



低功耗蓝牙技术(BLE)



2013年5月

主要内容

- ◆ **BLE概述**
- ◆ **低功耗蓝牙协议栈**
- ◆ **CC2540DK-MINI Kit 介绍**
- ◆ **BLE典型应用**
- ◆ **信驰达IOS APP介绍**
- ◆ **BLE通讯演示与实践**

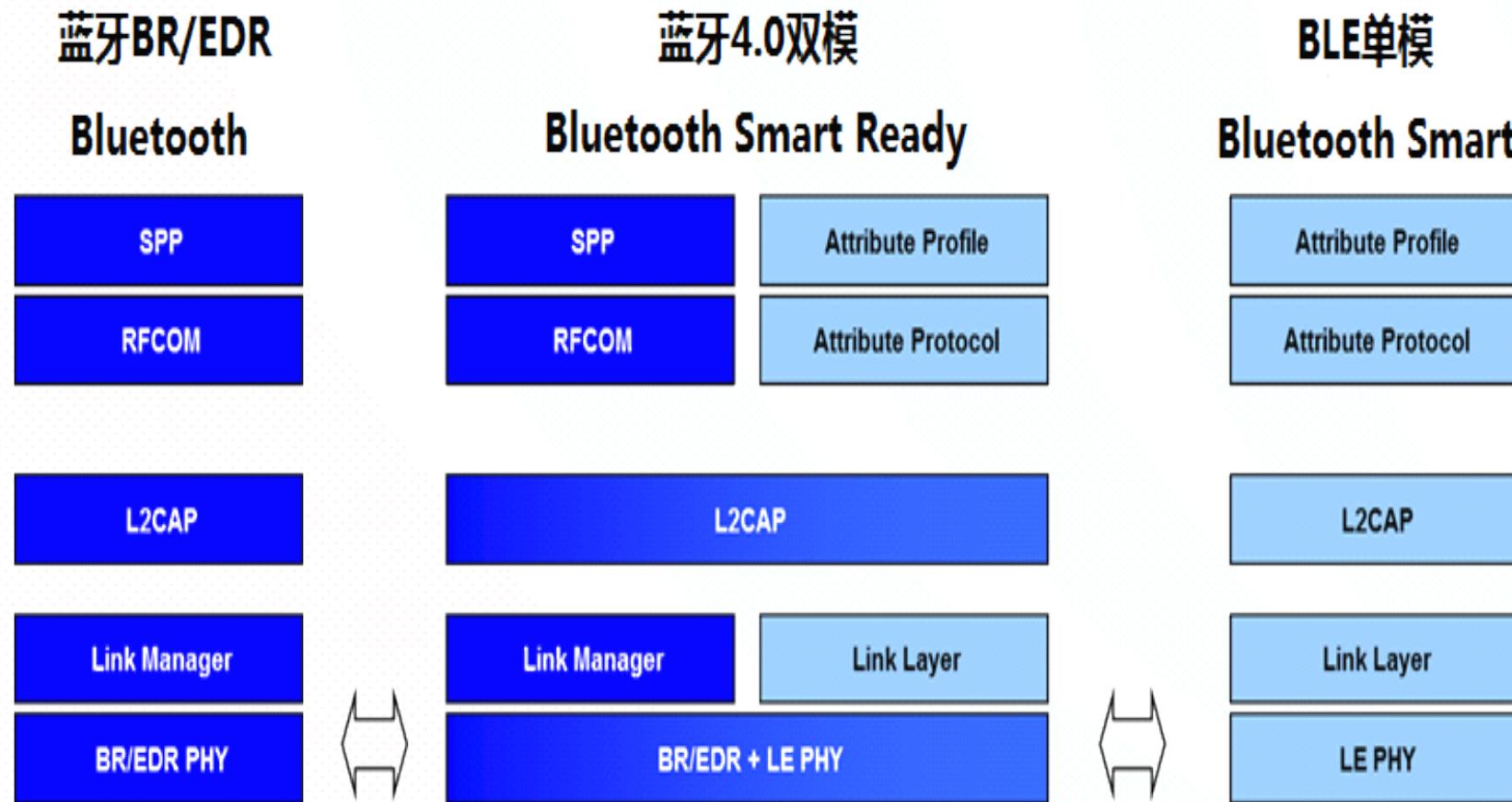
BLE概述

(Bluetooth Low Energy)

什么是低功耗蓝牙？

- ◆ 国际蓝牙联盟（BT-SIG，TI是企业成员之一）通过的一个标准蓝牙无线协议。
- ◆ 主要的新特性是在蓝牙标准版本上添加了4.0蓝牙规范(2010年6月)
- ◆ 针对无线应用程序与低功耗, 低延迟, 小数据包的传输需求
- ◆ 主要是围绕手机和个人电脑系统, 但也可以用于其他应用程序（设计苹果外围无需MFI认证）
- ◆ 预计在未来五年将有十亿的设备需求量
- ◆ 就单模而言和经典蓝牙设备不兼容

兼容性示意：



用途： (小包传输，手机扩展，低功耗)

- ◆ **2.4G蓝牙低功耗系统**
- ◆ 消费类电子产品
- ◆ 移动电话外围扩展设备
- ◆ 运动和休闲设备
- ◆ 健康医疗用品(血压计，体温计...)
- ◆ 汽车电子设备
- ◆ 人机接口设备(键盘，鼠标，遥控器...)
- ◆ **USB Dongle**

低功耗蓝牙协议栈 (Bluetooth Low Energy Protocol Stack)

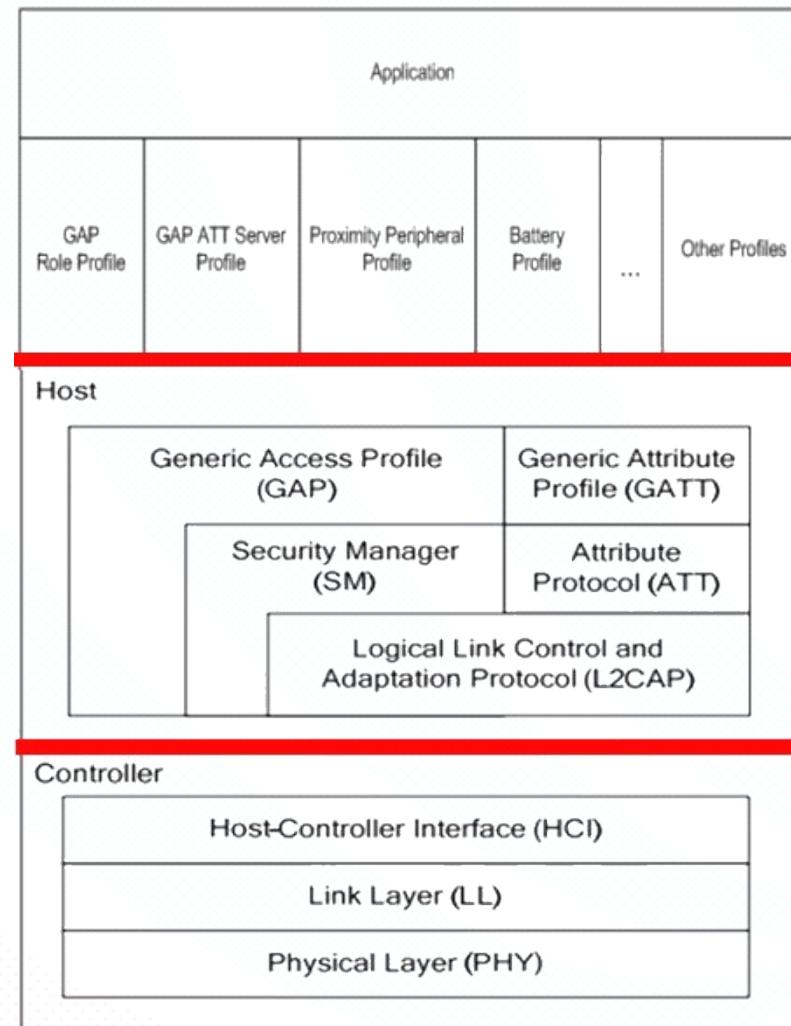


深圳信驰达科技

88-7

BLE协议栈的结构和配置：

- ◆ 协议栈有两部分组成：**Controller** 和 **Host**
- ◆ **Profiles** 和应用总是基于**GAP** 和**GATT** 之上
- ◆ 在**单芯片方案中**，**Controller** 和 **Host , profiles**, 和应用层都 在同一片芯片中
- ◆ 在**网络控制器模式**中，**host** 和 **controller** 是在一起运行的，但是应用和**profiles** 在另外一个器件上，比如PC或者其他微控制器，可以通过**UART, USB** 进行操作
- ◆ 在**双芯片模式**中，**Controller** 运行在一个控制器，而应用层，**profiles**和**Host** 是运行在另外一个控制器上
- ◆ **CC2540** 能够支持以上的任何配置



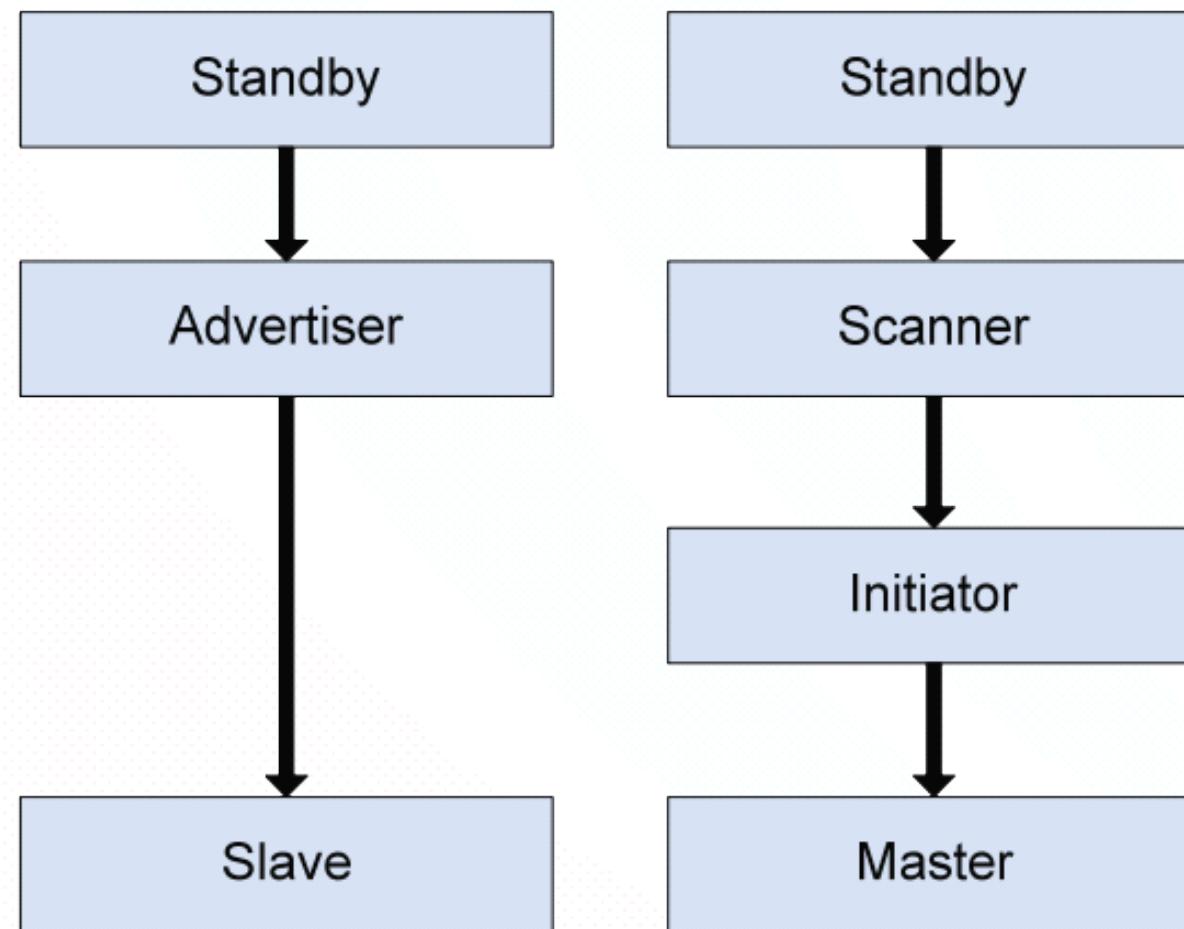
BLE:物理层(PHY)

- ◆ RF 规格特性
 - 运行在 **2.4 GHz ISM band**
 - **GFSK** 调制方式（高斯频移键控）
 - **40 频道2 MHz** 的通道间隙
 - 3 个固定的广播通道
 - 37 个自适应自动跳频数据通道
- ◆ 物理层可以和经典蓝牙RF组合成双模设备
- ◆ **2 MHz** 间隙能更好地防止相邻频道的干扰

BLE:拓扑结构和设备状态

- ◆ BLE 是一种星形拓扑结构:
 - 主设备管理着连接，并且可以连接多个从设备
 - 一个从设备只能连接一个主设备
- ◆ 做为一个BLE设备，有六种可能的状态:
 - 待机状态(**Standby**): 设备没有传输和发送数据，并且没有连接到任何设备
 - 广播状态(**Advertiser**): 周期性广播状态
 - 扫描状态(**Scanner**): 主动地寻找正在广播的设备
 - 发起连接状态(**Initiator**): 主动向某个设备发起连接
 - 主设备(**Master**): 作为主设备连接到其他设备
 - 从设备(**Slave**)：作为从设备连接到其他设备

BLE 连接状态流程图

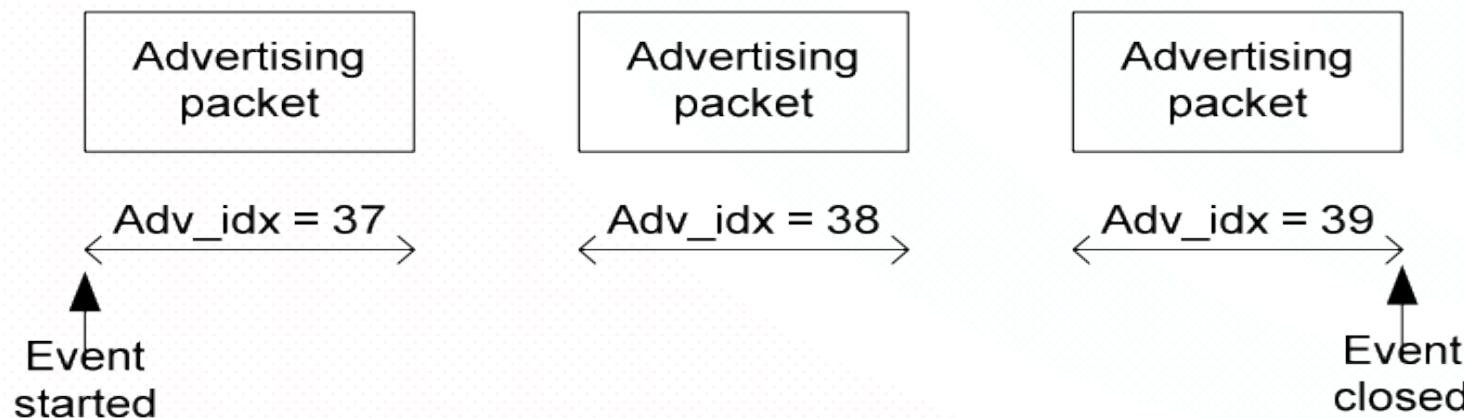


BLE 和 快递服务 类比

BLE		快递服务	
master	slave	某公司	派送/收件员
待机模式	待机模式	无快递需求	失业中
扫描模式	广播模式	寻找快递公司	上门咨询需求
扫描请求	扫描回应	咨询能提供哪些服务(西藏送不送? 能否代收货款? 等)	告之服务(业务范围, 收费标准, 等)
连接请求(连接参数)	—	签署业务合同(服务方式)	—
连接间隔	(如: 1秒1次)	取件频度	(如: 一天取一次)
潜伏期	(如: 2, 2秒1次)	如无收包事件(收取件), 是否同意间隔取件(寄出件)	(如: 隔一次取一次)
超时监管	(如: 5秒至少连1次)	需定时报到, 否则认为合作终止	(如: 至少一周取一次)
连接事件	参数更新请求	准备邮包, 等待取件	请求更改取件习惯
参数更新回应	—	同意更改(也可以不同意)	—
连接更新请求	—	重新签署业务合同(新服务方式)	—
连接事件	定时收包/发包	准备邮包, 等待取件	按时送包/取包
连接事件	...	准备邮包, 等待取件	...

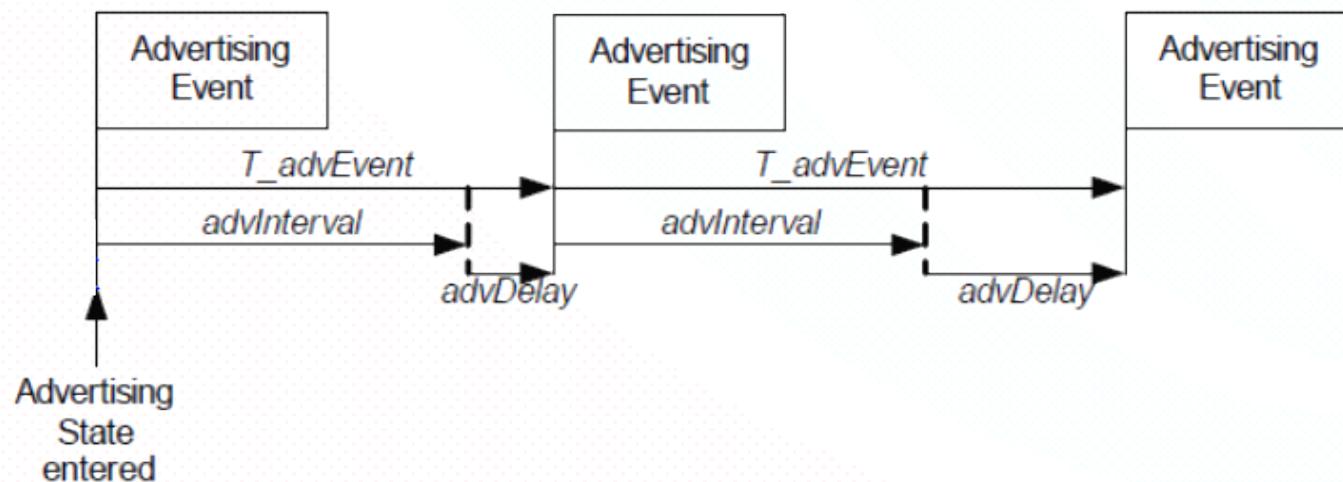
BLE: 广播事件

- ◆ 广播包的发送是单向的，不需要任何连接。
- ◆ 设备发送广播包进入广播状态
 - 广播包可以包含特定的数据定义，最大**31**个字节
 - 广播包可以直接指向某个**特定的**设备，也可以不指定
 - 广播中可以声明是可被连接的设备，或者是不可连接的设备
- ◆ 在一个广播事件中，广播包会分别在三个广播通道中被发送一次 (**37, 38, 39**)



BLE: 广播间隔

- ◆ 广播间隔，是两次广播事件之间的最小时间间隔
- ◆ 广播间隔的取值范围是在 **20ms ~ 10.24s** 之间
- ◆ 链路层会在每次广播事件期间产生一个**随机广播延时时间(0ms and 10ms)**，这个延时被加在广播间隔中，这样来避免多设备之间的数据碰撞。



BLE: 扫描事件

- ◆ 每次扫描设备打开**Radio** 接收器去监听广播设备，称为一个扫描事件
- ◆ 扫描事件交替发生在三个特定的广播通道中: **37, 38, 39**
- ◆ 扫描**频宽比 (Duty-Cycle)**, 关于扫描的两个时间参数:
 - 扫描间隔: 即扫描设备的扫描频度
 - 扫描窗口: 每次扫描事件持续的时间

BLE: 发起连接

- ◆除了扫描，设备也可以主动发起连接
- ◆发起状态的设备和扫描状态的设备区别在于：当它监听到一个可连接的广播，发起设备会发送一个**连接请求**，而扫描设备会发送一个**扫描请求**
- ◆连接请求包括一套**为从设备准备的连接参数**，安排连接事件发生的通道和时间
- ◆如果广播设备接收了连接，两个设备会进入连接状态，发起方会称为**Master**，而广播方会称为**Slave**

BLE: 连接参数

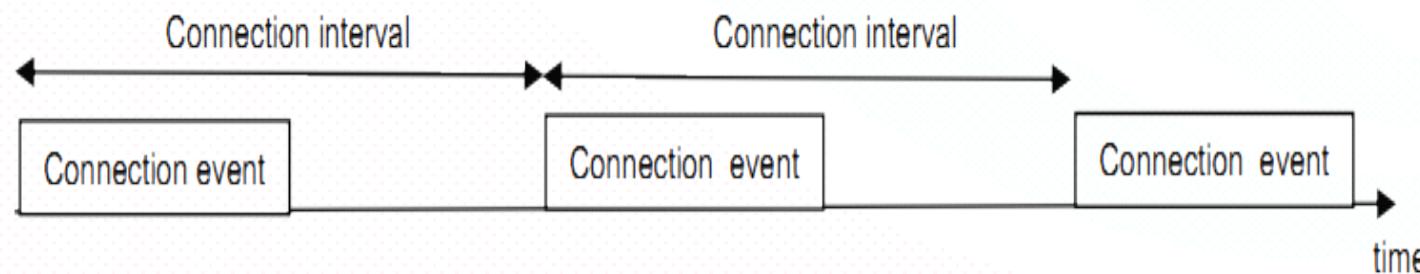
- ◆ 通道映射—指示连接使用的频道
- ◆ 跳频增量—一个**5-16**之间的随机，参与通道选择的算法
- ◆ 连接间隔—**1.25ms**的倍数，**7.5ms-4.0s**之间
- ◆ 监督超时—**10ms**的倍数，**100ms-32.0s**之间，必须大于

$$(1 + \text{slaveLatency}) * (\text{ConnInterval})$$

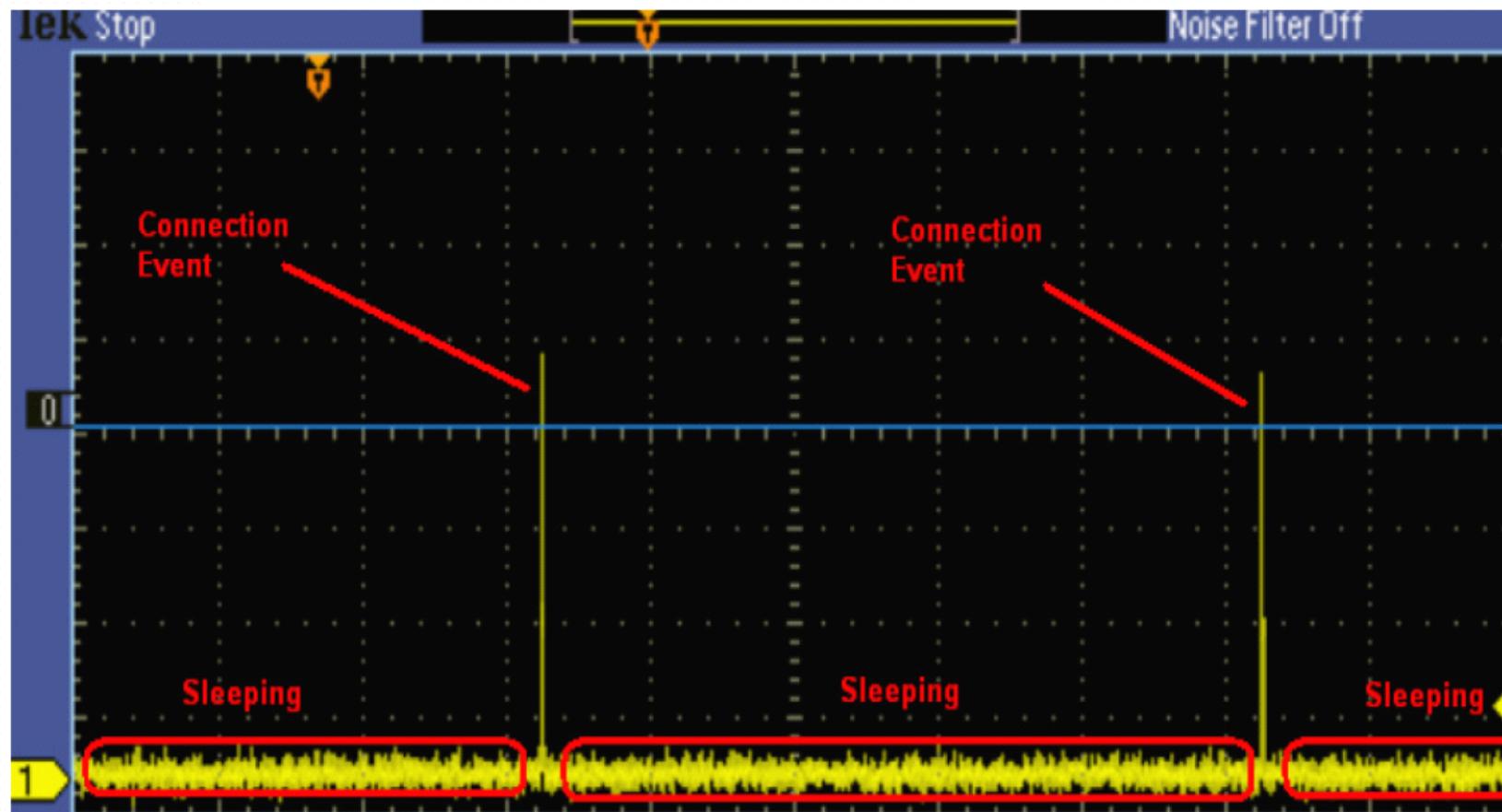
- ◆ 从机潜伏—**0-499**之间，不能超过
((supervisionTimeout / connInterval) – 1)

BLE: 连接事件

- ◆ 所有的通讯都发生在两个设备的**连接事件**期间
- ◆ 连接事件**周期地**发生，按照连接参数指定的间隔
- ◆ 每个事件发生在某个数据通道(**0-36**)，跳频增量参数决定了下次连接事件发生的通道
- ◆ 在每个连接事件期间，**Master** 先发送，**Slave**会在**150us**之后做出回应
- ◆ 即使一个**连接事件发生(或两者)**，双方都没有数据发送(例外情况是从设备潜伏使能)。这允许两个设备都承认对方仍然存在并保持活跃的连接。

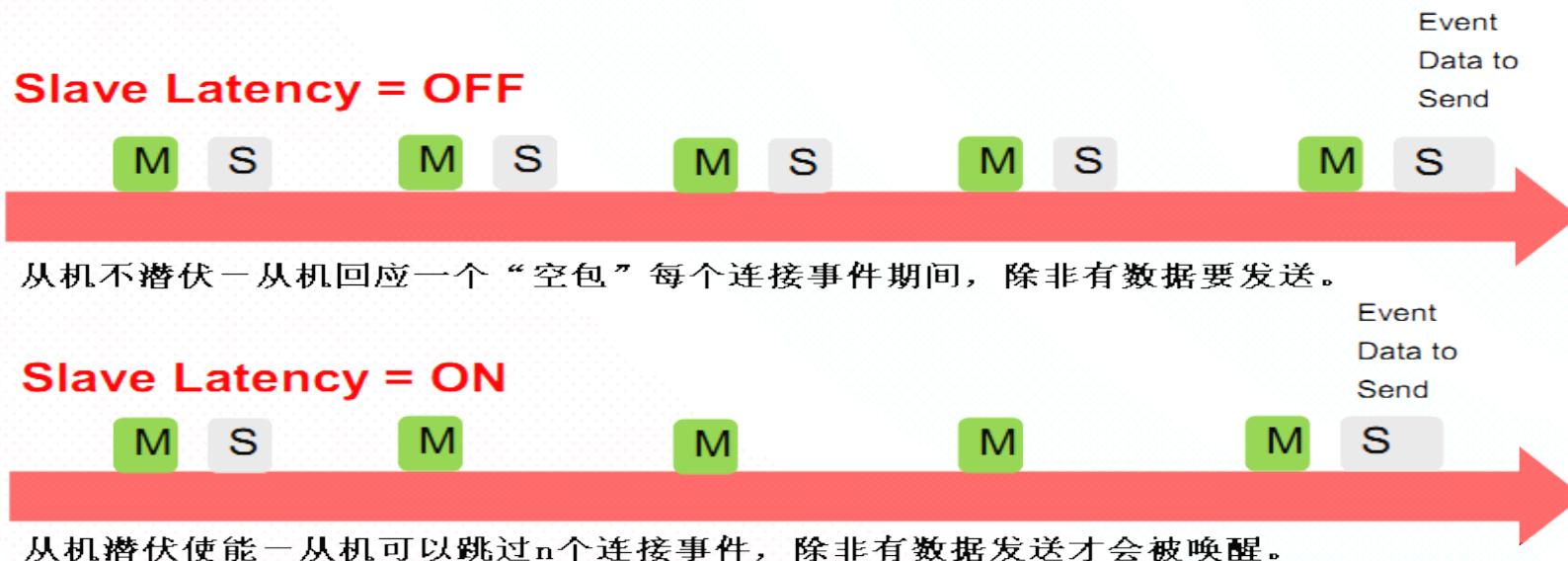


BLE:连接间隔:



BLE: Slave的潜伏

- ◆ 潜伏: Slave如果没有数据发送, 允许跳过连接事件
- ◆ 连接参数中的Slave 的潜伏值, 是允许从设备跳过的最大连接次数
- ◆ 在连接事件中, 如果slave 没有对master 的包做出回应, master 将会在后来的连接事件中重复发送, 直到slave 回应
- ◆ 两个有效的连接事件之间的最大时间跨度(假设slave 跳过了最大数目的连接事件), 称为“有效连接间隔”
- ◆ 从设备的潜伏值范围是0-499, 但是有效的连接间隔必须小于32.0s



BLE: 连接参数的设定

◆ 短间隔的连接事件:

- 两设备都会以高能耗运行
- 高数据吞吐量
- 发送等待时间短

◆ 长间隔的连接事件:

- 两设备都会以低能耗运行
- 低数据吞吐量
- 发送等待时间长

◆ 低或者0潜伏值:

- 从设备以高能耗运行
- 从设备可以快速收到来自中心设备的数据

◆ 高潜伏值:

- 外围设备在没有数据发送的情况下可以低能耗运行
- 外围设备无法及时收到来自中心设备的数据
- 中心设备能及时收到来自外围设备的数据

BLE: 终止连接

◆ Master 和 Slave 都可以主动断开连接

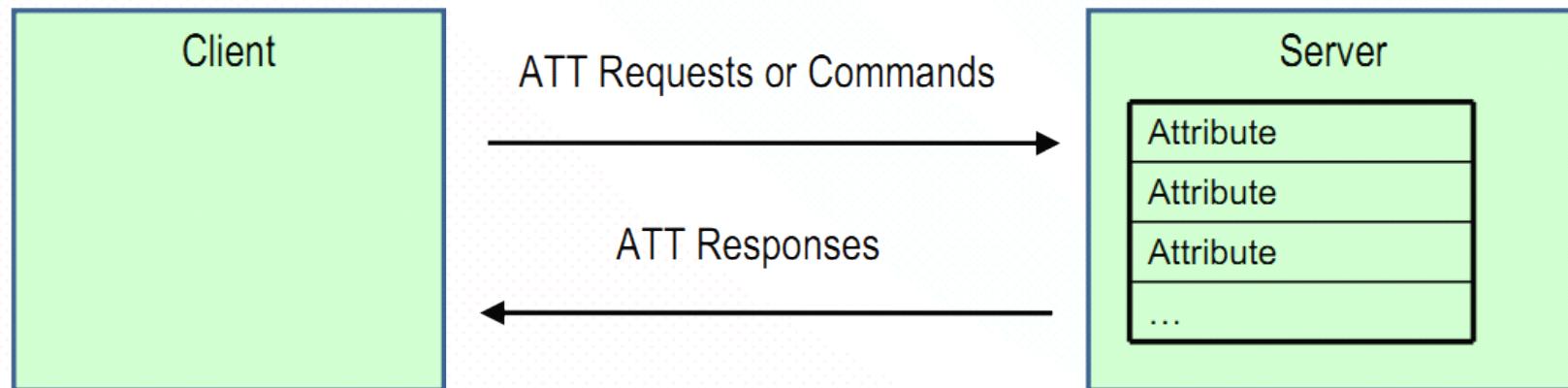
- 一边发起断开，另一边必须在在断开连接之前回应这个断开请求

◆ 监视超时而断开连接

- 监视超时参数指定了两个数据包之间的最大时间跨度
- 监视超时时间必须大于有效连接间隔而 小于32.0秒
- slave and master双方都维持着自己的监视超时计时器，在每次收到数据包时清零。
- 如果连接超时，设备会认为连接丢失，并且退出连接状态，返回广播，扫描或者待机模式。

BLE (ATT): Client / Server架构

- ◆ 服务设备提供数据，客户端使用这些数据
- ◆ 服务端通过操作属性的方式，提供数据访问服务
- ◆ 设备的服务/客户角色，不依赖于**GAP**层中心设备/外围设备角色，和**LL**层的**master/slave**角色定义
- ◆ 一个设备可能同时做为一个客户端和服务端，而两个设备上的属性不会相互影响。



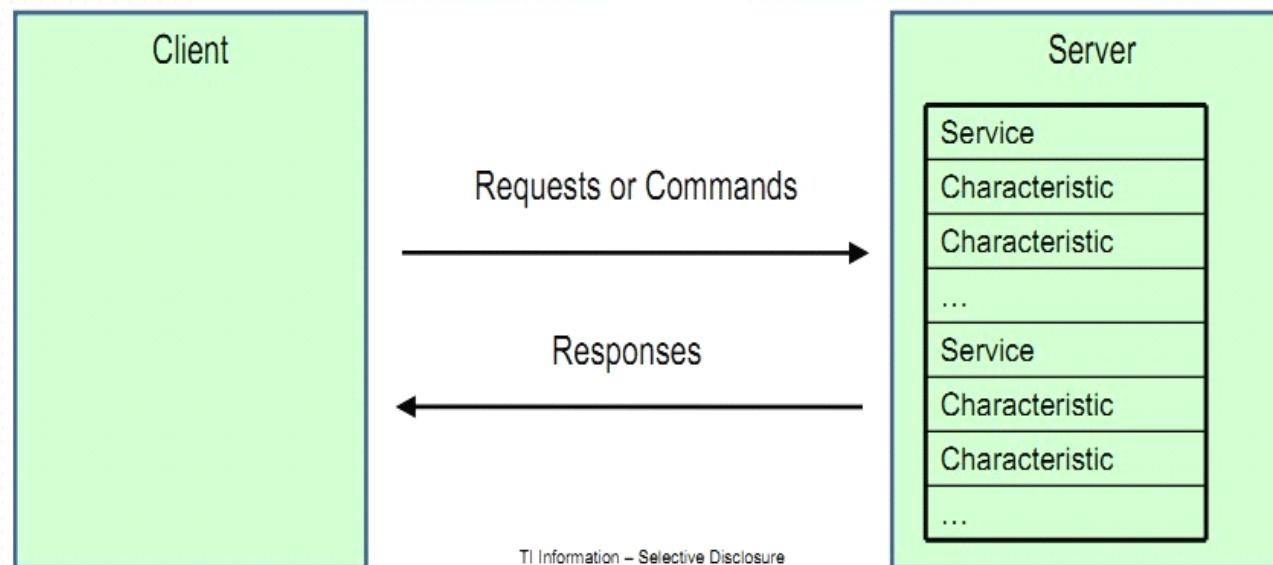
BLE (ATT): Attribute Table Example

- ◆ **Handle** – 属性在列表中的地址
- ◆ **Type** – 说明代表什么数据，可以是Bluetooth SIG 分配或者是客户自己定义的**UUID (universal unique identifier, 通用性, 唯一性)**
- ◆ **Permissions** – 权限，定义了client是否可以访问属性的值，以及特定的访问方式

Handle	Type	Permissions	Value
39	0x2800 (GATT Service UUID)	Read	E0:FF (2 bytes)
40	0x2803 (GATT Characteristic UUID)	Read	10:29:00:E1:FF (5 bytes)
41	0xFFE1 (Simple Keys state)	(none)	00 (1 byte)
42	0x2902 (GATT Client Characteristic Configuration UUID)	Read and Write	00:00 (2 bytes)

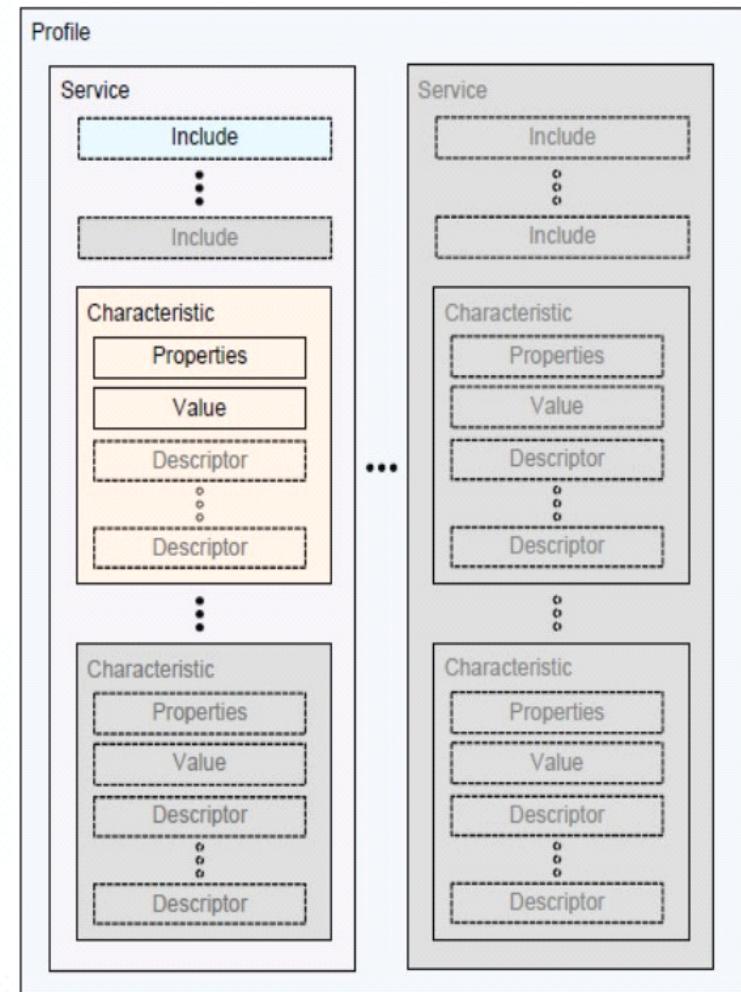
BLE: (GATT) Client / Server 架构

- ◆ GATT 指定了 **profile** 数据交换所在的结构
- ◆ 除了数据的封装方式不同， **client server** 和 **Attribute** 协议结构相同， 数据封装在 “**Services**” 里， 用 “**Characteristic**” 表示。



BLE (GATT):Profile 层次结构

- ◆ 为了实现用户的应用，profile 通常是由一个或者多个“services”组成
- ◆ 一个service 或许包含某个特征值“characteristic values”
(例如：在一个温度采集设备中，通常会包含一个温度的特征值)
- ◆ 每一个特征值必须有占用一个特征申明结构，其中包括它的其他特性，它是服务端和客户端共享的读写空间
- ◆ 这个特征值可以包含一个可选的描述(descriptor字串)，来指示这个特征值的含义



BLE (GATT): Service Example

- ◆ service 起始于 handle 39，用 0x2800 来指示这个起始位置，这个 0x2800 是 Bluetooth SIG 的相关数据手册定义的，做为 GATT Service 的 UUID。
- ◆ handle 39 所指向的属性值为 0xFFE0，这个是用户自定义 profile 中的按键服务 UUID (这里仅仅是一个例子， 0xFFE0 可能已经被 Bluetooth SIG 使用)
- ◆ 这个按键服务包含了所有后面的属性，直到下一个服务 UUID 定义处，或者到服务列表的结尾处。在这个例子中，按键服务的最后的属性所在地址为 handle 42，因此新的服务开始位置将会是 handle 43。

Handle	Type	Permissions	Value
39	0x2800 (GATT Primary Service UUID)	Read	E0:FF (2 bytes) (0xFFE0 = Simple Keys Service custom UUID)
40	0x2803 (GATT Characteristic Declaration UUID)	Read	10:29:00:E1:FF (5 bytes) (0xFFE1 = Simple Keys Value custom UUID) (0x0029 = handle 41) (0x10 = characteristic properties: notify only)
41	0xFFE1 (Simple Keys state)	(none)	00 (1 byte) (value indicates state of keys)
42	0x2902 (GATT Client Characteristic Configuration UUID)	Read and Write	00:00 (2 bytes) (value indicates whether notifications or indications are enabled)
43	0x2800 (GATT Primary Service UUID)	Read	A1:DD (2 bytes) (0xDDA1 = Other Service custom UUID)

BLE (GATT): Characteristic Declaration

- ◆ Handle 40 是一个特征值的声明，用0x2803来指示，这个0x2803 同样也是Bluetooth SIG 的相关数据手册定义的，做为 GATT Characteristic Declaration的UUID。
- ◆ 特征值的属性值包含5个字节的长度10:29:00:E1:FF：
 - (0xFFE1) – 表明特征值的属性类型 (0xffe1:客户自定义特征值的UUID)
 - (0x0029) – 是这个值所保存的位置handle (0x0029 = 41)
 - (0x10) – 表明对这个特征值的操作权限 (0x10 : notify only)

Handle	Type	Permissions	Value
39	0x2800 (GATT Primary Service UUID)	Read	E0:FF (2 bytes) (0xFFE0 = Simple Keys Service custom UUID)
40	0x2803 (GATT Characteristic Declaration UUID)	Read	10:29:00:E1:FF (5 bytes) (0xFFE1 = Simple Keys Value custom UUID) (0x0029 = handle 41) (0x10 = characteristic properties: notify only)
41	0xFFE1 (Simple Keys state)	(none)	00 (1 byte) (value indicates state of keys)
42	0x2902 (GATT Client Characteristic Configuration UUID)	Read and Write	00:00 (2 bytes) (value indicates whether notifications or indications are enabled)
43	0x2800 (GATT Primary Service UUID)	Read	A1:DD (2 bytes) (0xDDA1 = Other Service custom UUID)

BLE (GATT):Characteristic Configuration

- ◆ 另外做为特征值声明，可以有一个可选的描述信息。
- ◆ 这个例子中， handle 42 包含了特征值的配置信息， **0x2902** , 这个值同样也是 Bluetooth SIG 的相关数据手册定义的，做为 GATT Client Characteristic Configuration 的 UUID 。
- ◆ 这个配置值有读写权限，意味着 GATT 客户端可以改变这个值
- ◆ 如果把这个值(通知开关使能)从 **0x0000 (notifications off)** 改为 **0x0001 (notifications on)** , GATT 服务端将开始发送这个特征值的通知到 GATT 客户端。

Handle	Type	Permissions	Value
39	0x2800 (GATT Primary Service UUID)	Read	E0:FF (2 bytes) (0xFFE0 = Simple Keys Service custom UUID)
40	0x2803 (GATT Characteristic Declaration UUID)	Read	10:29:00:E1:FF (5 bytes) (0xFFE1 = Simple Keys Value custom UUID) (0x0029 = handle 41) (0x10 = characteristic properties: notify only)
41	0xFFE1 (Simple Keys state)	(none)	00 (1 byte) (value indicates state of keys)
42	0x2902 (GATT Client Characteristic Configuration UUID)	Read and Write	00:00 (2 bytes) (value indicates whether notifications or indications are enabled)
43	0x2800 (GATT Primary Service UUID)	Read	A1:DD (2 bytes) (0xDDA1 = Other Service custom UUID)

BLE (GATT):Client Commands

- ◆ 当两个BLE设备处于连接状态，客户端和服务端设备的通讯方式：
 - **Discover Characteristic by UUID** – 搜索服务端设备所能提供的所有匹配UUID规范的属性
 - **Read Characteristic Value** – 使用指定的handle 读特征值
 - **Write Characteristic Value** – 使用指定的handle 写特征值
- ◆ 除此之外，如果通知被使能，服务设备会自动向客户端设备发出下列信息：
 - **Notification** – 某个特征值被发送到客户端设备，而没有被读请求，并且不需要应答。
 - **Indication** – 某个特征值被发送到客户端，没有被读请求的情况下，但是在其他数据被发送之前必须被确认。

CC2540DK-MINI Kit

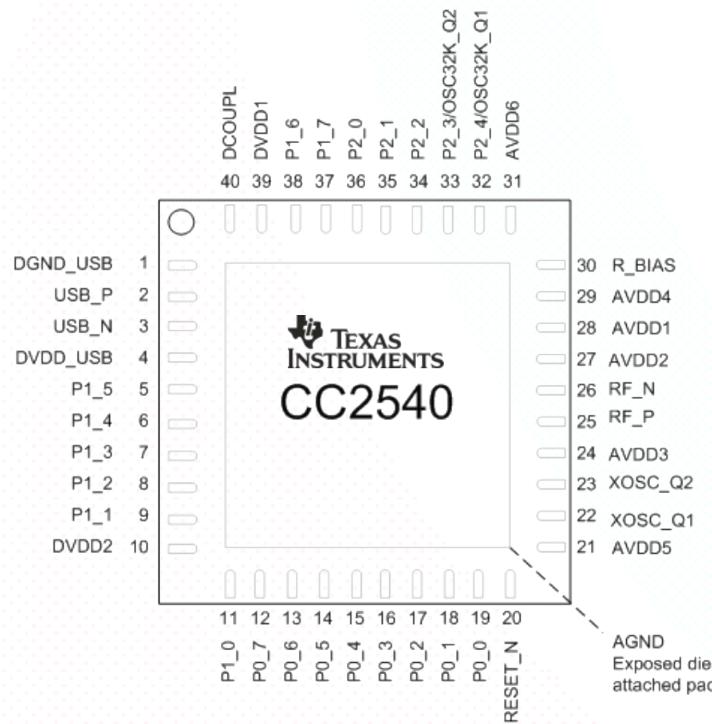
介绍



深圳信驰达科技

88-31

BLE – TI CC2540/1 解决方案(SoC)



- ◆ 8051 MCU - 128/256 kB in-system programmable **Flash** 8 kB SRAM
- ◆ Programmable Radio Supports
 - Bluetooth Low Energy (1Mbps GFSK)
- ◆ Digital peripherals
 - 21 GPIOs
 - 2 USART (UART or SPI)
 - Full Speed USB 2.0
 - 2x 16 bit, 2x 8-bit timers
 - Dedicated Link Layer timer for **Bluetooth LE protocol timing**
 - AES-128 encryption/decryption in HW
- ◆ Advanced analog peripherals
 - 8-channel 8-12 bit delta-sigma ADC
 - Ultra-low-power **analog comparator**
 - Integrated high-performance op-amp
- ◆ All in a 40-pin 6x6x0.85mm QFN package
- ◆ Pin compatible with CC2530/33 and CC2541

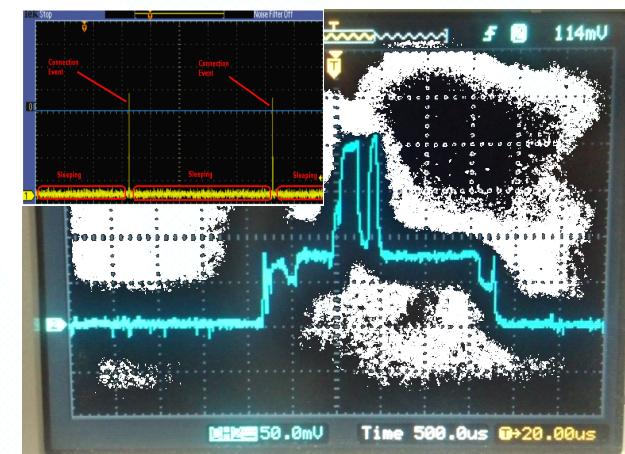
CC2540/2541功耗指标

◆ CC2540功率

- 工作模式 RX 低至: 19.6mA
- 工作模式 TX (-6 dBm): 24mA
- 功率模式 1 (3 μs唤醒) : 235 μA
- 功率模式 2 (睡眠定时器打开) : 0.9 μA
- 功率模式 3 (外部中断) : 0.4 μA
- 宽电源电压范围 (2V - 3.6V)
- 使用 TPS62730 旁通模式2MHz降压转换器
 - RX Down to 15.8 mA (3 V Supply)
 - TX (-6 dBm): 18.6 mA (3 V Supply)

◆ CC2541功率

- 工作模式 RX 低至: 17.9mA
- 工作模式 TX (0 dBm): 18.2mA
- 功率模式 1 (4 μs唤醒) : 270 μA
- 功率模式 2 (睡眠定时器打开) : 1 μA
- 功率模式 3 (外部中断) : 0.5 μA
- 宽电源电压范围 (2V - 3.6V)
- 使用 TPS62730 降压转换器
 - RX 低至: 14.7mA (3V电源)
 - TX (0 dBm): 14.3 mA (3V电源)



Case 3		
Time (us)	Current (mA)	Percent of events
400	6	100
500	8.3	
80	12.9	
200	21	
120	9.6	
112	27.5	
1240	7.7	
176	4.3	
0	0	0

Battery capacity (mAh):	200
Connection Interval (ms):	1000
Sleep Current with timer running (mA):	0.001
Average current draw during connection (mA):	0.027315972
Expected battery life (hours):	7321.723715
Expected battery life (days):	305.0718215
	0.027316

TI CC2540DK-MINI Hardware Kit



Debugger

- Works with keyfob and USB dongle
- Supports IAR and TI flash programmer



CC2540 Keyfob

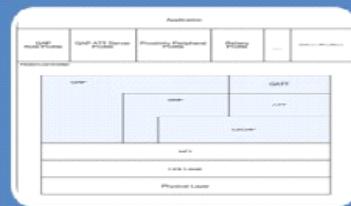
- Powered by CR2032 coin cell battery
- LED, buttons, buzzer, accelerometer
- Usually acts as peripheral, application is on chip.



USB Dongle

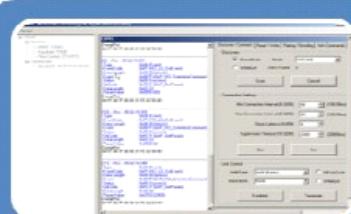
- Use Btool.exe to or custom app to send HCI commands.
- Usually acts as master (cell phone)

TI CC2540DK-MINI Software Kit



Stack Libraries

- Royalty free
- Full qualification
- Example Projects



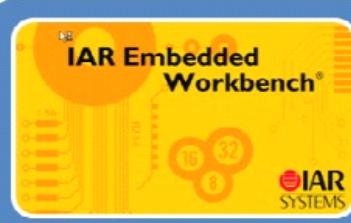
Btool Application

- Drives USB dongle with HCI commands
- Scan for devices, connect, authentication
- Log messages



SmartRF Flash Programmer

- Can flash CC2540
- Change address on device



IAR Compiler and IDE

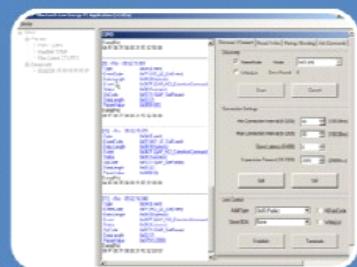
- Robust 8051 compiler with CC2540 support.
- 30 day free evaluation

TI CC2540DK-MINI Support



TI RF Sniffer

- Free
 - Works with Mini Kit USB Dongle



Example Applications

- SimplePeripheral – keypress, strings
 - KeyFobDemo – Accelerometer, buzzer, beeper, proximity, battery level.
 - Other SIG profile applications under development



Power Calc Applications Note

- Excel sheet to help calculate battery life expectancy

CC2540DK-MINI Kit – KeyFob



- ◆ CR2032 钮扣电池供电
- ◆ 典型的Peripheral 角色
运行参考源码
- ◆ 原理图PCB参考设计
- ◆ 按键,蜂鸣器,LED,加速度, AD电量采集示范
- ◆ Profile /Service添加的
源码参考

Application

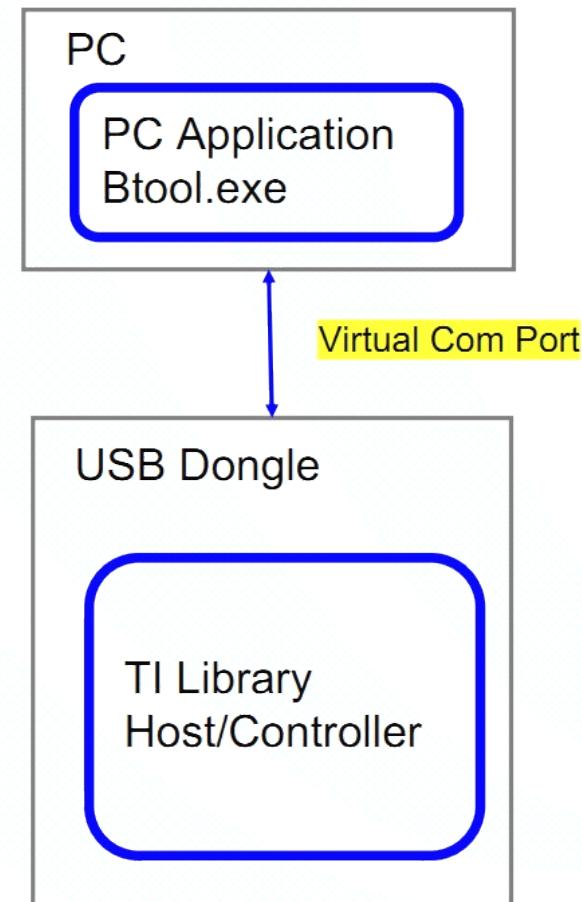
Profiles

TI Library
Host/Controller

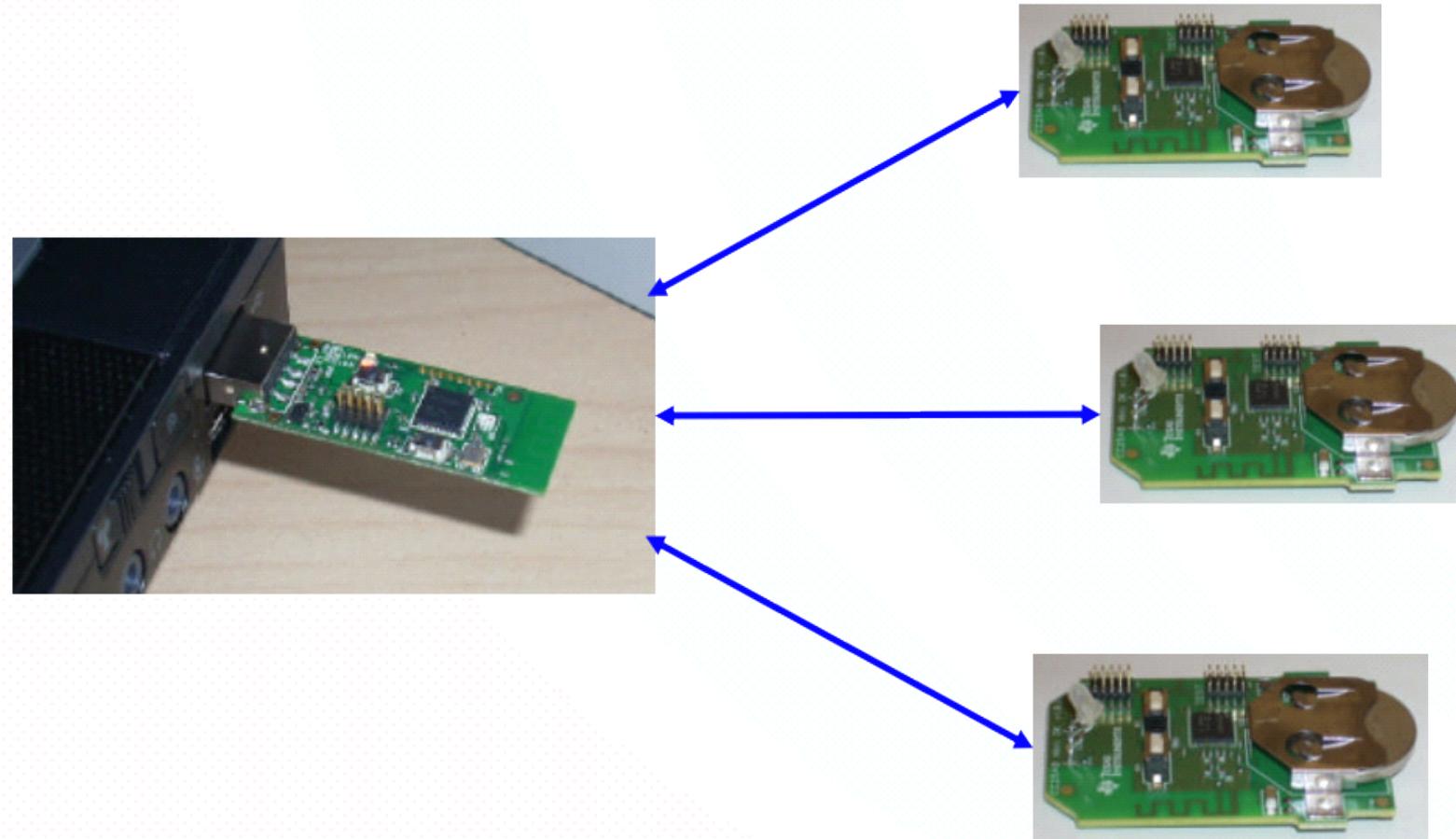
CC2540DK-MINI Kit – USB Dongle



- ◆ 典型**GAP Central** 角色设备源码参考
- ◆ **Network processor**
(Btool或串口终端操作)
- ◆ 标准**HID**设备免驱适配器，
虚拟键盘演示



CC2540DK-MINI Kit – Network Example



BlueTooth 4.0 (BLE)

典型应用



深圳信驰达科技

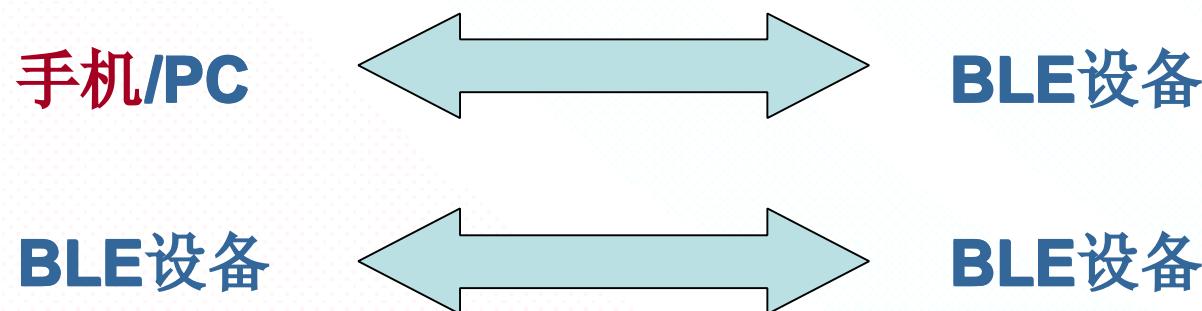
88-40

CC254x方案的优势和应用结构模式

◆ CC2540方案的优势

- 超低的峰值和待机能耗(最低0.4uA)
- 使用标准纽扣电池(CR2032)
- 低成本(SOC)
- 不同制造商的设备也能保持兼容(BlueTooth4.0)
- 射程有所加强(200英尺)

◆ 应用的结构模式



心率带应用



更丰富的医疗数据采集



BLE作为手机和各类电子产品的桥梁，让建立更丰富的用户数据库成为可能。

寻物和防丢（互相寻找，距离报警）



电动玩具（被遥控）



BLE技术提供了遥控器实现的新方案，用软件代替硬件，尽可能减少电池的使用，更多地避免了重复性开发。是**BLE**低碳环保的另外一个侧面。

键盘应用



CC2540 是BLE应用技术典型的SOC方案，丰富的资源让产品的成本降到最低。

娱乐游戏应用



通过**BLE**技术，和**PC**或者
智能电视互动。

遥控器应用



CC2540既可以工作在从设备状态，也可以作为主设备工作，完成和手机兼容的遥控功能。

智能家居监控



BLE接口技术让智能家居更贴心更方便，集成**BLE**的家居系统，在手机面板上即可完成智能操控或状态读取。

运动休闲

BLE作为一种通讯手段，
借助丰富的手机应用，
让健身运动更有乐趣，
也更专业。



设备外设（触摸笔）



USB技术让**PC**外设的创意无限；**BLE**技术同样让移动设备的外设提供了无尽的发挥空间。

运动手表



BLE技术会让
手表越来越
不像手表。

电子支付(NFC应用)



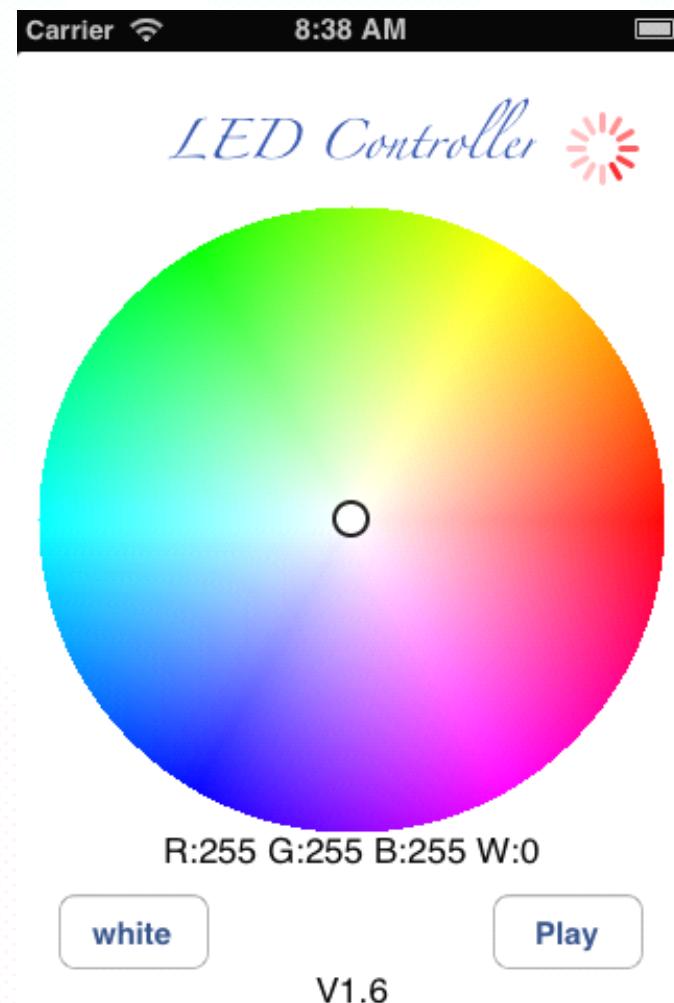
信驰达 iOS APP 介绍



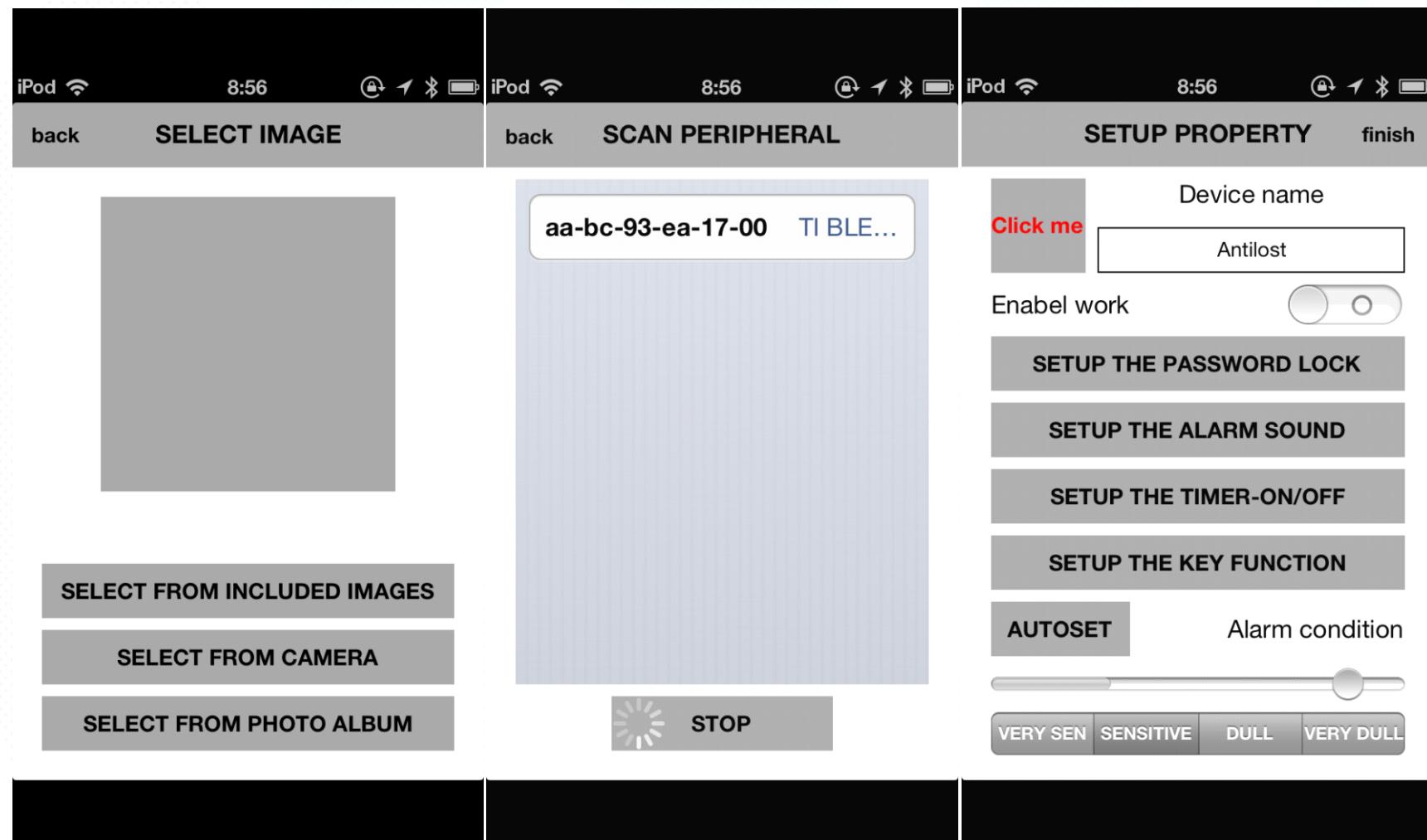
深圳信驰达科技

62-54

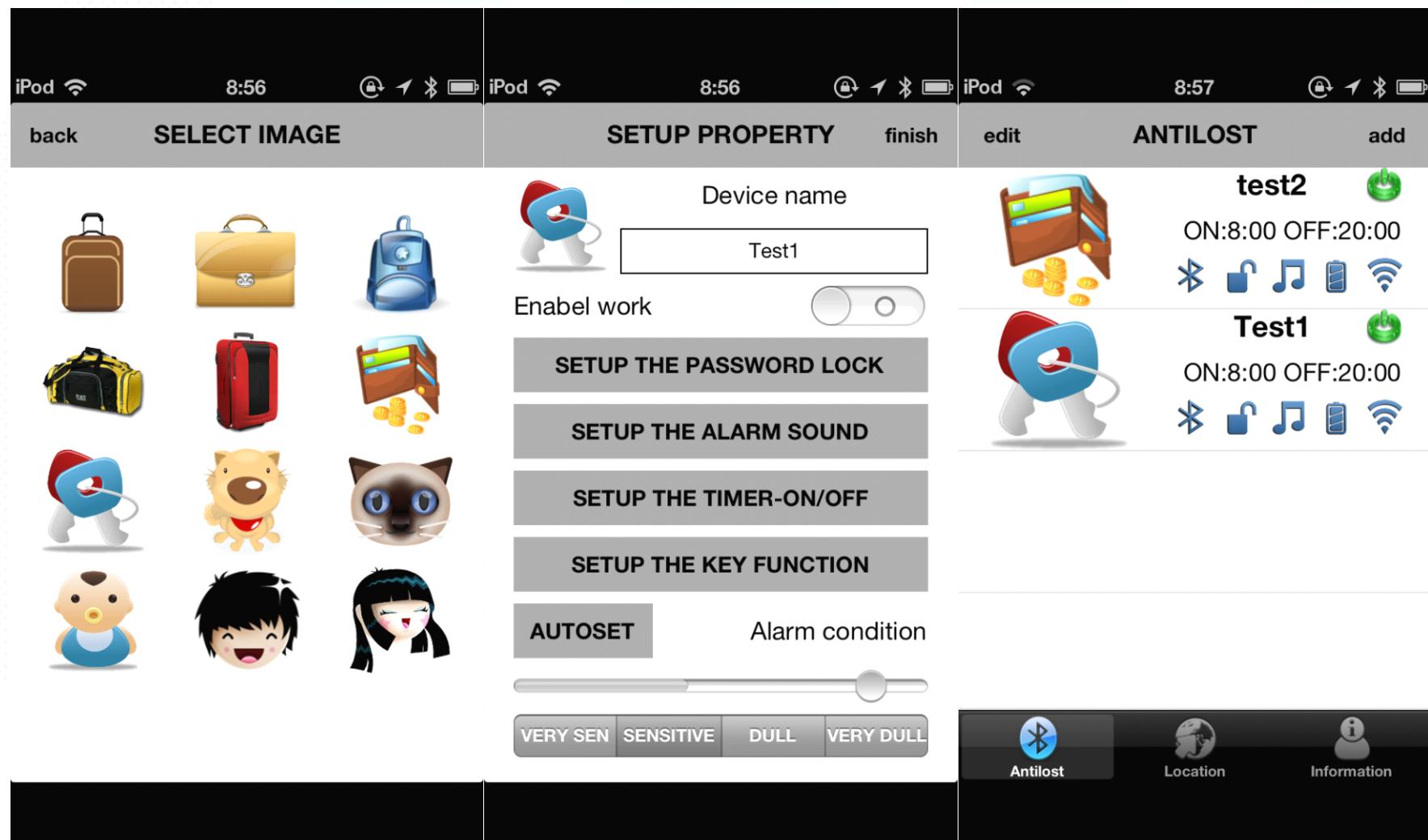
灯光控制APP(BLE LED Controller)



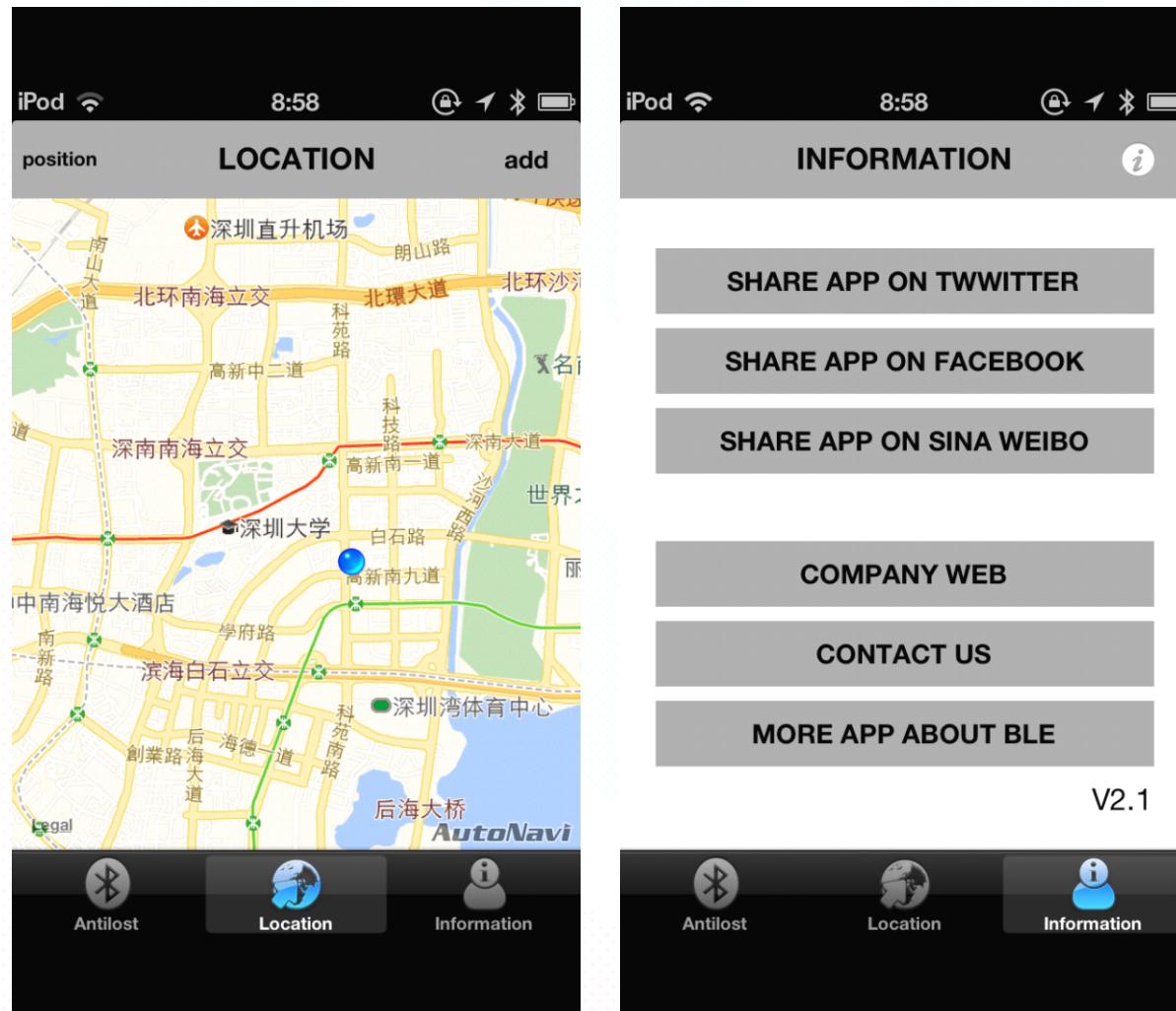
防丢器APP(BLE Anti-lost)-1



防丢器APP(BLE Anti-lost)-2



防丢器APP(BLE Anti-lost)-3



模块测试工具(BLE Transmit_Moudel)V1.2

The image displays three screenshots of a mobile application interface for testing a BLE module, specifically version V1.2. The interface is designed for an iPod/iPhone, as indicated by the status bar at the top of each screen.

Screenshot 1 (Left): Main Screen

- Device Name: TA1200-EA93B932
- UUID: 5FD44C5F-E6F2-456D-6C5E-2A3ECDDDB4954
- Status: STATIC:Init

Screenshot 2 (Middle): Main Screen

- Device Name: TA1200-EA93B932
- UUID: 5FD44C5F-E6F2-456D-6C5E-2A3ECDDDB4954
- Status: STATIC:Connected finish

Screenshot 3 (Right): Detailed Log Screen

- Device Name: TA1200-EA93B932
- TXc: 8 (green button)
- RXc: 8 (green button)
- Data Log:
 - IP: ABCDEFGHIJKLMNOP000005
PC: ABCDEFGHIJKLMNOP000005
 - IP: ABCDEFGHIJKLMNOP000006
PC: ABCDEFGHIJKLMNOP000006
 - IP: ABCDEFGHIJKLMNOP000007
PC: ABCDEFGHIJKLMNOP000007
 - IP: ABCDEFGHIJKLMNOP000008
PC: ABCDEFGHIJKLMNOP000008
- Receive: ABCDEFGHIJKLMNOP000008
- Buttons:
 - Up to 20 ASCII
 - Clear all
 - Send
 - Auto Mode

Contact us



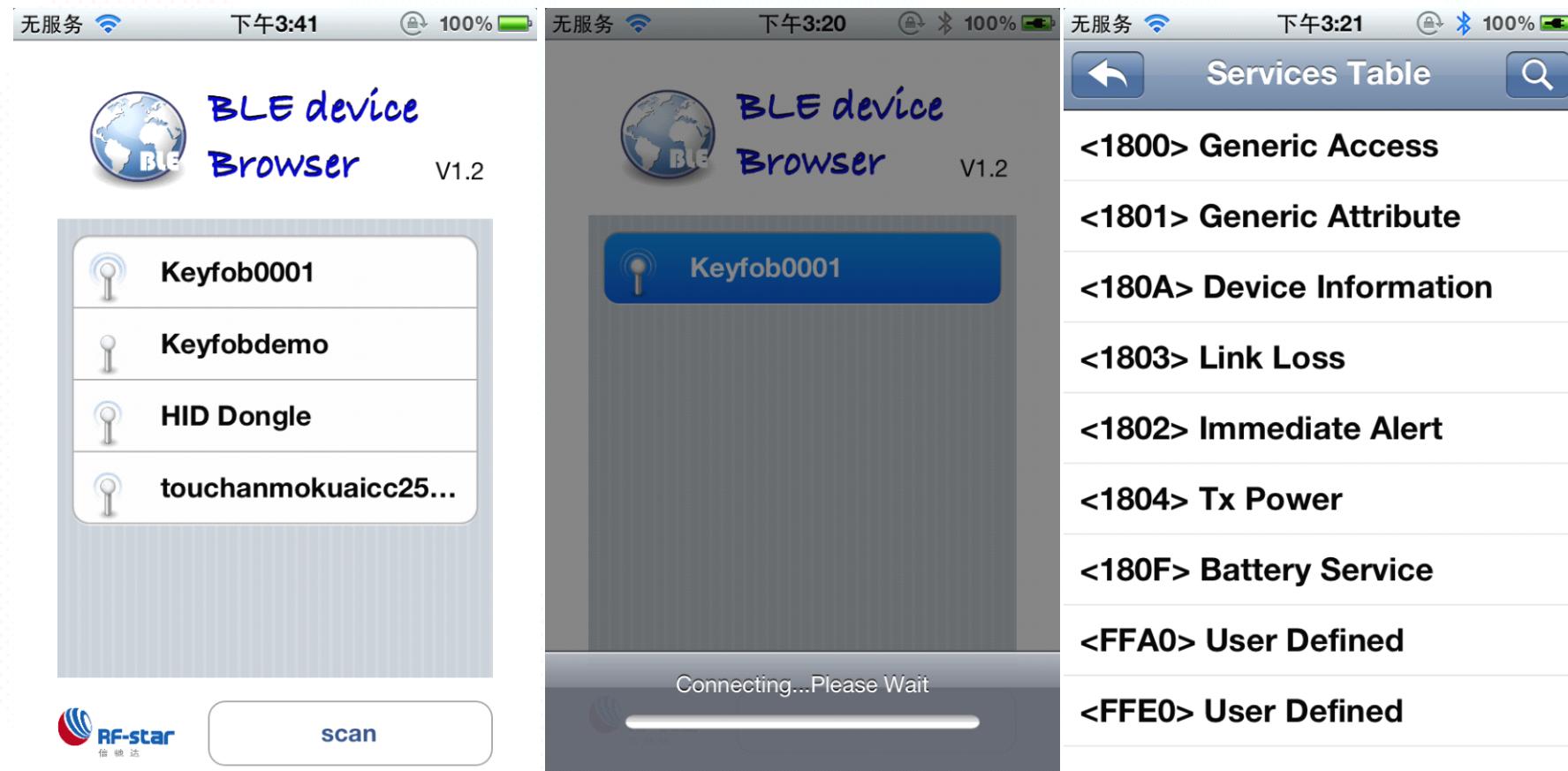
Contact us



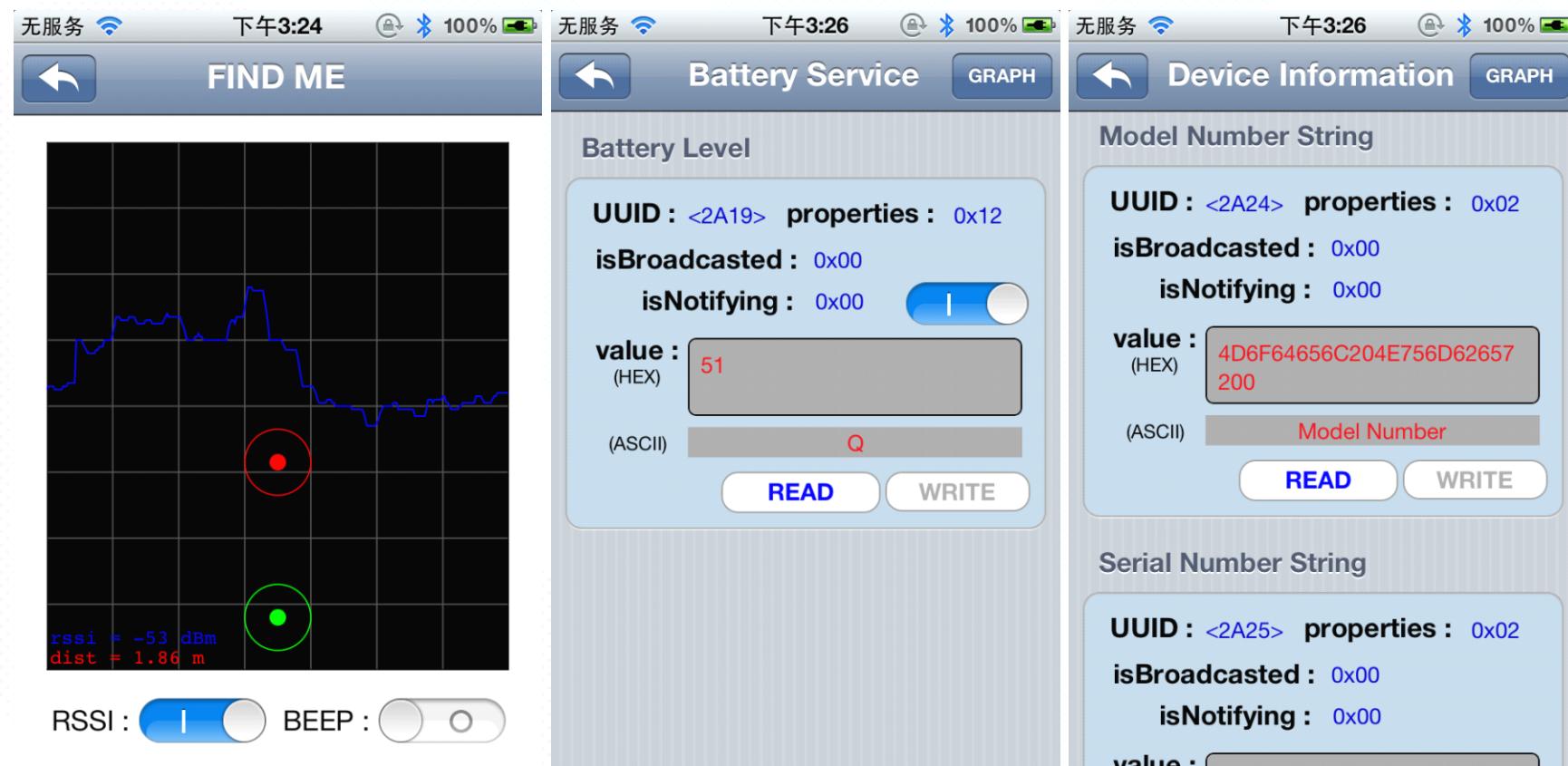
深圳信驰达科技

88-59

BLE浏览器APP(BLE Browser)-1



BLE浏览器APP(BLE Browser)-2



BLE浏览器APP(BLE Browser)-3



NOCNE = 3

CLEAR



NOCNE = 1

CLEAR

BLE通讯演示与实践



深圳信驰达科技

62-63

CC Debugger

调试器使用



CC Debugger调试器使用

- ◆ 驱动安装

C:\Program Files\IAR Systems\Embedded Workbench 6.0\8051\drivers\Texas Instruments

C:\Program Files\Texas Instruments\SmartRF Tools\Drivers\ceball

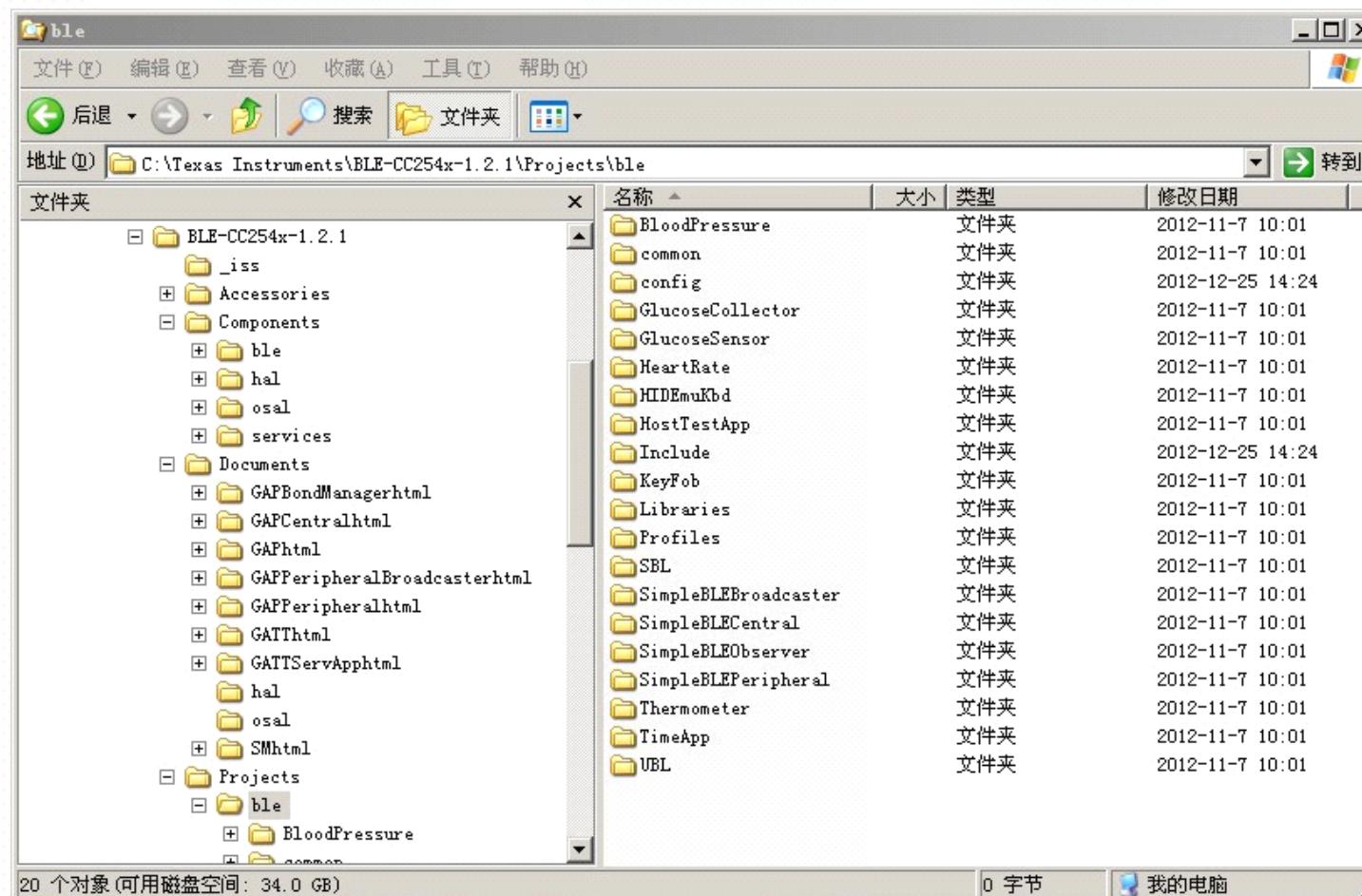
- ◆ 需要外部供电，红色线为第一脚地。

开发工具及驱动安装

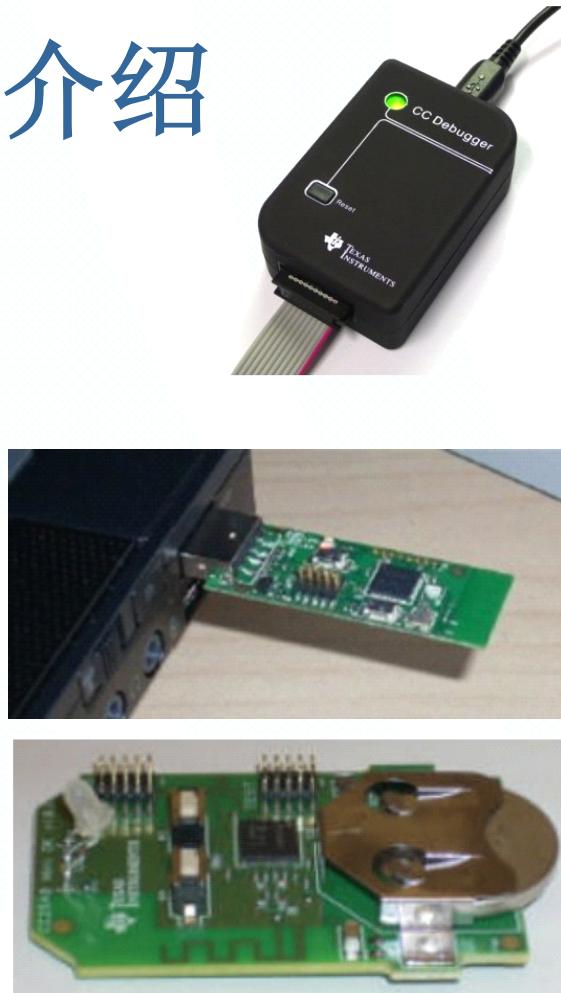
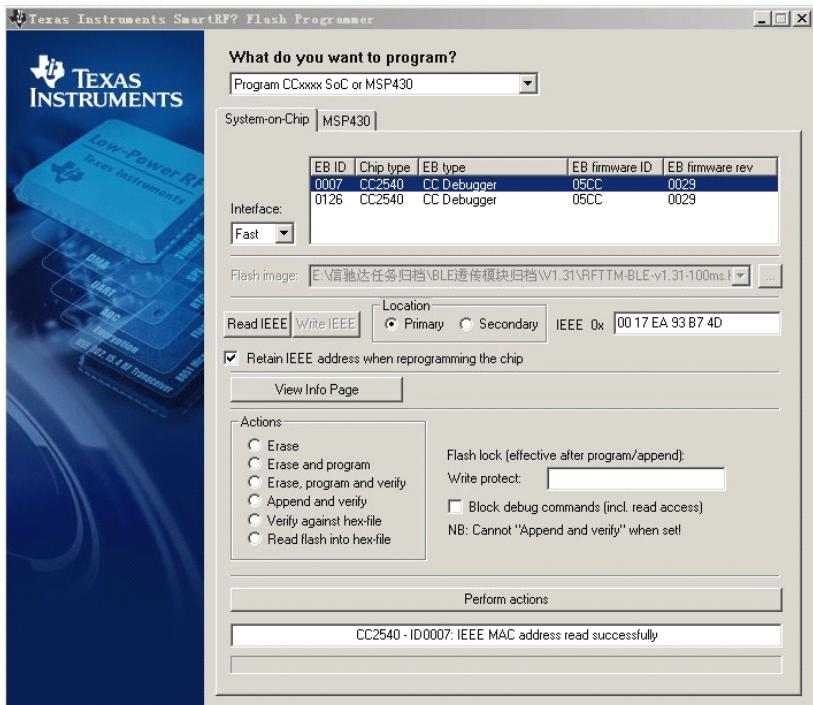


安装协议栈,Demo源码以及Btool

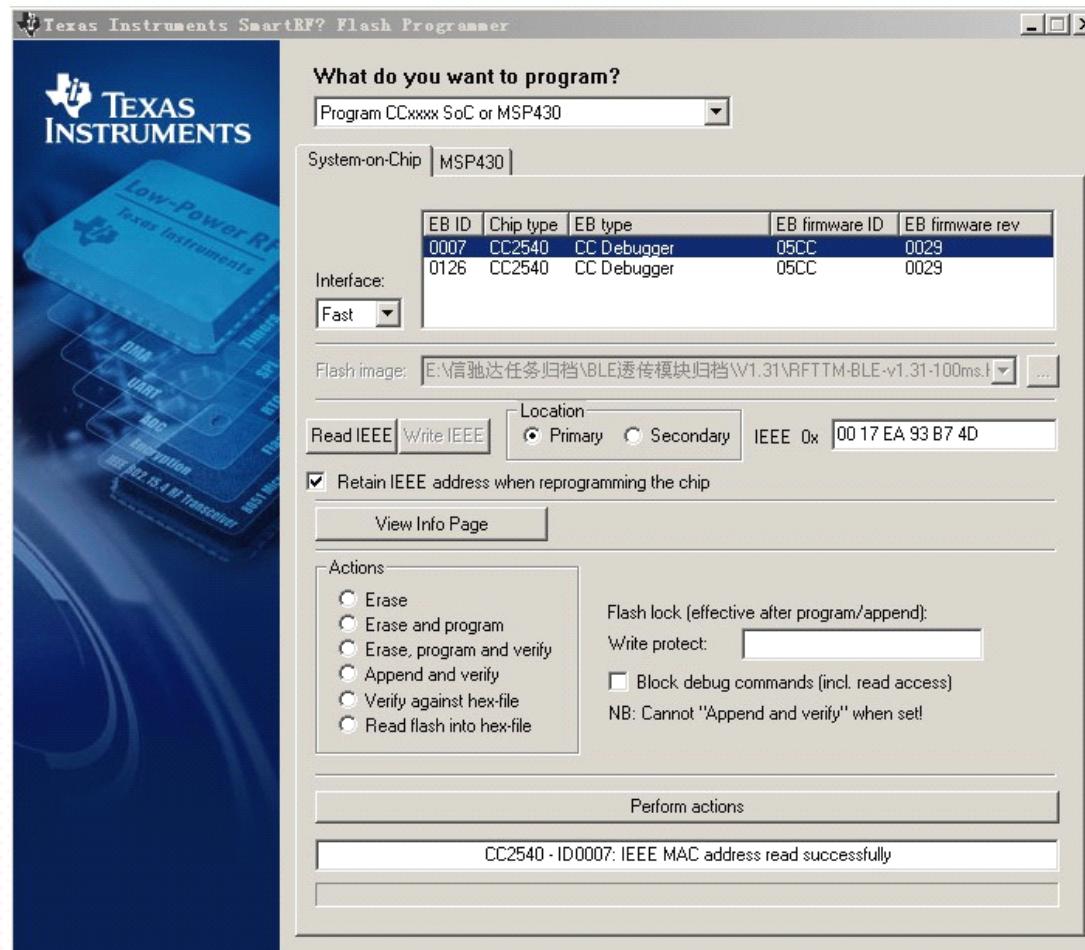
1，解压安装DK源码及工具包BLE-CC254x-1.2.1.exe到默认目录。



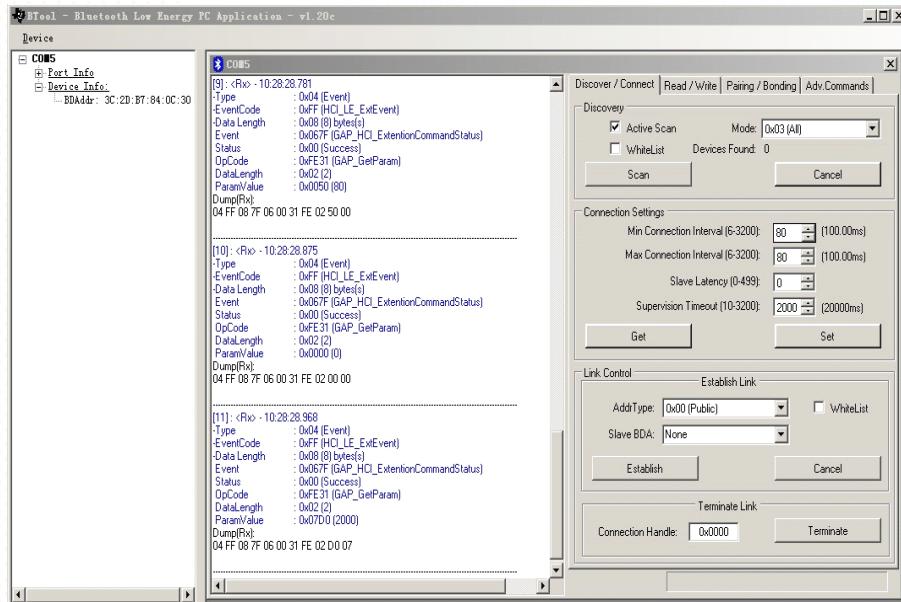
SmartRF Studio 介绍



SmartRF Studio (flash programmer)介绍

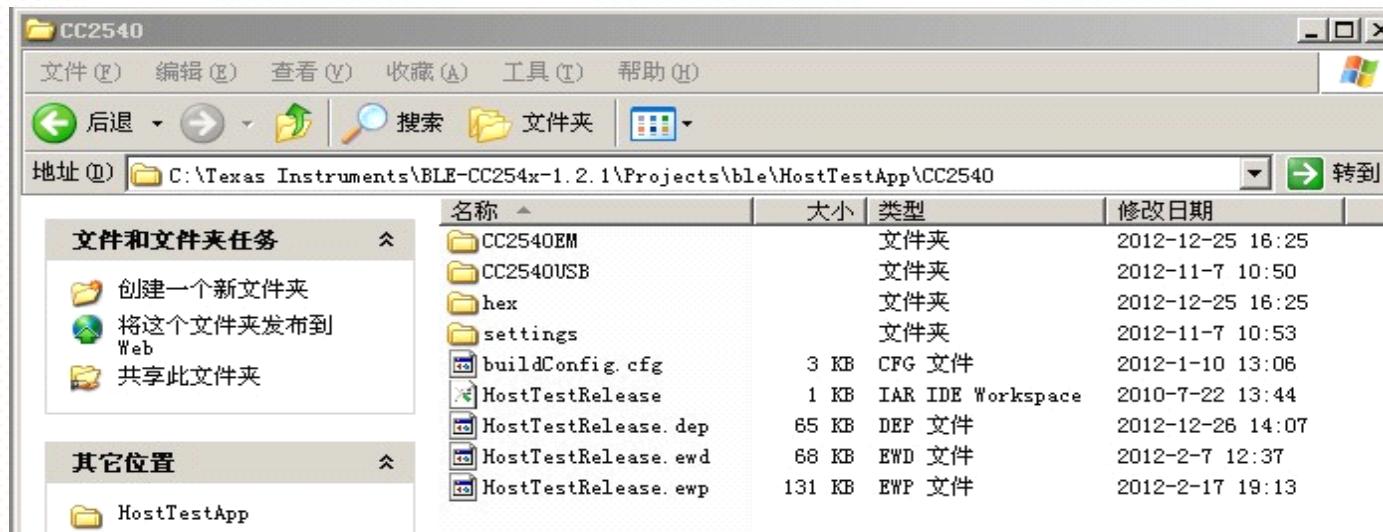


BTOOL的使用 (keyfob应用为例)



Dongle固件下载以及驱动安装

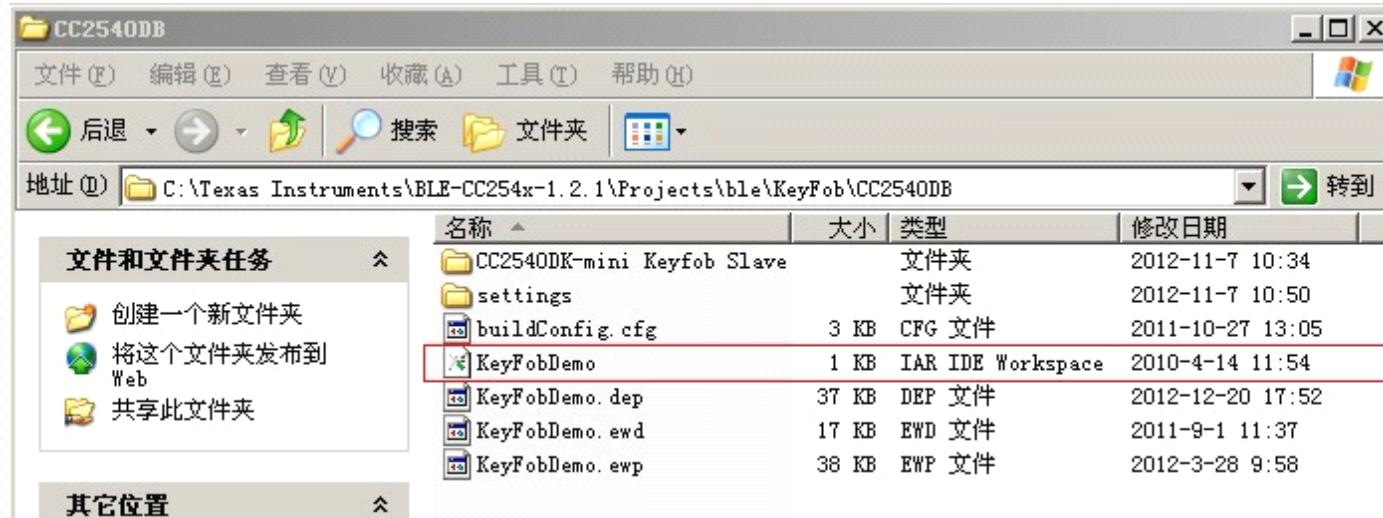
安装编译环境IAR-8051- 8.10



Dongle的USB驱动位置：



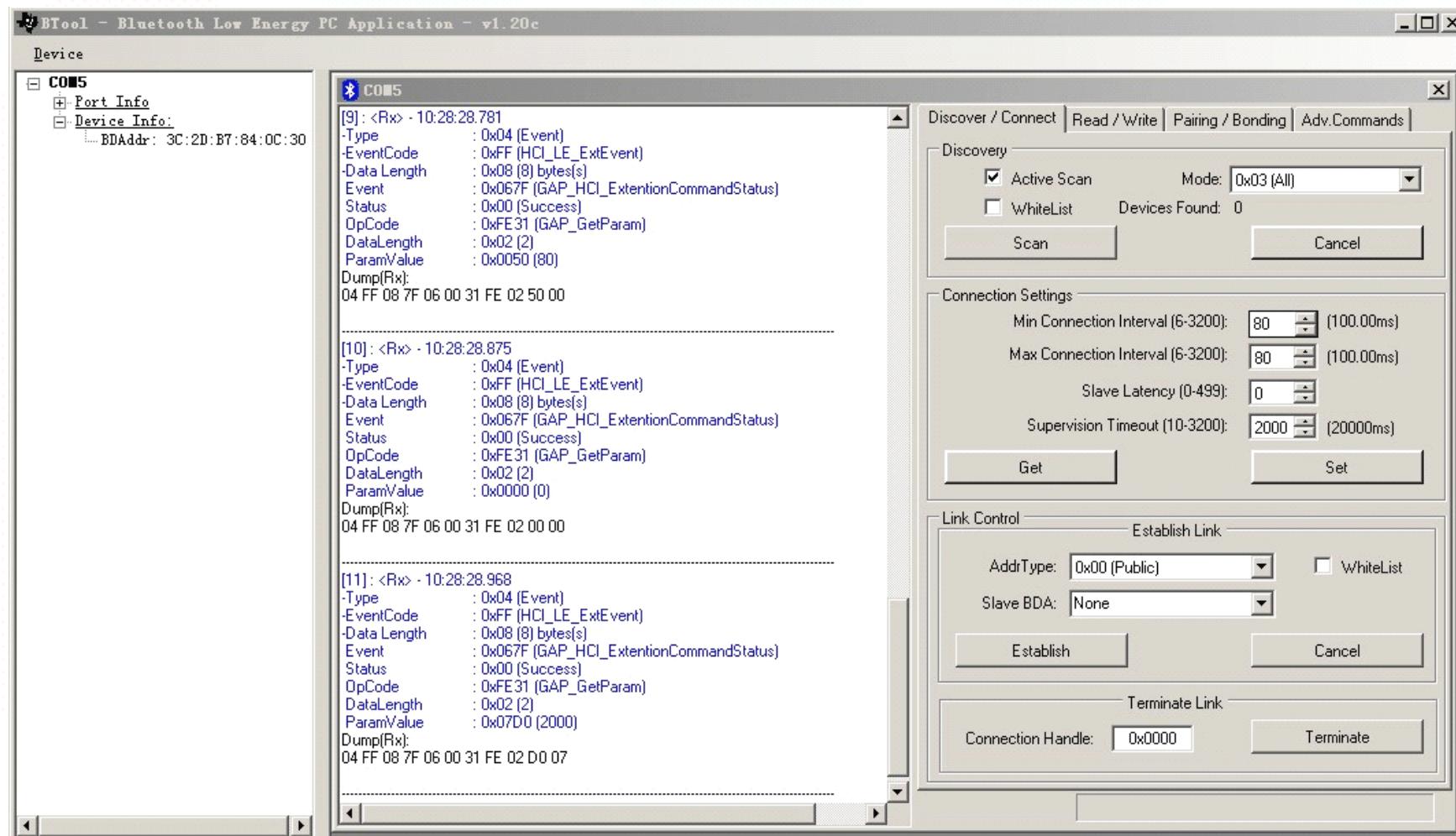
keyfob固件下载



keyfobDemo应用展示了以下功能：

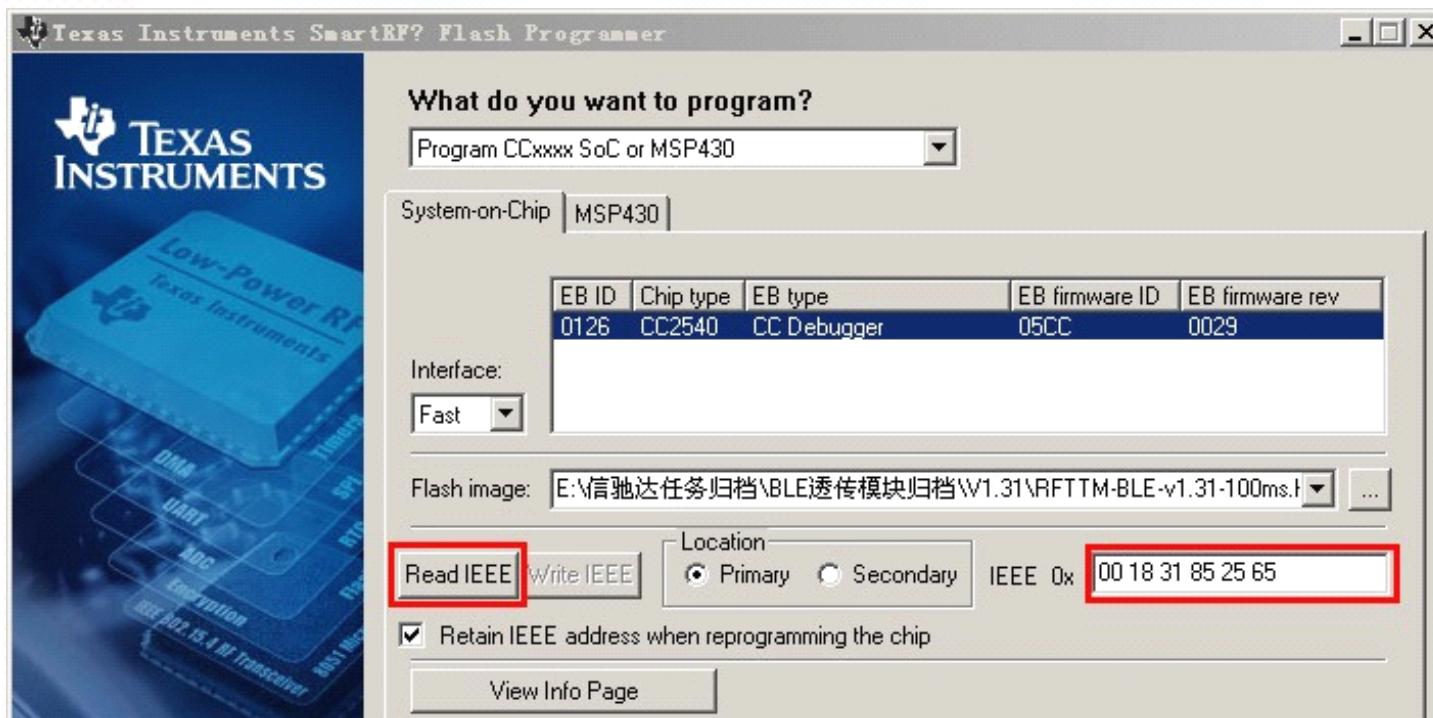
1. 电池电量提醒
2. 报告三轴加速度采集值
3. 距离改变提醒
4. 按键状态改变通知

BTOOL介绍



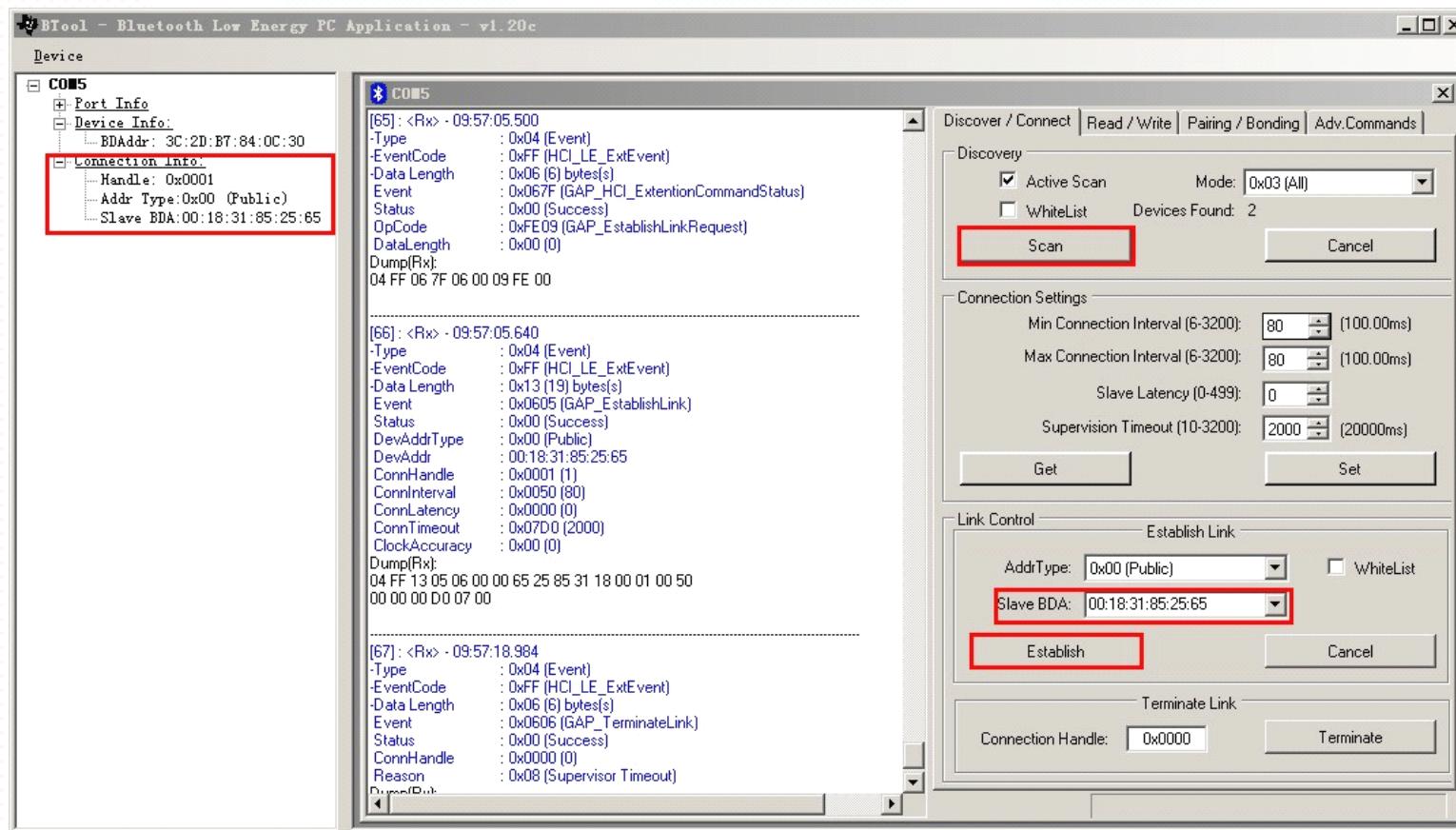
Btool操作示范—第1步，记住硬件地址

读取物理地址



Btool操作示范—第2步,扫描和连接

扫描，选取，连接指定从设备



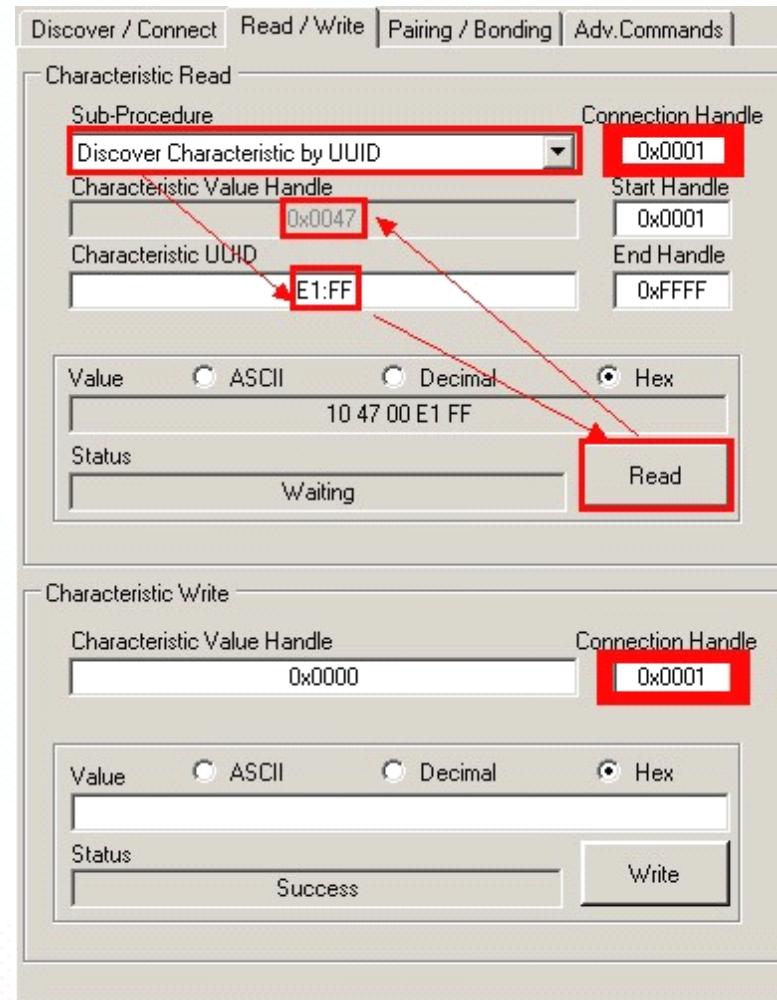
Btool操作示范—第3步,寻找UUID定义

寻找服务以及特征值对应的UUID

```
Workspace  
CC2540DK-mini Keyfob Slave  
Files  
KeyFobDemo - CC2540DK-...  
APP  
HAL  
LIB  
OSAL  
PROFILES  
accelerometer.c  
accelerometer.h  
battservice.c  
battservice.h  
devinfoservice.c  
devinfoservice.h  
gap.c  
gapbondmgr.c  
gapbondmgr.h  
gapgattserver.h  
gatt_uuid.c  
gatt_uuid.h  
gattservapp.h  
peripheral.c  
peripheral.h  
proxreporter.c  
proxreporter.h  
simplekeys.c  
simplekeys.h  
TOOLS  
KeyFob_Main.c | simplekeys.c simplekeys.h  
*****  
* CONSTANTS  
*/  
  
// Profile Parameters  
#define SK_KEY_ATTR 0 // RW uint8  
  
// SK Service UUID  
#define SK_SERV_UUID 0xFFE0  
  
// Key Pressed UUID  
#define SK_KEYPRESSED_UUID 0xFFE1  
  
// Key Values  
#define SK_KEY_LEFT 0x01  
#define SK_KEY_RIGHT 0x02  
  
// Simple Keys Profile Services bit fields  
#define SK_SERVICE 0x00000001  
  
*****  
* TYPEDEFS  
*/  
  
*****  
* MACROS  
*/
```

Btool操作示范—第4步，读取handle

通过UUID发现特征值
的handle (0x47)



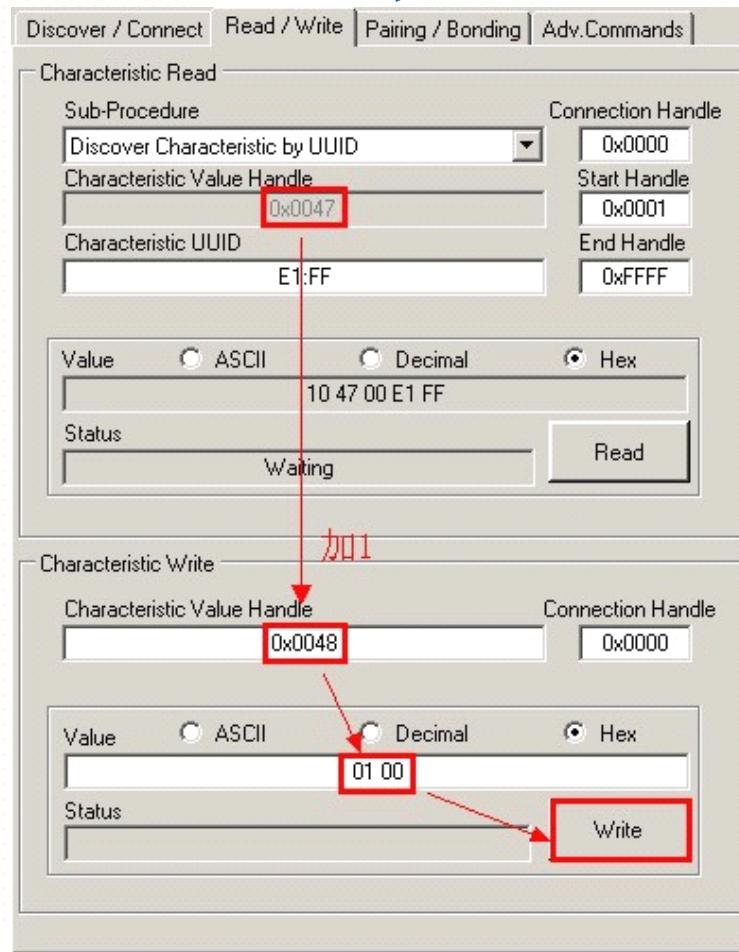
Btool操作示范—第5步，理解handle+1

协议规则决定了Handle列表，编译后会进行定位。

handle+1的位置（协议规定）是对应通知使能开关设置字节
(如果此特征值有通知开关属性)

Handle	Type	Permissions	Value
39	0x2800 (GATT Primary Service UUID)	Read	E0:FF (2 bytes) (0xFFE0 = Simple Keys Service custom UUID)
40	0x2803 (GATT Characteristic Declaration UUID)	Read	10 29:00 E1:FF (5 bytes) (0xFFE1 = Simple Keys Value custom UUID) (0x0029 = handle 41) (0x10 = characteristic properties: notify only)
41	0xFFE1 (Simple Keys state)	(none)	00 (1 byte) (value indicates state of keys)
42	0x2902 (GATT Client Characteristic Configuration UUID)	Read and Write	00:00 (2 bytes) (value indicates whether notifications or indications are enabled)
43	0x2800 (GATT Primary Service UUID)	Read	A1:DD (2 bytes) (0xDDA1 = Other Service custom UUID)

向通知使能对应 handle 写入 01 00



```

KeyFob_Main.c simplekeys.c simplekeys.h | gatt_uuid.c | gatt_uuid.h
=====
 * Profile Attributes - Table
 */
static gattAttribute_t simplekeysAttrTbl[SERVAPP_NUM_ATTR_SUPPORTED] =
{
    // Simple Keys Service 2800
    {
        { ATT_BT_UUID_SIZE, primaryServiceUUID }, /* type */
        GATT_PERMIT_READ, /* permissions */
        0, /* handle */
        (uint8 *)askService /* pValue */
    },
    // Characteristic Declaration for Keys 2803
    {
        { ATT_BT_UUID_SIZE, characterUUID }, /* type */
        GATT_PERMIT_READ,
        0,
        askCharProps
    },
    // Characteristic Value- Key Pressed
    {
        { ATT_BT_UUID_SIZE, keyPressedUUID }, /* type */
        0,
        0,
        askKeyPressed
    },
    // Characteristic configuration 2902
    {
        { ATT_BT_UUID_SIZE, clientCharCfgUUID }, /* type */
        GATT_PERMIT_READ | GATT_PERMIT_WRITE,
        0,
        (uint8 *)skConfig
    },
    // Characteristic User Description 2901 可选
    {
        { ATT_BT_UUID_SIZE, charUserDescUUID }, /* type */
        GATT_PERMIT_READ,
        0,
        skCharUserDesp
    },
}

```

Btool操作示范—第7步，按键变更通知

按下keyfob上的按键，BTool会收到对应的键值通知。键值分别为1, 2, 3。

```
Status : 0x00 (Success)
ConnHandle : 0x0000 (0)
PduLen : 0x03 (3)
Handle : 0x0047 (71)
Value : 02 ←
Dump(Rx):
04 FF 09 1B 05 00 00 00 03 47 00 02

-----
[206] : <Rx> - 11:08:01.359
-Type : 0x04 (Event)
-EventCode : 0xFF (HCI_LE_ExtEvent)
-Data Length : 0x09 (9) bytes(s)
Event : 0x051B (ATT_HandleValueNotification)
Status : 0x00 (Success)
ConnHandle : 0x0000 (0)
PduLen : 0x03 (3)
Handle : 0x0047 (71)
Value : 03 ←
Dump(Rx):
04 FF 09 1B 05 00 00 00 03 47 00 03

-----
[207] : <Rx> - 11:08:02.859
-Type : 0x04 (Event)
-EventCode : 0xFF (HCI_LE_ExtEvent)
-Data Length : 0x09 (9) bytes(s)
Event : 0x051B (ATT_HandleValueNotification)
Status : 0x00 (Success)
ConnHandle : 0x0000 (0)
PduLen : 0x03 (3)
Handle : 0x0047 (71)
Value : 01 ←
Dump(Rx):
04 FF 09 1B 05 00 00 00 03 47 00 01
```

Btool操作示范—第8步，加速度采集

- ◆ 打开加速度使能，向 **handle 0x0034**写入 **01**，常态为关闭以降低功耗
- ◆ **Discover Characteristic by UUID** 寻找三轴加速度通知**XYZ**轴使能**handle**值为 **0x003A, 0x003E, 0x0042,**
- ◆ 向 **handle 0x003B**写入 **01 00**,打开**X**轴通知
- ◆ 向 **handle 0x003F**写入 **01 00** ,打开**Y**轴通知
- ◆ 向 **handle 0x0043**写入 **01 00** ,打开**Z**轴通知
- ◆ 观察**BTool**中的加速度数据包

Btool操作示范—第9步，读取电量

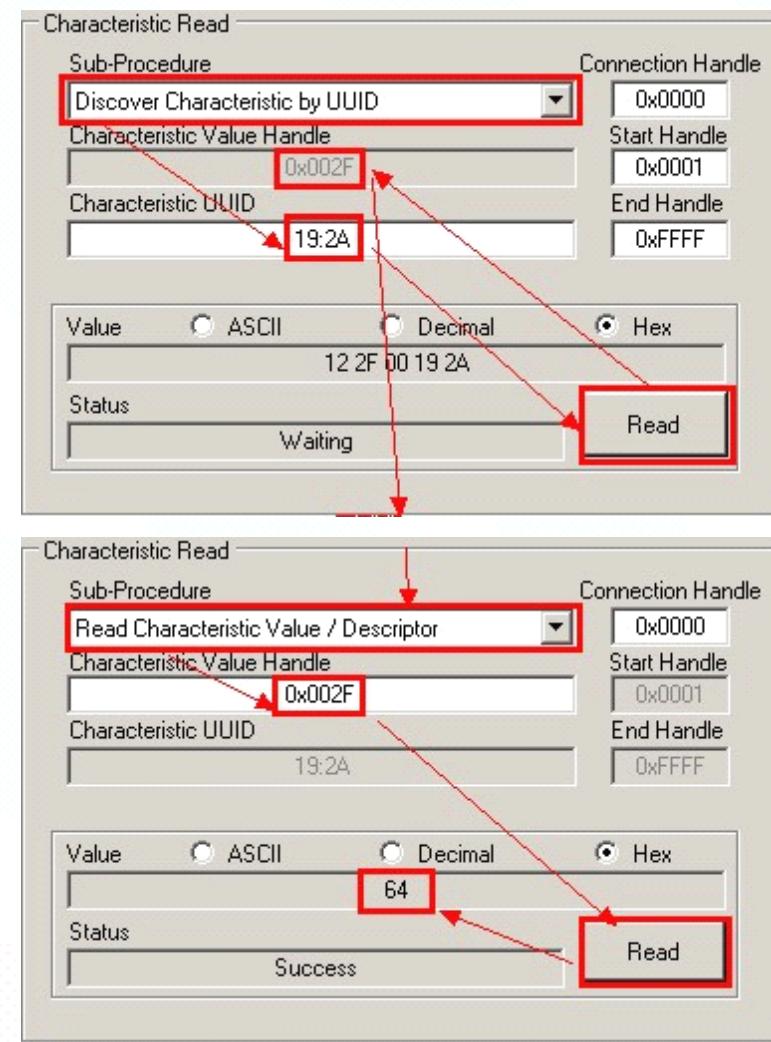
```
static gattAttribute_t battAttrTbl[] =
{
    // Battery Service
    {
        { ATT_BT_UUID_SIZE, primaryServiceUUID }, /* type */
        GATT_PERMIT_READ, /* permissions */
        0, /* handle */
        (uint8 *)&battService /* pValue */
    },

    // Battery Level Declaration
    {
        { ATT_BT_UUID_SIZE, characterUUID },
        GATT_PERMIT_READ,
        0,
        &battLevelProps
    },

    // Battery Level Value 0x2A19
    {
        { ATT_BT_UUID_SIZE, battLevelUUID },
        GATT_PERMIT_READ,
        0,
        &battLevel
    },

    // Battery Level Client Characteristic Configuration
    {
        { ATT_BT_UUID_SIZE, clientCharCfgUUID },
        GATT_PERMIT_READ | GATT_PERMIT_WRITE,
        0,
        (uint8 *) &battLevelClientCharCfg
    },

    // HID Report Reference characteristic descriptor, batter level input
    {
        { ATT_BT_UUID_SIZE, hidReportRefUUID },
        GATT_PERMIT_READ,
        0,
        hidReportRefBattLevel
    }
};
```



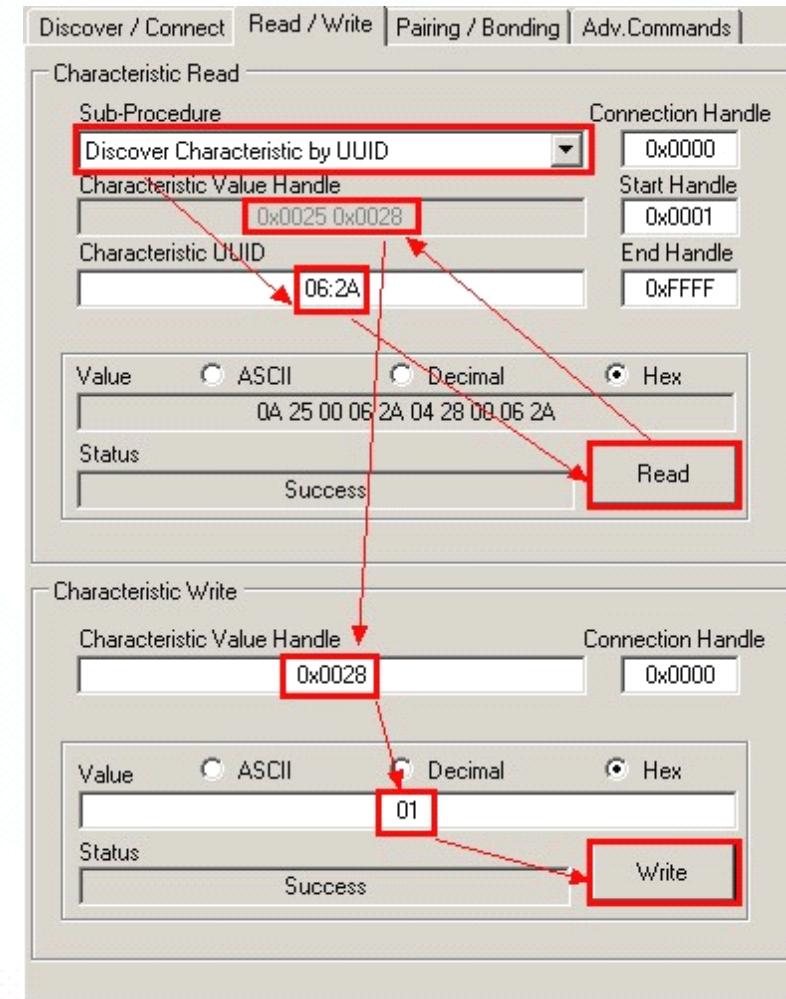
Btool操作示范—第10步,立即报警

```
// Immediate Alert Service Attribut Table
static gattAttribute_t imAlertAttrTbl[] =
{
    // Immediate Alert service
    {
        { ATT_BT_UUID_SIZE, primaryServiceUUID }, /* type */
        GATT_PERMIT_READ, /* permissions */
        0, /* handle */
        (uint8 *)&imAlertService /* pValue */
    },

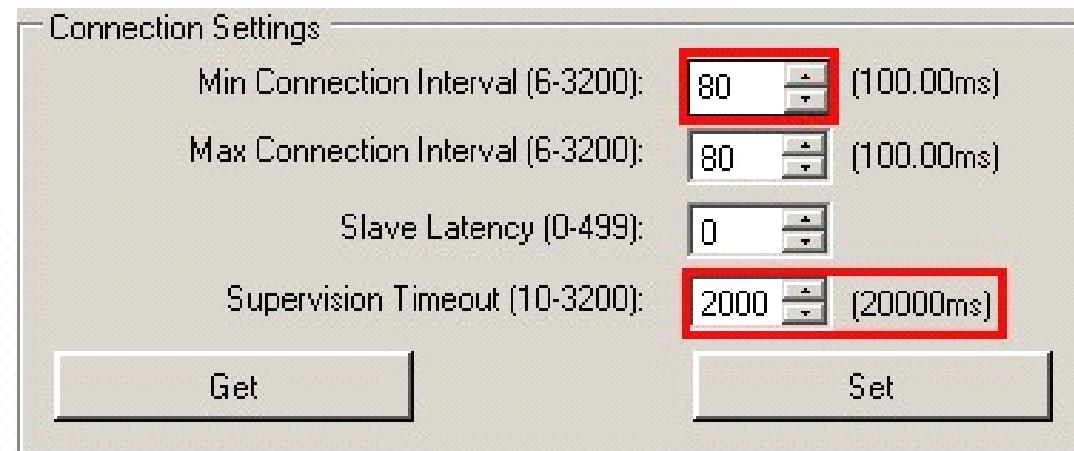
    // Characteristic Declaration
    {
        { ATT_BT_UUID_SIZE, characterUUID },
        GATT_PERMIT_READ,
        0,
        &imAlertLevelCharProps
    },

    // Alert Level attribute 0x2A06
    {
        { ATT_BT_UUID_SIZE, alertLevelUUID },
        GATT_PERMIT_WRITE,
        0,
        &imAlertLevel
    },
};
```

先向0x0025写入报警模式，连接情况下，拔掉dongle 20秒后，会产生丢失报警。



Btool操作示范—第11步,连接参数设定



预设**dongle**的连接参数，必须在连接前设置，之后会按照此参数进行通讯。

HID Emulated Keyboard

虚拟键盘演示



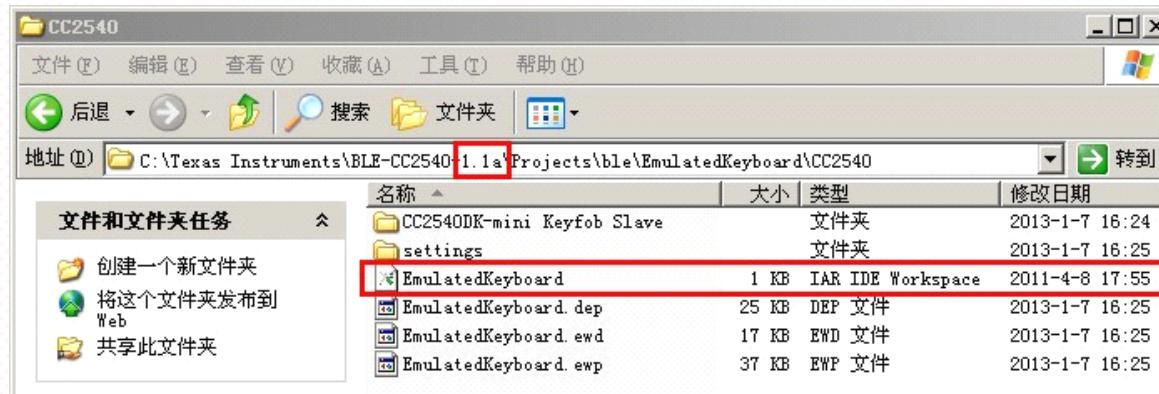
深圳信驰达科技

88-85

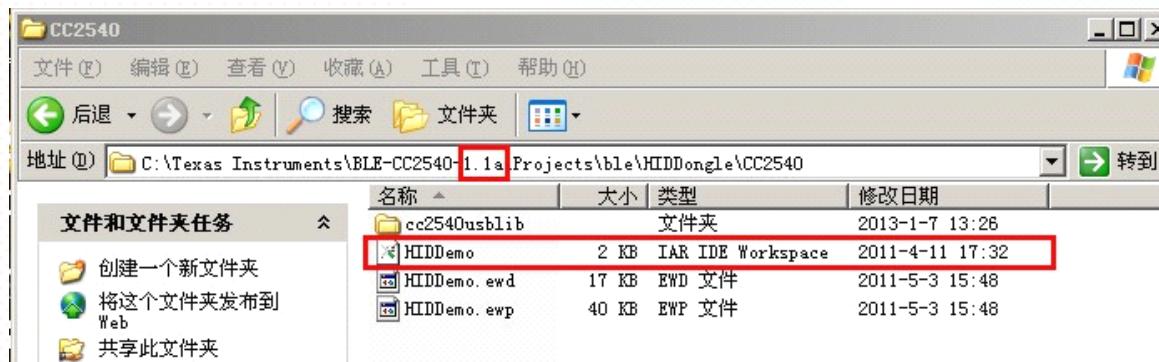
下载模拟键盘的应用代码

下载Keyfob以及Dongle固件(v1.1a):

HCI_EXT_SetBDADDRCmd(bdAddr);



GAPCentralRole_EstablishLink(TRUE, WL_NOTUSED,
ADDRTYPE_PUBLIC, connectAddr);



演示虚拟键盘

Dongle接入USB后，在设备管理器中会显示人体学输入设备，Dongle作为适配器，Keyfob作为键盘：



```
#define CONNECT_ADDR  
{ 0x22, 0x22, 0x22, 0x22, 0x22, 0x22 }
```



```
#define DEVICE_ADDRESS  
{ 0x22, 0x22, 0x22, 0x22, 0x22, 0x22 }
```

