

TI *Bluetooth*[®] low energy

Training, October 2013

Greg Stewart

gstewart@ti.com

Introduction

TI *Bluetooth* low energy benefits – CC2541

Cost Effective

- ✓ Highly integrated solution
 - Single-chip and WNP solution
 - Flash-based
- ✓ Low cost HW design
 - 6x6mm QFN package
 - 2 layers layout
- ✓ Quick software development
 - SW examples for all adopted profiles
 - iOS App source code

Flexibility

- ✓ Mature solution
 - TI first to market (2010)
 - >7000kits shipped
 - Leading market share (IMS/ABI)
- ✓ Flexible configuration
 - Central/Peripheral role switching
 - Combo Roles
 - Over-the-air SW update
 - Boot Image Manager (BMI)
- ✓ Support
 - www.ti.com/ble
 - Most active BLE community (www.ti.com/ble-forum)
 - All information publicly available

High Performance

- ✓ Powerful
 - >1 year battery life with CR2032
 - 97dB link budget
- ✓ Ultra-low power
 - Very low-power sleep modes
 - Short transition times between operating modes
 - Ref.design with dc-dc (TPS62730)



TEXAS INSTRUMENTS

TI *Bluetooth* low energy enabled Products

Kensington
Proximity tag



Nokia
Precious Tag



Lifesense
Blood Pressure Monitor



Ruwido/Swisscom
Remote Control



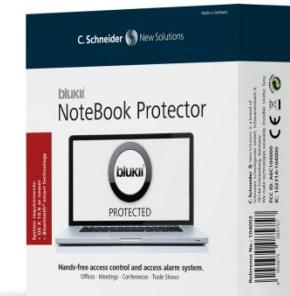
UnderArmor
Heart Rate Monitor



Kwikset
Smart door lock



Schneider
Notebook protector

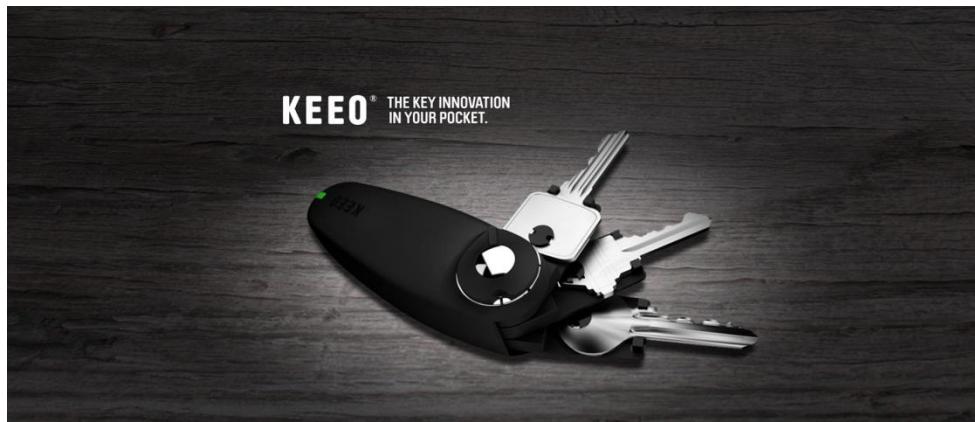


TI *Bluetooth* low energy enabled Products

Smartbiotics
Smart LED Bulbs



KEEO
Smart Key



Misfit Wearable's
Shine – Activity Monitor



TI *Bluetooth* low energy enabled Products

Kinetek



LifeTrak



TetherCell



UniKey



Parrot



BACTrack



TI *Bluetooth* low energy enabled Products

Murata



Polar



ZOMM



Schneider



Bluetrek



Recon



Ace Sensor



TI *Bluetooth* low energy enabled Products

Blast Motion



IDT



Griffin



Citizen



MonBaby



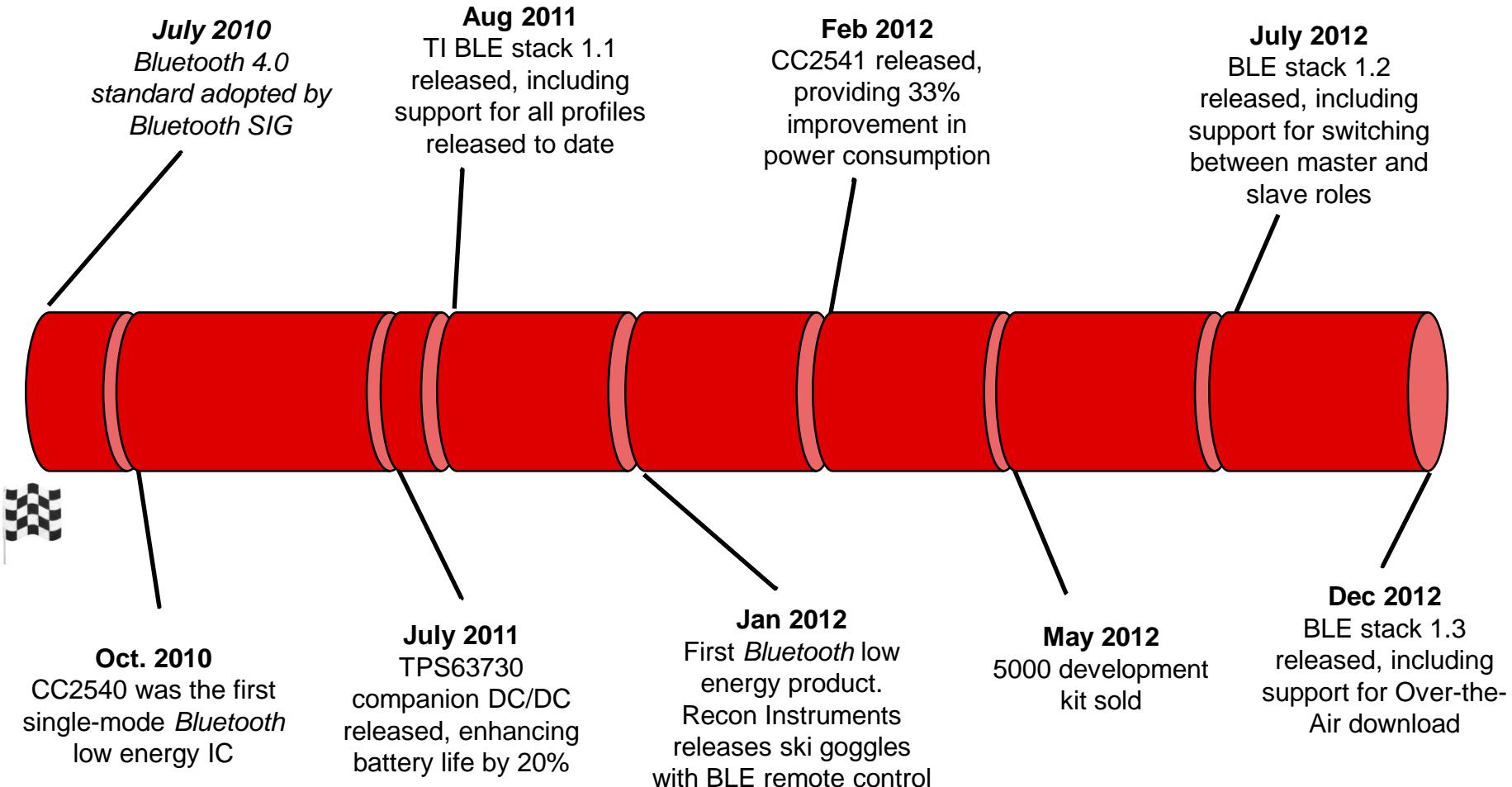
SoundOfMotion



Tom Tom



TI track record



"As shown with its certifications, new profile offerings and updated Bluetooth low energy stack, we are pleased to see TI playing a critical role in delivering Bluetooth v4.0 with low energy capabilities to new places – from health monitoring to fitness equipment, alert notifications and more."

Mike Foley, Ph. D., Executive Director, Bluetooth SIG

TI Hardware

TI *Bluetooth* low energy Hardware



CC2540
Flash SoC
USB



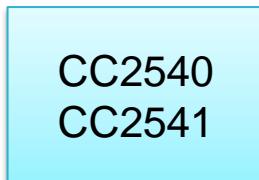
CC2541
Flash SoC
I2C



CC2541S
Flash WNP

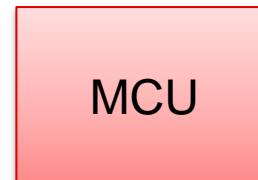
Two ways of using TI *Bluetooth* low energy

One-Chip Solution

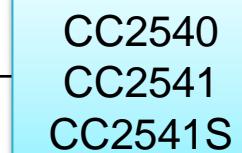


BLE Stack + Application + Profiles

Two-Chip Solution (Network Processor)



Application + Profiles



BLE Stack

TI *Bluetooth* low energy Hardware



CC2540
Flash SoC
USB



CC2541
Flash SoC
I2C



CC2541S
Flash WNP

Development Kits



CC2540DK-MINI
CC2541DK-MINI

The most easiest way of evaluating Bluetooth low energy



BOOST-CC2541S
BoosterPack for
MSP430, Stellaris
and C2000



CC2540DK
CC2541EMK

Advanced kit with several peripheral interfaces and features



CC2541DK-SENSOR

Targiting Smart Phone App developers.
Provides sensor data from 6 sensors



CC2541DK-RC

Operates as a mouse, keyboard and consumer remote using HOGP

Order Kit and Download SDK

- Visit ti.com/ble

Products Applications Tools & Software Support & Community Sample & Buy

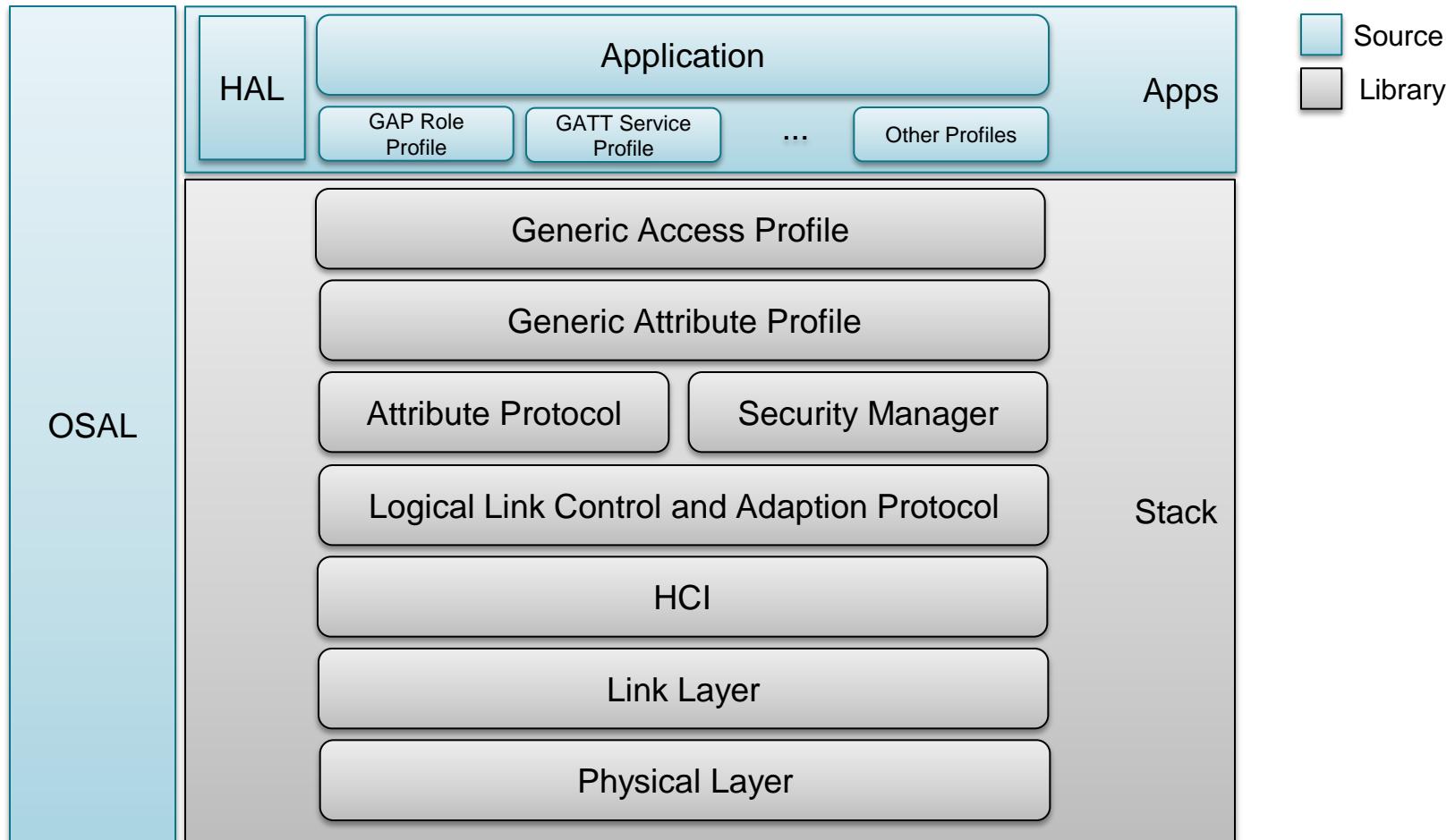
TI Home > Wireless Connectivity > Bluetooth® low energy

Bluetooth® low energy



Software Solution

TI BLE Software Solution



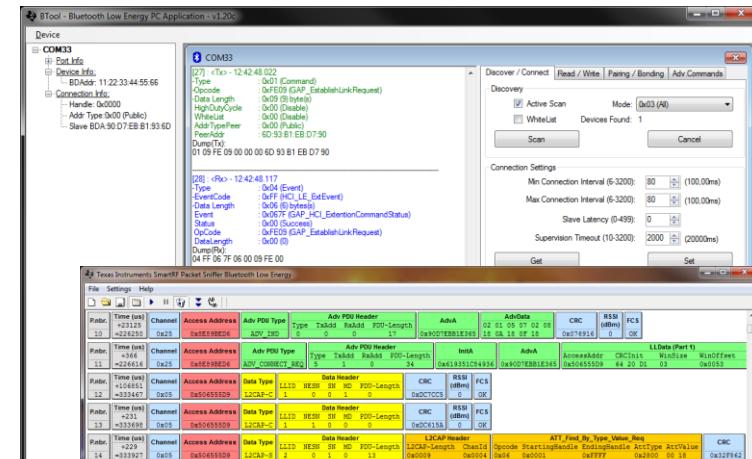
OSAL = Operating System Abstraction Layer (Prioritized task handling loop)

HAL = Hardware Abstraction Layer (Drivers and API for LEDs, Buttons etc)

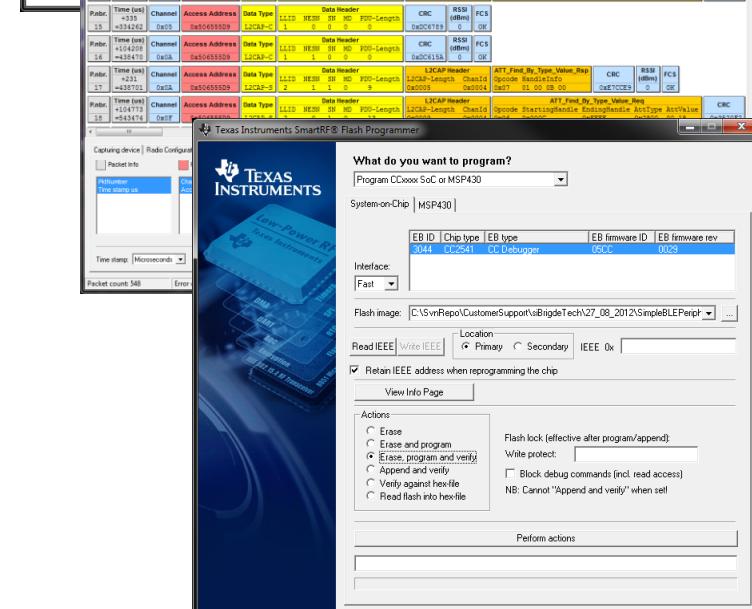
Full API to access all stack functionality in the stack (Library) from the Application and Profiles

TI *Bluetooth* low energy Tools

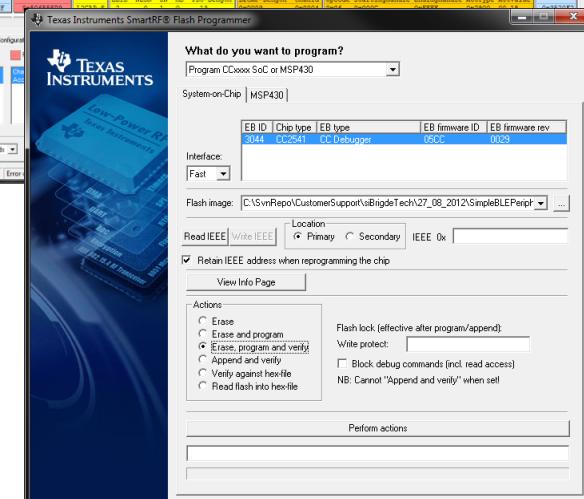
- BTool
 - Run and test *Bluetooth* low energy communication through a CC254x Network Processor



- Sniffer
 - Capture *Bluetooth* low energy communication live with full overview.



- Flash Programmer
 - Program CC254x devices
 - Read and write IEEE addresses



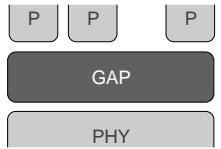
Documentation and Support

- Guides included in the BLE installer
 - TI BLE Software Developers Guide
 - TI BLE Sample Applications Guide
 - TI BLE Vendor Specific HCI Guide
 - TI BLE API Guide
- Online documentation
 - TI BLE Product Page (www.ti.com/ble)
 - TI BLE Wiki (www.ti.com/ble-wiki)
- Online Support
 - TI BLE E2E forum

Data Sheets
User Guides
Reference Designs
Application Notes

Application Examples
Walkthrough
Step by Step guides
FAQ

Bluetooth low energy Protocol Stack



GAP - Roles

- A *Bluetooth low energy device* can operate in four profile roles:

- **Broadcaster**

- An advertiser that is non-connectable
- *Example: Temperature Sensor*



- **Observer**

- Scans for advertisements, but cannot initiate connections
- *Example: Temperature Display*



- **Peripheral**

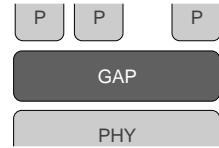
- An advertiser that is connectable
- Operates as a **slave** in a connection
- *Example: Watch*



- **Central**

- Scans for advertisements and initiates connections
- Operates as a **master** in connections.
- *Example: Smart Phone*





GAP – Multiple Roles

- The BLE specification multiple-role configurations:
 - Peripheral and Broadcaster
 - device operates as a slave in a single link layer connection
 - send out non-connectable advertisements
 - Peripheral and Observer:
 - device operates as a slave in a single link layer connection
 - Scan for advertisements without initiating a connection
 - Central and Broadcaster:
 - Device scans for advertisements and initiates connections as a master
 - Broadcast non-connectable advertisements

TI *Bluetooth* low energy Software Examples

- Profile Specific
 - Heart Rate
 - Glucose
 - HID Keyboard
 - Etc.
- "General"
 - SimpleBLEPeripheral
 - SimpleBLECentral
 - SimpleBLEBroadcaster
 - SimpleBLEObserver



Bluetooth SIG adopted Profiles

Cycling Speed and Cadence
Running Speed and Cadence
Blood Glucose
HID over GATT
Proximity
Find me
Health thermometer
Heart rate sensor
Time
Alert Notification
Battery Status

- Certified	(BLEv1.3.2, June '13)
- Certified	(BLEv1.3.2, June '13)
- Certified	(BLEv1.2, February '12)
- Certified	(BLEv1.2, February '12)
- Certified	(BLEv1.1, July '11)

Proprietary Profile

- Uses GATT Architecture for easy implementation
- Proprietary Services uses 128bit UUID
 - Example in Sensor Tag project (www.ti.com/ble-stack)
 - Custom example available at wiki (www.ti.com/ble-wiki)
 - UUIDs can be generated online (www.uuidgenerator.com)
- Use adopted Services (www.bluetooth.org)
 - Faster implementation
 - Ensure interoperability
- TI base UUID
 - F000XXXX-0451-4000-B000-000000000000

Memory Footprint Examples (BLEv1.3)

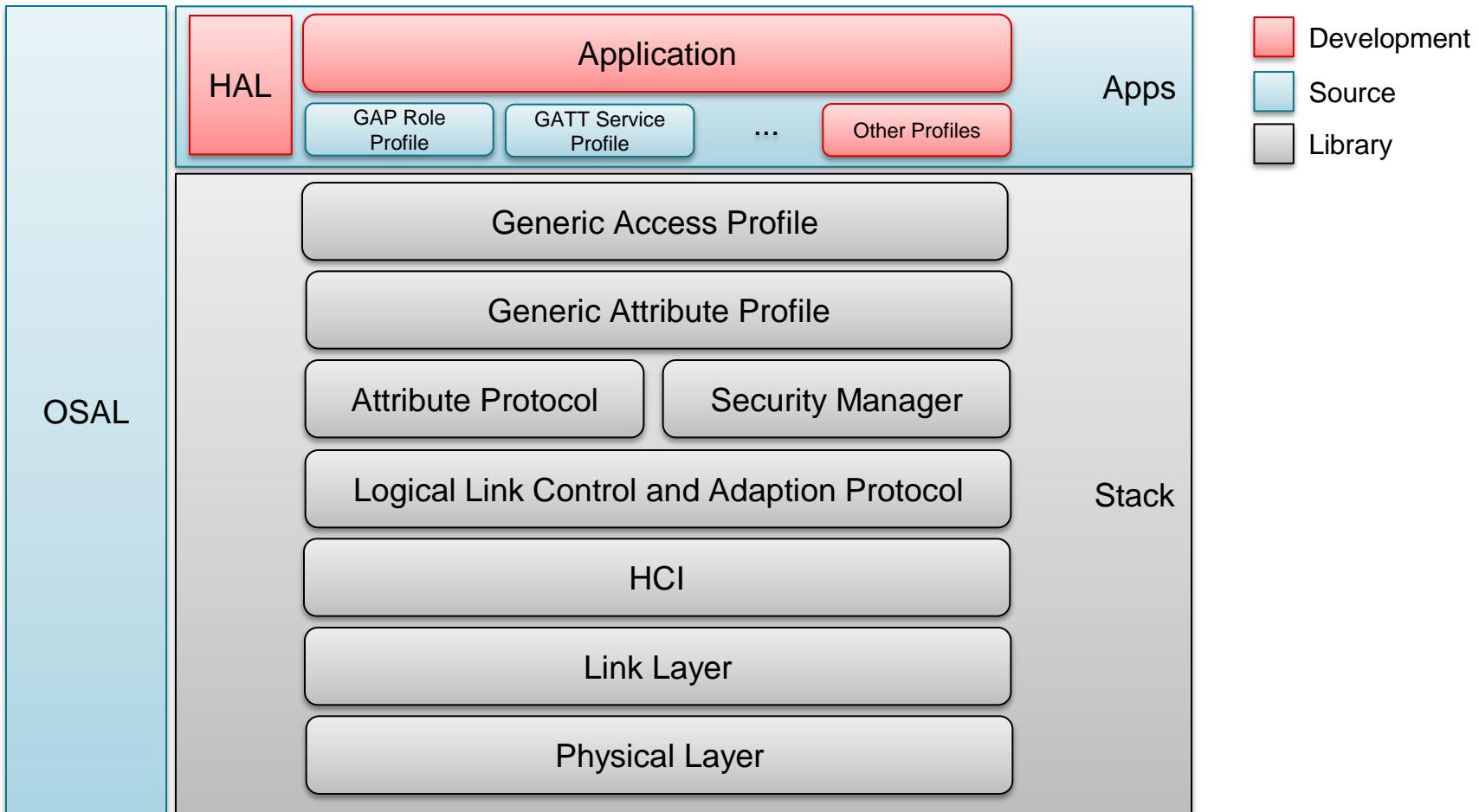
Project	Flash [KBytes]	RAM [KBytes]
SimpleBLEPeripheral	114	6.2
SimpleBLECentral	123	7.2
SimpleBLEObserver	43	3.3
SimpleBLEBroadcaster	39	3.3
KeyFobDemo	116	6.4
Heart Rate	112	6.1
HostTest (Network Processor)	151	7.5

Profiles	Flash [KBytes]	RAM [KBytes]
Device Information Service	1.068	0.176
Heart Rate Service	0.678	0.078
Battery Service	0.846	0.053
Proximity	0.984	0.094

Software Development Kit

- User Requirements
 - An intermediate level of knowledge of the C programming language
- Runs on embedded 8051 (MCU) with limited resources
 - Algorithms and application should be coded efficiently
 - C standard library should not be used
- Concepts of encapsulation and information hiding:
 - GetParameter and SetParameter functions etc.
 - Callbacks

Embedded Software



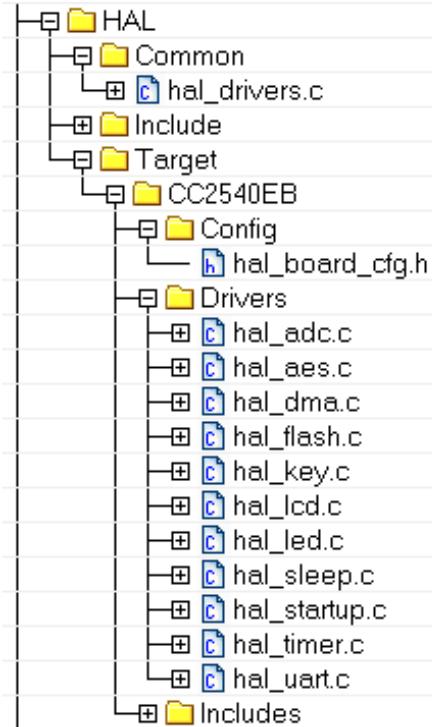
OSAL = Operating System Abstraction Layer (Prioritized task handling loop)

HAL = Hardware Abstraction Layer (Drivers and API for LEDs, Buttons etc)

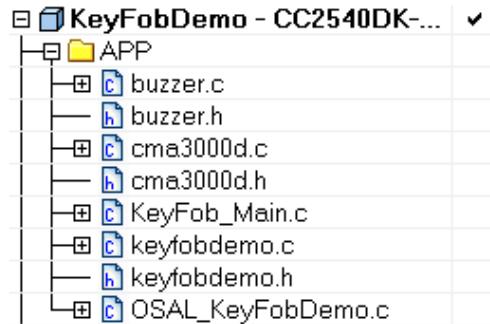
Full API to access all stack functionality in the stack (Library) from the Application and Profiles

Developer's resources

HAL



User Application



Other Profiles



- HAL
 - Some are very specific to our boards/ICs
- APP
 - User application
 - Specific drivers
- PROFILES
 - Adopted (Battery, Proximity)
 - Proprietary (Accelerometer, SimpleKeys)

Operating System Abstraction Layer (OSAL)

- The SW architecture is based around the Operating System Abstraction Layer (OSAL)
- The OSAL is a control loop that allows SW to setup execution of events
- Each subsystem of the software runs as an OSAL task, and has a unique task identifier (ID)
- Low task ID → Task has higher priority
- Typical Project has 12 OSAL tasks

ID	Task
0	Link Layer
1	HAL
2	HCI
3	OSAL Callback Timer
4	L2CAP
5	GAP
6	GATT
7	SM
8	GAP Role
9	GAP Bond Manager
10	GATT Server
11	Application

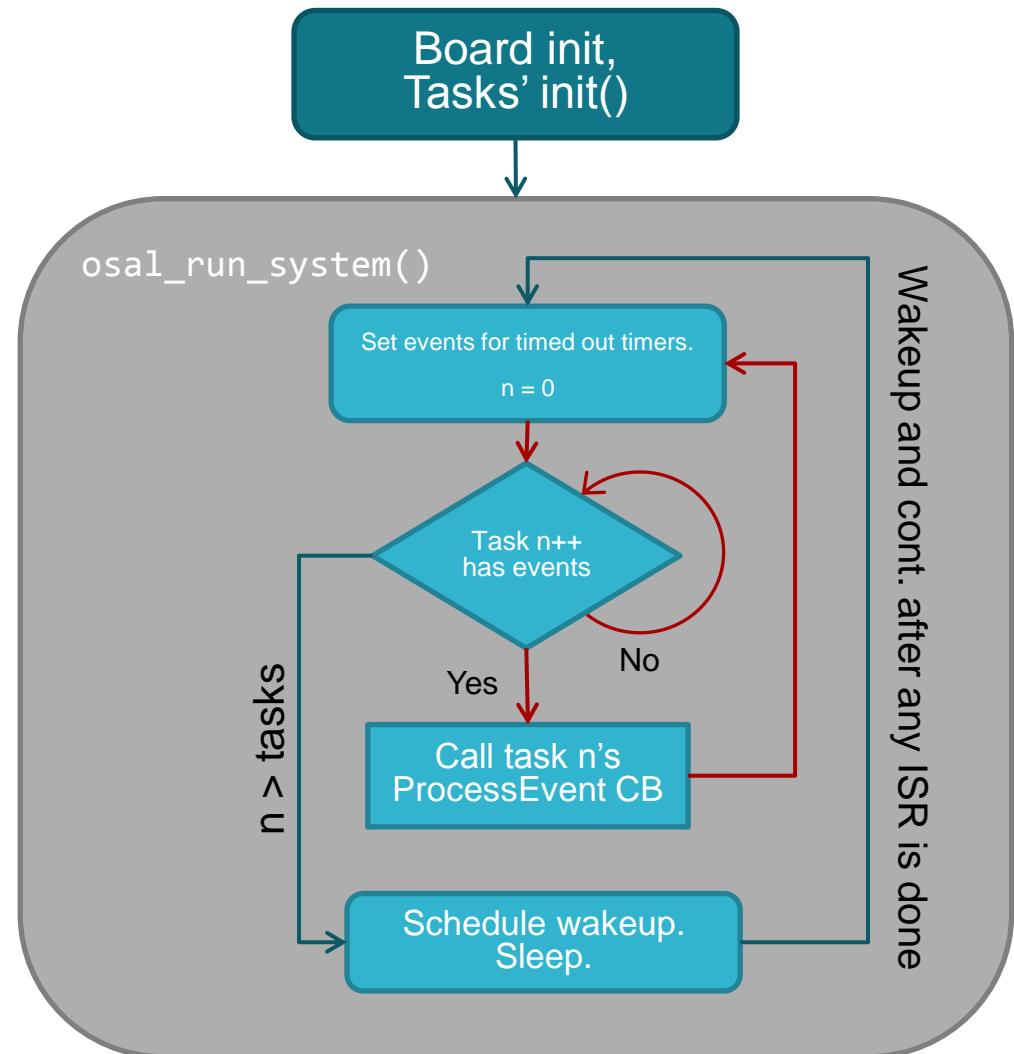
OSAL – Files and Key API's

- Key Files:
 - osal.c – API's for OSAL
 - osal.h – OSAL API declarations
- Key API's:

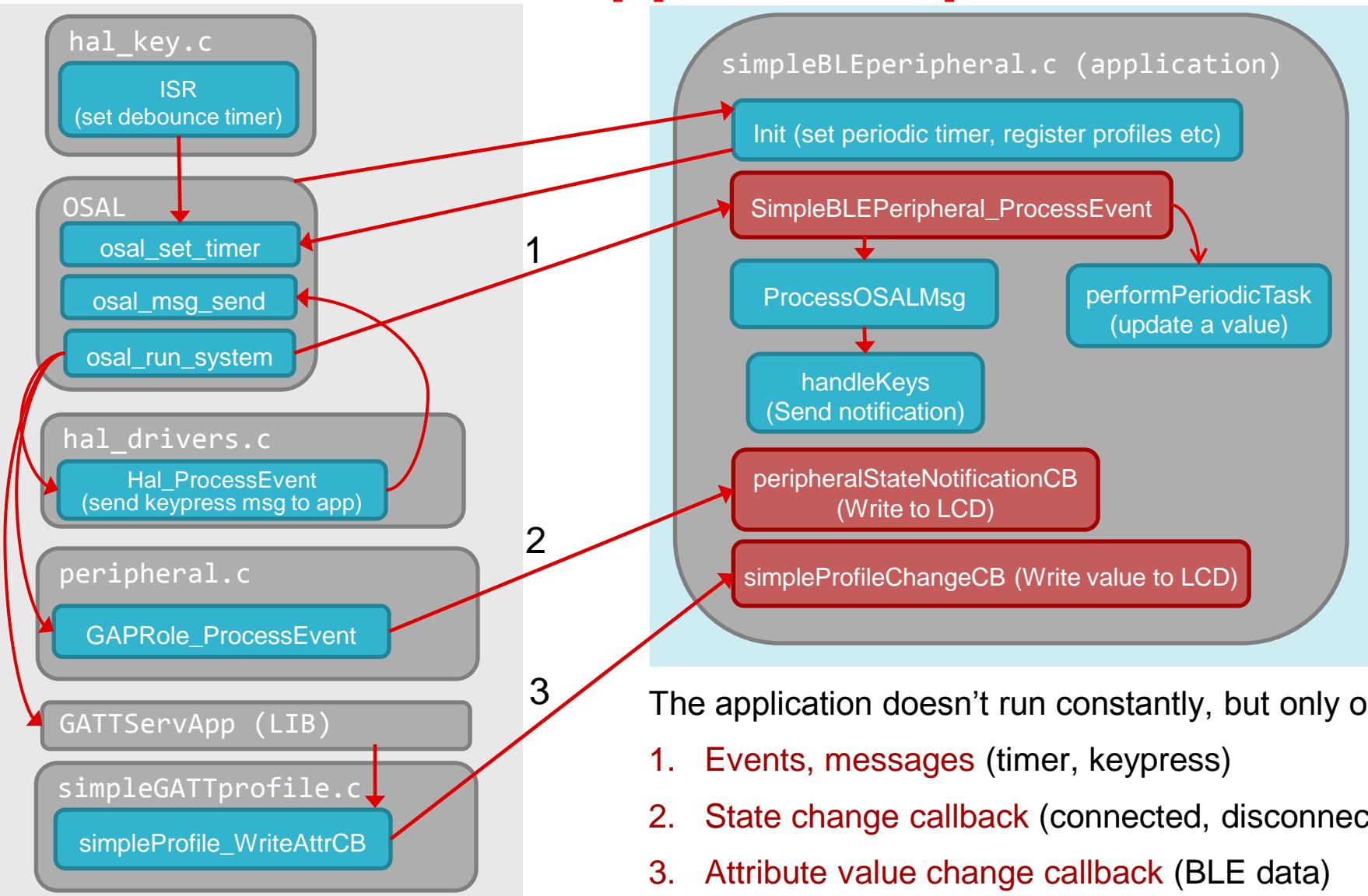
– osal_init_system	Initializes OSAL
– osal_start_system	Starts the OSAL main loop
– osal_set_event	Sets an OSAL event for a task
– osal_start_timerEx	Sets an OSAL event for a task at a scheduled moment in time
– osal_stop_timerEx	Cancels an existing OSAL event
– osal_msg_allocate	Dynamically allocates 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
- Additional information on the OSAL can be found in the OSAL API guide:
<C:\Texas Instruments\BLE-CC2540\Documents\osal\OSAL API.pdf>

Execution flow: OSAL's point of view

- OSAL is a prioritized work-queue
 - If two tasks have events, handle highest first
- The stack is event-driven
 - Act on the environment, then
 - Sleep until something happens
- Lower stack layers communicate via OSAL messages and events
 - Then call functions in the user application:
 - On attribute writes/reads
 - On connection loss/establishment



Execution flow: Application point of view

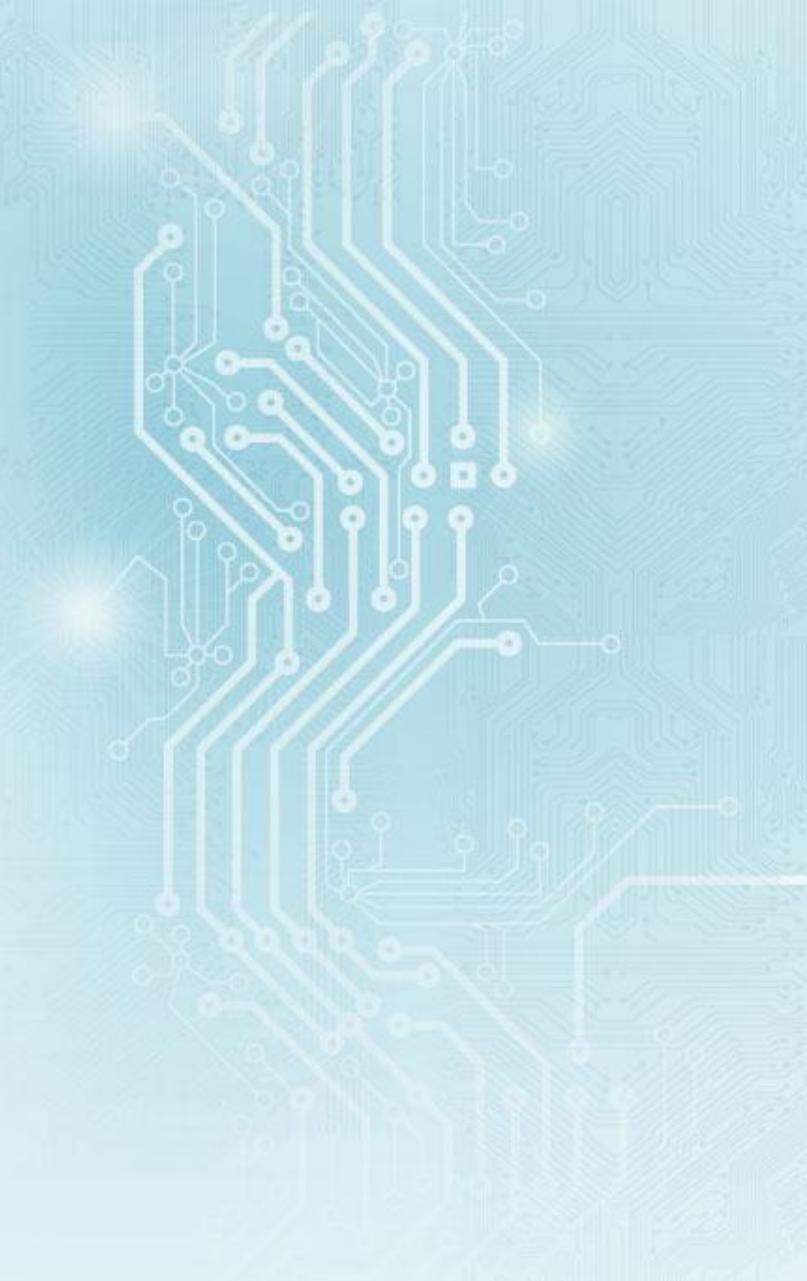


Hardware Abstraction Layer

- The Hardware Abstraction Layer (HAL) provides an application programming interface to hardware-related functions such as
 - ADC
 - UART
 - SPI
 - Flash
 - Timers
 - Sleep
 - Keys
 - LED
 - LCD
- Additional details on HAL functions can be found in the HAL API Guide:
<C:\Texas Instruments\BLE-CC2540\Documents\hal\HAL Driver API.pdf>

OAD Profile

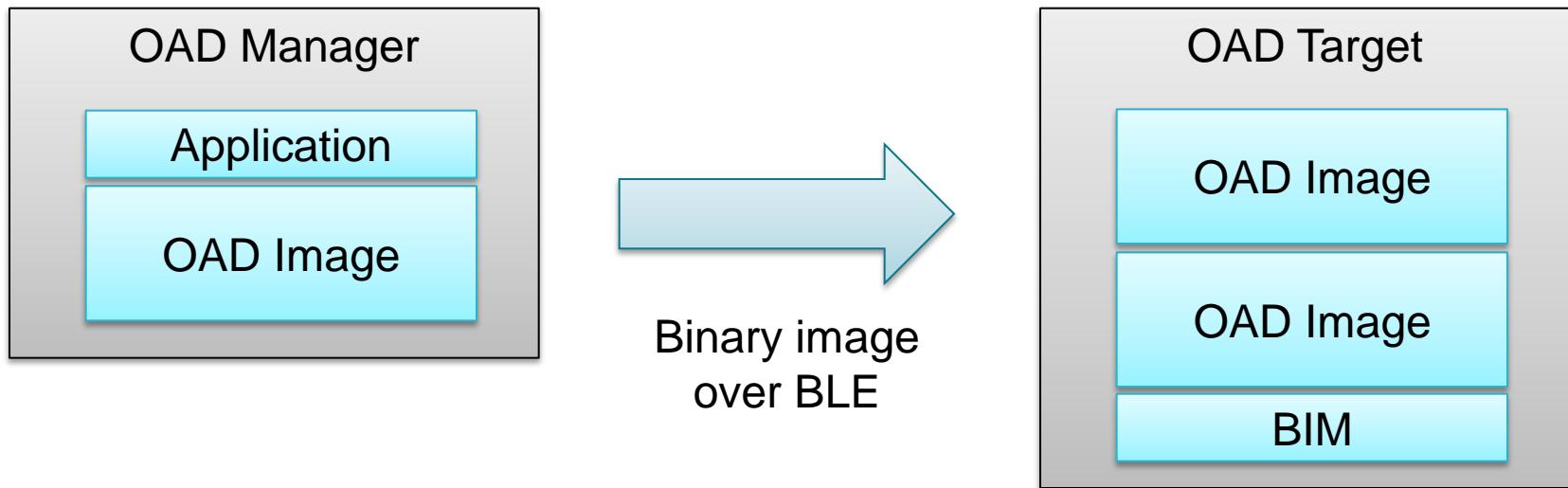
Over the Air Download



Introduction – OAD Profile

- What is OAD?
 - OAD is a feature that provides a solution for updating code in released products without the cost of physical access (i.e. programming header).
 - OAD is a client-server mechanism in which one device acts as the OAD image server (OAD manager) and the other device is the OAD image client (OAD target).
- What is a proprietary Profile?
 - Uses GATT Architecture for easy implementation
 - Proprietary Services uses 128bit UUID
 - Example in Sensor Tag project (www.ti.com/ble-stack)
 - Custom example available at wiki (www.ti.com/ble-wiki)
 - UUIDs can be generated online (www.uuidgenerator.com)
 - TI base UUID
 - F000XXXX-0451-4000-B000-000000000000

OAD System Overview



OAD Manager - Examples

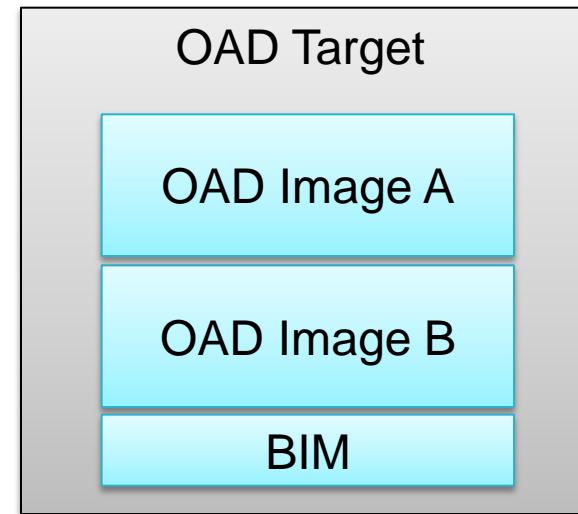


- Acts as a Central Client

- OADManager
 - Use SBL Tool towards a SmartRF05+EM that runs OADManager
- MultiTool
 - Runs on an iOS Device
- Device Monitor
 - Runs on PC and utilizes CC2540USB Dongle

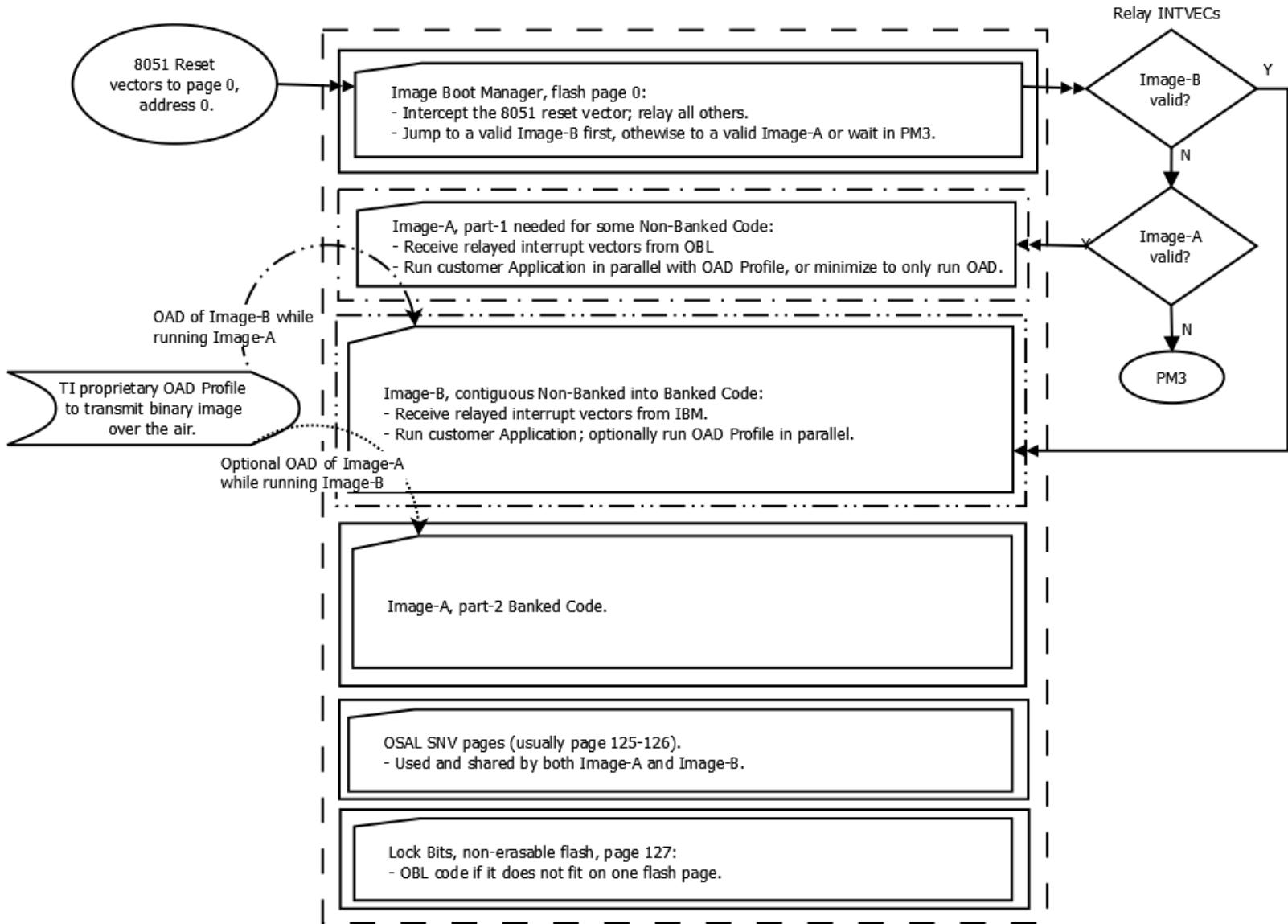
OAD Target - Examples

- Acts as a GATT Server



- OAD is a GATT Profile.
- Projects examples implementing OAD Target
 - SensorTag (CC2541)
 - A & B images
 - SimpleBLEPeripheral
 - A & B Images
 - Encrypted A & B Images

Memory Architecture



How to build OAD System

- Target Requirements
 - Target must be setup with BIM (Boot Image Manager)
 - ..\BLE-CC254x-1.3.1\Projects\ble\util\BIM\cc254x\BIM.eww
 - Target must be setup with Image A with OAD Profile
- Choose a OAD Manager platform , ex. iPhone/iPad or Device Monitor
- Build an Image B with OAD Profile.

How to build add OAD Profile

- Target Setup
 - Create a new configuration
 - ex. CC2541-OAD-ImgA
 - Add PROFILE files
 - oad_target.c/h
 - oad.h
 - Add definitions:
 - FEATURE_OAD
 - OAD_KEEP_NV_PAGES
 - FEATURE_OAD_BIM
 - HAL_IMAGE_A
 - Include path to the OAD Profile
 - Add Build Actions
 - "\$PROJ_DIR\$\..\..\common\CC2540\cc254x_ubl_pp.bat"
 - "\$PROJ_DIR\$" "ProdUBL"
 - "\$PROJ_DIR\$\CC2541-OAD-ImgA\Exe\ProjectName"
- In Application
 - Add OAD_AddService()
 - Add includes for
 - "oad.h"
 - "oad_target.h"
- Linker configuration file
 - \$PROJ_DIR\$\..\..\common\CC2541\cc254x_f256_imgA.xcl

OAD Target Profile

```
typedef struct {  
    uint16 crc1;      // CRC-shadow must be 0xFFFF.  
    uint16 ver;       // User-defined Image Version Number  
    uint16 len;        // Image length in 4-byte blocks  
    uint8 uid[4];     // User-defined Image Identification bytes.  
    uint8 res[4];     // Reserved space for future use.  
} img_hdr_t;
```

00:00:00:7C:41:41:41
Version Length User Defined ID

52	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	OAD Service
53	0x2803	GATT Characteristic Declaration	1C:36:00:00:00:00:00:00:00:B0:00:...	OAD Image Identify
54	F000F...	OAD Image Identify	00:00:00:7C:41:41:41:41	
55	0x2902	Client Characteristic Configuration		Write "01:00" to enable notifications, "00:00" to disable
	56	0x2901	Characteristic User Description	
57	0x2803	GATT Characteristic Declaration	1C:3A:00:00:00:00:00:00:00:B0:00:...	OAD Image Block
58	F000F...	OAD Image Block		
59	0x2902	Client Characteristic Configuration		Write "01:00" to enable notifications, "00:00" to disable
60	0x2901	Characteristic User Description		

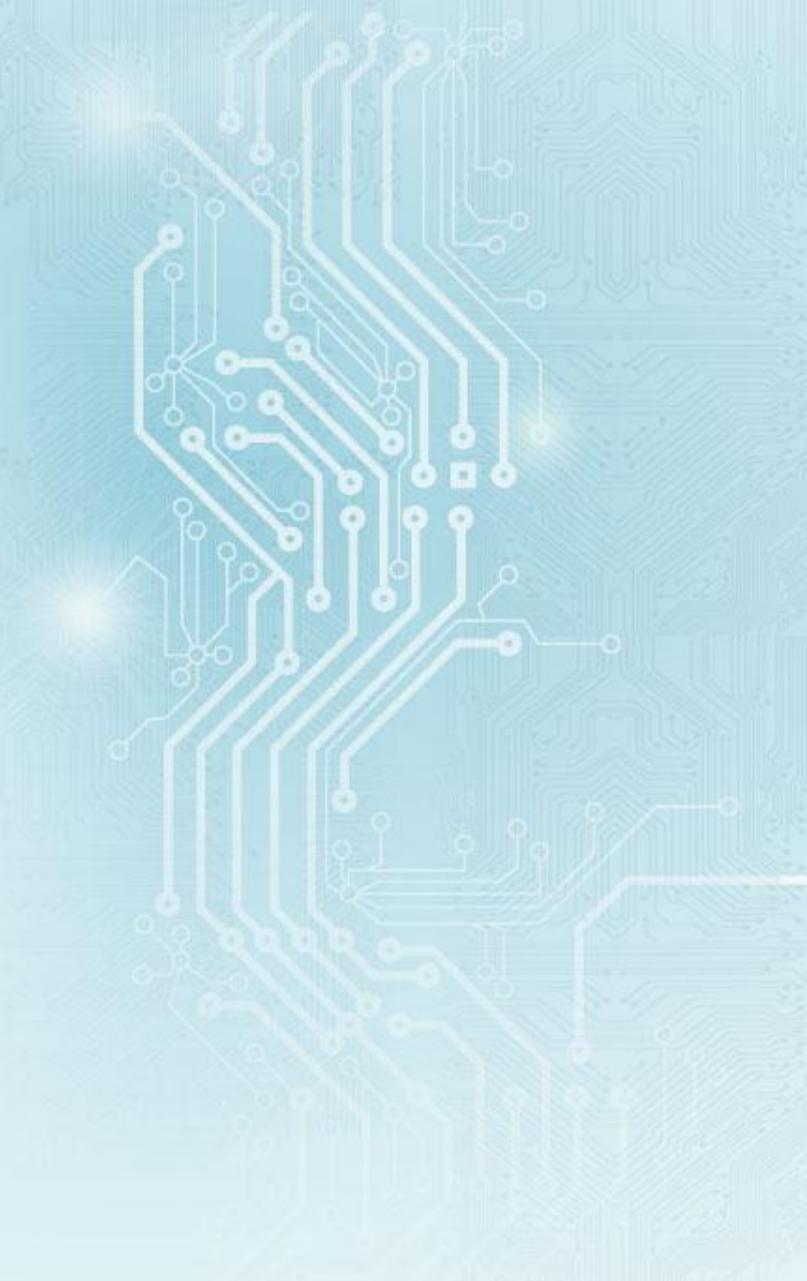
16 Bytes Image Block

OAD Resources

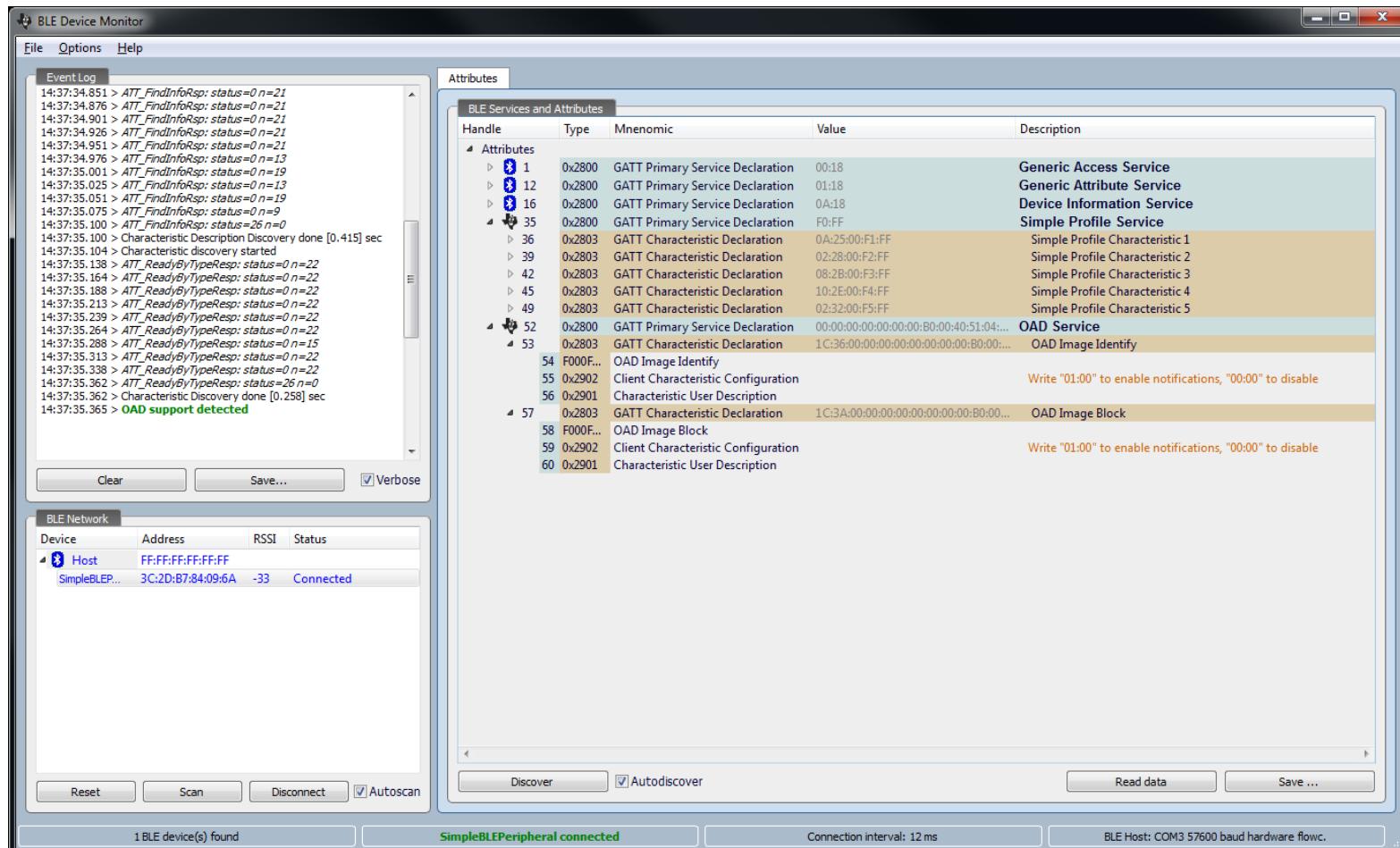
- Developer's Guide for Over Air Download for CC254x
 - Found at www.ti.com/blewiki

Device Monitor

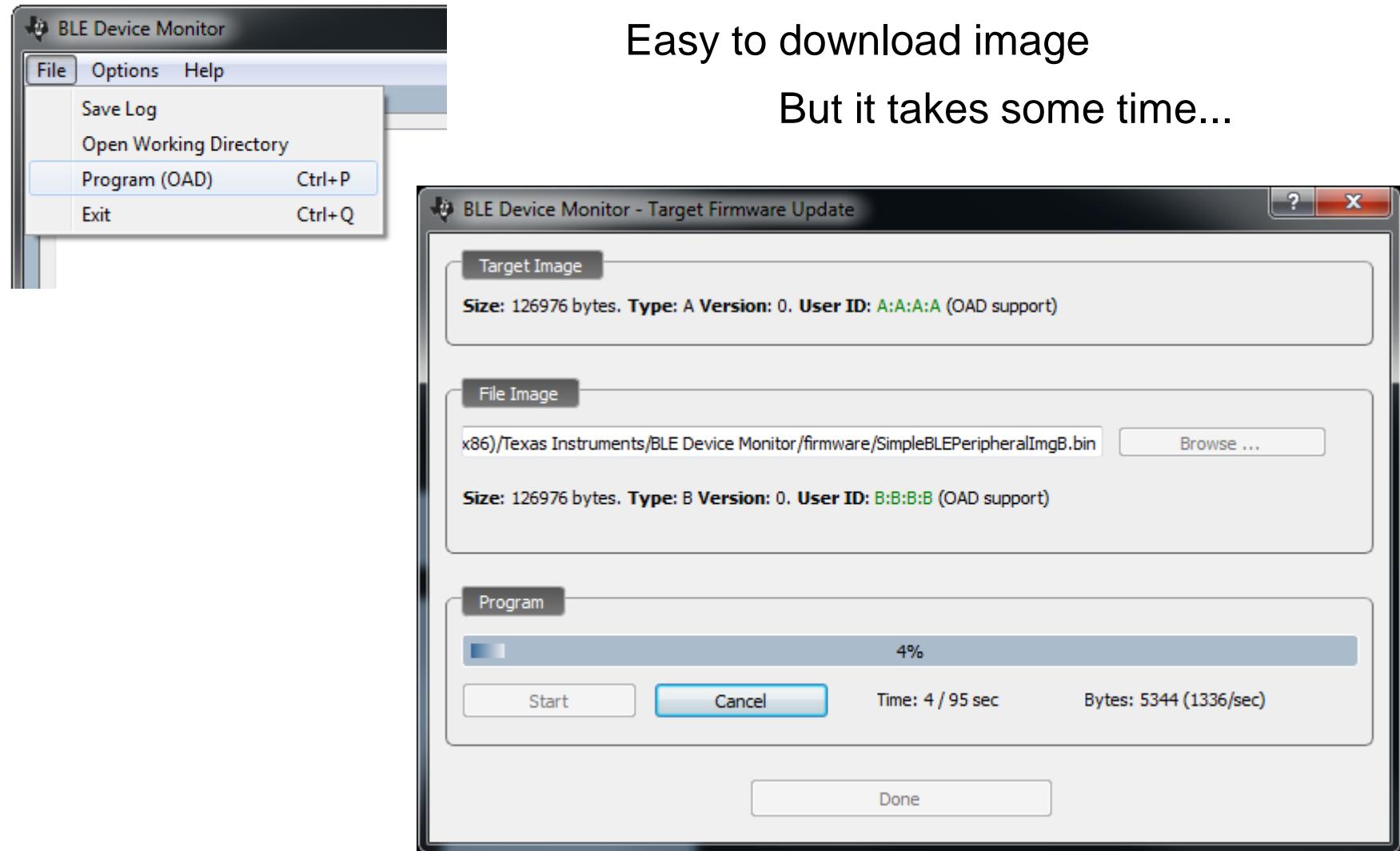
Debugging and Features



Device Monitor - Demo



Device Monitor



IAR Embedded Workbench

Debugging and Features

IAR Embedded Workbench IDE

- All software development on the CC2540/41 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

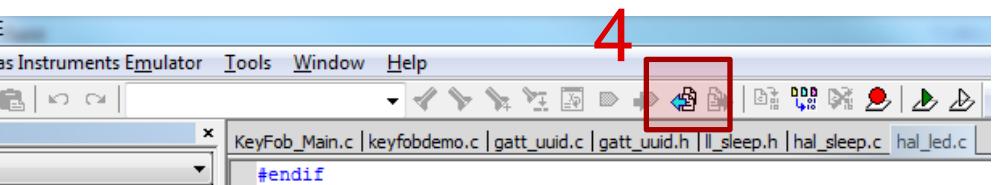


IAR Website: www.iar.com

Go to definition

If you ever wonder what a function does, how and where a macro or variable is defined:

1. Right click on a symbol you want to follow
2. Press Go to definition of...
3. Enjoy saving time
4. Go back whence you came



APP/keyfobdemo.c

A screenshot of the Texas Instruments IDE showing a context menu for the symbol 'HalLedSet'. The menu items include Cut, Copy, Paste, Complete, Match Brackets, Insert Template, Open Header/Source File, Go to definition of HalLedSet (highlighted), and Toggle Breakpoint (Code). Red numbers 1 and 2 are overlaid on the menu. The code snippet shows the declaration of the function:

```
// makes sure LEDs are off
HalLedSet( (HAL_LED 1 | HAL_LED 2), HAL_LED_MODE_OFF );
```

HAL/Target/CC2540EB/Drivers/hal_led.c

A screenshot of the Texas Instruments IDE showing the implementation of the 'HalLedSet' function. The code includes documentation comments and the implementation logic. Red number 3 is overlaid on the function definition. Red number 4 is overlaid on the 'endif' keyword at the bottom of the file.

```
/*
 * @fn      HalLedSet
 *
 * @brief   Turn ON/OFF/TOGGLE given LEDs
 *
 * @param   led - bit mask value of leds to be turned ON/OFF/TOGGLE
 *          mode - BLINK, FLASH, TOGGLE, ON, OFF
 * @return  None
 */
uint8 HalLedSet (uint8 leds, uint8 mode)
{

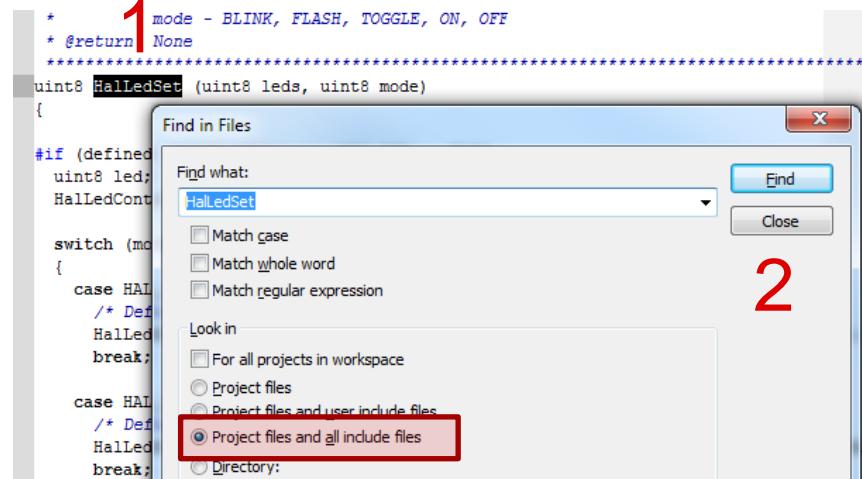
#if (defined (BLINK_LEDS)) && (HAL_LED == TRUE)
    uint8 led;
    HalLedControl_t *sts;

    switch (mode)
    {
        case HAL_LED_MODE_BLINK:
            /* Default blink, 1 time, D$ duty cycle */
            HalLedBlink (leds, 1, HAL_LED_DEFAULT_DUTY_CYCLE, HAL_LED_DEFAULT_FLASH_TIME);
            break;
    }
}
```

Find all in files..

To find where a certain variable is manipulated, where an OSAL Event is handled, or where a function is called:

1. Double click on the symbol
2. Press CTRL+SHIFT+F
3. Double click to jump in the code



Path	Line	String
C:\Texas Instruments\BLE-CC254x-1.2.1\..\hal_led.h	98	extern uint8 HalLedSet (uint8 led, uint8 mode);
C:\Texas Instruments\BLE-CC254x-1.2.1\..\hal_led.c	120	HalLedSet (HAL_LED_ALL, HAL_LED_MODE_OFF);
C:\Texas Instruments\BLE-CC254x-1.2.1\..\hal_led.c	129	* @fn HalLedSet
C:\Texas Instruments\BLE-CC254x-1.2.1\..\hal_led.c	137	uint8 HalLedSet (uint8 leds, uint8 mode)
C:\Texas Instruments\BLE-CC254x-1.2.1\..\hal_led.c	253	HalLedSet (leds, HAL_LED_MODE_ON); /*>= 100%, turn on */
C:\Texas Instruments\BLE-CC254x-1.2.1\..\hal_led.c	258	HalLedSet (leds, HAL_LED_MODE_OFF); /*No on time, turn off */
C:\Texas Instruments\BLE-CC254x-1.2.1\..\hal_led.c	340	HalLedSet (led, ((preBlinkState & led) != 0) ? HAL_LED_MODE_ON : HAL_LED_MODE_OFF);
C:\Texas Instruments\BLE-CC254x-1.2.1\..\keyfobdemo.c	369	HalLedSet ((HAL_LED_1 HAL_LED_2), HAL_LED_MODE_OFF);
C:\Texas Instruments\BLE-CC254x-1.2.1\..\keyfobdemo.c	692	HalLedSet ((HAL_LED_1 HAL_LED_2), HAL_LED_MODE_OFF);
C:\Texas Instruments\BLE-CC254x-1.2.1\..\keyfobdemo.c	708	HalLedSet (HAL_LED_1, HAL_LED_MODE_ON);
C:\Texas Instruments\BLE-CC254x-1.2.1\..\keyfobdemo.c	709	HalLedSet (HAL_LED_2, HAL_LED_MODE_FLASH);
C:\Texas Instruments\BLE-CC254x-1.2.1\..\keyfobdemo.c	736	HalLedSet ((HAL_LED_1 HAL_LED_2), HAL_LED_MODE_OFF);
C:\Texas Instruments\BLE-CC254x-1.2.1\..\keyfobdemo.c	753	HalLedSet (HAL_LED_1, HAL_LED_MODE_ON);
C:\Texas Instruments\BLE-CC254x-1.2.1\..\keyfobdemo.c	754	HalLedSet (HAL_LED_2, HAL_LED_MODE_FLASH);
C:\Texas Instruments\BLE-CC254x-1.2.1\..\keyfobdemo.c	783	HalLedSet ((HAL_LED_1 HAL_LED_2), HAL_LED_MODE_OFF);
C:\Texas Instruments\BLE-CC254x-1.2.1\..\OnBoard.c	106	HalLedSet (HAL_LED_ALL, HAL_LED_MODE_OFF);

Found 16 instances. Searched in 98 files.

Declaration
Definition
Calls to function

Break the Code

1. Put the cursor on a line
2. Press F9. Code broken.
3. Download, *Ctrl+D* or 
4. Run, *F5* or 
5. Step Over, *F10* or 

```
716 static void performPeriodicTask( void )
717 {
718     uint8 valueToCopy;
719     uint8 stat;
720
721     // Call to retrieve the value of the third characteristic in the profile
722     stat = SimpleProfile_GetParameter( SIMPLEPROFILE_CHAR3, &valueToCopy);
723
724     if( stat == SUCCESS )
725     {
726         /*
727         * Call to set that value of the fourth characteristic in the profile. Note
728         * that if notifications of the fourth characteristic have been enabled by
729         * a GATT client device, then a notification will be sent every time this
730         * function is called.
731         */
732     SimpleProfile_SetParameter( SIMPLEPROFILE_CHAR4, sizeof(uint8), &valueToCopy);
733 }
734 }
```

5

1,2

```
simpleBLEPeripheral.c SimpleBLEPeripheral_Main.c
716 static void performPeriodicTask( void )
717 {
718     uint8 valueToCopy;
719     uint8 stat;
720
721     // Call to retrieve the value of the third characteristic in the profile
722     stat = SimpleProfile_GetParameter( SIMPLEPROFILE_CHAR3, &valueToCopy);
723
724     if( stat == SUCCESS )
725     {
726         /*
727         * Call to set that value of the fourth characteristic in the profile. Note
728         * that if notifications of the fourth characteristic have been enabled by
729         * a GATT client device, then a notification will be sent every time this
730         * function is called.
731         */
732     SimpleProfile_SetParameter( SIMPLEPROFILE_CHAR4, sizeof(uint8), &valueToCopy);
733 }
734 }
```

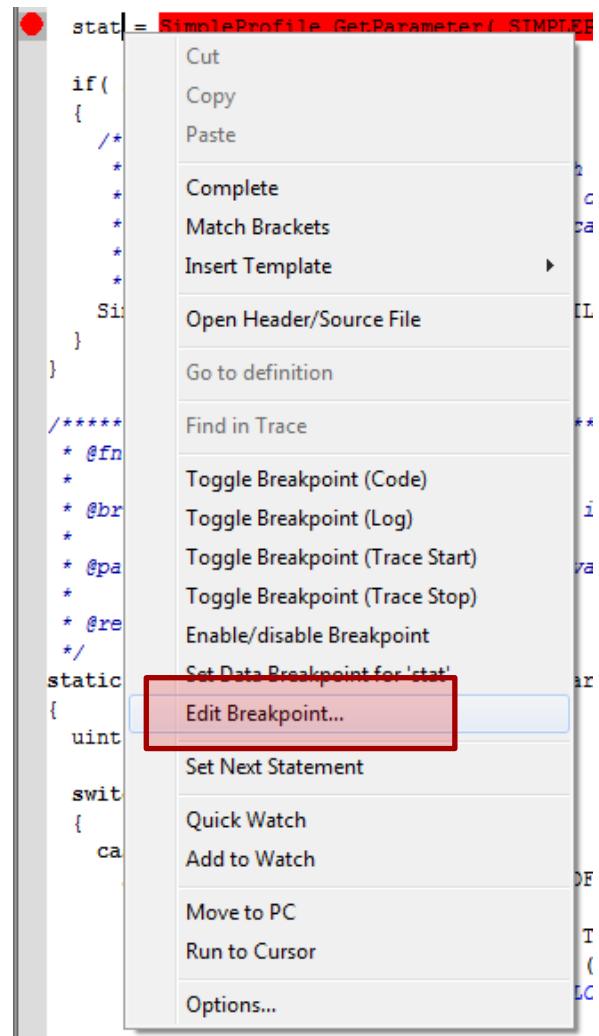
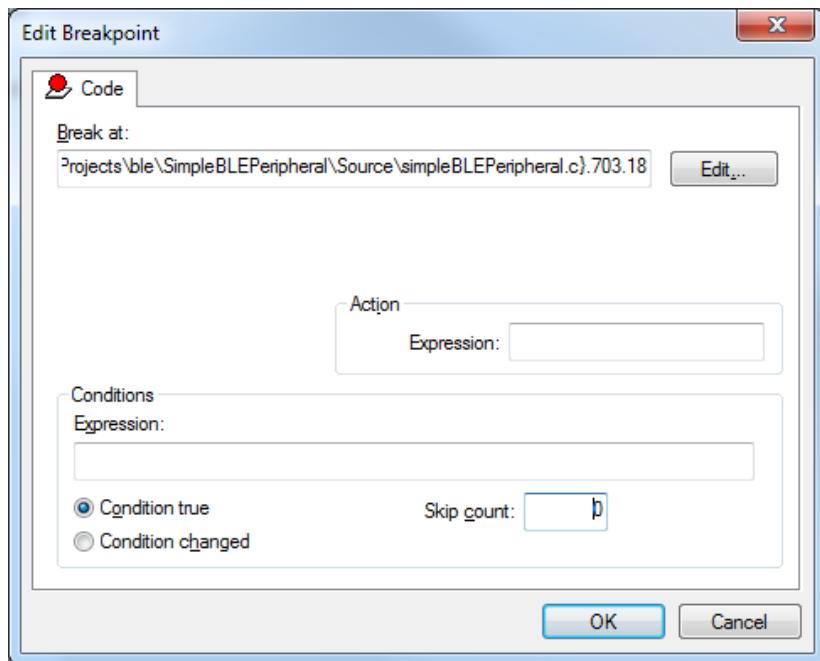
4

```
simpleBLEPeripheral.c SimpleBLEPeripheral_Main.c
716 static void performPeriodicTask( void )
717 {
718     uint8 valueToCopy;
719     uint8 stat;
720
721     // Call to retrieve the value of the third characteristic in the profile
722     stat = SimpleProfile_GetParameter( SIMPLEPROFILE_CHAR3, &valueToCopy);
723
724     if( stat == SUCCESS )
725     {
726         /*
727         * Call to set that value of the fourth characteristic in the profile. Note
728         * that if notifications of the fourth characteristic have been enabled by
729         * a GATT client device, then a notification will be sent every time this
730         * function is called.
731         */
732     SimpleProfile_SetParameter( SIMPLEPROFILE_CHAR4, sizeof(uint8), &valueToCopy);
733 }
734 }
```

Note that breaking the code during a connection will terminate the connection

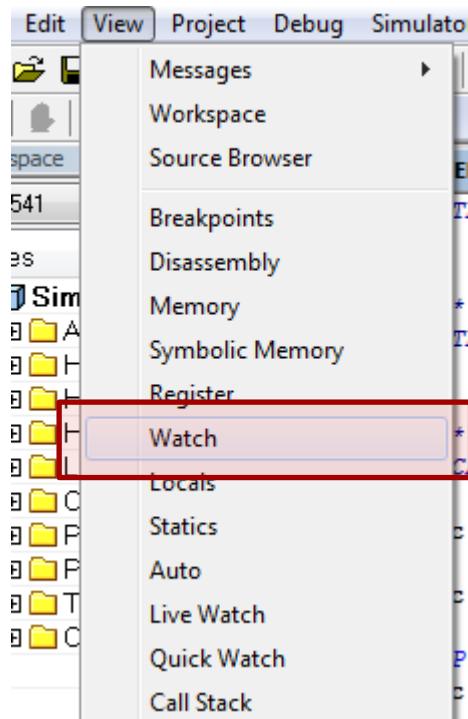
Hairline fracture

- Right click on a breakpoint
- Choose Edit Breakpoint...
- Don't break the code until it's helpful
 - Skip a number of breaks
 - Break on a condition



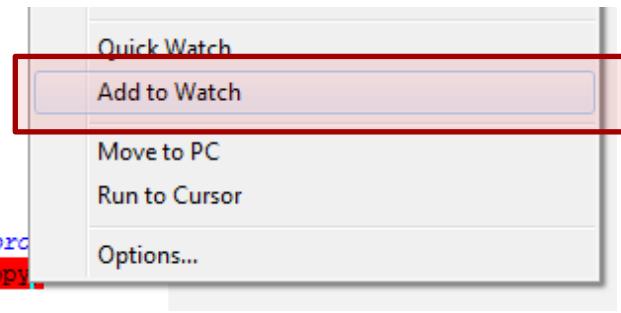
Watch the data

- View the watch window
- Add a variable to the watch list (right click)
- Next step
 - Run until breakpoint
 - Hit F10 to step over the statement
 - Check new value of variable in watch window



```
* @return  none
*/
static void performPeriodicTask( void )
{
    uint8 valueToCopy;
    uint8 stat;

    // Call to retrieve the value of the third characteristic in the pro
    stat = SimpleProfile_GetParameter( SIMPLEPROFILE_CHAR3, &valueToCopy );
```



How did I get here?

- Monitor
 - 1. Function Call Sequence: View -> Call Stack
 - 2. Local Variables: View -> Locals
 - 3. Memory: View -> Memory

The screenshot shows a debugger interface with three main windows:

- Call Stack**: Shows the call stack with the current frame at the top: `SimpleProfile_SetParameter()`. A red circle labeled "1" highlights the value `'\0' (0x03)` in the code editor.
- Locals**: Shows local variables: `valueToCopy` (Value: `'\0' (0x03)`, Type: `uint8`) and `stat` (Value: `<unavailable>`). A red circle labeled "2" highlights the value `'\0' (0x03)`.
- Registers**: Shows memory dump starting at address `0240`. A red circle labeled "3" highlights the value `00` at address `0260`.

Code in the editor:

```
716 static void performPeriodicTask( void )
717 {
718     uint8 valueToCopy;
719     uint8 stat;    uint8 valueToCopy = '\0' (0x03);
720
721     // Call to retrieve the value of the third characteristic in the profile
722     stat = SimpleProfile_GetParameter( SIMPLEPROFILE_CHAR3, &valueToCopy );
723
724     if( stat == SUCCESS )
725     {
726         /*
727          * Call to set that value of the fourth characteristic in the profile. Note
728          * that if notifications of the fourth characteristic have been enabled by
729          * a GATT client device, then a notification will be sent every time this
730          * function is called.
731         */
732     SimpleProfile_SetParameter( SIMPLEPROFILE_CHAR4, sizeof(uint8), &valueToCopy );
733 }
734 }
```

Watch Out

- Avoid compiling with optimization when debugging is on the Menu
 - Variables are often "Unavailable" with optimization on
 - Add `volatile` to the declaration, i.e. `volatile uint8 variable;`
 - If that doesn't work, move the declaration outside the function.
 - Single stepping (F11) will sometimes behave oddly or break IAR.
 - Assembly code is moved around and changed with optimization, so if debugging features are unavailable, try turning optimizations off.

Qualification and Certification

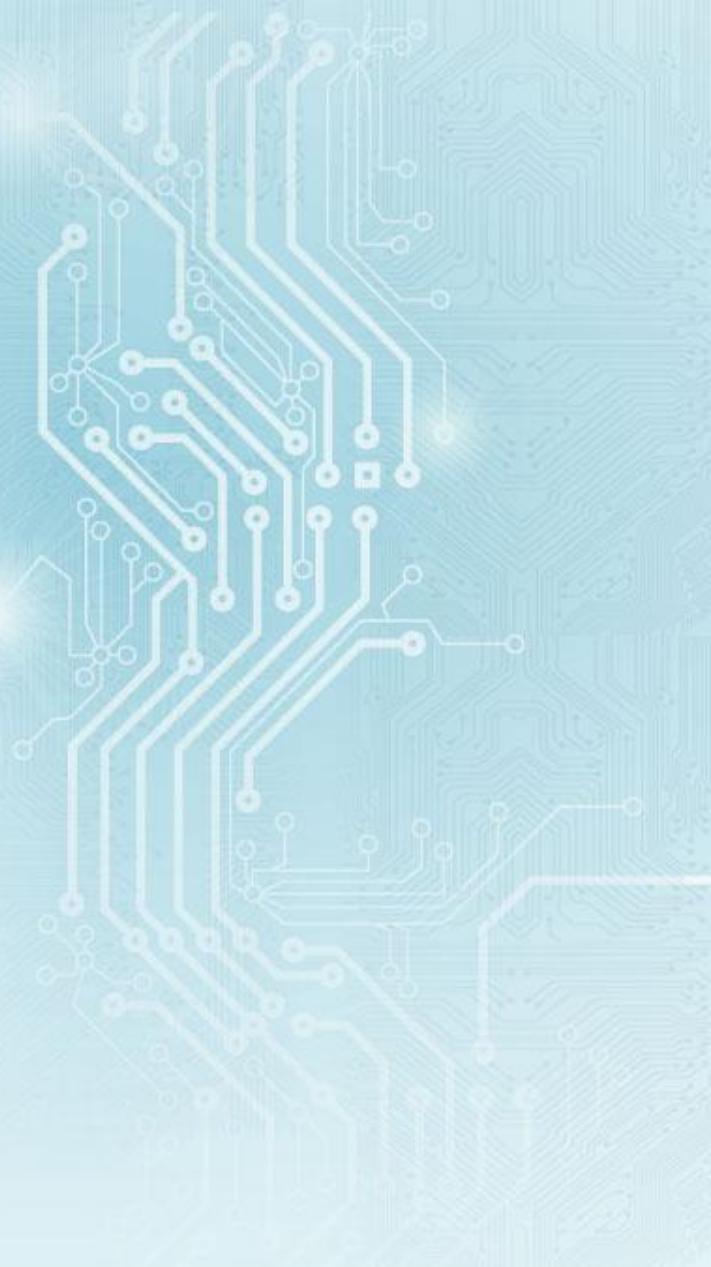
Introduction

TI BLE Subsystems

TI BLE Subsystem	Controller	Host	Profile
QDID	B016552	B017183	B020954
Include	HCI Link Layer RF PHY	GAP GATT ATT SMP L2CAP HCI	Health Thermometer Profile/Service Device Information Service Find Me Profile Immediate Alert Service Link Loss Service Proximity Profile Tx Power Service Heart Rate Profile/Service Time Profile Phone Alert Status Profile Alert Notification Profile Blood Pressure Profile/Service Battery Service HID Service Scan Parameters Service HID over GATT Profile Scan Parameters Profile. Running Speed and Cadence Profile/Service Cycling Speed and Cadence Profile/Service

SensorTag

Evaluation

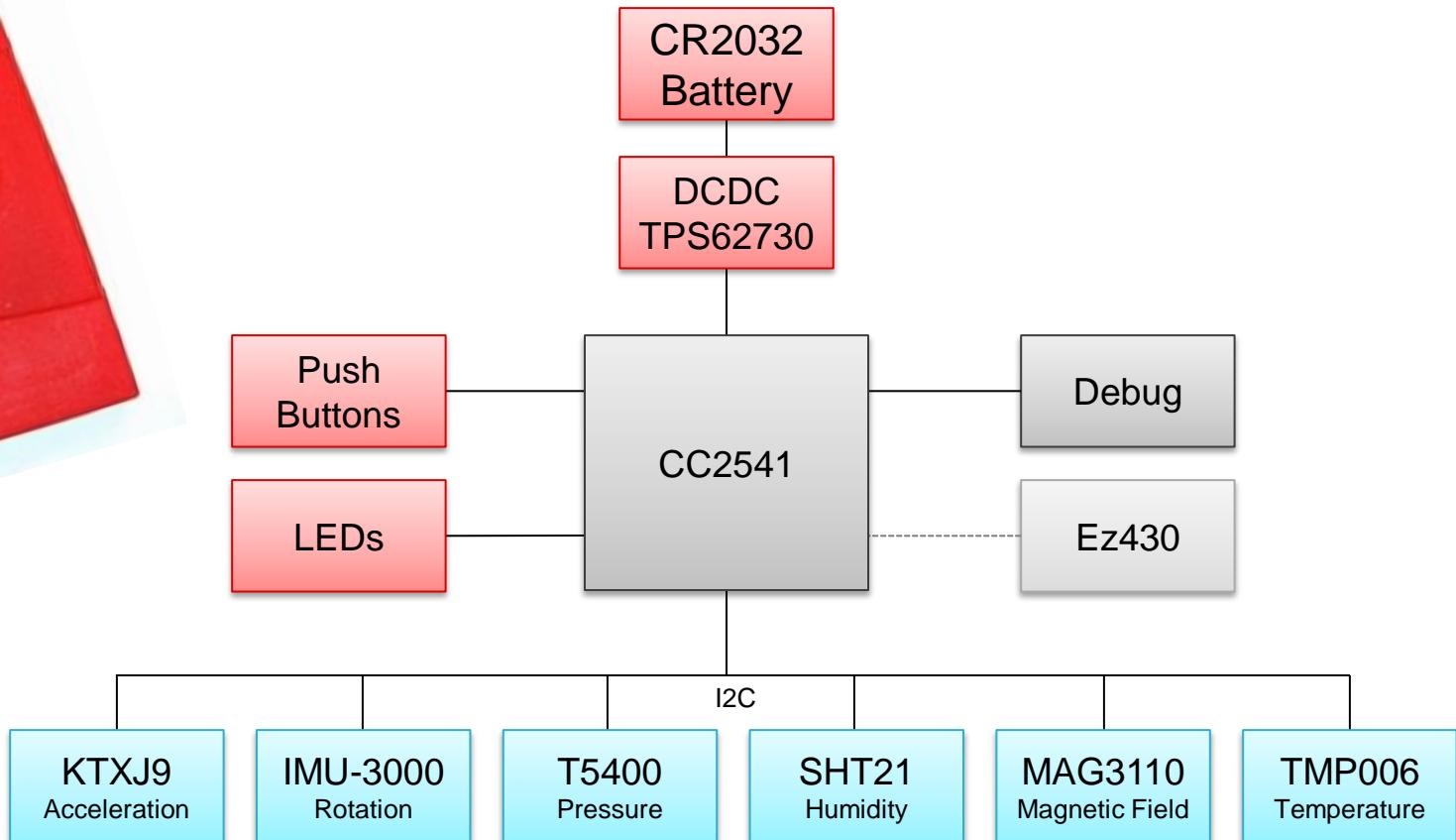


Bluetooth low energy SensorTag

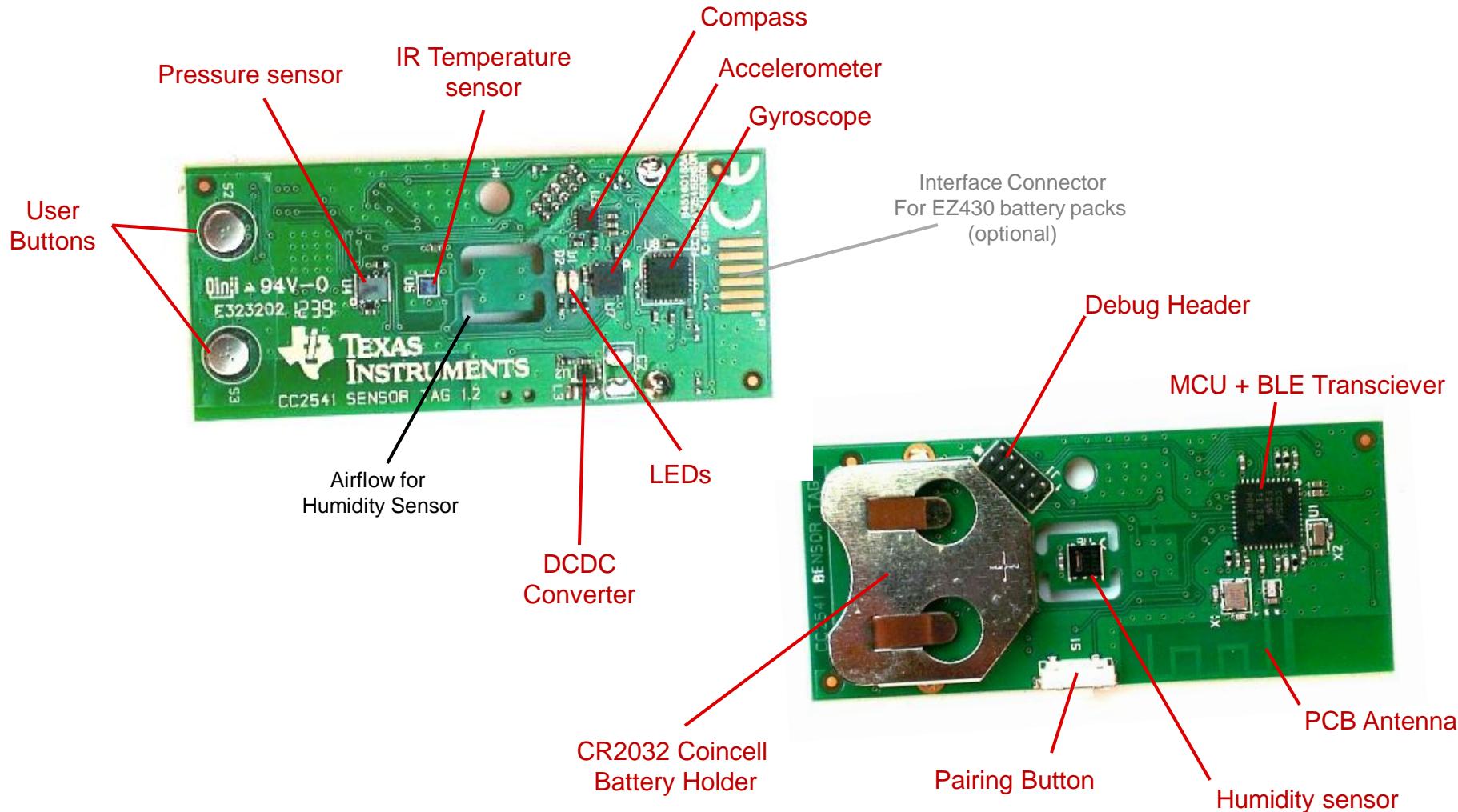
- Includes *Bluetooth* low energy
 - CC2541
- 6 sensors
 - Temperature
 - Humidity
 - Pressure
 - Accelerometer
 - Gyroscope
 - Magnetometer
- Single cell coin cell battery (CR2032)
- FCC, IC and ETSI certified solution



Bluetooth low energy SensorTag

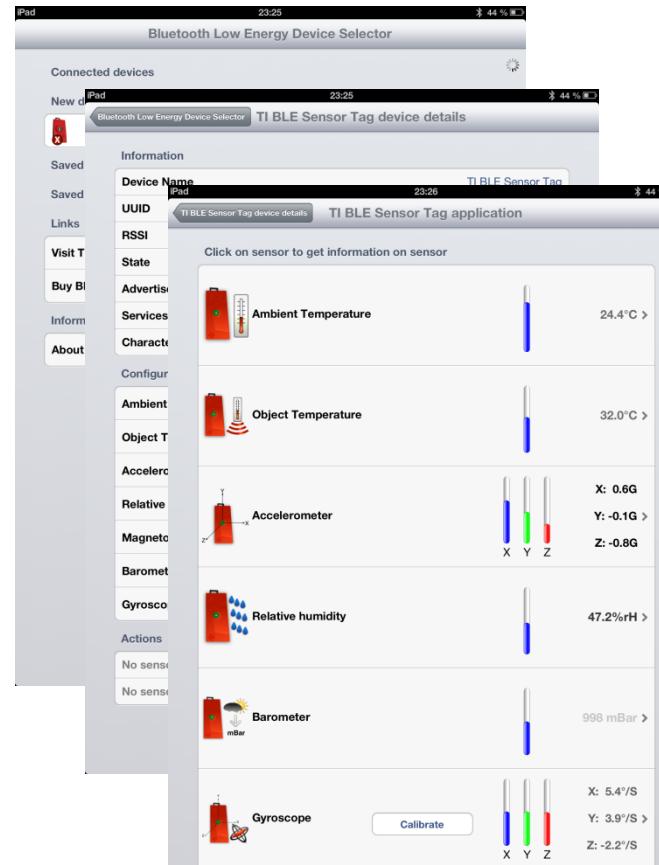


Bluetooth low energy SensorTag



Bluetooth low energy SensorTag

- Targeting Smart Phone App Developers
- Limitless App Possibilities
 - Health and fitness, medical, educational tools, toys, remote controls, mobile phone accessories, proximity and indoor locationing.
- The CC2541 SensorTag kit shortens the design time for Bluetooth low energy app development from months to hours
 - No embedded software design knowledge required.
 - The SensorTag includes all SW needed for App development
 - TI has a SensorTag App on the App Store providing developers with easy access to the tools they need.



Hands-On: SensorTag

Pre-requisites

- BLE Device Monitor (<http://www.ti.com/lit/zip/swrc258>)
- Packet Sniffer (<http://www.ti.com/tool/packet-sniffer>)

Kit Contents

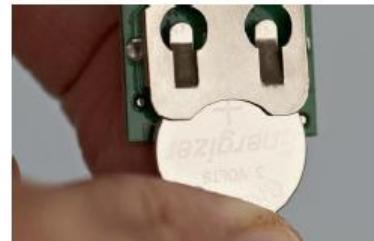


- SensorTag
- 1x CR2032 Battery
- Documentation

The RF boards in this kit are FCC and IC certified and tested/complies with ETSI/R&TTE over temperature from 0 to +35°C¹. The RF board have integrated PCB antenna.

Assembly of the SensorTag

Insert the battery in the battery connector and place the printed circuit board in the black plastic cover with the battery facing down.



Note that when inserting the battery for the first time, the PCB contact point surface has a thin layer of solder residue that may prevent contact with the battery. Remove and insert the battery a few times to power the SensorTag.

Push the side button to check if the battery is correctly inserted. When the button is pushed the LED marked D1 should start to blink.

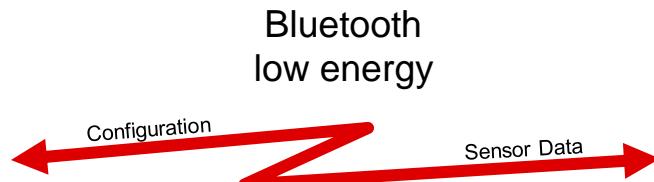
Add the transparent plastic cover to close the inner hard plastic pieces.



Complete the assembly of the SensorTag by adding the red plastic cover.



System Overview



Peripheral Device

- Slave
- Server (has data)



BLE Sniffer

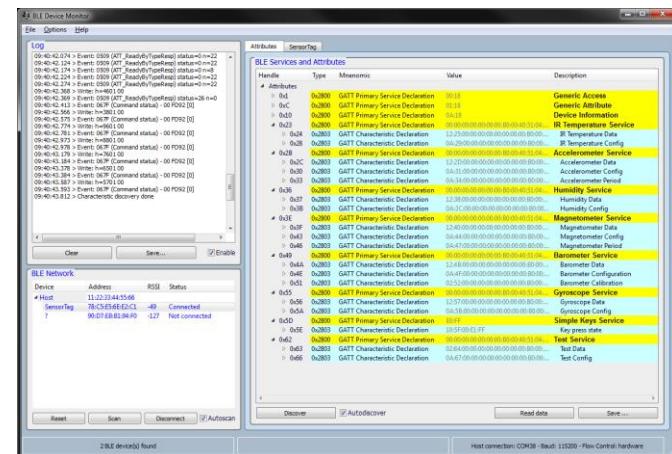


Central Device

- Master
- Client (wants data)

BLE Device Monitor

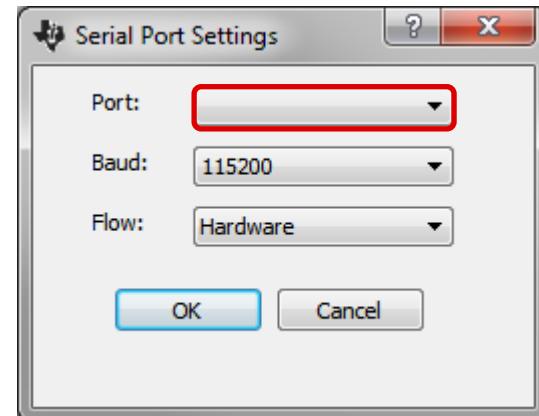
- Windows Application
 - The BLE Device Monitor allows you to
 - Discover
 - Read
 - Alter attributes
- on any *Bluetooth* low energy device



- More information about the BLE Device Monitor at:
www.ti.com/SensorTag

Setup BLE Device Monitor

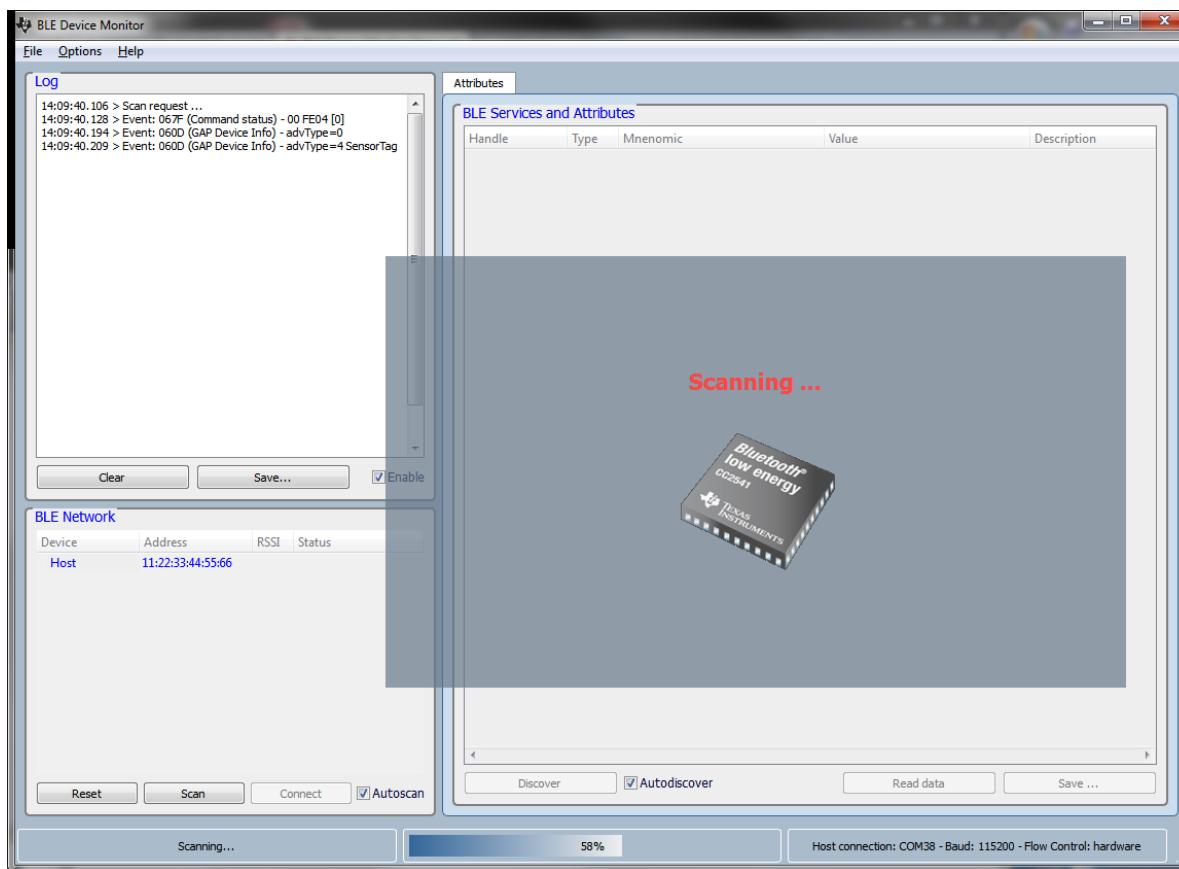
- Download: <http://www.ti.com/litv/zip/swrc258a>
- Connect CC2540 USB dongle,
 - Program the Dongle with CC2540_USBdongle_HostTestRelease_All.hex
(C:\Program Files\Texas Instruments\BLE Device Monitor\firmware)
 - Install USB CDC Driver, usb_cdc_driver_cc2540.inf
(C:\Program Files\Texas Instruments\BLE Device Monitor)



- Start BLE Device Manager
 - In Options, select Port

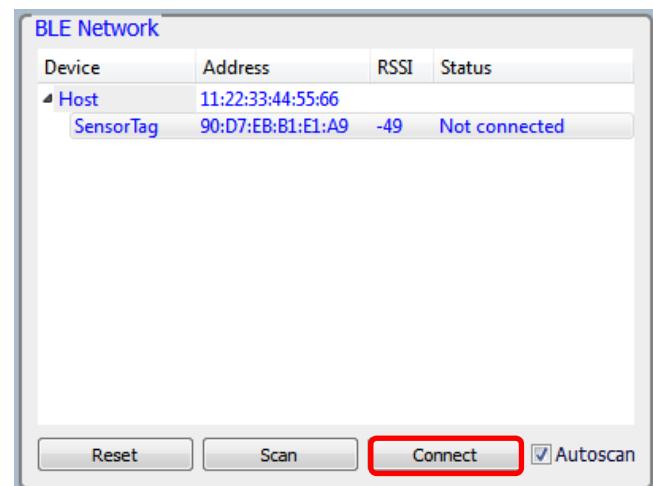
Discover the SensorTag

- Press Side Button to start Advertise



Connect to the SensorTag

- Select the SensorTag and click on “Connect”



- Status will change from **Not Connected** to **Connected** and all *Bluetooth* low energy Services and Attributes are shown.

Using Attributes Tab

- The Attribute tab will list all the GATT Primary Services that is available for the *Bluetooth* low energy device.
- Click on the + to display the corresponding characteristics to the Service.

BLE Services and Attributes				
Handle	Type	Mnemonic	Value	Description
Attributes				
0x1	0x2800	GATT Primary Service Declaration	00:18	Generic Access
0xC	0x2800	GATT Primary Service Declaration	01:18	Generic Attribute
0x10	0x2800	GATT Primary Service Declaration	0A:18	Device Information
0x23	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	IR Temperature Service
0x2B	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	Accelerometer Service
0x36	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	Humidity Service
0x3E	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	Magnetometer Service
0x49	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	Barometer Service
0x55	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	Gyroscope Service
0x5D	0x2800	GATT Primary Service Declaration	E0:FF	Simple Keys Service
0x62	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	Test Service

BLE Device Monitor

File Options Help

Log

```
09:40:42.074 > Event: 0509 (ATT_ReadyByTypeResp) status=0 n=22
09:40:42.124 > Event: 0509 (ATT_ReadyByTypeResp) status=0 n=22
09:40:42.174 > Event: 0509 (ATT_ReadyByTypeResp) status=0 n=8
09:40:42.224 > Event: 0509 (ATT_ReadyByTypeResp) status=0 n=22
09:40:42.274 > Event: 0509 (ATT_ReadyByTypeResp) status=0 n=22
09:40:42.368 > Write: h=460100
09:40:42.369 > Event: 0509 (ATT_ReadyByTypeResp) status=26 n=0
09:40:42.413 > Event: 067F (Command status) - 00 FD92 [0]
09:40:42.566 > Write: h=380100
09:40:42.575 > Event: 067F (Command status) - 00 FD92 [0]
09:40:42.774 > Write: h=960100
09:40:42.781 > Event: 067F (Command status) - 00 FD92 [0]
09:40:42.973 > Write: h=880100
09:40:42.978 > Event: 067F (Command status) - 00 FD92 [0]
09:40:43.179 > Write: h=760100
09:40:43.184 > Event: 067F (Command status) - 00 FD92 [0]
09:40:43.378 > Write: h=650100
09:40:43.384 > Event: 067F (Command status) - 00 FD92 [0]
09:40:43.587 > Write: h=570100
09:40:43.593 > Event: 067F (Command status) - 00 FD92 [0]
09:40:43.812 > Characteristic discovery done
```

BLE Services and Attributes

Handle	Type	Mnemonic	Value	Description
Attributes				
0x1	0x2800	GATT Primary Service Declaration	00:18	Generic Access
0xC	0x2800	GATT Primary Service Declaration	01:18	Generic Attribute
0x10	0x2800	GATT Primary Service Declaration	0A:18	Device Information
0x23	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	IR Temperature Service
0x24	0x2803	GATT Characteristic Declaration	12:25:00:00:00:00:00:00:00:B0:00:...	IR Temperature Data
0x28	0x2803	GATT Characteristic Declaration	0A:29:00:00:00:00:00:00:00:B0:00:...	IR Temperature Config
0x2B	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	Accelerometer Service
0x2C	0x2803	GATT Characteristic Declaration	12:2D:00:00:00:00:00:00:00:B0:00:...	Accelerometer Data
0x30	0x2803	GATT Characteristic Declaration	0A:31:00:00:00:00:00:00:B0:00:...	Accelerometer Config
0x33	0x2803	GATT Characteristic Declaration	0A:34:00:00:00:00:00:00:00:B0:00:...	Accelerometer Period
0x36	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	Humidity Service
0x37	0x2803	GATT Characteristic Declaration	12:38:00:00:00:00:00:00:00:B0:00:...	Humidity Data
0x3B	0x2803	GATT Characteristic Declaration	0A:3C:00:00:00:00:00:00:00:B0:00:...	Humidity Config
0x3E	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	Magnetometer Service
0x3F	0x2803	GATT Characteristic Declaration	12:40:00:00:00:00:00:00:00:B0:00:...	Magnetometer Data
0x43	0x2803	GATT Characteristic Declaration	0A:44:00:00:00:00:00:00:00:B0:00:...	Magnetometer Config
0x46	0x2803	GATT Characteristic Declaration	0A:47:00:00:00:00:00:00:00:B0:00:...	Magnetometer Period
0x49	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	Barometer Service
0x4A	0x2803	GATT Characteristic Declaration	12:4B:00:00:00:00:00:00:00:B0:00:...	Barometer Data
0x4E	0x2803	GATT Characteristic Declaration	0A:4F:00:00:00:00:00:00:00:B0:00:...	Barometer Configuration
0x51	0x2803	GATT Characteristic Declaration	02:52:00:00:00:00:00:00:00:B0:00:...	Barometer Calibration
0x55	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	Gyroscope Service
0x56	0x2803	GATT Characteristic Declaration	12:57:00:00:00:00:00:00:00:B0:00:...	Gyroscope Data
0x5A	0x2803	GATT Characteristic Declaration	0A:5B:00:00:00:00:00:00:00:B0:00:...	Gyroscope Config
0x5D	0x2800	GATT Primary Service Declaration	E0:FF	Simple Keys Service
0x5E	0x2803	GATT Characteristic Declaration	10:5F:00:E1:FF	Key press state
0x62	0x2800	GATT Primary Service Declaration	00:00:00:00:00:00:B0:00:40:51:04:...	Test Service
0x63	0x2803	GATT Characteristic Declaration	02:64:00:00:00:00:00:00:00:B0:00:...	Test Data
0x66	0x2803	GATT Characteristic Declaration	0A:67:00:00:00:00:00:00:00:B0:00:...	Test Config

BLE Network

Device	Address	RSSI	Status
Host	11:22:33:44:55:66		
SensorTag	78:C5:E5:6E:E2:C1	-49	Connected
?	90:D7:EB:B1:94:F0	-127	Not connected

Buttons

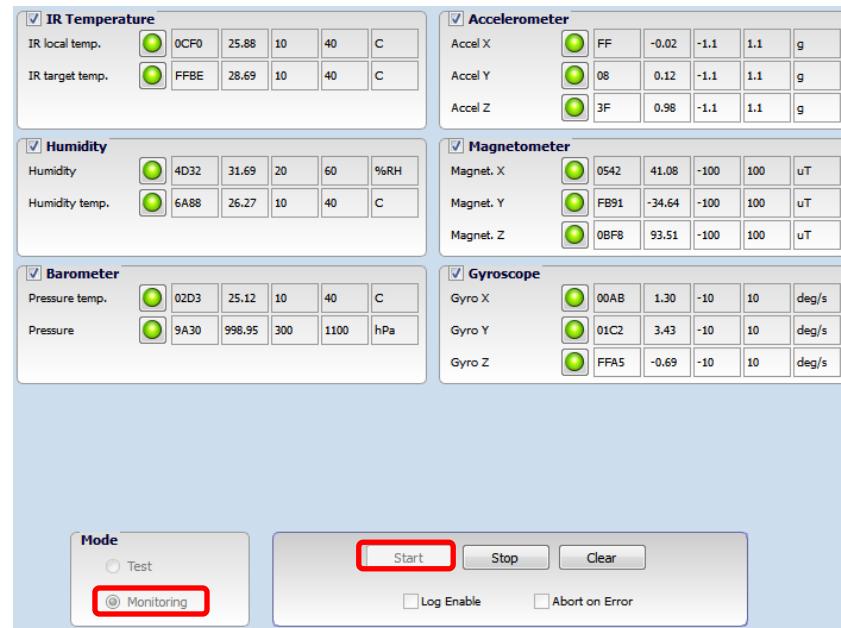
- Clear
- Save...
- Enable
- Discover
- Autodiscover
- Read data
- Save ...

2 BLE device(s) found

Host connection: COM38 - Baud: 115200 - Flow Control: hardware

Using SensorTag Tab

- **Monitor mode** provides numerical readings of the sensors
 1. Select Monitoring
 2. Select the sensors
 3. Click on “Start”



- **Test Mode** is a production test functionality of the SensorTag which will enable you to test all the Sensors and peripherals that are available on the SensorTag. The result will be stored in time stamped log file.

SensorTag Source Code

The screenshot shows the IAR Embedded Workbench IDE interface. The title bar reads "IAR Embedded Workbench IDE". The menu bar includes File, Edit, View, Project, Texas Instruments Emulator, Tools, Window, and Help. The workspace shows a project named "SensorTag - CC2541DK-SEN...". The left pane displays the file structure:

- APP
- HAL
- LIB
- NPI
- OSAL
- PROFILES
 - accelerometerservice.c
 - accelerometerservice.h
 - barometerservice.c
 - barometerservice.h
 - devinfservice.c
 - devinfservice.h
 - gap.c
 - gapbondmgr.c
 - gapbondmgr.h
 - gapgatserver.h
 - gatt_uid.c
 - gattserver.h
 - gyroservice.c
 - gyroservice.h
 - humidityservice.c
 - humidityservice.h
 - intempservice.c
 - intempservice.h
 - magnetometerservice.c
 - magnetometerservice.h
 - peripheral.c
 - peripheral.h
 - simpleGATTprofile.c
 - simpleGATTprofile.h
 - simplekeys.c
 - simplekeys.h
 - st_util.c
 - st_util.h
 - testservice.c
 - testservice.h
- TOOLS
- Output

The main code editor window displays the "SensorTag.c" file content, which is a C program for the SensorTag. The code includes various defines and configurations for the sensor services. The right pane shows the build status and messages.

```
115 // Whether to enable automatic parameter update request when a connection is formed
116 #define DEFAULT_ENABLE_UPDATE_REQUEST FALSE
117
118 // General discoverable mode advertises indefinitely
119 #define DEFAULT_DISCOVERABLE_MODE GAP_ADTYPE_FLAGS_LIMITED
120
121 // Minimum connection interval (units of 1.25ms, 80=100ms) if automatic parameter update request is enabled
122 #define DEFAULT_DESIRED_MIN_CONN_INTERVAL 80
123
124 // Maximum connection interval (units of 1.25ms, 800=1000ms) if automatic parameter update request is enabled
125 #define DEFAULT_DESIRED_MAX_CONN_INTERVAL 800
126
127 // Slave Latency to use if automatic parameter update request is enabled
128 #define DEFAULT_DESIRED_SLAVE_LATENCY 0
129
130 // Supervision timeout value (units of 10ms, 1000=10s) if automatic parameter update request is enabled
131 #define DEFAULT_DESIRED_CONN_TIMEOUT 1000
132
133 // Company Identifier: Texas Instruments Inc. (13)
134 #define TI_COMPANY_ID 0x000D
135
136 #define INVALID_CONNHANDLE 0xFFFF
137
138 // Length of bd addr as a string
139 #define B_ADDR_STR_LEN 15
140
141 #if defined ( PLUS_BROADCASTER )
142 #define ADV_IN_CONN_WAIT 500 // delay 500 ms
143 #endif
144
145 // Side key bit
146 #define SK_KEY_SIDE 0x04
147
148 // Number of power-on self-test runs
149 #define N_TEST_RUNS 10
150
151 // Self-test bit values
152 #define ST_IRTEMP 0x01
153 #define ST_HUMID 0x02
154 #define ST_MAGN 0x04
155 #define ST_ACC 0x08
156 #define ST_PRESS 0x10
157 #define ST_GYRO 0x20
158
159 // Common values for turning a sensor on and off + config/status
160 #define ST_CFG_SENSOR_DISABLE 0x00
161 #define ST_CFG_SENSOR_ENABLE 0x01
162 #define ST_CFG_CALIBRATE 0x02
163 #define ST_CFG_ERROR 0xF
164
165 ****
166 * TVPFDFFS
```

Build Messages

Total number of errors: 0
Total number of warnings: 0

Ready Errors 0, Warnings 0 NUM

Evaluation of CC2541DK-MINI

CC2541DK-MINI Unboxing

Quick Start Guide



Keyfob Casing



CC2541 Keyfob



CC2540USB Dongle

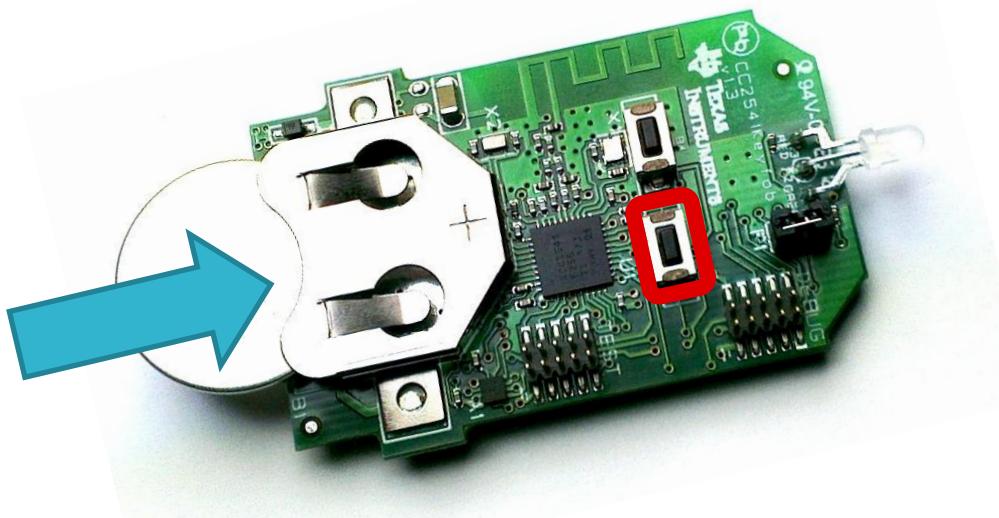


CC Debugger

Both CC2541 Keyfob and CC2540USB Dongle are pre-programmed

Power up the Keyfob

- There are two ways of powering the CC2541 Keyfob
 - Using a CR2032 Coin Cell Battery
 - Using CC Debugger (Note that the header must be mounted on P1)



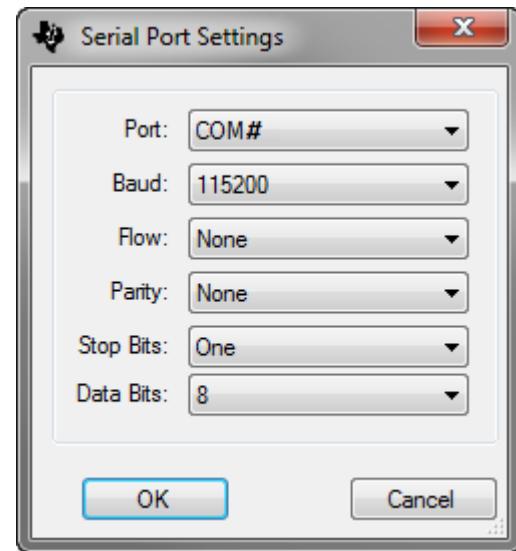
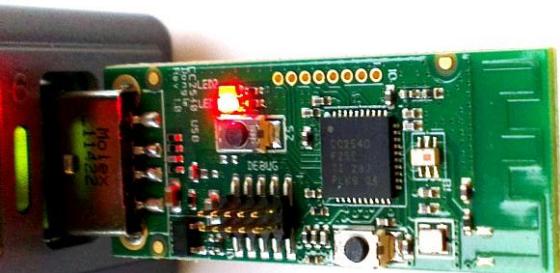
OR



- Device LED should lit green for 1 second
- Press Right button to start advertisement – Device LED should blink red during the advertisement mode.

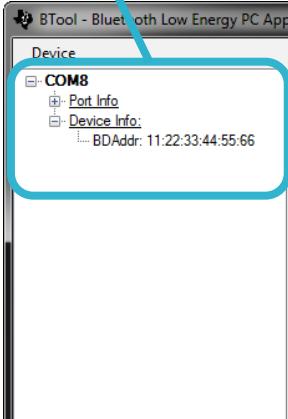
Setup PC environment

1. Insert CC2540 USB Dongle
2. Install driver, located at ..\BLE-CC254x-1.3\Accessories\Drivers
3. Start BTool
 1. Choose Port (dependent on your machine)
 2. Press OK

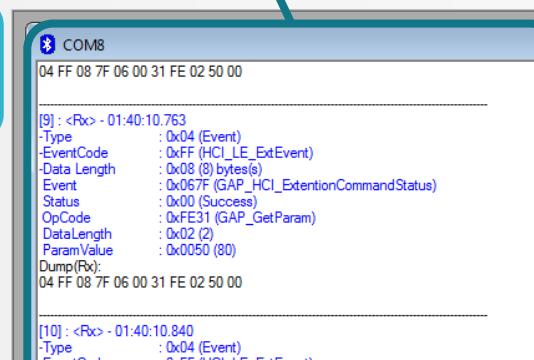


Evaluate CC2541DK-MINI

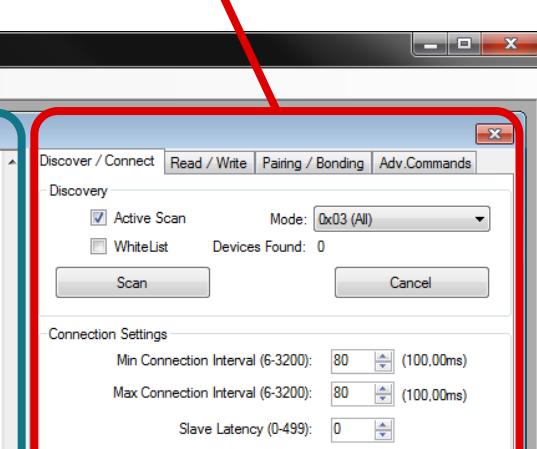
Device information



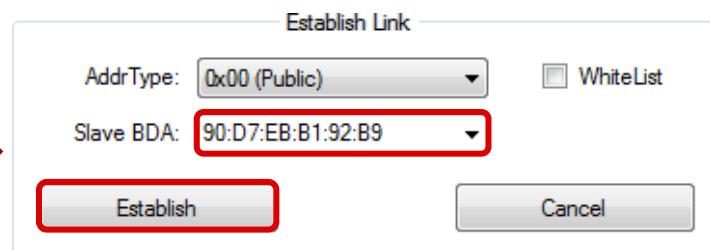
Message Log



Device Control

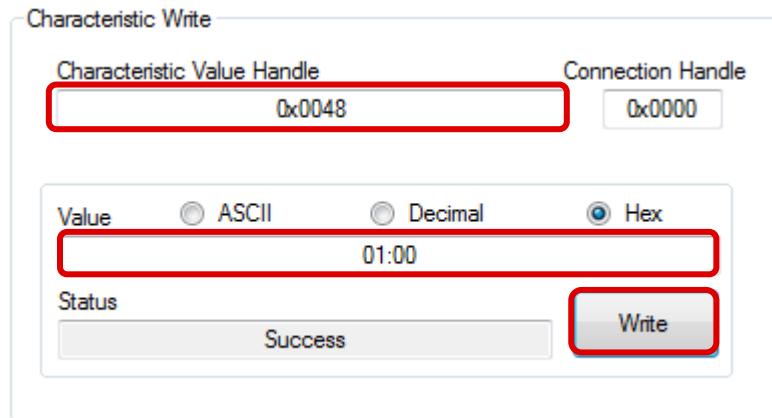


- Connect to CC2541 Keyfob



Interact with the Keyfob - Buttons

- Button Notifications
 - In Btool: Enable notifications by writing `01:00` to handle `0x0048`



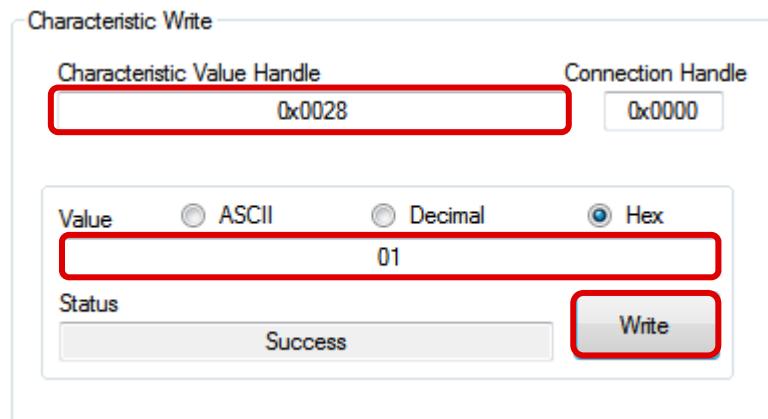
- On Keyfob: Press buttons to send notification



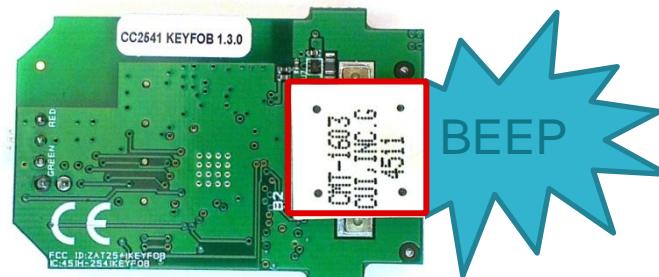
```
[42] : <Rx> - 11:02:26.367
-Type : 0x04 (Event)
-EventCode : 0xFF (HCI_LE_ExtEvent)
-Data Length : 0x09 (9) bytes(s)
Event : 0x051B (ATT_HandleValueNotification)
Status : 0x00 (Success)
ConnHandle : 0x0000 (0)
PduLen : 0x03 (3)
Handle : 0x0047 (71)
Value : 01
Dump(Rx):
04 FF 09 1B 05 00 00 00 03 47 00 01
```

Interact with the Keyfob - Alert

- Immediate Alert
 - In Btool: Trigger a buzzer alert by writing 01:00 to handle 0x0028



{
01 for Low Alert
02 for High Alert
00 to Turn Off



Interact with the Keyfob - Accelerometer

- Accelerometer
 - In Btool: Enable accelerometer by writing 01:00 to handle 0x0034
 - In Btool: Enable x-axis notifications by writing 01:00 to handle 0x003B

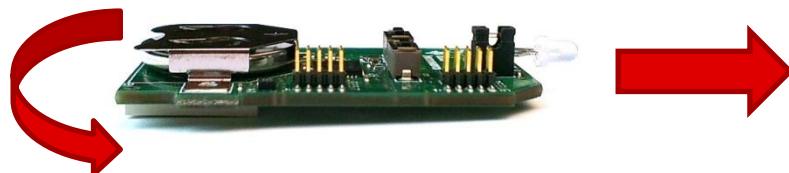
Characteristic Write

Characteristic Value Handle	0x0034	Connection Handle	0x0000
Value	<input type="radio"/> ASCII	<input type="radio"/> Decimal	<input checked="" type="radio"/> Hex
01:00			
Status	Success		
		Write	

Characteristic Write

Characteristic Value Handle	0x003B	Connection Handle	0x0000
Value	<input type="radio"/> ASCII	<input type="radio"/> Decimal	<input checked="" type="radio"/> Hex
01:00			
Status	Success		
		Write	

- Move the Keyfob



```
[160] : <Rx> - 03:33:36.833
-Type : 0x04 (Event)
-EventCode : 0xFF (HCI_LE_ExtEvent)
-Data Length : 0x09 (9) bytes(s)
Event : 0x051B (ATT_HandleValueNotification)
Status : 0x00 (Success)
ConnHandle : 0x0000 (0)
PduLen : 0x03 (3)
Handle : 0x003A (58)
Value : E6
Dump(Rx):
04 FF 09 1B 05 00 00 00 03 3A 00 E6
```

Human Interface Device over Generic Attribute Profile

**HID over GATT Profile
HOGP**

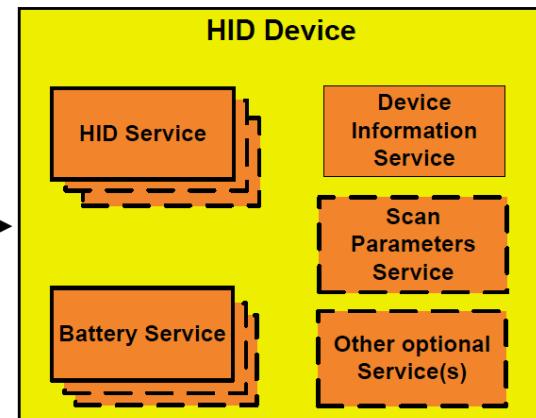
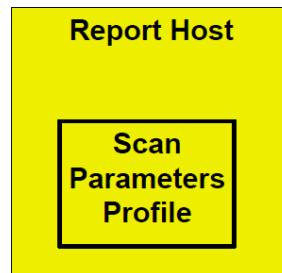
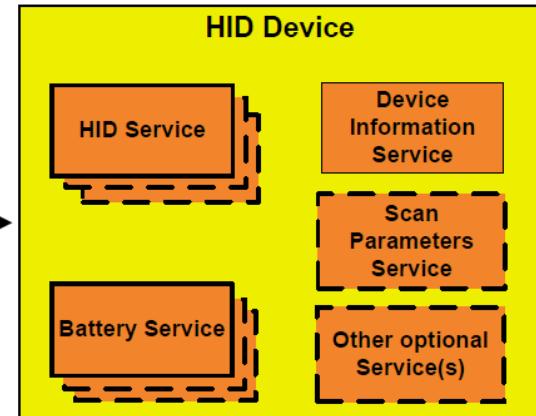
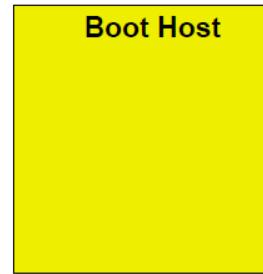
BLE HID: Introduction

- "HID Over GATT" Profile specification was approved by the BT SIG in December 2011
- Specification is publicly available at bluetooth.org
- Microsoft has already announced native support for LE HID in Windows 8

HID over GATT Profile

Profile in yellow,
Services in orange

- Host and Device behave according to HoG Profile
- Data is exchanged using characteristics in the HID Service
- HID Service is a wrapper for usual USB HID reports.



- ↑ Notification
- ↑ Read
- ↓ Write
- ↓ WriteNoRsp

Attribute table: HOGP req. services

Battery service

Scan Param. Service

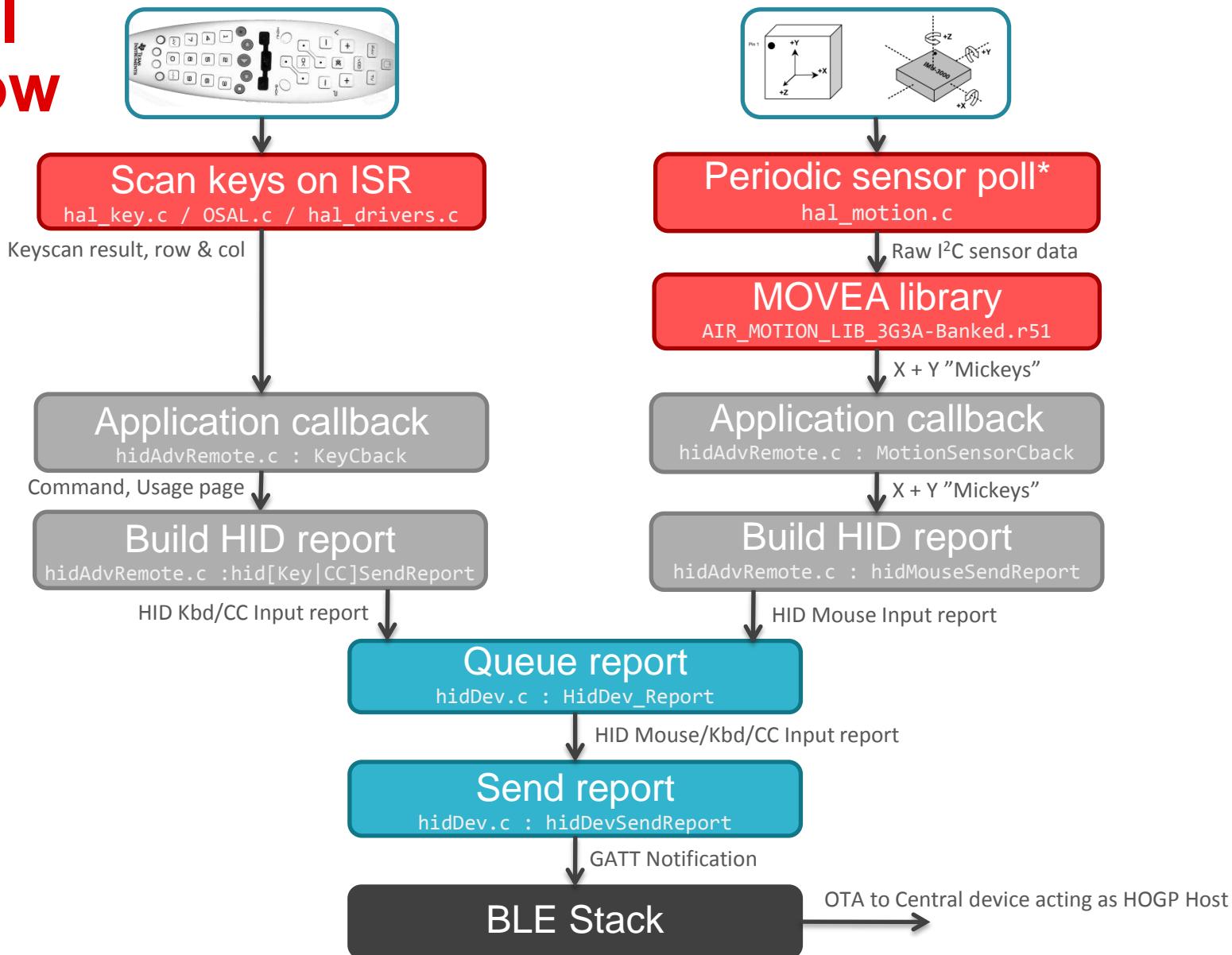
HID Service

Handle	Uuid	Uuid Description	Value	
0x0023	0x2800	GATT Primary Service Declaration	0F:18	
0x0024	0x2803	GATT Characteristic Declaration	12:25:00:19:2A	
0x0025	0x2A19	Battery Level	00	↑↑
0x0026	0x2902	Client Characteristic Configuration	9E:00	↑↓
0x0027	0x2908	Report Reference	04:01	↑
0x0028	0x2800	GATT Primary Service Declaration	13:18	
0x0029	0x2803	GATT Characteristic Declaration	04:2A:00:4F:2A	
0x002A	0x2A4F	Scan Interval Window		↓ ←
0x002B	0x2803	GATT Characteristic Declaration	10:2C:00:31:2A	
0x002C	0x2A31	Scan Refresh		↑
0x002D	0x2902	Client Characteristic Configuration	9E:00	↑↓ ←
0x002E	0x2800	GATT Primary Service Declaration	12:18	
0x002F	0x2802	GATT Include Declaration	23:00:27:00:0F:18	
0x0030	0x2803	GATT Characteristic Declaration	02:31:00:4A:2A	
0x0031	0x2A4A	HID Information		↑ ←
0x0032	0x2803	GATT Characteristic Declaration	04:33:00:4C:2A	
0x0033	0x2A4C	HID Control Point		↓ ←
0x0034	0x2803	GATT Characteristic Declaration	06:35:00:4E:2A	
0x0035	0x2A4E	Protocol Mode		↑↓ ←
0x0036	0x2803	GATT Characteristic Declaration	02:37:00:4B:2A	
0x0037	0x2A4B	Report Map	05:01:09:02:A1:01:85:01:09:01:A1:00:05:09:19:01:29:03:15:00:25:01	← USB HID Report Descriptor
0x0038	0x2907	External Report Reference	19:2A	↑
0x0039	0x2803	GATT Characteristic Declaration	12:3A:00:4D:2A	
0x003A	0x2A4D	Report		↑↑ ←
0x003B	0x2902	Client Characteristic Configuration	9E:00	↑↓
0x003C	0x2908	Report Reference	01:01	↑
0x003D	0x2803	GATT Characteristic Declaration	12:3E:00:4D:2A	
0x003E	0x2A4D	Report		↑↑ ←
0x003F	0x2902	Client Characteristic Configuration	9E:00	↑↓
0x0040	0x2908	Report Reference	02:01	↑
0x0041	0x2803	GATT Characteristic Declaration	12:42:00:4D:2A	
0x0042	0x2A4D	Report		↑↑ ←
0x0043	0x2902	Client Characteristic Configuration	9E:00	↑↓
0x0044	0x2908	Report Reference	03:01	↑
0x0045	0x2803	GATT Characteristic Declaration	0E:46:00:4D:2A	
0x0046	0x2A4D	Report		↑↓ ←
0x0047	0x2908	Report Reference	00:02	↑
0x0048	0x2803	GATT Characteristic Declaration	12:49:00:22:2A	
0x0049	0x2A22	Boot Keyboard Input Report		↑↑ ←
0x004A	0x2902	Client Characteristic Configuration	9E:00	↑↓
0x004B	0x2803	GATT Characteristic Declaration	0E:4C:00:32:2A	
0x004C	0x2A32	Boot Keyboard Output Report		↑↓ ←
0x004D	0x2803	GATT Characteristic Declaration	12:4E:00:33:2A	
0x004E	0x2A33	Boot Mouse Input Report		↑↑ ←
0x004F	0x2902	Client Characteristic Configuration	9E:00	↑↓
0x0050	0x2803	GATT Characteristic Declaration	0A:51:00:4D:2A	
0x0051	0x2A4D	Report		↑↓ ←
0x0052	0x2908	Report Reference	00:03	↑

IN = To host

OUT = To device

Typical dataflow



* Application turns off polling when idle, then restarts on Acc ISR if applicable

What does a report contain

0x0039	0x2803	GATT Characteristic Declaration	12:3A:00:4D:2A
0x003A	0x2A4D	Report	

- Mouse, 4 bytes data
 - Buttons
 - MickeysX
 - MickeysY
 - Wheel

0x0041	0x2803	GATT Characteristic Declaration	12:42:00:4D:2A
0x0042	0x2A4D	Report	

- Consumer Control, 2 bytes
 - Bitmap of command

0x003D	0x2803	GATT Characteristic Declaration	12:3E:00:4D:2A
0x003E	0x2A4D	Report	

- Keyboard, 8 bytes
 - Modifier
 - Reserved
 - Keycode 1..6
- Significance determined by which characteristic report sent via.

```
static void hidMouseSendReport( uint8 mouseStates, int8 mickeysX, int8 mickeysY )
{
    uint8 buf[HID_MOUSE_IN_RPT_LEN];

    // No need to include Report Id
    buf[0] = mouseStates;           // Buttons
    buf[1] = mickeysX;             // X
    buf[2] = mickeysY;             // Y
    buf[3] = 0;                    // Wheel

    HidDev_Report( HID_RPT_ID_MOUSE_IN, HID_REPORT_TYPE_INPUT,
                   HID_MOUSE_IN_RPT_LEN, buf );
}

static void hidCCSendReport( uint8 cmd, bool keyPressed, uint8 keyRepeated )
{
    // Only send the report if something meaningful to report
    if ( keyRepeated == 0 )
    {
        uint8 buf[HID_CC_IN_RPT_LEN] = { 0, 0 };

        // No need to include Report Id
        if ( keyPressed )
        {
            hidCCBuildReport( buf, cmd );
        }

        HidDev_Report( HID_RPT_ID_CC_IN, HID_REPORT_TYPE_INPUT,
                       HID_CC_IN_RPT_LEN, buf );
    }
}

static void hidKeyboardSendReport( uint8 modifiers, uint8 keycode )
{
    static uint8 prevKeyCode = HAL_KEY_CODE_NOKEY;
    static uint8 prevModifiers = HID_KEYBOARD_RESERVED;

    // Only send the report if something meaningful to report
    if ( ( prevKeyCode != keycode ) || ( prevModifiers != modifiers ) )
    {
        uint8 buf[HID_KEYBOARD_IN_RPT_LEN];

        // No need to include Report Id
        buf[0] = modifiers;           // Modifier keys
        buf[1] = 0;                  // Reserved
        buf[2] = keycode;             // Keycode 1
        buf[3] = 0;                  // Keycode 2
        buf[4] = 0;                  // Keycode 3
        buf[5] = 0;                  // Keycode 4
        buf[6] = 0;                  // Keycode 5
        buf[7] = 0;                  // Keycode 6

        HidDev_Report( HID_RPT_ID_KEY_IN, HID_REPORT_TYPE_INPUT,
                       HID_KEYBOARD_IN_RPT_LEN, buf );
    }
}
```

iOS BLE App Development

Introduction

Developing for iOS

- Development happens in Xcode
- Xcode runs on OSX [Mountain] Lion
- Xcode and simulator is free
 - BLE requires physical iOS device
- Physical device download requires Apple Developer Account
 - Costs 99\$ per year at developer.apple.com/programs/ios



1. Develop

Develop your application with the iOS SDK and a wealth of technical resources in the iOS Dev Center.
[Learn more ▶](#)



2. Test

Test and debug your code on iPad, iPhone and iPod touch to finalize your applications.
[Learn more ▶](#)



3. Distribute

Distribute your apps on the App Store and reach millions of iPad, iPhone, and iPod touch users.
[Learn more ▶](#)

87

Using Xcode

The screenshot shows the Xcode interface with the following highlights:

- Run button:** Circled in red at the top left of the toolbar.
- Scheme dropdown:** Shows "SensorTagEX > iOS Device", circled in red at the top center.
- Project Navigator:** Shows the project structure under "SensorTagEX".
 - CoreBluetooth.framework:** Circled in red.
 - SensorTagEX group:** Contains "Sensors.h", "Sensors.m", "SensorTagApplicationViewController.h", and "SensorTagApplicationViewController.m". "SensorTagApplicationViewController.m" is selected and highlighted with a yellow oval.
 - BLEUtility group:** Contains "deviceCellTemplate.h", "deviceCellTemplate.m", "deviceSelector.h", "deviceSelector.m", "BLEDevice.h", "BLEDevice.m", "BLEUtility.h", and "BLEUtility.m". "BLEUtility.m" is highlighted with a green oval.
 - Supporting Files:** Contains "Icon@2x-114.png" and "Icon-57.png".
 - Frameworks:** Contains "CoreBluetooth.framework".
 - Products:** Contains "SensorTagEX".
- Code Editor:** Displays the "SensorTagApplicationViewController.m" file with the following code snippet:

```
247 }
248 -(void) configureSensorTag {
249     // Configure sensor tag, turning on Sensors and setting update period for sensors etc ...
250
251     if ([[self sensorEnabled:@"Ambient temperature active"] || ([self sensorEnabled:@"IR temperature
252         active"])] {
253         // Enable Temperature sensor
254         CBUUID *sUUID = [CBUUID UUIDWithString:[self.d.setupData valueForKey:@"IR temperature service
255             UUID"]];
256         CBUUID *cUUID = [CBUUID UUIDWithString:[self.d.setupData valueForKey:@"IR temperature config
257             UUID"]];
258         uint8_t data = 0x01;
259         [BLEUtility writeCharacteristic:self.d.p sUUID:sUUID cUUID:cUUID data:[NSData
260             dataWithBytes:&data length:1]];
261         cUUID = [CBUUID UUIDWithString:[self.d.setupData valueForKey:@"IR temperature data UUID"]];
262         [BLEUtility setNotificationForCharacteristic:self.d.p sUUID:sUUID cUUID:cUUID enable:YES];
263
264         if ([self sensorEnabled:@"Ambient temperature active"]) [self.sensorsEnabled
265             addObject:@"Ambient temperature"];
266         if ([self sensorEnabled:@"IR temperature active"]) [self.sensorsEnabled addObject:@"IR
267             temperature"];
268
269     if ([self sensorEnabled:@"Accelerometer active"]){
270         CBUUID *sUUID = [CBUUID UUIDWithString:[self.d.setupData valueForKey:@"Accelerometer service
271             UUID"]];
272         CBUUID *cUUID = [CBUUID UUIDWithString:[self.d.setupData valueForKey:@"Accelerometer config
273             UUID"]];
274         CBUUID *pUUID = [CBUUID UUIDWithString:[self.d.setupData valueForKey:@"Accelerometer period
275             UUID"]];
276     }
277 }
```

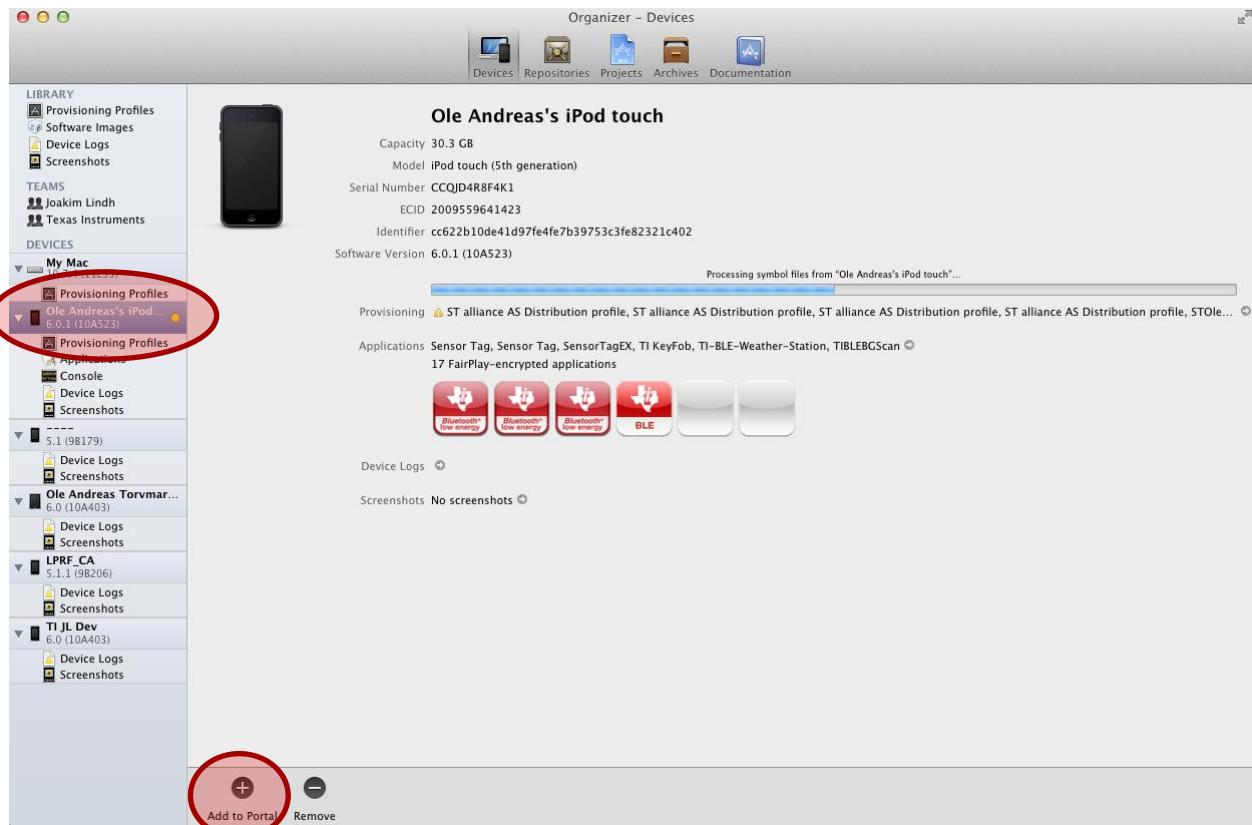
iOS apps are separated* into Model, View and Controller (MVC).

In our example, CoreBluetooth callbacks are in the **ViewControllers**, and some CoreBt functions are wrapped for convenience in **BLEUtility**

*ideally

Adding a device to Xcode

- Open Organizer (under Window)
- Click on device
- Click add to Portal
- Play button works



The iOS Source Code

- The communication is asynchronous between app and iOS's CoreBluetooth
- We issue commands to CBCentralManager and CBPeripheral
 - [BLEUtility writeCharacteristic ...] →
 - [peripheral.writeValue:data ...];
- Then act as *delegate* for CBCentralManager
 - centralManagerDidUpdateState
 - didDiscoverPeripheral
 - didConnectPeripheral
- And for CBPeripheral
 - didDiscoverServices
 - didUpdateValueForCharacteristic
 - didWriteValueForCharacteristic
- *Delegates implement callbacks according to a protocol.* A design pattern used in iOS.

```
1 // BLEUtility.m
2 // Created by Ole Andreas Torvmark on 9/22/12.
3 // Copyright (c) 2012 Texas Instruments. All rights reserved.
4 //
5 #import "BLEUtility.h"
6 @implementation BLEUtility
7
8 +(void)writeCharacteristic:(CBPeripheral *)peripheral sUUID:(NSString *)sUUID cUUID:(NSString *)cUUID
9   data:(NSData *)data {
10   // Sends data to BLE peripheral to process HID and send EHIF command to PC
11   for ( CService *service in peripheral.services ) {
12     if ([service.UUID isEqual:[CBUUID UUIDWithString:sUUID]]) {
13       for ( CBCharacteristic *characteristic in service.characteristics ) {
14         if ([characteristic.UUID isEqual:[CBUUID UUIDWithString:cUUID]]) {
15           /* EVERYTHING IS FOUND, WRITE characteristic ! */
16           [peripheral writeValue:data forCharacteristic:characteristic type:
17             CBCharacteristicWriteWithResponse];
18         }
19       }
20     }
21   }
22 }
23
24
25 +(void)writeCharacteristic:(CBPeripheral *)peripheral sCUUID:(CBUUID *)sCUUID cCUUID:(CBUUID *)
26   cCUUID data:(NSData *)data {
27   // Sends data to BLE peripheral to process HID and send EHIF command to PC
28   for ( CService *service in peripheral.services ) {
29     if ([service.UUID isEqual:sCUUID]) {
30       for ( CBCharacteristic *characteristic in service.characteristics ) {
31         if ([characteristic.UUID isEqual:cCUUID]) {
32           /* Everything is found, WRITE characteristic ! */
33           [peripheral writeValue:data forCharacteristic:characteristic type:
34             CBCharacteristicWriteWithResponse];
35         }
36       }
37     }
38   }
39 }
40
41
42 +(void)readCharacteristic:(CBPeripheral *)peripheral sUUID:(NSString *)sUUID cUUID:(NSString *)cUUID {
43   for ( CService *service in peripheral.services ) {
44     if ([service.UUID isEqual:[CBUUID UUIDWithString:sUUID]]) {
45       for ( CBCharacteristic *characteristic in service.characteristics ) {
46         if ([characteristic.UUID isEqual:[CBUUID UUIDWithString:cUUID]]) {
47           /* Everything is found, read characteristic ! */
48           [peripheral readValueForCharacteristic:characteristic];
49         }
50       }
51     }
52   }
53 }
54
55
56 +(void)readCharacteristic:(CBPeripheral *)peripheral sCUUID:(CBUUID *)sCUUID cCUUID:(CBUUID *)
57   cCUUID {
58   for ( CService *service in peripheral.services ) {
59     if ([service.UUID isEqual:sCUUID]) {
60       for ( CBCharacteristic *characteristic in service.characteristics ) {
61         if ([characteristic.UUID isEqual:cCUUID]) {
62           /* Everything is found, read characteristic ! */
63         }
64       }
65     }
66   }
67 }
```

The iOS BLE Limitations

- Uses Private resolvable address instead of fixed IEEE address
 - Cannot use whitelist towards iOS devices
 - Cannot use directed advertisements towards iOS devices
- Connection parameter limitations
 - Connection parameters must be updated by slave device(!)
 - Interval must be higher than 20ms; combined with latency, less than 2s
 - Supervisor timeout <= 6s
 - Slave latency >= 4
- BLE data hidden from the user:
 - IEEE address of peripheral
 - Characteristic handles
 - Descriptors
 - Connection parameters

References

- Getting started with iOS development
 - Setting up account and development environment (Xcode)
 - Developing, installing and testing an app
 - Distributing on the appstore (pretty complicated)
 - <https://developer.apple.com/programs/ios/gettingstarted/>
- Objective-C and iOS
 - If you are familiar with C:
 - <http://en.wikipedia.org/wiki/Objective-C> (very helpful)
 - Your first iOS app
 - http://developer.apple.com/library/ios/#documentation/iphone/conceptual/iphone101/Articles/00_Introduction.html
 - BLE Blog by Ole Andreas Torvmark (TI Contractor in Oslo)
 - <http://ble.stalliance.no/>
- Sensortag example project
 - <http://www.ti.com/sensortag-wiki>

Frequently Asked Questions

FAQ / known issues

- Using IAR v.8.20, compile/linker error
 - Answer: We recommend 8.10.4 or 8.11.4.bacause version 8.20 generates an extra (unnecessary) instruction. Workaround: In hal_sleep.c, remove `PCON = halSleepPconValue;` in `halSetSleepMode` replacing it with the line `asm("MOV 0x87,halSleepPconValue");`
- Porting projects from BLEv1.2.1 to BLEv1.3
 - Answer: See porting guide at wiki

Thank You

- TI *Bluetooth* low energy Solutions
<http://www.ti.com/ble>
- TI *Bluetooth* low energy Wiki
<http://www.ti.com/ble-wiki>
- TI BLE-Stack
<http://www.ti.com/ble-stack>
- TI *Bluetooth* low energy E2E Forum
<http://www.ti.com/ble-forum>
- TI *Bluetooth* low energy SensorTag
<http://www.ti.com/sensortag>

