LPRF San Diego
Bluetooth Low Energy Deep Dive

May 2011

Agenda

- **Introduction**
- **Bluetooth Low Energy Protocol Stack (2.5 hours)**
  - Stack Architecture / Overview
  - Link Layer – Basics of BLE communication
  - Generic Access Profile (GAP) – Roles, Device Discovery, Connections, Security
  - Attribute Protocol (ATT) – Attribute Table, Reading and Writing Data
  - Generic Attribute Profile (GATT) – Profiles, Services, Characteristics
- **BLE Industry and Technology Update**
- **CC2540 BLE Software (2.5 hours)**
  - CC2540 Hardware Overview
  - CC2540 BLE Software Architecture and Structure
  - SimpleBLEPeripheral Project – Framework for Custom Applications
  - GAP Role Profiles and Bond Manager
  - GATT Profiles and Services
  - CC2540DK-MINI Kit Overview
- **Hands-on Labs (3 hours)**

Goals for this Training

- Gain a basic understanding of what Bluetooth low energy is, and how BLE communications work at the link-layer
- Understand Bluetooth low energy access control and data communication at the top-layers of the protocol stack
- Learn about the current state of BLE from an industry and technology perspective
- Become familiar with the architecture of CC2540 BLE software, including the OSAL, HAL, BLE stack, profiles, application, and how all of the pieces work together
- Be able to get started with the CC2540DK-MINI kit and use BTool to create a BLE connection
- Be able to open up, build, and debug projects on the CC2540 using IAR Embedded Workbench and the CC Debugger
- Be able to modify the existing GAP role profiles
- Be able to modify existing GATT attribute profiles, or create new ones
- Understand the sample applications

What is Bluetooth Low Energy?

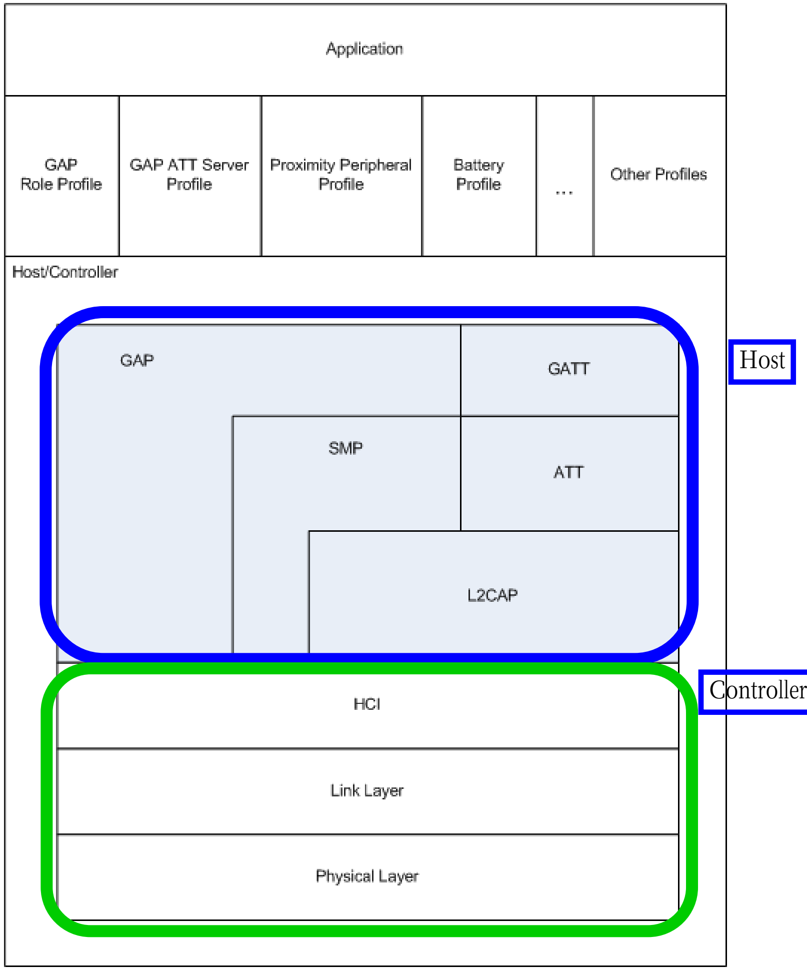
- A wireless protocol standard overseen by the Bluetooth Special Interest Group (BT-SIG), comprised of member companies including Texas Instruments
- The primary new feature added to the Bluetooth standard in version 4.0 of the Bluetooth core specification (adopted in June 2010)
- Targeted towards wireless applications with low-power, low-latency, and low-throughput requirements
- Primarily centered around the mobile phone and PC ecosystem, but can be used for other applications as well
- Expected to be found in **billions** of devices over the next five years
- Not backwards compatible with classic Bluetooth devices

Agenda

- **Introduction**
- **Bluetooth Low Energy Protocol Stack (2.5 hours)**
  - Stack Architecture / Overview
  - Link Layer – Basics of BLE communication
  - Generic Access Profile (GAP) – Roles, Device Discovery, Connections, Security
  - Attribute Protocol (ATT) – Attribute Table, Reading and Writing Data
  - Generic Attribute Profile (GATT) – Profiles, Services, Characteristics
- **BLE Industry and Technology Update**
- **CC2540 BLE Software (2.5 hours)**
  - CC2540 Hardware Overview
  - CC2540 BLE Software Architecture and Structure
  - SimpleBLEPeripheral Project – Framework for Custom Applications
  - GAP Role Profiles and Bond Manager
  - GATT Profiles and Services
  - CC2540DK-MINI Kit Overview
- **Hands-on Labs (3 hours)**

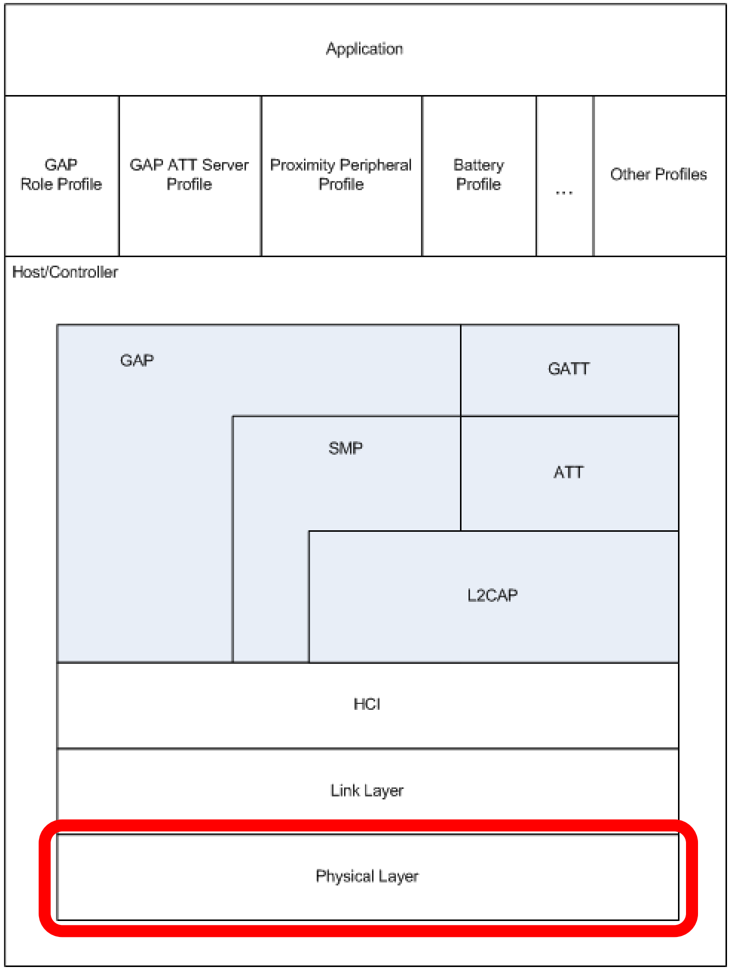
Bluetooth Low Energy Protocol Stack Architecture / Configurations

- Protocol stack consists of two main sections:
  - **Controller**
  - **Host**
- Profiles and Application sit on top of the GAP and GATT layers of the host
- In a “single-device solution” (or “single-chip solution”), the host, controller, profiles, and application are all implemented together on the same chip
- In a “dual-device solution”, the BLE controller is implemented on one device, while the host, application, and profiles are implemented separately
- In a “network processor”, the host and controller are implemented together, but the application and profiles sit on another device (such as a PC or external microcontroller)
- CC2540 can support any of these configurations



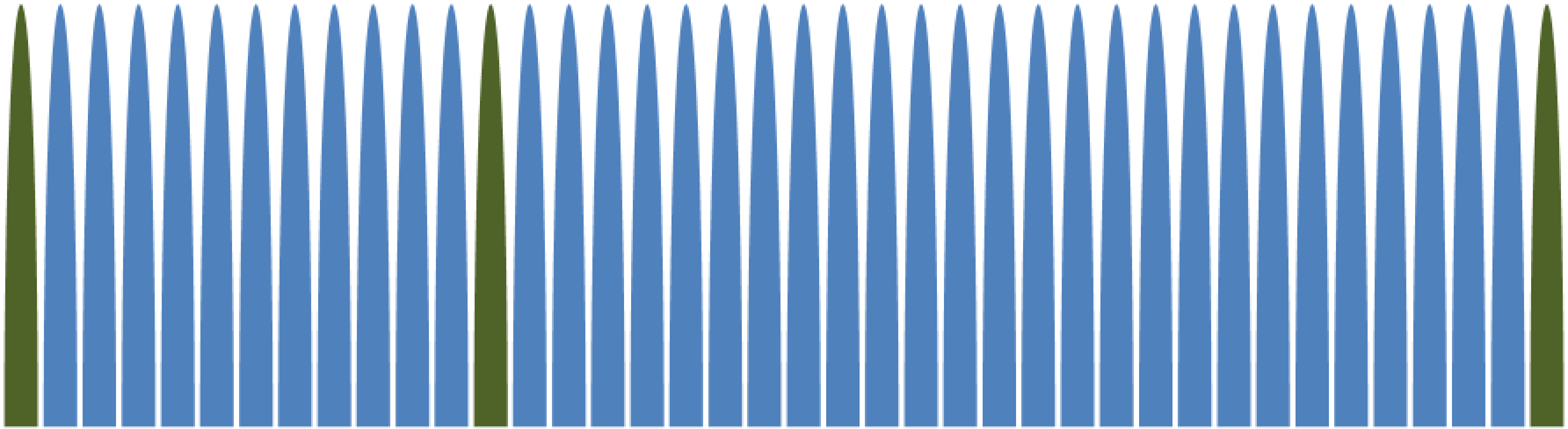
Bluetooth Low Energy Protocol Stack: Physical Layer

- RF Specifications
  - Operates in 2.4 GHz ISM band
  - GFSK modulation 不同通道的频率间隔为2MHz
  - 40 channels with 2 MHz spacing
    - 3 fixed advertisting channels for broadcasting, which avoid 802.11 interferance 3个固定通道用作广播
    - 37 adaptively frequency hopped dynamic data channels 37个动态数据通道
- Physical layer can be combined with standard Bluetooth RF in a dual-mode device 物理层可以与经典蓝牙联合——兼容？
- 2 MHz spacing allows for better adjacent channel rejection



BLE Link Layer: Channels

3 Advertising Channels and 37 Data Channels

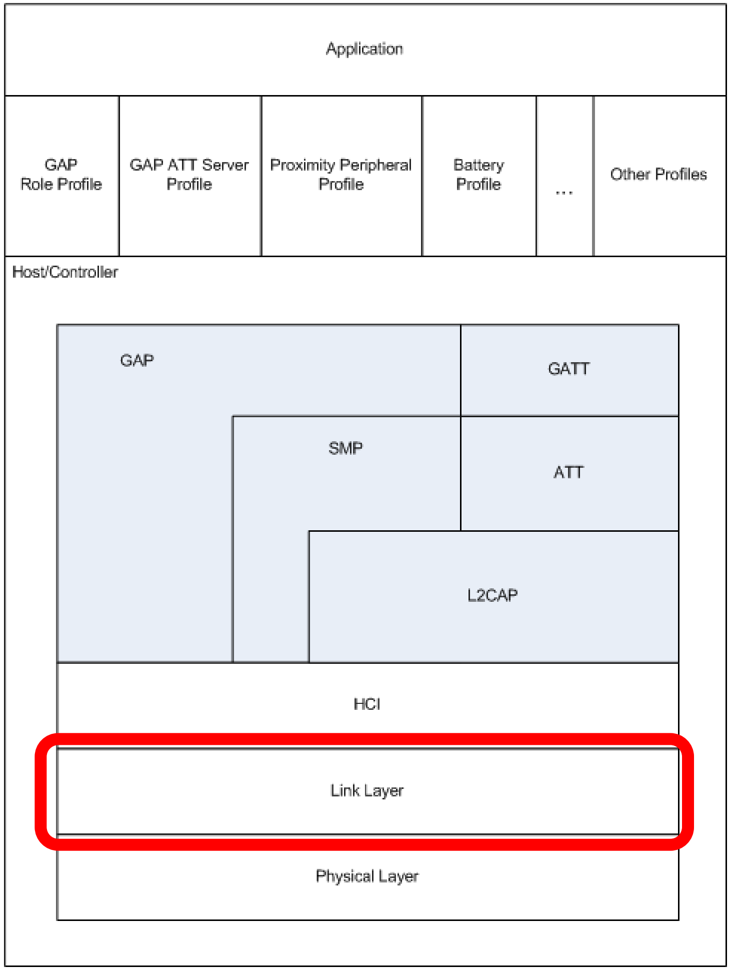


LL	Frequency
37	2402 MHz
0	2404 MHz
1	2406 MHz
2	2408 MHz
3	2410 MHz
4	2412 MHz
5	2414 MHz
6	2416 MHz
7	2418 MHz
8	2420 MHz
9	2422 MHz
10	2424 MHz
38	2426 MHz
11	2428 MHz
12	2430 MHz
13	2432 MHz
14	2434 MHz
15	2436 MHz
16	2438 MHz
17	2440 MHz
18	2442 MHz
19	2444 MHz
20	2446 MHz
21	2448 MHz
22	2450 MHz
23	2452 MHz
24	2454 MHz
25	2456 MHz
26	2458 MHz
27	2460 MHz
28	2462 MHz
29	2464 MHz
30	2466 MHz
31	2468 MHz
32	2470 MHz
33	2472 MHz
34	2474 MHz
35	2476 MHz
36	2478 MHz
39	2480 MHz

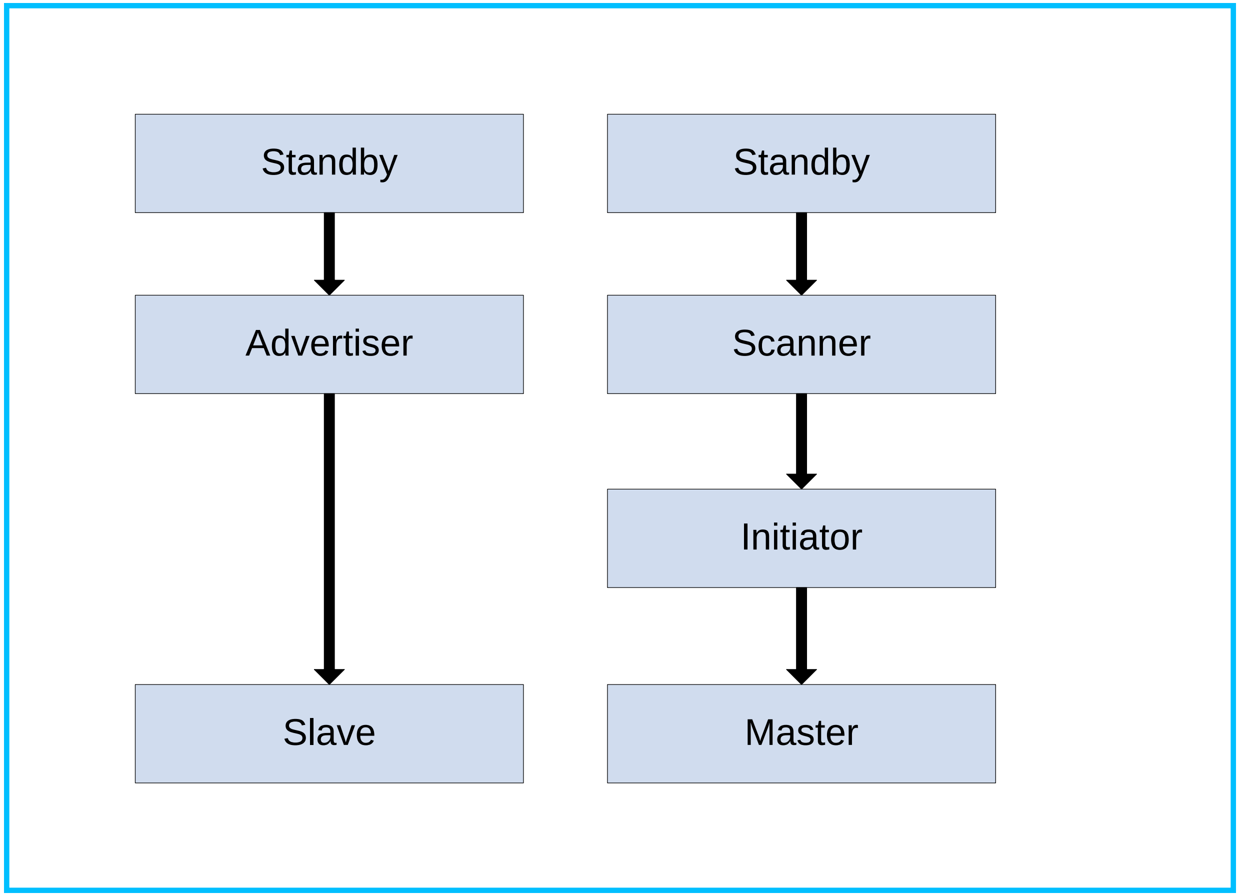
BLE Link Layer: States and Network Topology

星型拓扑网络结构
一主最多可连接3个从机
从机只能拥有一个主机

- There are six possible Link Layer states of a BLE device:
  - **Standby** - device is not transmitting or receiving any data, and is not connected to any other device
  - **Advertiser** - periodically broadcasting advertisements
  - **Scanner** - actively looking for advertisers
  - **Initiator** - actively trying to initiate a connection with another device
  - **Master** - connected to another device as a master
  - **Slave** - connected to another device as a slave
- BLE is a star topology network:
  - Master device “manages” the connection, and can be connected to multiple slaves
  - Slave device can only be connected to one master



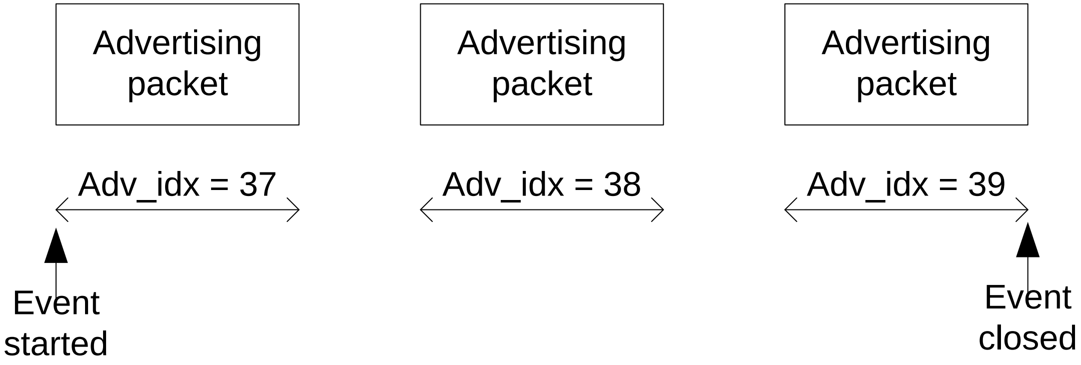
BLE Link Layer: States Flow Chart



BLE Link Layer: Advertisement Events

- A device in the advertising state transmits advertising packets
  - Advertising packets can contain a data payload 广播包可以包含广播数据
  - Advertising packets can be directed towards a specific scanner device, or undirected 广播包可以无指定或对指定的设备发送
  - Advertisements can be connectable or non-connectable (and therefore just used for broadcast of data) 广播中可以声明该器件是可以连接还是不可连接的
- During one “advertising event”, an advertisement packet is transmitted on each of the three advertising channels (37, 38, and 39)

在一次广播事件中，广播包三个广播频道中同时发送



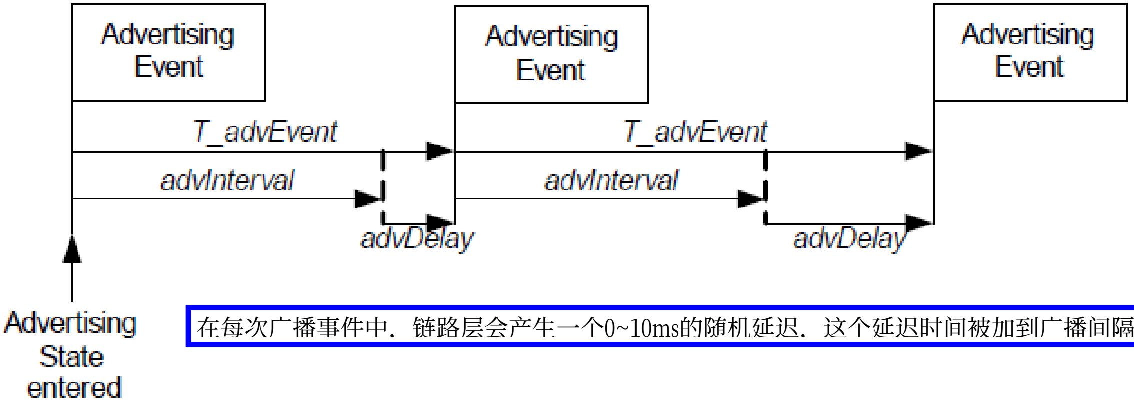
BLE Link Layer: Advertisement Intervals

广播器件有一个叫做"advertising interval"的参数，这个参数决定两次广播事件的最大时间间隔

- The advertising device has an “advertising interval”, which is the minimum amount of time between two advertising events
- Advertising Interval can be any amount of time between 20ms and 10.24s

广播时间间隔可以是20ms~10.24s

- The Link Layer generates a pseudo-random amount of time between 0ms and 10ms (“advertising delay”) during each advertising event. This delay is added to the advertising interval before the next advertising event, in order to prevent “beating” from multiple devices



在每次广播事件中．链路层会产生一个0~10ms的随机延迟．这个延迟时间被加到广播间隔中去．避免多器件的碰撞

BLE Link Layer: Advertisement Types

- Advertising packets can contain a data payload, and therefore broadcast data without a connection
- Four types of advertisements:
  - Connectable undirected- any scanner device can initiate a connection with this advertiser 未指定、可连接
  - Connectable directed- only one specific device can initiate a connection with this advertiser 指定、可连接
  - Non-connectable undirected- no devices can initiate a connection with this advertiser; primarily used for general broadcast of data 未指定、不可连接
  - Discoverable undirected- any scanner device can request more information from the advertising device, but no devices can initiate a connection with it 未指定、不可见

BLE Link Layer: Scanning

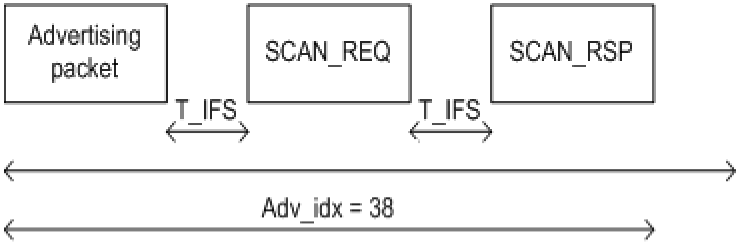
- Passive Scanning

被动扫描：
—扫描者监听广播频道的广播包
—当接收到一个广播包后，将其上传到host层

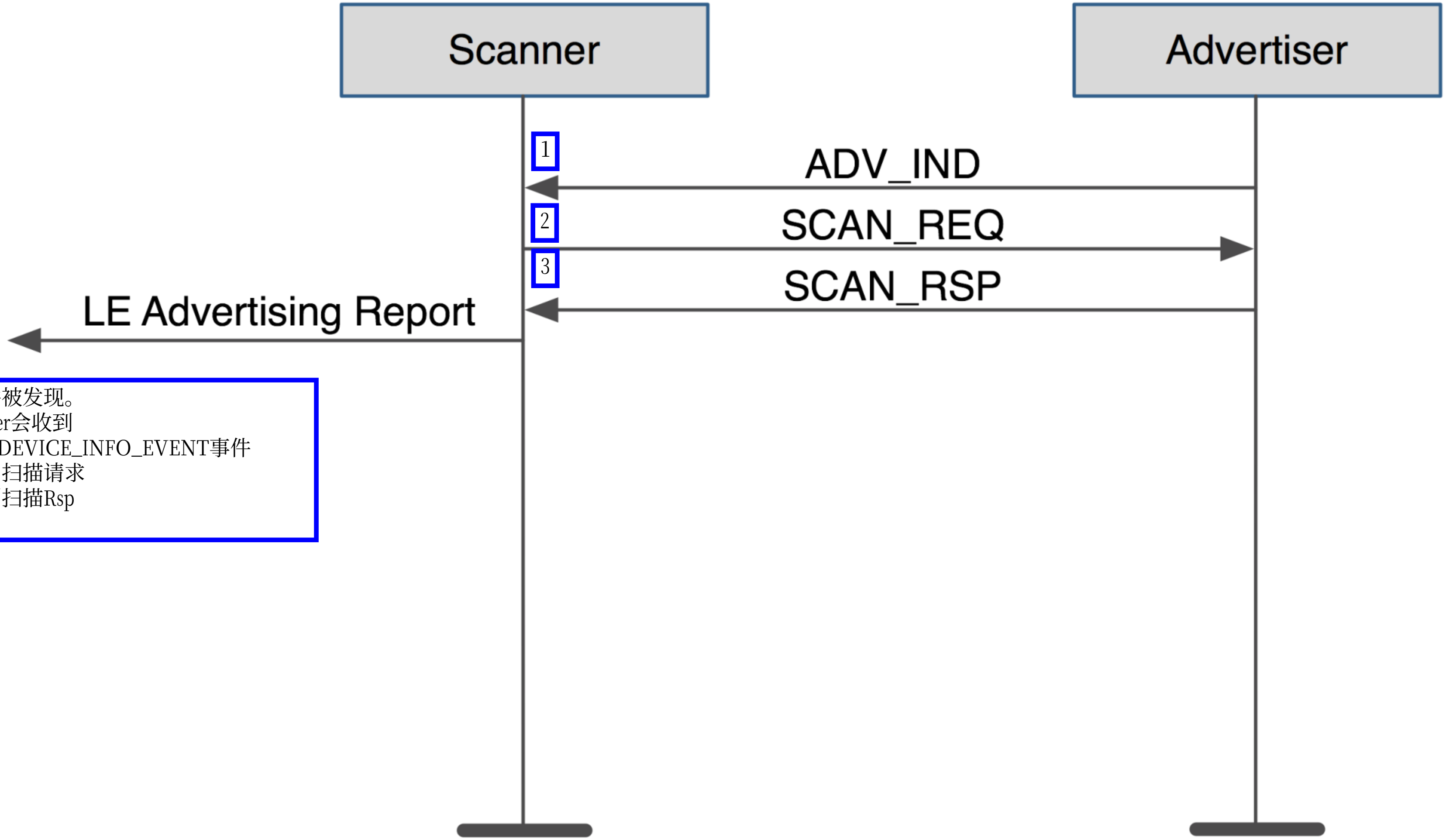
  - Scanner listens on advertising channels for advertising packets
  - When an advertisement packet is received, it passes the information up to the host

主动扫描
—扫描者监听广播频道的广播包
—当收到广播包后，回应一个"scan request"包
—广播设备则响应一个"scan reponse"包

- Active Scanning
  - Scanner listens on advertising channels for advertising packets
  - When an advertisement packet is received, it responds with a “scan request” packet
  - Advertiser then responds back with a “scan response” packet (this packet can contain additional data from advertiser)

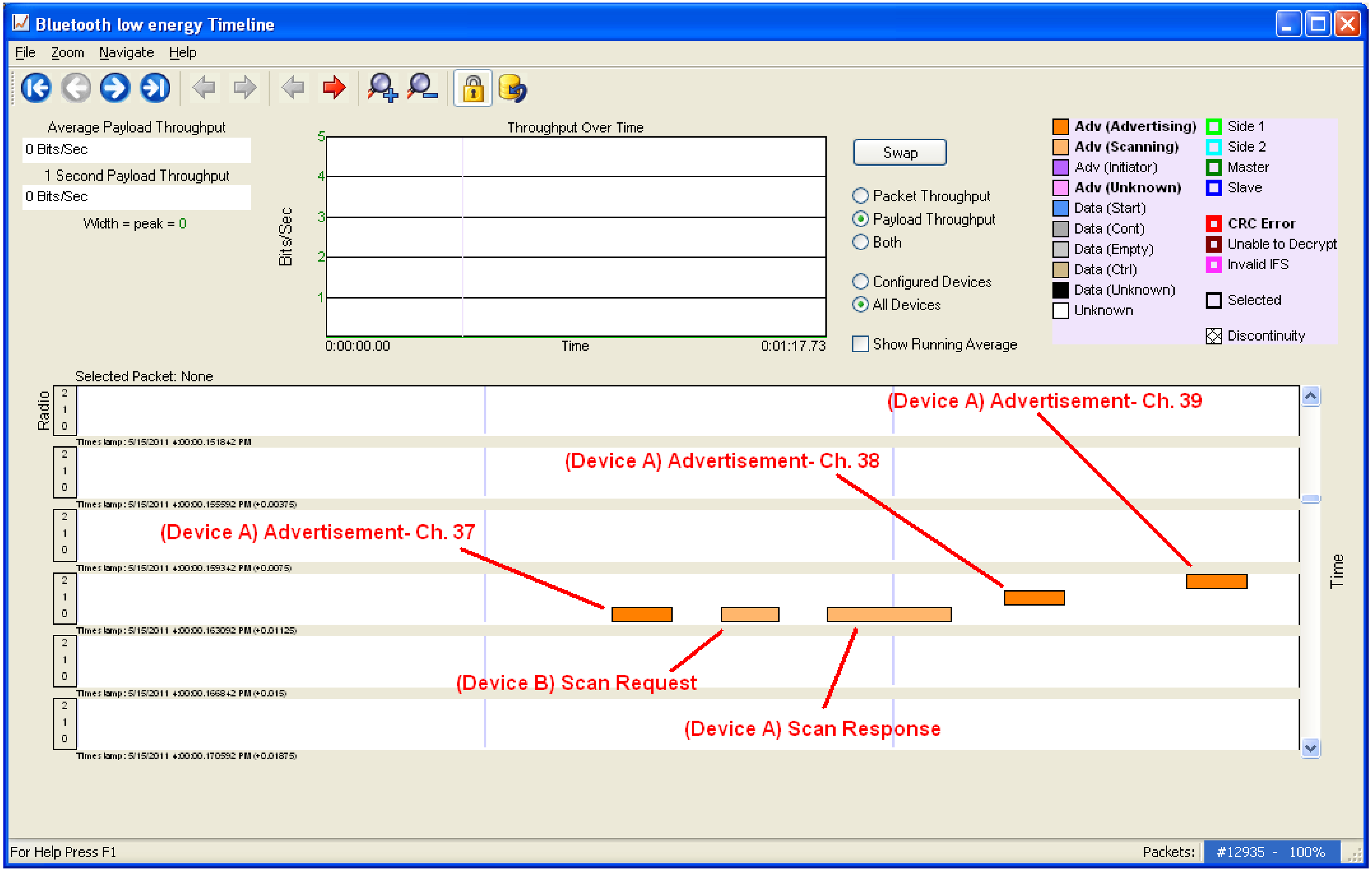


Active Scanning Packet Flow



1.器件被发现。
Scanner会收到
GAP_DEVICE_INFO_EVENT事件
2.发出扫描请求
3.收到扫描Rsp

Demonstration: Advertising and Scanning



Average Payload Throughput
0 Bits/Sec

1 Second Payload Throughput
0 Bits/Sec

Width = peak = 0

0:00:00.00 Time 0:01:17.73

Selected Packet: None

Timestamp: 5/15/2011 4:00:00.151842 PM

Timestamp: 5/15/2011 4:00:00.155592 PM (+0.00375)

Timestamp: 5/15/2011 4:00:00.159342 PM (+0.0075)

Timestamp: 5/15/2011 4:00:00.163092 PM (+0.01125)

Timestamp: 5/15/2011 4:00:00.166842 PM (+0.015)

Timestamp: 5/15/2011 4:00:00.170592 PM (+0.01875)

Packets: #12935 - 100%

(Device A) Advertisement- Ch. 39

(Device A) Advertisement- Ch. 38

(Device A) Advertisement- Ch. 37

(Device B) Scan Request

(Device A) Scan Response

BLE Link Layer: Connection Initiation

- After a scanner device has scanned a connectable advertisement message, it can become an “initiator” by sending a “connection request” packet to the advertiser

在扫描设备 扫描到一个可连接的广播消息后，扫描设备可以通过发送"connection request"数据包给广播设备，而成为(连接的)发起者

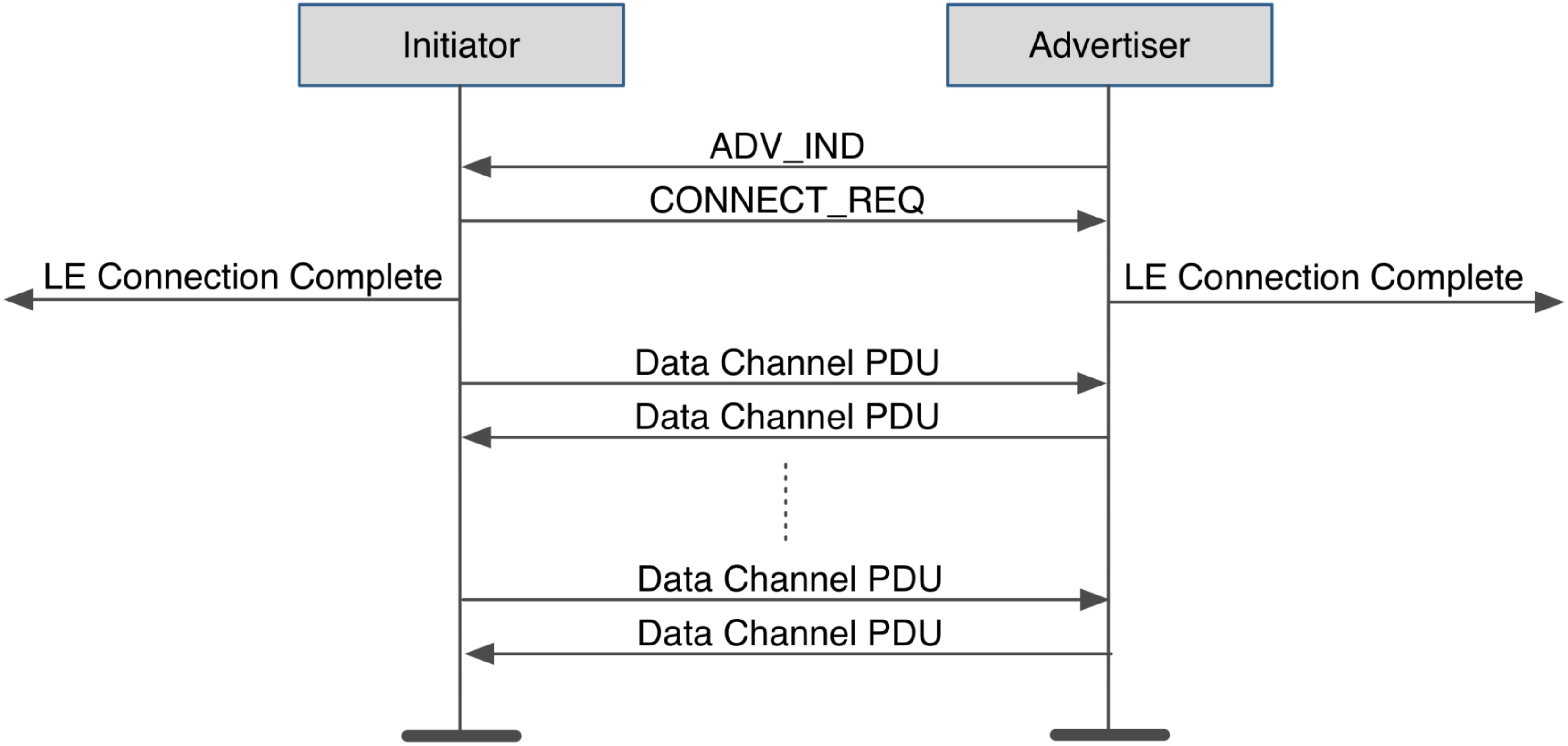
- Connection request contains a set of link layer parameters for the slave device, which dictate the channels and timing requirements for the connection

Connection Request包含从机设备一系列的链路层参数，这些参数声明连接时的通道及时序要求。

- If the advertiser accepts the connection, both devices enter a connected state, with the initiator becoming the “master” and the advertiser becoming the “slave”

如果广播设备接受了连接，那么这两个器件就进入了连接状态。而发起者就成为了主机，广播设备则成为了从机

Connection Request Packet Flow



BLE Link Layer: Connection Parameters

- Channel Map- indicates which data channels are used during the connection 数据通道
- Hop Increment- random value between 5 and 16 for channel selection algorithm
- Connection Interval- multiple of 1.25ms in range of 7.5ms and 4.0s 连接间隔
- Supervision Timeout- multiple of 10ms in the range of 100ms and 32.0s. Must be larger than: 监督超时

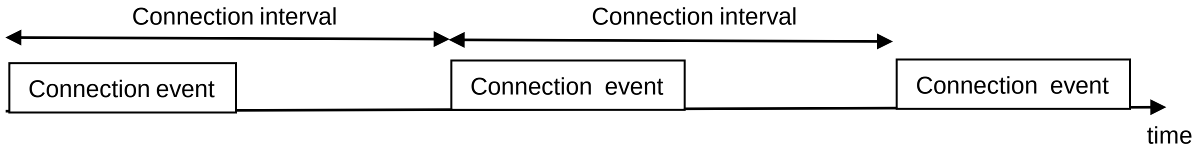
  (1 + slaveLatency) * (ConnInterval)
- Slave Latency- any value between 0 and 499, though it cannot exceed: 外设空闲时可跳过连接事件的次数

  ((supervisionTimeout / connInterval) – 1)

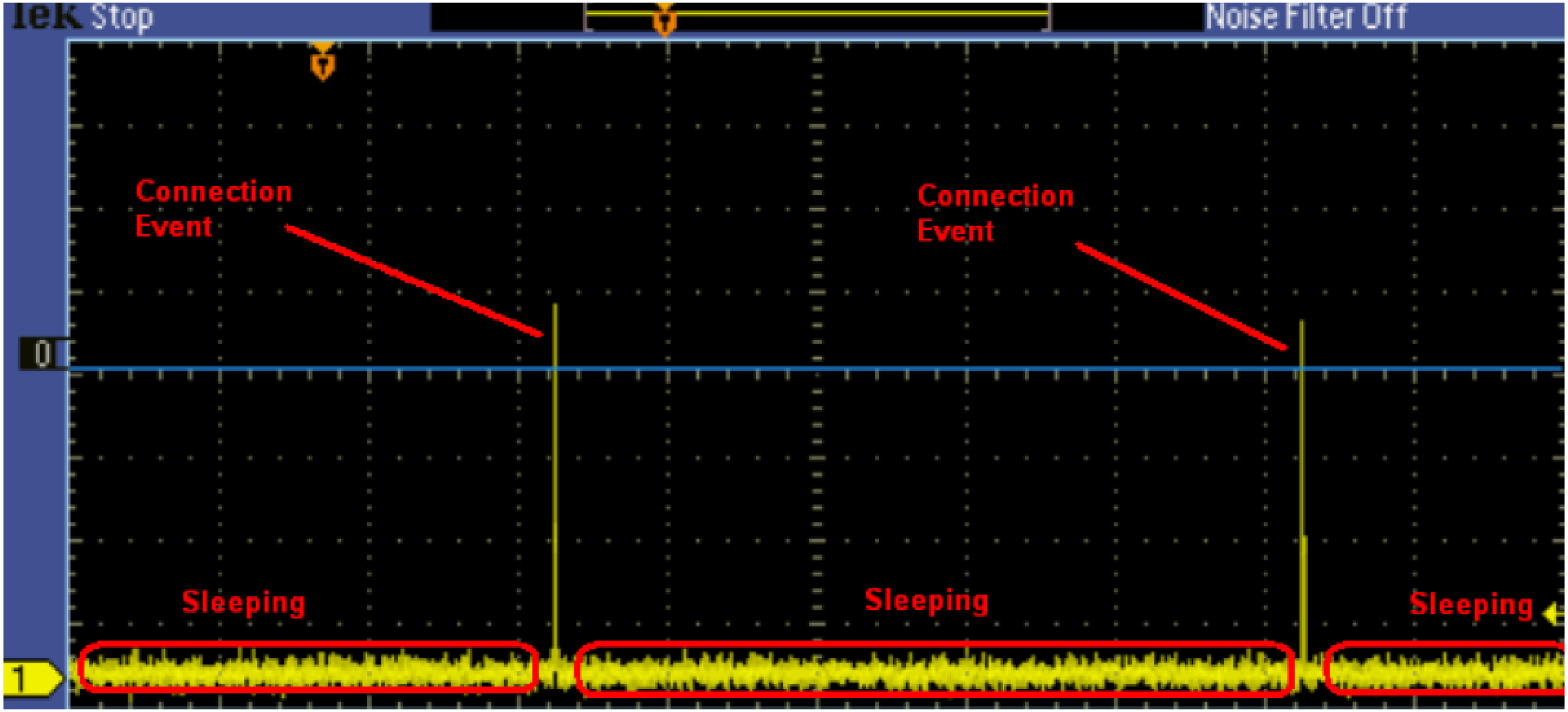
BLE Link Layer: Connection Events

- All communications between two connected devices occur in “connection events” 两个已连接的设备的所有通信发生在连接事件中
- Connection events occurs periodically, with the connection interval parameter specifying the period 连接事件周期性地发生，这个周期由连接间隔参数指定
- Each event occurs on one data channel (channels 0-36), with the hop increment parameter determining the next channel for the next event 每个事件都发生在数据频
次事件使用的频道的跳频
- During each connection event, the master transmits first, and the slave responds 150us later 在每个连接事件中，主机先发起通信，而从机则在150us的延迟后响应
- Master and slave can continue transmitting back and forth as many times as they want during a single connection event 主机和从机可以在一次连接事件中反复地进行多次数据传输
- Connection events occur even when one (or both) sides have no data to send (the exception to this is when slave latency is enabled; more information on next slide). This allows both devices to acknowledge that the other is still there and keeps the connection active.

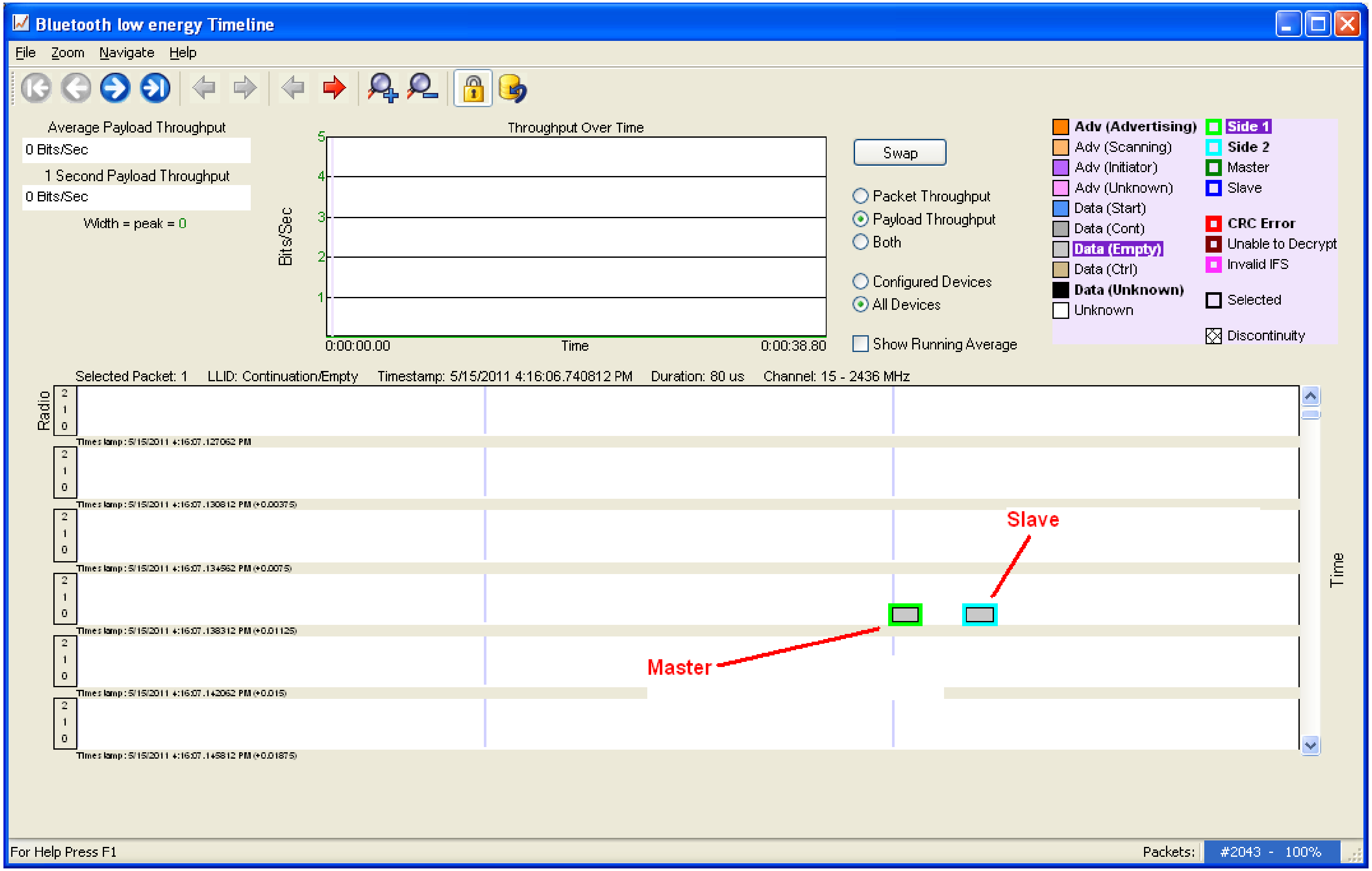
即便没有数据要传输，连接事件依然要执行，这两个器件可以确认另一方还处于连接状态



Connection Interval



Demonstration: Connection



Average Payload Throughput
0 Bits/Sec

1 Second Payload Throughput
0 Bits/Sec

Width = peak = 0

○ Packet Throughput
● Payload Throughput
○ Both

○ Configured Devices
● All Devices

□ Show Running Average

Selected Packet: 1 LLID: Continuation/Empty Timestamp: 5/15/2011 4:16:06.740812 PM Duration: 80 us Channel: 15 - 2436 MHz

Timestamp: 5/15/2011 4:16:07.127062 PM
Timestamp: 5/15/2011 4:16:07.130812 PM (+0.00375)
Timestamp: 5/15/2011 4:16:07.134562 PM (+0.0075)
Timestamp: 5/15/2011 4:16:07.138312 PM (+0.01125)
Timestamp: 5/15/2011 4:16:07.142062 PM (+0.015)
Timestamp: 5/15/2011 4:16:07.145812 PM (+0.01875)

For Help Press F1
Packets: #2043 - 100%

BLE Link Layer: Slave Latency

- Slave latency allows for a slave device to skip connection events if it does not have any data to send

  允许从机设备在没有数据要发送时．可跳过一定次数的连接事件

- The slave latency connection parameter specifies the maximum number of connection events that the slave can skip

  Slave latency 这个参数决定了从机可跳过连接事件的最大次数

- If slave doesn’t respond to master’s packet during a connection event, master will resend the packet in subsequent connection events until the slave responds

  如果在连接事件中，从机没有回应主机的数据包，则主机会一直重发这个数据包直到从机响应(或超时)

- The typical amount of time between two connection events (assuming that the slave skips the maximum number events) is often referred to as the “effective connection interval”

  两次连接事件的时间间隔又称为"有效的连接间隔"

- Example: if connection interval is 100ms and slave latency is set to 4, then the effective connection interval would be 500ms, since slave typically skips four connection events at 100ms intervals
- The slave latency can be any value between 0 and 499, though the effective connection interval must be less than 32.0s

  从机跳过连接事件的次数为0~499可变，但实际上有效连接间隔必须小于32s

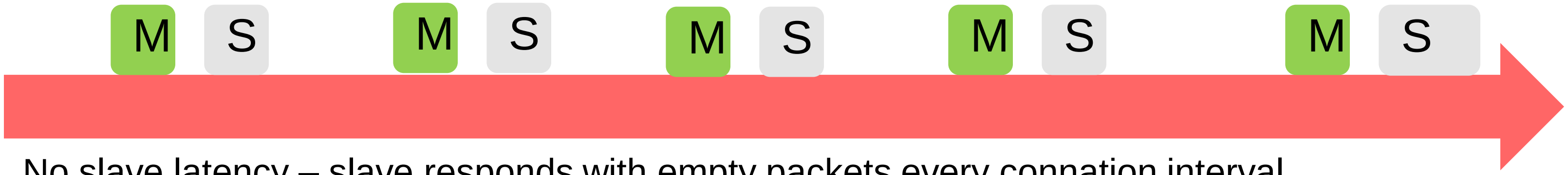
BLE Link Layer: Slave Latency

Slave Latency = OFF

Event

Data to

Send



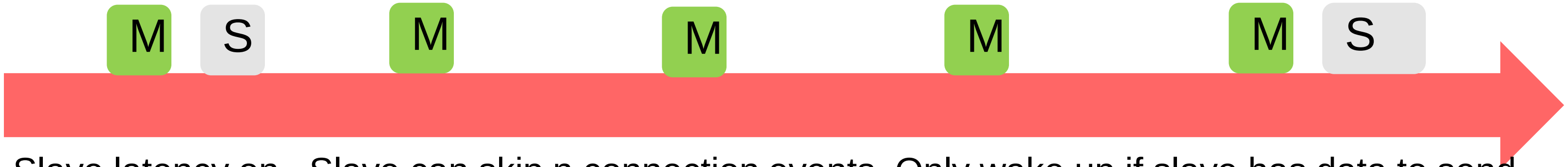
No slave latency – slave responds with empty packets every connation interval.

Slave Latency = ON

Event

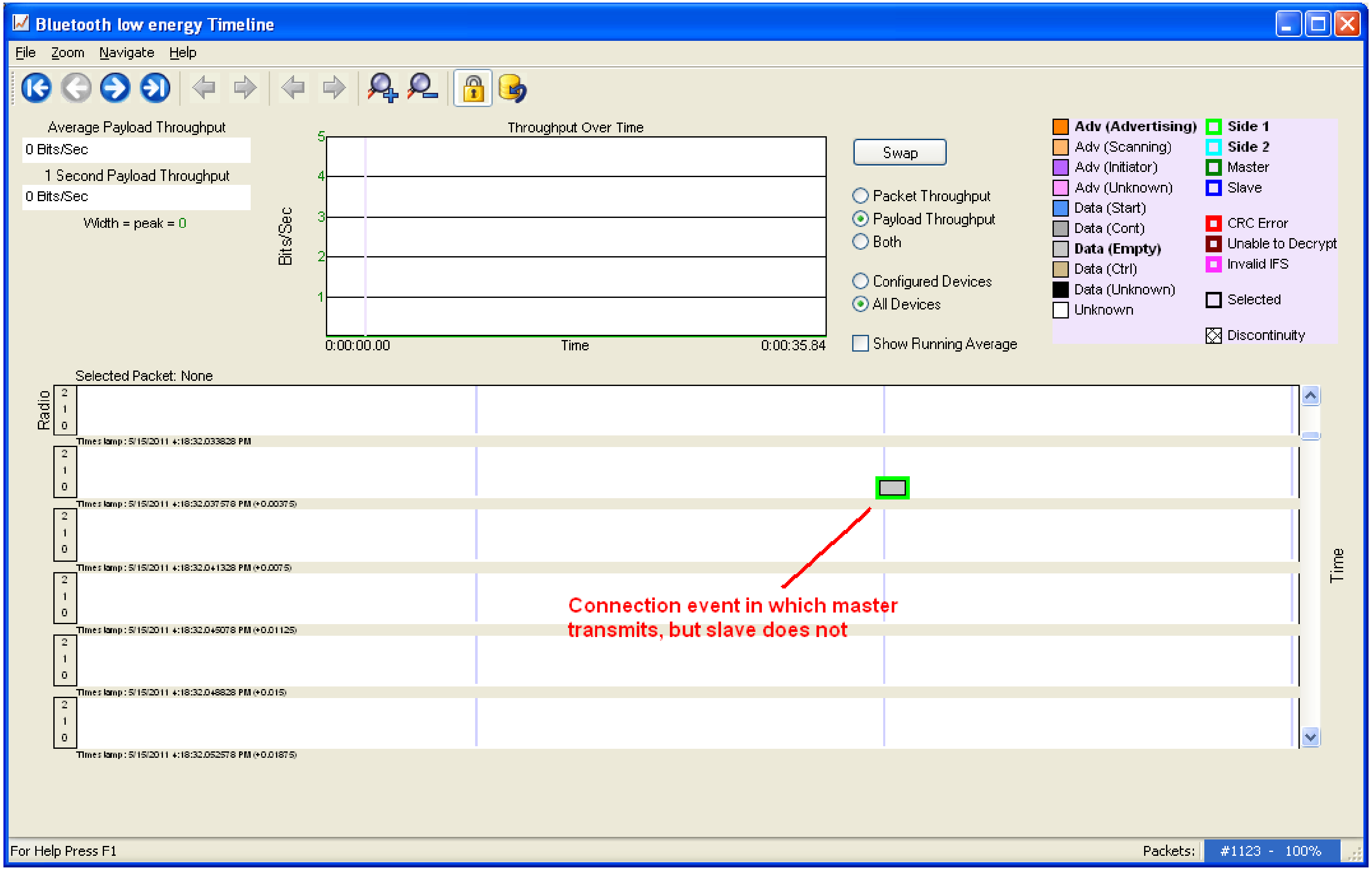
Data to

Send



Slave latency on. Slave can skip n connection events. Only wake up if slave has data to send.

Demonstration: Slave Latency



Average Payload Throughput: 0 Bits/Sec

1 Second Payload Throughput: 0 Bits/Sec

Width = peak = 0

0:00:00.00 – 0:00:35.84

○ Packet Throughput
● Payload Throughput
○ Both

○ Configured Devices
● All Devices

□ Show Running Average

Selected Packet: None

Timestamp : 5/15/2011 4:18:32.033828 PM
Timestamp : 5/15/2011 4:18:32.037578 PM (+0.00375)
Timestamp : 5/15/2011 4:18:32.041328 PM (+0.0075)
Timestamp : 5/15/2011 4:18:32.045078 PM (+0.01125)
Timestamp : 5/15/2011 4:18:32.048828 PM (+0.015)
Timestamp : 5/15/2011 4:18:32.052578 PM (+0.01875)

#1123 - 100%

Connection event in which master transmits, but slave does not

BLE Link Layer:
Connection Parameters Tradeoffs

- Short connection interval:
  - Higher power consumption for both devices
  - Higher throughput in both directions
  - Shorter wait for data to be sent in either direction
- Long connection interval:
  - Lower power consumption for both devices
  - Lower throughput in both directions
  - Longer wait for data to be sent in either direction
- Low / Zero slave latency:
  - Higher power consumption for peripheral
  - Peripheral receives data sent from central device sooner
- High slave latency:
  - Lower power consumption for peripheral during periods when it has no data to send to central device
  - Peripheral may not immediately receive data being sent from central device

短的连接间隔：
—造成彼此更高的器件功耗
—更高的吞吐量
—数据发送短延迟
长的连接间隔：
—节约彼此的器件功耗
—较低的吞吐量
—发送数据延迟长

低/零外设跳连：
—更多的外设功耗
—外设接收数据实时性好
高外设跳连：
—节约外设器件的功耗
—外设接收数据的实时性差

BLE Link Layer:
Connection Update Request

- If the slave does not like the connection parameters (interval, slave latency, or supervision timeout), it can send a connection update request to the master

如果从机不想使用当前的连接参数，可以向主机发送连接更新请求

- Connection update request allows slave device to request a desired connection interval range (minimum and maximum), as well as desired slave latency and supervision timeout

连接更新请求允许从机请求合适的连接间隔范围、合适的跳连次数及监督超时

- Slave device can send a connection update request at any time, allowing for slave applications to dynamically adjust the connection parameters based on application

从机设备可以在任何时候发送连接更新请求，这使得从机设备的应用程序可以动态地调整连接参数

BLE Link Layer: Connection Termination

不论是主机还是从机，都可以无条件地终止当前的连接

一方请求终止，而另一方则必须在断开连接状态之前响应

- A connection can be voluntarily terminated by either the master or the slave for any reason
  - One side initiates termination, and the other side must respond accordingly before both devices exit the connected state
- Connection can also be terminated as a result of a supervision timeout

  连接亦可以由于监督超时而终止
  - The supervision timeout parameter specifies the maximum amount of time that either the master or slave can go before receiving a link-layer packet

    监督超时参数指定两个器件接收链路层数据包的最大时间间隔
  - Supervision timeout value must be greater than the effective connection interval and less than 32.0 seconds

    监督超时必须大于有效的连接间隔，且小于32
  - Both slave and master device maintain their own “Supervision timer”, which resets to zero every time a packet is received
  - If supervision timer ever reaches the supervision timeout, the device considers the connection lost, and exits the connection state (returning to the advertising, scanning, or standby state)

主机和从机保存各自的监督计时器，每次收到数据包，则清零监督计时，一旦监督计时器的数值达到监督超时的数值，就认为连接已经丢失并退出连接状态（回到广播、扫描或者待机状态）

BLE Link Layer: Direct Test Mode

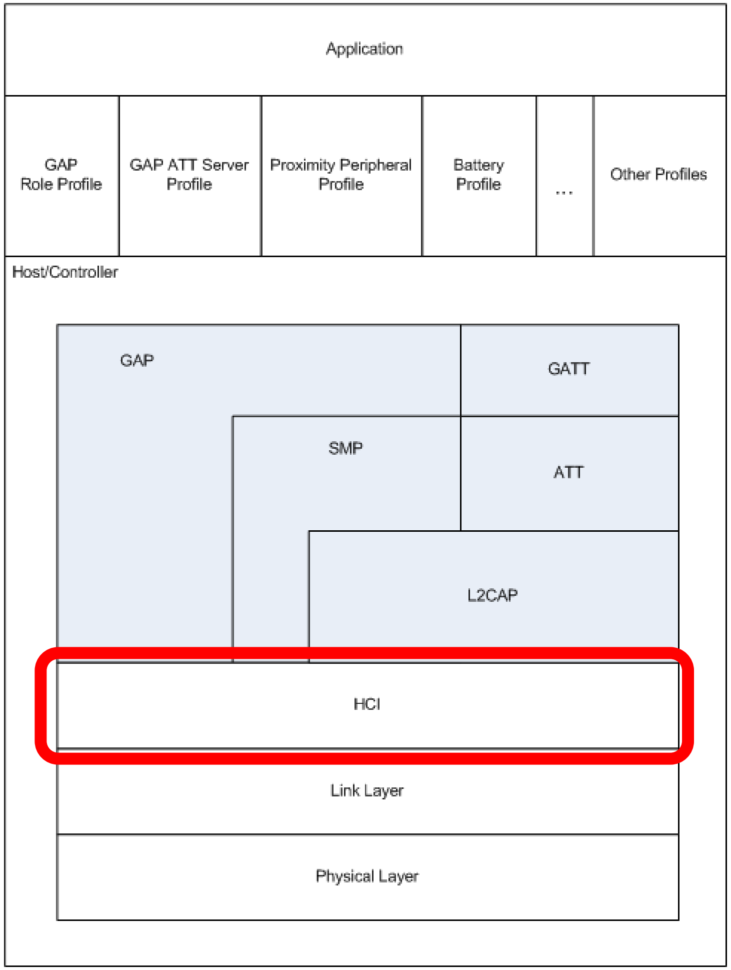
- Allows a tester to directly control the device under test (DUT) in either Rx or Tx mode on any channel with any amount of data

- Used by Bluetooth low energy testers such as the Anritsu MT8852B for RF performance testing

- TI working directly with Anritsu to provide a simple means for test during manufacturing

Bluetooth Low Energy Protocol Stack: Host/Controller Interface (HCI) Overview

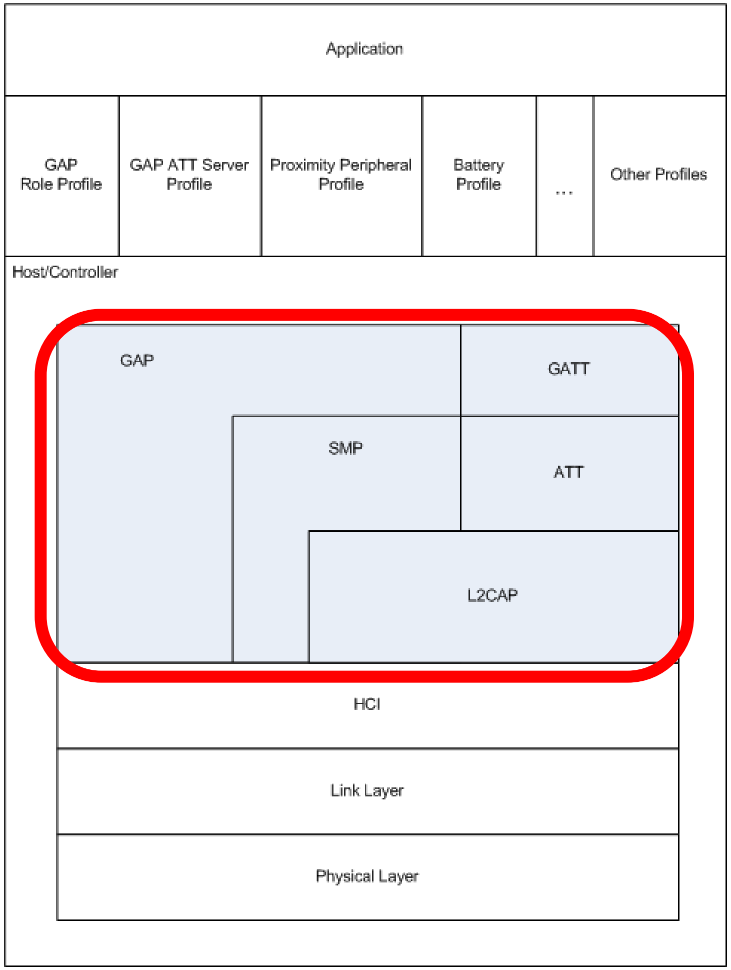
- Reused from standard Bluetooth specification, with new additional commands for low energy-specific functions
- Thin layer; doesn't perform any processing
- In a dual-chip solution (with separate host and controller) allows for host to communicate with controller over a standard interface (UART, USB, SDIO, etc.)
- Used internally by the CC2540 BLE protocol stack for communication between higher and lower layers
- Also allows for custom "vendor-specific commands". In the CC2540, vendor-specific commands can be used by an external source to directly interface with the entire stack or application. This is called a "Network Processor"



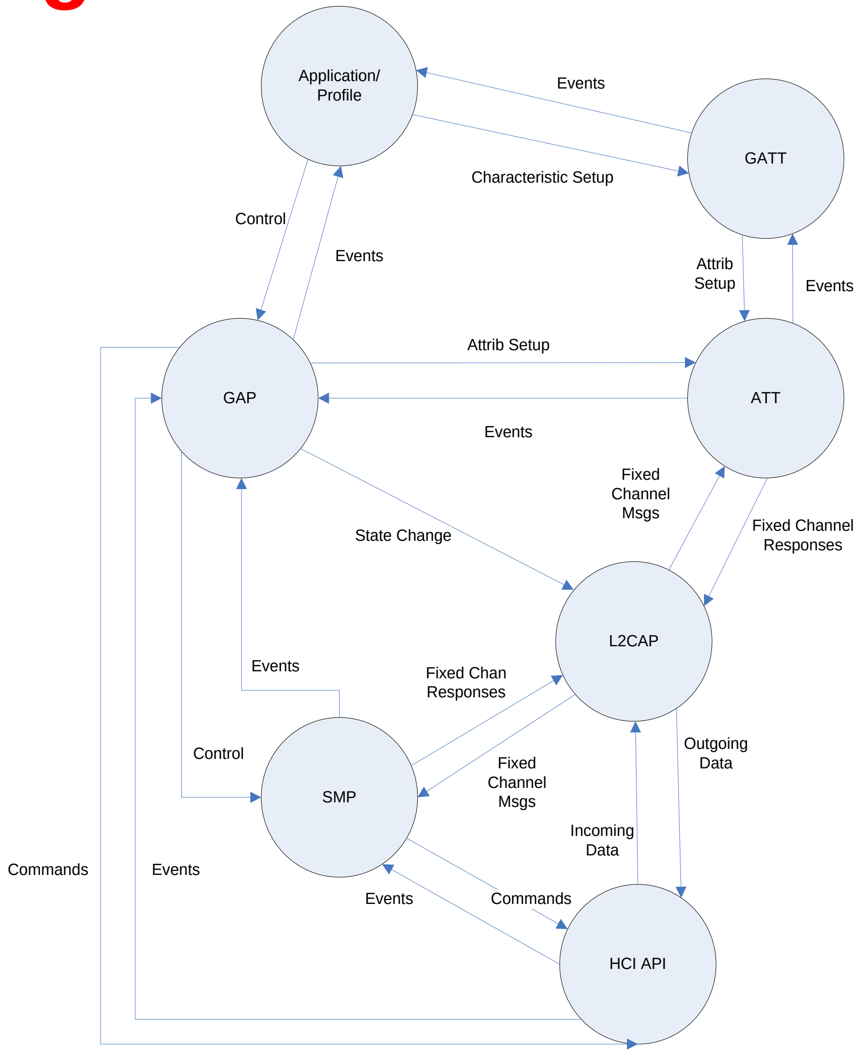
Bluetooth Low Energy Protocol Stack: Host Overview

- The host uses the HCI API to communicate with the lower layers

- The different layers of the host stack manage control messages, event messages, and transmission of data



Bluetooth Low Energy Protocol Stack: Host Message and Data Flow Chart



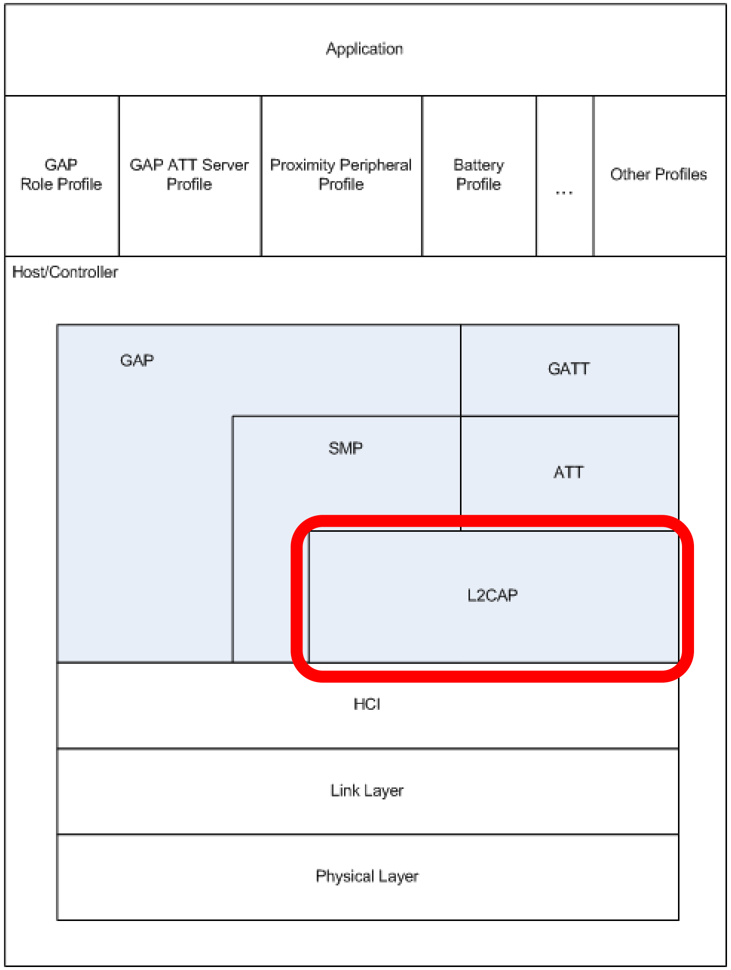
TI confidential information - Strictly Private

Bluetooth Low Energy Protocol Stack: Logical Link Control and Adaptation Protocol (L2CAP)

- Permits upper level protocols and applications to transmit and receive upper layer data packets up to 23 bytes in length

允许协议+应用程序 传输/接收最大23个字节的数据包

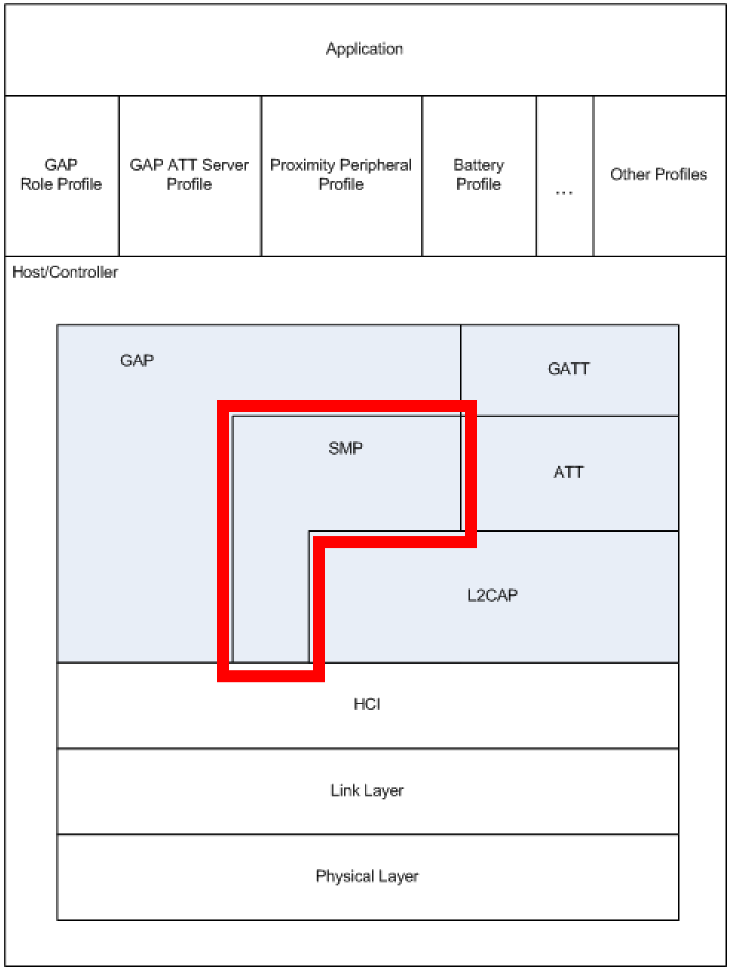
- Provides channel management, allowing for logical channels between two endpoints, supported by the link layer
- Connection Parameter Updates



Bluetooth Low Energy Protocol Stack: Security Manager Protocol (SMP)

配对——设备间加密？
授权——设备信任
绑定——保存安全及身份信息，用于长期的relationship

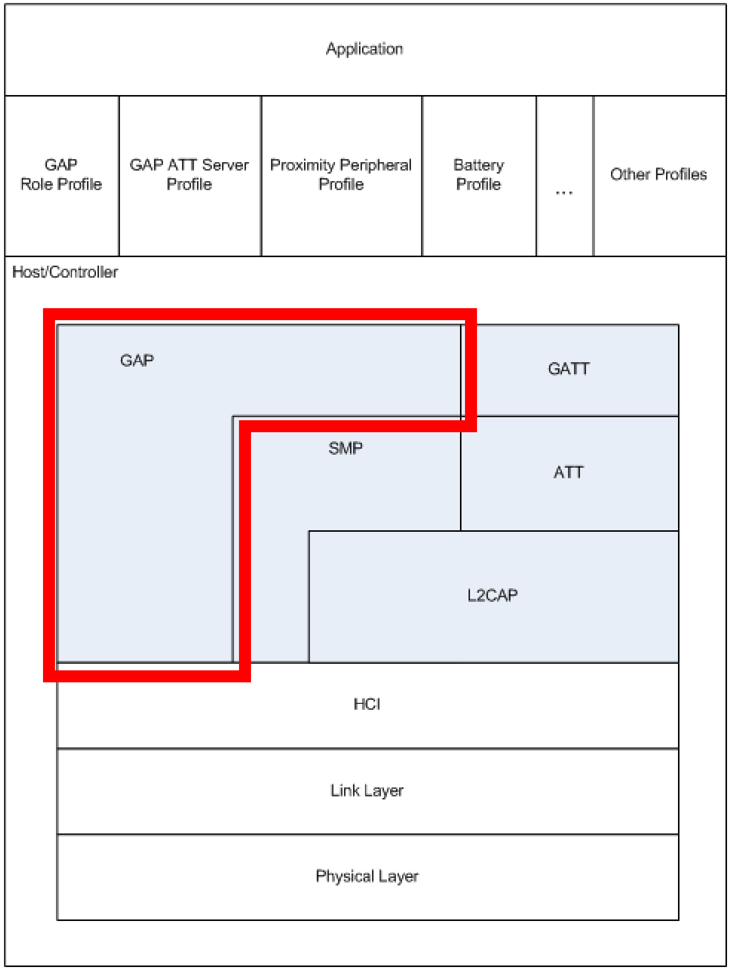
- Performs authentication and key management
- Uses AES-128 as the encryption algorithm for security procedures
- Defines protocol to setup secure link
- Works with GAP to manage relationships between devices:
  - Pairing – encryption between two devices once a connection has been established between them
  - Authentication – verification that a peer device can be trusted, providing protection against “Man-in-the-Middle” attacks
  - Bonding – long-term relationship between devices; security and identity information is saved for re-use next time the devices are connected



Bluetooth Low Energy Protocol Stack: Generic Access Profile (GAP) Overview

- Defines generic procedures for connection-related services:
  - Device Discovery
  - Link Establishment
  - Link Management
  - Link Termination
  - Initiation of security features

- Many GAP functions correspond directly to the functions of the Link Layer in the controller



BLE Generic Access Profile (GAP): Profile Roles

- The GAP layer works in one of four profile roles:
  - Broadcaster – an advertiser that is non-connectable
  - Observer – scans for advertisements, but cannot initiate connections.
  - Peripheral – an advertiser that is connectable and can operate as a slave in a single link layer connection.
  - Central – scans for advertisements and initiates connections; operates as a master in a single or multiple link layer connections.

Temperature Sensor (Broadcaster) → Temperature Display (Observer)

Figure 1 – Temperature Sensor (Broadcaster)

Figure 2 – Temperature Display (Observer)

Watch (Peripheral) ← → Mobile Phone (Central)

Figure 3 - Watch (Peripheral)

Figure 4 – Mobile Phone (Central)

BLE Generic Access Profile (GAP): Profile Multi-Roles

- The BLE specification allows for a few different possible multiple-role configurations:
  - Peripheral and Broadcaster – device operates as a slave in a single link layer connection, but meanwhile also can send out non-connectable advertisements (supported in Beta stack)
  - Peripheral and Observer – device operates as a slave in a single link layer connection, but meanwhile also can scan for advertisements without initiating a connection (not supported in Beta stack)
  - Central and Broadcaster – device scans for advertisements and initiates connections as a master, but also can broadcast non-connectable advertisements (not supported in Beta stack)

BLE Generic Access Profile (GAP): Discoverable Modes

- GAP supports three different discoverable modes:
  - Non-discoverable Mode – No advertisements 无广播
  - Limited Discoverable Mode – Device advertises for a limited amount of time before returning to the standby state 限时的广播
  - General Discoverable Mode – Devices advertises continuously 不断的广播
- GAP uses the HCI to communicate with the controller to turn advertising on and off in the link layer GAP通过HCI与链路层广播开关控件通信
- Peripheral role devices may send out either connectable or non-connectable advertisements while in a discoverable mode Peripheral器件可以发送可连接/不可连接的广播
- Broadcaster role device may only send out non-connectable advertisements while in a discoverable mode Broadcaster器件只发送不可连接的广播

BLE Generic Access Profile (GAP): Advertisement and Scan Response Data

- GAP manages the data that is sent out in advertisement and scan response packets

GAP管理广播数据以及扫描的响应包

- The BLE spec defines several types of advertisement data (“AD” types):
  - Device Services (e.g. “I am a remote control”)
  - Service Solicitation (e.g. “I want a remote control to talk to me”)
  - Device Name
  - “Flags” - describes the discoverable mode and whether device supports standard Bluetooth or just Low Energy
  - Tx Power Level
  - Slave preferred connection interval range
  - Security support
  - Manufacturer-specific data

BLE Generic Access Profile (GAP): Pairing

- Pairing can be initiated by either the central or peripheral device

配对可以由中央设备或外设发起

- During pairing, the two devices generate and exchange short-term keys (STK) which can be used to decrypt data packets

配对过程中，两个设备产生并交换一个短期的密匙，用于数据包的解密

- In addition, either device can request to enable “bonding” to create a long-term relationship between the two devices
  - A long-term key (LTK) is generated, exchanged, and stored allowing device to re-encrypt the link quickly upon re-connection, without going through the complete pairing process once again
  - Profile / Service configuration data is remembered, so that the user does not need to re-configure the device every time they re-connect

器件可以通过请求绑定来产生一个长期使用的密匙，这样在下次连接时就可以跳过配对过程

- During the pairing process, each device states whether it wants authentication to the other device

BLE Generic Access Profile (GAP): Pairing (continued)

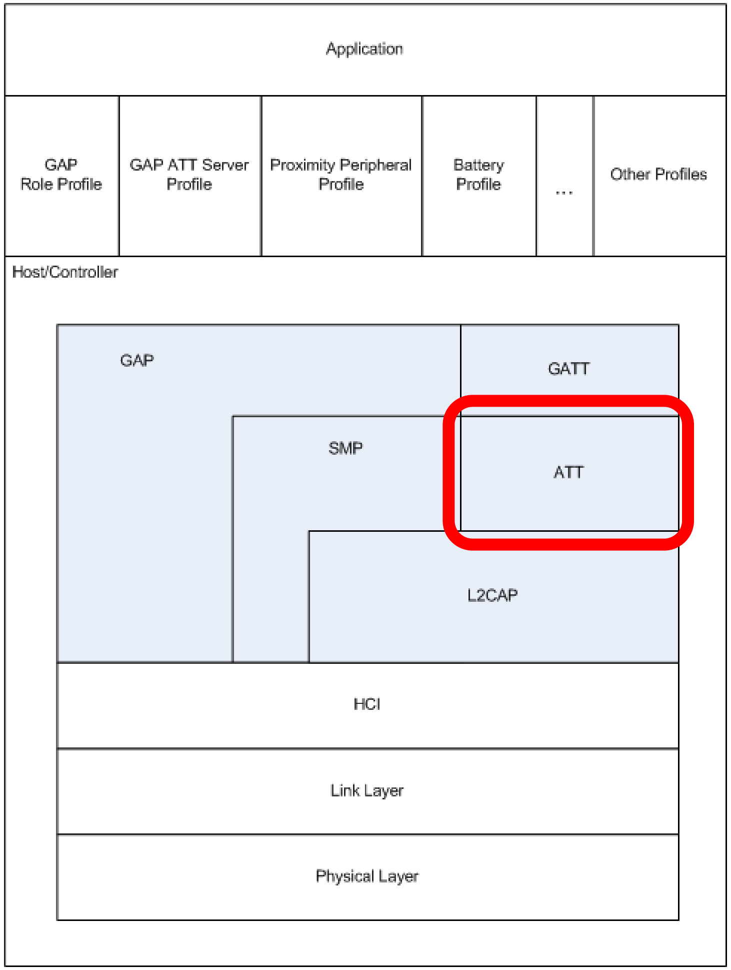
- Each device also states it's input/output capabilities from among these options:
  - DisplayOnly – no way user can input anything into device, but it can output data
  - DisplayYesNo – user can input "yes" or "no" but nothing else; can also display data
  - KeyboardOnly – user can input a password or PIN, but no display
  - NoInputNoOutput – device has no means for user input, and has no display
  - KeyboardDisplay – device has a means for display as well as for input

- Based on the combination of the capabilities of the two devices, one of two methods of pairing will be used:
  - Passkey entry – one device will display a randomly generator passkey, while the other will require the user to input the passkey. This allows for an authenticated link (MITM protection)
  - "Just Works" – the pairing process completes without requiring a passkey to be entered. The link will <u>not</u> be authenticated, but is encrypted

- If either one of the two devices does not require authentication, then Just Works will be used by default, allowing the user to skip passkey entry

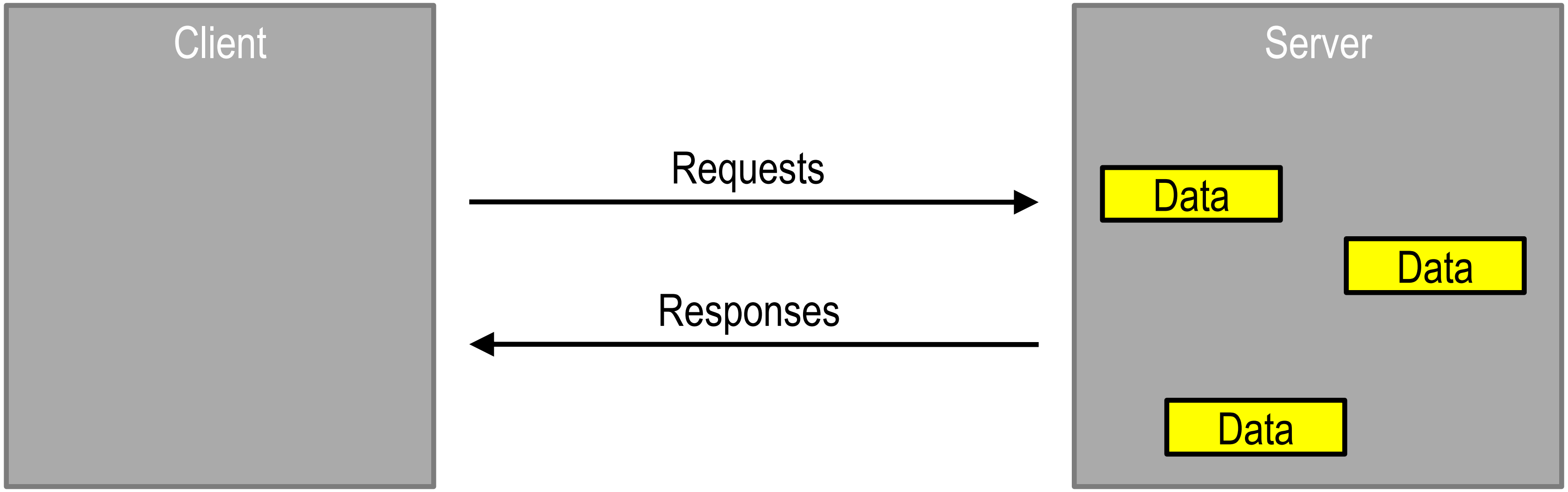
Bluetooth Low Energy Protocol Stack: Attribute Protocol (ATT) Overview

- An attribute is a discrete value that has associated with it the following three properties:
  1. A handle (address)
  2. A type
  3. A set of permissions
- ATT defines the over-the-air protocol for reading, writing, and discovering attributes
- Allows for different permissions to be assigned to attributes, including whether they are readable or writeable, and whether additional security is required for access



Attribute Protocol (ATT): Client / Server Architecture

- Servers have data, Clients want to use this data
- Servers expose data using attributes
- The Client / Server role of a device is independent of the GAP central / peripheral role or the link layer master / slave role
- It is possible for a device to act as both a client and server simultaneously, though the attributes on one device have no effect on the attributes on the other device



BLE Attribute Protocol (ATT): Attribute Table Example

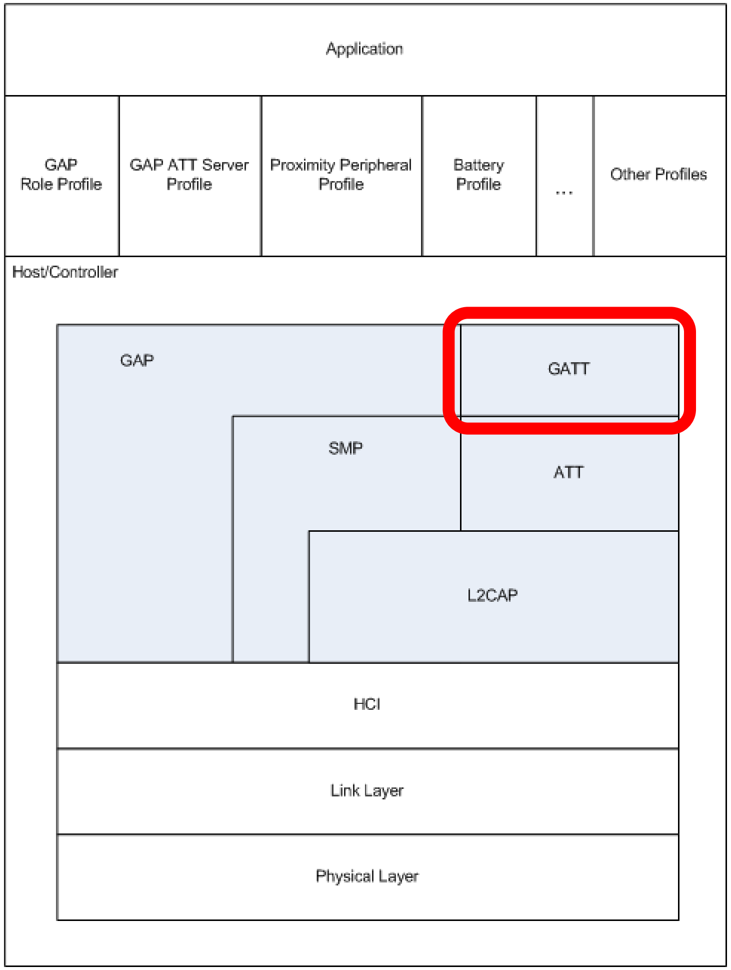
句柄——属性在列表中的位置
类型——
权限——读写等

- Handle – The address of the attribute in the table
- Type – Tells what the data represents; can be a UUID (universal unique identifier) assigned by the Bluetooth SIG, or a custom type
- Permissions – Enforces if and how the attribute client can access the attribute’s value

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)

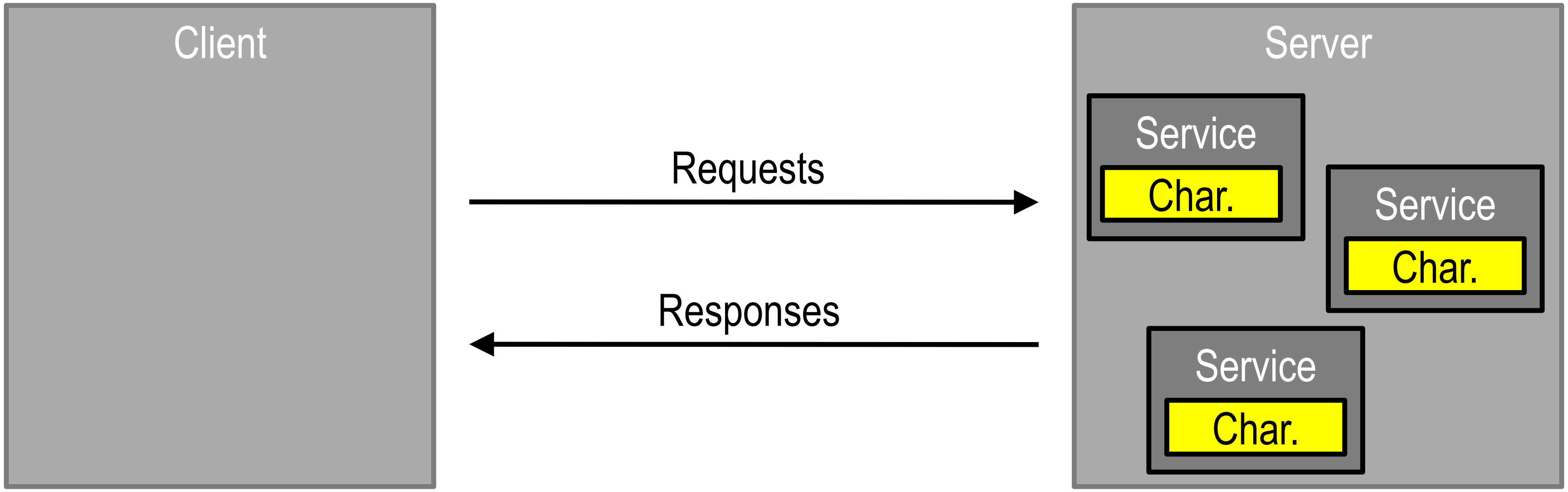
Bluetooth Low Energy Protocol Stack: Generic Attribute Profile (GATT) Overview

- Designed for use by the application or a profile, so that an attribute client can communicate with attribute server
- GATT defines:
  - Procedures for using the attribute protocol (ATT) to discover, read, write, and obtain indications of these attributes
  - The grouping and relationship of characteristics within a service or profile
  - Procedures for configuring the broadcast of attributes



Generic Attribute Profile (GATT): Client / Server Architecture

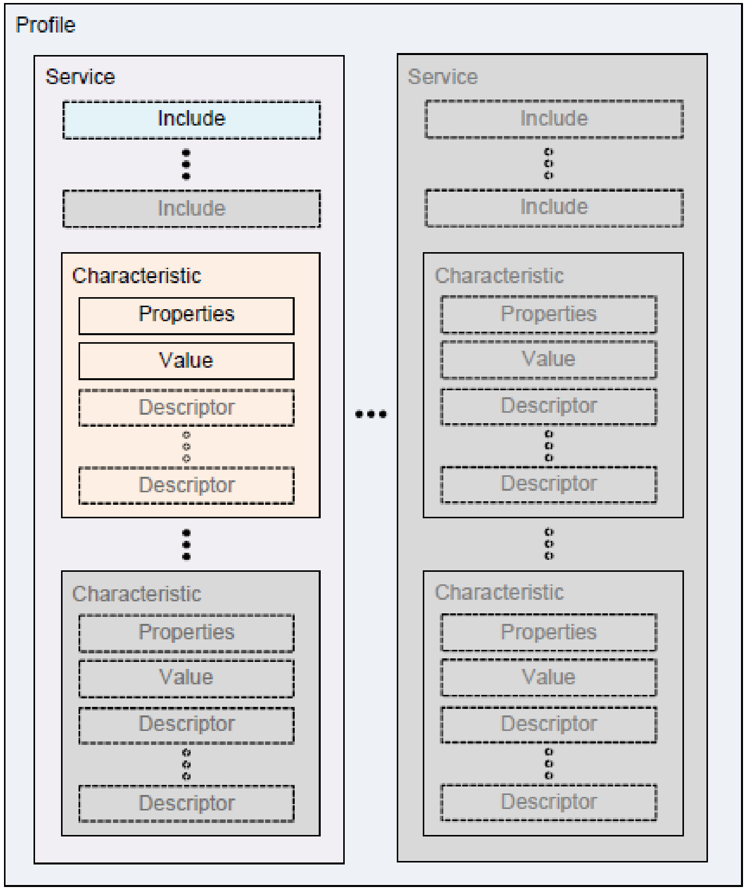
- GATT specifies the structure in which profile data is exchanged
- Same client server architecture as Attribute Protocol, except that data is encapsulated in “Services” and data is exposed in “Characteristic”



BLE Generic Attribute Profile (GATT): Profile Hierarchy

PROFILE由1个或多个services组成,
一个service可以包含多个属性（特征值）。
特征值必须强制拥有Declaration属性，且位于特征值之前。
其他的类似configuration或的description都是可选的属性

- A profile is composed of one or more “services” necessary to fulfill a use-case
- A service may contain certain attributes called “characteristic values”, which are values used by a service (example: in a temperature sensor, the attribute containing the temperature itself is the characteristic value)
- A characteristic value must have a mandatory “characteristic declaration” attribute immediately before the value, containing the properties of the characteristic
- Characteristics may also contain optional “descriptor” attributes, with fields such as a configuration or a description



BLE Generic Attribute Profile (GATT): Service Example

- Start of service at handle 39 is indicated by type 0x2800, which is defined by the Bluetooth SIG *Assigned Numbers* Document as the UUID for a GATT Service
- The value of the attribute at handle 39 is 0xFFE0, which is a used for the Simple Keys custom profile (this is just an example; the value of 0xFFE0 might already be used by the Bluetooth SIG)
- The service includes all subsequent attributes up until right before the next service in the table (or until the end of the table if there are no more services). In this example, the last attribute in the Simple Keys service is at handle 42, since a new service starts at 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 Generic Attribute Profile (GATT): Characteristic Declaration

- Handle 40 is a characterisitic declaration, as indicated by type 0x2803 (defined by the Bluetooth SIG *Assigned Numbers* Document as the UUID for a GATT Characteristic Declaration)
- The characteristic declaration attribute value is 5 bytes long:
  - First two bytes (0xFFE1) – indicates the type of the characteristic value attribute
  - Next two bytes (0x0029) – indicates the handle of the characteristic value attribute
  - Next byte (0x10) – indicates the permissions of the characterisitic value attribute

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 Generic Attribute Profile (GATT): Characteristic Configuration

- In addition to the characteristic declaration and the characteristic value itself, a characteristic optionally may have descriptors, which contain more information or configuration data related to the characteristic value
- In this example, handle 42 contains a GATT Client Characteristic Configuration, as indicated by type 0x2902 (defined by the Bluetooth SIG *Assigned Numbers* Document as the UUID for a GATT Client Characteristic Configuration)
- The configuration value has write permissions, meaning that the GATT Client can change the value
- By changing the value from 0x0000 (notifications off) to 0x0001 (notifications on), the GATT server will begin sending notifications of the characteristic value to the GATT client, as determined by the profile

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 Generic Attribute Profile (GATT): Characteristics Additional Information

- Other optional characteristic descriptors include fields for a characteristic value description, configuration for broadcast of the characteristic value in advertisements, and information on the units and format of the characteristic value
- The characteristic includes the declaration and all subsequent attributes up until right before the next characteristic declaration or service in the table (or until the end of the table if there are no more services).
- In this example, the characteristic consists of handles 40 through 42, since the declaration is at handle 40 and a new service starts at handle 43

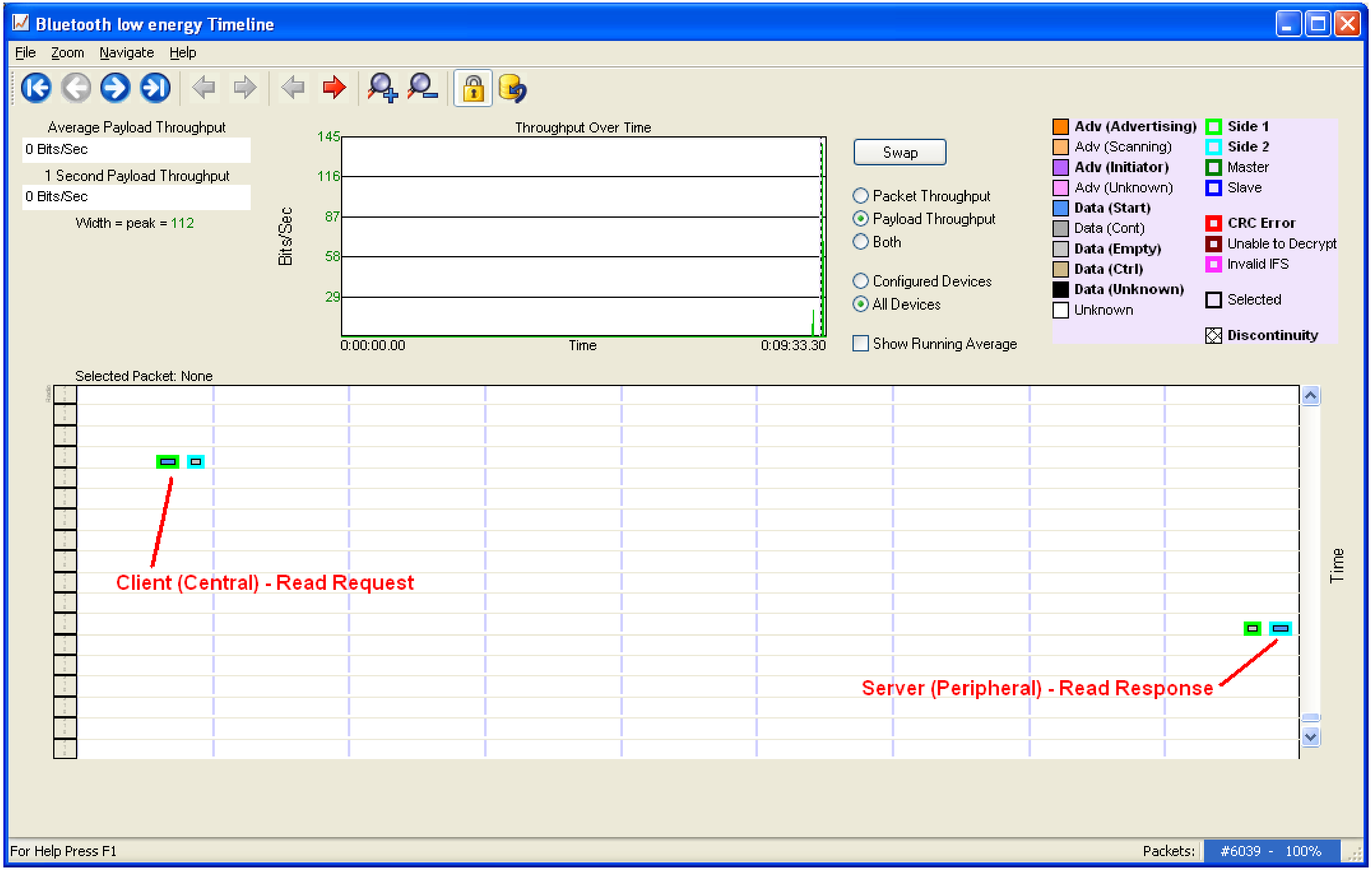
Handle	Type	Permissions	Value
39	0x2800 (GATT Primary Service UUID)	Read	E0:FF (2 bytes) (0xFFE0 = Simple Keys Service custom UUID)
40	0x2803 (GATT Characteristic Declaration UUID)	Read	10:29:00:E1:FF (5 bytes) (0xFFE1 = Simple Keys Value custom UUID) (0x0029 = handle 41) (0x10 = characteristic properties: 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 Generic Attribute Profile (GATT): Client Commands

- When two BLE devices are in the connected state, the GATT client device can perform several different sub-procedures to communicate with the GATT server device:
  - Discover Characteristic by UUID – search the GATT server for all attributes with type that matches the specified UUID
  - Read Characteristic Value – read the value of the characteristic at the specified handle
  - Write Characteristic Value – write a new value to the characteristic at the specified handle

- A GATT server device, when configured to do so, can send out messages to the GATT client device without being prompted:
  - Notification – The value a characteristic is sent from the server to the client without receiving a read request, and does not need to be acknowledged
  - Indication – The value a characteristic is sent from the server to the client without receiving a read request, but must be acknowledged before any further data can be sent

Demonstration: ATT / GATT



Average Payload Throughput
0 Bits/Sec

1 Second Payload Throughput
0 Bits/Sec

Width = peak = 112

○ Packet Throughput
● Payload Throughput
○ Both

○ Configured Devices
● All Devices

□ Show Running Average

Selected Packet: None

Client (Central) - Read Request

Server (Peripheral) - Read Response

Packets: #6039 - 100%

Agenda

- **Introduction**
- **Bluetooth Low Energy Protocol Stack (2.5 hours)**
  - Stack Architecture / Overview
  - Link Layer – Basics of BLE communication
  - Generic Access Profile (GAP) – Roles, Device Discovery, Connections, Security
  - Attribute Protocol (ATT) – Attribute Table, Reading and Writing Data
  - Generic Attribute Profile (GATT) – Profiles, Services, Characteristics
- **BLE Industry and Technology Update**
- **CC2540 BLE Software (2.5 hours)**
  - CC2540 Hardware Overview
  - CC2540 BLE Software Architecture and Structure
  - SimpleBLEPeripheral Project – Framework for Custom Applications
  - GAP Role Profiles and Bond Manager
  - GATT Profiles and Services
  - CC2540DK-MINI Kit Overview
- **Hands-on Labs (3 hours)**

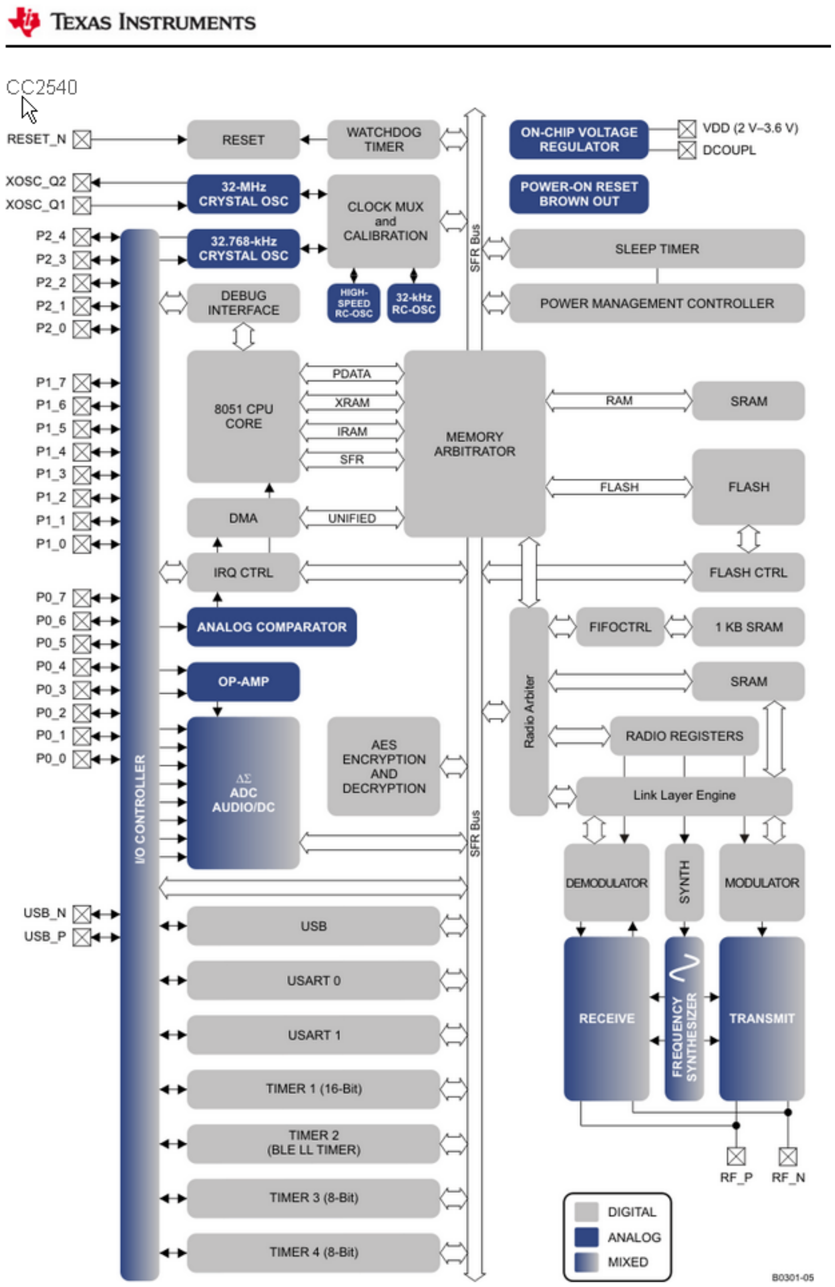
BLE Industry and Technology Update

Agenda

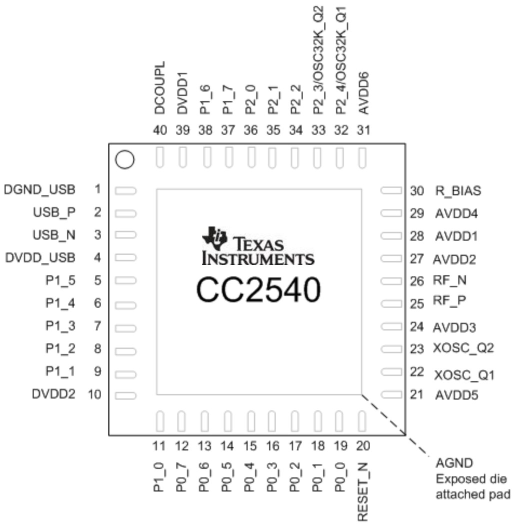
- **Introduction**
- **Bluetooth Low Energy Protocol Stack (2.5 hours)**
  - Stack Architecture / Overview
  - Link Layer – Basics of BLE communication
  - Generic Access Profile (GAP) – Roles, Device Discovery, Connections, Security
  - Attribute Protocol (ATT) – Attribute Table, Reading and Writing Data
  - Generic Attribute Profile (GATT) – Profiles, Services, Characteristics
- **BLE Industry and Technology Update**
- **CC2540 BLE Software (2.5 hours)**
  - CC2540 Hardware Overview
  - CC2540 BLE Software Architecture and Structure
  - SimpleBLEPeripheral Project – Framework for Custom Applications
  - GAP Role Profiles and Bond Manager
  - GATT Profiles and Services
  - CC2540DK-MINI Kit Overview
- **Hands-on Labs (3 hours)**

TI CC2540 Solution

- System on chip
- RF Transceiver + 8051MCU
- Master or Slave
- Programmable flash
- 8KB RAM
- Full SW stack – royalty free
- First to RTM & meet full qualification

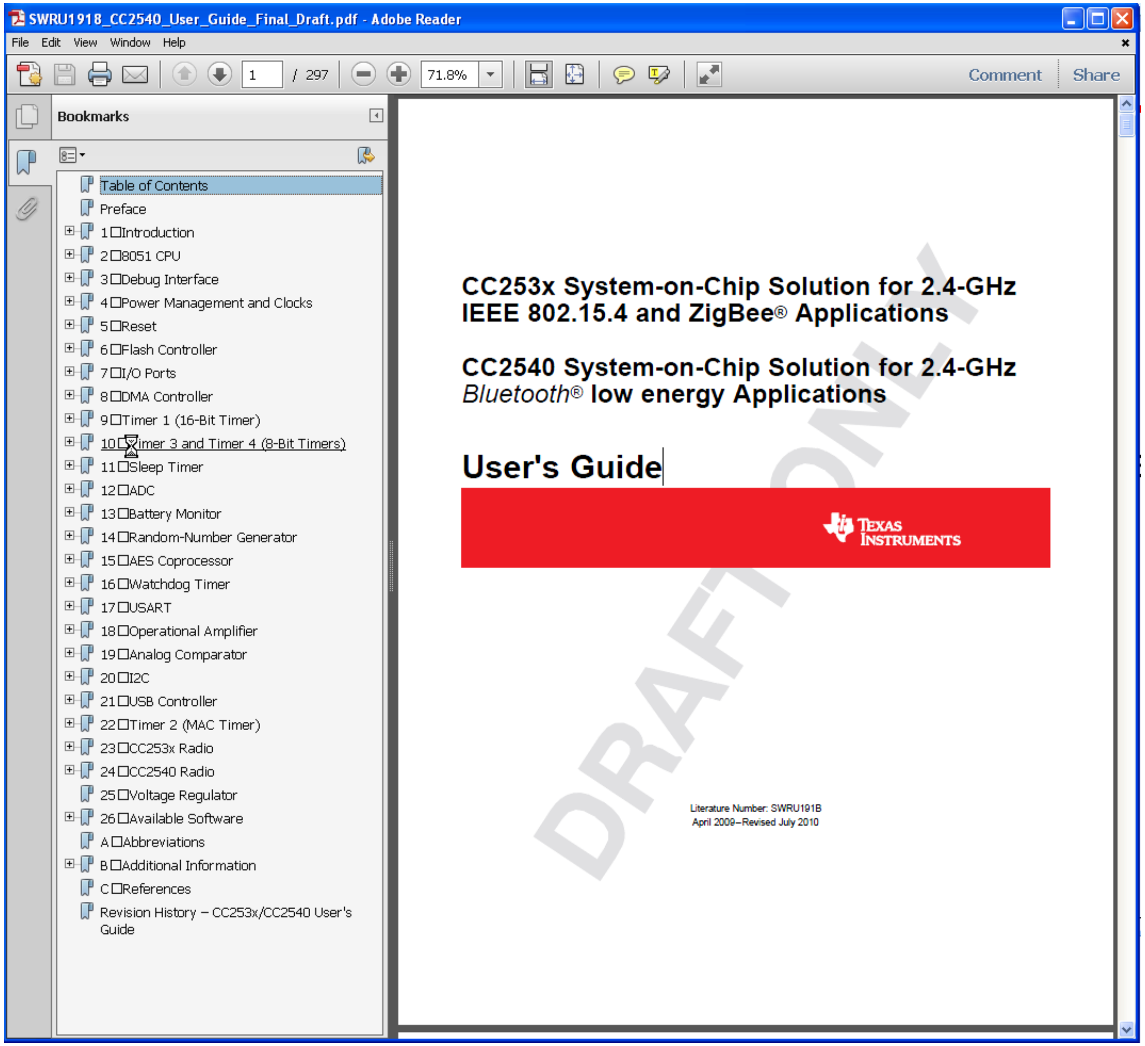


CC2540 System-on-a-chip (SoC)



- **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 User’s Guide



SWRU1918_CC2540_User_Guide_Final_Draft.pdf - Adobe Reader

CC253x System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee® Applications

CC2540 System-on-Chip Solution for 2.4-GHz *Bluetooth®* low energy Applications

User's Guide

Literature Number: SWRU191B
April 2009–Revised July 2010

TI CC2540DK-MINI Hardware Kit

Debugger

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

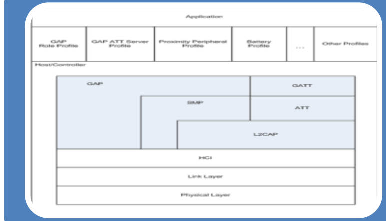
CC2540 Keyfob

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

USB Dongle

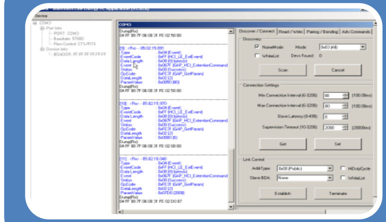
- Use Btool.exe to or custom app to send HCI commands.
- Usually acts as master (cell phone)

TI CC2540DK-MINI Software



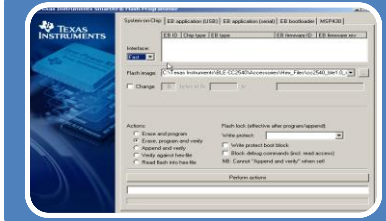
Stack Libraries

- Royalty free
- Full qualification
- Example Projects



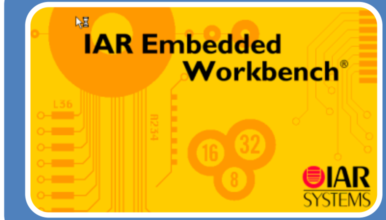
Btool Application

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



SmartRF Flash Programmer

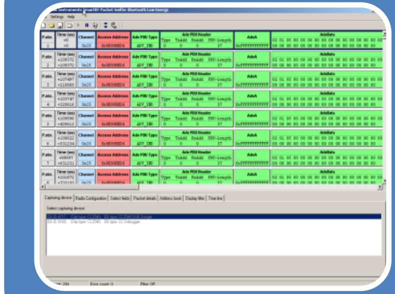
- Can flash CC2540
- Change address on device



IAR Compiler and IDE

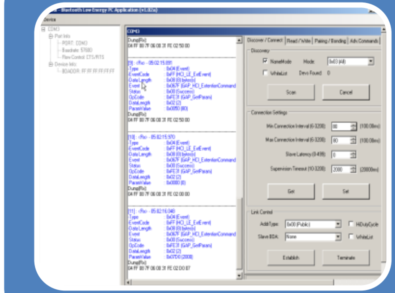
- Robust 8051 compiler with CC2540 support.
- 30 day free evaluation

TI CC2540DK-MINI Support



TI RF Sniffer

- Free
- Works with Mini Kit USB Dongle



```
Dump(Rx):
04 FF 08 7F 06 00 31 FE 02 50 00

[9] : <Rx> - 05:02:15.891
-Type           : 0x04 (Event)
-EventCode      : 0xFF (HCI_LE_ExtEvent)
-Data Length    : 0x08 (8) bytes(s)
 Event          : 0x067F (GAP_HCI_ExtentionCommand
 Status         : 0x00 (Success)
 OpCode         : 0xFE31 (GAP_GetParam)
 DataLength     : 0x02 (2)
 ParamValue     : 0x0050 (80)
Dump(Rx):
04 FF 08 7F 06 00 31 FE 02 50 00

[10] : <Rx> - 05:02:15.970
-Type           : 0x04 (Event)
-EventCode      : 0xFF (HCI_LE_ExtEvent)
-Data Length    : 0x08 (8) bytes(s)
 Event          : 0x067F (GAP_HCI_ExtentionCommand
 Status         : 0x00 (Success)
 OpCode         : 0xFE31 (GAP_GetParam)
 DataLength     : 0x02 (2)
 ParamValue     : 0x0000 (0)
Dump(Rx):
04 FF 08 7F 06 00 31 FE 02 00 00

[11] : <Rx> - 05:02:16.048
-Type           : 0x04 (Event)
-EventCode      : 0xFF (HCI_LE_ExtEvent)
-Data Length    : 0x08 (8) bytes(s)
 Event          : 0x067F (GAP_HCI_ExtentionCommand
 Status         : 0x00 (Success)
 OpCode         : 0xFE31 (GAP_GetParam)
 DataLength     : 0x02 (2)
 ParamValue     : 0x07D0 (2000)
Dump(Rx):
04 FF 08 7F 06 00 31 FE 02 D0 07
```

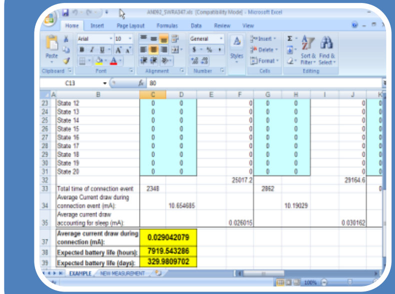
☑ NameMode Mode: 0x03 (All)
☐ WhiteList Devs Found: 0

Min Connection Interval (6-3200): 80 (100.00ms)
Max Connection Interval (6-3200): 80 (100.00ms)
Slave Latency (0-499): 0
Supervision Timeout (10-3200): 2000 (20000ms)

AddrType: 0x00 (Public) ☐ HiDutyCycle
Slave BDA: None ☐ WhiteList

Example Applications

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



	A	B	C	D	E	F	G	H	I	J	K
23		State 12	0	0		0	0	0		0	0
24		State 13	0	0		0	0	0		0	0
25		State 14	0	0		0	0	0		0	0
26		State 15	0	0		0	0	0		0	0
27		State 16	0	0		0	0	0		0	0
28		State 17	0	0		0	0	0		0	0
29		State 18	0	0		0	0	0		0	0
30		State 19	0	0		0	0	0		0	0
31		State 20	0	0		0	0	0		0	0
32						25017.2				29164.6	
33		Total time of connection event	2348				2862				0
34		Average Current draw during connection event (mA):		10.654685				10.19029			
35		Average current draw accounting for sleep (mA):				0.026015				0.030162	
37		**Average current draw during connection (mA):**	**0.029042079**								
38		**Expected battery life (hours):**	**7919.543286**								
39		**Expected battery life (days):**	**329.9809702**								

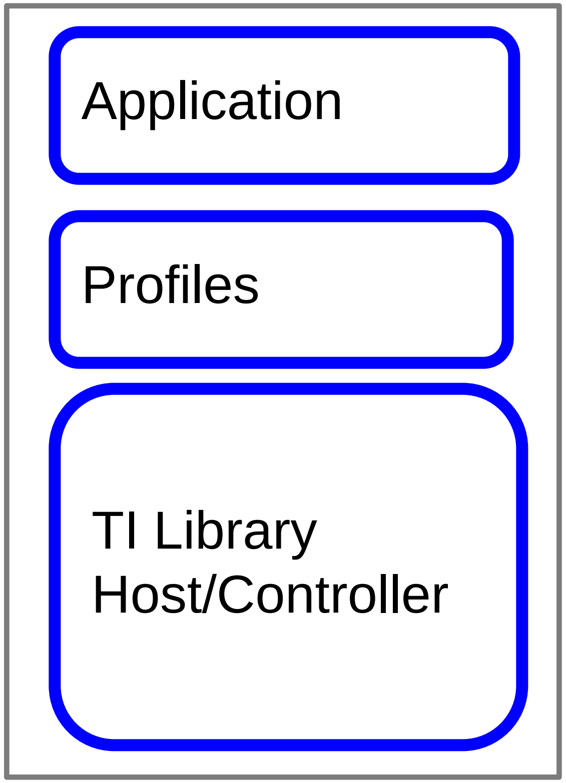
Power Calc Applications Note

- Excel sheet to help calculate battery life expectancy

Texas Instruments

CC2540DK-MINI Kit – KeyFob Hardware

- Powered by CR2032 coin cell
- Peripheral
- Peripheral/Broadcaster
- Reference Design
- DC/DC converter available



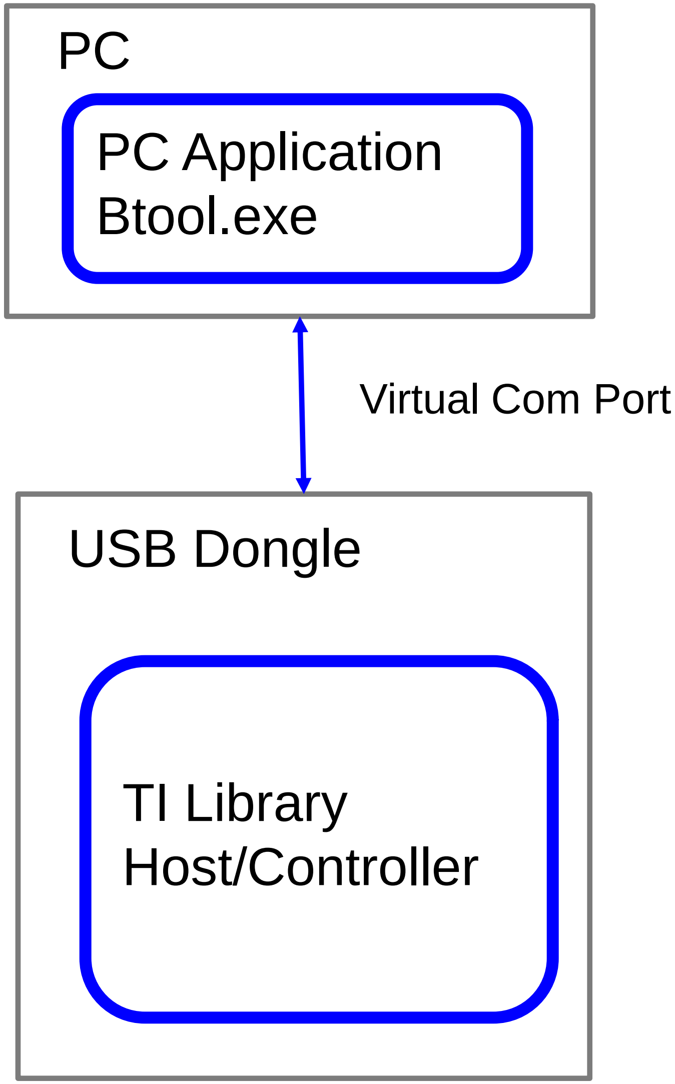
Application

Profiles

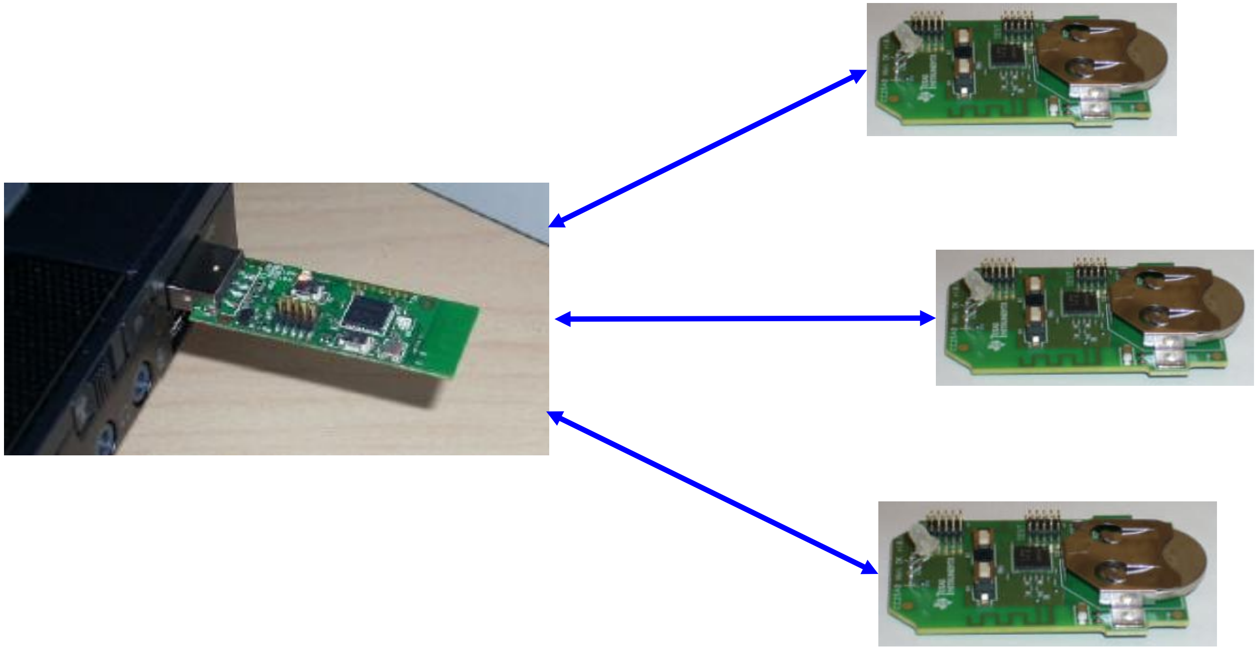
TI Library
Host/Controller

CC2540DK-MINI Kit – USB Dongle

- GAP Central Device
- Network processor



CC2540DK-MINI Kit – Network Example



IAR Embedded Workbench IDE: Overview

- All software development on the CC2540 is done using IAR Embedded Workbench for 8051 Integrated Development Environment (IDE)

- IAR Embedded Workbench for 8051 includes:
  - C Compiler
  - Assembler
  - Library Builder
  - Support for Hardware Debugger

- IAR Website: www.iar.com

CC2540 SDK: General Notes

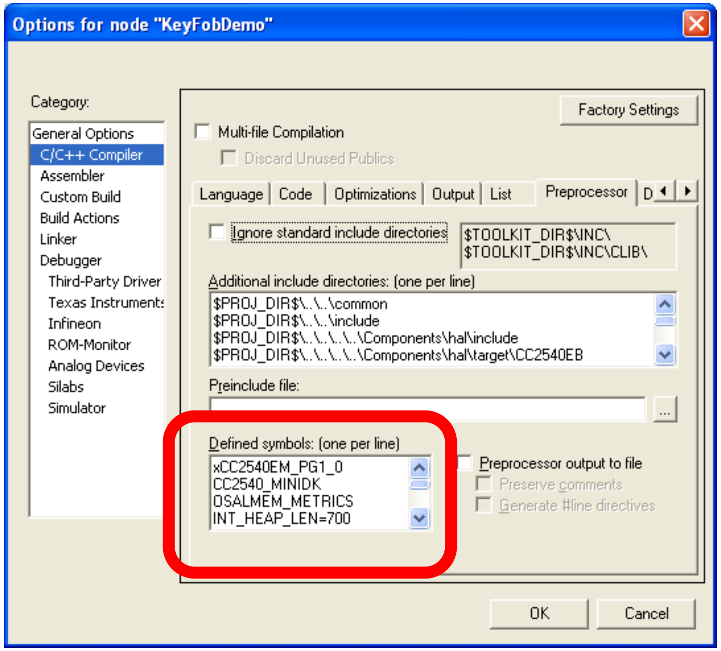
- An intermediate level of knowledge of the C programming language is required in order to modify or develop software for the CC2540
- The CC2540 software runs on an embedded 8051 microcontroller (MCU) with limited resources
  - Algorithms and application should be coded efficiently
  - C standard library should not be used
- Many of the modules in the CC2540 BLE software use the concepts of encapsulation and information hiding:
  - GetParameter and SetParameter functions to access data within the module
  - Callbacks – function pointers that must be registered with the module

CC2540 Embedded Software: Source Code and Project Notes

- Capital letters at the beginning of a function or variable indicate that it is public or global; lowercase letters indicate private or local

- Defined variable types:
  - uint8 – 8-bit unsigned integer
  - uint16 – 16-bit unsigned integer
  - uint32 – 32-bit unsigned integer
  - int8 – 8-bit signed integer
  - int16 – 16-bit signed integer
  - Int32 – 32-bit signed integer

- Build is dependent on having a correct set of preprocessor defined symbols, which can be found in the IAR project options menu



Options for node "KeyFobDemo"

☐ Multi-file Compilation
☐ Discard Unused Publics

☐ Ignore standard include directories

```
$TOOLKIT_DIR$\INC\
$TOOLKIT_DIR$\INC\CLIB\
```

Additional include directories: (one per line)

```
$PROJ_DIR$\..\..\common
$PROJ_DIR$\..\..\include
$PROJ_DIR$\..\..\..\..\Components\hal\include
$PROJ_DIR$\..\..\..\..\Components\hal\target\CC2540EB
```

Preinclude file:

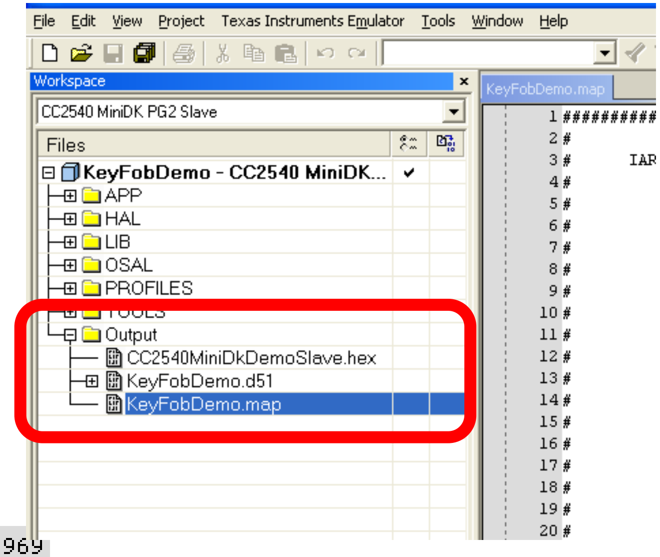
Defined symbols: (one per line)

```
xCC2540EM_PG1_0
CC2540_MINIDK
OSALMEM_METRICS
INT_HEAP_LEN=700
```

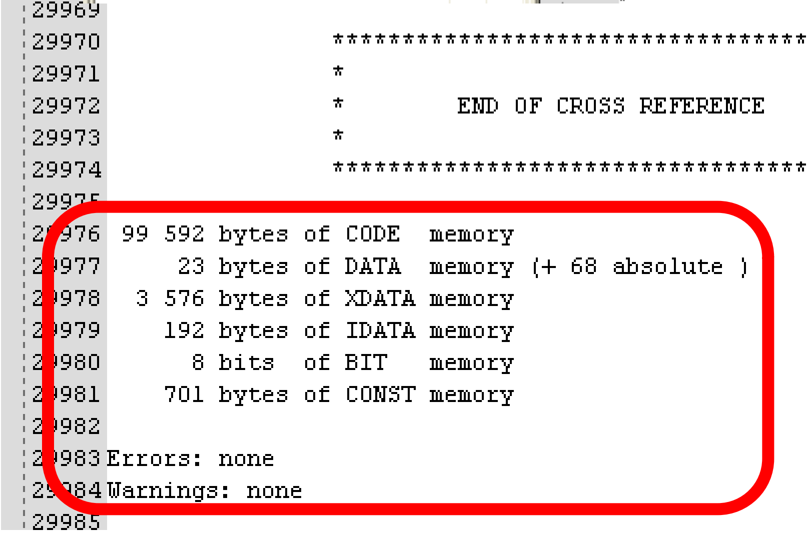
☐ Preprocessor output to file
☐ Preserve comments
☐ Generate #line directives

CC2540 Embedded Software: Map File

- After building a project, IAR generates a “map file” under the “Output” group



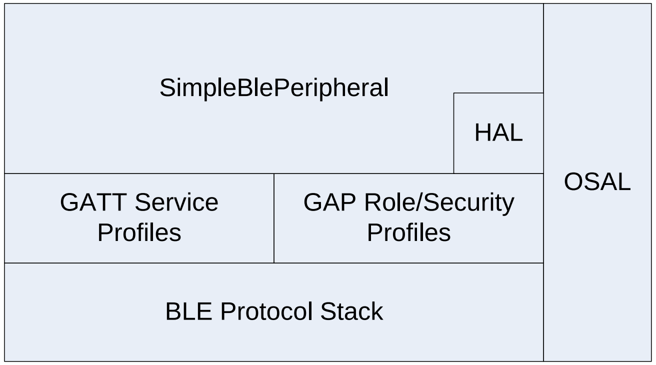
- The end of the map file contains a summary of the code memory and RAM used by the project



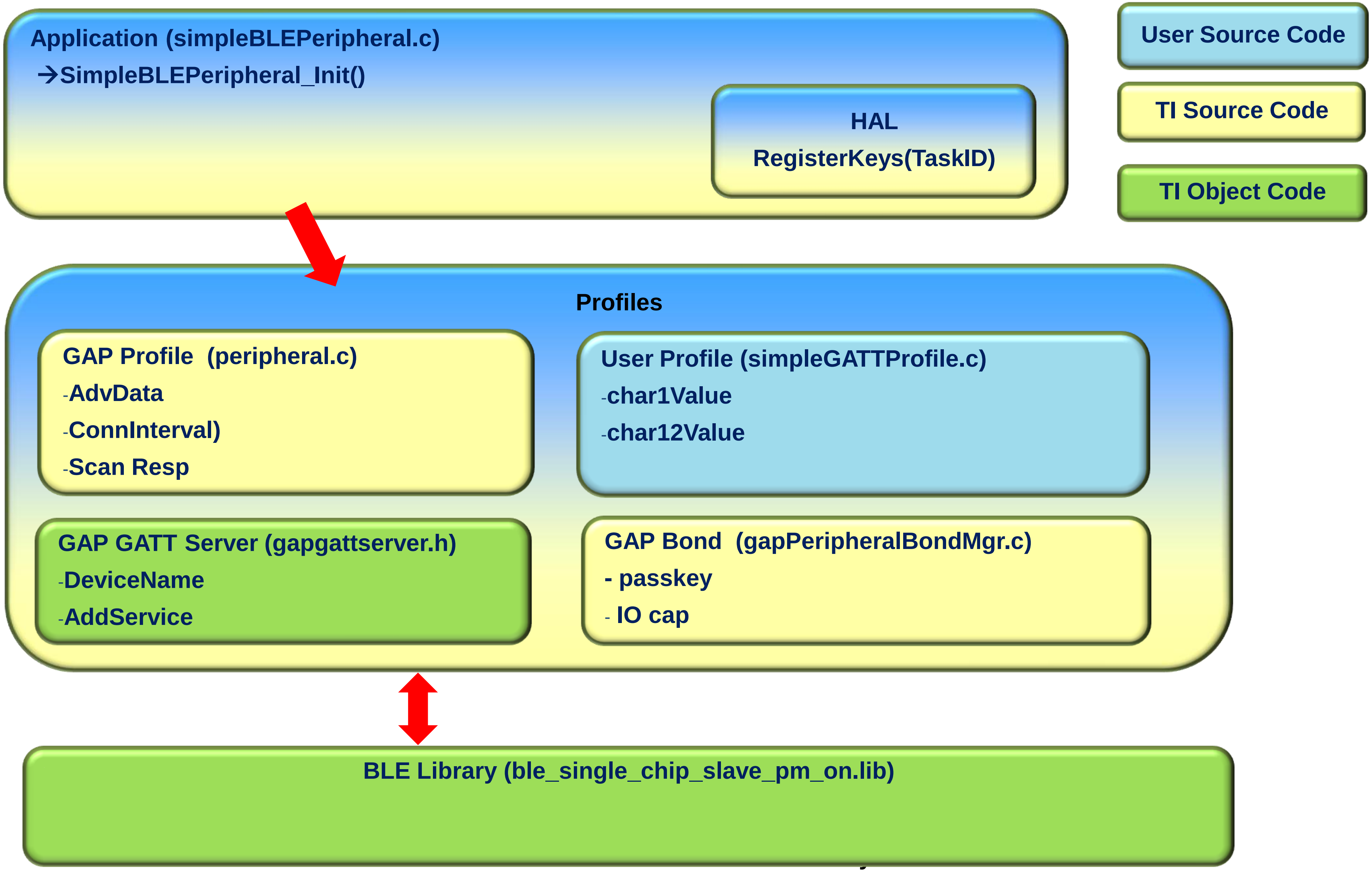
```
                    **********************************
                    *
                    *        END OF CROSS REFERENCE
                    *
                    **********************************

 99 592 bytes of CODE  memory
     23 bytes of DATA  memory (+ 68 absolute )
  3 576 bytes of XDATA memory
    192 bytes of IDATA memory
      8 bits  of BIT   memory
    701 bytes of CONST memory

Errors: none
Warnings: none
```

CC2540 Embedded Software Application Overview

- Five major parts of the application software:
  - Operating System Abstraction Layer (OSAL)
  - Hardware Abstraction Layer (HAL)
  - KeyFobDemo Application
  - BLE Protocol Stack
  - Profiles: GAP Role, GAP Security, and GATT Services



Application Startup



Application (simpleBLEPeripheral.c)

→SimpleBLEPeripheral_Init()

HAL

RegisterKeys(TaskID)

User Source Code

TI Source Code

TI Object Code

Profiles

GAP Profile (peripheral.c)

-AdvData

-ConnInterval)

-Scan Resp

User Profile (simpleGATTProfile.c)

-char1Value

-char12Value

GAP GATT Server (gapgattserver.h)

-DeviceName

-AddService

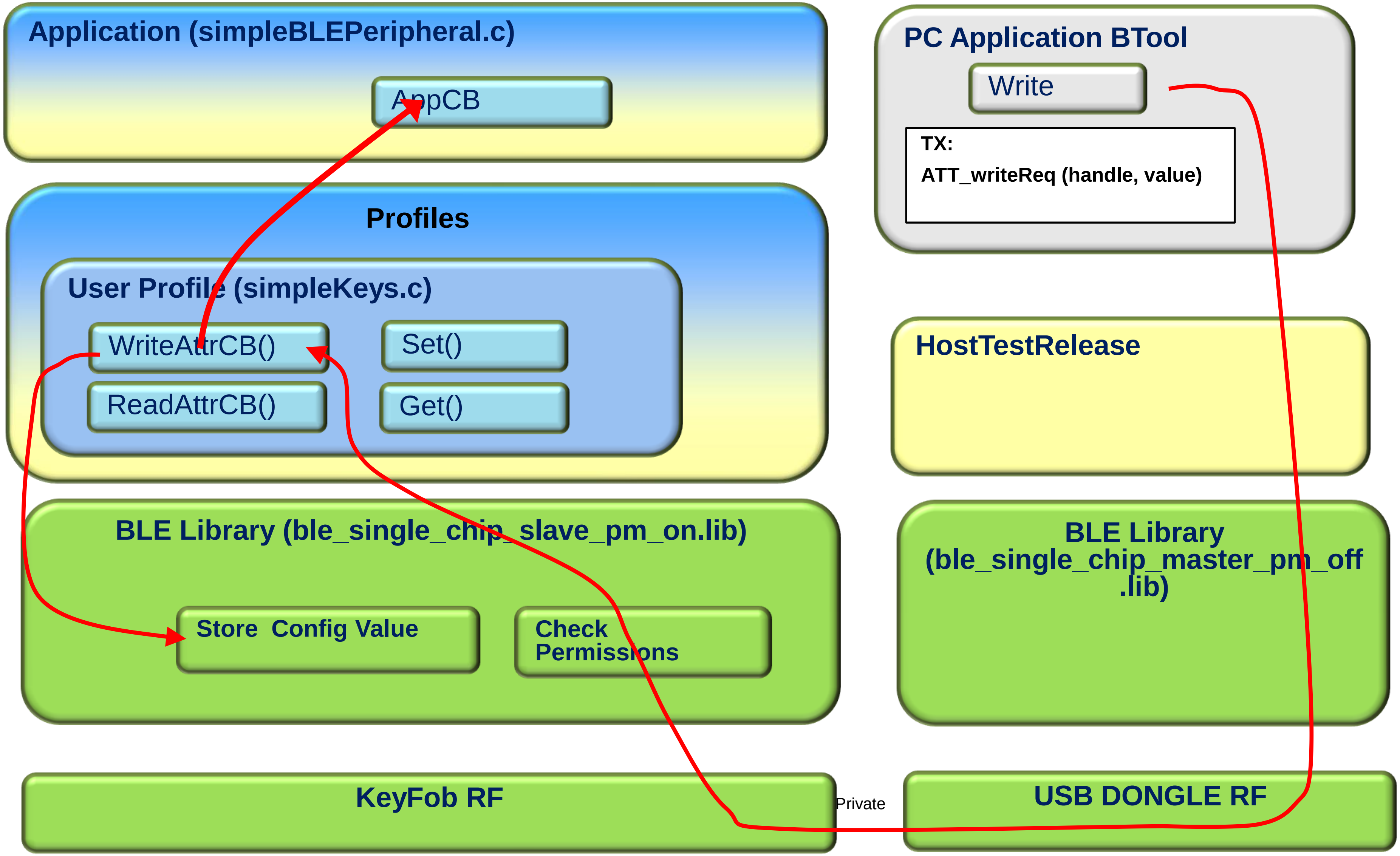
GAP Bond (gapPeripheralBondMgr.c)

- passkey

- IO cap

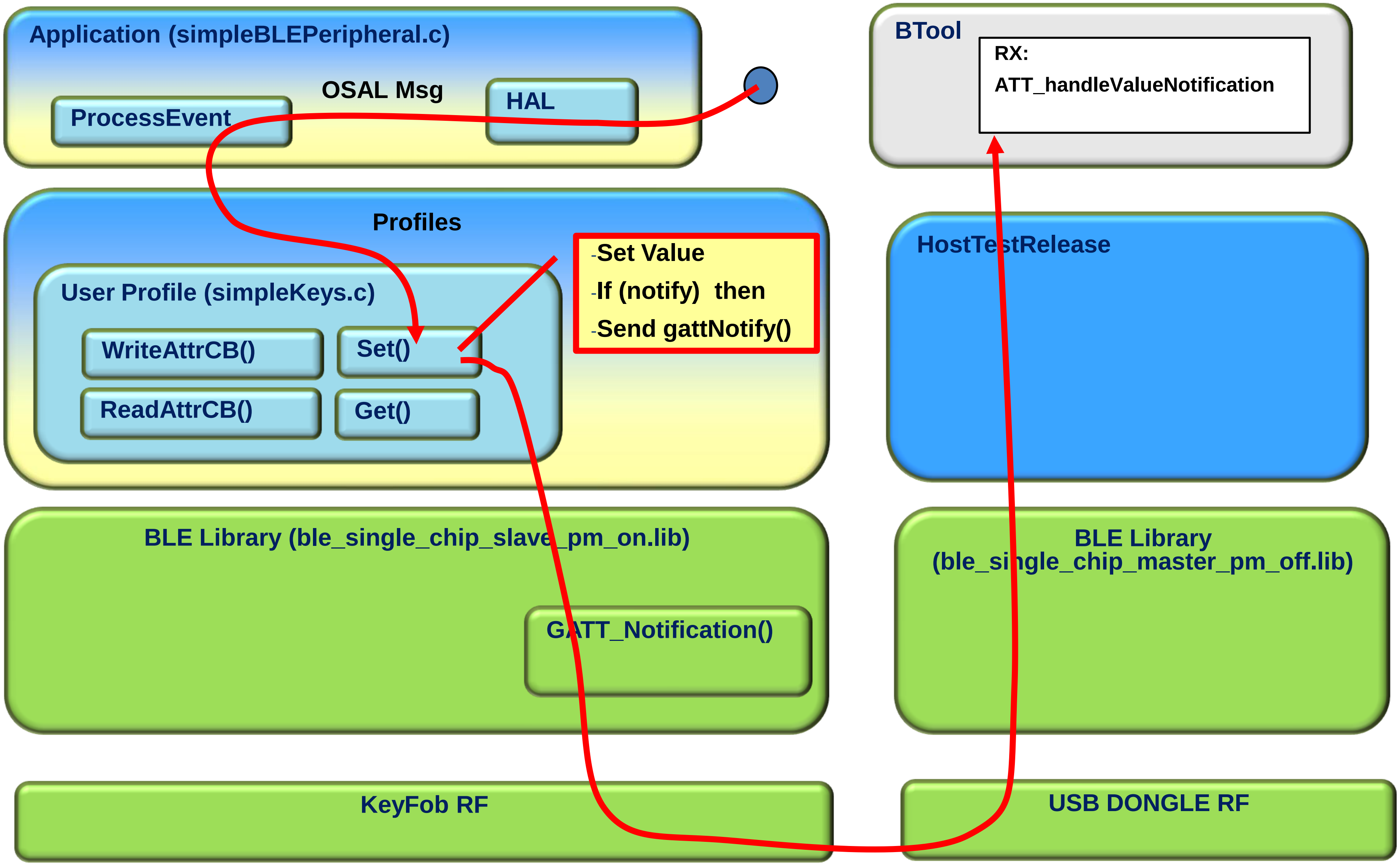
BLE Library (ble_single_chip_slave_pm_on.lib)

Application – Turn on Notifications



Application (simpleBLEPeripheral.c)
AppCB
Profiles
User Profile (simpleKeys.c)
WriteAttrCB()
Set()
ReadAttrCB()
Get()
BLE Library (ble_single_chip_slave_pm_on.lib)
Store Config Value
Check Permissions
KeyFob RF
Private
PC Application BTool
Write
TX:
ATT_writeReq (handle, value)
HostTestRelease
BLE Library (ble_single_chip_master_pm_off .lib)
USB DONGLE RF

Application – Keypress Notification



Application (simpleBLEPeripheral.c)

ProcessEvent

OSAL Msg

HAL

BTool

RX:

ATT_handleValueNotification

Profiles

User Profile (simpleKeys.c)

WriteAttrCB()

Set()

ReadAttrCB()

Get()

- Set Value
- If (notify) then
- Send gattNotify()

HostTestRelease

BLE Library (ble_single_chip_slave_pm_on.lib)

GATT_Notification()

BLE Library (ble_single_chip_master_pm_off.lib)

KeyFob RF

USB DONGLE RF

CC2540 Bluetooth Low Energy Software: Operating System Abstraction Layer (OSAL)

- The software architecture of the CC2540 is based around the Operating System Abstraction Layer (OSAL)
- The OSAL is not an actual operating system (OS) in the traditional sense, but rather a control loop that allows software to setup execution of events
- Each subsystem of the software runs as an OSAL task, and has a unique task identifier (ID)
- The lower the task ID, the higher the priority for the task
- The SimpleBLEPeripheral Project has 12 OSAL tasks (task ID in parenthesis):
  - Link Layer (0)
  - HAL (1)
  - HCI (2)
  - OSAL Callback Timer (3)
  - L2CAP (4)
  - GATT (5)
  - GAP (6)
  - SM (7)
  - Peripheral Role Profile (8)
  - GAP Bond Manager (9)
  - GATT Server (10)
  - SimpleBLEPeripheral(11)

Operating System Abstraction Layer (OSAL): Task Setup

- Each task is required to have two functions:
  - Initialization (example: SimpleBLEPeripheral_Init)
  - Event Handler (example: SimpleBLEPeripheral_ProcessEvent)
- Every application that uses the OSAL must define a function called “osalInitTasks” (void parameters and void return)
- This function calls each task’s intialization function, and sets up it’s task ID
- Every application must also create a global variable called “tasksArr”, which is array consisting of one pointer to each task’s event handler function
- The order of the elements in the array must be exactly the same as the order of the task IDs
- Application must also create a global variable called “tasksEvents”, which is an array consisting of one uint16 value for each task
  - All elements of the tasksEvents array must be initialized to zero
  - Each element of the array represents the pending events for that task

Operating System Abstraction Layer (OSAL): Events

- An OSAL “event” is a scheduled process for a task to run
- Any OSAL task can define up to 15 events in addition to the mandatory SYS_EVENT_MSG event (0x8000), which is used for OSAL messaging
- Events can be set using one of two OSAL API functions:
  - osal_set_event – immediately schedules the event to occur
  - osal_start_timerEx – schedules the event to occur at a specific time in the future (set in milliseconds)
- An event set up using osal_start_timerEx can be cancelled by calling OSAL API function osal_stop_timerEx
- Each element in the tasksEvents array acts as a 16-bit mask for each task, with any set bit indicating that a specific event is scheduled for that task
- In example below, bit 8 of task 1 is set, indicating that the event with a defined mask value of 0x0100 should be processed

	MSB															LSB
Task 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Task 1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Task 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Task 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Operating System Abstraction Layer (OSAL): Main Loop

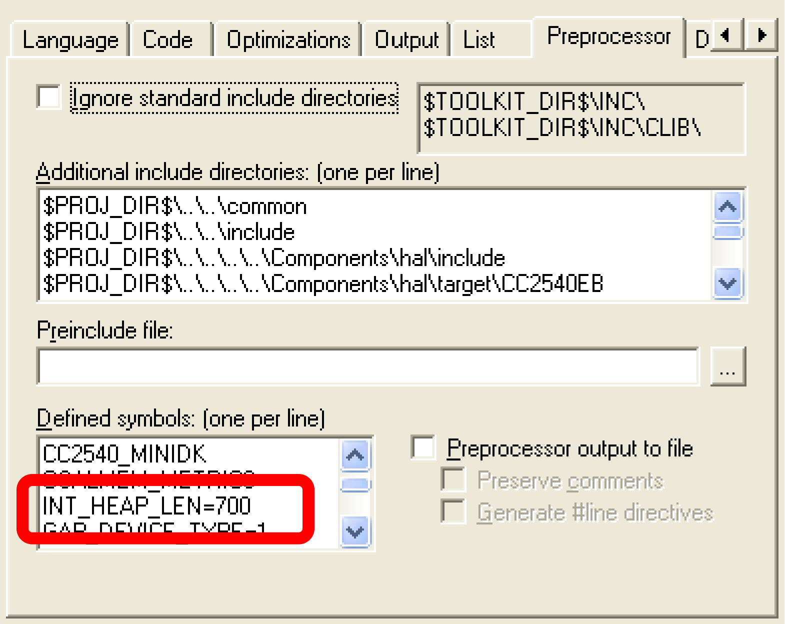
- The OSAL main loop is run when the function osal_start_system is called
- The loop checks each element of the tasksEvents array for a non-zero value (which would indicate that at least one event bit is set)
- The loop always processes a pending event with a lower task ID first
- When a non-zero value is found, OSAL will call the task's event handler function, using the pointer from tasksArr
- After the event is processed, it is up to the task to clear the event bit; if it doesn't get cleared the OSAL will keep calling the event handler function
- If every single element in the tasksEvents array has a zero value (meaning that none of the tasks have any events scheduled) the OSAL puts the processor into power savings mode, in which memory remains stored and timers continue running
- Processor will wake up when an interrupt occurs or when an OSAL timer schedules a task event

Operating System Abstraction Layer (OSAL): Message Management

- OSAL provides a system for different subsystems of the software to communicate with each other by sending or receive messages
- Messages can contain any type of data and can be any size
- Process to send a message:
  - Allocate memory using osal_msg_allocate
  - Copy data into allocated memory space, including a header indicating the type
  - Call osal_msg_send, indicating destination task for the message
- OSAL signals to receiving task that a message is arriving by setting the SYS_EVENT_MSG flag for that task
- The receving task's event handler function retrieves the data and calls it's local message processing function (example: simpleBLEPeripheral_ProcessOSALMsg)
- The receiving task must deallocate the memory using the function osal_msg_deallocate

Operating System Abstraction Layer (OSAL): Memory Management

- OSAL APIs for memory allocation and deallocation:
  - osal_mem_alloc
  - osal_mem_free
- Heap size set with preprocessor defined symbol INT_HEAP_LEN
- If heap size is set too high, CC2540 may run out of memory
- Check map file to verify that memory has not exceeded limits (8kB)



□ Ignore standard include directories

```
$TOOLKIT_DIR$\INC\
$TOOLKIT_DIR$\INC\CLIB\
```

Additional include directories: (one per line)

```
$PROJ_DIR$\..\..\common
$PROJ_DIR$\..\..\include
$PROJ_DIR$\..\..\..\..\Components\hal\include
$PROJ_DIR$\..\..\..\..\Components\hal\target\CC2540EB
```

Preinclude file:

Defined symbols: (one per line)

```
CC2540_MINIDK
OSALMEM_METRICS
INT_HEAP_LEN=700
GAP_DEVICE_TYPE=1
```

□ Preprocessor output to file

□ Preserve comments

□ Generate #line directives

Operating System Abstraction Layer (OSAL): Files and Key API's

- Key Files:
  - osal.c – API's for OSAL
  - osal.h – OSAL API declarations
- Key API's:
  - osal_init_system – initializes OSAL; must be called in main
  - osal_start_system – starts the OSAL main loop
  - osal_set_event – sets an OSAL event for a task
  - osal_start_timerEx – sets an OSAL event for a task at a scheduled moment in time
  - osal_stop_timerEx – cancels an existing OSAL event that was scheduled using osal_start_timerEx
  - osal_msg_allocate – dynamically allocates memory for an OSAL message
  - osal_msg_send – sends an OSAL message to a specific task
  - osal_msg_deallocate – deallocates an OSAL message (call this from receiving task)
  - osal_mem_alloc – dynamically allocates memory
  - osal_mem_free – free previously allocated memory
- The following OSAL function must be defined by the application:
  - OsalInitTasks – set up task ID's for each task used by OSAL
- Additional information on the OSAL can be found in the OSAL API guide:

C:\Texas Instruments\BLE-CC2540\Documents\osal\OSAL API.pdf

CC2540 Bluetooth Low Energy Software: Hardware Abstraction Layer (HAL) Overview

- The Hardware Abstraction Layer (HAL) provides an application programming interface to hardware-related functions
  - ADC
  - UART
  - SPI
  - Flash
  - Timers
  - Keys
  - LCD Driver
- Additional details on HAL functions can be found in the HAL API Guide:

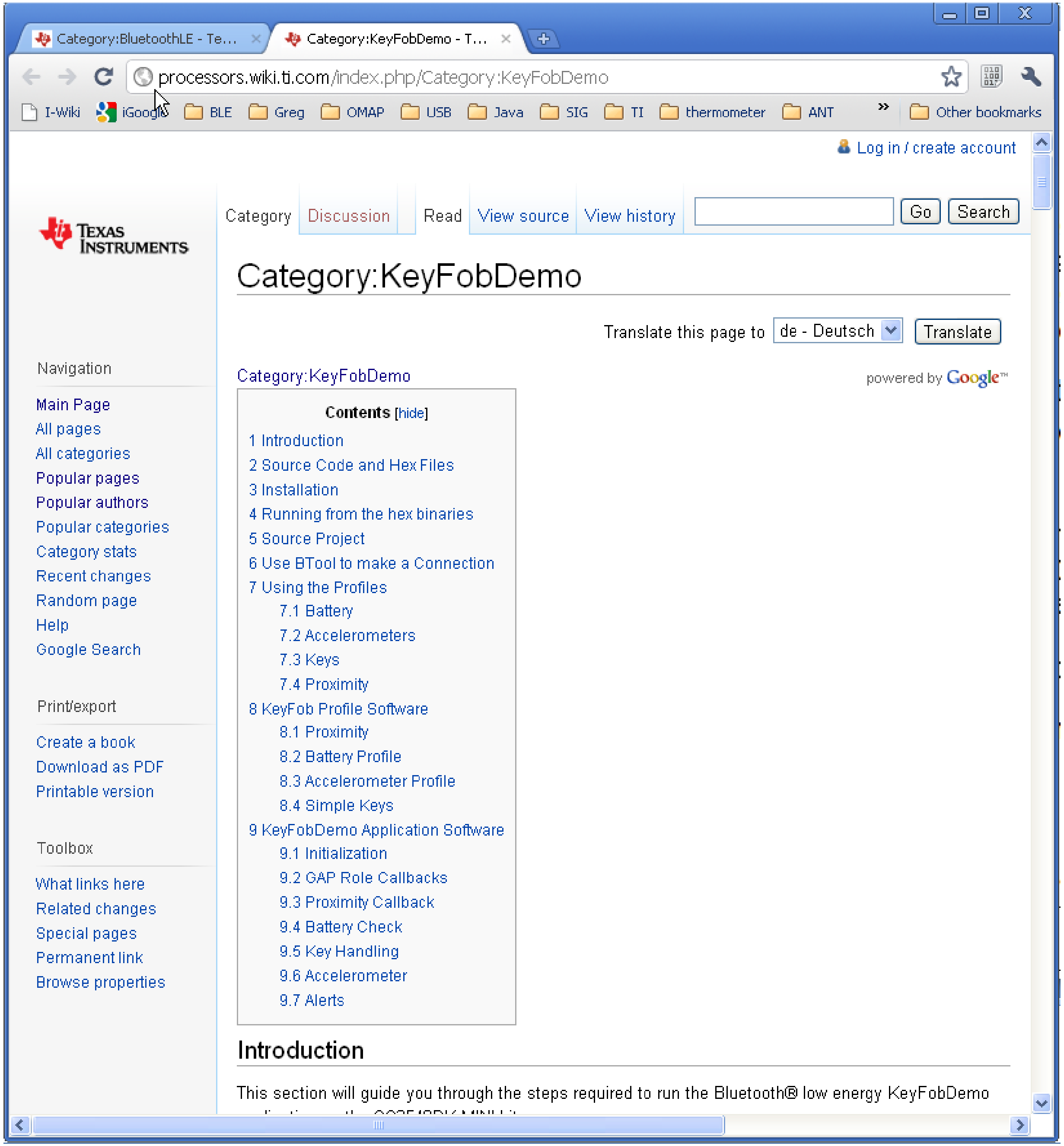
C:\Texas Instruments\BLE-CC2540\Documents\hal\HAL Driver API.pdf

CC2540 Bluetooth Low Energy Software: KeyFobDemo Application Overview

- The KeyFobDemo application provides a demonstration of a simple wireless Bluetooth Low Energy connection
  - Advertise and connect with master device
  - Key press notifications
  - Proximity alarm
  - Battery percentage measurement
  - Accelerometer data notification

KeyFobDemo

- Source
- Instructions



Category:KeyFobDemo

Category:KeyFobDemo

Contents [hide]

Introduction

This section will guide you through the steps required to run the Bluetooth® low energy KeyFobDemo

KeyFobDemo Application: Files

- The following application files are a part of the KeyFobDemo project:
  - KeyFob_Main.c – contains the main function, which performs HAL and OSAL intialization, and calls osal_start_system to start the main loop
  - OSAL_KeyFobDemo.c – defines the global tasksArr and tasksEvents arrays and the osalInitTasks function as required by OSAL
  - keyfobdemo.c – main application module, including the application task initialization and event handler functions
  - keyfobdemo.h – header file for application; defines the application OSAL task events
  - buzzer.c – controls the audio buzzer on the keyfob
  - cma3000d.c – controls and reads data from the accelerometer

KeyFobDemo Application: Startup

- The application starts with the main function in the file KeyFob_Main.c
- The KeyFobApp_Init function is called during task initialization
  - Sets Peripheral Role profile initial parameters
  - Sets GATT profile initial parameters
  - Initializes each GATT service
  - Initializes buzzer
  - Registers with HAL to receive OSAL message when key presses occur
  - Uses osal_start_timerEx to set a KEYFOB_START_DEVICE_EVT after a 500ms delay
- After the 500ms delay, application task event process handler function gets called due to KEYFOB_START_DEVICE_EVT flag getting set
  - GAPRole_StartDevice called to turn advertisements on
  - Application callbacks registered with proximity and accelerometer profiles
  - Timer set for future BATTERY_CHECK_EVT after 5 seconds
  - Proximity attribute values set in profile using ProxPeriph_GetParameter
  - KEYFOB_START_DEVICE Event flag cleared

KeyFobDemo Application: Key Handling

- Application registers with HAL during intialization by calling function RegisterForKeys, allowing HAL to know the application task ID
- Key presses are handled by HAL using interrupts
- When the state of one of the keys changes, an OSAL message with type KEY_CHANGE is sent to the application
- Application calls local function keyfobapp_HandleKeys
  - Checks which keys were pressed
  - If device is not connected, checks peripheral role profile to see whether device is advertising or not, and toggles advertisements on or off
  - Sets the state of the keys value in the Simple Keys profile using the function SimpleKeys_SetParameter
- If a proximity alert is active and the keyfob is beeping, pressing the left key will stop the alert
- If the device is in a connected state and notifications of the key presses have been enabled, the keyfob will send a GATT notification to the master device over the air (more information on this later)

KeyFobDemo Application: Proximity Alerts

- The demo application contains a proximity profile, which is based on a draft specification from the BT SIG
- Allows an alarm to be set based on the proximity of the keyfob to the master device, or when connection drops due to supervision timeout
- During application initialization, the application registers private function proximityAttrCB with proximity profile
- Proximity profile calls proximityAttrCB to notify application every time any of the proximity profile characteristics has been changed
- Proximity profile characteristics:
  - Link Loss Alert – when set, triggers a “low” (low-pitched buzzer sound) or “high” (high-pitched buzzer sound with blinking LED) alert if a supervision timeout occurs
  - Path Loss Alert – the master device sets this if the path loss drops below a certain level, triggered an immediate high or low alert to warn the user that the peripheral device is about to go out of range
  - Tx Power Level – the power level of the peripheral transmitter; used by the central device to calculate the path loss

KeyFobDemo Application: Proximity Alerts (continued)

- When proximityAttrCB gets called from profile, the application looks at the current state of the device and:
  - Stores the link loss alert setting, in order to trigger an alarm if the alert is enabled and the connection drops
  - Calls keyfobapp_PerformAlert if the path loss alert was enabled
  - Calls keyfobapp_StopAlert if the path loss alert was turned off
- The function keyfobapp_PerformAlert does the following:
  - Determines whether the device is in a “link-loss” state or a “path-loss” state (or neither)
  - If necessary, starts the buzzer and/or LED
- The function keyfobapp_StopAlert stops any active alert
- When alert is active, the buzzer stays on for 200ms, and off for 800ms using the OSAL timer to generate “beeps”
- After 10 beeps, the buzzer will stop
- If left keyfob button is pressed during an alert, the buzzer and LED will stop

KeyFobDemo Application: Battery Percentage Measurement

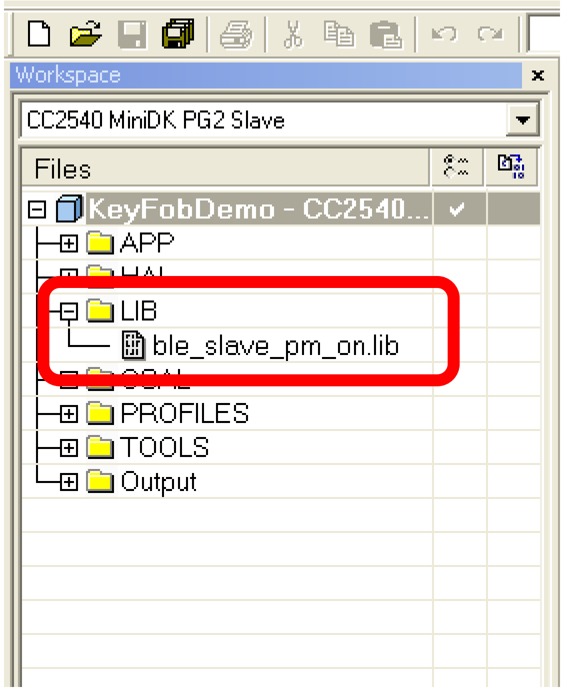
- Battery service allows remote device to read the battery percentage remaining on the keyfob
- Algorithm for measuring battery percentage uses HAL ADC API and is not optimized and is for demonstration purposes only
- Battery voltage is read by the ADC every 5 seconds, using the OSAL timer to set the application task event BATTERY_CHECK_EVT
- Every time the event occurs, a new OSAL timer is set to schedule the following BATTERY_CHECK_EVT in 5 seconds
- The function checkBattery is called, which does the following:
  - Sets the ADC reference voltage to the internal 1.25V regulator
  - Perform ADC conversion
  - Based on ADC reading, perform calculation of battery percentage (algorithm is explained in source code comments)
  - Check whether battery level is below 20%, and if so is in a “critical” state
  - Calls Battery_SetParameter to update the battery profile with current state and level values, allowing a GATT client device to read back the values

KeyFobDemo Application: Accelerometer

- The keyfob contains a 3-axis accelerometer
- During application initialization, the application registers private function accelEnablerChangeCB with accelerometer profile
- Profile calls the callback to notify application if remote device has written a TRUE value to the accelerometer enabler characteristic value, then the application begins to perform accelerometer reads
- If the remote device has written a FALSE value, then reads are stopped
- When enabled, application uses an OSAL timer to schedule the ACCEL_READ_EVT every 50ms
- Every time the event occurs, local function accelRead gets called:
  - Calls accReadAcc function (in file cma3000d.c) to get data from each axis
  - Calls Accel_SetParameter three times to set each of the three X, Y, and Z- axis data values in the accelerometer profile
- If the device is in a connected state and notifications of the accelerometer data have been enabled, the keyfob will send a GATT notification to the master device over the air (more information on this later)

CC2540 Bluetooth Low Energy Software: BLE Stack Overview

- The BLE protocol stack is based on the approved Bluetooth Core specification version 4.0 (June 30, 2010)
- Protocol stack provided as a single library file in KeyFobDemo application (three versions provided: one for each hardware platform)
- Application usually does not need to directly call protocol stack API’s
- Profiles provide a means for application to send and receive control messages and data with stack


Workspace

CC2540 MiniDK PG2 Slave

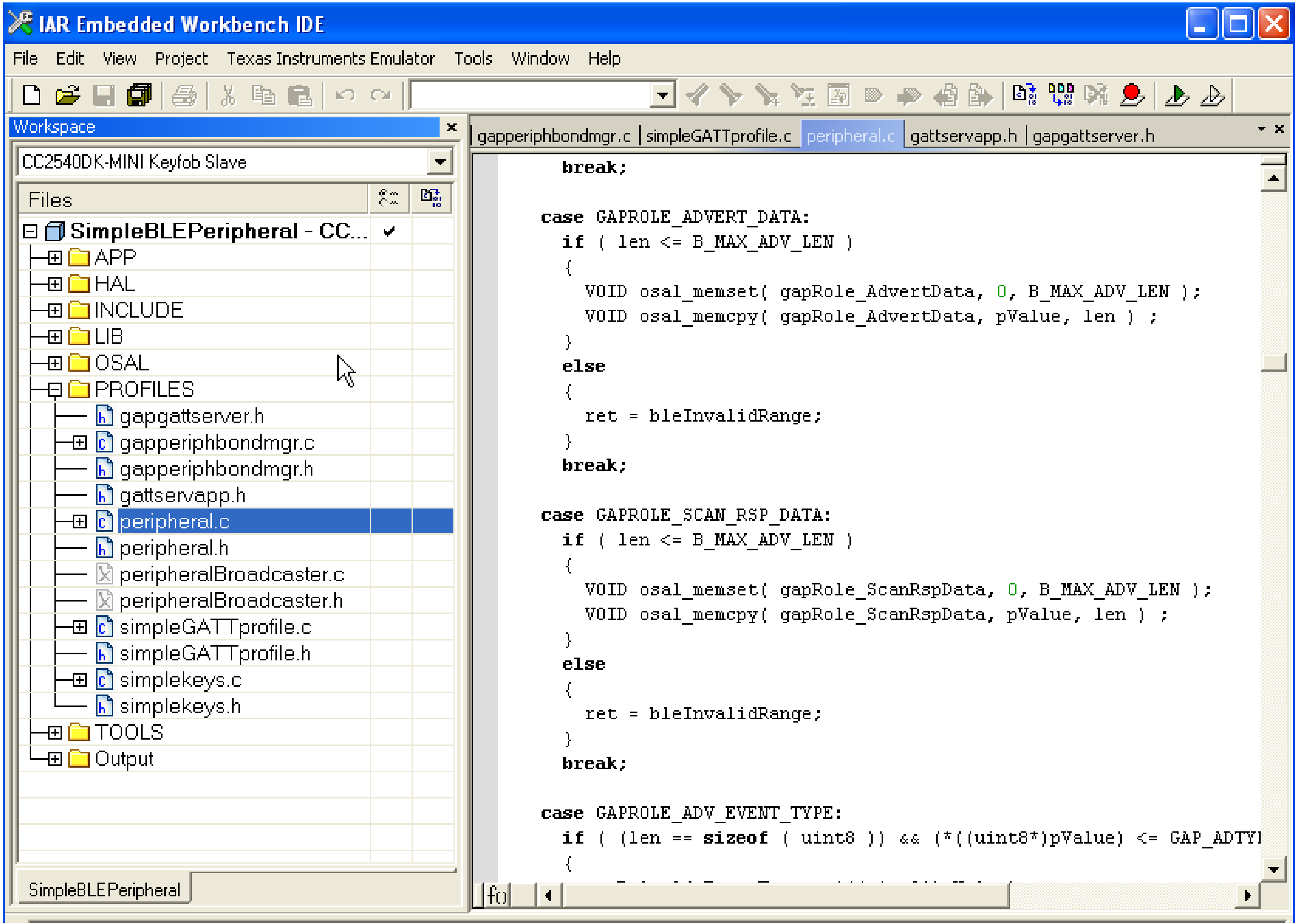
Files

- KeyFobDemo - CC2540...
  - APP
  - HAL
  - LIB
    - ble_slave_pm_on.lib
  - OSAL
  - PROFILES
  - TOOLS
  - Output

CC2540 Bluetooth Low Energy Software: Profiles Overview

- Profiles provide a layer of software between the application and the BLE protocol stack
- Allow developer to perform basic BLE functions without having in-depth knowledge of the stack
- Directly communicate with the top two layers of the BLE stack
  - **GAP Peripheral Role Profile** – Handles advertisements, scan requests, connections, and connection parameters
  - **GAP Peripheral Bond Manager** – Handles responses to pairing and bonding requests, and the storage and management of security keys
  - **GATT Profiles** – Maintain GATT attributes in table, processing of read and write requests, and notifications

GAP ROLE PROFILE



```
    break;

  case GAPROLE_ADVERT_DATA:
    if ( len <= B_MAX_ADV_LEN )
    {
      VOID osal_memset( gapRole_AdvertData, 0, B_MAX_ADV_LEN );
      VOID osal_memcpy( gapRole_AdvertData, pValue, len ) ;
    }
    else
    {
      ret = bleInvalidRange;
    }
    break;

  case GAPROLE_SCAN_RSP_DATA:
    if ( len <= B_MAX_ADV_LEN )
    {
      VOID osal_memset( gapRole_ScanRspData, 0, B_MAX_ADV_LEN );
      VOID osal_memcpy( gapRole_ScanRspData, pValue, len ) ;
    }
    else
    {
      ret = bleInvalidRange;
    }
    break;

  case GAPROLE_ADV_EVENT_TYPE:
    if ( (len == sizeof ( uint8 )) && (*((uint8*)pValue) <= GAP_ADTY
    {
```

GAP Peripheral Role Profile: Purpose

- Allows device to act as a GAP peripheral and perform the following:
  - Turn advertising on and off
  - Send connectable advertisements and accept connection requests
  - Request automatic updates of link-layer connection parameters to a central device:
    - Connection interval
    - Slave latency
    - Supervision timeout
  - Notify application of connection state changes

GAP Peripheral Role Profile: Public Functions

- Peripheral Role Profile is an OSAL task, and contains initialization and event processing functions called by OSAL:
  - GAPRole_Init
  - GAPRole_ProcessEvent
- Profile contains several parameters, accessed through:
  - GAPRole_SetParameter
  - GAPRole_GetParameter
- Initialization from application:
  - GAPRole_StartDevice
- Terminate a connection:
  - GAPRole_TerminateConnection

GAP Peripheral Role Profile: Initialization

- OSAL initializes Peripheral Role with call to GAPRole_Init
- Application registers two callback functions with Peripheral Role Profile by passing function pointers as parameter to GAPRole_StartDevice function:
  - peripheralStateNotificationCB – notifies application that the peripheral device has changed GAP states (for example, devices goes from advertising to being in a connection)
  - rssiAvailableCB – notifies application of the RSSI when it becomes available (set to NULL in KeyFobDemo application since it does not use RSSI information)
- When GAPRole_StartDevice is called:
  - Profile signals GAP to begin advertising (if enabled)
  - Profile registers itself with GAP as the task to receive GAP event messages (this allows profile to always know the connection status)
- In KeyFobDemo application, GAPRole_StartDevice is not called until 500ms delay (triggered by KEYFOB_START_DEVICE_EVT)

GAP Peripheral Role Profile: Key Parameters

- GAPROLE_ADVERT_DATA – Advertisement data string
- GAPROLE_SCAN_RSP_DATA – Scan response data string
- GAPROLE_ADVERT_ENABLED – a TRUE or FALSE value indicating if advertising is enabled
- GAPROLE_RSSI_READ_RATE – amount of time (in ms) of RSSI readings
- GAPROLE_PARAM_UPDATE_ENABLE – enabled automatic connection parameter update requests if master establishes a connection with unwanted parameters (TRUE or FALSE)
- GAPROLE_MIN_CONN_INTERVAL – the minimum connection interval for the device (in units of 1.25ms as per link layer specification)
- GAPROLE_MAX_CONN_INTERVAL – the maximum connection interval for the device (in units of 1.25ms as per link layer specification)
- GAPROLE_SLAVE_LATENCY – the connection slave latency setting
- GAPROLE_TIMEOUT_MULTIPLIER – the connection supervision timeout setting

GAP Peripheral Role Profile: Advertisement and Scan Response Data

- The GAPROLE_ADVERT_DATA and GAPROLE_SCAN_RSP_DATA parameters allow application to set the GAP data sent to a central or observer device while in the advertising state

- Data must conform to GAP specification for “AD types”:
  - The first byte contains the length of the data
  - The second byte contains a value indicating the AD type accoring to spec (ex. 0x09 = Local Name, 0x01 = Flags)

GAP Peripheral Role Profile: AD Types Used

- In KeyFobDemo application:
  - Advertisement Data String:

0x0A (length 10)	0x09 (name)	0x50 ‘P’	0x72 ‘r’	0x6F ‘o’	0x78 ‘x’	0x69 ‘i’	0x6D ‘m’	0x69 ‘i’	0x74 ‘t’	0x79 ‘y’

  - Scan Response Data String:

0x02 (2)	0x01 (flags)	0x02 (General Discoverable)

    - By setting “General Discoverable”, device will continuously advertise as long as advertisements are enabled
    - If set to “Limited Discoverable” (0x01), when advertisements are enabled the device will advertise for a limited time, stop for 10 seconds, and repeat

GAP Peripheral Role Profile: After Link Establishment

- Once a connection is established, GAP sends an OSAL message of type GAP_EST_LINK_REQ_EVENT to GAP application (peripheral role profile)
- RSSI read timer starts
- Profile calls the callback function peripheralStateNotificationCB to notify application that GAP state has changed
- Profile checks the connection interval and slave latency setting for the connection, and (if enabled) will send an automatic update request if:
  - Interval falls outside the range set by GAPROLE_MIN_CONN_INTERVAL and GAPROLE_MAX_CONN_INTERVAL parameters
  - OR latency setting does not equal GAPROLE_SLAVE_LATENCY parameter value
  - OR supervision timeout settings does not equal GAPROLE_TIMEOUT_MULTIPLIER parameter value
- Update parameter request sent with connection parameter values in profile
- Profile uses osal_start_timerEx to set the UPDATE_PARAMS_TIMEOUT_EVT OSAL event for itself after a set time (calculated based on the max amount of time)
  - If update parameter response is received before timeout, osal_stop_timerEx called to cancel event
  - If timeout expires before response is received, peripheral device terminates connection

GAP Peripheral Role Profile: RSSI Measurement

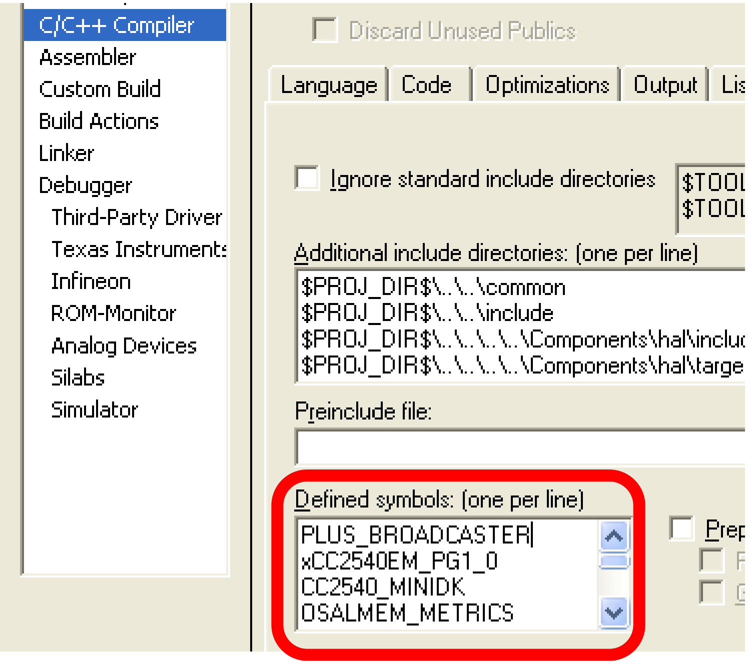
- The peripheral role profile can provide RSSI measurements to the application with the callback function rssiAvailableCB (this feature is not used by the KeyFobDemo application)
- RSSI can only be read when device is in a connection
- RSSI value only updated when data is received (in future release, RSSI will update with each link layer connection event)
- GAPROLE_RSSI_READ_RATE parameter sets the amount of time in milliseconds between RSSI reads
- When device enters connected state, profile calls osal_start_timerEx to schedule an RSSI_READ_EVT
- Every time RSSI_READ_EVT occurs:
  - Profile calls HCI_ReadRssiCmd function
  - Peripheral role profile receives OSAL message from GAP (message type HCI_GAP_EVENT_EVENT) containing RSSI reading
  - Profile calls callback function rssiAvailableCB to notify application of value

GAP Peripheral Role Profile: After Link Termination

- When a connection is terminated for any reason, GAP sends OSAL message to GAP application (peripheral role profile) of type GAP_TERMINATE_LINK_EVENT
- Profile calls the callback function peripheralStateNotificationCB to notify application that GAP state has changed, and whether the link terminated due to supervision timeout, or due to a terminate link request
- Profile schedules a START_ADVERTISING_EVT using the osal_start_timerEx function, with the amount of time determined by the value of the parameter GAPROLE_ADVERT_OFF_TIME

GAP Peripheral Role Profile: Switching to multi-role profile

- In addition to peripheral role profile, Release includes a peripheral / broadcaster multi-role profile
- To use multi-role profile:
  - Exclude the files “peripheral.c” and “peripheral.h” from the KeyFobDemo project (right-click on files and select “options” in IAR, then check the box for “Exclude from build”)
  - Add the files “peripheralBroadcaster.c” and “peripheralBroadcaster.h” to the project under the “Profiles” group
- In IAR Project options (compiler settings), add the preprocessor defined symbol “PLUS_BROADCASTER”
- All functions have the same names and work identical to the functions in peripheral.c
- Advertisements can now be enabled or disabled by setting GAPROLE_ADVERT_ENABLED parameter value to TRUE while in a connected state
- Advertisements will be non-connectable



□ Discard Unused Publics

□ Ignore standard include directories

$TOOL
$TOOL

Additional include directories: (one per line)

```
$PROJ_DIR$\..\..\common
$PROJ_DIR$\..\..\include
$PROJ_DIR$\..\..\..\..\Components\hal\incluc
$PROJ_DIR$\..\..\..\..\Components\hal\targe
```

Preinclude file:

Defined symbols: (one per line)

```
PLUS_BROADCASTER
xCC2540EM_PG1_0
CC2540_MINIDK
OSALMEM_METRICS
```

□ Prep
□ F
□ G

GATT Service Profiles: Overview

- Allows device to implement a GATT service:
  - As defined by Bluetooth SIG
  - Custom
- Provides means for application to read and write service data on the attribute table
- Lets a remote GATT client access characteristics through:
  - GATT reads
  - GATT writes
  - GATT notifications and indications
- Verifies the validity of data being written from a remote device
- GATT service profiles typically do not need to be OSAL tasks, and are accessed directly by the protocol stack and by the application
- Most GATT service profiles have a very similar structure
- New GATT service profiles can be easily created by copying an existing profile and renaming variables and functions

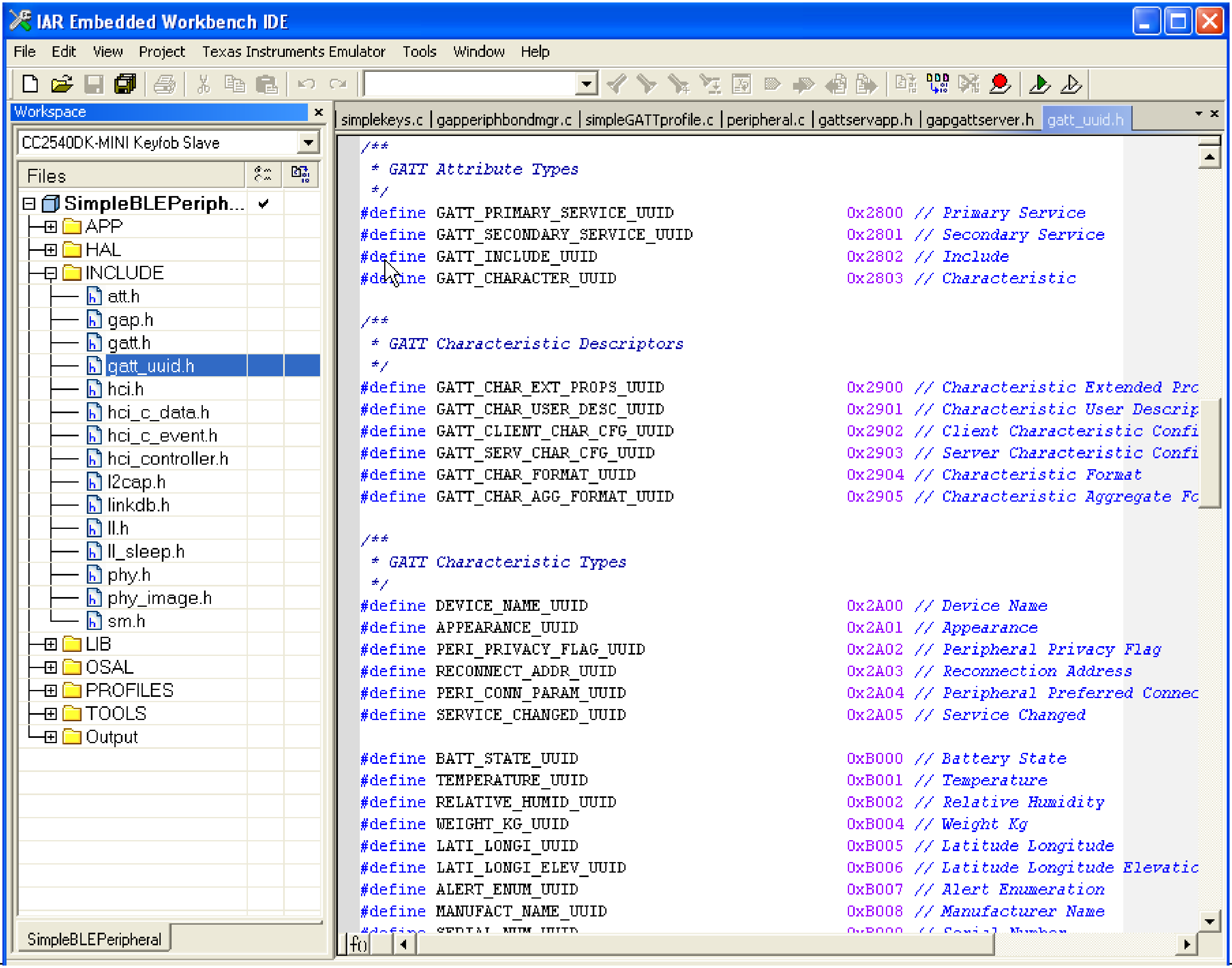
GATT Service Profiles: Typical Functions

- Public functions:
  - ProfileName_AddService – registers attribute list and callback functions with GATT server
  - ProfileName_RegisterAppCBs – allows function to register application callback functions with profiles.
  - ProfileName_SetParameter – allows application to set attribute data values; also sends out notifications of characteristics when enabled
  - ProfileName_GetParameter – allows application to get attribute values
- Private GATT server callback functions:
  - profileName_ReadAttrCB – called when a GATT read request is received from a GATT client; returns attribute data to GATT server for read response
  - profileName_ValidateWriteAttrCB – called when a GATT write request is received from a GATT client; validates data being written and writes new value if data is valid; sends write response with appropriate error message if data is invalid

GATT Service Profiles: Structure

- Attribute value variables are defined as static and are local to the module
- Standard UUID’s (from BT SIG) are defined in gatt_uuid.h
- Custom UUID’s are defined in profiles own header file
- In addition to attribute values, profile defines an array of type gattAttribute_t, in which each element contains data related to each attibute:
  - Attribute type (UUID length in bytes and UUID itself)
  - Permissions
  - Handle – profile initializes this to zero, and server updates when building the table
  - Pointer to data value

GATT Service Profiles - UUID



```
/**
 * GATT Attribute Types
 */
#define GATT_PRIMARY_SERVICE_UUID                  0x2800 // Primary Service
#define GATT_SECONDARY_SERVICE_UUID                0x2801 // Secondary Service
#define GATT_INCLUDE_UUID                          0x2802 // Include
#define GATT_CHARACTER_UUID                        0x2803 // Characteristic

/**
 * GATT Characteristic Descriptors
 */
#define GATT_CHAR_EXT_PROPS_UUID                   0x2900 // Characteristic Extended Pr
#define GATT_CHAR_USER_DESC_UUID                   0x2901 // Characteristic User Descrip
#define GATT_CLIENT_CHAR_CFG_UUID                  0x2902 // Client Characteristic Confi
#define GATT_SERV_CHAR_CFG_UUID                    0x2903 // Server Characteristic Confi
#define GATT_CHAR_FORMAT_UUID                      0x2904 // Characteristic Format
#define GATT_CHAR_AGG_FORMAT_UUID                  0x2905 // Characteristic Aggregate F

/**
 * GATT Characteristic Types
 */
#define DEVICE_NAME_UUID                           0x2A00 // Device Name
#define APPEARANCE_UUID                            0x2A01 // Appearance
#define PERI_PRIVACY_FLAG_UUID                     0x2A02 // Peripheral Privacy Flag
#define RECONNECT_ADDR_UUID                        0x2A03 // Reconnection Address
#define PERI_CONN_PARAM_UUID                       0x2A04 // Peripheral Preferred Connec
#define SERVICE_CHANGED_UUID                       0x2A05 // Service Changed

#define BATT_STATE_UUID                            0xB000 // Battery State
#define TEMPERATURE_UUID                           0xB001 // Temperature
#define RELATIVE_HUMID_UUID                        0xB002 // Relative Humidity
#define WEIGHT_KG_UUID                             0xB004 // Weight Kg
#define LATI_LONGI_UUID                            0xB005 // Latitude Longitude
#define LATI_LONGI_ELEV_UUID                       0xB006 // Latitude Longitude Elevati
#define ALERT_ENUM_UUID                            0xB007 // Alert Enumeration
#define MANUFACT_NAME_UUID                         0xB008 // Manufacturer Name
#define SERIAL_NUM_UUID                            0xB009 // Serial Number
```

GATT Service Profiles: InitService Function

- InitService function called by application
- When InitService function is called, two variables must be created
  - gattService_t service – includes the number of attributes from the service, and the attribute array itself
  - gattServiceCBs_t serviceCBs – includes two function pointers: ReadAttrCB and ValidateWriteAttrCB (if service doesn't have any readable or writeable attributes, the corresponding pointer can be set to NULL)
- Function calls GATTServApp_RegisterService, with the two variables as parameters to register the attributes and callback functions with the GATT server application

GATT Service Profiles: RegisterAppCBs Function

- Only required if profile needs to notify application of information related to the profile

- Examples:
  - In proximity service profile, application needs to know if link-loss or path-loss alert characteristic values have changed
  - In accelerometer profile, application needs to know if accelerometer enabler characteristic value is changed

- Profile must define a type for the callback function pointer in the header file

GATT Service Profiles: Notifications

- Notification / indication handling is typically part of the SetParameter function

- The criteria for when to send notifications or indications can either be set in profile itself or in the application
  - Might be defined by a profile specification

Agenda

- **Bluetooth Low Energy Protocol Stack**
  - Link Layer - Basics of BLE communication
  - GAP (Generic Access Profile) - GAP roles, advertisements, connections
  - GATT (Generic Attribute Profile) - Attribute table data format, reads, writes, notifications

- **CC2540 Software Overview**
  - IAR Embedded Workbench IDE - Development environment overview, CC Debugger
  - CC2540 Software - Architecture and Structure, KeyFobDemo Application
  - OSAL (Operating System Abstraction Layer) - Task setup and initialization, events and processing, messaging and memory managers
  - GAP Role Profiles - Peripheral and Peripheral / Broadcaster role profiles
  - GATT Profiles - Structure and format, initialization, application callbacks

➢ **Questions / Hands-On**

Hands-on

1. Walkthrough- flash devices, set secondary address set
2. Walkthrough- USB Dongle driver install
3. Walkthrough- Use BTool to enable SimpleGATTProfile notifications
4. Exercise- Use BTool; enable button press notifications

(open up IAR)

3. Exercise – change periodic notification time, and have increment
4. Walkthrough – Turn on LED upon connection
5. Exercise – Set characteristic values upon disconnect (need to show demo before exercise)
6. Walkthrough - Add a 5th characteristic to SimpleGATTProfile

Labs

KeyPress

- Use TI flash programmer to assign address and flash images.
- Enable keypress notifications from keyfob to USB dongle.
- Use Btool to see logging

Sniffer

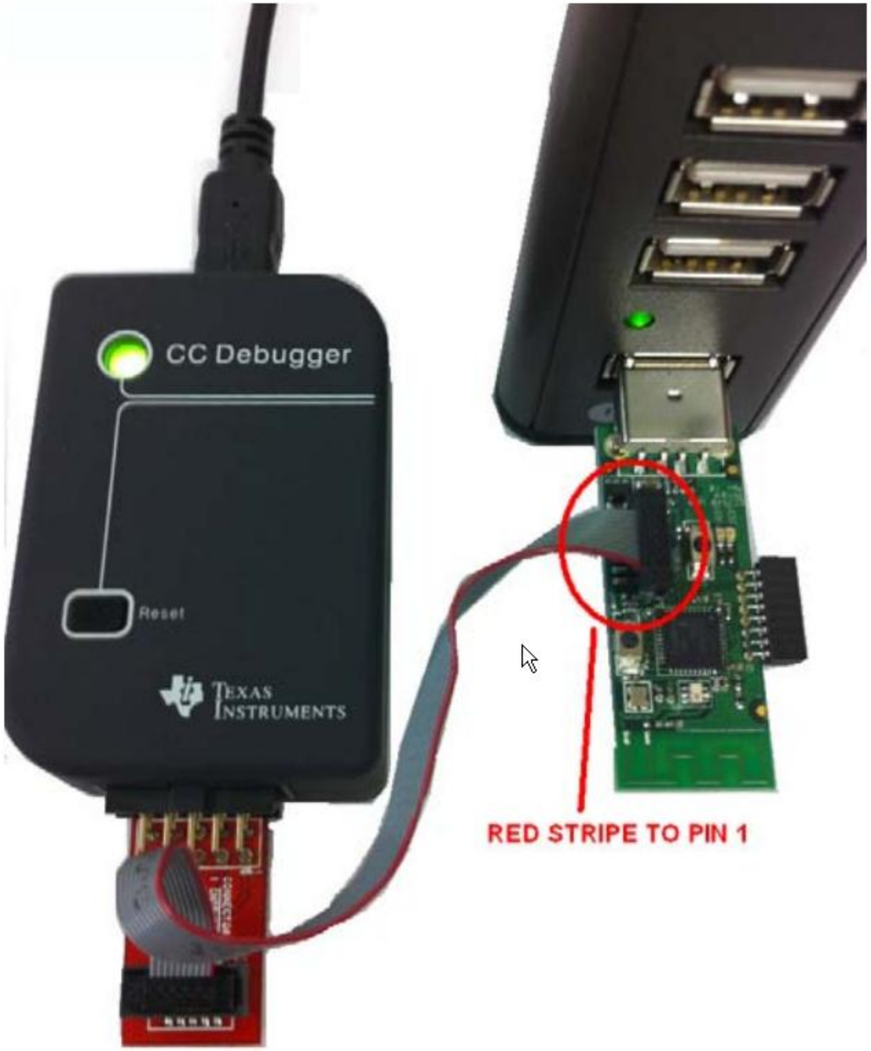
- Load sniffer image into USB Dongle
- Enable advertisements on keyfob and watch on sniffer.

Accelerometer (optional)

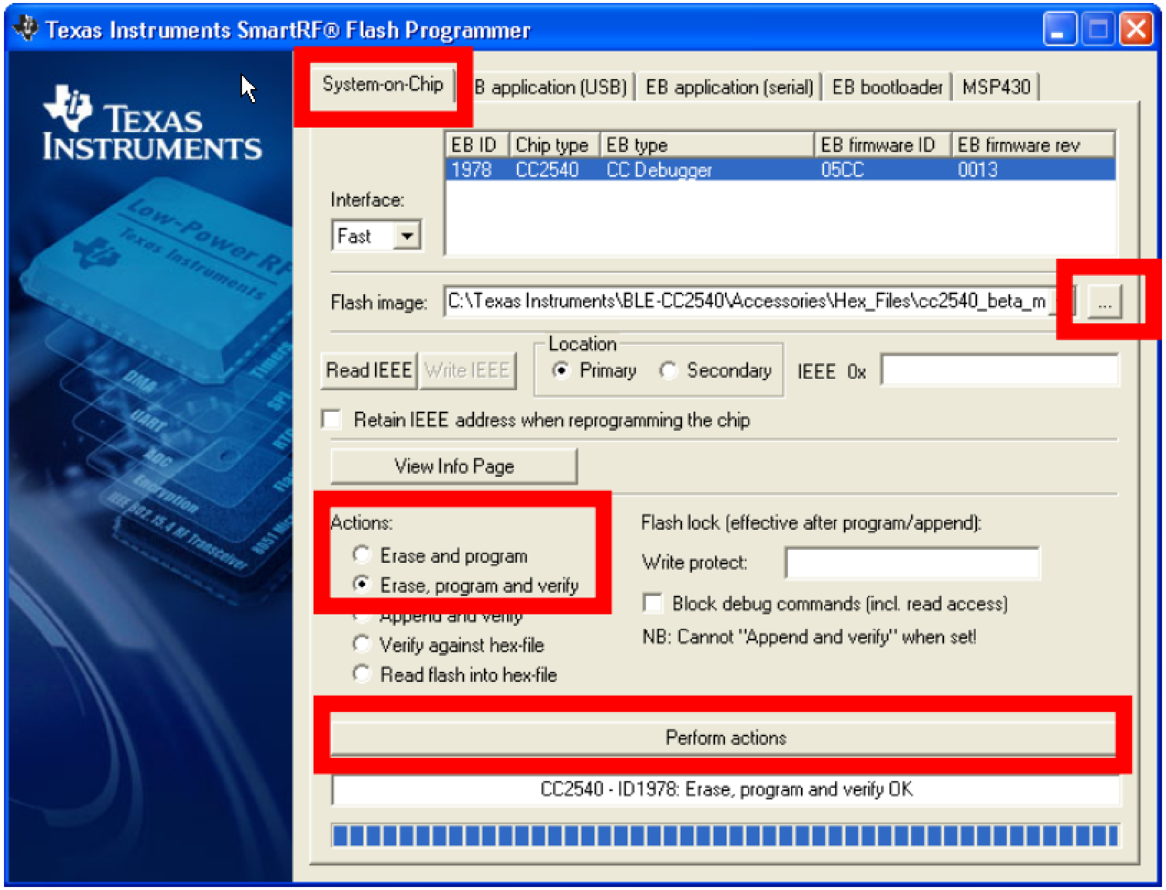
- Load keyfobdemo application
- Enable accelerometer notifications

Lab #1.1

1. Connect USB Dongle and CC Debugger as shown
2. CC debugger light should be green and USB dongle LED should be green.



Lab #1.2



EB ID	Chip type	EB type	EB firmware ID	EB firmware rev
1978	CC2540	CC Debugger	05CC	0013

Interface: Fast

Flash image: C:\Texas Instruments\BLE-CC2540\Accessories\Hex_Files\cc2540_beta_m

Location: ● Primary ○ Secondary

IEEE 0x

□ Retain IEEE address when reprogramming the chip

Actions:
○ Erase and program
● Erase, program and verify
○ Append and verify
○ Verify against hex-file
○ Read flash into hex-file

Flash lock (effective after program/append):

Write protect:

□ Block debug commands (incl. read access)

NB: Cannot "Append and verify" when set!

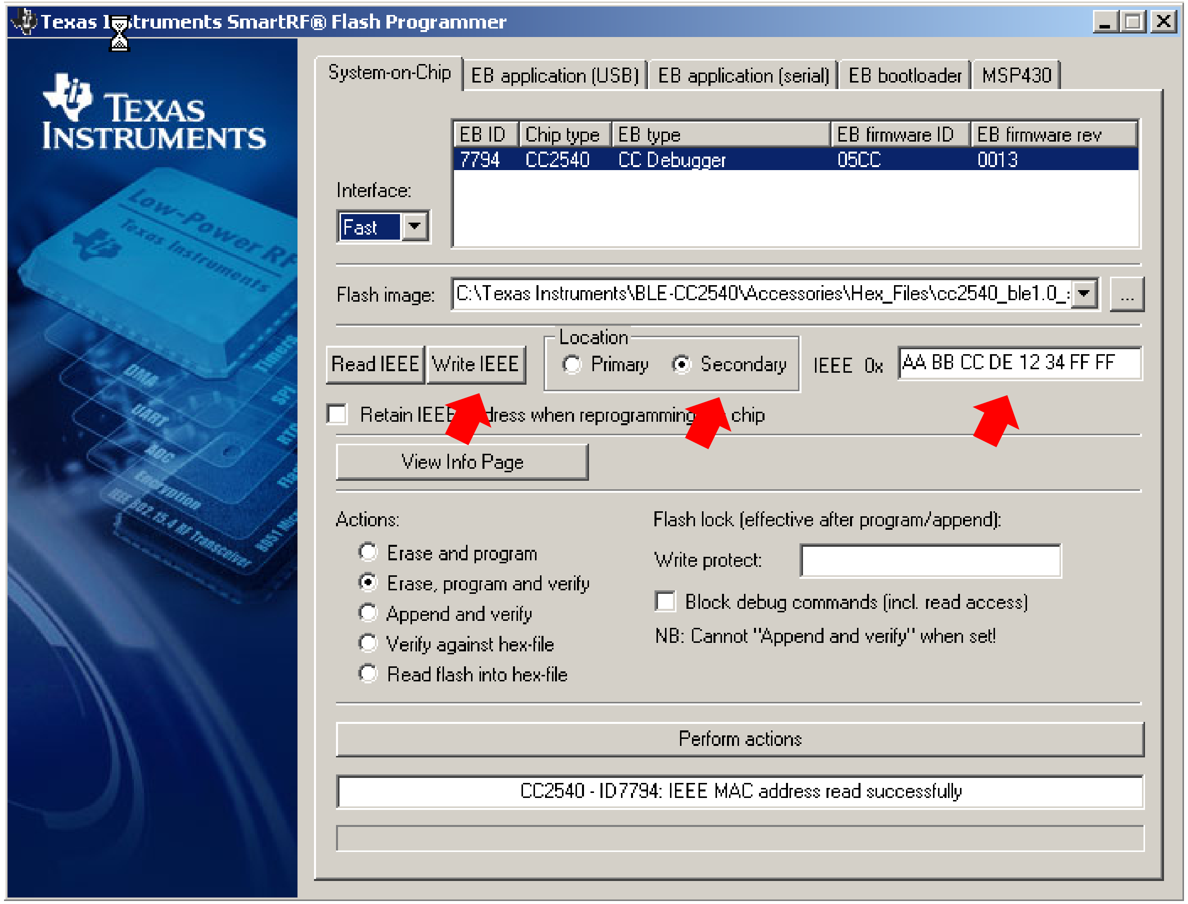
CC2540 - ID1978: Erase, program and verify OK

1. Use SmartRF flash programmer to download .hex file

C:\Texas Instruments\BLE-CC2540\Accessories\Hex_Files\cc2540_ble1.0_master_usb_dongle.hex

* USB Dongle LED should turn red

Lab #1.2b



EB ID	Chip type	EB type	EB firmware ID	EB firmware rev
7794	CC2540	CC Debugger	05CC	0013

Interface: Fast

Flash image: C:\Texas Instruments\BLE-CC2540\Accessories\Hex_Files\cc2540_ble1.0_:

Location ○ Primary ● Secondary

IEEE 0x AA BB CC DE 12 34 FF FF

□ Retain IEEE address when reprogramming the chip

Actions:

○ Erase and program
● Erase, program and verify
○ Append and verify
○ Verify against hex-file
○ Read flash into hex-file

Flash lock (effective after program/append):

Write protect:

□ Block debug commands (incl. read access)

NB: Cannot "Append and verify" when set!

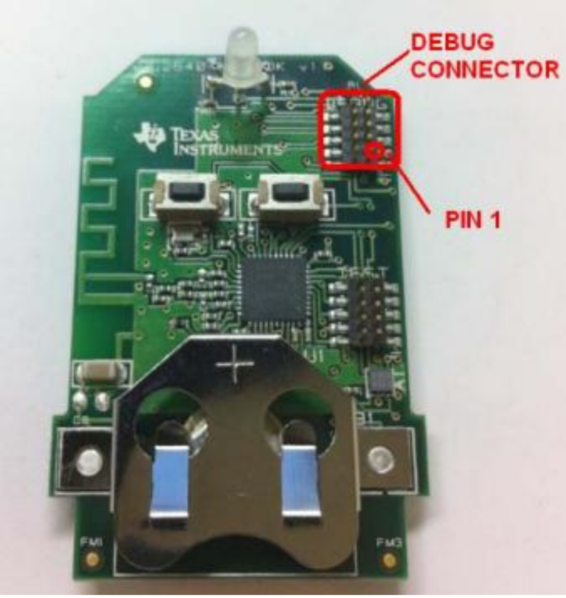
CC2540 - ID7794: IEEE MAC address read successfully

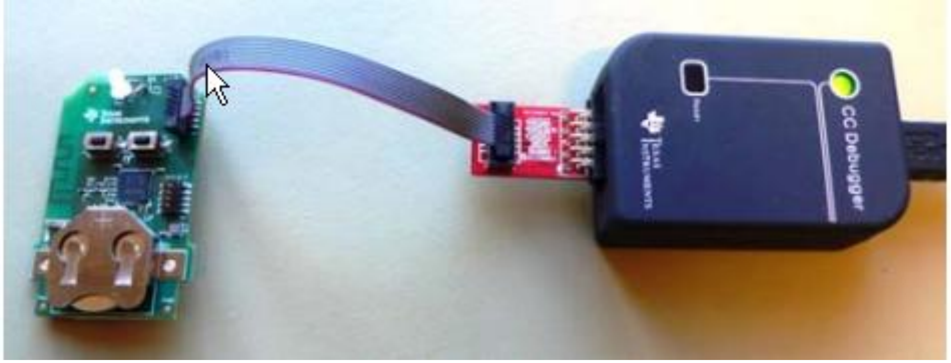
1. Use SmartRF flash programmer change address

- Click secondary radio button
- Enter in new address – left six bytes
- Click Write IEEE
- Read back to verify

Lab #1.3

1. Connect keyfob and CC Debugger as shown
2. Insert battery
3. Press button to stop buzzer
4. CC debugger light should be green. (may need to detach and attach USB cable)

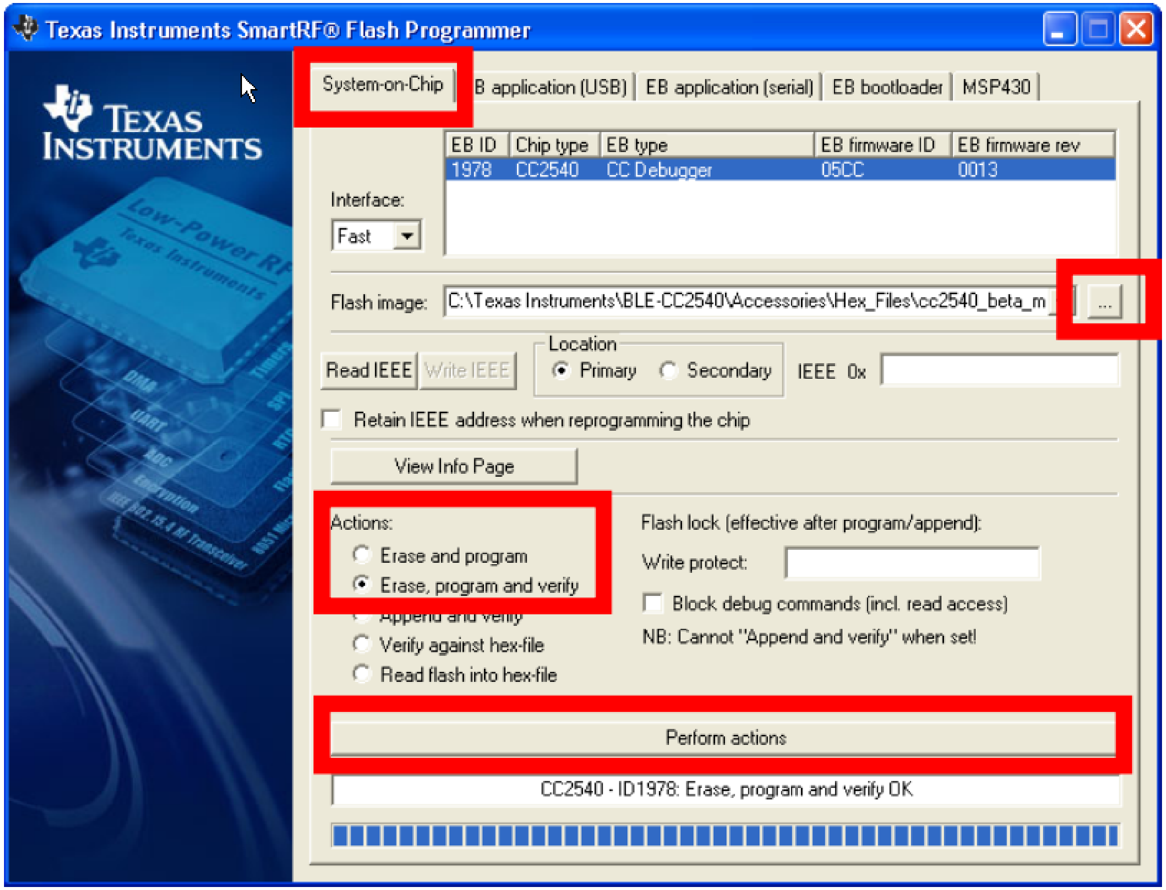




1. Use SmartRF flash programmer to download .hex file

C:\Texas Instruments\BLE-CC2540\Accessories\Hex_Files\cc2540_ble1.0_slave_keyfob.hex

Lab #1.4



EB ID	Chip type	EB type	EB firmware ID	EB firmware rev
1978	CC2540	CC Debugger	05CC	0013

Interface: Fast

Flash image: C:\Texas Instruments\BLE-CC2540\Accessories\Hex_Files\cc2540_beta_m

Location ● Primary ○ Secondary

IEEE 0x

□ Retain IEEE address when reprogramming the chip

Actions:
○ Erase and program
● Erase, program and verify
○ Append and verify
○ Verify against hex-file
○ Read flash into hex-file

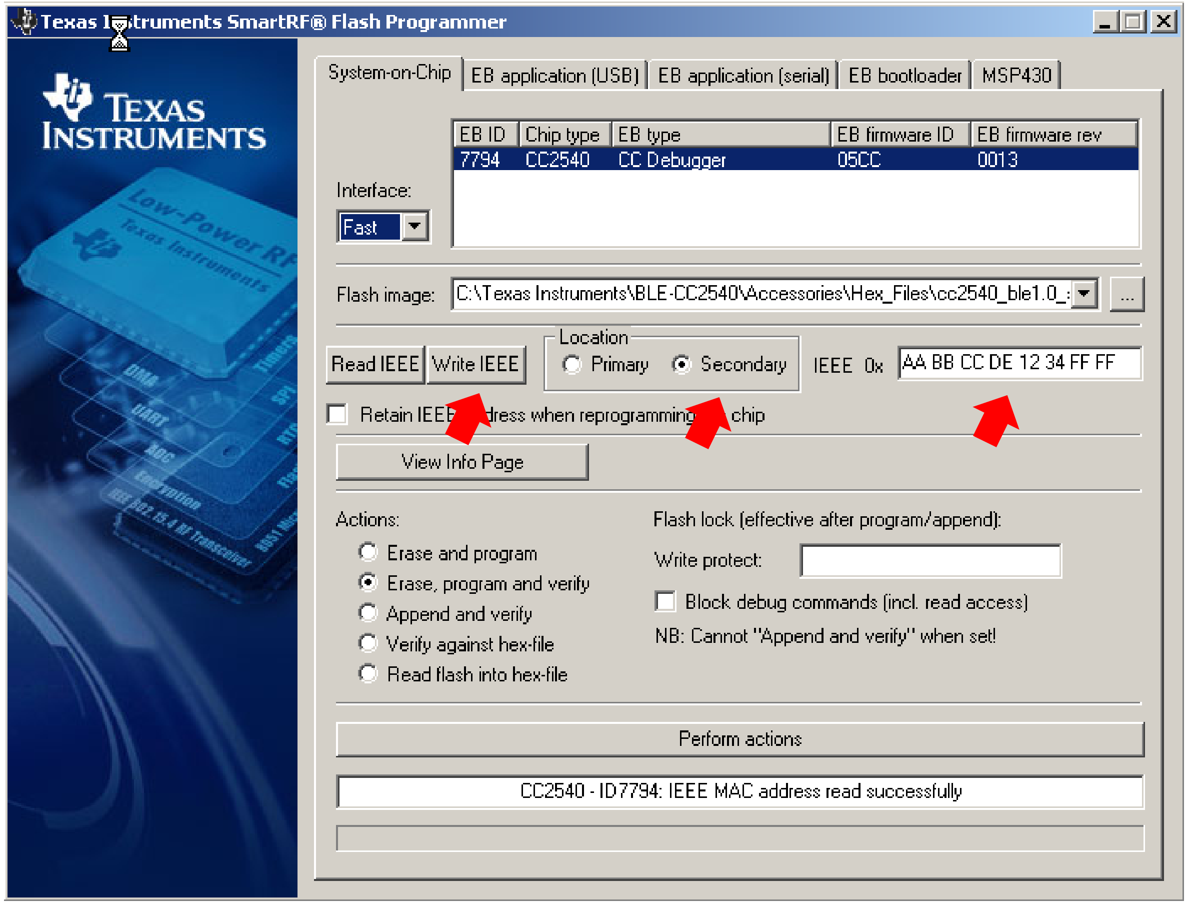
Flash lock (effective after program/append):
Write protect:
□ Block debug commands (incl. read access)
NB: Cannot "Append and verify" when set!

CC2540 - ID1978: Erase, program and verify OK

1. Use SmartRF flash programmer to download .hex file

C:\Texas Instruments\BLE-CC2540\Accessories\Hex_Files\cc2540_ble1.0_slave_keyfob.hex

Lab #1.4b



Texas Instruments SmartRF® Flash Programmer

System-on-Chip | EB application (USB) | EB application (serial) | EB bootloader | MSP430

EB ID	Chip type	EB type	EB firmware ID	EB firmware rev
7794	CC2540	CC Debugger	05CC	0013

Interface: Fast

Flash image: C:\Texas Instruments\BLE-CC2540\Accessories\Hex_Files\cc2540_ble1.0_

Read IEEE | Write IEEE

Location: ○ Primary ● Secondary

IEEE 0x AA BB CC DE 12 34 FF FF

□ Retain IEEE address when reprogramming the chip

View Info Page

Actions:

- ○ Erase and program
- ● Erase, program and verify
- ○ Append and verify
- ○ Verify against hex-file
- ○ Read flash into hex-file

Flash lock (effective after program/append):

Write protect:

□ Block debug commands (incl. read access)

NB: Cannot "Append and verify" when set!

Perform actions

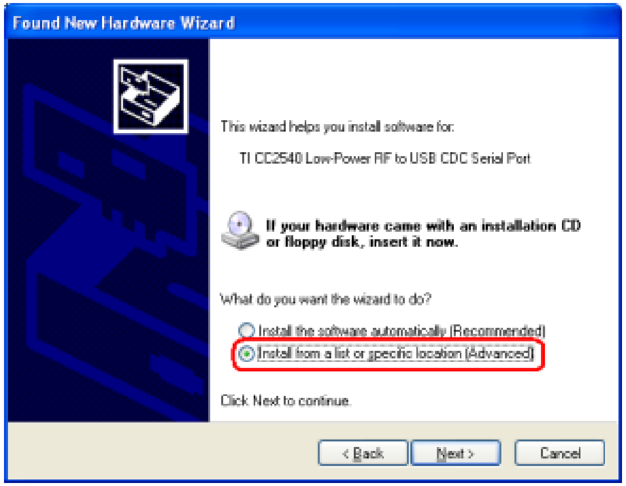
CC2540 - ID7794: IEEE MAC address read successfully

1. Use SmartRF flash programmer change address

- Click secondary radio button
- Enter in new address – left six bytes
- Click Write IEEE
- Read back to verify

Lab #1.5

1. Plug in USB Dongle, install driver from - C:\Texas Instruments\BLE-CC2540\Accessories\Drivers
2. Use Device Manager to determine COM used for USB Dongle.
3. Start Btool.exe (C:\Texas Instruments\BLE-CC2540\Projects\Btool)
4. Open Device (this is COM port which USB Dongle shows up as)


Found New Hardware Wizard

This wizard helps you install software for:

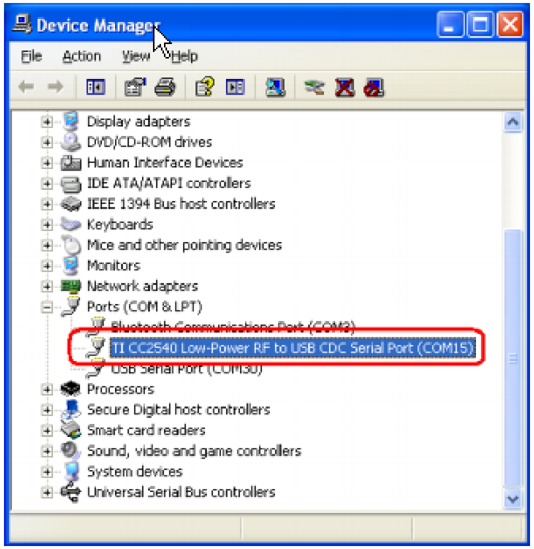
TI CC2540 Low-Power RF to USB CDC Serial Port

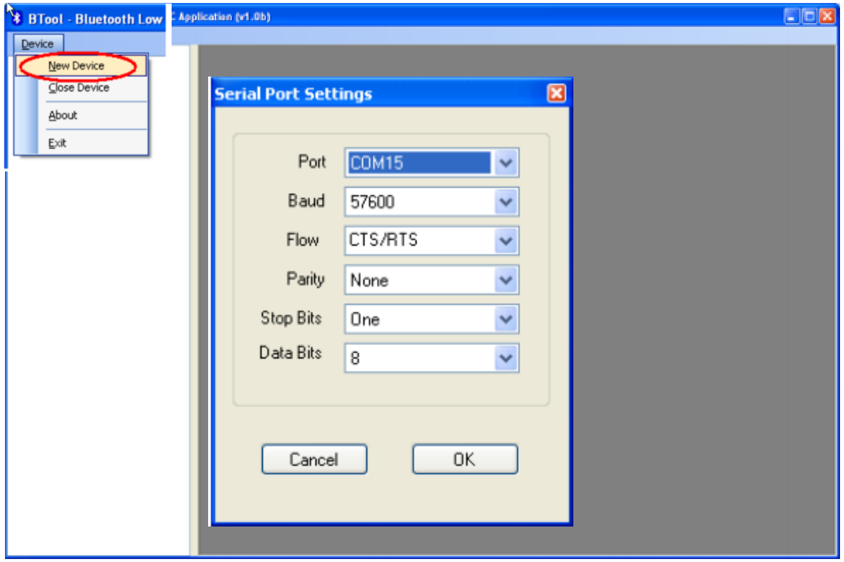
If your hardware came with an installation CD or floppy disk, insert it now.

What do you want the wizard to do?

○ Install the software automatically (Recommended)
● Install from a list or specific location (Advanced)

Click Next to continue.

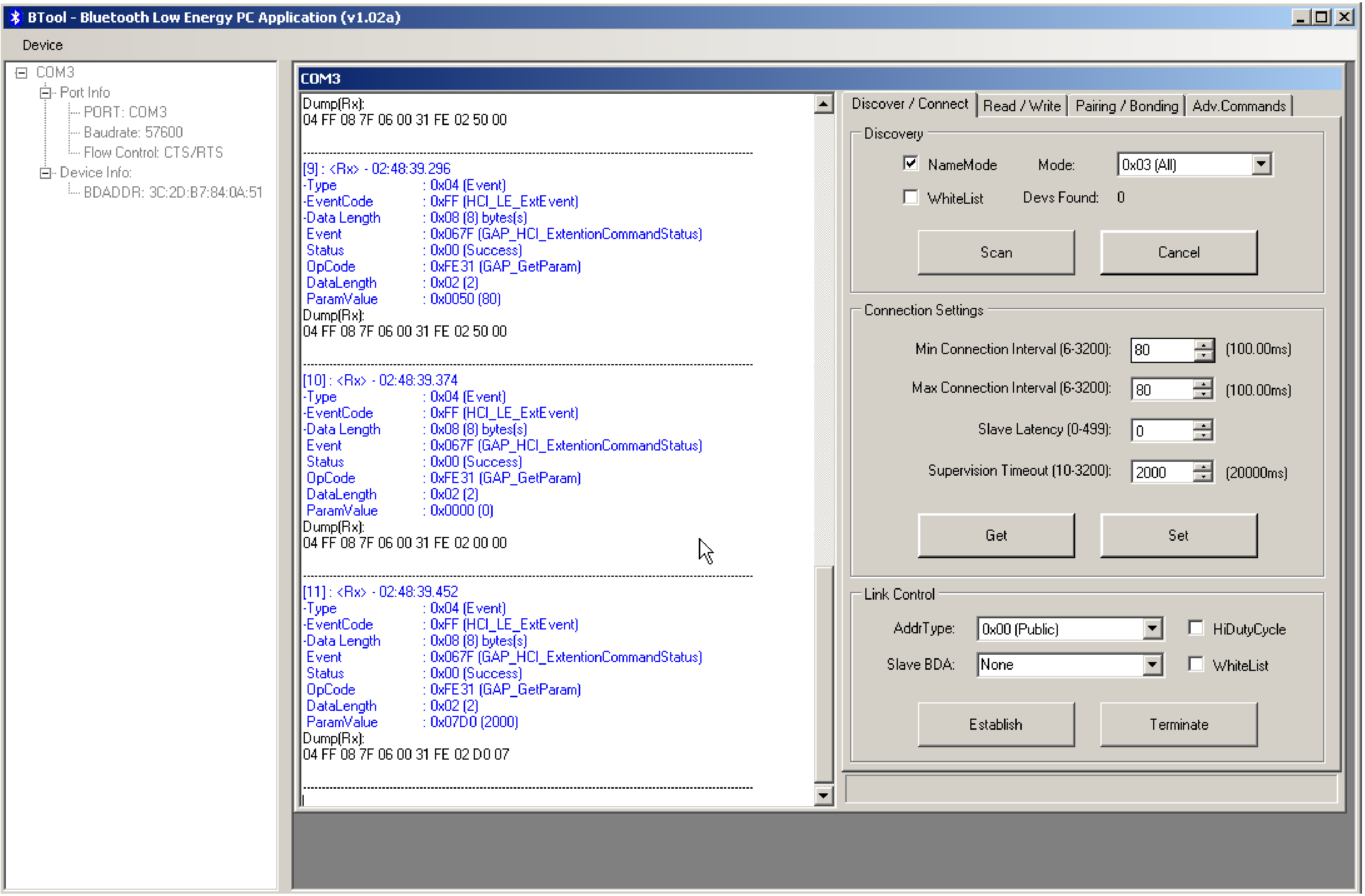

Device Manager
File
Action
View
Help
Display adapters
DVD/CD-ROM drives
Human Interface Devices
IDE ATA/ATAPI controllers
IEEE 1394 Bus host controllers
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
Bluetooth Communications Port (COM3)
TI CC2540 Low-Power RF to USB CDC Serial Port (COM15)
USB Serial Port (COM30)
Processors
Secure Digital host controllers
Smart card readers
Sound, video and game controllers
System devices
Universal Serial Bus controllers



Serial Port Settings

Port: COM15
Baud: 57600
Flow: CTS/RTS
Parity: None
Stop Bits: One
Data Bits: 8

Lab #1.6



BTool - Bluetooth Low Energy PC Application (v1.02a)

Device

- COM3
  - Port Info
    - PORT: COM3
    - Baudrate: 57600
    - Flow Control: CTS/RTS
  - Device Info:
    - BDADDR: 3C:2D:B7:84:0A:51

COM3

```
Dump(Rx):
04 FF 08 7F 06 00 31 FE 02 50 00

------------------------------------------------------------
[9] : <Rx> - 02:48:39.296
-Type              : 0x04 (Event)
-EventCode         : 0xFF (HCI_LE_ExtEvent)
-Data Length       : 0x08 (8) bytes(s)
 Event             : 0x067F (GAP_HCI_ExtentionCommandStatus)
 Status            : 0x00 (Success)
 OpCode            : 0xFE31 (GAP_GetParam)
 DataLength        : 0x02 (2)
 ParamValue        : 0x0050 (80)
Dump(Rx):
04 FF 08 7F 06 00 31 FE 02 50 00

------------------------------------------------------------
[10] : <Rx> - 02:48:39.374
-Type              : 0x04 (Event)
-EventCode         : 0xFF (HCI_LE_ExtEvent)
-Data Length       : 0x08 (8) bytes(s)
 Event             : 0x067F (GAP_HCI_ExtentionCommandStatus)
 Status            : 0x00 (Success)
 OpCode            : 0xFE31 (GAP_GetParam)
 DataLength        : 0x02 (2)
 ParamValue        : 0x0000 (0)
Dump(Rx):
04 FF 08 7F 06 00 31 FE 02 00 00

------------------------------------------------------------
[11] : <Rx> - 02:48:39.452
-Type              : 0x04 (Event)
-EventCode         : 0xFF (HCI_LE_ExtEvent)
-Data Length       : 0x08 (8) bytes(s)
 Event             : 0x067F (GAP_HCI_ExtentionCommandStatus)
 Status            : 0x00 (Success)
 OpCode            : 0xFE31 (GAP_GetParam)
 DataLength        : 0x02 (2)
 ParamValue        : 0x07D0 (2000)
Dump(Rx):
04 FF 08 7F 06 00 31 FE 02 D0 07

------------------------------------------------------------
```

Discovery

☑ NameMode Mode: 0x03 (All)

☐ WhiteList Devs Found: 0

Connection Settings

Min Connection Interval (6-3200): 80 (100.00ms)

Max Connection Interval (6-3200): 80 (100.00ms)

Slave Latency (0-499): 0

Supervision Timeout (10-3200): 2000 (20000ms)

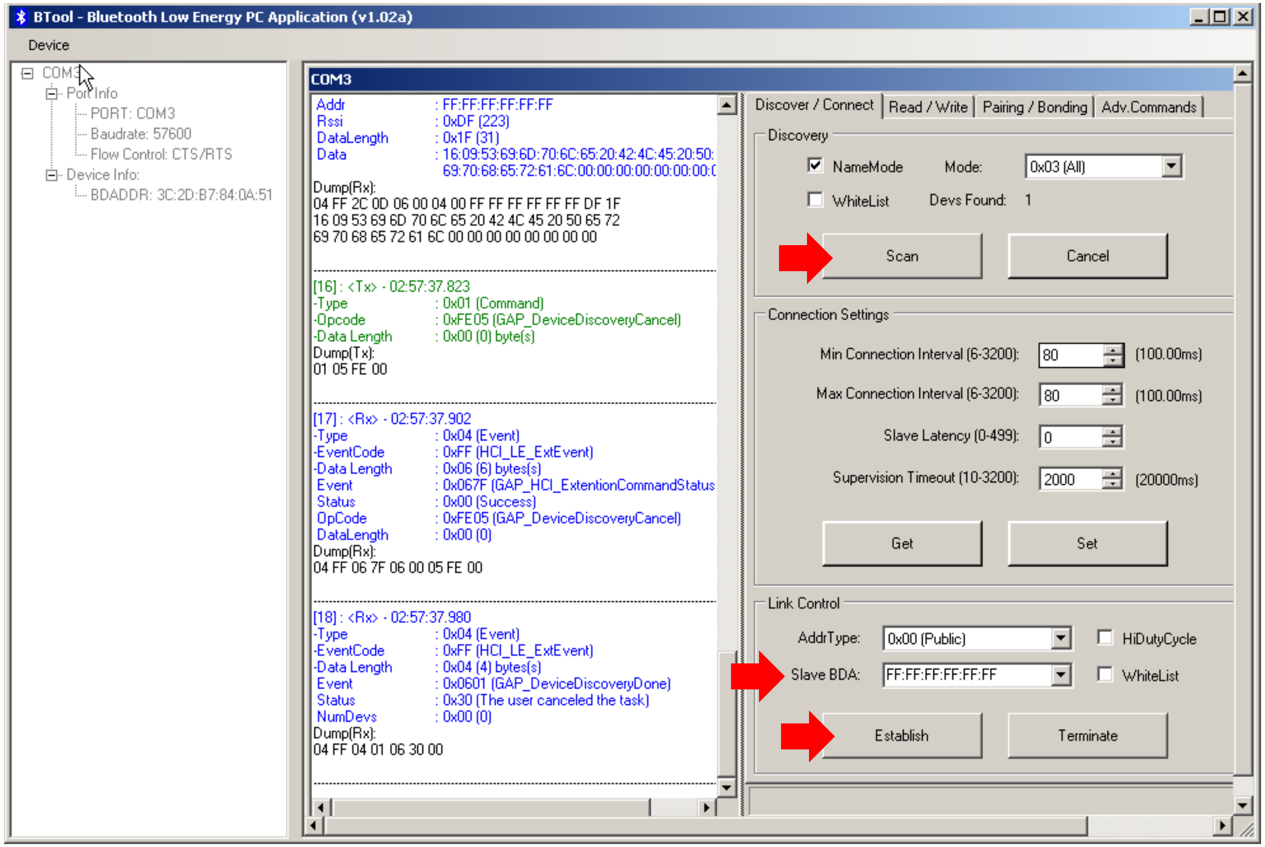
Link Control

AddrType: 0x00 (Public) ☐ HiDutyCycle

Slave BDA: None ☐ WhiteList

Lab #1.7

1. Press button on keyfob to begin advertising for 20 sec
2. Scan, select you keyfob address, and establish



```
COM3
  Port Info
    PORT: COM3
    Baudrate: 57600
    Flow Control: CTS/RTS
  Device Info:
    BDADDR: 3C:2D:B7:84:0A:51
```

```
 Addr                : FF:FF:FF:FF:FF:FF
 Rssi                : 0xDF (223)
 DataLength          : 0x1F (31)
 Data                : 16:09:53:69:6D:70:6C:65:20:42:4C:45:20:50:
                       69:70:68:65:72:61:6C:00:00:00:00:00:00:00:0
Dump(Rx):
04 FF 2C 0D 06 00 04 00 FF FF FF FF FF FF DF 1F
16 09 53 69 6D 70 6C 65 20 42 4C 45 20 50 65 72
69 70 68 65 72 61 6C 00 00 00 00 00 00 00 00

------------------------------------------------------------
[16] : <Tx> - 02:57:37.823
-Type                : 0x01 (Command)
-Opcode              : 0xFE05 (GAP_DeviceDiscoveryCancel)
-Data Length         : 0x00 (0) byte(s)
Dump(Tx):
01 05 FE 00

------------------------------------------------------------
[17] : <Rx> - 02:57:37.902
-Type                : 0x04 (Event)
-EventCode           : 0xFF (HCI_LE_ExtEvent)
-Data Length         : 0x06 (6) bytes(s)
 Event               : 0x067F (GAP_HCI_ExtentionCommandStatus
 Status              : 0x00 (Success)
 OpCode              : 0xFE05 (GAP_DeviceDiscoveryCancel)
 DataLength          : 0x00 (0)
Dump(Rx):
04 FF 06 7F 06 00 05 FE 00

------------------------------------------------------------
[18] : <Rx> - 02:57:37.980
-Type                : 0x04 (Event)
-EventCode           : 0xFF (HCI_LE_ExtEvent)
-Data Length         : 0x04 (4) bytes(s)
 Event               : 0x0601 (GAP_DeviceDiscoveryDone)
 Status              : 0x30 (The user canceled the task)
 NumDevs             : 0x00 (0)
Dump(Rx):
04 FF 04 01 06 30 00

------------------------------------------------------------
```

Discovery

☑ NameMode   Mode: 0x03 (All)

☐ WhiteList   Devs Found: 1

Connection Settings

Min Connection Interval (6-3200): 80 (100.00ms)

Max Connection Interval (6-3200): 80 (100.00ms)

Slave Latency (0-499): 0

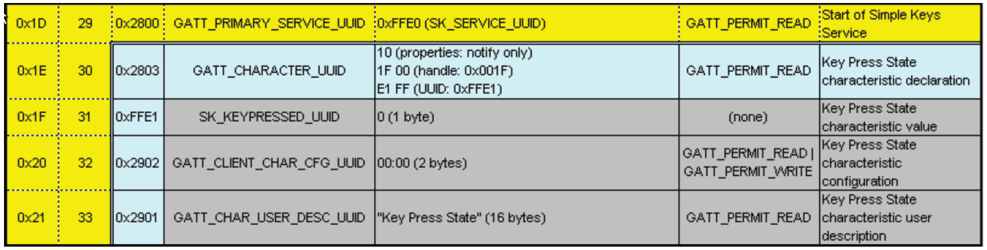
Supervision Timeout (10-3200): 2000 (20000ms)

Link Control

AddrType: 0x00 (Public)   ☐ HiDutyCycle

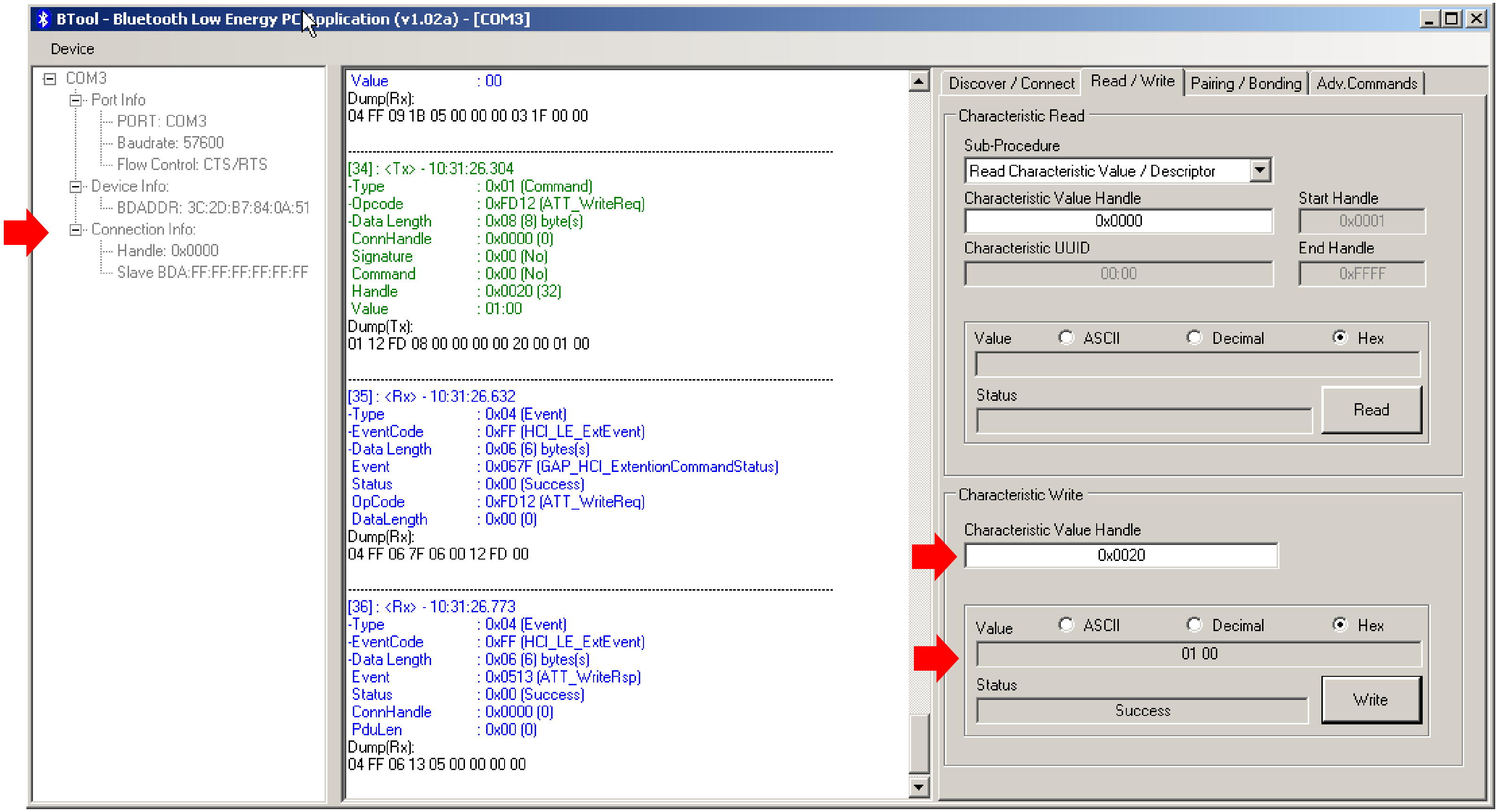
Slave BDA: FF:FF:FF:FF:FF:FF   ☐ WhiteList

Lab #1.8



0x1D	29	0x2800	GATT_PRIMARY_SERVICE_UUID	0xFFE0 (SK_SERVICE_UUID)	GATT_PERMIT_READ	Start of Simple Keys Service
0x1E	30	0x2803	GATT_CHARACTER_UUID	10 (properties: notify only) 1F 00 (handle: 0x001F) E1 FF (UUID: 0xFFE1)	GATT_PERMIT_READ	Key Press State characteristic declaration
0x1F	31	0xFFE1	SK_KEYPRESSED_UUID	0 (1 byte)	(none)	Key Press State characteristic value
0x20	32	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Key Press State characteristic configuration
0x21	33	0x2901	GATT_CHAR_USER_DESC_UUID	"Key Press State" (16 bytes)	GATT_PERMIT_READ	Key Press State characteristic user description

1. Verify connection in left pane
2. Write “01 00” to address 0x0020
3. Press button on keyfob to see notifications



BTool - Bluetooth Low Energy PC Application (v1.02a) - [COM3]

```
COM3
  Port Info
    PORT: COM3
    Baudrate: 57600
    Flow Control: CTS/RTS
  Device Info:
    BDADDR: 3C:2D:B7:84:0A:51
  Connection Info:
    Handle: 0x0000
    Slave BDA:FF:FF:FF:FF:FF:FF
```

```
 Value                : 00
Dump(Rx):
04 FF 09 1B 05 00 00 00 03 1F 00 00

-----------------------------------------------------------------------
[34] : <Tx> - 10:31:26.304
-Type                 : 0x01 (Command)
-Opcode               : 0xFD12 (ATT_WriteReq)
-Data Length          : 0x08 (8) byte(s)
 ConnHandle           : 0x0000 (0)
 Signature            : 0x00 (No)
 Command              : 0x00 (No)
 Handle               : 0x0020 (32)
 Value                : 01:00
Dump(Tx):
01 12 FD 08 00 00 00 00 20 00 01 00

-----------------------------------------------------------------------
[35] : <Rx> - 10:31:26.632
-Type                 : 0x04 (Event)
-EventCode            : 0xFF (HCI_LE_ExtEvent)
-Data Length          : 0x06 (6) bytes(s)
 Event                : 0x067F (GAP_HCI_ExtentionCommandStatus)
 Status               : 0x00 (Success)
 OpCode               : 0xFD12 (ATT_WriteReq)
 DataLength           : 0x00 (0)
Dump(Rx):
04 FF 06 7F 06 00 12 FD 00

-----------------------------------------------------------------------
[36] : <Rx> - 10:31:26.773
-Type                 : 0x04 (Event)
-EventCode            : 0xFF (HCI_LE_ExtEvent)
-Data Length          : 0x06 (6) bytes(s)
 Event                : 0x0513 (ATT_WriteRsp)
 Status               : 0x00 (Success)
 ConnHandle           : 0x0000 (0)
 PduLen               : 0x00 (0)
Dump(Rx):
04 FF 06 13 05 00 00 00 00
```

Characteristic Read

Sub-Procedure: Read Characteristic Value / Descriptor

Characteristic Value Handle: 0x0000

Start Handle: 0x0001

Characteristic UUID: 00:00

End Handle: 0xFFFF

Value ○ ASCII ○ Decimal ● Hex

Status

Characteristic Write

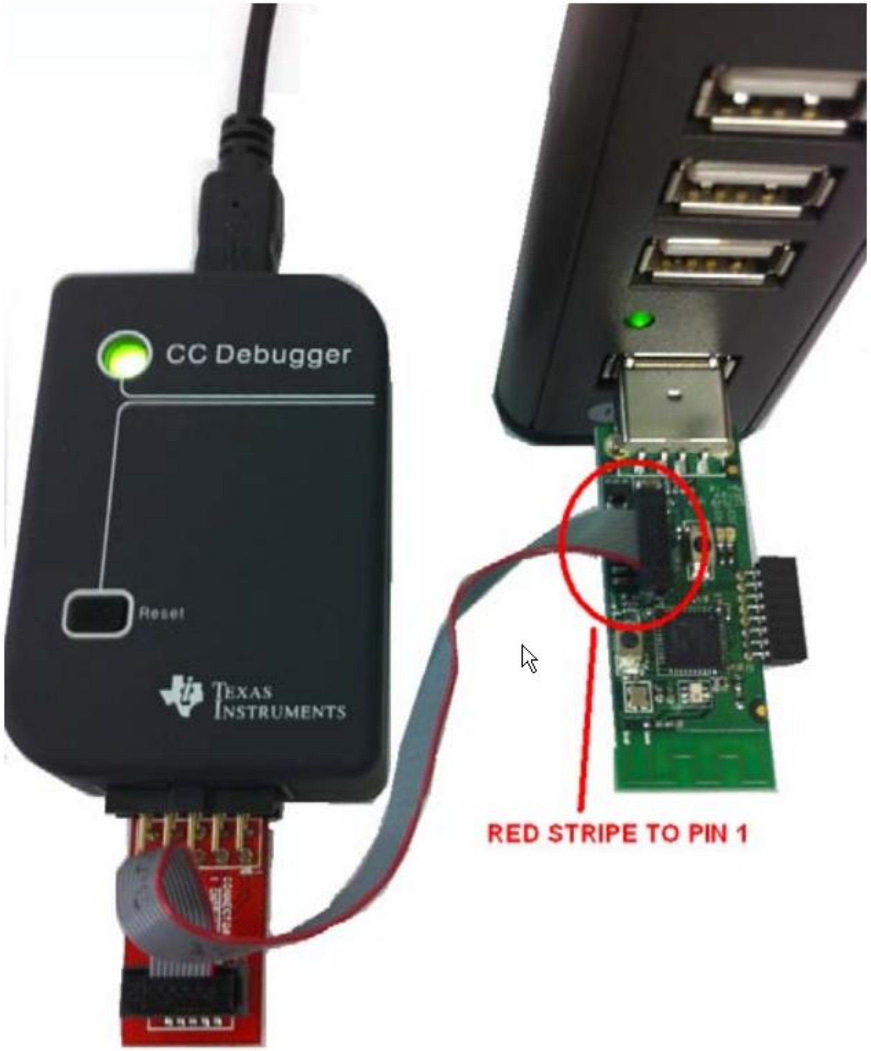
Characteristic Value Handle: 0x0020

Value ○ ASCII ○ Decimal ● Hex: 01 00

Status: Success

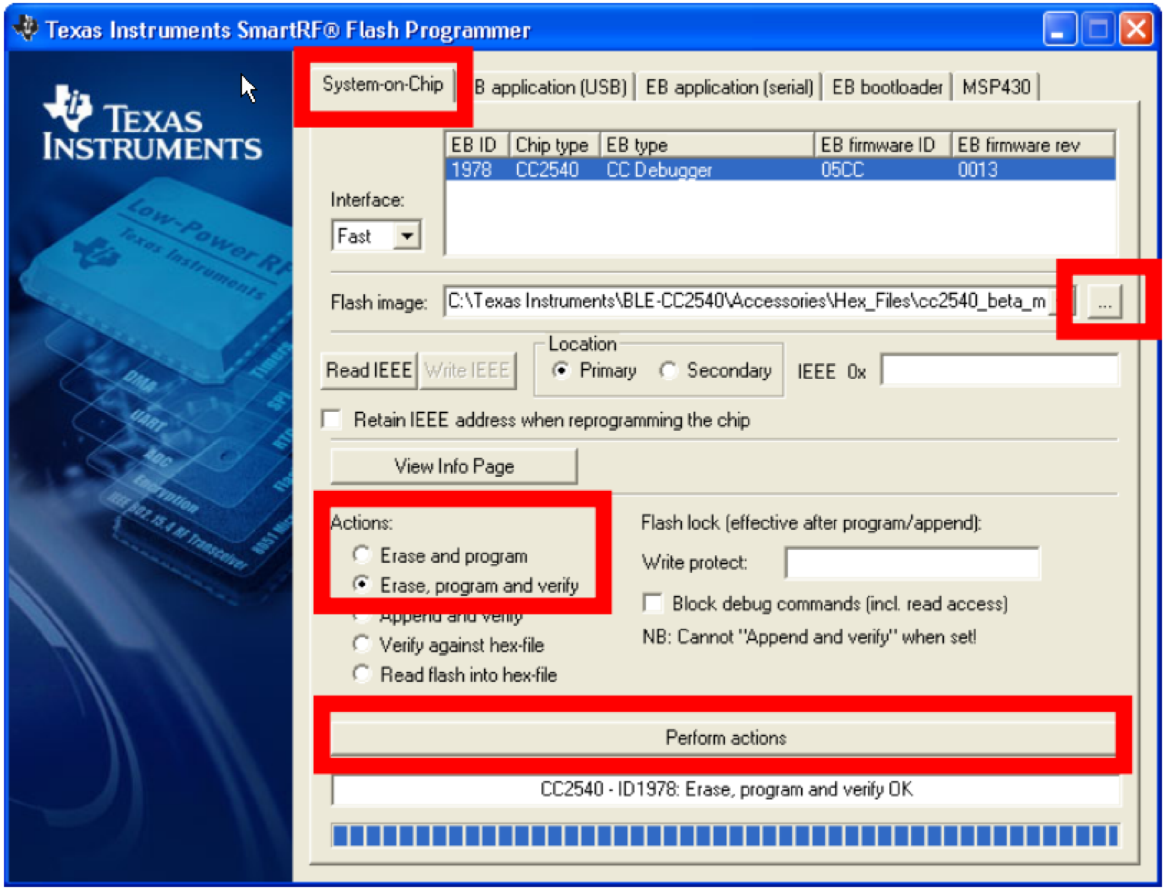
Lab #2.1

1. Connect USB Dongle and CC Debugger as shown
2. CC debugger light should be green.



RED STRIPE TO PIN 1

Lab #2.2



EB ID	Chip type	EB type	EB firmware ID	EB firmware rev
1978	CC2540	CC Debugger	05CC	0013

Interface: Fast

Flash image: C:\Texas Instruments\BLE-CC2540\Accessories\Hex_Files\cc2540_beta_m

Location: ● Primary ○ Secondary

IEEE 0x

□ Retain IEEE address when reprogramming the chip

Actions:
○ Erase and program
● Erase, program and verify
○ Append and verify
○ Verify against hex-file
○ Read flash into hex-file

Flash lock (effective after program/append):
Write protect:
□ Block debug commands (incl. read access)
NB: Cannot "Append and verify" when set!

CC2540 - ID1978: Erase, program and verify OK

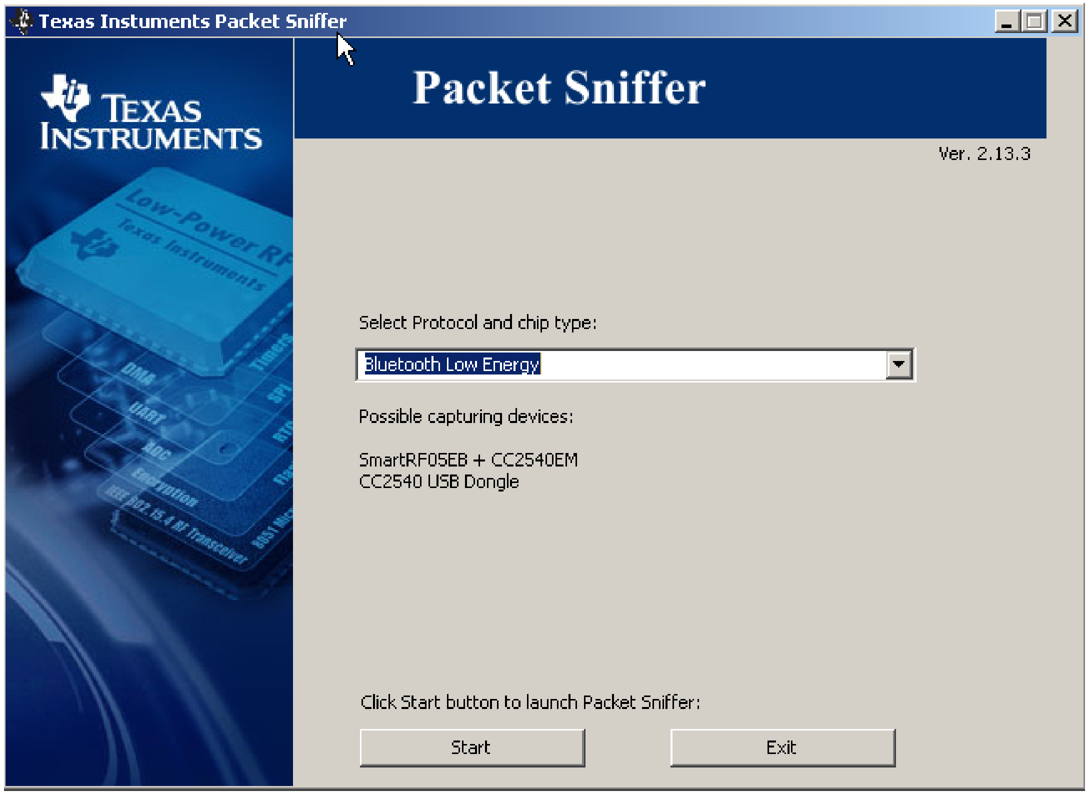
1. Use SmartRF flash programmer to download .hex file

C:\Program Files\Texas Instruments\Packet Sniffer\General\Firmware\sniffer_fw_cc2540_usb.hex

* USB Dongle LED should turn green

Lab #2.3

1. Start Packet Sniffer application
2. Select Bluetooth Low Energy



Ver. 2.13.3

Select Protocol and chip type:

Bluetooth Low Energy

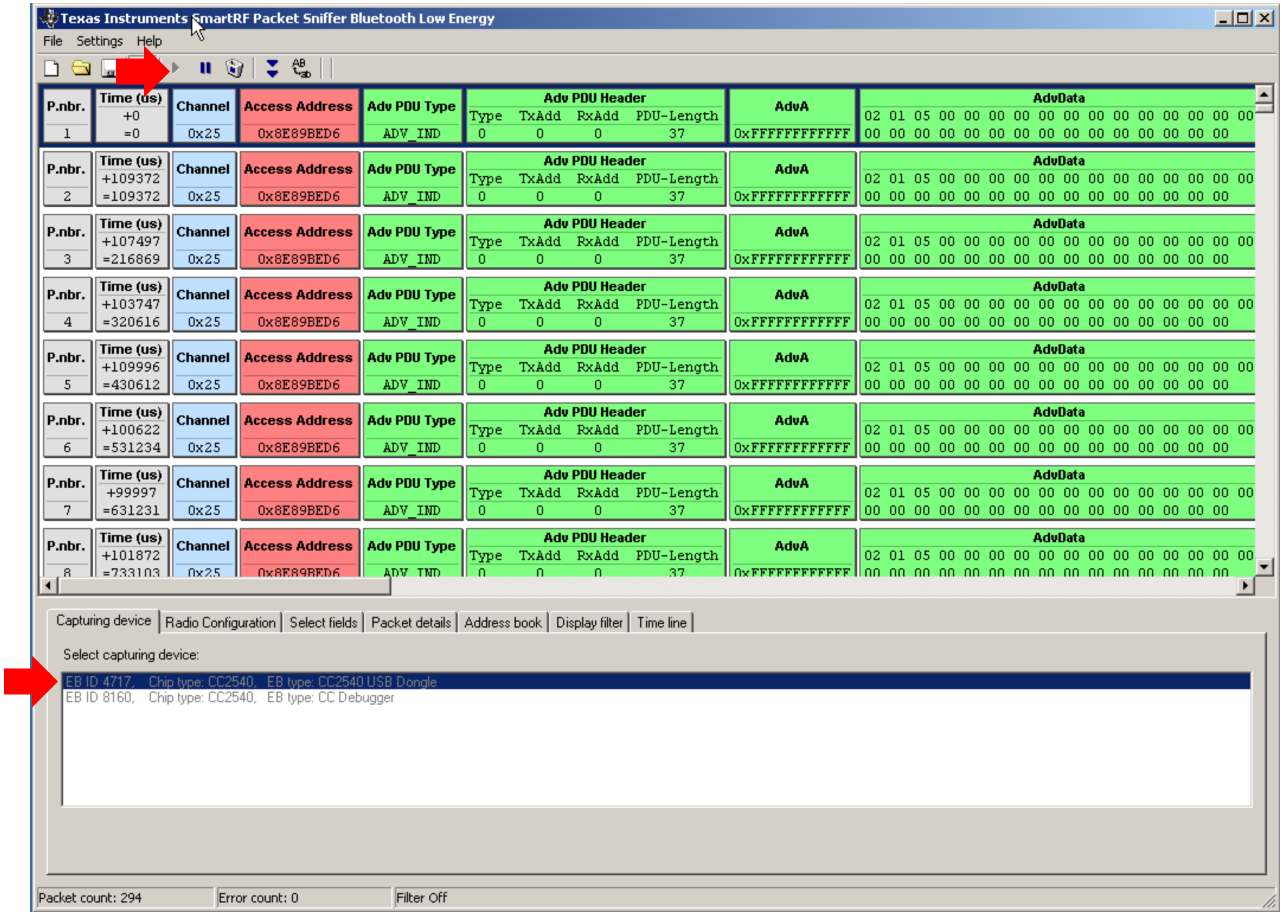
Possible capturing devices:

SmartRF05EB + CC2540EM
CC2540 USB Dongle

Click Start button to launch Packet Sniffer:

Lab #2.4

1. Select CC2540USB Dongle
2. Press play
3. Press button on keyfob to start advertising



P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData
					Type	TxAdd	RxAdd	PDU-Length		
1	+0 =0	0x25	0x8E89BED6	ADV_IND	0	0	0	37	0xFFFFFFFFFFFF	02 01 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2	+109372 =109372	0x25	0x8E89BED6	ADV_IND	0	0	0	37	0xFFFFFFFFFFFF	02 01 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
3	+107497 =216869	0x25	0x8E89BED6	ADV_IND	0	0	0	37	0xFFFFFFFFFFFF	02 01 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4	+103747 =320616	0x25	0x8E89BED6	ADV_IND	0	0	0	37	0xFFFFFFFFFFFF	02 01 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5	+109996 =430612	0x25	0x8E89BED6	ADV_IND	0	0	0	37	0xFFFFFFFFFFFF	02 01 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
6	+100622 =531234	0x25	0x8E89BED6	ADV_IND	0	0	0	37	0xFFFFFFFFFFFF	02 01 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7	+99997 =631231	0x25	0x8E89BED6	ADV_IND	0	0	0	37	0xFFFFFFFFFFFF	02 01 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
8	+101872 =733103	0x25	0x8E89BED6	ADV_IND	0	0	0	37	0xFFFFFFFFFFFF	02 01 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

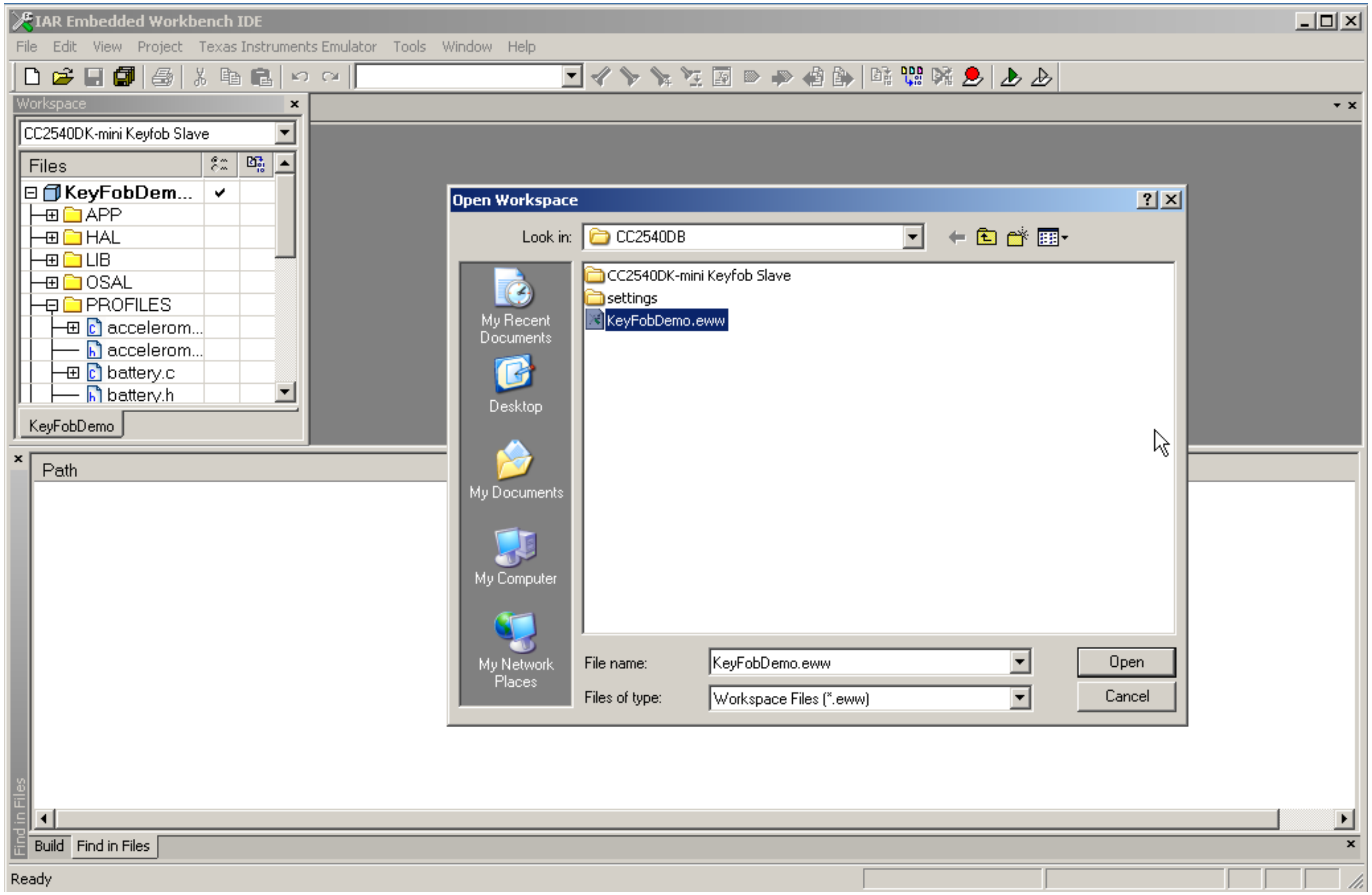
Select capturing device:

EB ID 4717, Chip type: CC2540, EB type: CC2540 USB Dongle
EB ID 8160, Chip type: CC2540, EB type: CC Debugger

Packet count: 294 | Error count: 0 | Filter Off

Lab #3.1 – Accelerometer and IAR

1. Download and unzip project from - http://processors.wiki.ti.com/images/e/ec/Keyfobdemo.zip
2. Extract to C:\Texas Instruments\BLE-CC2540
3. Open workspace C:\Texas Instruments\BLE-CC2540\Projects\ble\KeyFob\CC2540DB\KeyFobDemo.eww
4. Press play button to download and debug
5. Press Go button



Workspace: CC2540DK-mini Keyfob Slave

- KeyFobDem... ✔
  - APP
  - HAL
  - LIB
  - OSAL
  - PROFILES
    - accelerom...
    - accelerom...
    - battery.c
    - battery.h

Look in: CC2540DB

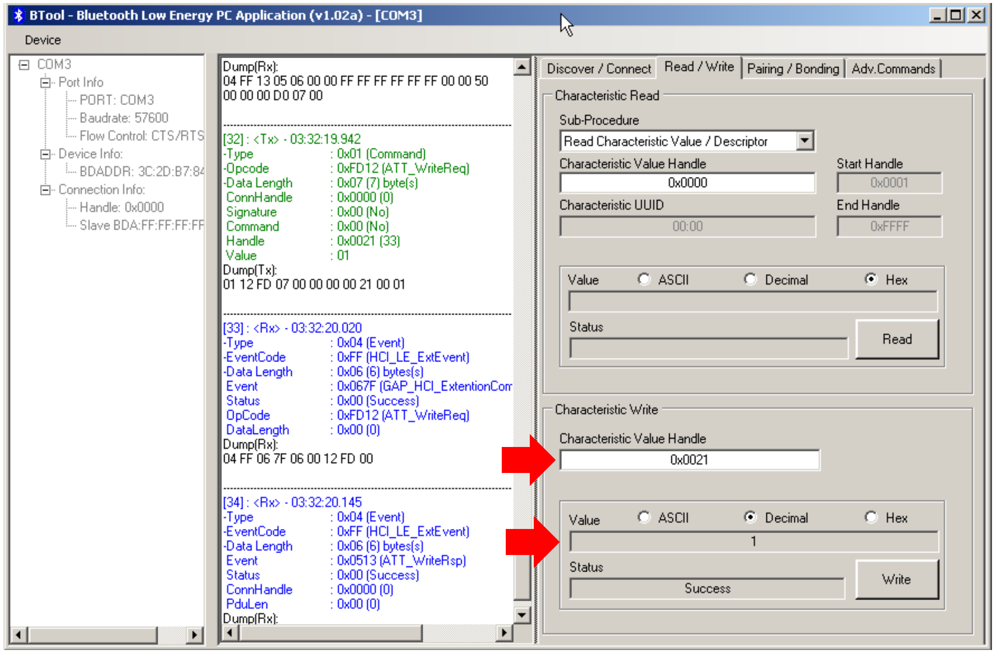
- CC2540DK-mini Keyfob Slave
- settings
- KeyFobDemo.eww

File name: KeyFobDemo.eww

Files of type: Workspace Files (*.eww)

Lab #3.2 – Accelerometer and IAR

1. Start Btool
2. Scan and Connect
3. Write a “1” to 0x0021 to enable the accelerometer
4. Write a “01 00”to 0x0028 to enable the X axis notifications
5. Move the keyfob to generate notifications.



BTool - Bluetooth Low Energy PC Application (v1.02a) - [COM3]

```
COM3
  Port Info
    PORT: COM3
    Baudrate: 57600
    Flow Control: CTS/RTS
  Device Info:
    BDADDR: 3C:2D:B7:84
  Connection Info:
    Handle: 0x0000
    Slave BDA:FF:FF:FF:FF
```

```
Dump(Rx):
04 FF 13 05 06 00 00 FF FF FF FF FF FF 00 00 50
00 00 00 D0 07 00
------------------------------------------------
[32] : <Tx> - 03:32:19.942
-Type              : 0x01 (Command)
-Opcode            : 0xFD12 (ATT_WriteReq)
-Data Length       : 0x07 (7) byte(s)
 ConnHandle        : 0x0000 (0)
 Signature         : 0x00 (No)
 Command           : 0x00 (No)
 Handle            : 0x0021 (33)
 Value             : 01
Dump(Tx):
01 12 FD 07 00 00 00 00 21 00 01
------------------------------------------------
[33] : <Rx> - 03:32:20.020
-Type              : 0x04 (Event)
-EventCode         : 0xFF (HCI_LE_ExtEvent)
-Data Length       : 0x06 (6) bytes(s)
 Event             : 0x067F (GAP_HCI_ExtentionCom
 Status            : 0x00 (Success)
 OpCode            : 0xFD12 (ATT_WriteReq)
 DataLength        : 0x00 (0)
Dump(Rx):
04 FF 06 7F 06 00 12 FD 00
------------------------------------------------
[34] : <Rx> - 03:32:20.145
-Type              : 0x04 (Event)
-EventCode         : 0xFF (HCI_LE_ExtEvent)
-Data Length       : 0x06 (6) bytes(s)
 Event             : 0x0513 (ATT_WriteRsp)
 Status            : 0x00 (Success)
 ConnHandle        : 0x0000 (0)
 PduLen            : 0x00 (0)
Dump(Rx):
```

Characteristic Read

Sub-Procedure: Read Characteristic Value / Descriptor

Characteristic Value Handle: 0x0000

Start Handle: 0x0001

Characteristic UUID: 00:00

End Handle: 0xFFFF

Value ○ ASCII ○ Decimal ● Hex

Status

Characteristic Write

Characteristic Value Handle: 0x0021

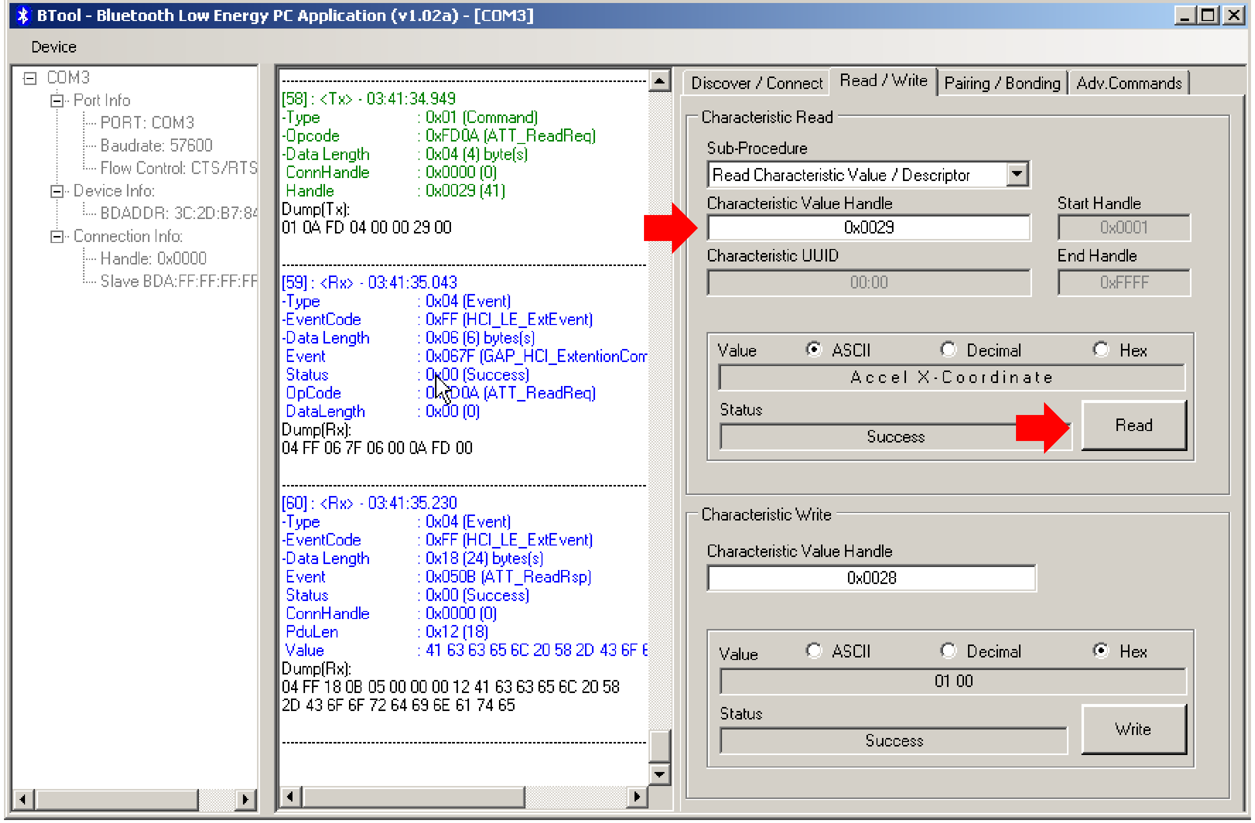
Value ○ ASCII ● Decimal ○ Hex

1

Status: Success

Lab #3.2 – Accelerometer and IAR

1. Enter 0X0029 in handle field
2. Click Read and change to ASCII
3. You should see Accel X-Coordinate
4. Change this in code with IAR.



BTool - Bluetooth Low Energy PC Application (v1.02a) - [COM3]

Device

```
⊟ COM3
  ⊟ Port Info
      PORT: COM3
      Baudrate: 57600
      Flow Control: CTS/RTS
  ⊟ Device Info:
      BDADDR: 3C:2D:B7:84
  ⊟ Connection Info:
      Handle: 0x0000
      Slave BDA:FF:FF:FF:FF
```

```
------------------------------------------------------------
[58] : <Tx> - 03:41:34.949
-Type            : 0x01 (Command)
-Opcode          : 0xFD0A (ATT_ReadReq)
-Data Length     : 0x04 (4) byte(s)
 ConnHandle      : 0x0000 (0)
 Handle          : 0x0029 (41)
Dump(Tx):
01 0A FD 04 00 00 29 00

------------------------------------------------------------
[59] : <Rx> - 03:41:35.043
-Type            : 0x04 (Event)
-EventCode       : 0xFF (HCI_LE_ExtEvent)
-Data Length     : 0x06 (6) bytes(s)
 Event           : 0x067F (GAP_HCI_ExtentionCom
 Status          : 0x00 (Success)
 OpCode          : 0xFD0A (ATT_ReadReq)
 DataLength      : 0x00 (0)
Dump(Rx):
04 FF 06 7F 06 00 0A FD 00

------------------------------------------------------------
[60] : <Rx> - 03:41:35.230
-Type            : 0x04 (Event)
-EventCode       : 0xFF (HCI_LE_ExtEvent)
-Data Length     : 0x18 (24) bytes(s)
 Event           : 0x050B (ATT_ReadRsp)
 Status          : 0x00 (Success)
 ConnHandle      : 0x0000 (0)
 PduLen          : 0x12 (18)
 Value           : 41 63 63 65 6C 20 58 2D 43 6F 6
Dump(Rx):
04 FF 18 0B 05 00 00 00 12 41 63 63 65 6C 20 58
2D 43 6F 6F 72 64 69 6E 61 74 65

------------------------------------------------------------
```

Characteristic Read

Sub-Procedure: Read Characteristic Value / Descriptor

Characteristic Value Handle: 0x0029

Start Handle: 0x0001

Characteristic UUID: 00:00

End Handle: 0xFFFF

Value: ● ASCII ○ Decimal ○ Hex

Accel X-Coordinate

Status: Success

Characteristic Write

Characteristic Value Handle: 0x0028

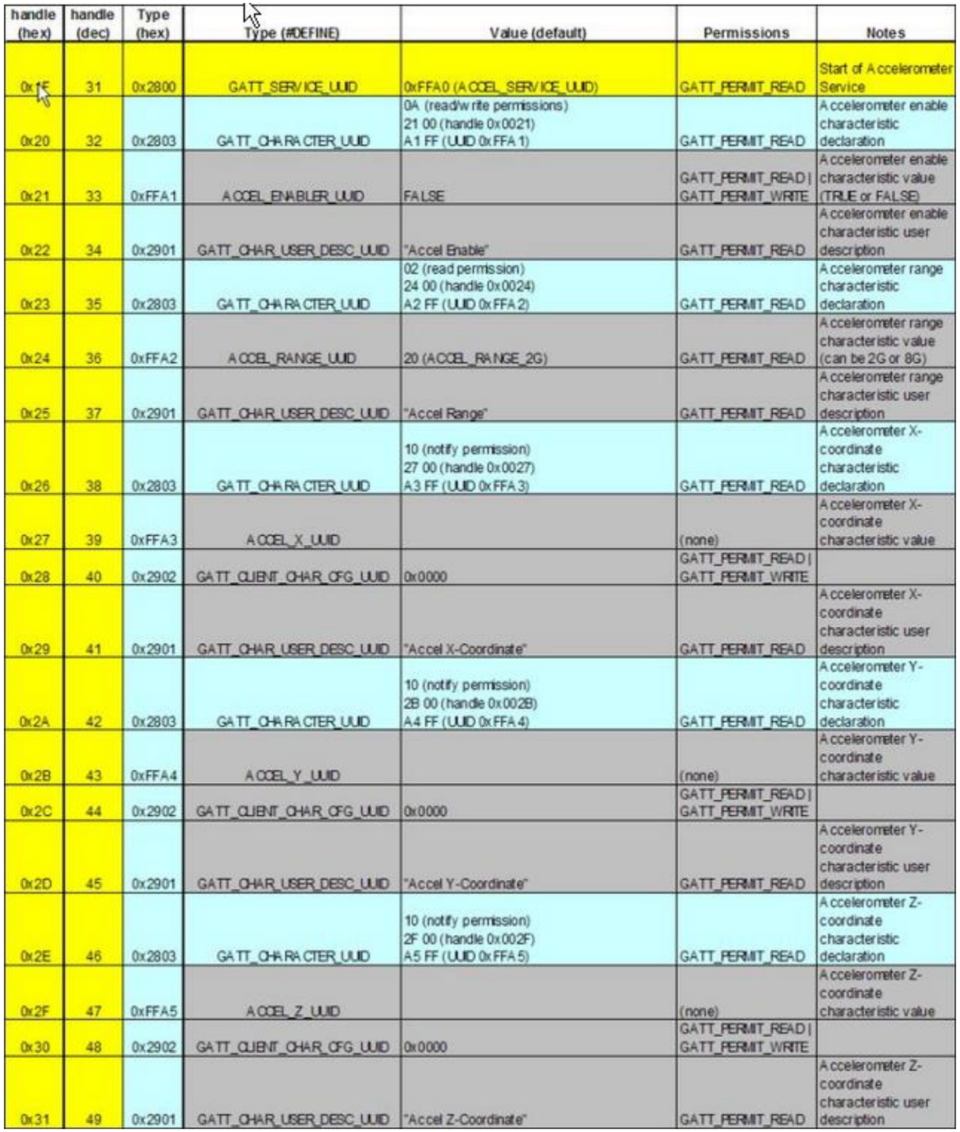
Value: ○ ASCII ○ Decimal ● Hex

01 00

Status: Success

Lab #3.2 – Accelerometer and IAR

Portion of KeyFobDemo application table showing accelerometer GATT table.



handle (hex)	handle (dec)	Type (hex)	Type (#DEFINE)	Value (default)	Permissions	Notes
0x1F	31	0x2800	GATT_SERVICE_UUID	0xFFA0 (ACCEL_SERVICE_UUID)	GATT_PERMIT_READ	Start of Accelerometer Service
0x20	32	0x2803	GATT_CHARACTER_UUID	0A (read/write permissions) 21 00 (handle 0x0021) A1 FF (UUID 0xFFA1)	GATT_PERMIT_READ	Accelerometer enable characteristic declaration
0x21	33	0xFFA1	ACCEL_ENABLER_UUID	FALSE	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Accelerometer enable characteristic value (TRUE or FALSE)
0x22	34	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel Enable"	GATT_PERMIT_READ	Accelerometer enable characteristic user description
0x23	35	0x2803	GATT_CHARACTER_UUID	02 (read permission) 24 00 (handle 0x0024) A2 FF (UUID 0xFFA2)	GATT_PERMIT_READ	Accelerometer range characteristic declaration
0x24	36	0xFFA2	ACCEL_RANGE_UUID	20 (ACCEL_RANGE_2G)	GATT_PERMIT_READ	Accelerometer range characteristic value (can be 2G or 8G)
0x25	37	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel Range"	GATT_PERMIT_READ	Accelerometer range characteristic user description
0x26	38	0x2803	GATT_CHARACTER_UUID	10 (notify permission) 27 00 (handle 0x0027) A3 FF (UUID 0xFFA3)	GATT_PERMIT_READ	Accelerometer X-coordinate characteristic declaration
0x27	39	0xFFA3	ACCEL_X_UUID		(none)	Accelerometer X-coordinate characteristic value
0x28	40	0x2902	GATT_CLIENT_CHAR_CFG_UUID	0x0000	GATT_PERMIT_READ | GATT_PERMIT_WRITE	
0x29	41	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel X-Coordinate"	GATT_PERMIT_READ	Accelerometer X-coordinate characteristic user description
0x2A	42	0x2803	GATT_CHARACTER_UUID	10 (notify permission) 2B 00 (handle 0x002B) A4 FF (UUID 0xFFA4)	GATT_PERMIT_READ	Accelerometer Y-coordinate characteristic declaration
0x2B	43	0xFFA4	ACCEL_Y_UUID		(none)	Accelerometer Y-coordinate characteristic value
0x2C	44	0x2902	GATT_CLIENT_CHAR_CFG_UUID	0x0000	GATT_PERMIT_READ | GATT_PERMIT_WRITE	
0x2D	45	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel Y-Coordinate"	GATT_PERMIT_READ	Accelerometer Y-coordinate characteristic user description
0x2E	46	0x2803	GATT_CHARACTER_UUID	10 (notify permission) 2F 00 (handle 0x002F) A5 FF (UUID 0xFFA5)	GATT_PERMIT_READ	Accelerometer Z-coordinate characteristic declaration
0x2F	47	0xFFA5	ACCEL_Z_UUID		(none)	Accelerometer Z-coordinate characteristic value
0x30	48	0x2902	GATT_CLIENT_CHAR_CFG_UUID	0x0000	GATT_PERMIT_READ | GATT_PERMIT_WRITE	
0x31	49	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel Z-Coordinate"	GATT_PERMIT_READ	Accelerometer Z-coordinate characteristic user description

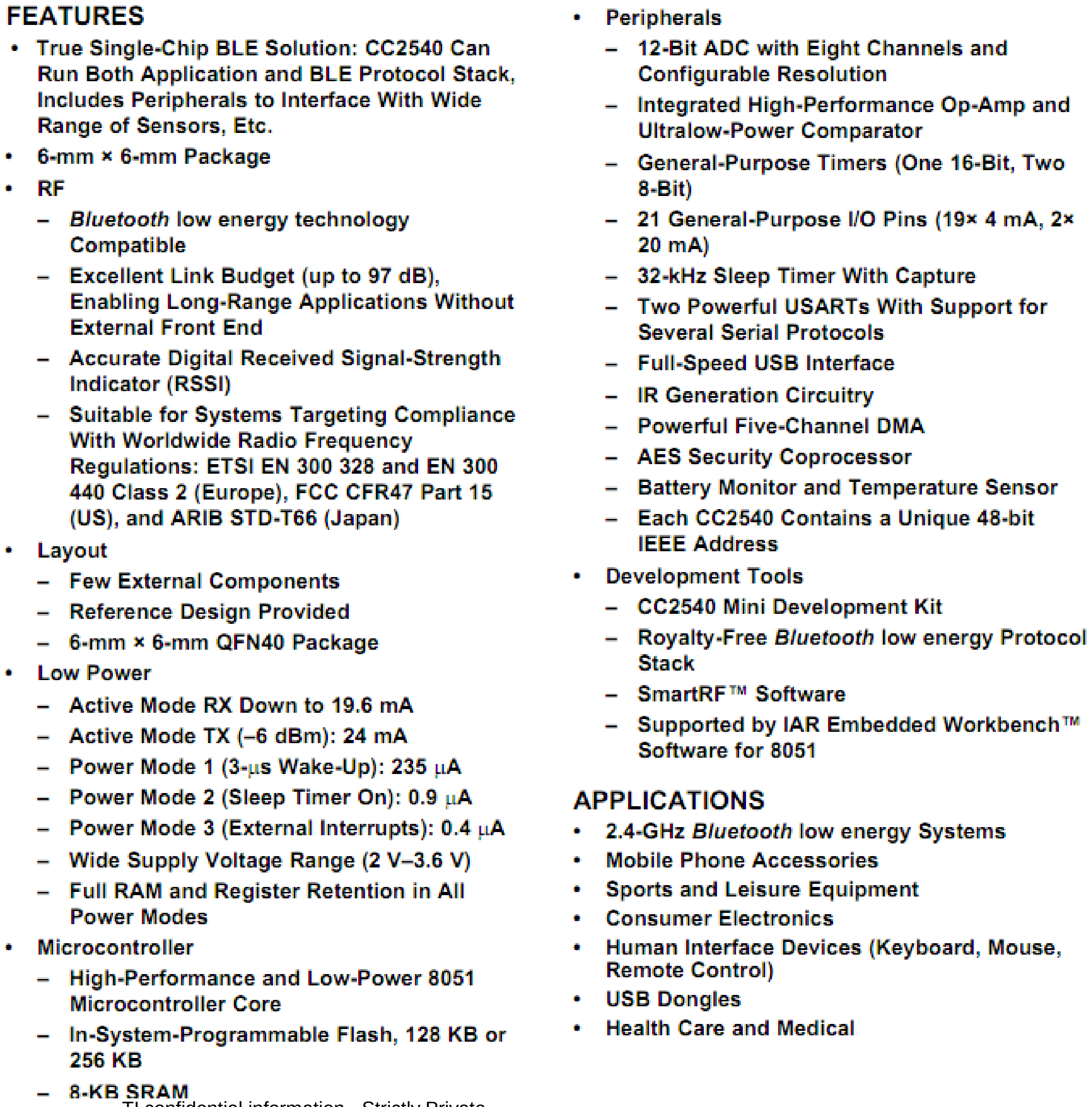
Links

Description	Link
TI Bluetooth, overview, link to Dual mode, data sheets	www.ti.com/bluetoothlowenergy
BLE Stack and tools	www.ti.com/blestack
CC2540 Datasheets, application notes	www.ti.com/cc2540
Hardware sharepoint	http://srvoswod34.norway.design.ti.com/wiki/CC2540_Project
LPRF Wiki Page – Keyfobdemo source	http://processors.wiki.ti.com/index.php/Category:LPRF
SmartRF Flash Programmer	http://focus.ti.com/docs/toolsw/folders/print/flash-programmer.html
Bluetooth SIG	http://www.bluetooth.com/English/Products/Pages/low_energy.aspx

Documents

Description	Link
Quick Start Guide	http://focus.ti.com/lit/ml/swru272/swru272.pdf
Mini Kit User Guide	http://focus.ti.com/lit/ug/swru270a/swru270a.pdf
Software Development Guide	http://www.ti.com/litv/pdf/swru271
CC2540 User Guide	http://focus.ti.com/lit/ug/swru191b/swru191b.pdf

Data Sheet



FEATURES

- True Single-Chip BLE Solution: CC2540 Can Run Both Application and BLE Protocol Stack, Includes Peripherals to Interface With Wide Range of Sensors, Etc.
- 6-mm × 6-mm Package
- RF
  - *Bluetooth* low energy technology Compatible
  - Excellent Link Budget (up to 97 dB), Enabling Long-Range Applications Without External Front End
  - Accurate Digital Received Signal-Strength Indicator (RSSI)
  - Suitable for Systems Targeting Compliance With Worldwide Radio Frequency Regulations: ETSI EN 300 328 and EN 300 440 Class 2 (Europe), FCC CFR47 Part 15 (US), and ARIB STD-T66 (Japan)
- Layout
  - Few External Components
  - Reference Design Provided
  - 6-mm × 6-mm QFN40 Package
- Low Power
  - Active Mode RX Down to 19.6 mA
  - Active Mode TX (–6 dBm): 24 mA
  - Power Mode 1 (3-µs Wake-Up): 235 µA
  - Power Mode 2 (Sleep Timer On): 0.9 µA
  - Power Mode 3 (External Interrupts): 0.4 µA
  - Wide Supply Voltage Range (2 V–3.6 V)
  - Full RAM and Register Retention in All Power Modes
- Microcontroller
  - High-Performance and Low-Power 8051 Microcontroller Core
  - In-System-Programmable Flash, 128 KB or 256 KB
  - 8-KB SRAM
- Peripherals
  - 12-Bit ADC with Eight Channels and Configurable Resolution
  - Integrated High-Performance Op-Amp and Ultralow-Power Comparator
  - General-Purpose Timers (One 16-Bit, Two 8-Bit)
  - 21 General-Purpose I/O Pins (19× 4 mA, 2× 20 mA)
  - 32-kHz Sleep Timer With Capture
  - Two Powerful USARTs With Support for Several Serial Protocols
  - Full-Speed USB Interface
  - IR Generation Circuitry
  - Powerful Five-Channel DMA
  - AES Security Coprocessor
  - Battery Monitor and Temperature Sensor
  - Each CC2540 Contains a Unique 48-bit IEEE Address
- Development Tools
  - CC2540 Mini Development Kit
  - Royalty-Free *Bluetooth* low energy Protocol Stack
  - SmartRF™ Software
  - Supported by IAR Embedded Workbench™ Software for 8051

APPLICATIONS

- 2.4-GHz *Bluetooth* low energy Systems
- Mobile Phone Accessories
- Sports and Leisure Equipment
- Consumer Electronics
- Human Interface Devices (Keyboard, Mouse, Remote Control)
- USB Dongles
- Health Care and Medical

TI confidential information - Strictly Private