ECEN 5823-001
Internet of Things Embedded Firmware

Lecture #8

20 September 2018

Be Boulder.

University of Colorado Boulder

BLE assignment demo

Agenda

- Energy Mode Assignment feedback
- Class Announcements
- Reading Assignment
- BLE Assignment
- Bluetooth Classic (continued)

Energy Mode Assignment Feedback

- If the code is non-functional, the questions cannot be answered. Why?
- Using “magic numbers.” Is this best coding practices?
- I have set up a separate assignment/dropbox for Energy Mode Assignment resubmissions – due Friday the 21st at 11:59pm
  - Answers to the questions must be included
- This assignment is for anyone that would like to resubmit their Energy Mode Assignment for regrading. Below is the grading policy for this re-submission.
  - Maximum score is 8.5 pts
  - For every point you score above your first submission, you will gain 0.5 pts to the maximum award of 8.5 pts.
  - Example 1: If initial score is 0 and the re-submission score is 10, your final score will be 5.
  - Example 2: If initial score is 8 and the re-submission score is 10, the final score will not be 9, but 8.5.

Class Announcements

- Quiz #4 is due at 11:59 on Sunday, September 23rd, 2018
- Homework #1: Energy Mode resubmissions due by 11:59pm on Friday, September the 21st
- Homework #2: I2C Load Management Assignment is due on Sunday, September 23rd, at 11:59pm

Bluetooth Health Temp Service Assignment

ECEN 5823
HTP Assignment
Fall 2018

Objective: To take the temperature measured by the Si7021 and communicate it via BLE to the Silicon Labs' Blue Gecko iPhone or Android phone app.

Note: This assignment will begin with the completed I2C temp sensor load power management assignment.

Due (submission): Saturday, September 29th, 2018 at 11:59pm

Instructions:

1. Make any changes required to the I2C temp sensor load power management assignment.

Using the Bluetooth Stack on the Blue Gecko

- What drives Low Energy Firmware?

- It saves energy by allowing which peripheral to be turned off as much as possible?

Using the Bluetooth Stack on the Blue Gecko

- With Bluetooth being a long energy radio protocol, its code is based on ….

Bluetooth Events

- Enabling the CPU to be off as much as possible with the following ….

evt = **gecko_wait_event();**

Using the Bluetooth Stack on the Blue Gecko

- evt = **gecko_wait_event();**
  - The Bluetooth stack must control the level of sleep to enable it to wake up on schedule to handle Bluetooth events
  - It sets a timer and only returns if a Bluetooth stack event has occurred
    - Similar to the Enter_Sleep() routine which only returns when an interrupt has occured

Using the Bluetooth Stack on the Blue Gecko

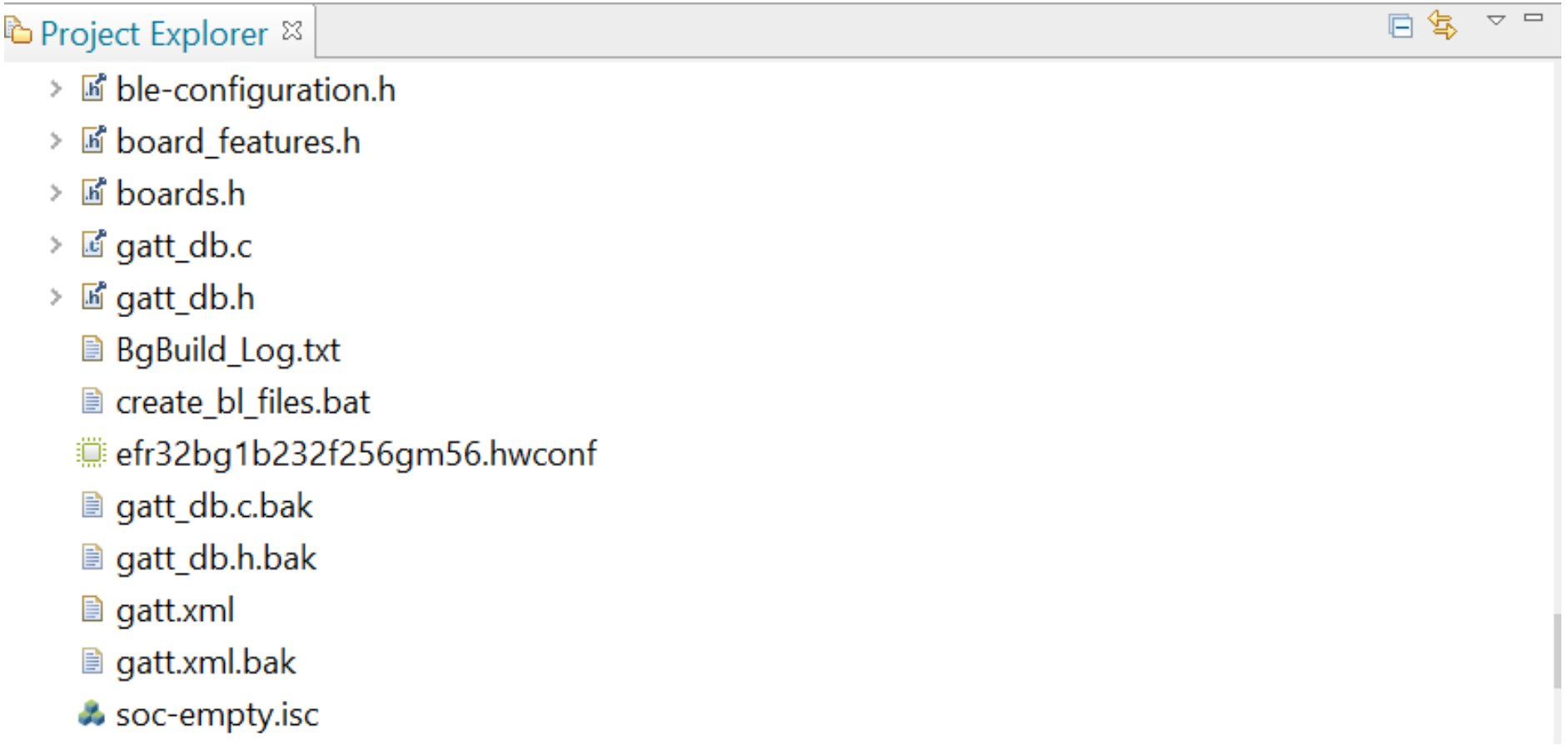
- evt = **gecko_wait_event();**
  - This will replace your Enter_Sleep() routine that you have created and being used
    - For now, in your code, you can simply comment out the code inside your Enter_Sleep() routines. We will discuss how to manage this in more detail in Thursday's lecture.

Using the Bluetooth Stack on the Blue Gecko

- The Bluetooth Stack requires the following resources:
  - RAM memory – always needed, must be retained in sleep modes
  - RTCC – always needed for sleep timing
  - LDMA – used for handling BGAPI commands in NCP mode
  - UART – used for receiving/transmitting BGAPI commands/responses in NCP mode
  - PROTIMER – used for protocol timing when receiving/transmitting packets
  - RADIO – used for receiving/transmitting packets

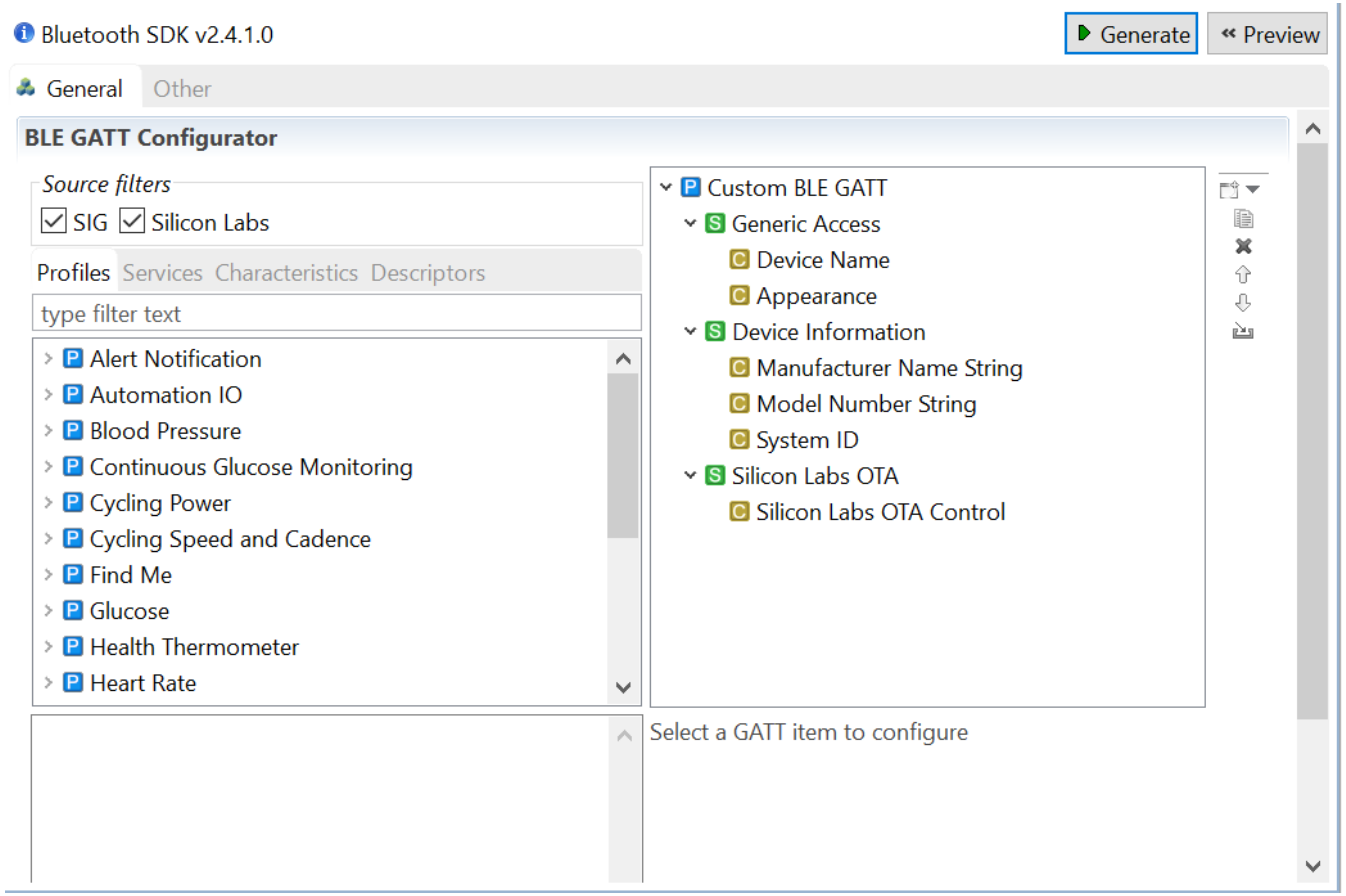
Using the Bluetooth Stack on the Blue Gecko

- How to install / specify a Bluetooth service or profile?
  - By its .isc file



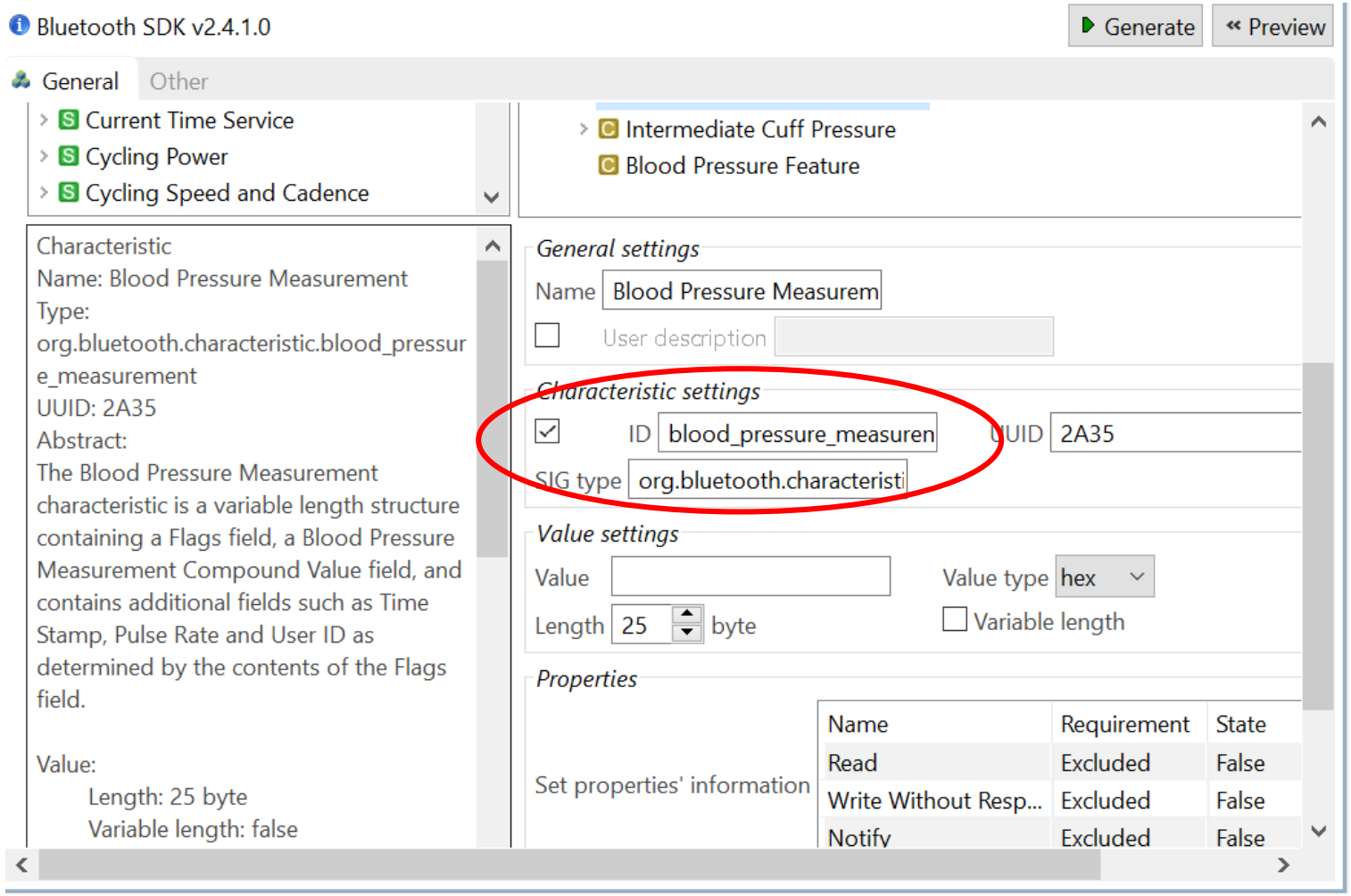
Using the Bluetooth Stack on the Blue Gecko

- The .isc file will open and provide the following gui interface



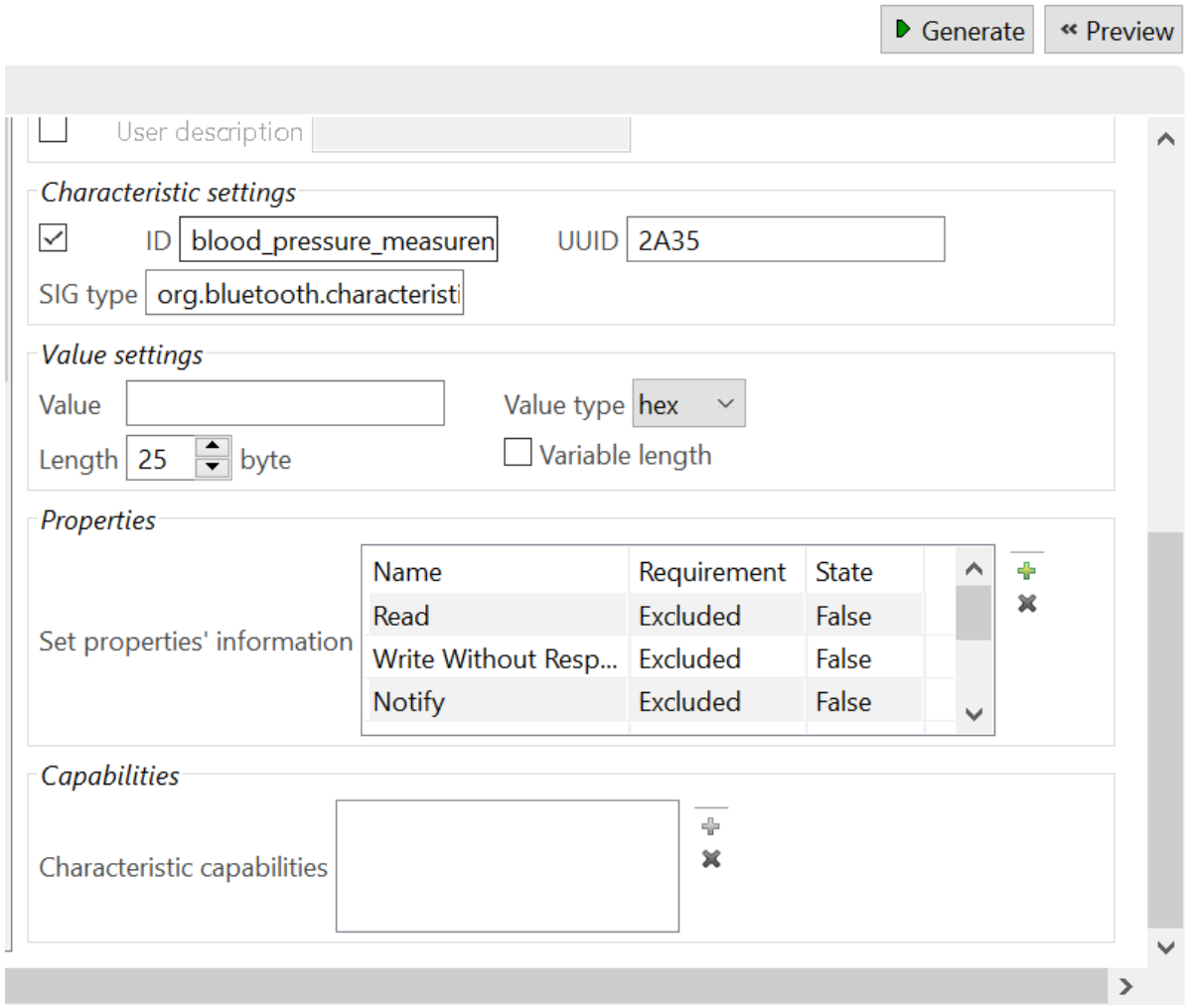
Using the Bluetooth Stack on the Blue Gecko

- How do I know how to address an attribute or characteristic?
  - By its characteristic ID

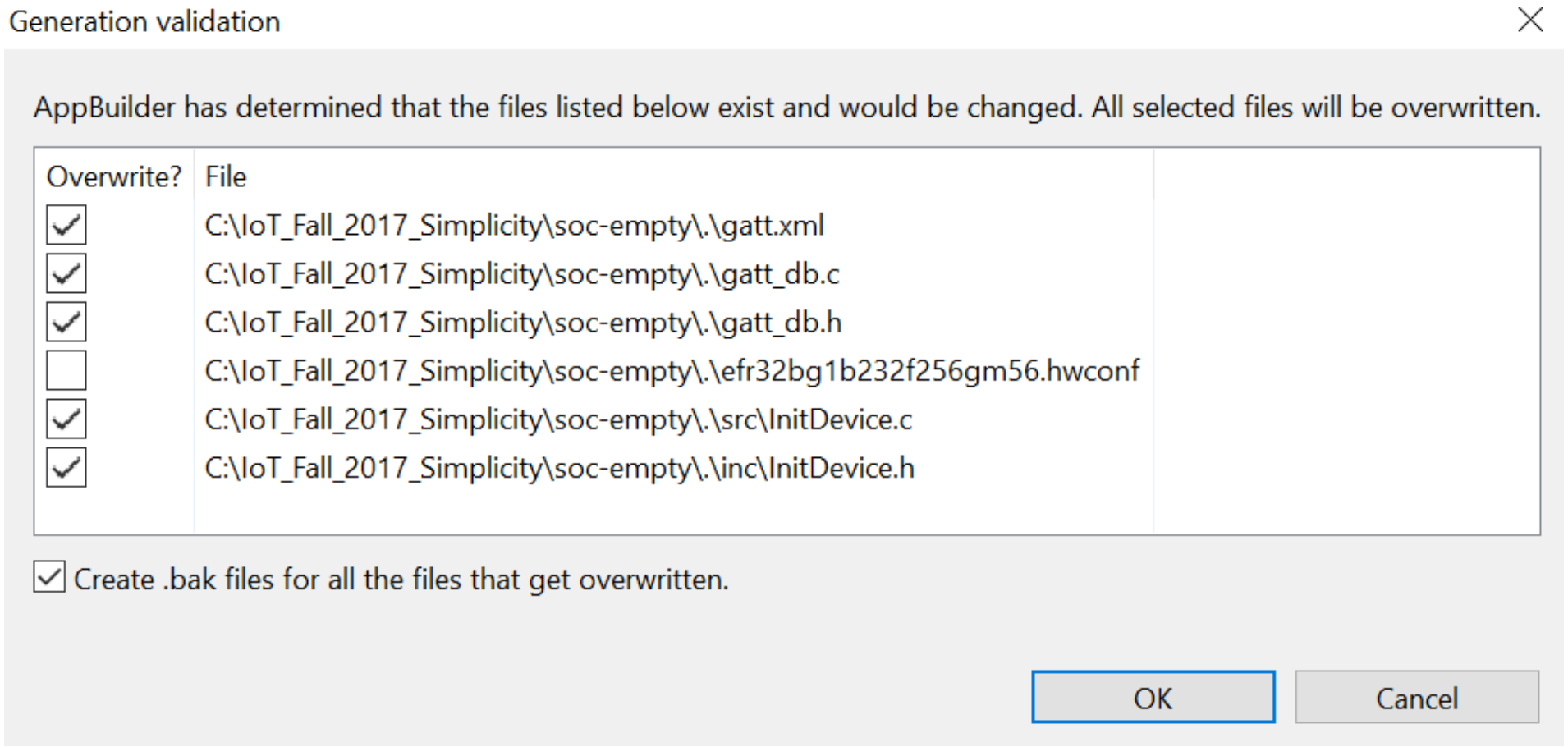


Using the Bluetooth Stack on the Blue Gecko

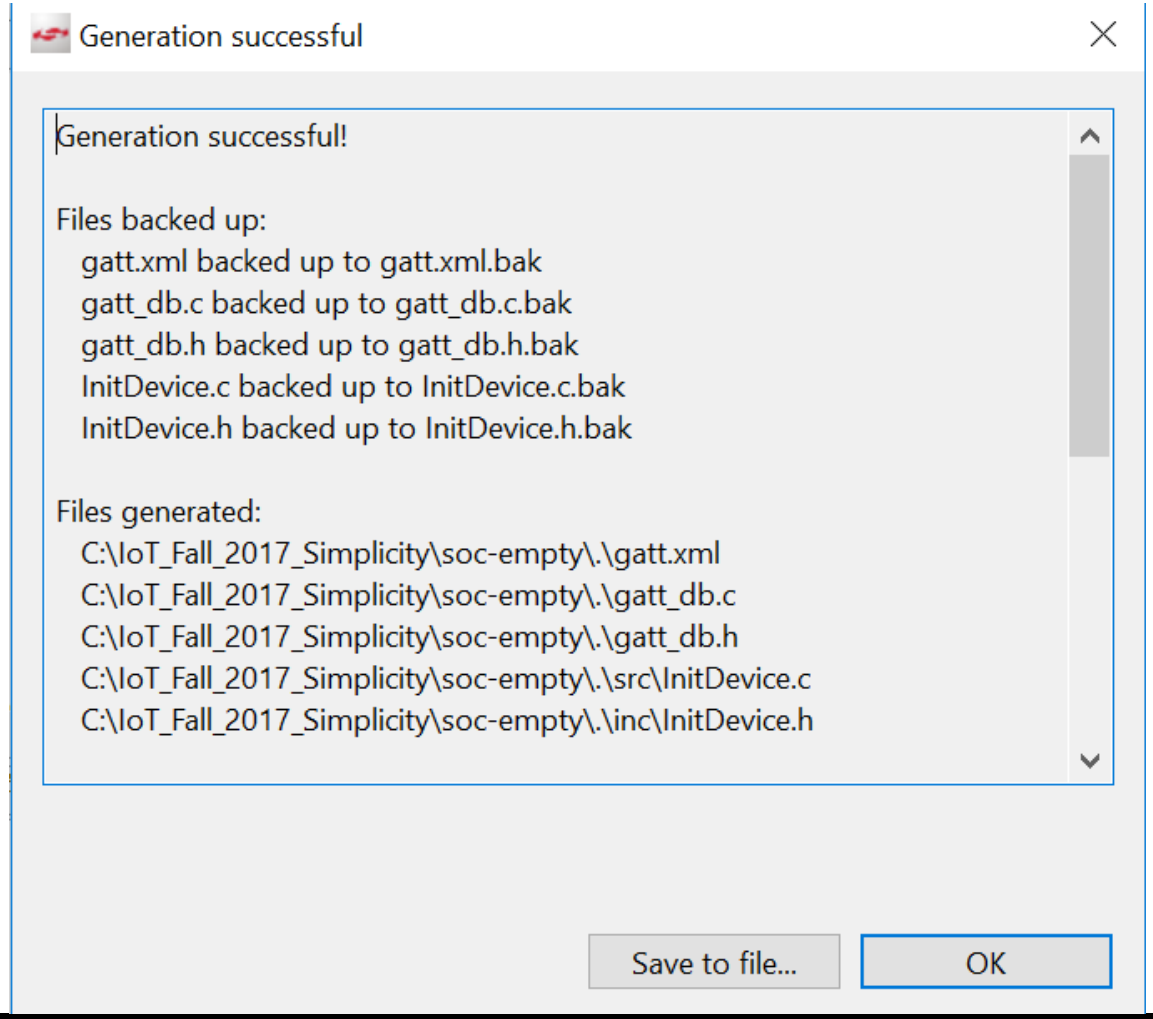
- From this GUI interface, you can set:
  - Characteristic ID
  - UUID
  - The initial value
  - Value type
  - Characteristic length
  - Its properties
- Then you must Generate the service/profile



Using the Bluetooth Stack on the Blue Gecko

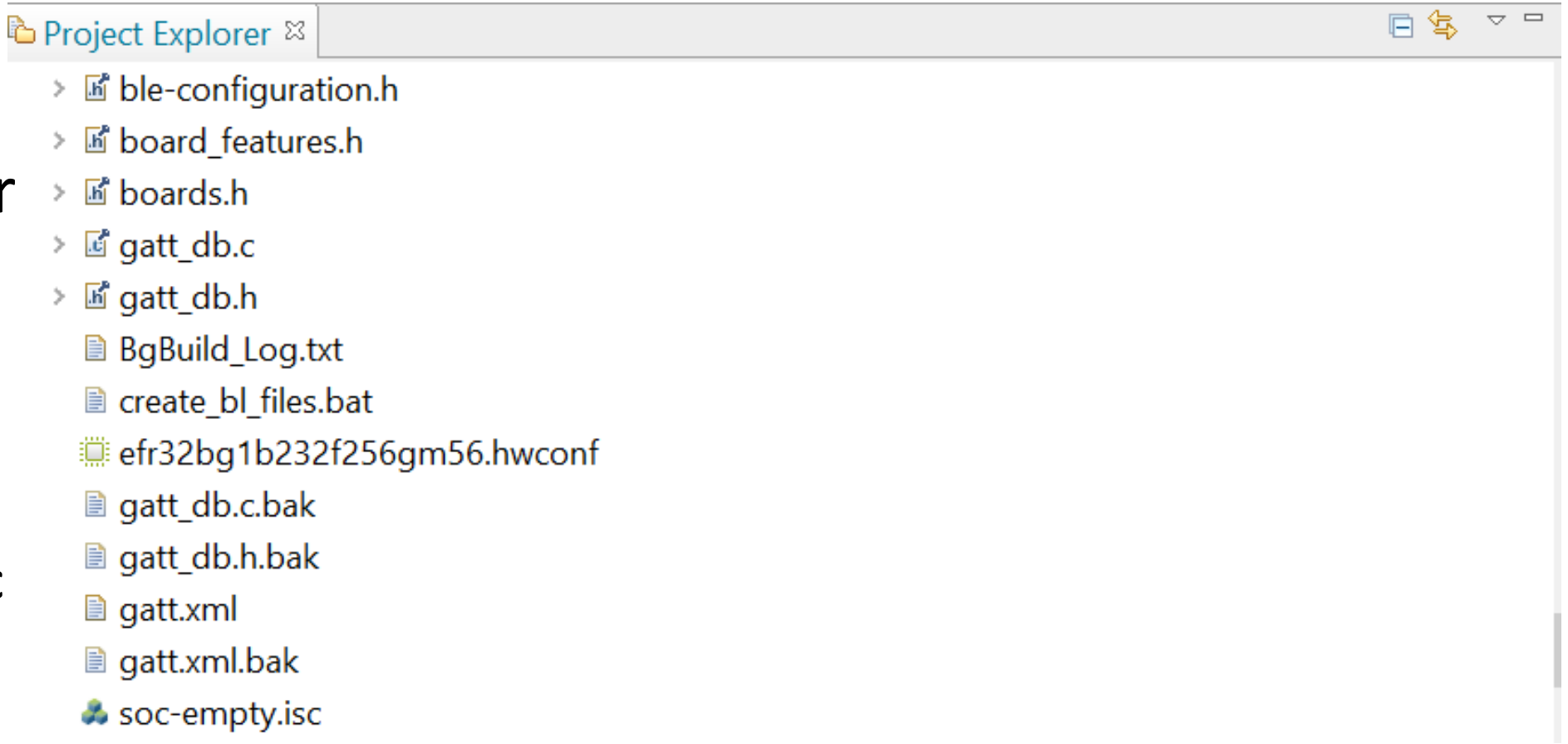


Using the Bluetooth Stack on the Blue Gecko



Using the Bluetooth Stack on the Blue Gecko

- Where do I find the characteristic or attribute address once it has been generated?
  - In its gatt_db.c file



Using the Bluetooth Stack on the Blue Gecko

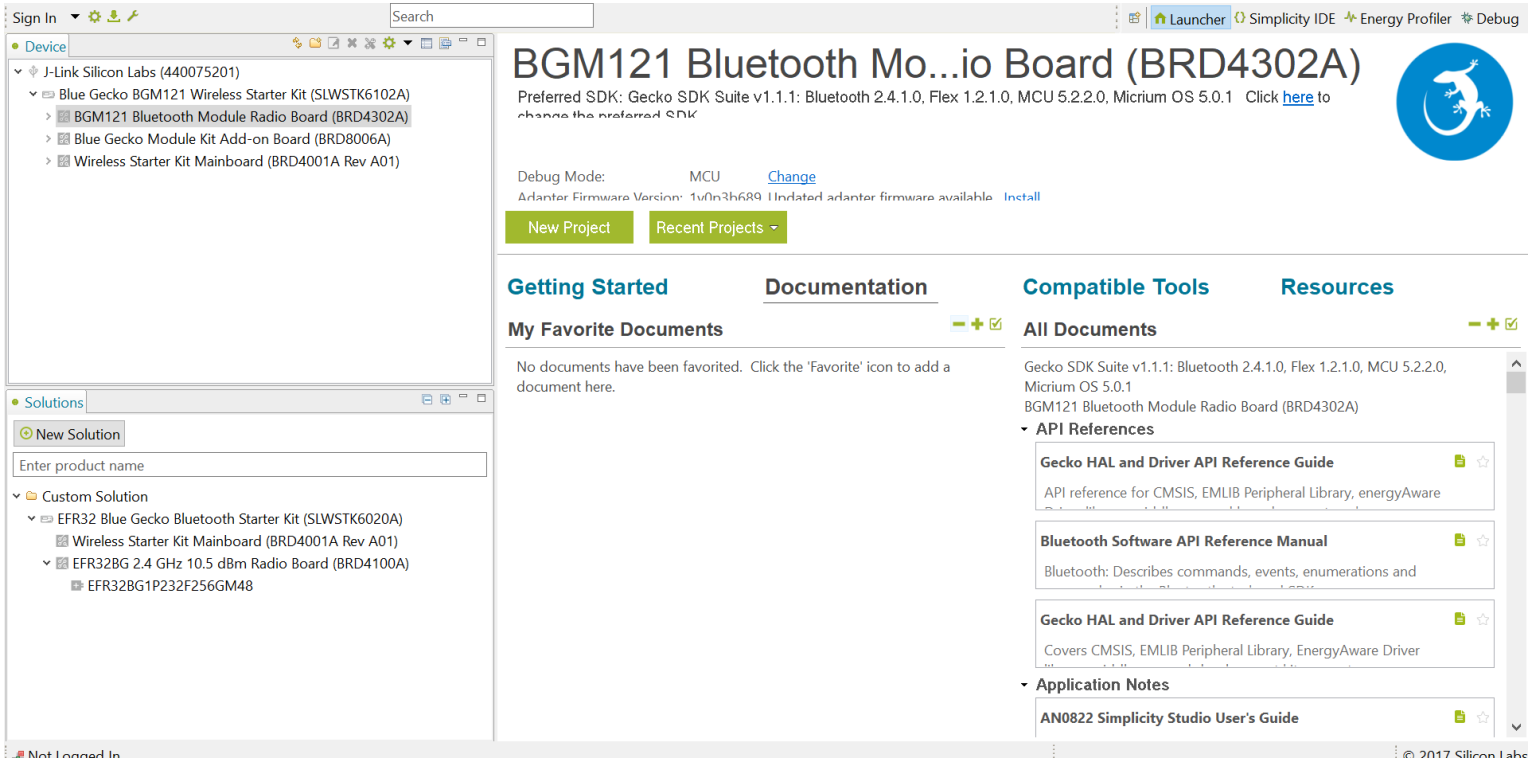
```
// Copyright 2017 Silicon Laboratories, Inc.
//
//

/***********************************************************************
 * Autogenerated file, do not edit.
 ***********************************************************************/

#ifndef GATT_DB_H
#define GATT_DB_H

#include "bg_gattdb_def.h"

extern const struct bg_gattdb_def bg_gattdb_data;

#define gattdb_service_changed_char              3
#define gattdb_device_name                       7
#define gattdb_ota_control                      19
#define gattdb_blood_pressure_measurement          22
#define gattdb_intermediate_cuff_pressure          25
#define gattdb_blood_pressure_feature           28

#endif
```

Using the Bluetooth Stack on the Blue Gecko

- How does the application know when to respond to an even? An Interrupt?
- It knows to respond to an event because the following code returns:
  - evt = **gecko_wait_event();**
    - It sets a timer and only returns if a Bluetooth stack event has occurred
- Responding to the events will not be an ISR, but a switch statement in the applications while (1) routine
  - Remember that BLE responds to a single event, not retaining state

Using the Bluetooth Stack on the Blue Gecko

- The next question is how to determine which case statement will be executed



Using the Bluetooth Stack on the Blue Gecko

- Through the Bluetooth API reference manual, you can locate:
  - the event ID
  - data structure

2.5.2.1 evt_gatt_server_attribute_value

This event indicates that the value of an attribute in the local GATT database has been changed by a remote GATT client. Parameter att_opcode describes which GATT procedure was used to change the value.

Table 2.123. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x07	lolen	Minimum payload length
2	0x0a	class	Message class: Generic Attribute Profile Server
3	0x00	method	Message ID
4	uint8	connection	Connection handle
5-6	uint16	attribute	Attribute Handle
7	uint8	att_opcode	Attribute opcode which informs the procedure from which attribute the value was received
8-9	uint16	offset	Value offset
10	uint8array	value	Value

C Functions

```
/* Event id */
gecko_evt_gatt_server_attribute_value_id

/* Event structure */
struct gecko_msg_gatt_server_attribute_value_evt_t
{
  uint8 connection;,
  uint16 attribute;,
  uint8 att_opcode;,
  uint16 offset;,
  uint8array value;
};
```

Using the Bluetooth Stack on the Blue Gecko

- Events include a header and data
- Header information is accessed by evt->header
- Data is accessed by
  - evt->data.event_id_minus_the_gecko_msg_in_front_and_minus_evt_t_at_the_end.data_element

Using the Bluetooth Stack on the Blue Gecko

C Functions

```
/* Event id */
gecko_evt_gatt_server_characteristic_status_id

/* Event structure */
struct gecko_msg_gatt_server_characteristic_status_evt_t
{
  uint8 connection;,
  uint16 characteristic;,
  uint8 status_flags;,
  uint16 client_config_flags;
};
```

characterist_status = evt->data.gatt_server_characterist_status.status_flags;

Using the Bluetooth Stack on the Blue Gecko

```
While (1) {
    evt = gecko_wait_event();
    switch (BGLIB_MSG_ID(evt->header)){
        case gecko_evt_system_boot_id:
            Break;
        case gecko_evt_le_connection_closed_id:
            Break;
        default:
            Break;
    }
}
```

Using the Bluetooth Stack on the Blue Gecko

- case gecko_evt_system_boot_id:
  - Used for the first time to initialize the Bluetooth stack and specify the type of advertising

```
 /* Set advertising parameters. The third parameter '7' sets
advertising on all channels. */
        gecko_cmd_le_gap_set_adv_parameters(Bt_Min_Adv,
Bt_Max_Adv, Bt_Adv_Channels);

 /* Start general advertising and enable connections. */
       gecko_cmd_le_gap_set_mode(le_gap_general_discoverable,
le_gap_undirected_connectable);
```

Using the Bluetooth Stack on the Blue Gecko

- case gecko_evt_le_connection_closed_id:
  - Used to reset the Bluetooth device to begin advertising
-

```
case gecko_evt_le_connection_closed_id:
        /* Check if need to boot to dfu mode */
        if (boot_to_dfu) {
          /* Enter to DFU OTA mode */
          gecko_cmd_system_reset(2);
        } else {
          /* Restart advertising after client has disconnected */
          gecko_cmd_le_gap_set_mode(le_gap_general_discoverable,
le_gap_undirected_connectable);
        }
        break;
```

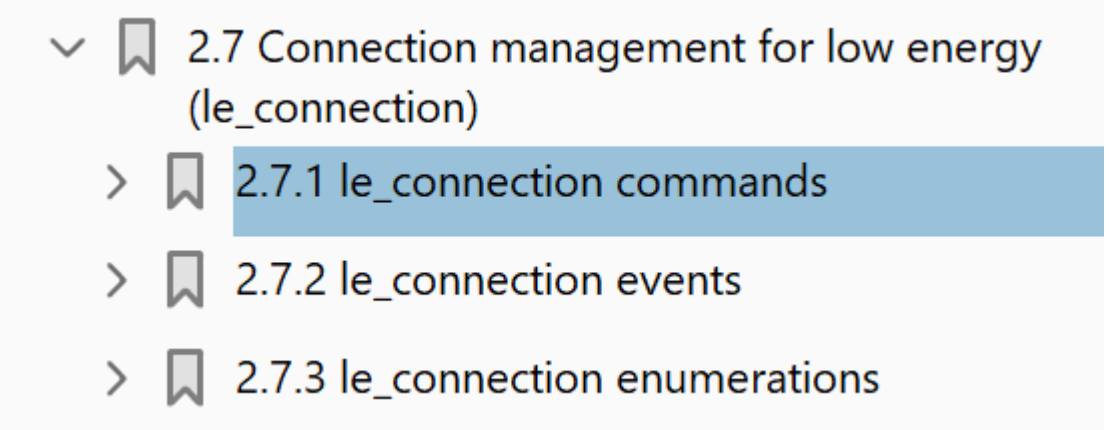
Using the Bluetooth Stack on the Blue Gecko

- case gecko_evt_le_connection_closed_id:
  - Used to reset the Bluetooth device to begin advertising
-

```
case gecko_evt_le_connection_closed_id:
        /* Check if need to boot to dfu mode */
        if (boot_to_dfu) {
          /* Enter to DFU OTA mode */
          gecko_cmd_system_reset(2);
        } else {
          /* Restart advertising after client has disconnected */
          gecko_cmd_le_gap_set_mode(le_gap_general_discoverable,
le_gap_undirected_connectable);
        }
        break;
```

Using the Bluetooth Stack on the Blue Gecko

- The Bluetooth stacks all specifies commands, cmd, and responses
- Responses can be read back immediately after issuing the command



Using the Bluetooth Stack on the Blue Gecko

- Command:
  - gecko_cmd_le_connection_disable_slave_latency(uint8 connection, uint8 disable);
- Response:
  - evt->data.rsp_le_connection_disable_slave_latency.result

```
/* Function */
struct  gecko_msg_le_connection_disable_slave_latency_rsp_t  *gecko_cmd_le_connection_disable_slave_latency(uint
8 connection, uint8 disable);

/* Response id */
gecko_rsp_le_connection_disable_slave_latency_id

/* Response structure */
struct gecko_msg_le_connection_disable_slave_latency_rsp_t
{
  uint16 result;
};
```

Using the Bluetooth Stack on the Blue Gecko

- You can send a command to send a notification and indication by using the appropriate Bluetooth Stack API
- You will be using this command to send the temperature via the Health Temperature Service

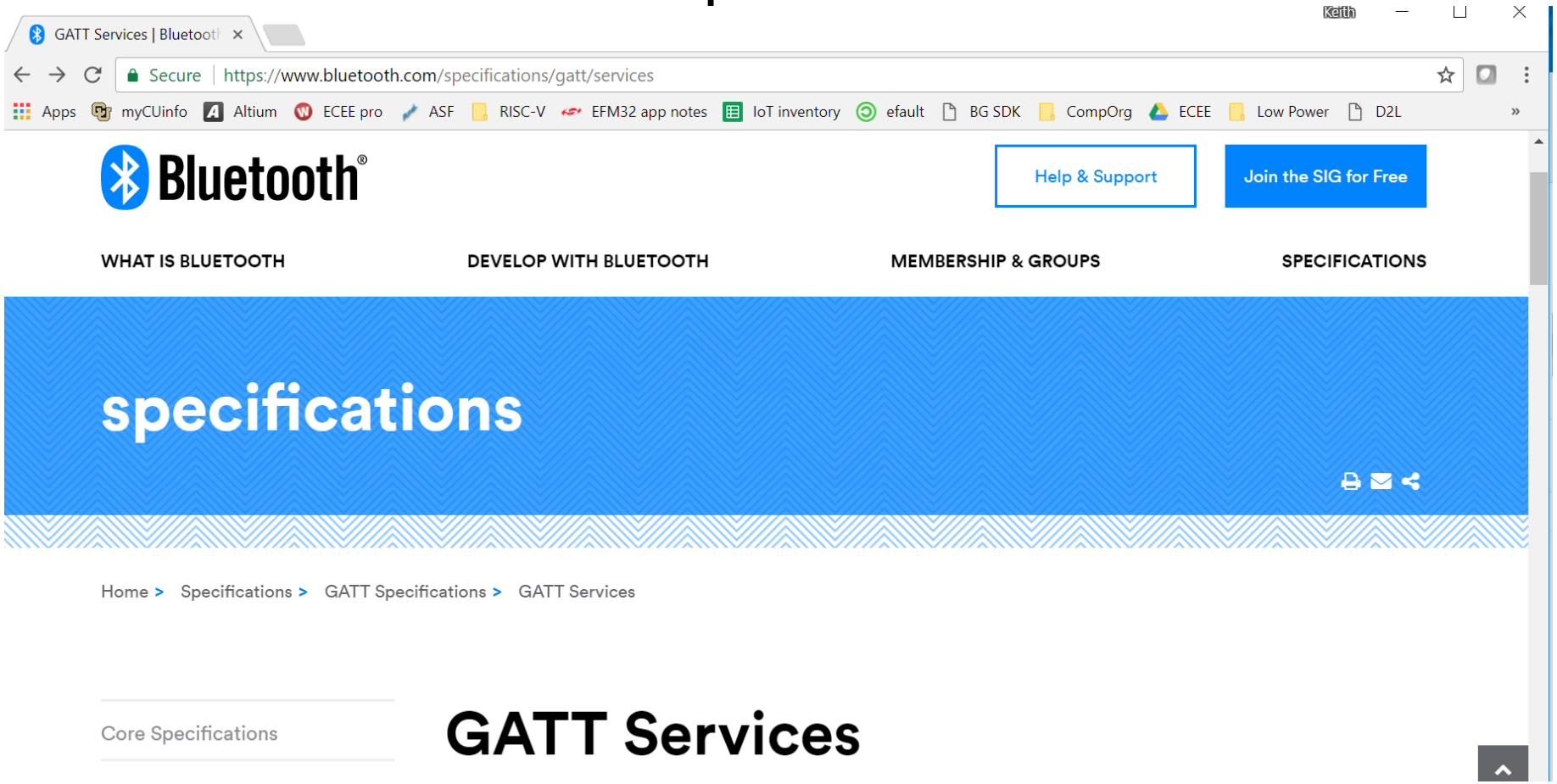
2.5.1.4 cmd_gatt_server_send_characteristic_notification

This command can be used to send notifications or indications to a remote GATT client. At most ATT_MTU - 3 amount of data can be sent once. Notification or indication is sent only if the client has enabled them by setting the corresponding flag to the Client Characteristic Configuration descriptor. A new indication cannot be sent before a confirmation from the GATT client is first received. The confirmation is indicated by gatt_server_characteristic_status event.

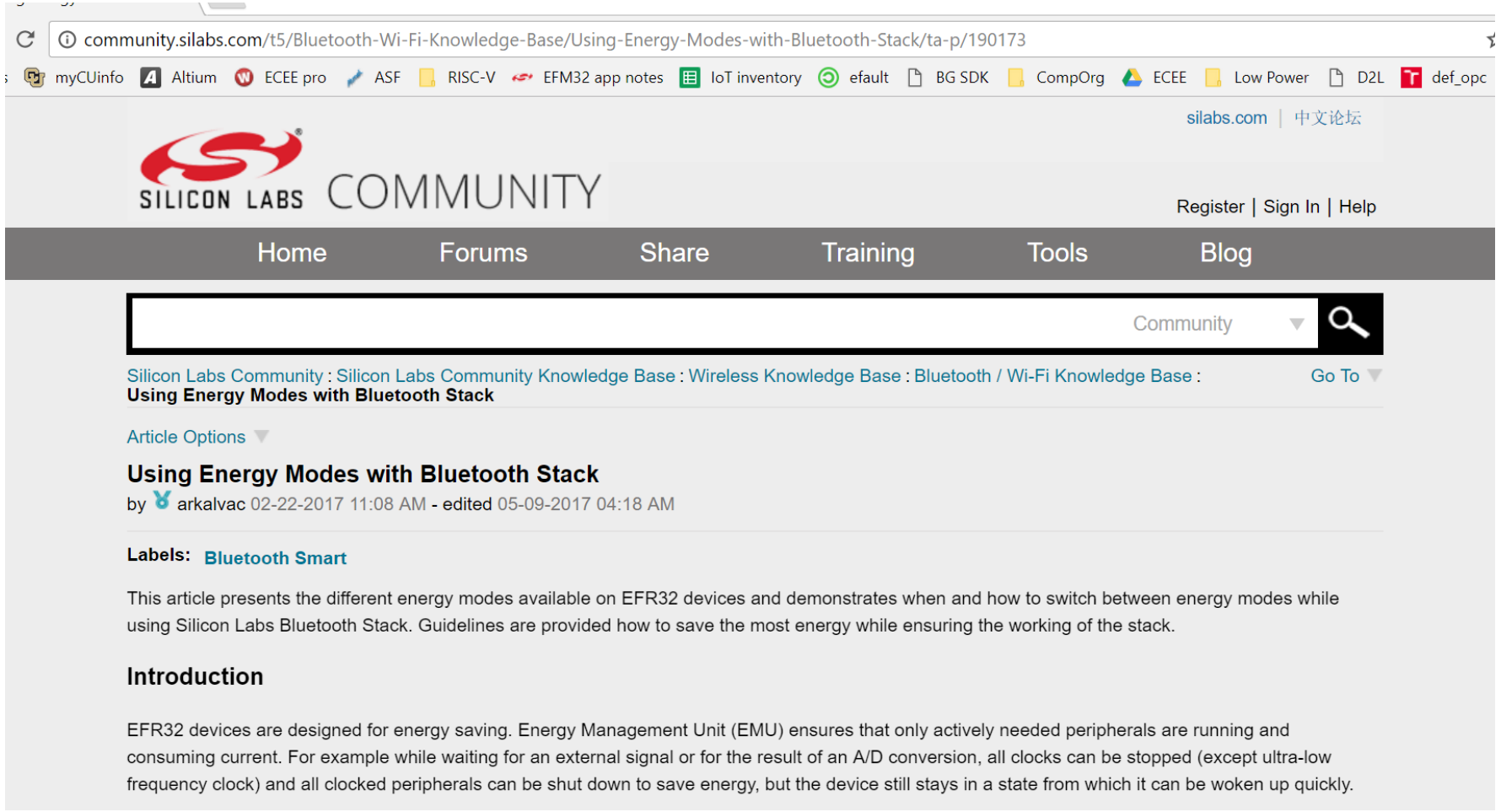
Table 2.111. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x04	lolen	Minimum payload length
2	0x0a	class	Message class: Generic Attribute Profile Server
3	0x05	method	Message ID
4	uint8	connection	Handle of the connection over which the notification or indication is sent. Values: • **0xff:** Sends notification or indication to all connected devices. • **Other:** Connection handle
5-6	uint16	characteristic	Characteristic handle
7	uint8array	value	Value to be notified or indicated

Bluetooth website to further understand defined services and profiles



Setting Application Energy Modes



Setting Application Energy Modes

Enabling EM2 Deep Sleep in the Stack

Regarding sleep policy Silicon Labs Bluetooth stack can be configured in two ways: putting the device into EM1 Sleep mode or into EM2 Deep Sleep mode, while EM0 Active mode is not needed. If sleep mode is not configured, EM1 is used. To enable EM2 Deep Sleep in BGScript add this line to the hardware.xml configuration file:

```
<sleep enable=”true” />
```

To enable Deep Sleep in a C project, add this line to the config structure, which will be passed to gecko_init():

```
config.sleep.flags=SLEEP_FLAGS_DEEP_SLEEP_ENABLE;
```

In the software examples provided with Silicon Labs Bluetooth SDK Deep Sleep is enabled by default (except NCP empty target).
The stack puts the device into EM1 Sleep or into EM2 Deep Sleep mode every time the processor is not needed, and starts a timer to wake it up when needed: when a communication window opens or a task is to be done.
The default template of a Bluetooth stack based application looks like this

```
while(1)
{
    evt = gecko_wait_event();

    switch(BGLIB_MSG_ID(evt->header))
    {
          //handling of different stack events
    }
}
```

Setting Application Energy Modes

Temporarily Disable EM2 Deep Sleep Mode

The Deep Sleep enable bit in the stack configuration cannot be dynamically changed. That is once Deep Sleep is enabled/disabled, it cannot be withdrawn. However, there are two ways to temporarily disable going to EM2 Deep Sleep mode: using sleep driver or using wake up pin.

With sleep driver the EM2 Deep sleep mode can be disabled (/blocked) temporarily by using **SLEEP_SleepBlockBegin(sleepEM2)**. To Re-enable EM2 Deep Sleep mode use **SLEEP_SleepBlockEnd(sleepEM2)**. If EM2 is disabled (/blocked), then the stack will switch between EM0 and EM1 temporarily.

To access these functions sleep.h has to be included in your source file! sleep.c is already precompiled with the stack, hence it has not to be added to the project!

Note, that multiple issuing of SLEEP_SleepBlockBegin() requires multiple issuing of SLEEP_Sleep_BlockEnd(). Every SLEEP_SleepBlockBegin() increases the corresponding counter and every SLEEP_SleepBlockEnd() decreases it.

Setting Application Energy Modes

SILICON LABS

UG136: Silicon Labs *Bluetooth* ® C Application Developer's Guide

This document is an essential reference for everybody developing C-based applications for the Silicon Labs Wireless Gecko products using the Silicon Labs Bluetooth stack. The guide covers the Bluetooth stack architecture, application development flow, usage and limitations of the MCU core and peripherals, stack configuration options, and stack resource usage.

The purpose of the document is to capture and fill in the blanks between the Bluetooth Stack API reference, Gecko SDK API reference, and Wireless Gecko reference manuals, when developing Bluetooth applications for the Wireless Geckos. This document exposes details that will help developers make the most out of the available hardware resources.

KEY POINTS

- Project structure and development flow
- Bluetooth stack and Wireless Gecko configuration
- Interrupt handling
- Event and sleep management
- Resource usage and available resources

```
/**
* Main
*/
void main()
{
  ...

      //Event loop
  while(1)
  {
    ...

      //External signal indication (comes from the interrupt handler)
    case gecko_evt_system_external_signal_id:
      // Handle GPIO IRQ and do something
      // External signal command's parameter can be accessed using
     // event->data.evt_system_external_signal.extsignals
      break;
    ...
  }
}

/**
* Handle GPIO interrupts and trigger system_external_signal event
*/
void GPIO_ODD_IRQHandler()
{
  static bool radioHalted = false;

  uint32_t flags = GPIO_IntGet();
  GPIO_IntClear(flags);

  //Send gecko_evt_system_external_signal_id event to the main loop
  gecko_external_signal(...);
  }
}
```