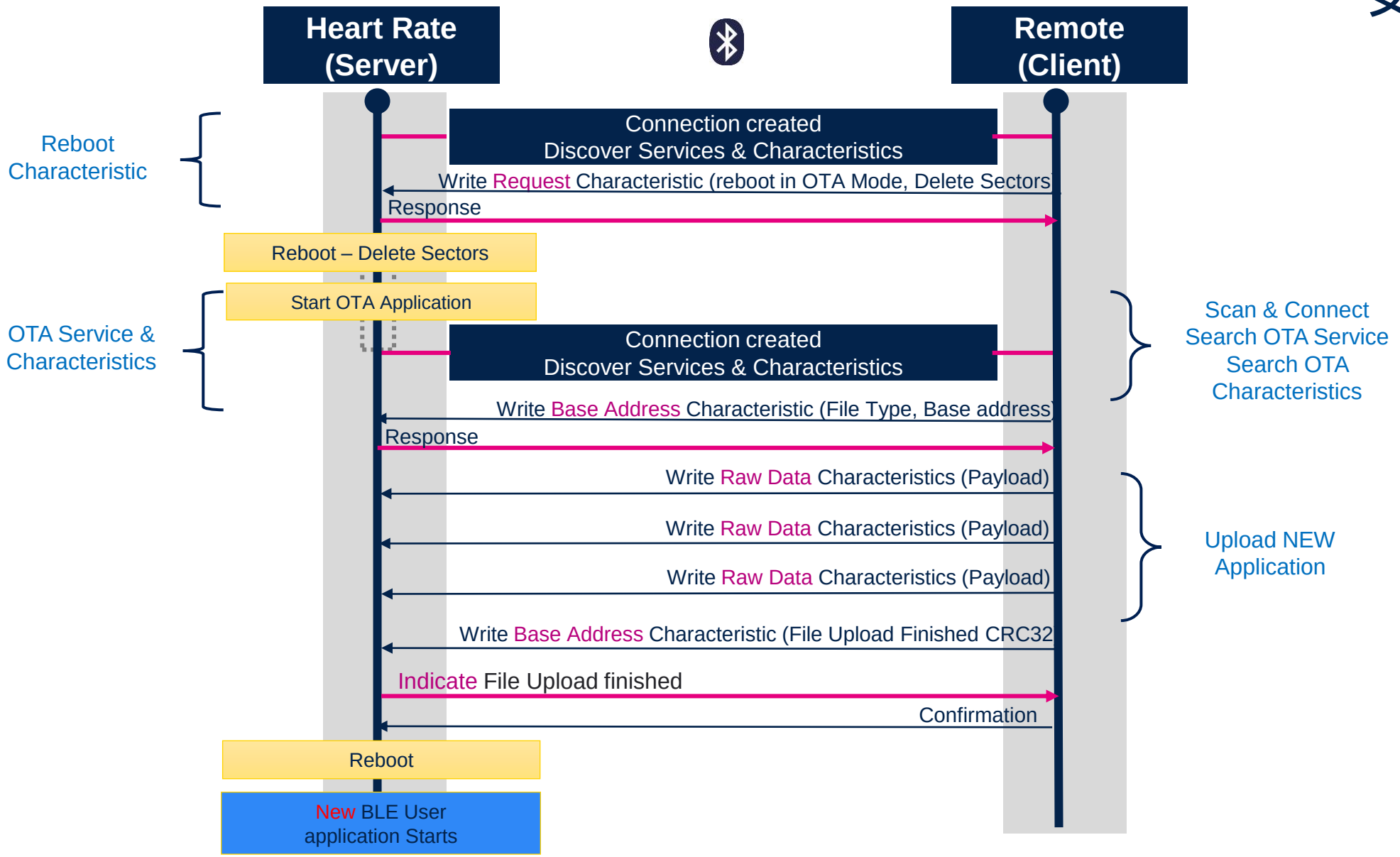
Bit(s)	Field
15	Reserved
14	Thread support
13	OTA reboot request
12-0	Reserved

- 厂商数据：
  - 0xFF
- 广播结构：
  - ID:0x86（支持OTA）
- Feature Group B：
  - 0x20：OTA使能设备

*参考：AN5289

life.augmented

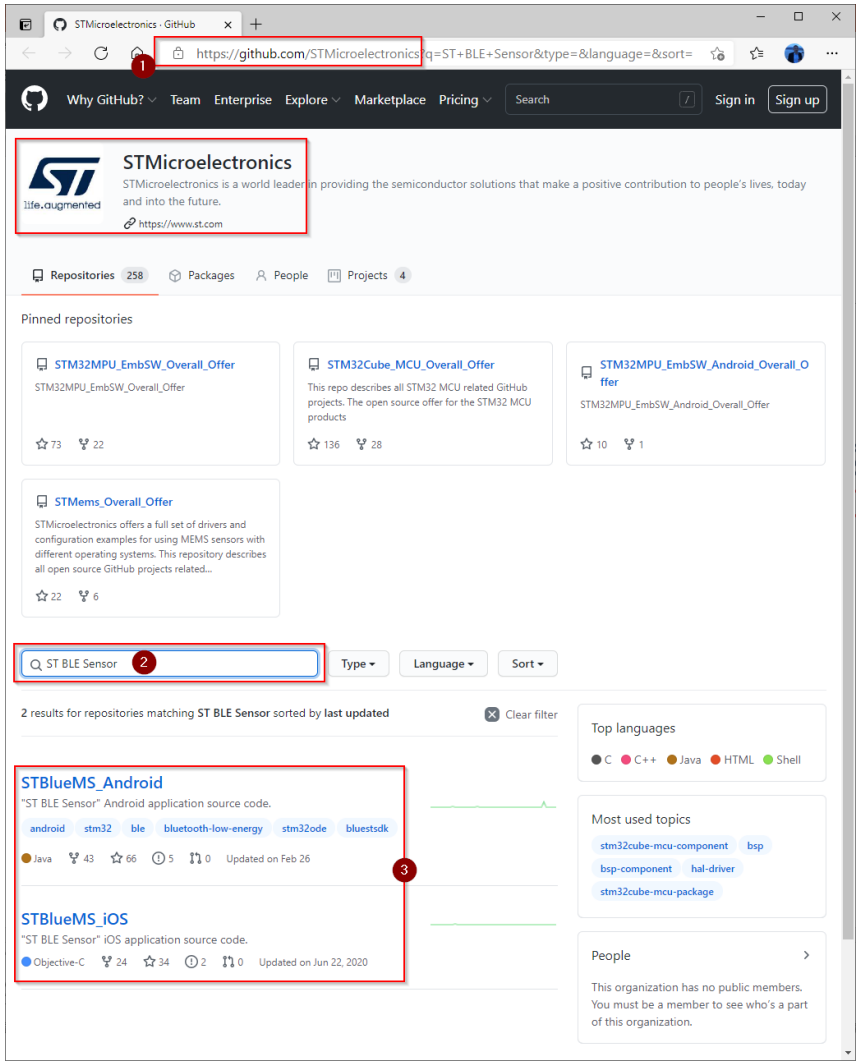
数据流


Heart Rate (Server)
Remote (Client)
Reboot Characteristic
Connection created
Discover Services & Characteristics
Write Request Characteristic (reboot in OTA Mode, Delete Sectors)
Response
Reboot – Delete Sectors
Start OTA Application
OTA Service & Characteristics
Connection created
Discover Services & Characteristics
Scan & Connect
Search OTA Service
Search OTA Characteristics
Write Base Address Characteristic (File Type, Base address)
Response
Write Raw Data Characteristics (Payload)
Write Raw Data Characteristics (Payload)
Write Raw Data Characteristics (Payload)
Upload NEW Application
Write Base Address Characteristic (File Upload Finished CRC32
Indicate File Upload finished
Confirmation
Reboot
New BLE User application Starts

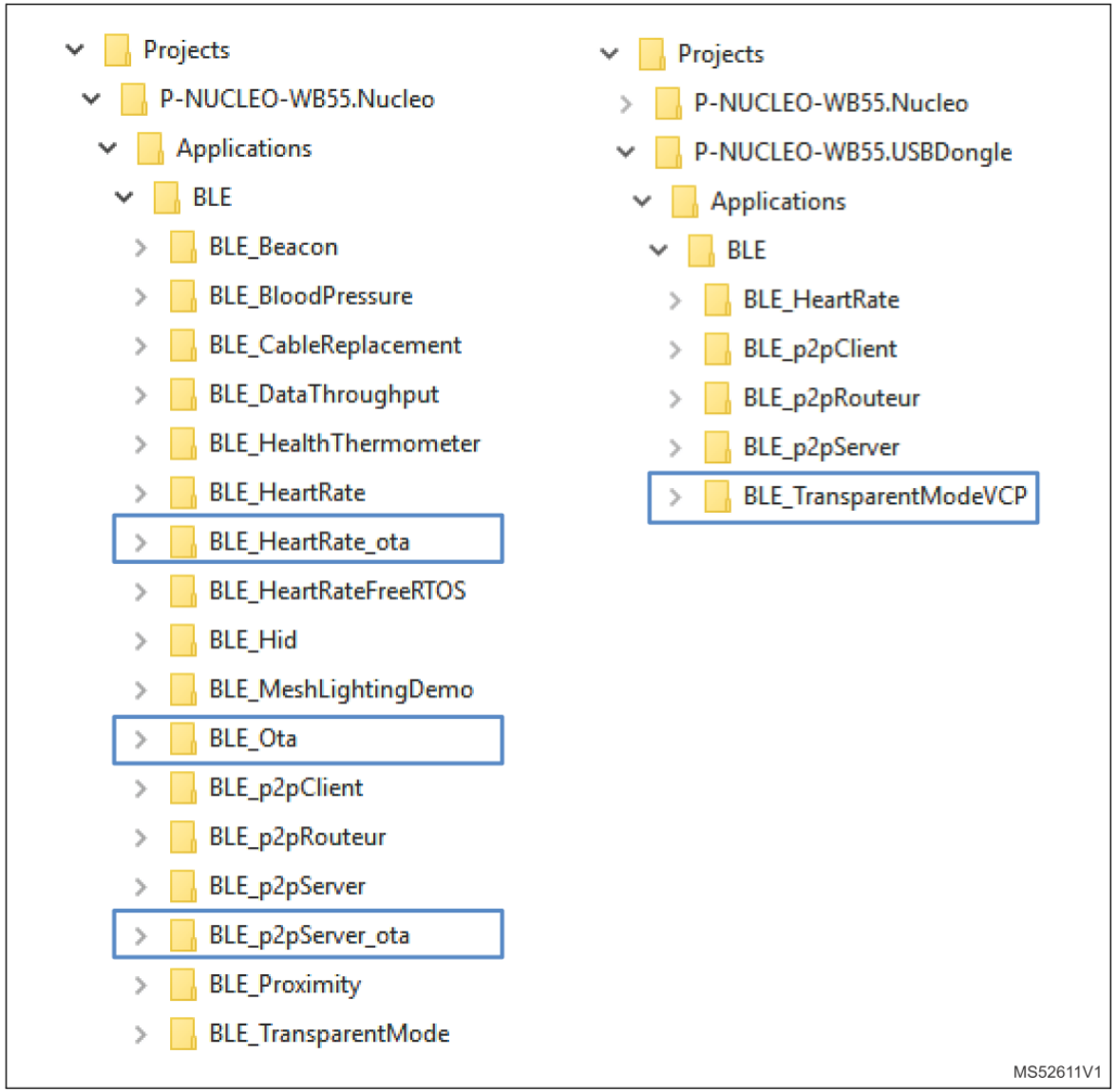
2. 如何使BLE应用具备OTA功能?

Smart Phone App - ST BLE Sensor

- 源码：
  - https://github.com/STMicroelectronics
  - https://github.com/STMicroelectronics/STBlueMS_iOS
  - https://github.com/STMicroelectronics/STBlueMS_Android

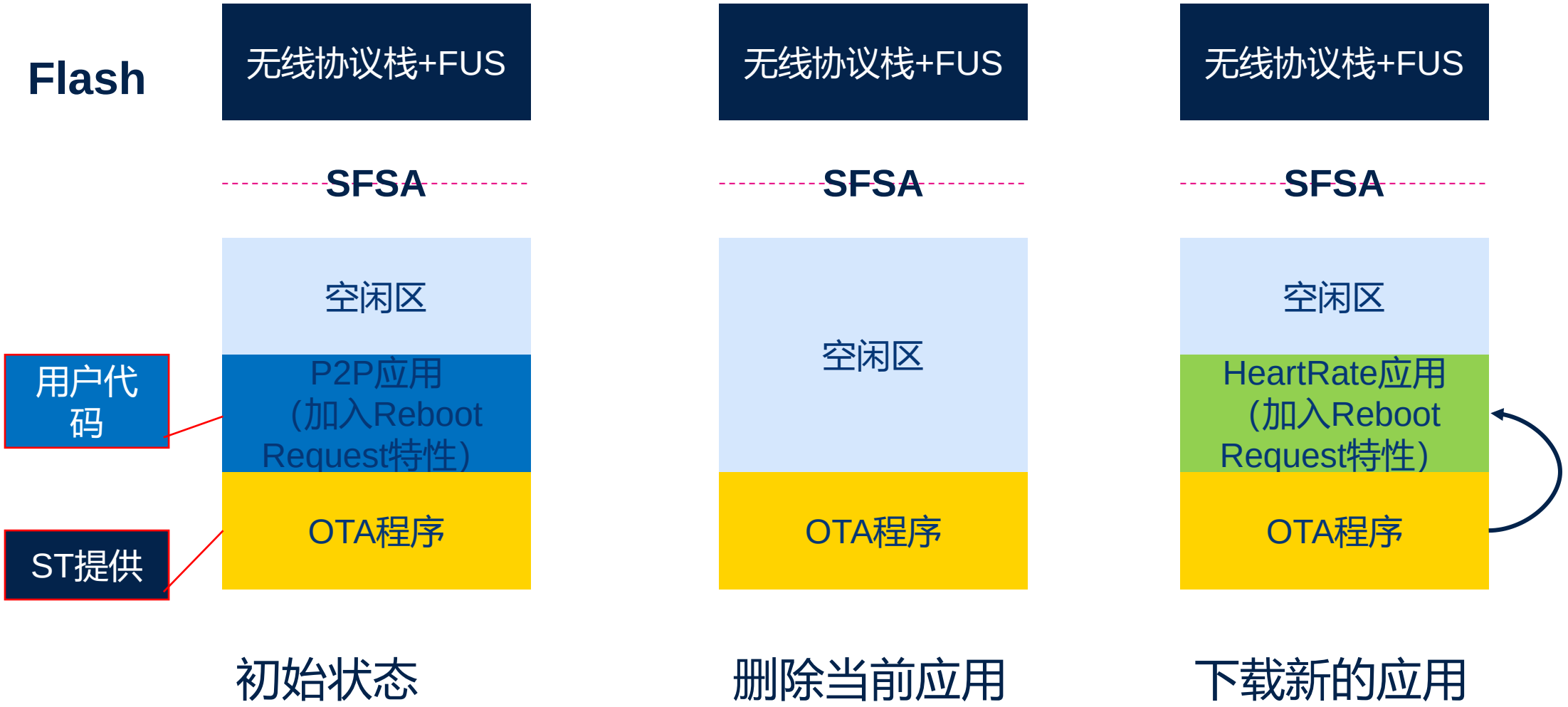


OTA工程目录


Projects
P-NUCLEO-WB55.Nucleo
Applications
BLE
BLE_Beacon
BLE_BloodPressure
BLE_CableReplacement
BLE_DataThroughput
BLE_HealthThermometer
BLE_HeartRate
BLE_HeartRate_ota
BLE_HeartRateFreeRTOS
BLE_Hid
BLE_MeshLightingDemo
BLE_Ota
BLE_p2pClient
BLE_p2pRouteur
BLE_p2pServer
BLE_p2pServer_ota
BLE_Proximity
BLE_TransparentMode
Projects
P-NUCLEO-WB55.Nucleo
P-NUCLEO-WB55.USBDongle
Applications
BLE
BLE_HeartRate
BLE_p2pClient
BLE_p2pRouteur
BLE_p2pServer
BLE_TransparentModeVCP
MS52611V1

如何使BLE应用具备OTA功能?

---添加Reboot Request特性



初始状态　　删除当前应用　　下载新的应用

如何使BLE应用具备OTA功能?
---Link文件修改

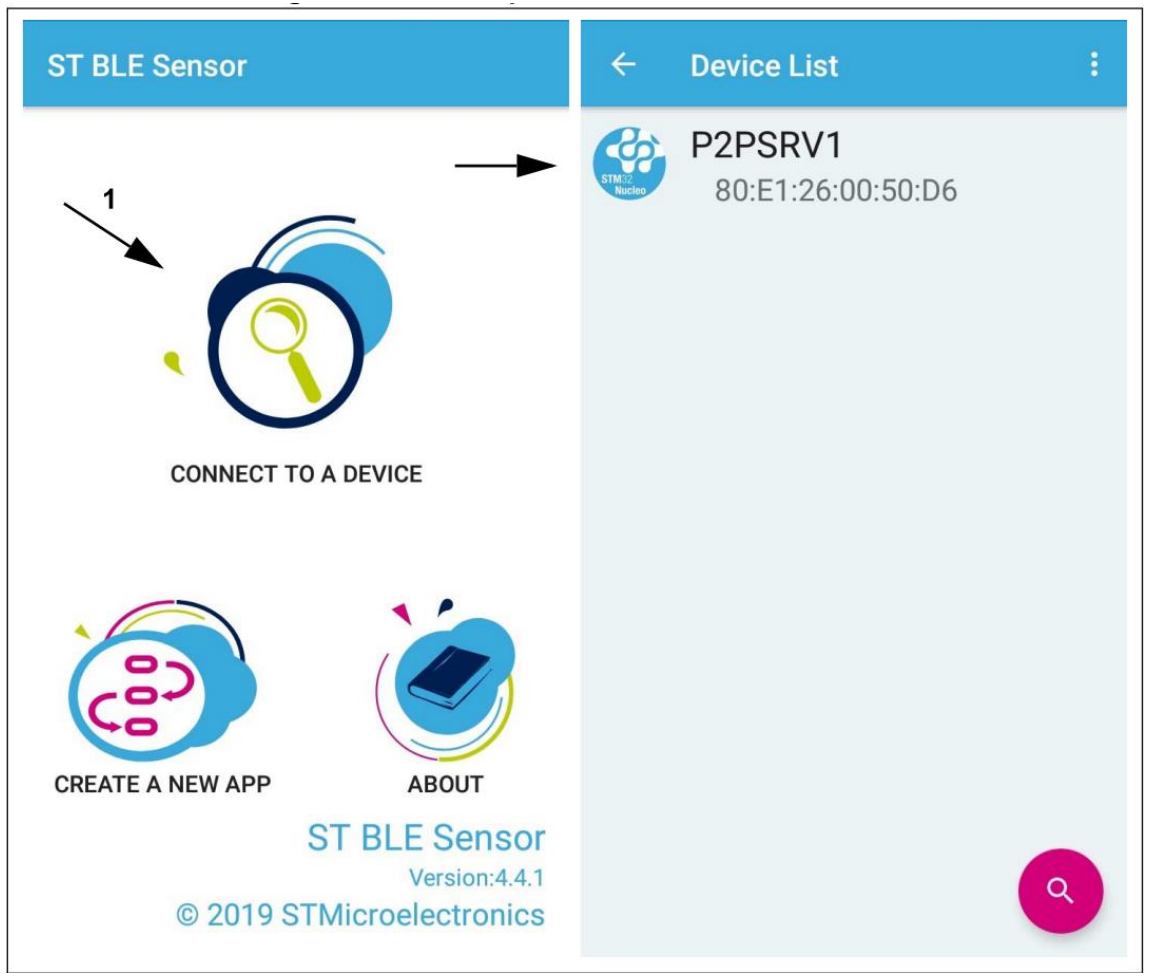
```
/*-CPU Vector Table- Without OTA support*/
define symbol __ICFEDIT_intvec_start__ = 0x08000000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__   = 0x0807FFFF;
define symbol __ICFEDIT_region_RAM1_start__ = 0x20000000;
define symbol __ICFEDIT_region_RAM1_end__   = 0x2002FFFF;
```

```
/*-CPU Vector Table- With OTA application*/
define symbol __ICFEDIT_intvec_start__ = 0x08007000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x08007000;
define symbol __ICFEDIT_region_ROM_end__   = 0x0807FFFF;
define symbol __ICFEDIT_region_RAM1_start__ = 0x20000004;
define symbol __ICFEDIT_region_RAM1_end__   = 0x2002FFFF;
```

life.augmented

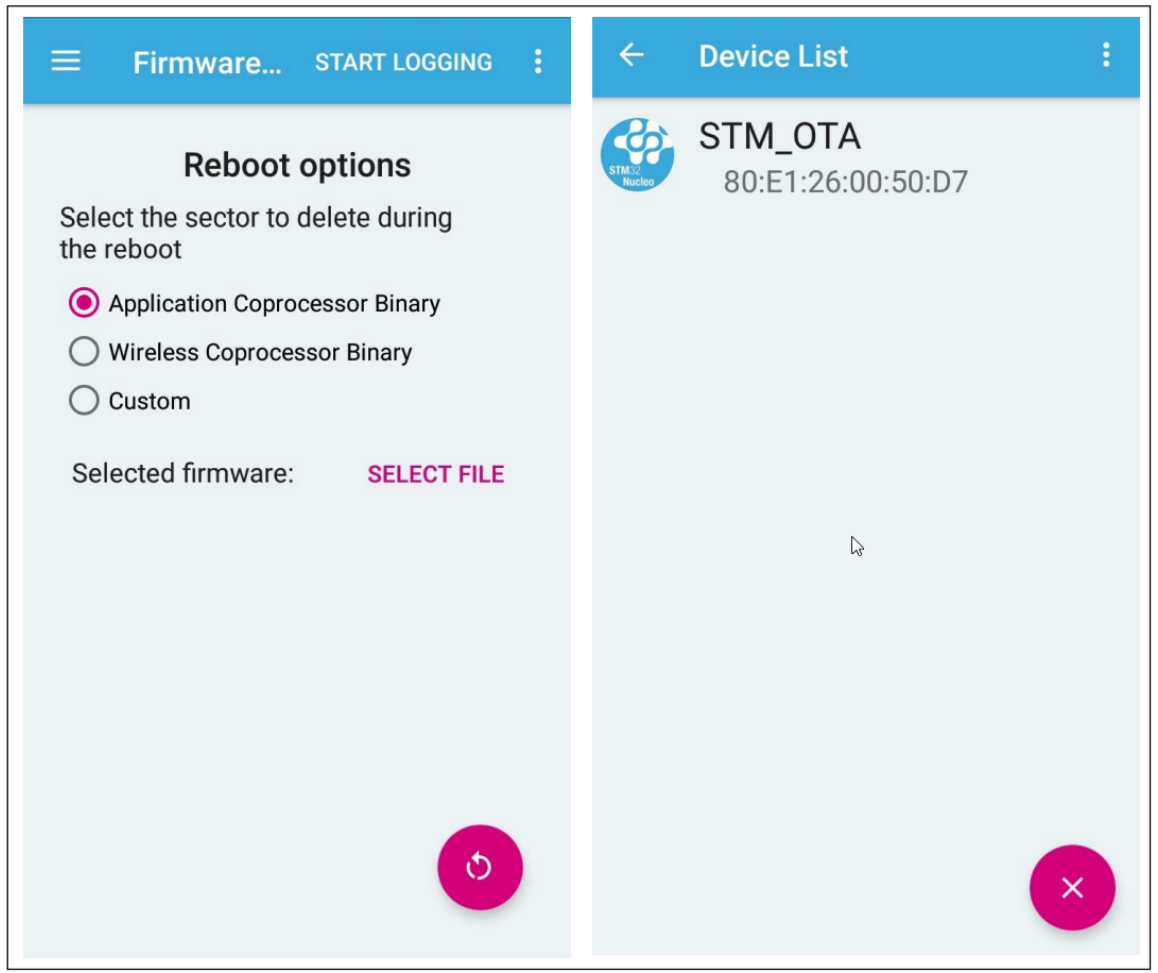
3. OTA：Step By Step

OTA升级：方式一 ---ST BLE Sensor



- 点击Connect To A Device，开始扫描
- 扫描完成后，在Device List中会出现ST可识别的设备
  - Service UUID
  - Feature Group
- 点击Device List中的设备，进行连接

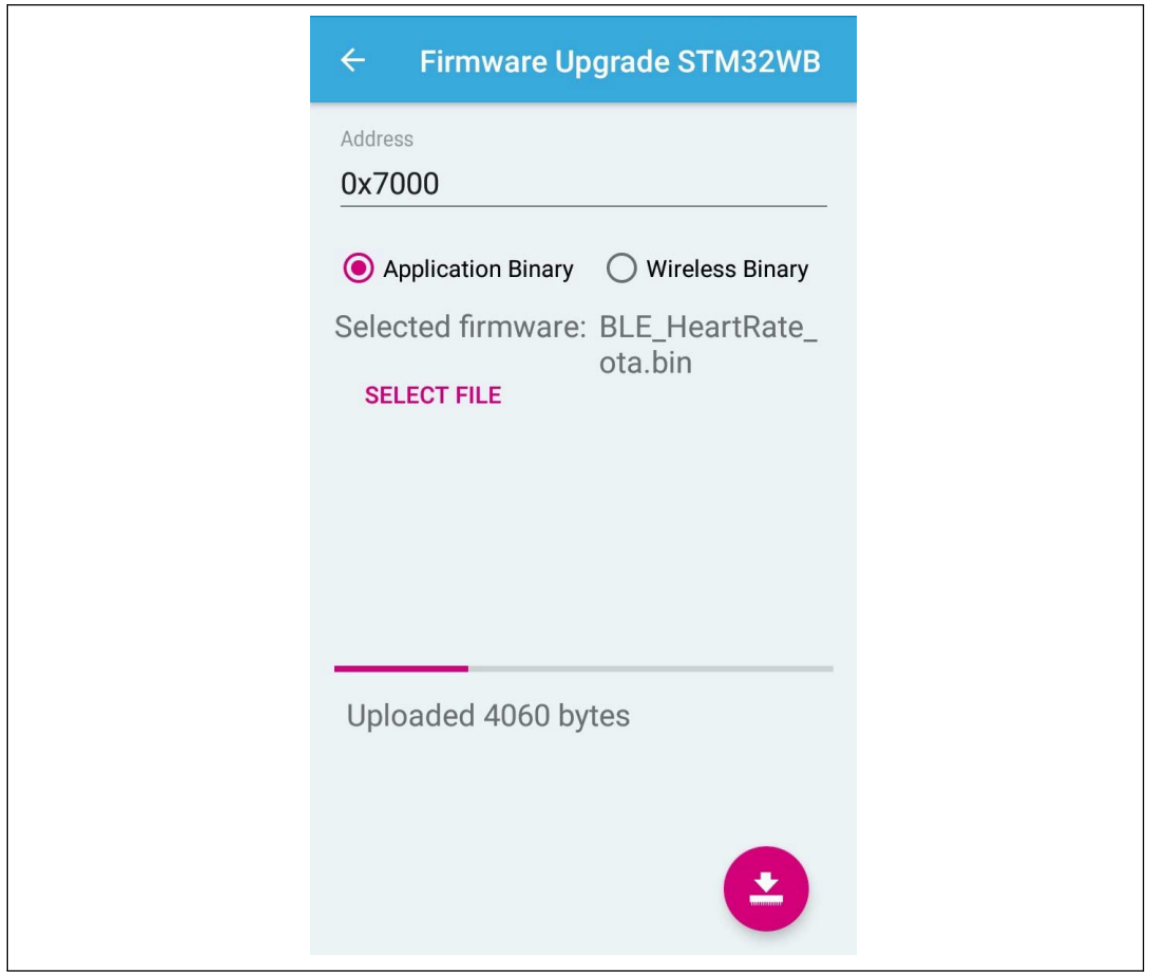
OTA升级：方式一 ---ST BLE Sensor



- 打开Firmware Update选项
- 选择应用程序更新
- 点击右下重启按钮
- 重启后，重新扫描：
  - 如果出现STM_OTA的广播设备，表明固件进入了OTA程序。

life.augmented

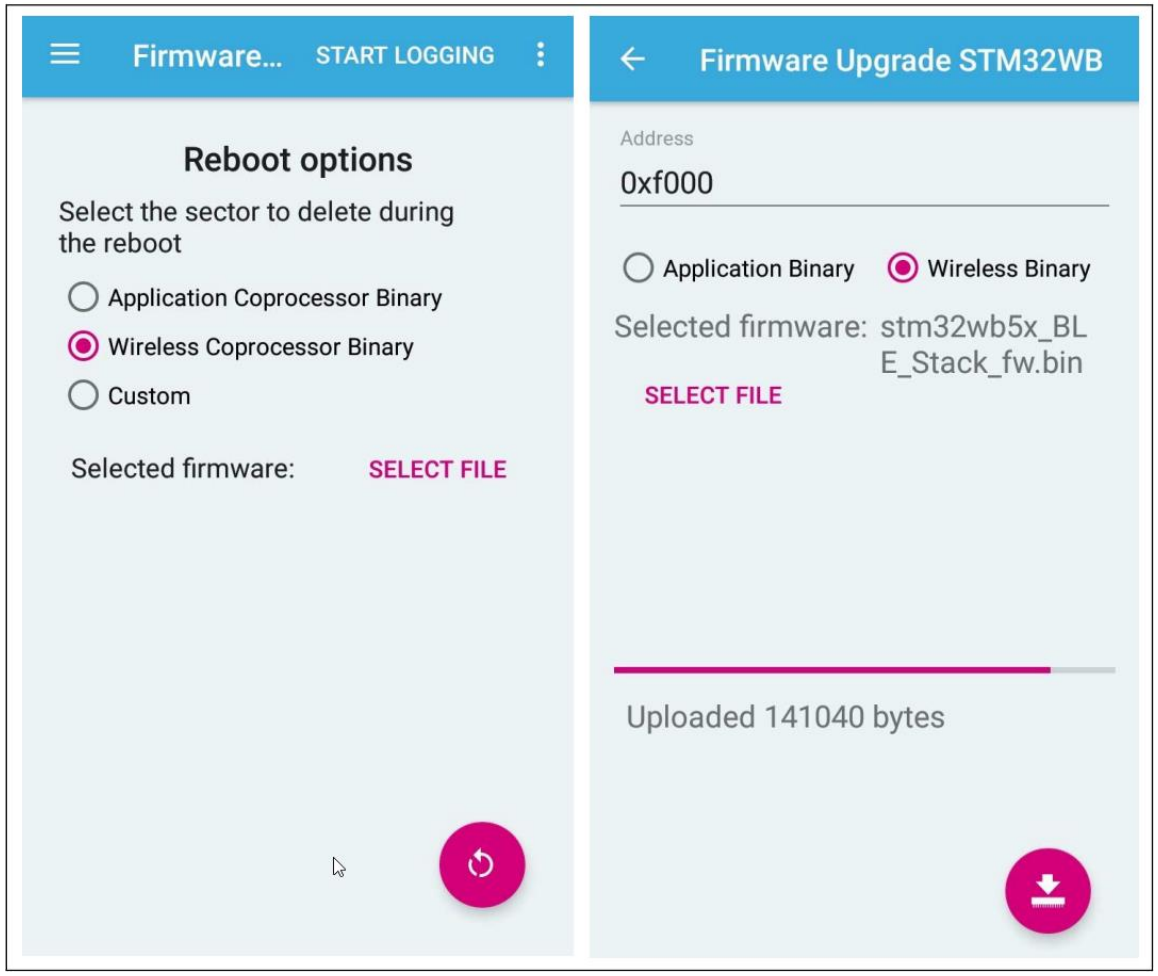
OTA升级：方式一 ---ST BLE Sensor



- 再次连接设备
- 选择应用偏移地址：0x7000
- 勾选应用程序
- 选择应用程序固件
- 点击下载

life.augmented

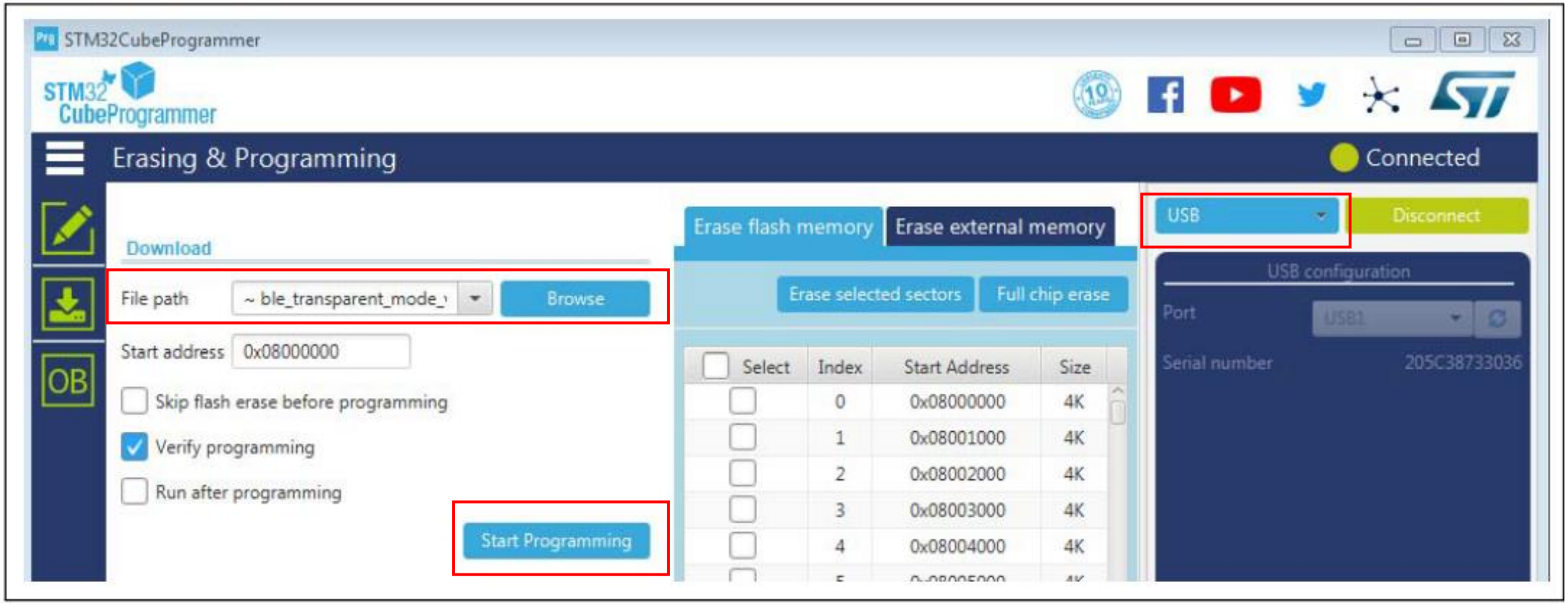
OTA升级：方式一
--- ST BLE Sensor



- 再次连接设备
- 选择协议偏移地址：0xf000
- 勾选无线协议栈
- 选择无线协议栈固件
- 点击下载

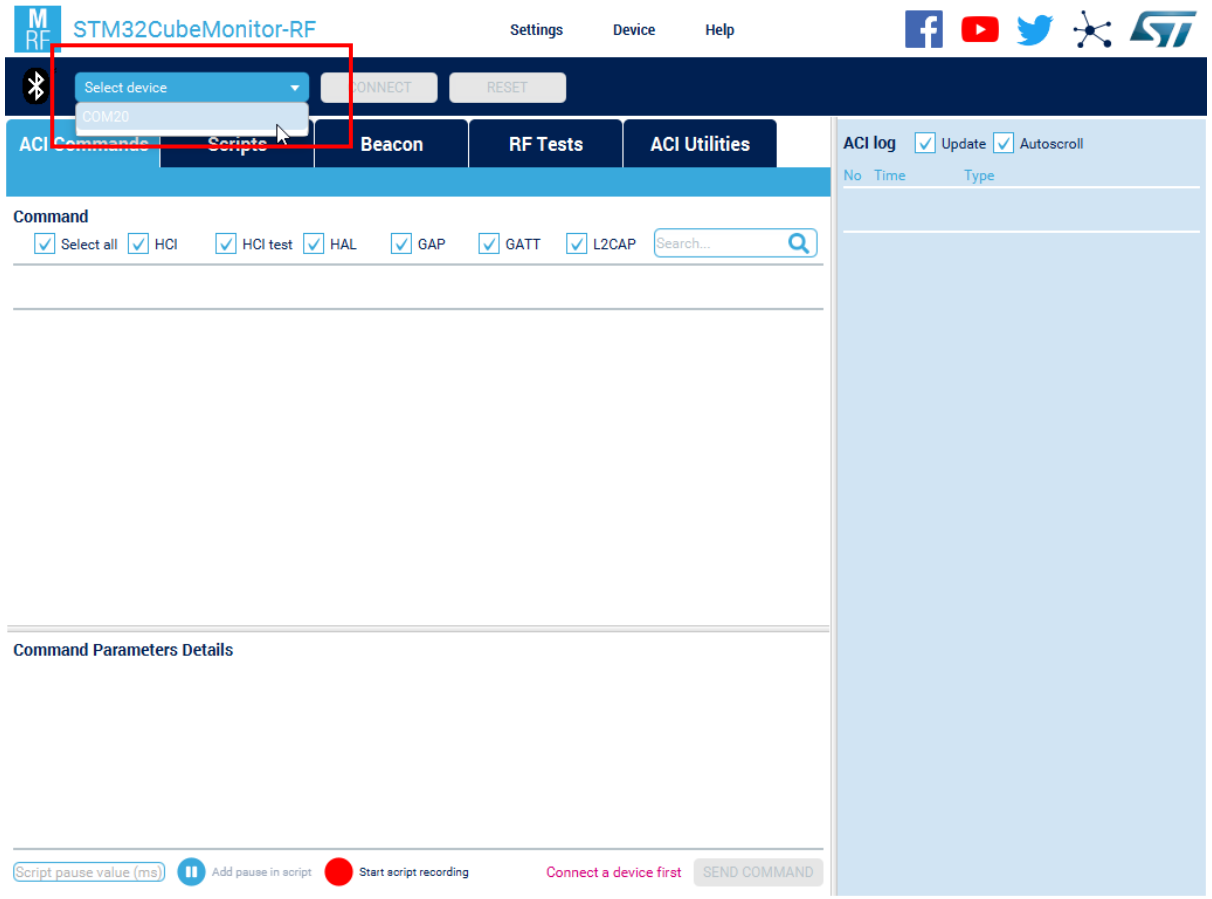
life.augmented

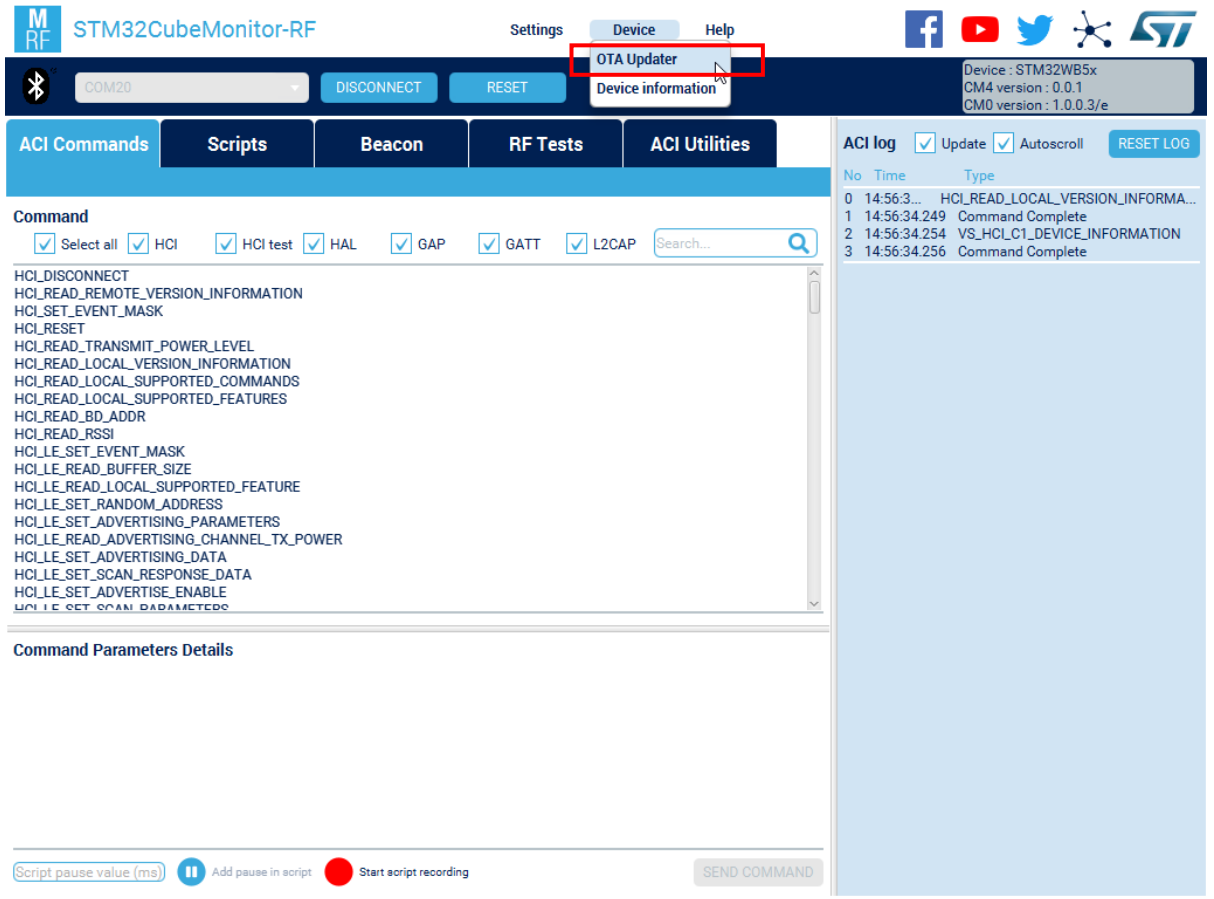
OTA升级：方式二
--- STM32CubeMonitor-RF（准备工作）



- USB Dongle SW2开关拨到位置1
- 将USB Dongle插入电脑USB口，使用USB连接
- 下载最新的Transparent mode 固件到USB Dongle

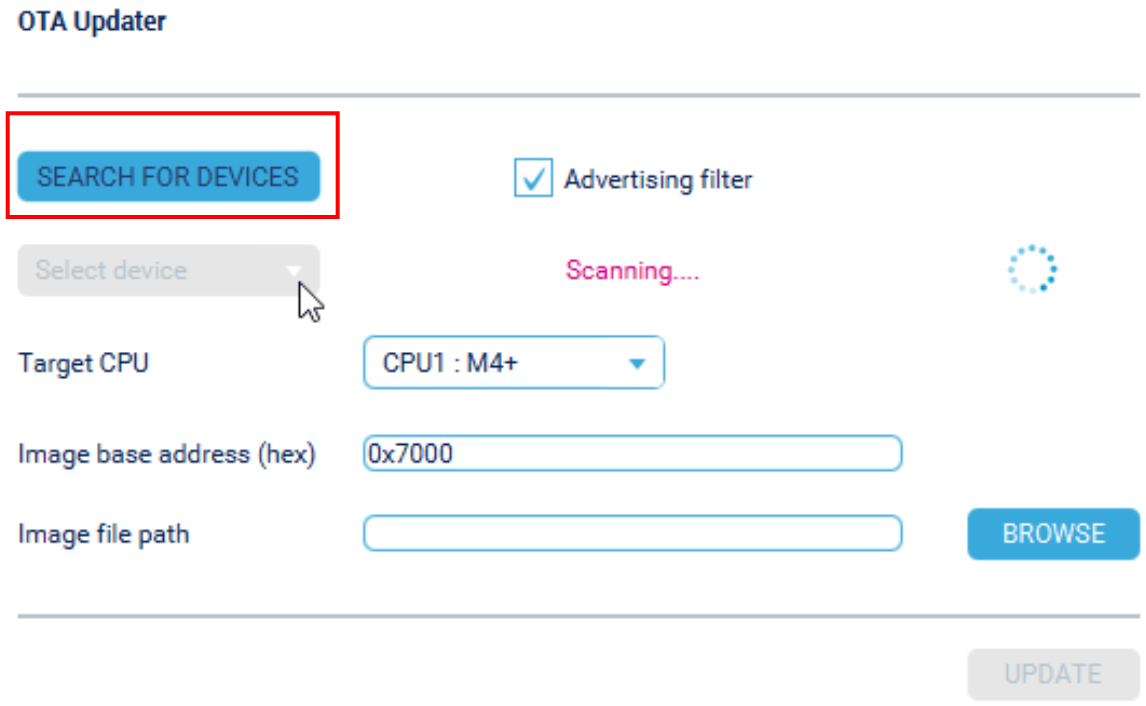
OTA升级：方式二
--- STM32CubeMonitor-RF（USB Dongle连接）

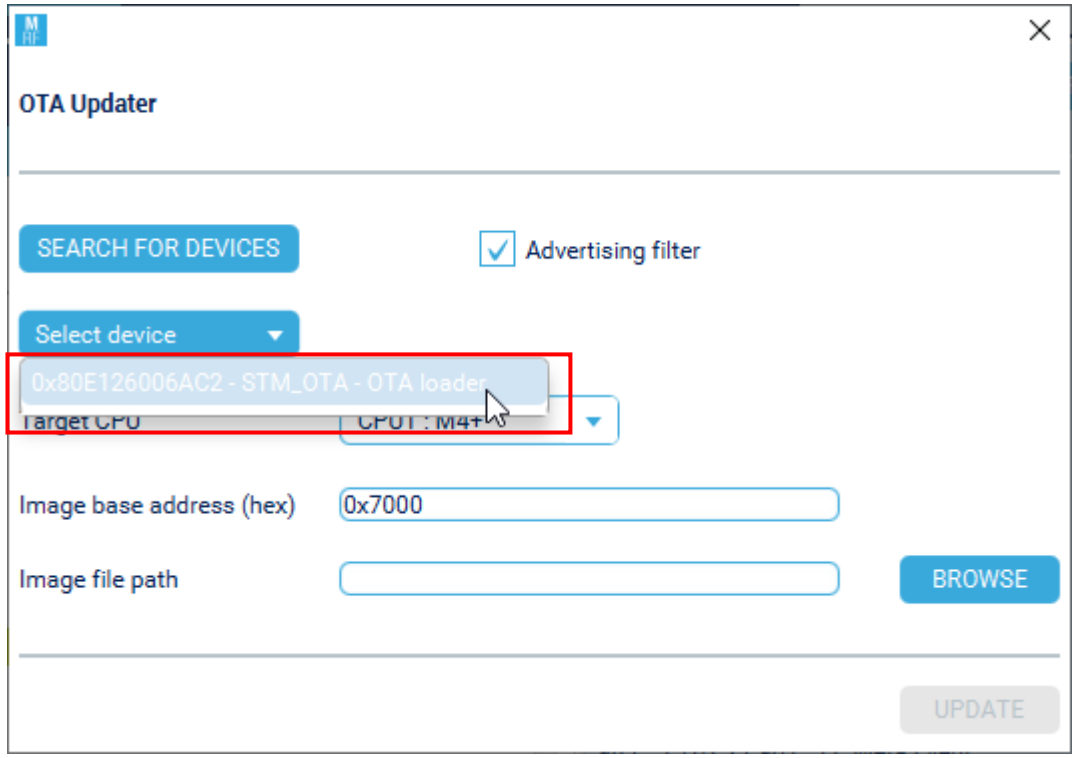




life.augmented

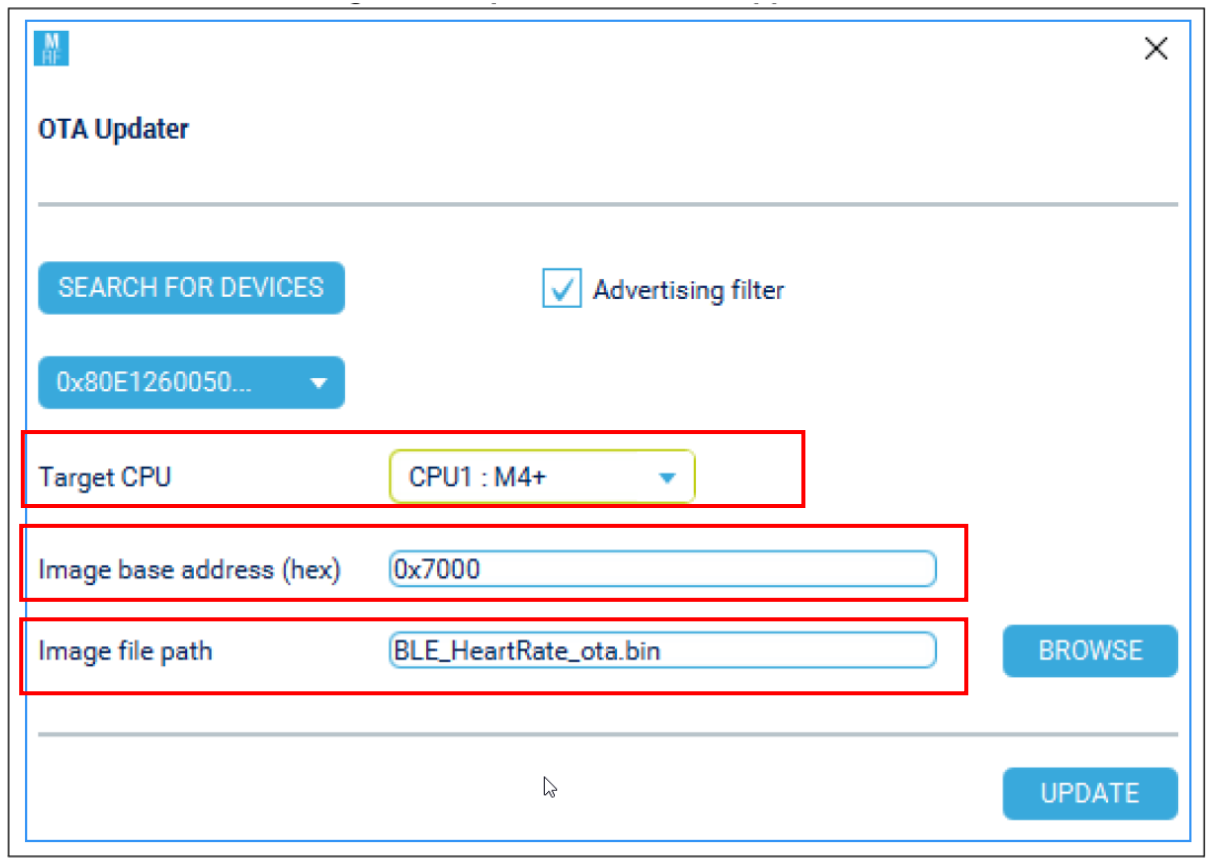
OTA升级：方式二
--- STM32CubeMonitor-RF（扫描和连接）





life.augmented

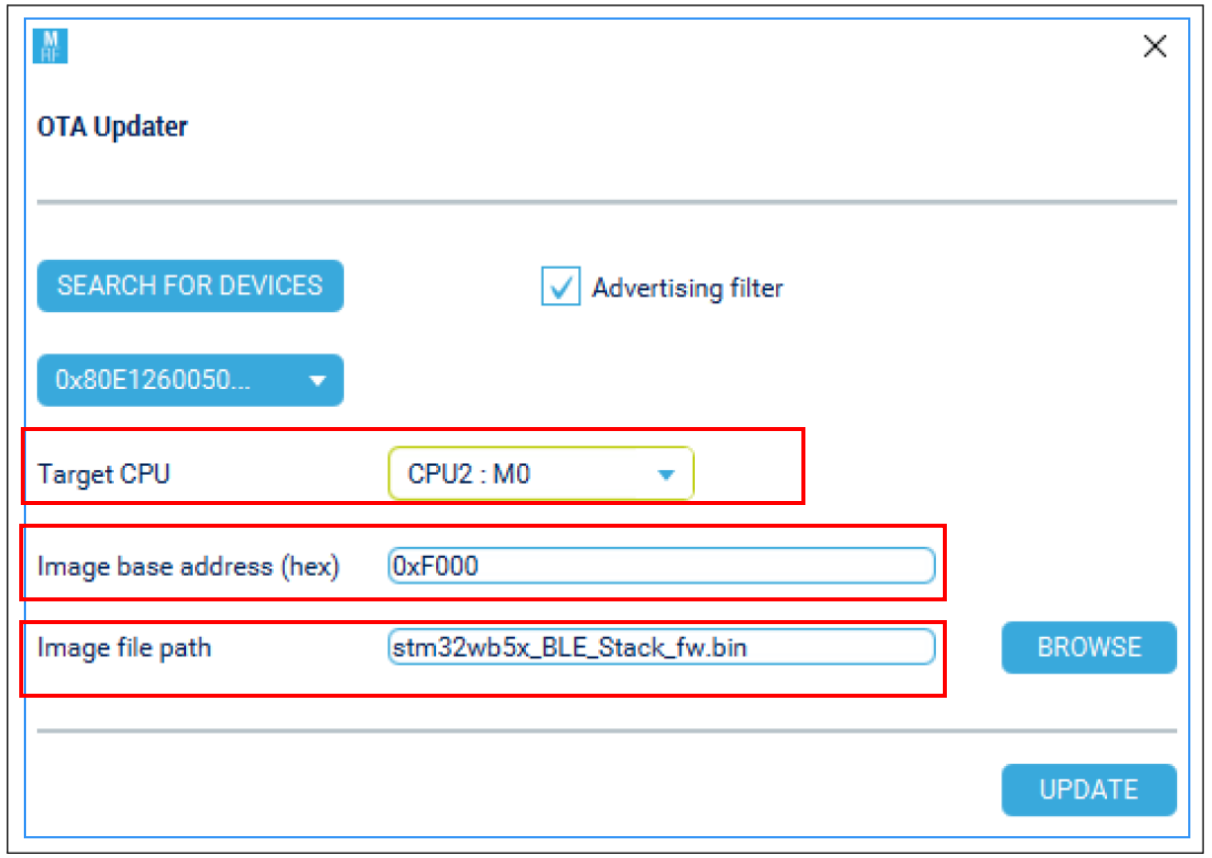
OTA升级：方式二
--- STM32CubeMonitor-RF（更新应用）



- 连接OTA设备
- 选择目标CPU：CPU1 M4+
- 选择基地址：0x7000
- 选择应用固件
- 点击UPDATE

life.augmented

OTA升级：方式二
--- STM32CubeMonitor-RF（更新协议栈）



- 连接OTA设备
- 选择目标CPU：CPU2 M0
- 选择基地址：0xF000
- 选择协议栈固件
- 点击UPDATE

life.augmented

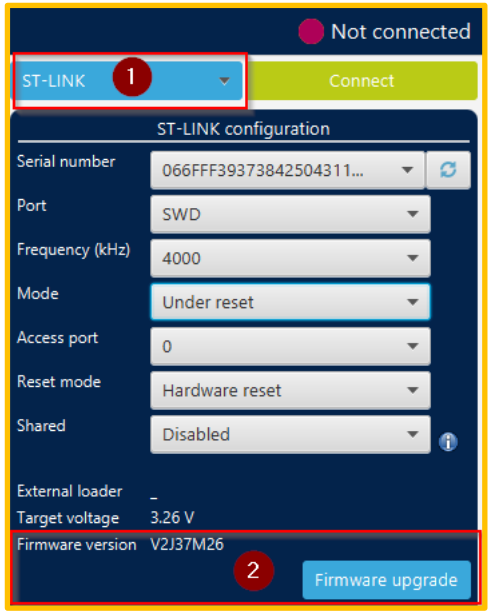
life.augmented

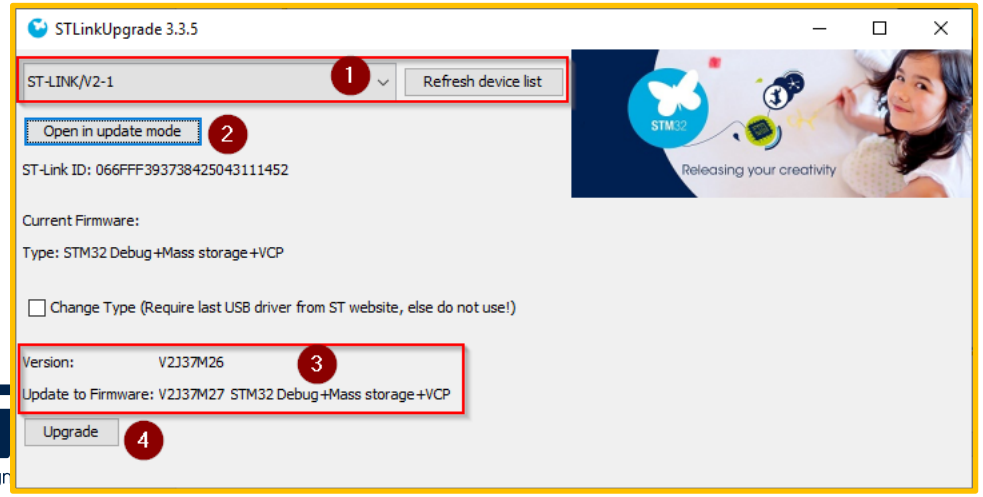
STM32WB固件升级之CubeProgGUI

STM32

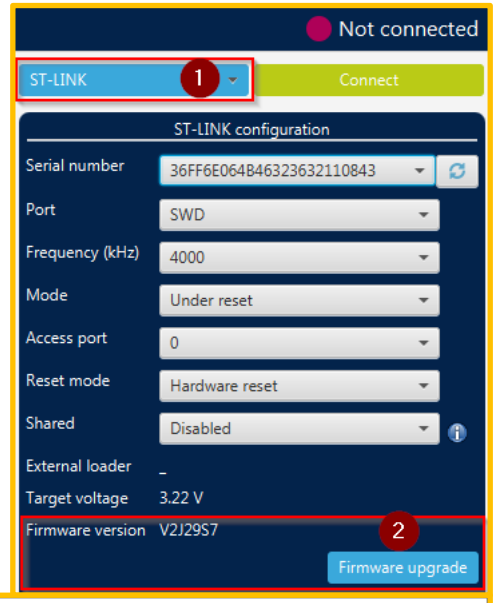
GUI通过SWD升级STLINK 固件

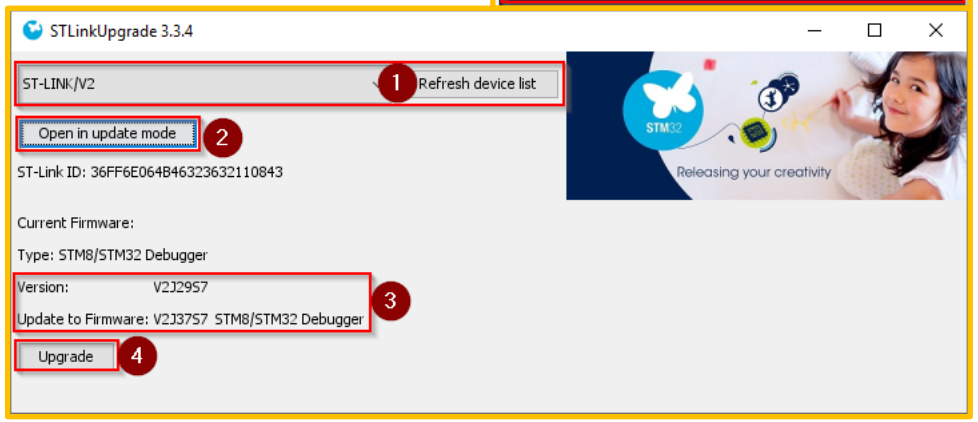
- Nucleo板载STLINKv2.1





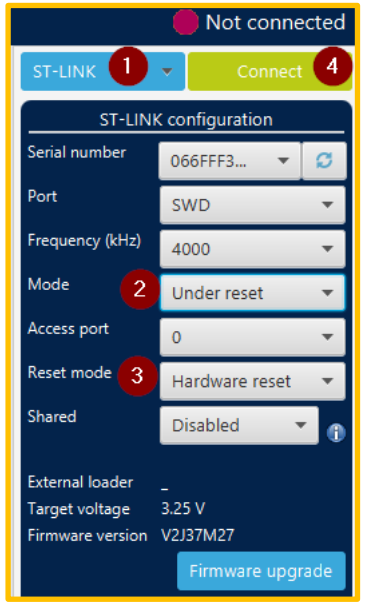
- STLINKv2

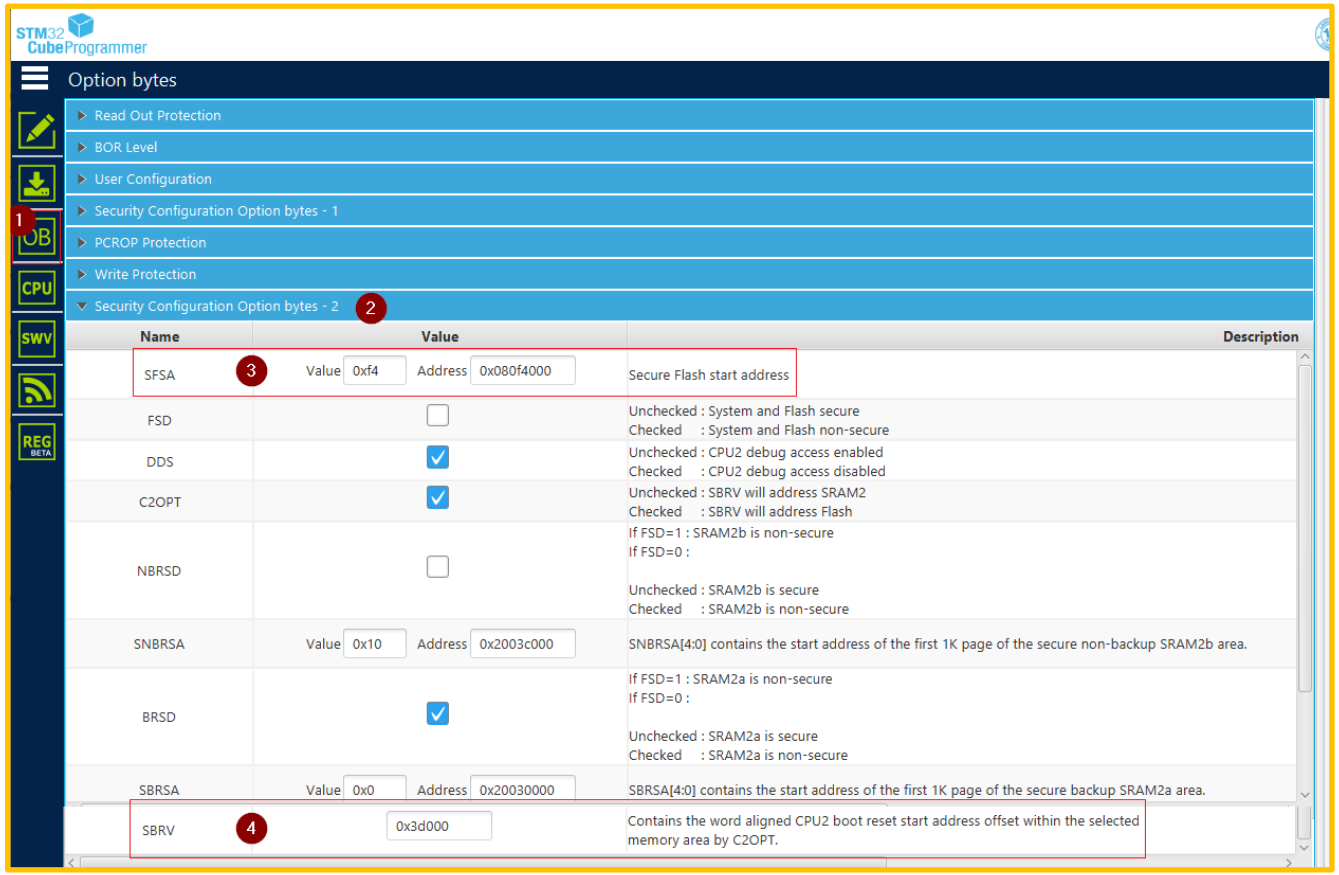




GUI查看Option Bytes SFSA & SBRV

M0+正在运行FUS	M0+正在运行stack
SBRV==0x3D800 (0x3D800*4==F6000) => M0+正在运行FUS v0.5.3 SBRV==0x3D000 (0x3D000*4==F4000) => M0+正在运行FUS v1.0.1/v1.0.2/v1.1.0/v1.1.1/v1.1.2	SBRV != 0x3D800 (0x3D800*4==F6000) && SBRV != 0x3D000 (0x3D000*4==F4000) && SBRV*4 == SFSA*0x1000 => M0+正在运行stack





共享设备信息表DevInfoTable解析参考结构体

M0+ 运行FUS时

#pragma pack(1) typedef struct {	Offset (the Xth Bytes)	The Xth Words
uint32_t DeviceInfoTableValid; /*0xA94656B9:Device info table valid*/	0	1
uint8_t Reserved1; uint8_t LastFUSActiveState; uint8_t LastWirelessStackState; uint8_t CurrentWirelessStackType;	4	2
uint8_t SafeBootVersion_Branch_Build; uint8_t SafeBootVersion_SubVersion; uint8_t SafeBootVersion_Minor; uint8_t SafeBootVersion_Major;	8	3
/* Fus Info */ uint8_t FusVersion_Build:4; uint8_t FusVersion_Branch:4; uint8_t FusVersionSub; uint8_t FusVersionMinor; uint8_t FusVersionMajor; uint16_t FusMemorySizeFlash; /*< Multiple of 4K */ uint8_t FusMemorySizeSram2A; /*< Multiple of 1K */ uint8_t FusMemorySizeSram2B; /*< Multiple of 1K */	12 16	4 5
/* Wireless stack version */ uint8_t WirelessStackVersion_Build:4; uint8_t WirelessStackVersion_Branch:4; uint8_t WirelessStackVersion_Sub; uint8_t WirelessStackVersion_Minor; uint8_t WirelessStackVersion_Major; uint16_t WirelessStackMemorySizeFlash; /*< Multiple of 4K */ uint8_t WirelessStackMemorySizeSram2A; /*< Multiple of 1K */ uint8_t WirelessStackMemorySizeSram2B; /*< Multiple of 1K */	20 24	6 7
uint32_t BLEInfo; uint32_t ThreadInfo; uint32_t Reserved4; uint32_t UID64_32L; uint32_t UID64_32H; uint16_t DeviceID;	28 32 36 40 44 48	8 9 10 11 12 13
}DeviceInfoTable_t;		

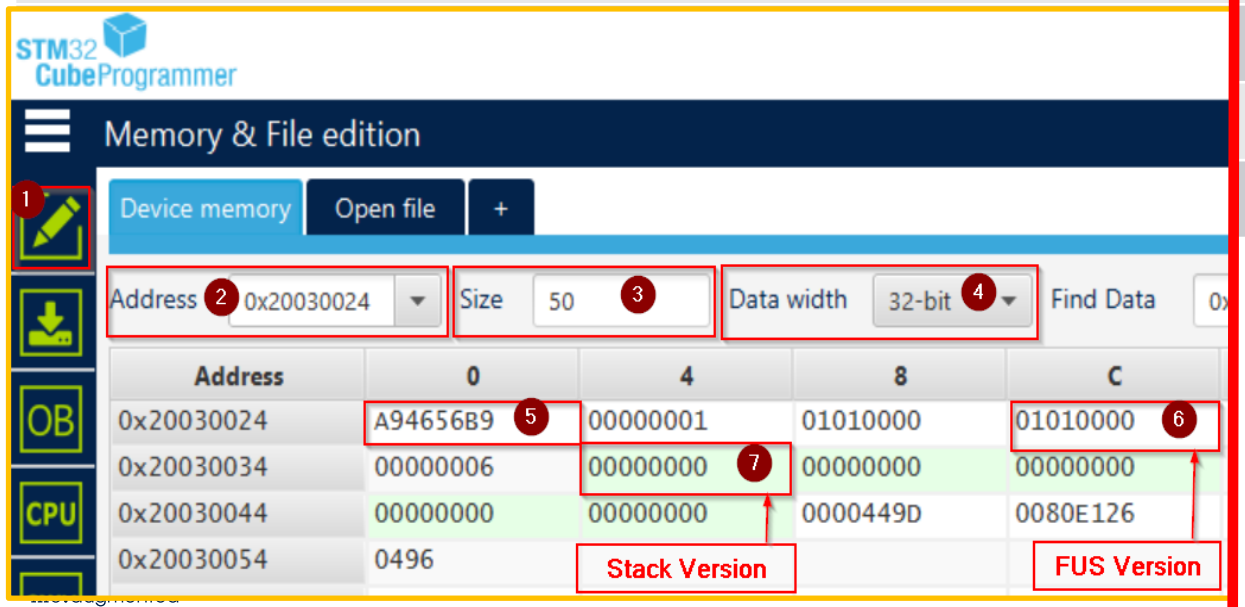
M0+ 运行Stack时

typedef struct {		Offset (the Xth Bytes)	The Xth Words
MB_SafeBootInfoTable_t SafeBootInfoTable;	typedef PACKED_STRUCT { uint32_t Version; } MB_SafeBootInfoTable_t;	0	1
MB_FusInfoTable_t FusInfoTable;	typedef PACKED_STRUCT { uint32_t Version; uint32_t MemorySize; uint32_t FusInfo; } MB_FusInfoTable_t;	4 8 12	2 3 4
MB_WirelessFwInfoTable_t WirelessFwInfoTable;	typedef PACKED_STRUCT { uint32_t Version; uint32_t MemorySize; uint32_t InfoStack; uint32_t Reserved; } MB_WirelessFwInfoTable_t;	16 20 24 28	5 6 7 8
} MB_DeviceInfoTable_t;			

M0+运行FUS和Stack时，设备信息表DevInfoTable解析结构体是不一样的!!!

查看FUS & Stack 版本理论方法

M0+ 运行FUS时	M0+ 运行Stack时
注意：如果M4运行的程序（如GPIO toggle）不能启动M0+，FUS & Stack版本是无法获取的！！！	
如果M4的程序不可以擦除时，参考以下方法从FUS 和STACK的offset处解析出FUS和Stack 版本信息和Flash Memory占用信息。	
如果将要升级FUS或Stack，或M4的程序可以擦除时，建议烧录FUS_Operator.bin并参考以上方法来查看FUS & Stack版本。本LAT使用FUS_Operator应用以上方法。	
STM32CubeProgrammer GUI 通过SWD 以Hot Plug模式和Software reset模式连接STM32WB，从0x20030000读取32bits 即一个Word,0x2003xxxx, 此数值是DevInfoTable存储的地址，在从此地址0x2003xxxx读取至少50Bytes的共享设备信息表(注意：M4中运行的程序不同可能0x2003xxxx也不一样!!!) 。	
如果0x2003xxxx的第一个Word的值是0xA94656B9，说明M0+运行的FUS，则参考AN5185的Table 5 Device information table来解析， 偏移12Bytes的word即第4个word的值 为FUS版本， 偏移为20Bytes的word值即第6个word为Stack的版本(Stack版本为0x00000000表示该WB为全新样品或已删除Stack)	



查看FUS & Stack 版本理论方法

M0+ 运行FUS时

注意：如果M4运行的程序（如GPIO toggle）不能启动M0+，FUS & Stack版本是无法获取的！！！

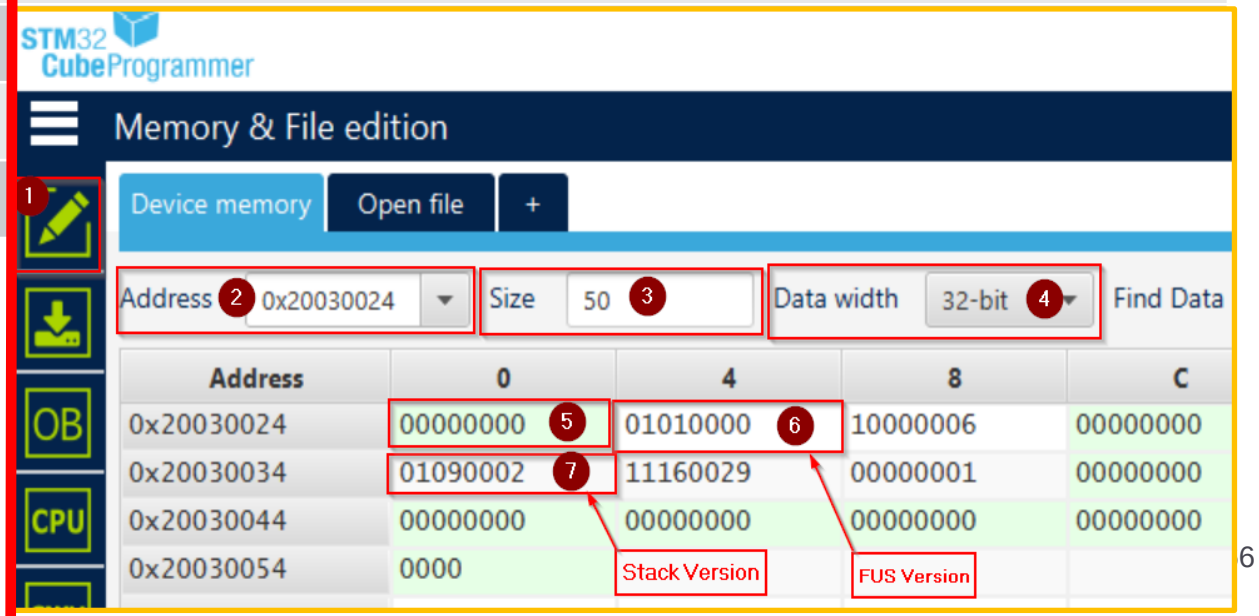
如果M4的程序不可以擦除时，参考以下方法从FUS 和STACK的offset处解析出FUS和Stack 版本信息和Flash Memory占用信息。

如果将要升级FUS或Stack，或M4的程序可以擦除时，建议烧录FUS_Operator.bin并参考以上方法来查看FUS & Stack版本。本LAT使用FUS_Operator应用以上方法。

STM32CubeProgrammer GUI 通过SWD 以Hot Plug模式和Software reset模式连接STM32WB，从0x20030000读取32bits 即一个Word,0x2003xxxx, 此数值是DevInfoTable存储的地址，在从此地址0x2003xxxx读取至少50Bytes的共享设备信息表(注意：M4中运行的程序不同可能0x2003xxxx也不一样!!!) 。

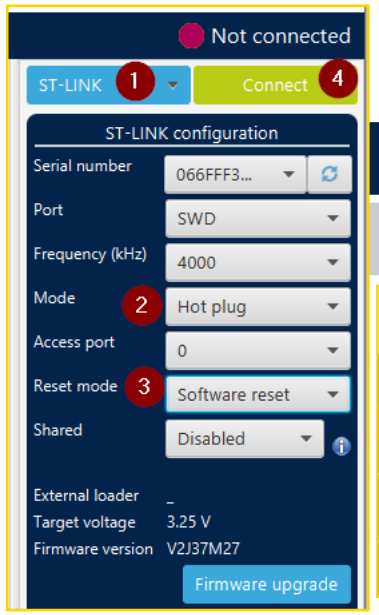
M0+ 运行Stack时

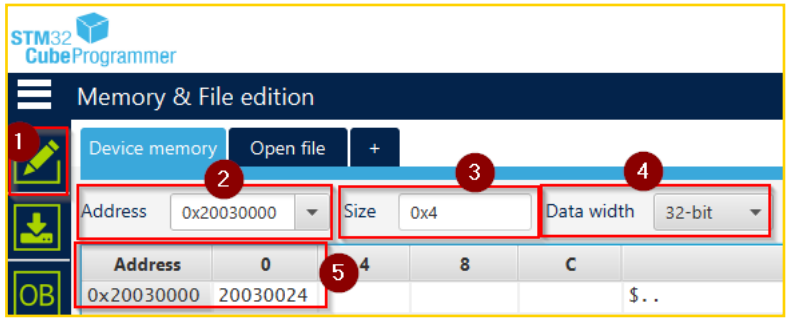
如果0x2003xxxx的第一个Word的值是0x00000000，说明M0+运行的Stack，则参考(STM32Cube_FW_WB_V1.x.0\Middlewares\ST\STM32_WPAN\interface\patterns\ble_thread\tl\mbox_def.h)中的 MB_DeviceInfoTable_t来解析，偏移4Bytes的word即第2个word的值 为FUS版本，偏移为16Bytes的word即第5个word值为Stack的版本。

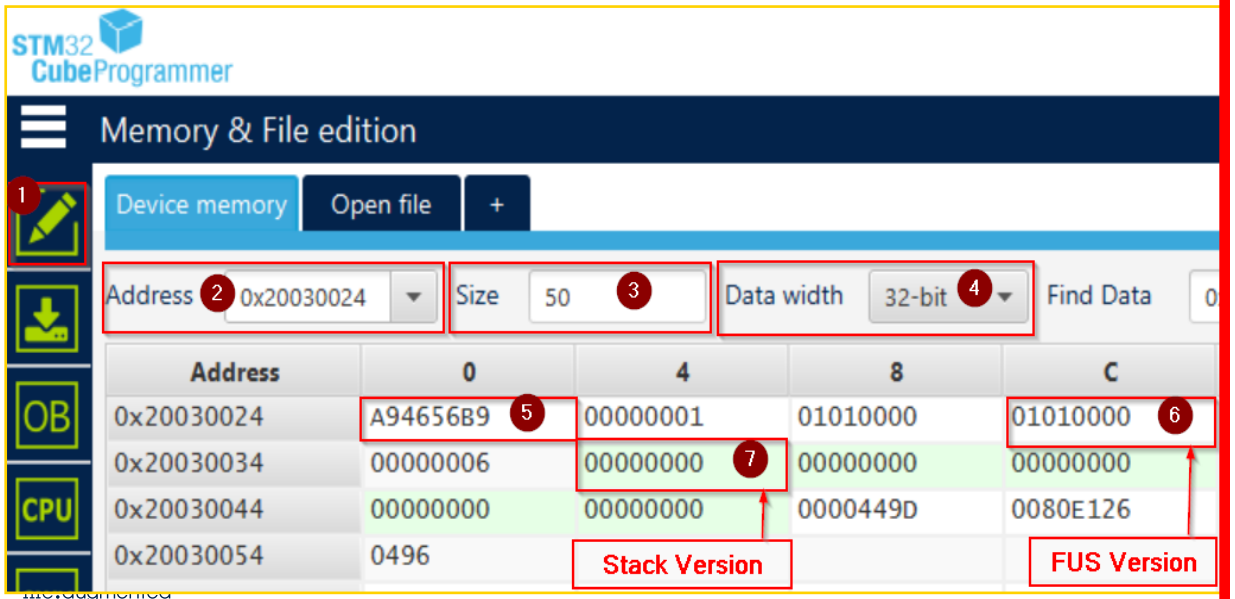


GUI升级前查看FUS & Stack 版本

M0+ 运行FUS时

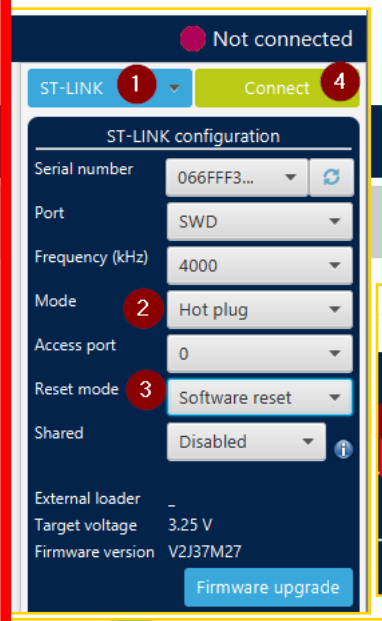


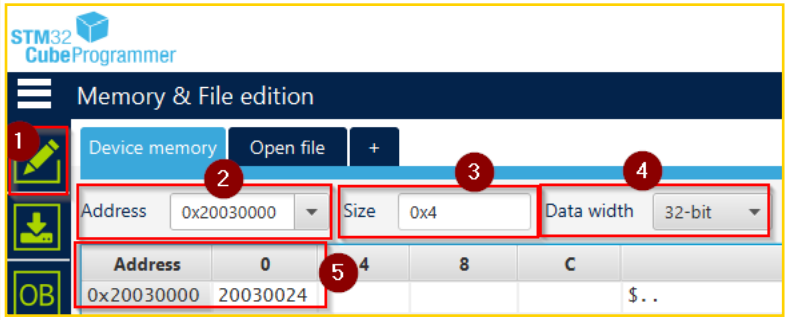


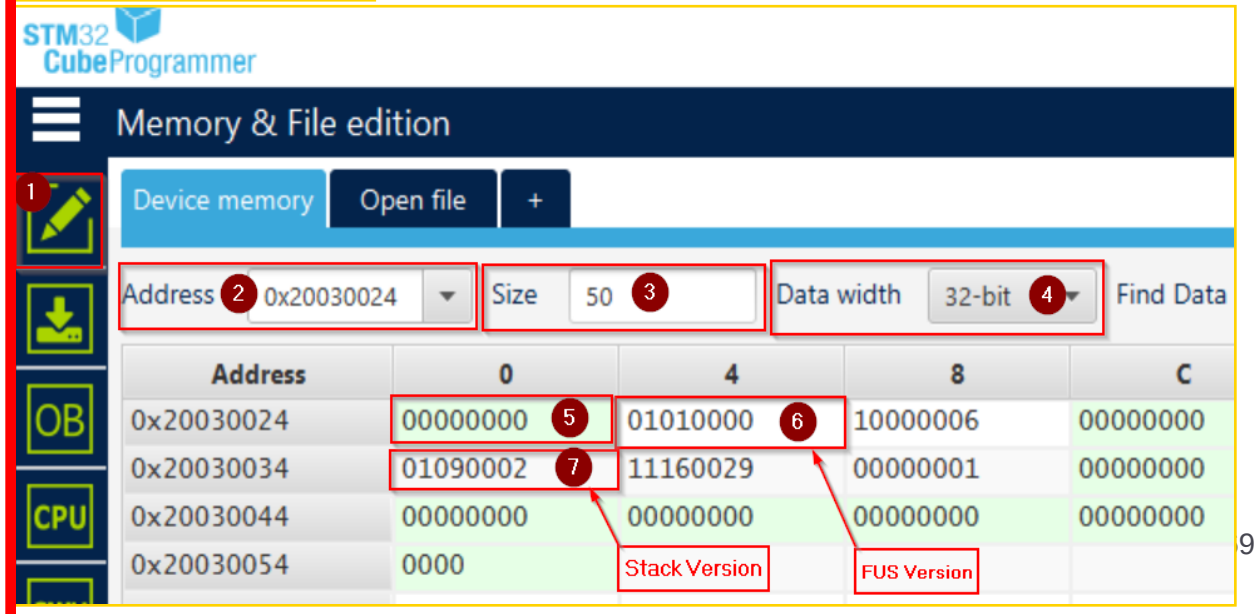


GUI升级前查看FUS & Stack 版本

M0+ 运行Stack时







FUS升级原则

- FUS版本差异请参考AN5185 Table 1. FUS versions

- ~~FUSv0.5.3 => FUSv1.0.1 可升级，不再推荐~~
- ~~FUSv0.5.3 => FUSv1.0.2 可升级，仅用过渡，不做最终再推荐~~
- FUSv0.5.3 => FUSv1.0.2 => FUSv1.1.0 可升级，推荐

- ~~FUSv1.0.1 => FUSv1.0.2 不支持不允许~~
- FUSv1.0.1 => FUSv1.1.0 可升级，推荐

- ~~FUSv1.0.2 => FUSv1.1.0 可升级，不再推荐~~
- FUSv1.0.2 => FUSv1.1.0 可升级，推荐

- ~~FUSv1.1.0 => FUSv1.1.2 不支持不允许，也无需升级，保持不升级~~
- ~~FUSv1.1.1 => FUSv1.1.2 不支持不允许，也无需升级，保持不升级~~
- ~~FUSv1.1.2 => FUSv1.1.2 不支持不允许，也无需升级，保持不升级~~

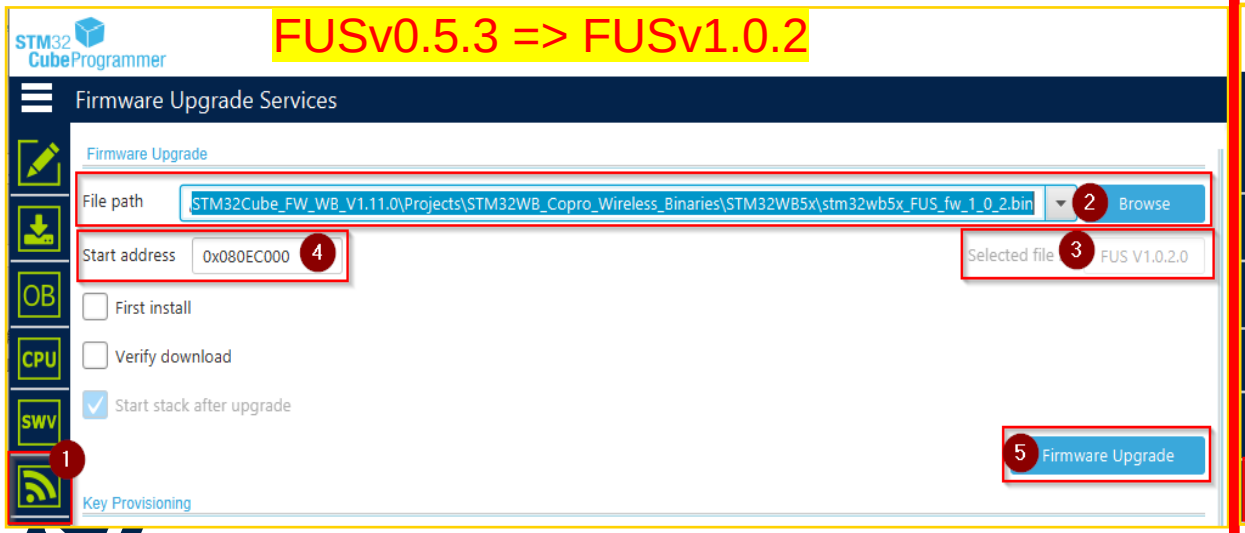
- FUSv0.5.3需要先升级到FUSv1.0.2，再升级到FUSv1.1.0
- FUSv1.0.1/FUSv1.0.2建议升级到FUSv1.1.0
- FUSv1.1.0/FUSv1.1.1不能也无需升级到FUSv1.1.0

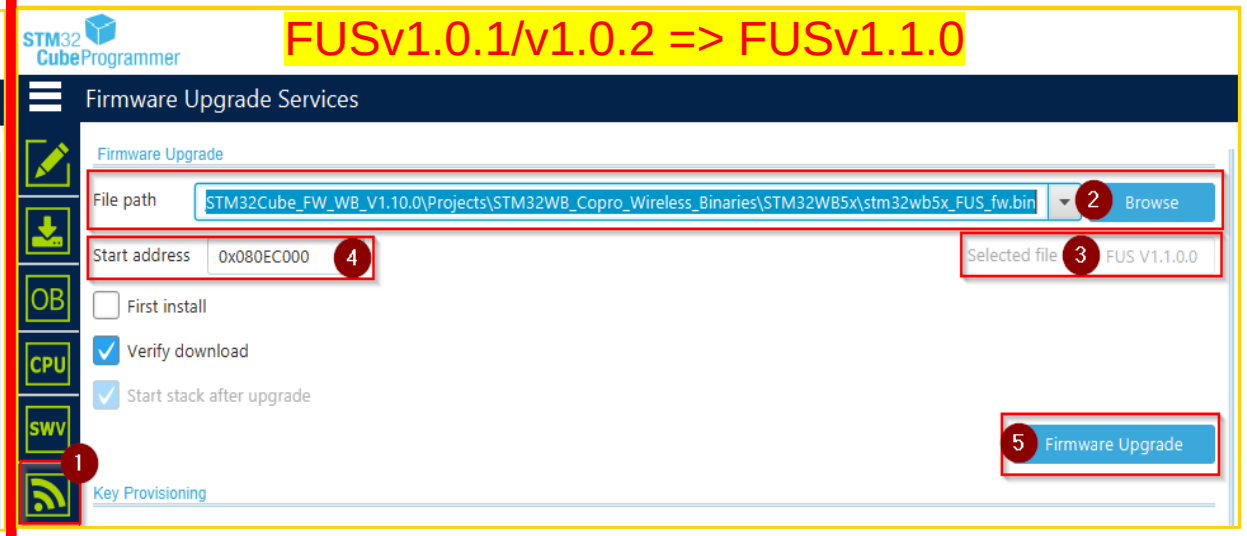
- FUS不要同版本重复升级

GUI升级FUS

- FUS文件路径STM32Cube_FW_WB_V1.x.x\Projects\STM32WB_Copro_Wireless_Binaries\STM32WB5x
- stm32wb5x_FUS_fw_1_0_2.bin
- stm32wb5x_FUS_fw.bin
- FUS的版本信息和针对不同STM32WBxx的烧写地址参考Release_Notes.html

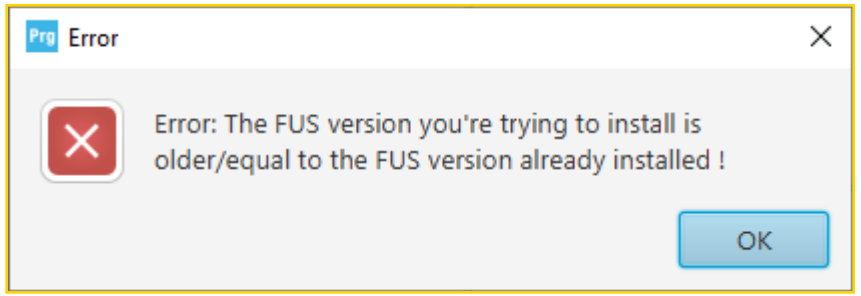
Wireless Coprocessor Binary	STM32WB5xxG(1M)	STM32WB5xxY(640k)	STM32WB5xxE(512K)	STM32WB5xxC(256K)	Version	Date
stm32wb5x_FUS_fw_1_0_2.bin	0x080EC000	0x0809A000	0x0807A000	0x0803A000	v1.0.2	04/10/2019
stm32wb5x_FUS_fw.bin	0x080EC000	0x0809A000	0x0807A000	0x0803A000	v1.1.2	02/05/2021





GUI不升级FUSv1.1.0/v1.1.1/v1.1.2

- Error: The FUS version you're trying to install is older/equal to the FUS version already installed !



Stack安装原则

- STM32WB55/50的stack存放于STM32Cube_FW_WB_V1.x.x\Projects\STM32WB_Copro_Wireless_Binaries\STM32WB5x
- STM32WB35/30的stack存放于STM32Cube_FW_WB_V1.x.x\Projects\STM32WB_Copro_Wireless_Binaries\STM32WB3x
- STM32WB15/10的stack存放于STM32Cube_FW_WB_V1.x.x\Projects\STM32WB_Copro_Wireless_Binaries\STM32WB1x
-
- STM32WB5x和STM32WB3x目录下的stack是目前一样的只是重新命名了，但是烧写地址是不一样的!!!
- STM32WB1x的stack与STM32WB5x & STM32WB3x是不一样的，同样的烧写地址也不一致!!!
- 不同stack对应于不同STM32WBxx的烧写地址请参考相应目录下的Release_Notes.html
- 烧写stack的操作方法和FUS的升级操作方法是一样的，只是文件不同，下载地址不同！！！

STM32WB5x的stack

STM32Cube_FW_WB_V1.x.x\Projects\STM32WB_Copro_Wireless_Binaries\STM32WB5x\Release_Notes.html

Wireless Coprocessor Binary	STM32WB5xxG(1M)	STM32WB5xxY(640k)	STM32WB5xxE(512K)	STM32WB5xxC(256K)	Version	Date
stm32wb5x_BLE_HCILayer_fw.bin	0x080E0000	0x0808C000	0x0806C000	0x802C000	v1.11.0	02/05/2021
stm32wb5x_BLE_HCI_AdvScan_fw.bin	0x080EA000	0x08096000	0x08076000	0x8036000	v1.11.0	02/05/2021
stm32wb5x_BLE_LLD_fw.bin	0x080EC000	0x08098000	0x08078000	0x08038000	v1.11.0	02/05/2021
stm32wb5x_BLE_Mac_802_15_4_fw.bin	0x080B5000	0x08061000	0x08041000	0x08001000	v1.11.0	02/05/2021
stm32wb5x_BLE_Stack_full_fw.bin	0x080CA000	0x08076000	0x08056000	0x8016000	v1.11.0	02/05/2021
stm32wb5x_BLE_Stack_light_fw.bin	0x080D6000	0x08082000	0x08062000	0x8022000	v1.11.0	02/05/2021
stm32wb5x_BLE_Thread_dynamic_fw.bin	0x08070000	0x0801C000	0x00	0x00	v1.11.0	02/05/2021
stm32wb5x_BLE_Thread_static_fw.bin	0x08072000	0x0801E000	0x00	0x00	v1.11.0	02/05/2021
stm32wb5x_BLE_Zigbee_FFD_dynamic_fw.bin	0x08077000	0x08023000	0x08003000	0x00	v1.11.0	02/05/2021
stm32wb5x_BLE_Zigbee_FFD_static_fw.bin	0x08079000	0x08025000	0x08005000	0x00	v1.11.0	02/05/2021
stm32wb5x_BLE_Zigbee_RFD_dynamic_fw.bin	0x08086000	0x08032000	0x08012000	0x00	v1.11.0	02/05/2021
stm32wb5x_BLE_Zigbee_RFD_static_fw.bin	0x08087000	0x08033000	0x08013000	0x00	v1.11.0	02/05/2021
stm32wb5x_Mac_802_15_4_fw.bin	0x080E3000	0x0808F000	0x0806F000	0x0802F000	v1.11.0	02/05/2021
stm32wb5x_Phy_802_15_4_fw.bin	0x080DE000	0x0808A000	0x0806A000	0x0802A000	v1.11.0	02/05/2021
stm32wb5x_Thread_FTD_fw.bin	0x08098000	0x08044000	0x08024000	0x00	v1.11.0	02/05/2021
stm32wb5x_Thread_MTD_fw.bin	0x080AB000	0x08057000	0x08037000	0x00	v1.11.0	02/05/2021
stm32wb5x_Zigbee_FFD_fw.bin	0x080A7000	0x08053000	0x08033000	0x00	v1.11.0	02/05/2021
stm32wb5x_Zigbee_RFD_fw.bin	0x080B5000	0x08061000	0x08041000	0x08001000	v1.11.0	02/05/2021

life.augmented

STM32WB3x和STM32WB1x的stack

STM32Cube_FW_WB_V1.x.x\Projects\STM32WB_Copro_Wireless_Binaries\STM32WB3x\Release_Notes.html

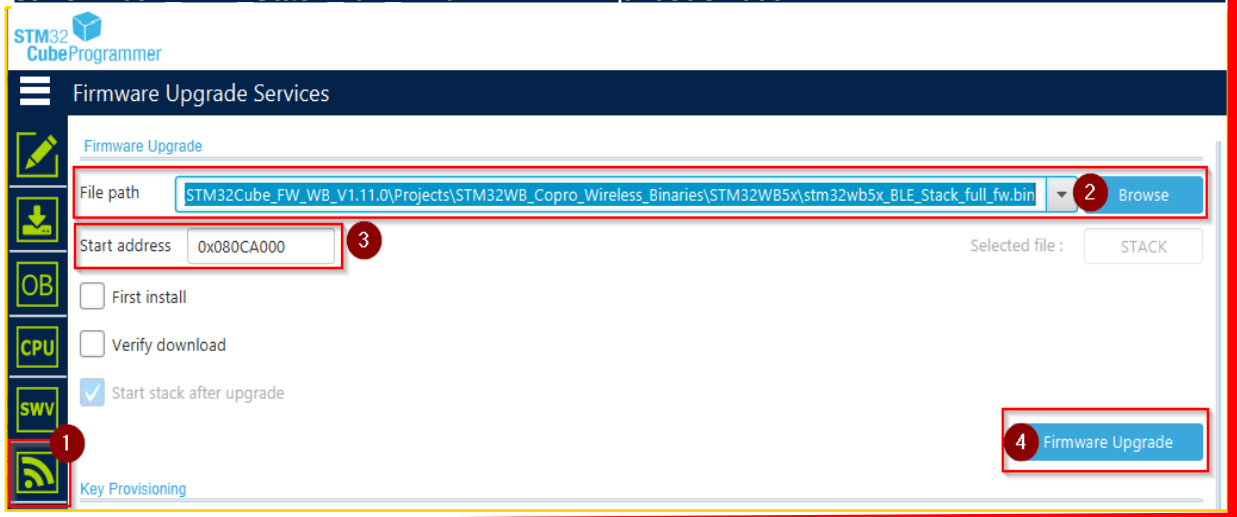
Wireless Coprocessor Binary	STM32WB3x(512K)	Version	Date
stm32wb3x_BLE_HCILayer_fw.bin	0x08060000	v1.11.0	02/05/2021
stm32wb3x_BLE_HCI_AdvScan_fw.bin	0x0806A000	v1.11.0	02/05/2021
stm32wb3x_BLE_LLD_fw.bin	0x0806C000	v1.11.0	02/05/2021
stm32wb3x_BLE_Mac_802_15_4_fw.bin	0x08035000	v1.11.0	02/05/2021
stm32wb3x_BLE_Stack_full_fw.bin	0x0804A000	v1.11.0	02/05/2021
stm32wb3x_BLE_Stack_light_fw.bin	0x08056000	v1.11.0	02/05/2021
stm32wb3x_Mac_802_15_4_fw.bin	0x08063000	v1.11.0	02/05/2021
stm32wb3x_Phy_802_15_4_fw.bin	0x0805E000	v1.11.0	02/05/2021
stm32wb3x_Thread_FTD_fw.bin	0x08018000	v1.11.0	02/05/2021
stm32wb3x_Thread_MTD_fw.bin	0x0802B000	v1.11.0	02/05/2021
stm32wb3x_Zigbee_FFD_fw.bin	0x08027000	v1.11.0	02/05/2021
stm32wb3x_Zigbee_RFD_fw.bin	0x08035000	v1.11.0	02/05/2021

STM32Cube_FW_WB_V1.x.x\Projects\STM32WB_Copro_Wireless_Binaries\STM32WB1x\Release_Notes.html

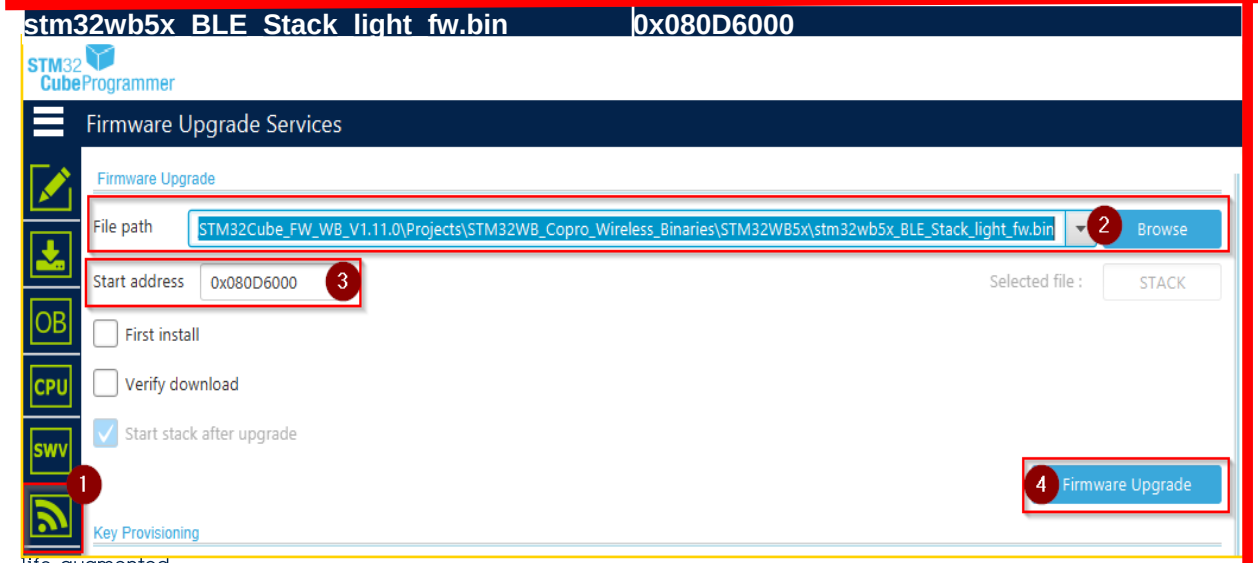
Wireless Coprocessor Binary	stm32wb1x(320K)	Version	Date
stm32wb1x_BLE_HCILayer_fw.bin	0x08032000	v1.11.0	12/02/2021
stm32wb5x_BLE_HCI_AdvScan_fw.bin	0x0803B800	v1.11.0	12/02/2021
stm32wb1x_BLE_LLD_fw.bin	0x0803E000	v1.11.0	12/02/2021
stm32wb1x_BLE_Stack_full_fw.bin	0x0801B800	v1.11.0	12/02/2021
stm32wb1x_BLE_Stack_light_fw.bin	0x08027800	v1.11.0	12/02/2021

GUI安装BLE_Stack_full

stm32wb5x_BLE_Stack_full_fw.bin	0x080CA000

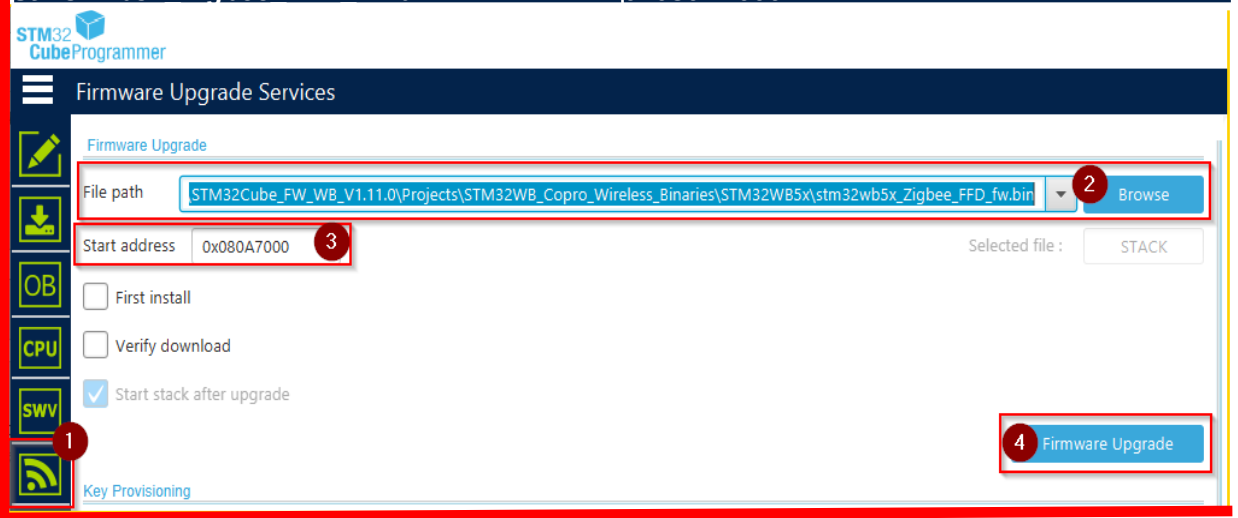


GUI安装BLE_Stack_light

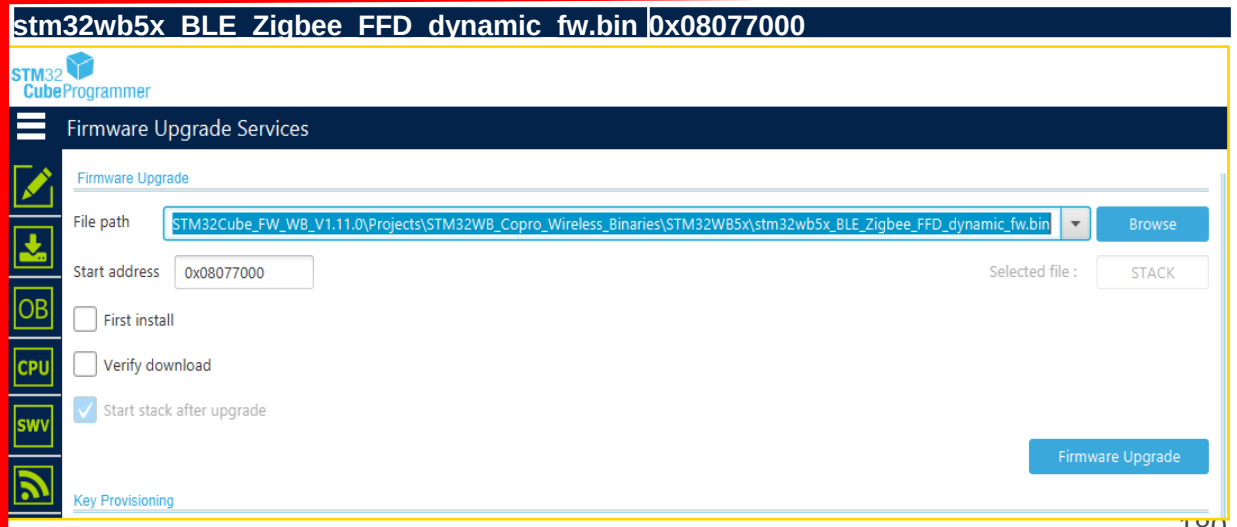


GUI安装Stack Zigbee_FFD

stm32wb5x_Zigbee_FFD_fw.bin	0x080A7000



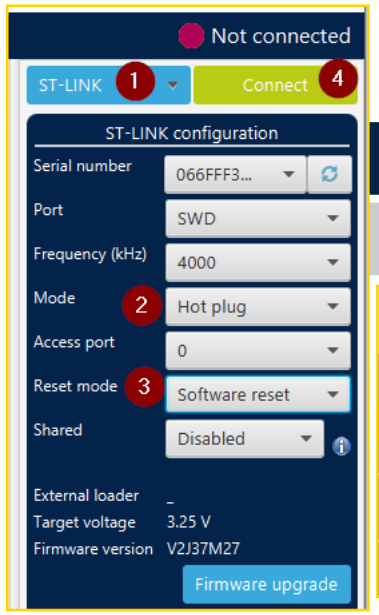
GUI安装Stack BLE_Zigbee_FFD_dynamic

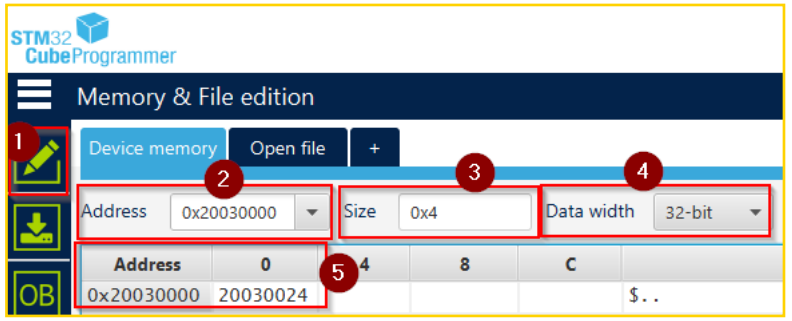


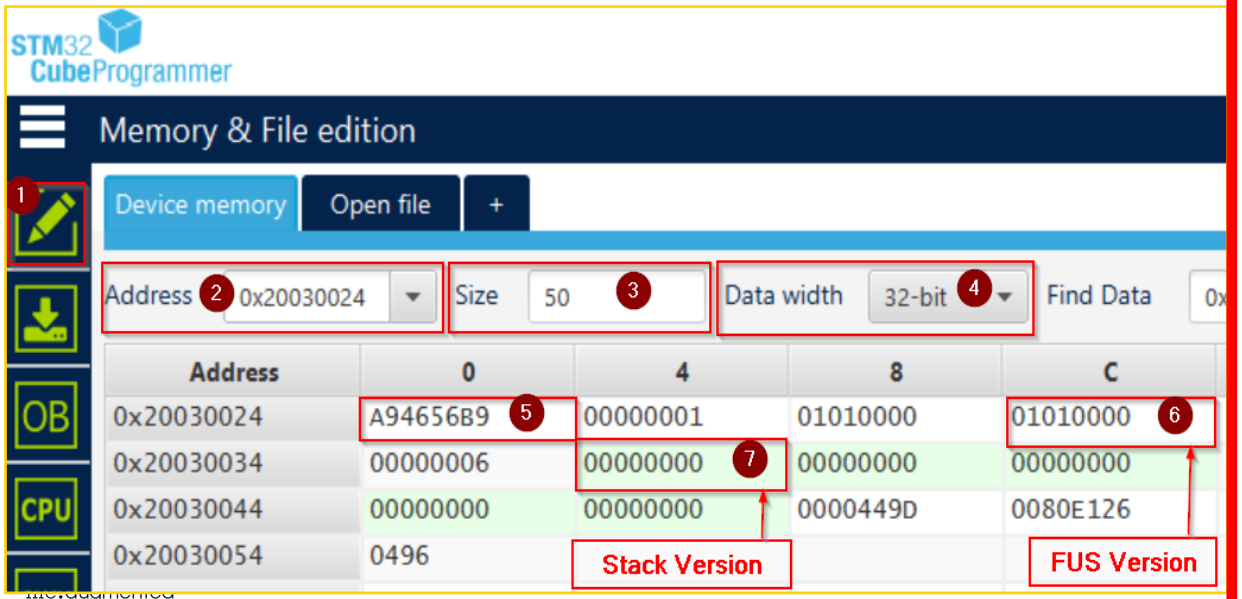
life.augmented

GUI升级后查看FUS & Stack 版本
同 “GUI升级前查看FUS & Stack 版本”

M0+ 运行FUS时

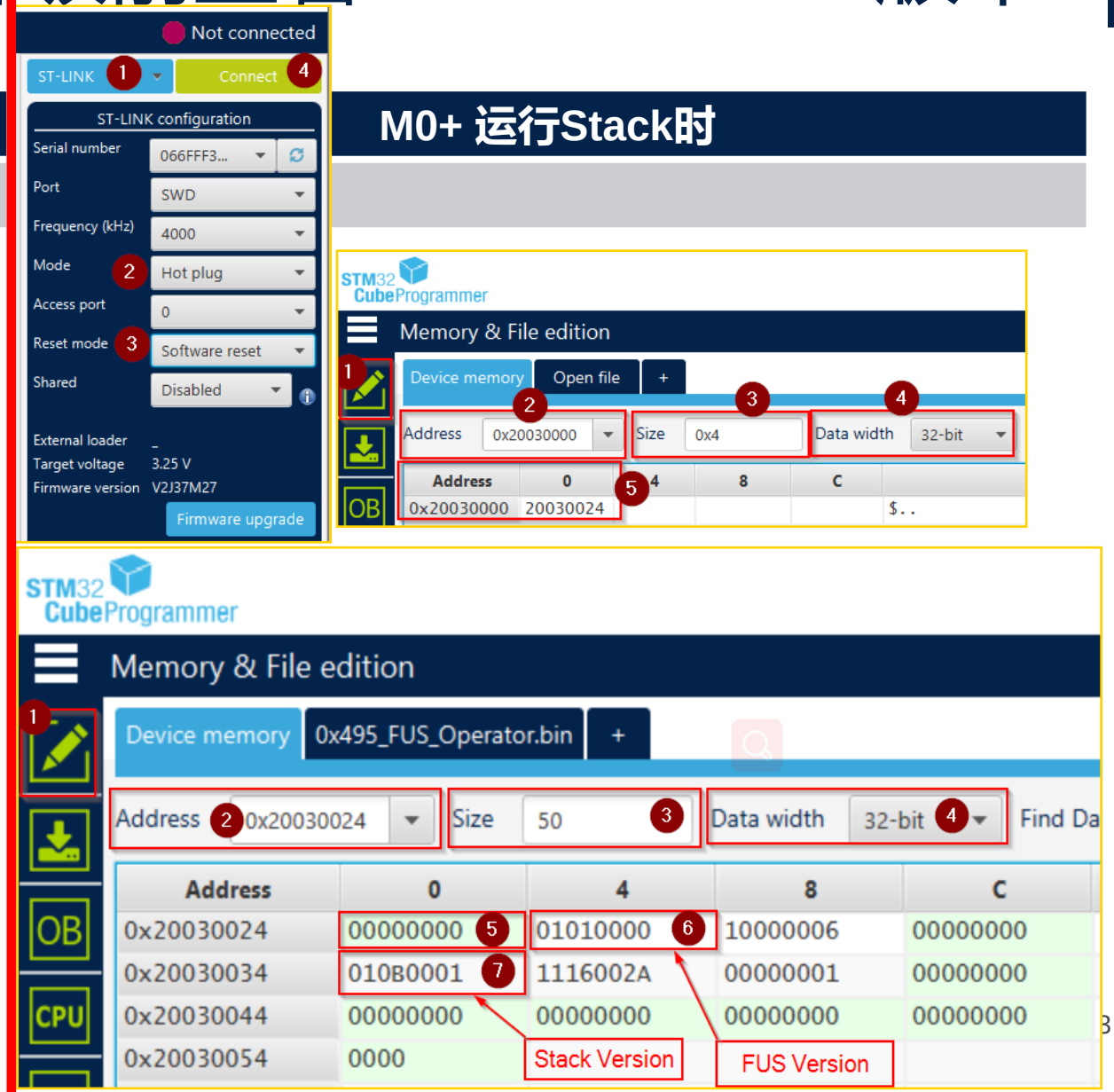






GUI升级后查看FUS & Stack 版本
同 “GUI升级前查看FUS & Stack 版本”

M0+ 运行Stack时



life.augmented

Thank you