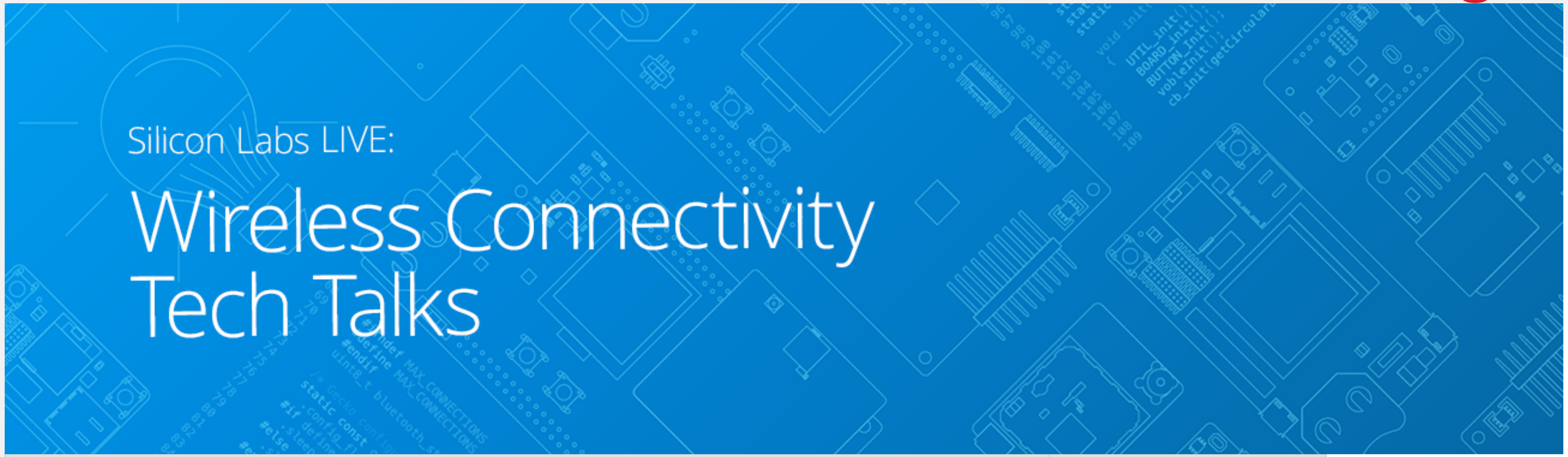
Tech Talks LIVE Schedule – Presentation will begin shortly

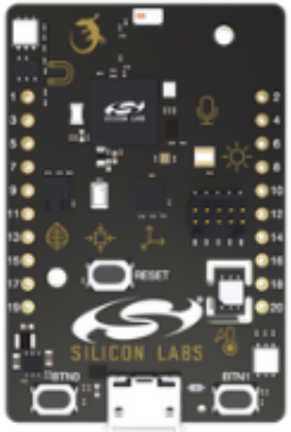


Silicon Labs LIVE:

Wireless Connectivity Tech Talks

Topic	Date
Multiprotocol Wireless: Real Application of Dynamic Multiprotocol	Tuesday, June 9
Wireless Coexistence	Thursday, June 11
Bluetooth Software Structure: Learn the APIs and State Machines	**Tuesday, June 16**
Add a Peripheral to a Project in No Time: With 32-bit Peripheral Github Library	Thursday, June 18
Low power wireless to the next level: Energy Friendly PMIC + Energy Friendly Radio BG22	Tuesday, June 23
Talk with an Alexa: Using Zigbee to Connect with an Echo Plus	Thursday, June 25
Z-Wave Software Structure: Learn about Command Classes and Reference Code	Tuesday, June 30
Building a Proper Mesh Test Environment: How This Was Solved in Boston	Thursday, July 2

Fill out the survey for a chance to win a Thunderboard BG22 eval board!



Find Past Recorded Sessions at: https://www.silabs.com/support/training

WELCOME

Silicon Labs LIVE:

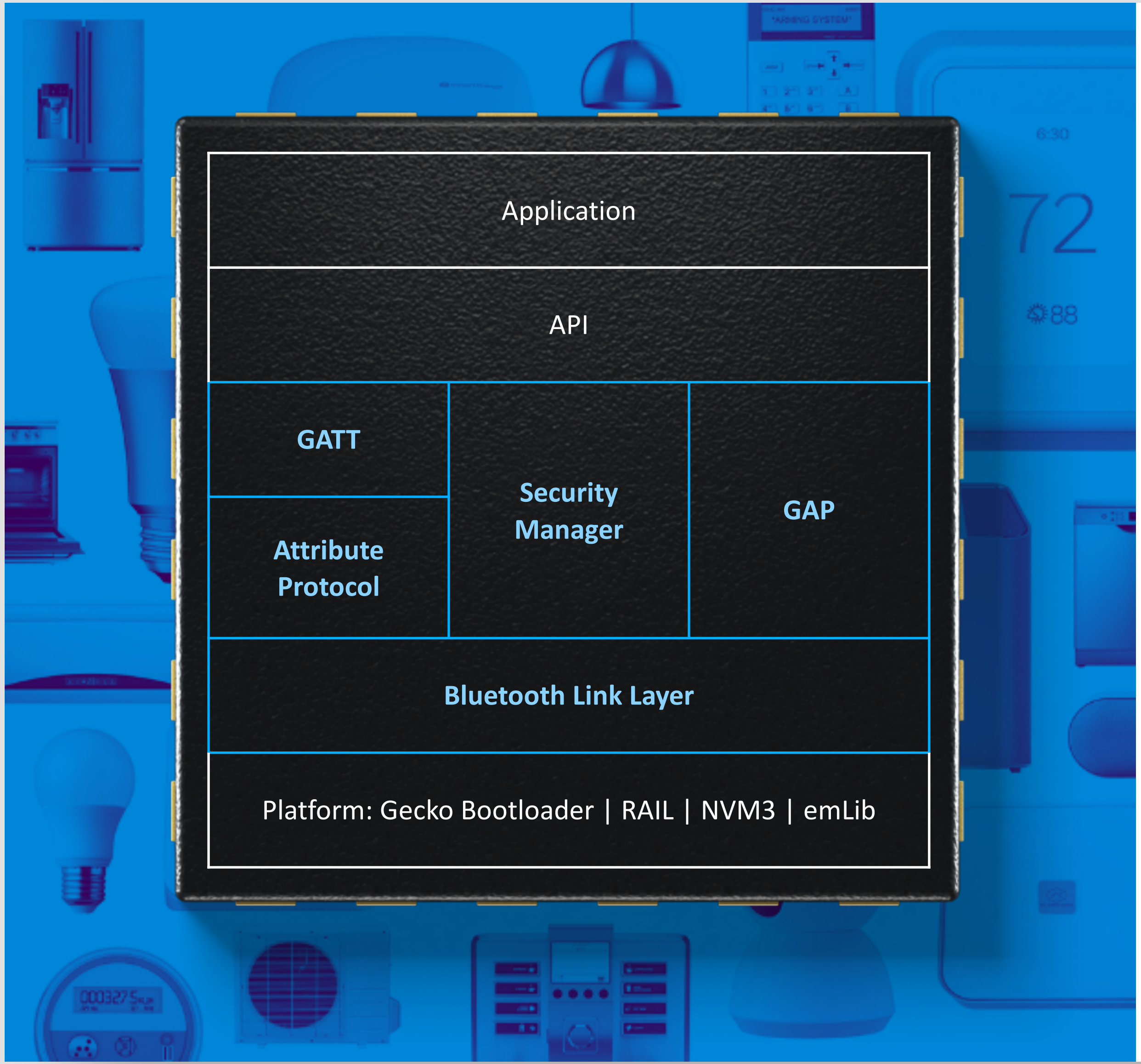
Wireless Connectivity Tech Talks

Bluetooth Software Structure: Learn the APIs and State Machine

Agenda

1. Application Development Flow
2. Project Structure
3. GATT Configurator
4. Configuring the stack
5. Event Handling

Bluetooth LE Software



A Bluetooth 5.2 compliant Bluetooth stack, with:

- Bluetooth 5.2 Dynamic TX power control
- Bluetooth 5.1 Direction Finding
- Bluetooth 5.0 standard features
- Relevant Bluetooth 4.x features

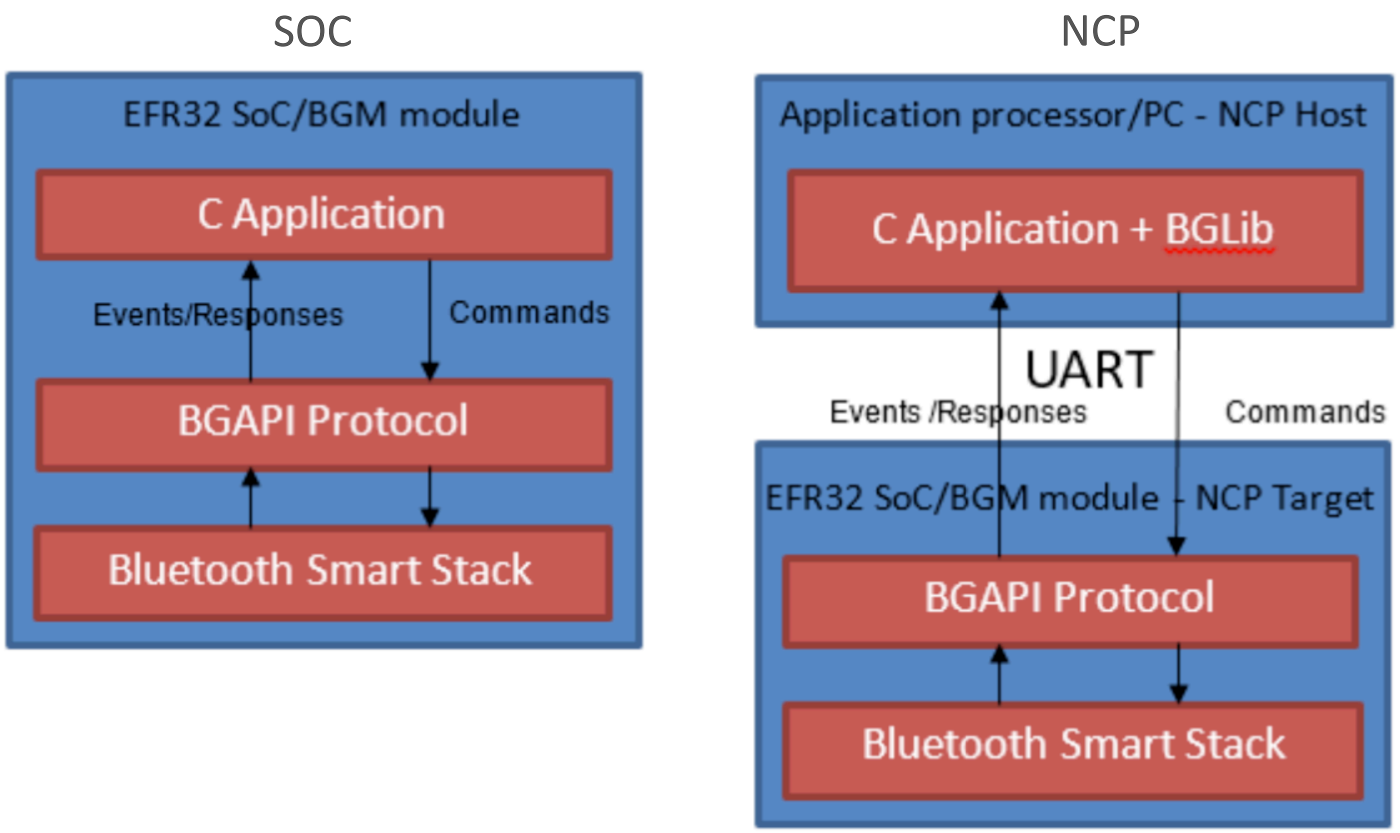
Packed with advanced functionality

- Multiple connections and advertisers
- Concurrent advertising, scanning and LE connections
- Optimized throughput and power consumption

Built on top of the common EFR32 software platform

- Gecko bootloader
- emLib for MCU peripherals and drivers
- NVM3 key/value pair data storage with wear leveling
- RAIL radio driver

SOC vs NCP System Models



Source: AN1042: Using the Silicon Labs *Bluetooth*® Stack in Network Co-Processor Mode

Network Co-Processor (NCP) architecture

- Bluetooth stack runs on the Blue Gecko SoC
- Provides Bluetooth API over UART I/F
- **Host API**
- Host API is 100% identical to SoC API
- **NCP features**
- AES-128 encrypted UART communications
- 4-wire UART with RTS/CTS
- 802.11 co-existence interface via GPIO pins

Where should I start from?

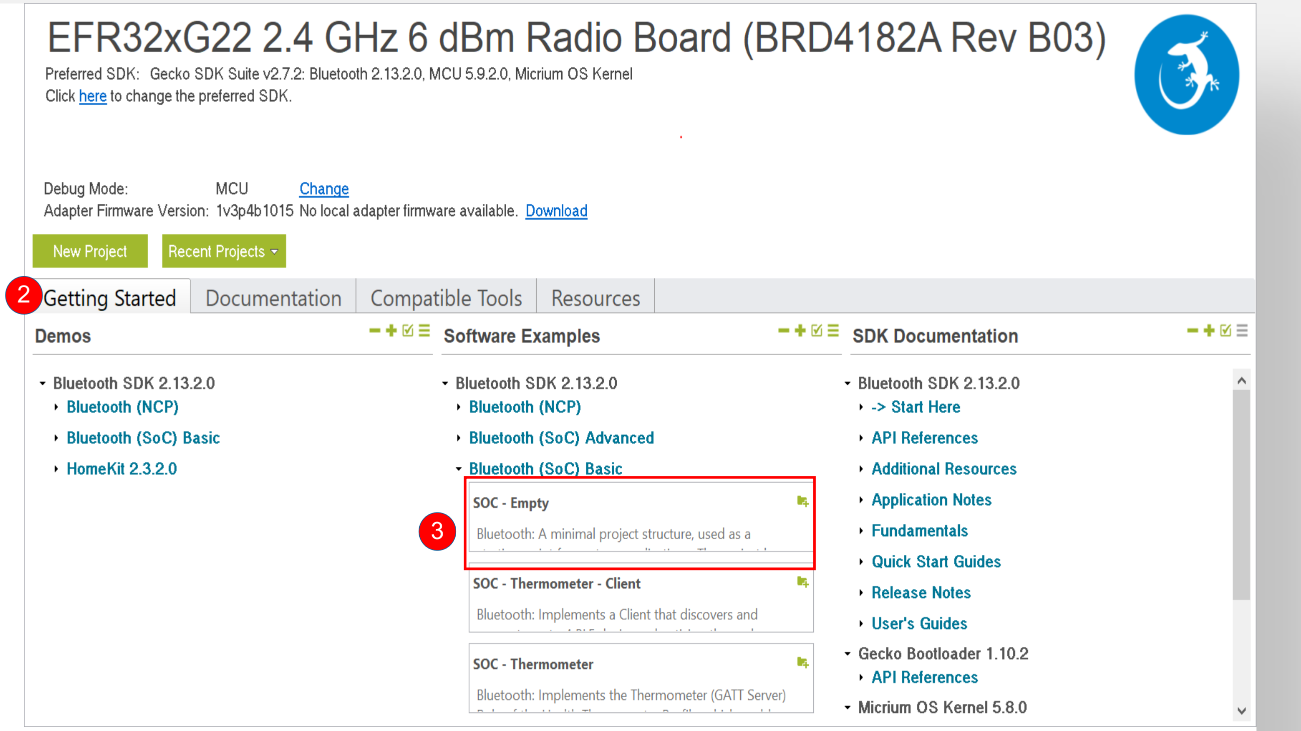
Empty Soc Example Project

1. **Plug Wireless Starter Kit into USB Port. Click on the kit listed under Device Pane in upper left.**

2. **Click on the Getting Started Tab from the Launcher dashboard in Simplicity Studio.**

3. **Click on the SoC-Empty project listed under Software Examples.**

4. **Click Yes when asked to switch to Simplicity IDE perspective and create the project.**



EFR32xG22 2.4 GHz 6 dBm Radio Board (BRD4182A Rev B03)

Preferred SDK: Gecko SDK Suite v2.7.2: Bluetooth 2.13.2.0, MCU 5.9.2.0, Micrium OS Kernel

Click here to change the preferred SDK.

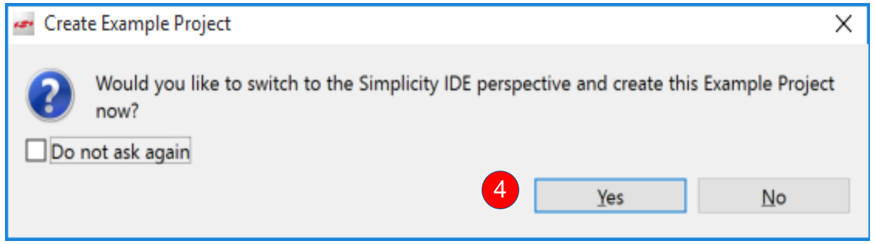
Debug Mode: MCU Change

Adapter Firmware Version: 1v3p4b1015 No local adapter firmware available. Download

SOC - Empty: Bluetooth: A minimal project structure, used as a

SOC - Thermometer - Client: Bluetooth: Implements a Client that discovers and

SOC - Thermometer: Bluetooth: Implements the Thermometer (GATT Server)

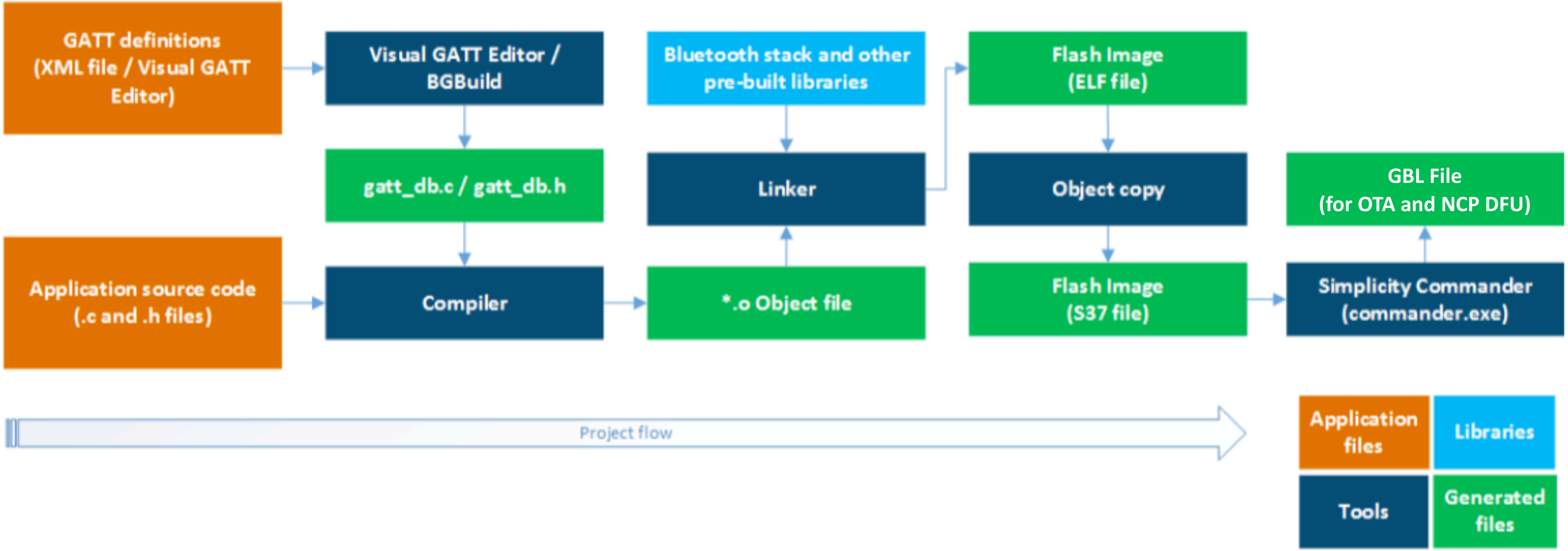


Create Example Project

Would you like to switch to the Simplicity IDE perspective and create this Example Project now?

☐ Do not ask again

Application Build Flow



GATT definitions (XML file / Visual GATT Editor)

Visual GATT Editor / BGBuild

gatt_db.c / gatt_db.h

Application source code (.c and .h files)

Compiler

*.o Object file

Bluetooth stack and other pre-built libraries

Linker

Flash Image (ELF file)

Object copy

Flash Image (S37 file)

Simplicity Commander (commander.exe)

GBL File (for OTA and NCP DFU)

Project flow

Application files

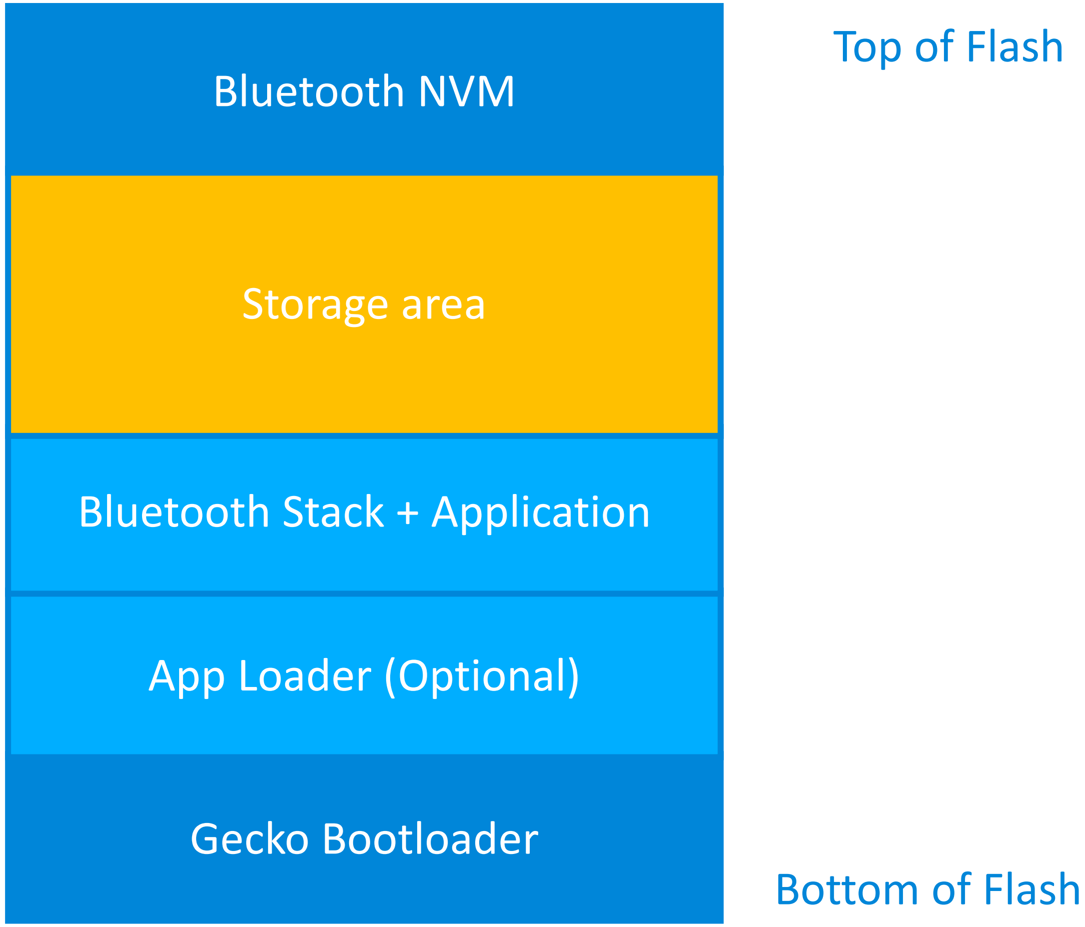
Libraries

Tools

Generated files

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

Device Memory organization



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

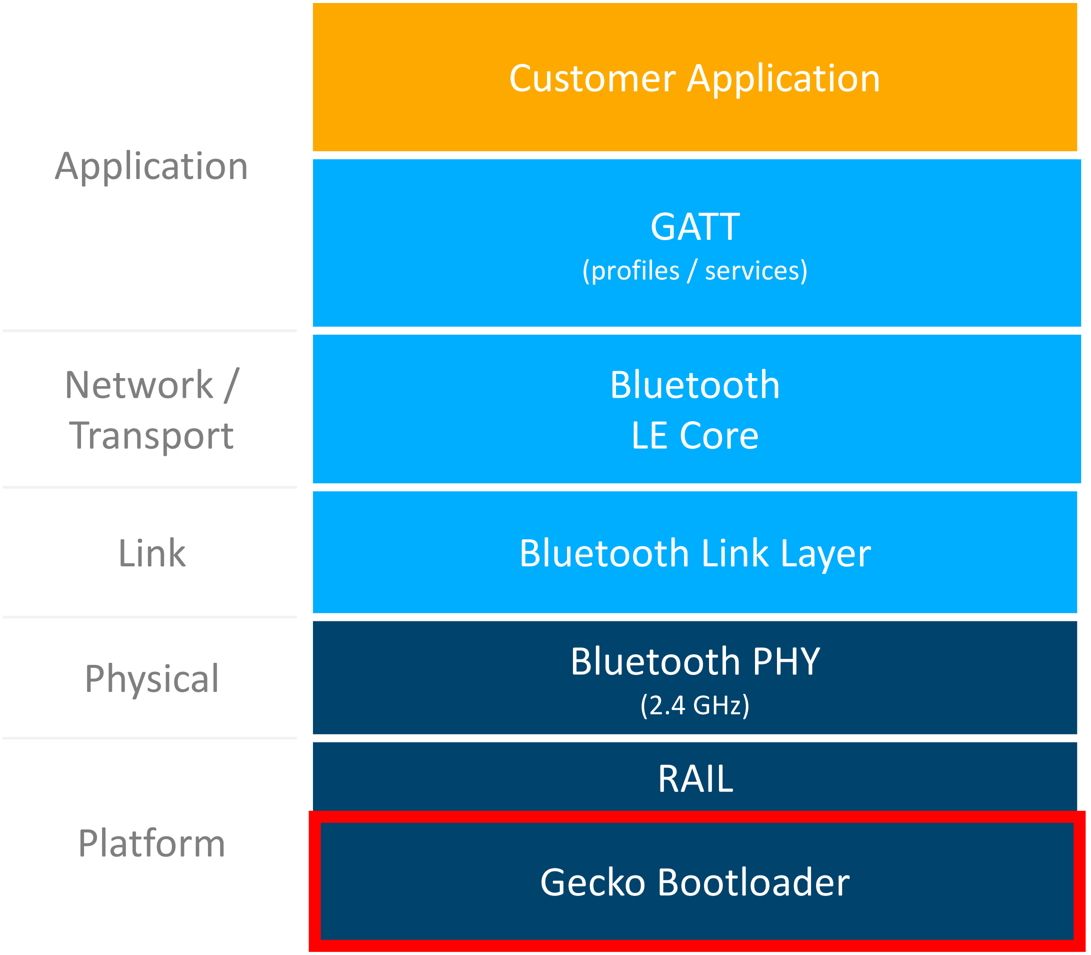
Project Structure

Gecko Bootloader

- Distributed as source
- Has its own Project Structure
- NCP – Stand Alone Bootloader
- SOC – Application Bootloader

AN1086 - Gecko bootloader Bluetooth

UG266 – Gecko bootloader user guide



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

Project Structure

Bluetooth Stack and RAIL Binaries

Bluetooth Stack binaries:

libbluetooth.a: Bluetooth stack library.

binapploader.o: AppLoader, provides the optional OTA functionality.

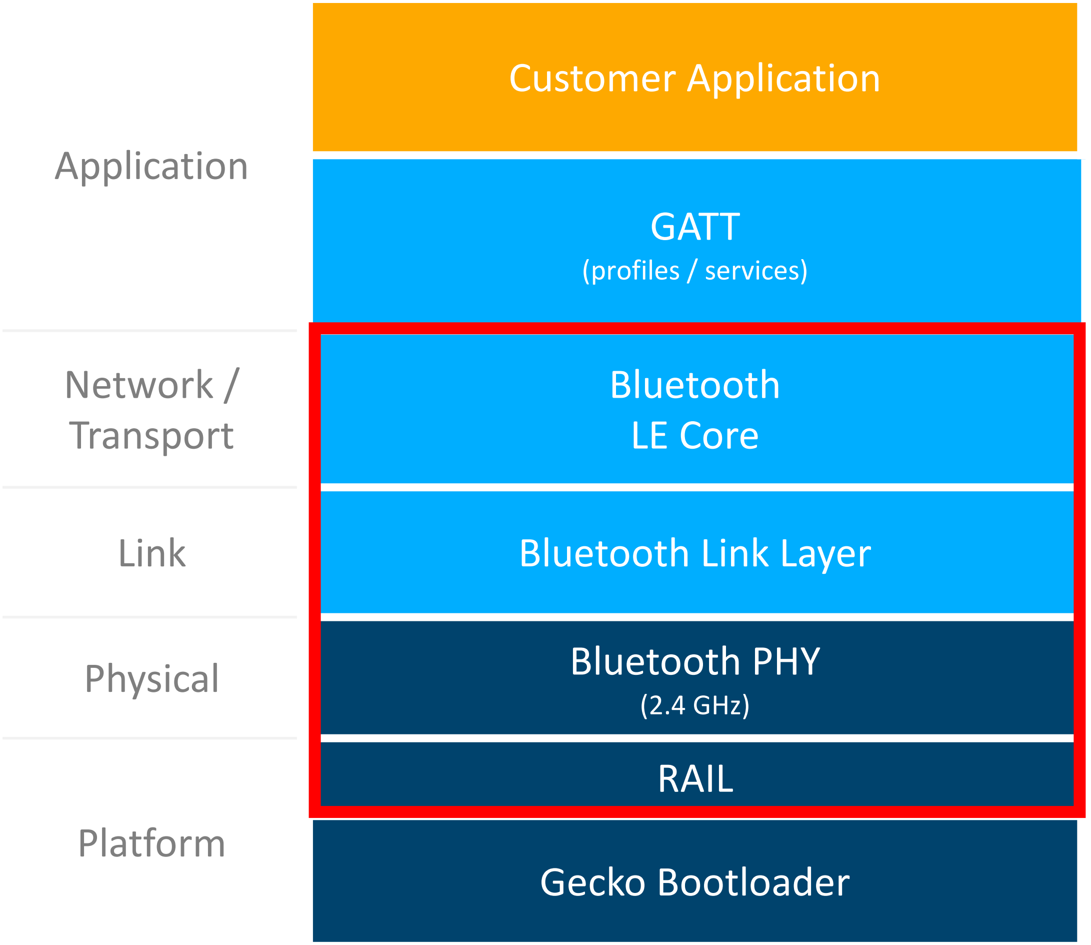
libmbedtls.a: mbed TLS cryptographic library for the Bluetooth Stack.

libpsstore.a: PS Store functionality for the Bluetooth stack

Rail Libraries:

librail_module_<soc family><compiler>_release.a

lib- rail_config<modulename>.a



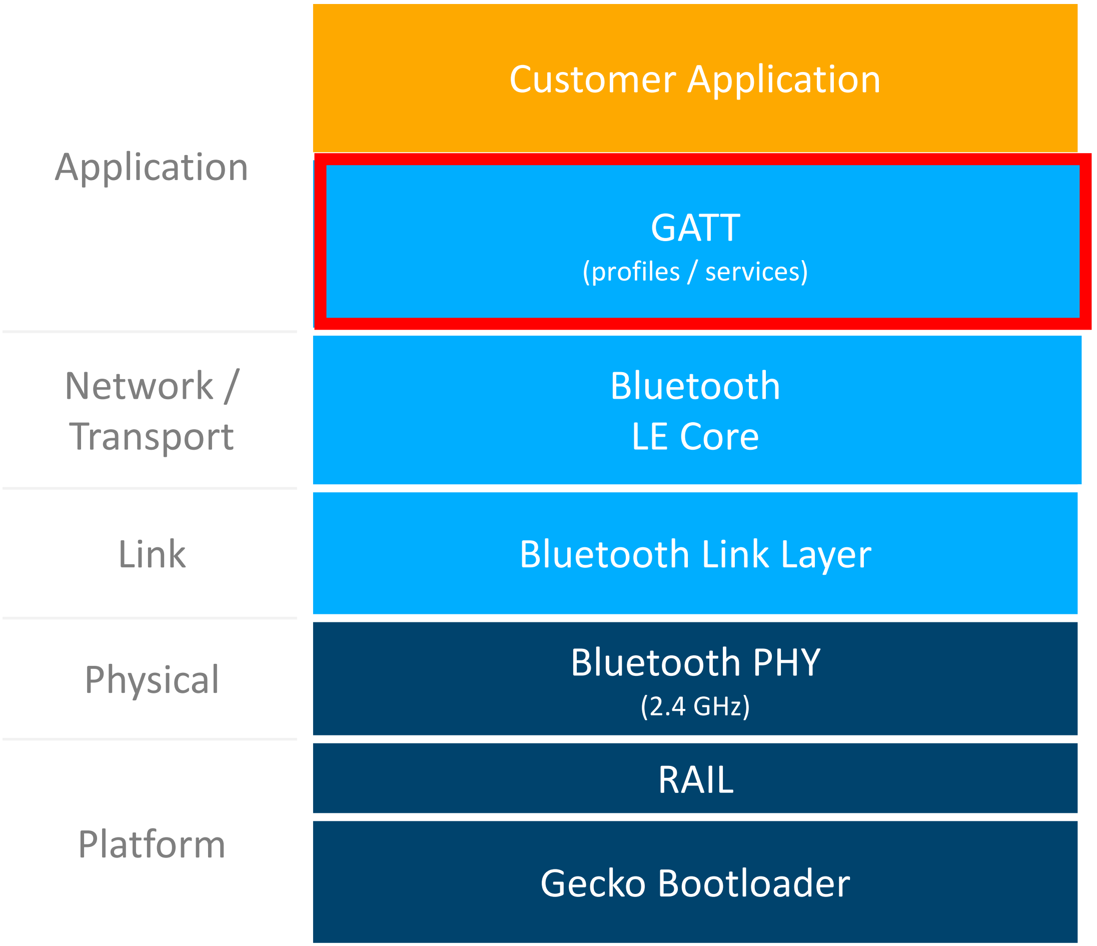
Source: UG136: Silicon Labs *Bluetooth* ® C Application Developer's Guide

Project Structure

GATT related files

- User Friendly based GUI that creates gatt_db.c and gatt_db.h and xml files

- GATT Services and Characteristics need to be created in compilation time and can only be updated via device OTA/DFU upgrade

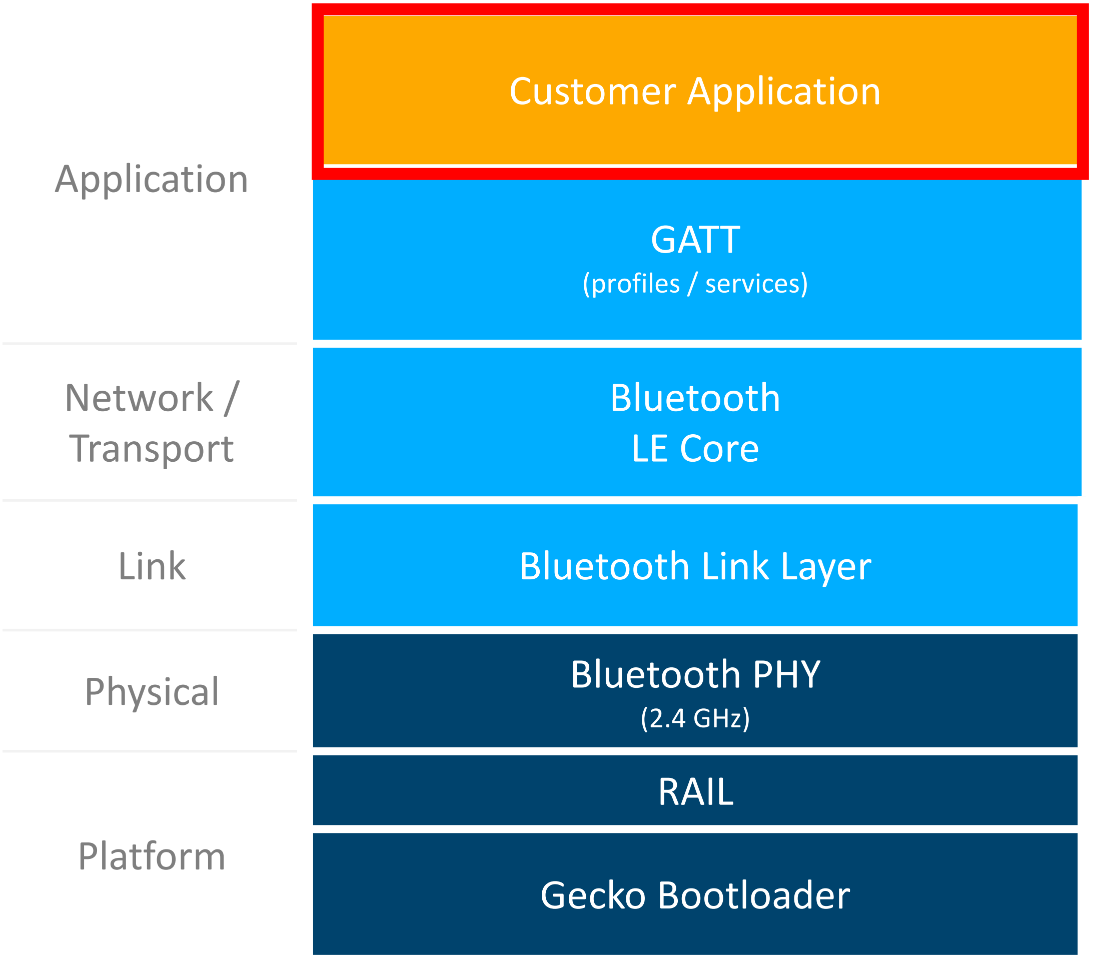


Source: UG136: Silicon Labs *Bluetooth* ® C Application Developer's Guide

Project Structure

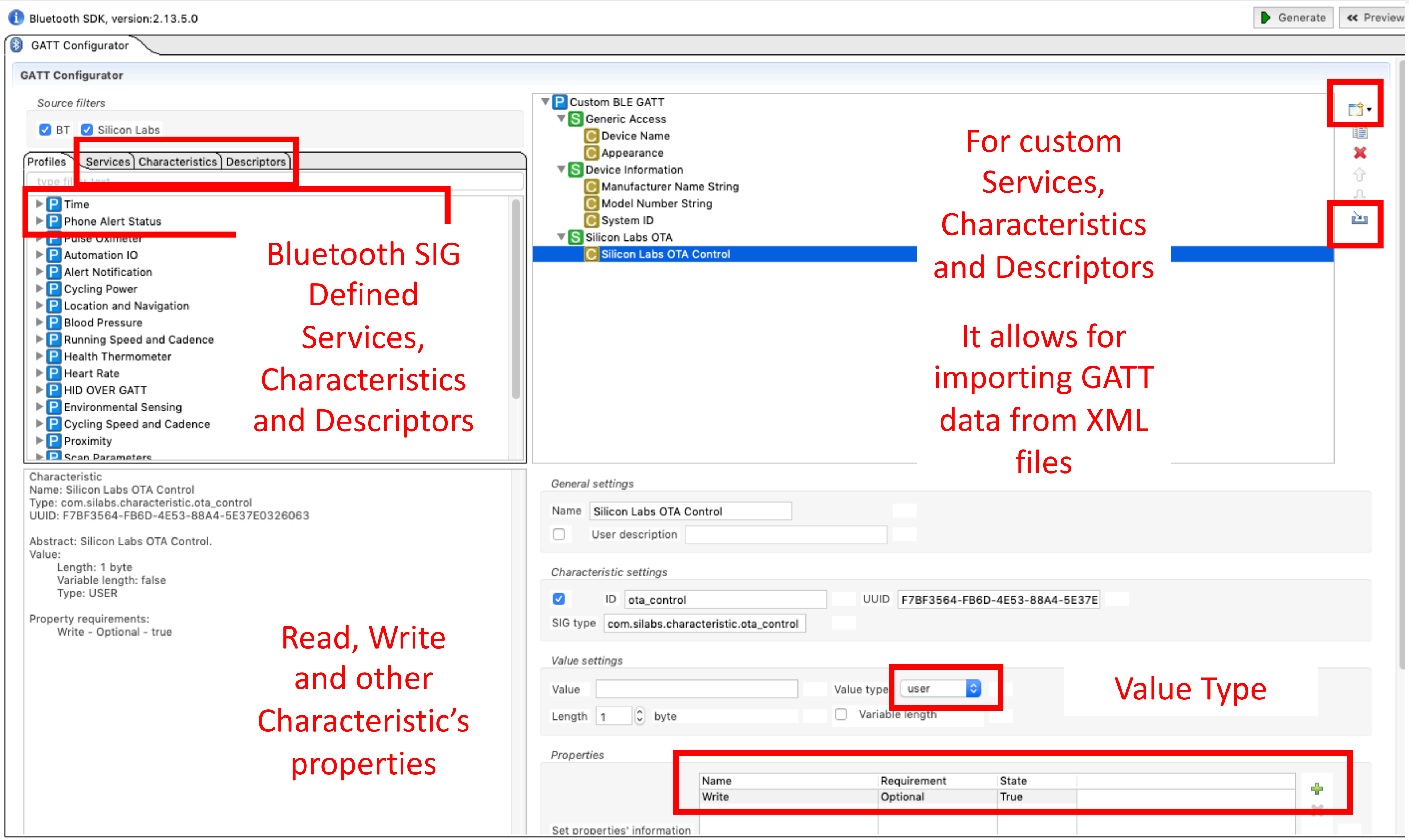
EMLIB and EMDRV

- Silicon Labs Peripheral Support drivers and Libraries
- Usage examples can be found on Silabs GitHub
- Include and Source files can be found at *<Simplicity Studio Gecko SDK>\platform\*.



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

GATT Configurator



Bluetooth SDK, version:2.13.5.0

Generate | Preview

GATT Configurator

GATT Configurator

Source filters

☑ BT ☑ Silicon Labs

Profiles | Services | Characteristics | Descriptors

- P Time
- P Phone Alert Status
- P Pulse Oximeter
- P Automation IO
- P Alert Notification
- P Cycling Power
- P Location and Navigation
- P Blood Pressure
- P Running Speed and Cadence
- P Health Thermometer
- P Heart Rate
- P HID OVER GATT
- P Environmental Sensing
- P Cycling Speed and Cadence
- P Proximity
- P Scan Parameters

```
Characteristic
Name: Silicon Labs OTA Control
Type: com.silabs.characteristic.ota_control
UUID: F7BF3564-FB6D-4E53-88A4-5E37E0326063

Abstract: Silicon Labs OTA Control.
Value:
        Length: 1 byte
        Variable length: false
        Type: USER

Property requirements:
        Write - Optional - true
```

- P Custom BLE GATT
  - S Generic Access
    - C Device Name
    - C Appearance
  - S Device Information
    - C Manufacturer Name String
    - C Model Number String
    - C System ID
  - S Silicon Labs OTA
    - C Silicon Labs OTA Control

General settings

Name: Silicon Labs OTA Control

□ User description:

Characteristic settings

☑ ID: ota_control UUID: F7BF3564-FB6D-4E53-88A4-5E37E

SIG type: com.silabs.characteristic.ota_control

Value settings

Value: Value type: user

Length: 1 byte □ Variable length

Properties

Name	Requirement	State
Write	Optional	True

Set properties' information

Bluetooth SIG Defined Services, Characteristics and Descriptors

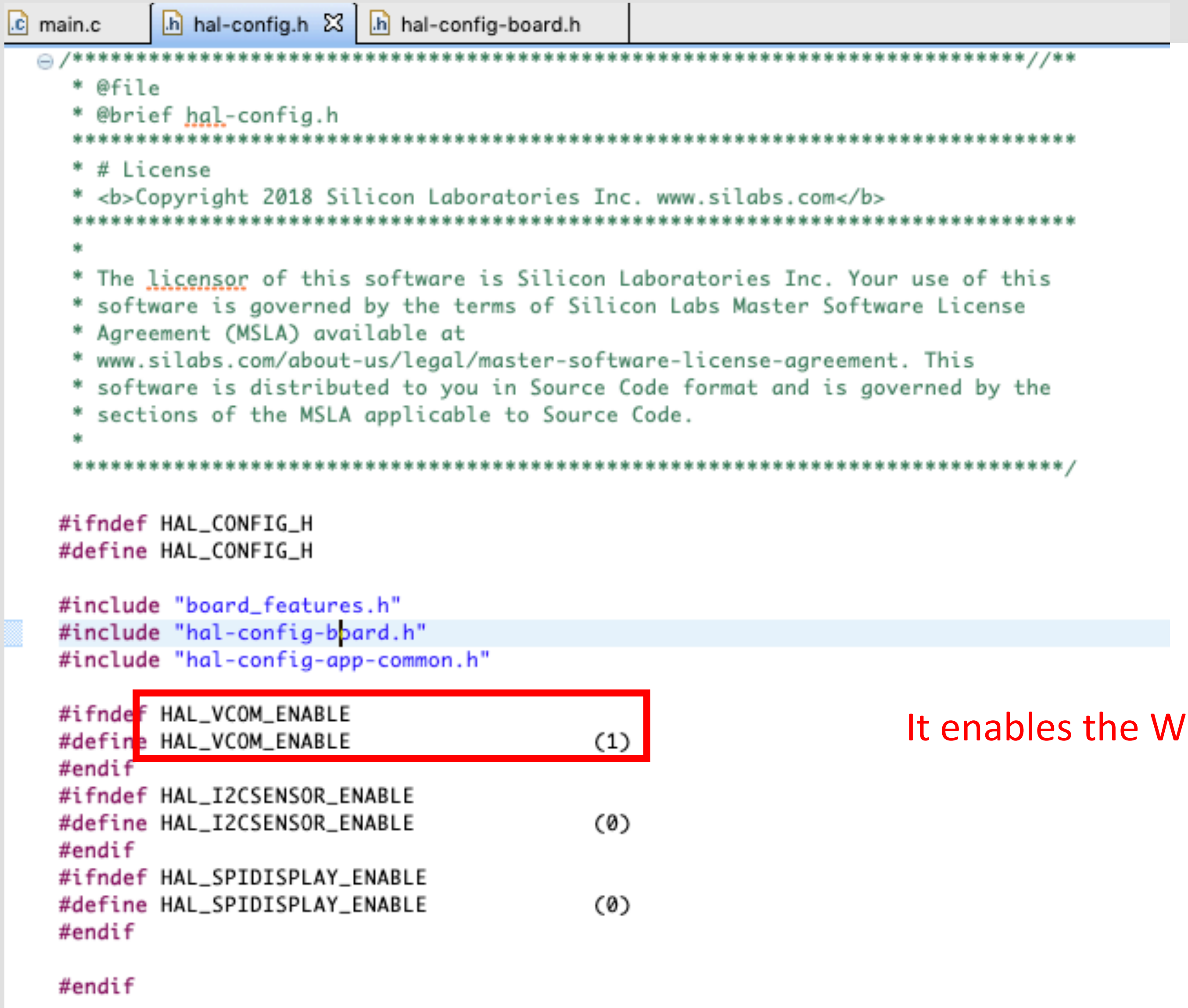
For custom Services, Characteristics and Descriptors

It allows for importing GATT data from XML files

Read, Write and other Characteristic's properties

Value Type

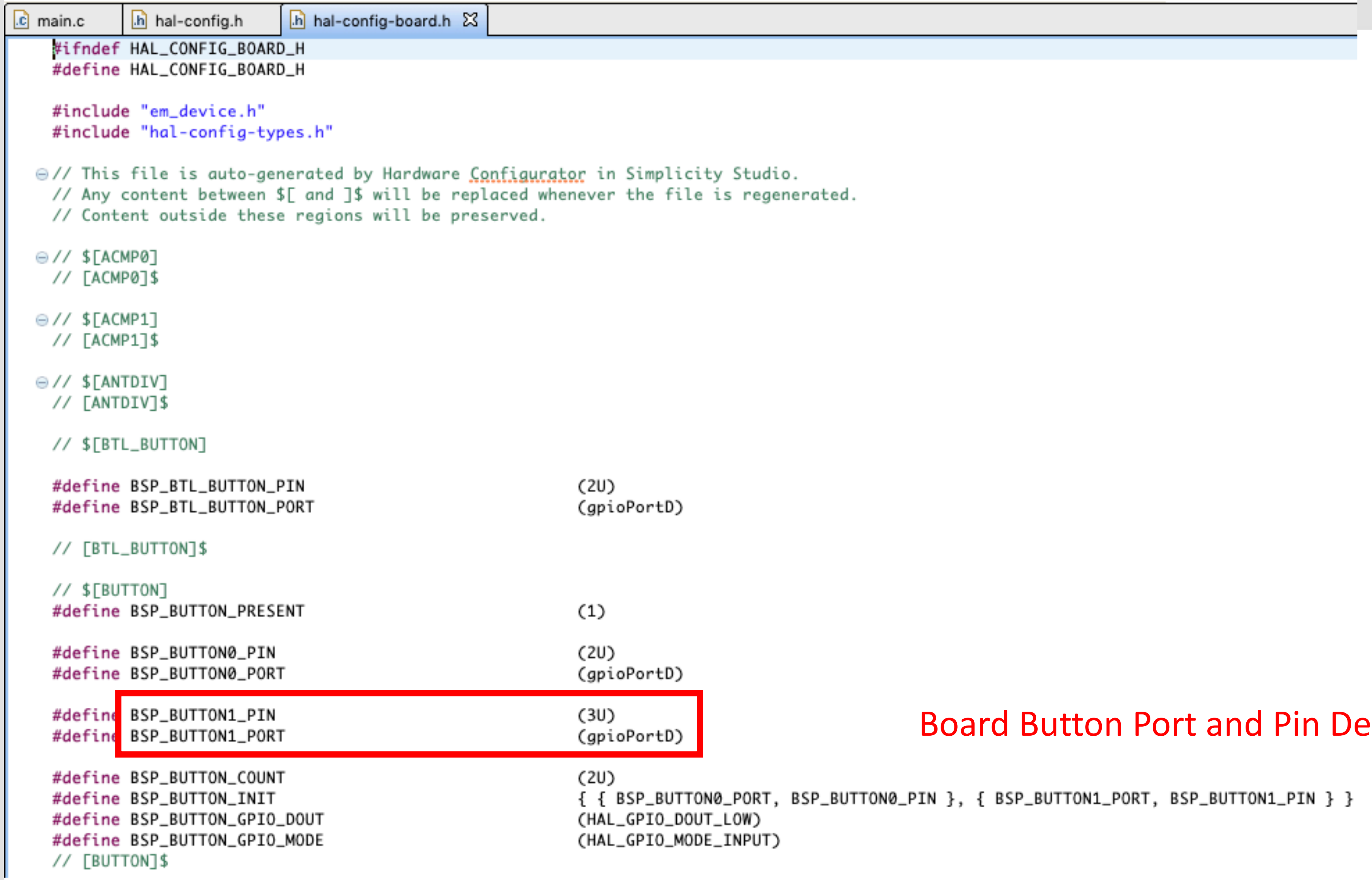
Hardware Configuration – hal-config.h



```
/***************************************************************************//**
 * @file
 * @brief hal-config.h
 *******************************************************************************
 * # License
 * <b>Copyright 2018 Silicon Laboratories Inc. www.silabs.com</b>
 *******************************************************************************
 *
 * The licensor of this software is Silicon Laboratories Inc. Your use of this
 * software is governed by the terms of Silicon Labs Master Software License
 * Agreement (MSLA) available at
 * www.silabs.com/about-us/legal/master-software-license-agreement. This
 * software is distributed to you in Source Code format and is governed by the
 * sections of the MSLA applicable to Source Code.
 *
 ******************************************************************************/

#ifndef HAL_CONFIG_H
#define HAL_CONFIG_H

#include "board_features.h"
#include "hal-config-board.h"
#include "hal-config-app-common.h"

#ifndef HAL_VCOM_ENABLE
#define HAL_VCOM_ENABLE                        (1)
#endif
#ifndef HAL_I2CSENSOR_ENABLE
#define HAL_I2CSENSOR_ENABLE                   (0)
#endif
#ifndef HAL_SPIDISPLAY_ENABLE
#define HAL_SPIDISPLAY_ENABLE                  (0)
#endif

#endif
```

It enables the WSTK built-in USB to UART converter.

Hardware Configuration – hal-config-board.h

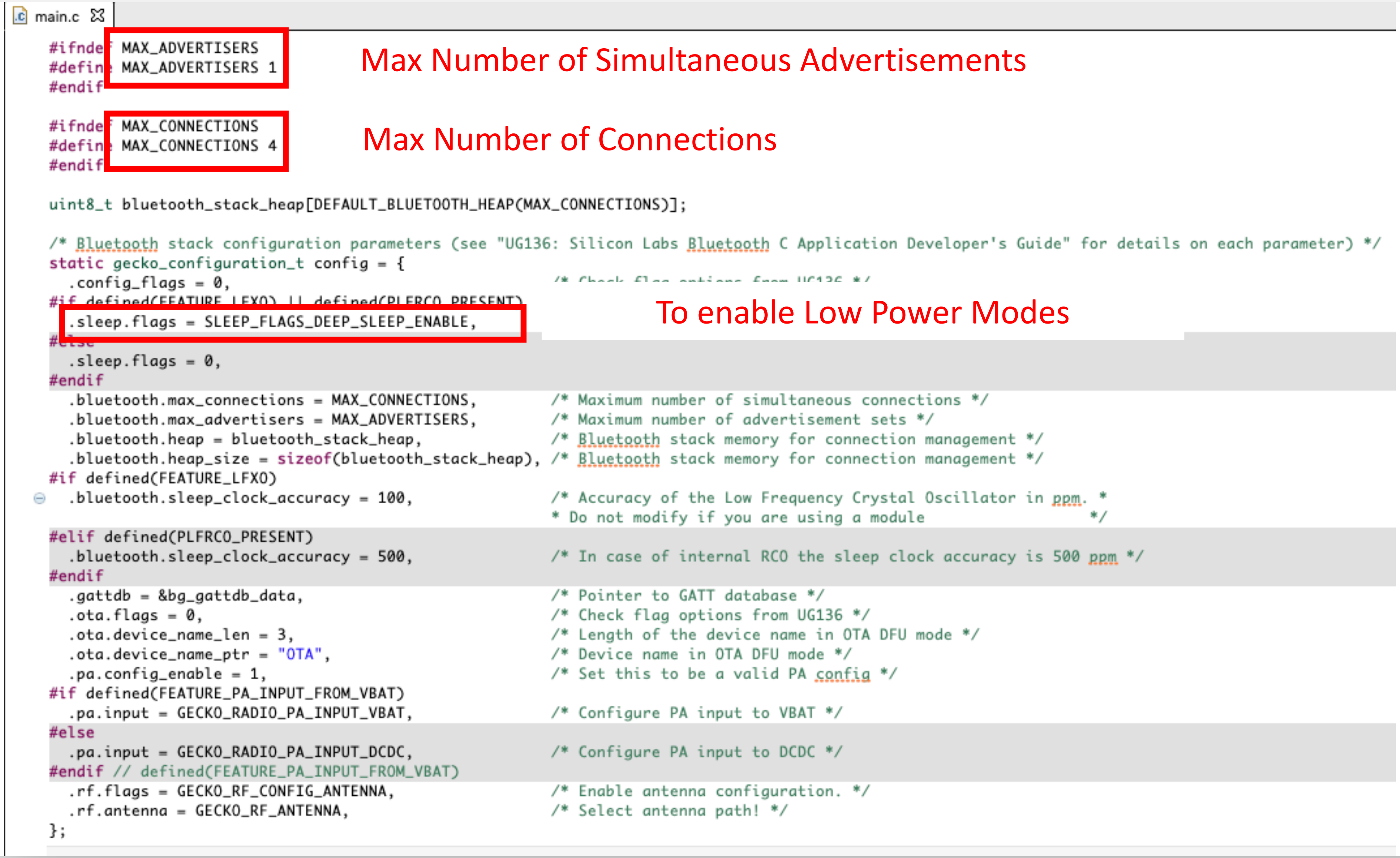


```c
#ifndef HAL_CONFIG_BOARD_H
#define HAL_CONFIG_BOARD_H

#include "em_device.h"
#include "hal-config-types.h"

// This file is auto-generated by Hardware Configurator in Simplicity Studio.
// Any content between $[ and ]$ will be replaced whenever the file is regenerated.
// Content outside these regions will be preserved.

// $[ACMP0]
// [ACMP0]$

// $[ACMP1]
// [ACMP1]$

// $[ANTDIV]
// [ANTDIV]$

// $[BTL_BUTTON]

#define BSP_BTL_BUTTON_PIN                           (2U)
#define BSP_BTL_BUTTON_PORT                          (gpioPortD)

// [BTL_BUTTON]$

// $[BUTTON]
#define BSP_BUTTON_PRESENT                           (1)

#define BSP_BUTTON0_PIN                              (2U)
#define BSP_BUTTON0_PORT                             (gpioPortD)

#define BSP_BUTTON1_PIN                              (3U)
#define BSP_BUTTON1_PORT                             (gpioPortD)

#define BSP_BUTTON_COUNT                             (2U)
#define BSP_BUTTON_INIT                              { { BSP_BUTTON0_PORT, BSP_BUTTON0_PIN }, { BSP_BUTTON1_PORT, BSP_BUTTON1_PIN } }
#define BSP_BUTTON_GPIO_DOUT                         (HAL_GPIO_DOUT_LOW)
#define BSP_BUTTON_GPIO_MODE                         (HAL_GPIO_MODE_INPUT)
// [BUTTON]$
```

Board Button Port and Pin Definitions

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

Configuring the BLE Stack – main.c



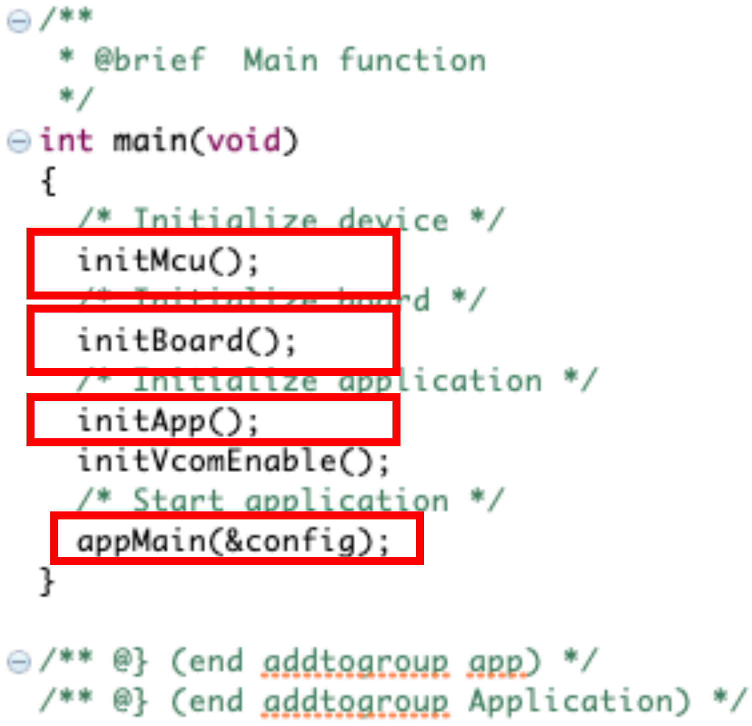
```c
#ifndef MAX_ADVERTISERS
#define MAX_ADVERTISERS 1
#endif

#ifndef MAX_CONNECTIONS
#define MAX_CONNECTIONS 4
#endif

uint8_t bluetooth_stack_heap[DEFAULT_BLUETOOTH_HEAP(MAX_CONNECTIONS)];

/* Bluetooth stack configuration parameters (see "UG136: Silicon Labs Bluetooth C Application Developer's Guide" for details on each parameter) */
static gecko_configuration_t config = {
  .config_flags = 0,                                      /* Check flag options from UG136 */
#if defined(FEATURE_LFXO) || defined(PLFRCO_PRESENT)
  .sleep.flags = SLEEP_FLAGS_DEEP_SLEEP_ENABLE,
#else
  .sleep.flags = 0,
#endif
  .bluetooth.max_connections = MAX_CONNECTIONS,           /* Maximum number of simultaneous connections */
  .bluetooth.max_advertisers = MAX_ADVERTISERS,           /* Maximum number of advertisement sets */
  .bluetooth.heap = bluetooth_stack_heap,                 /* Bluetooth stack memory for connection management */
  .bluetooth.heap_size = sizeof(bluetooth_stack_heap),    /* Bluetooth stack memory for connection management */
#if defined(FEATURE_LFXO)
  .bluetooth.sleep_clock_accuracy = 100,                  /* Accuracy of the Low Frequency Crystal Oscillator in ppm. *
                                                          * Do not modify if you are using a module                  */
#elif defined(PLFRCO_PRESENT)
  .bluetooth.sleep_clock_accuracy = 500,                  /* In case of internal RCO the sleep clock accuracy is 500 ppm */
#endif
  .gattdb = &bg_gattdb_data,                              /* Pointer to GATT database */
  .ota.flags = 0,                                         /* Check flag options from UG136 */
  .ota.device_name_len = 3,                               /* Length of the device name in OTA DFU mode */
  .ota.device_name_ptr = "OTA",                           /* Device name in OTA DFU mode */
  .pa.config_enable = 1,                                  /* Set this to be a valid PA config */
#if defined(FEATURE_PA_INPUT_FROM_VBAT)
  .pa.input = GECKO_RADIO_PA_INPUT_VBAT,                  /* Configure PA input to VBAT */
#else
  .pa.input = GECKO_RADIO_PA_INPUT_DCDC,                  /* Configure PA input to DCDC */
#endif // defined(FEATURE_PA_INPUT_FROM_VBAT)
  .rf.flags = GECKO_RF_CONFIG_ANTENNA,                    /* Enable antenna configuration. */
  .rf.antenna = GECKO_RF_ANTENNA,                         /* Select antenna path! */
};
```

Max Number of Simultaneous Advertisements

Max Number of Connections

To enable Low Power Modes

Device Initialization



```
/**
  * @brief  Main function
  */
int main(void)
{
    /* Initialize device */
    initMcu();
    /* Initialize board */
    initBoard();
    /* Initialize application */
    initApp();
    initVcomEnable();
    /* Start application */
    appMain(&config);
}

/** @} (end addtogroup app) */
/** @} (end addtogroup Application) */
```

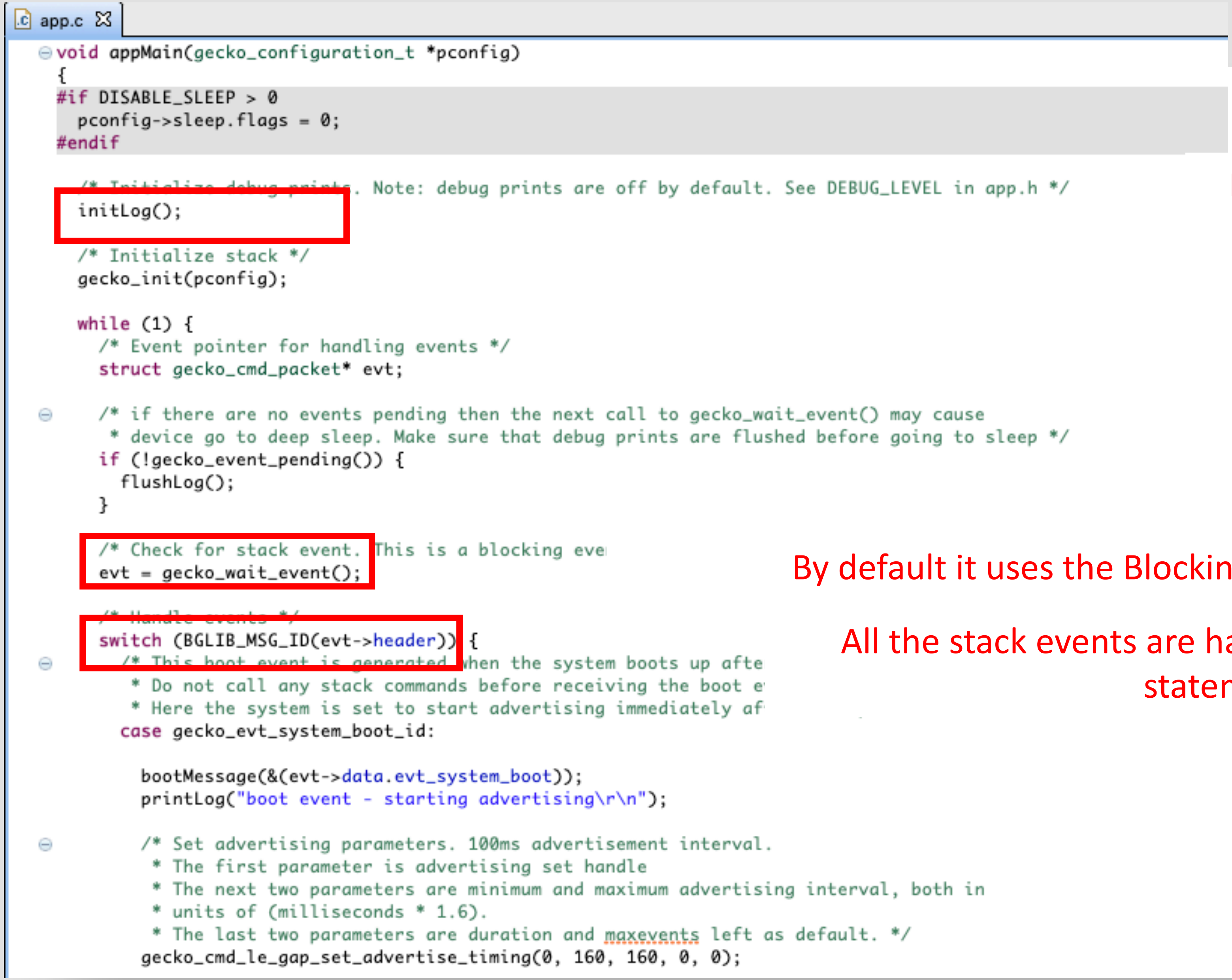
The initMCU() function is used to initialize MCU core. It Initializes clocks , DC-DC configuration and power modes.

The initBoard() function is used to initialize board features, such as initializing the external flash. Peripheral initializations can be added to this function.

The initApp() function is used to initialize application-specific features, such as enabling the SPI display on the WSTK. The function must be called after initBoard().

The appMain() function is the one that invokes the main Bluetooth state machine by passing the stack configurations.

App.c –>appMain()-> Bluetooth Stack Event Handling



```
void appMain(gecko_configuration_t *pconfig)
{
#if DISABLE_SLEEP > 0
  pconfig->sleep.flags = 0;
#endif

  /* Initialize debug prints. Note: debug prints are off by default. See DEBUG_LEVEL in app.h */
  initLog();

  /* Initialize stack */
  gecko_init(pconfig);

  while (1) {
    /* Event pointer for handling events */
    struct gecko_cmd_packet* evt;

    /* if there are no events pending then the next call to gecko_wait_event() may cause
     * device go to deep sleep. Make sure that debug prints are flushed before going to sleep */
    if (!gecko_event_pending()) {
      flushLog();
    }

    /* Check for stack event. This is a blocking eve
    evt = gecko_wait_event();

    /* Handle events */
    switch (BGLIB_MSG_ID(evt->header)) {
      /* This boot event is generated when the system boots up afte
       * Do not call any stack commands before receiving the boot e
       * Here the system is set to start advertising immediately af
      case gecko_evt_system_boot_id:

        bootMessage(&(evt->data.evt_system_boot));
        printLog("boot event - starting advertising\r\n");

        /* Set advertising parameters. 100ms advertisement interval.
         * The first parameter is advertising set handle
         * The next two parameters are minimum and maximum advertising interval, both in
         * units of (milliseconds * 1.6).
         * The last two parameters are duration and maxevents left as default. */
        gecko_cmd_le_gap_set_advertise_timing(0, 160, 160, 0, 0);
```

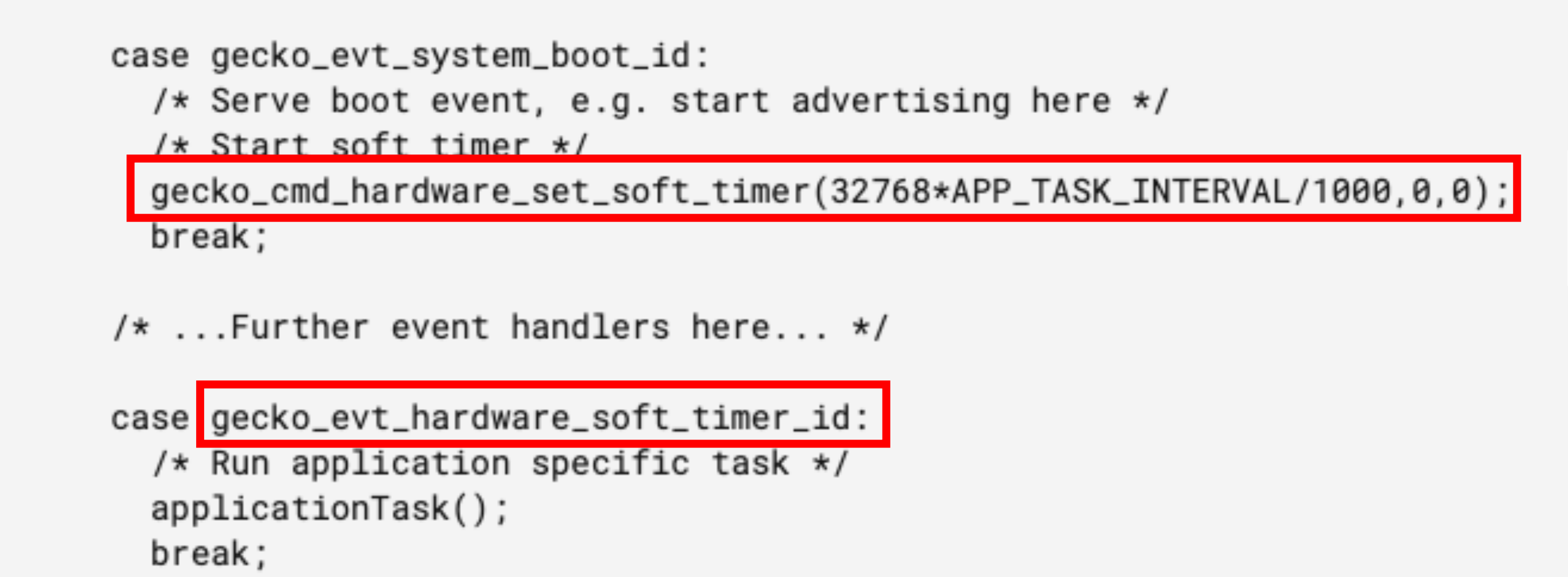
Built-in enabling user print logs.
Enabled at app.h.

By default it uses the Blocking Event Listener.

All the stack events are handled under this switch statement

Soft-Timers

- 16 Concurrent Soft-Timers
- Internal Stack Timer of the 32.768Khz oscillator
- One-shot or continuous operations

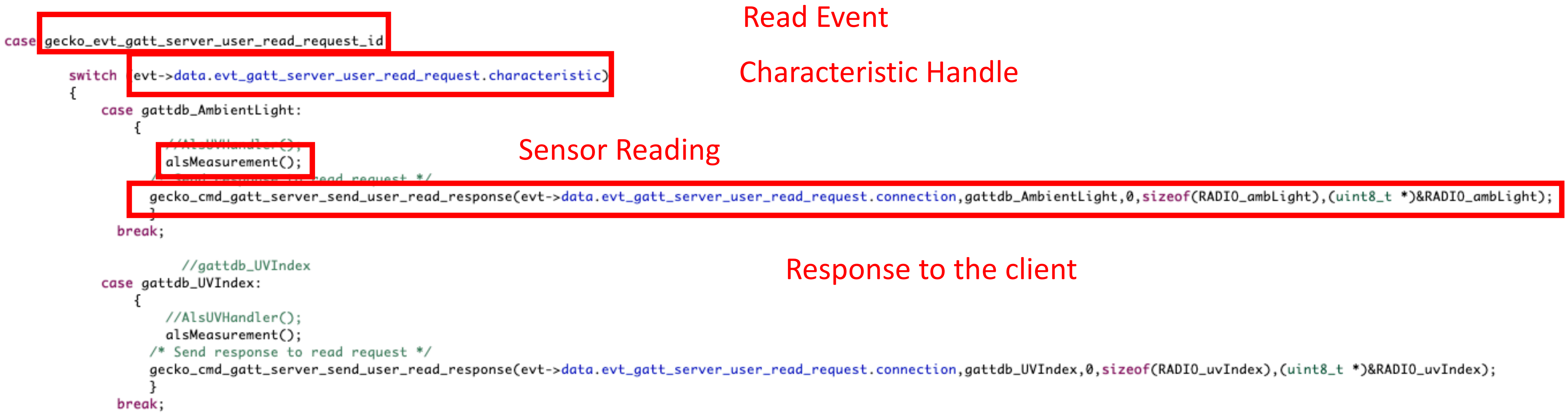


```
case gecko_evt_system_boot_id:
  /* Serve boot event, e.g. start advertising here */
  /* Start soft timer */
  gecko_cmd_hardware_set_soft_timer(32768*APP_TASK_INTERVAL/1000,0,0);
  break;

/* ...Further event handlers here... */

case gecko_evt_hardware_soft_timer_id:
  /* Run application specific task */
  applicationTask();
  break;
```

Soft-Timer API Command

Soft-Timer Event

GATT Characteristic Read Example (user type)



```
case gecko_evt_gatt_server_user_read_request_id

        switch (evt->data.evt_gatt_server_user_read_request.characteristic)
        {
            case gattdb_AmbientLight:
                {
                    //AlsUVHandler();
                    alsMeasurement();
                  /* Send response to read request */
                  gecko_cmd_gatt_server_send_user_read_response(evt->data.evt_gatt_server_user_read_request.connection,gattdb_AmbientLight,0,sizeof(RADIO_ambLight),(uint8_t *)&RADIO_ambLight);
                  }
              break;

                      //gattdb_UVIndex
            case gattdb_UVIndex:
                {
                    //AlsUVHandler();
                    alsMeasurement();
                  /* Send response to read request */
                  gecko_cmd_gatt_server_send_user_read_response(evt->data.evt_gatt_server_user_read_request.connection,gattdb_UVIndex,0,sizeof(RADIO_uvIndex),(uint8_t *)&RADIO_uvIndex);
                  }
              break;
```

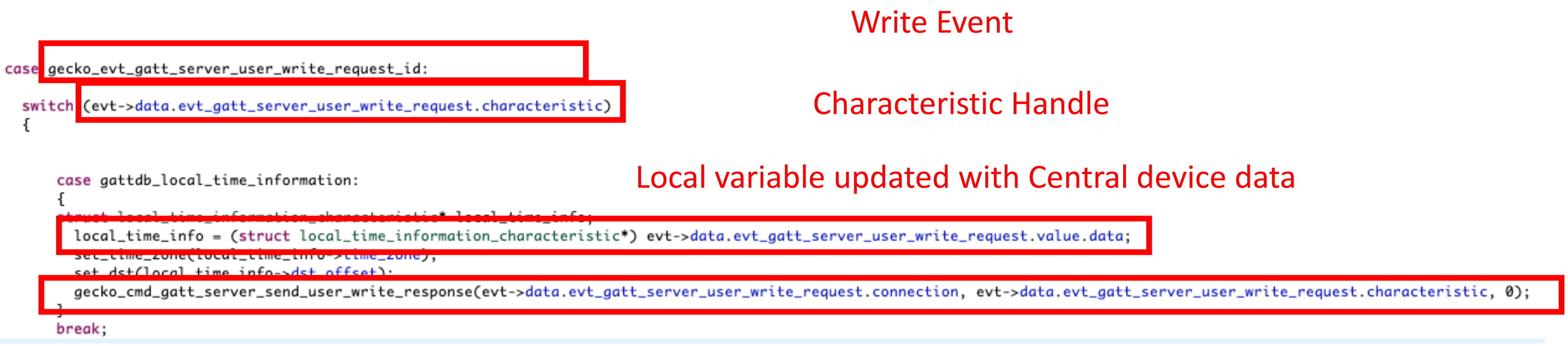
Read Event

Characteristic Handle

Sensor Reading

Response to the client

GATT Characteristic Write Example (user type)



Write Event

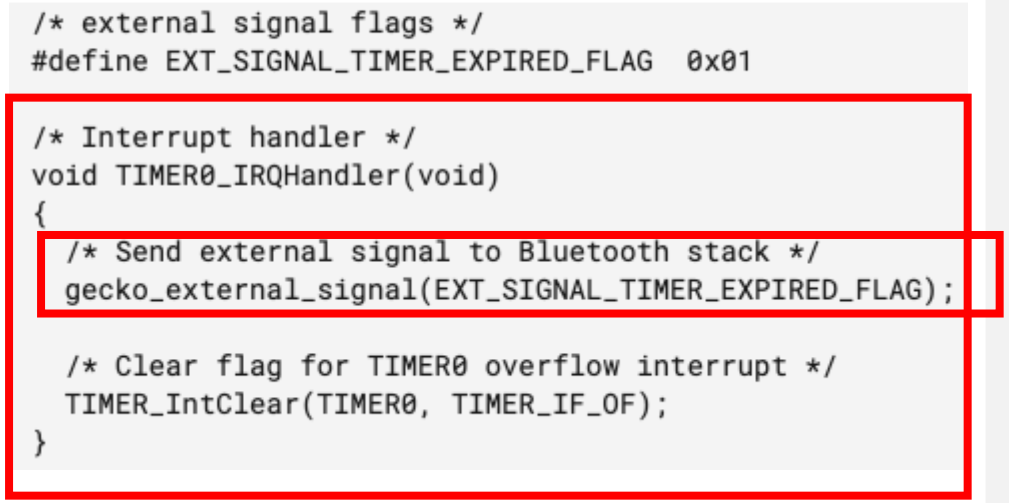
Characteristic Handle

Local variable updated with Central device data

```
case gecko_evt_gatt_server_user_write_request_id:

  switch (evt->data.evt_gatt_server_user_write_request.characteristic)
  {

      case gattdb_local_time_information:
      {
      struct local_time_information_characteristic* local_time_info;
        local_time_info = (struct local_time_information_characteristic*) evt->data.evt_gatt_server_user_write_request.value.data;
        set_time_zone(local_time_info->time_zone);
        set_dst(local_time_info->dst_offset);
        gecko_cmd_gatt_server_send_user_write_response(evt->data.evt_gatt_server_user_write_request.connection, evt->data.evt_gatt_server_user_write_request.characteristic, 0);
      }
      break;
```

Response to Central device

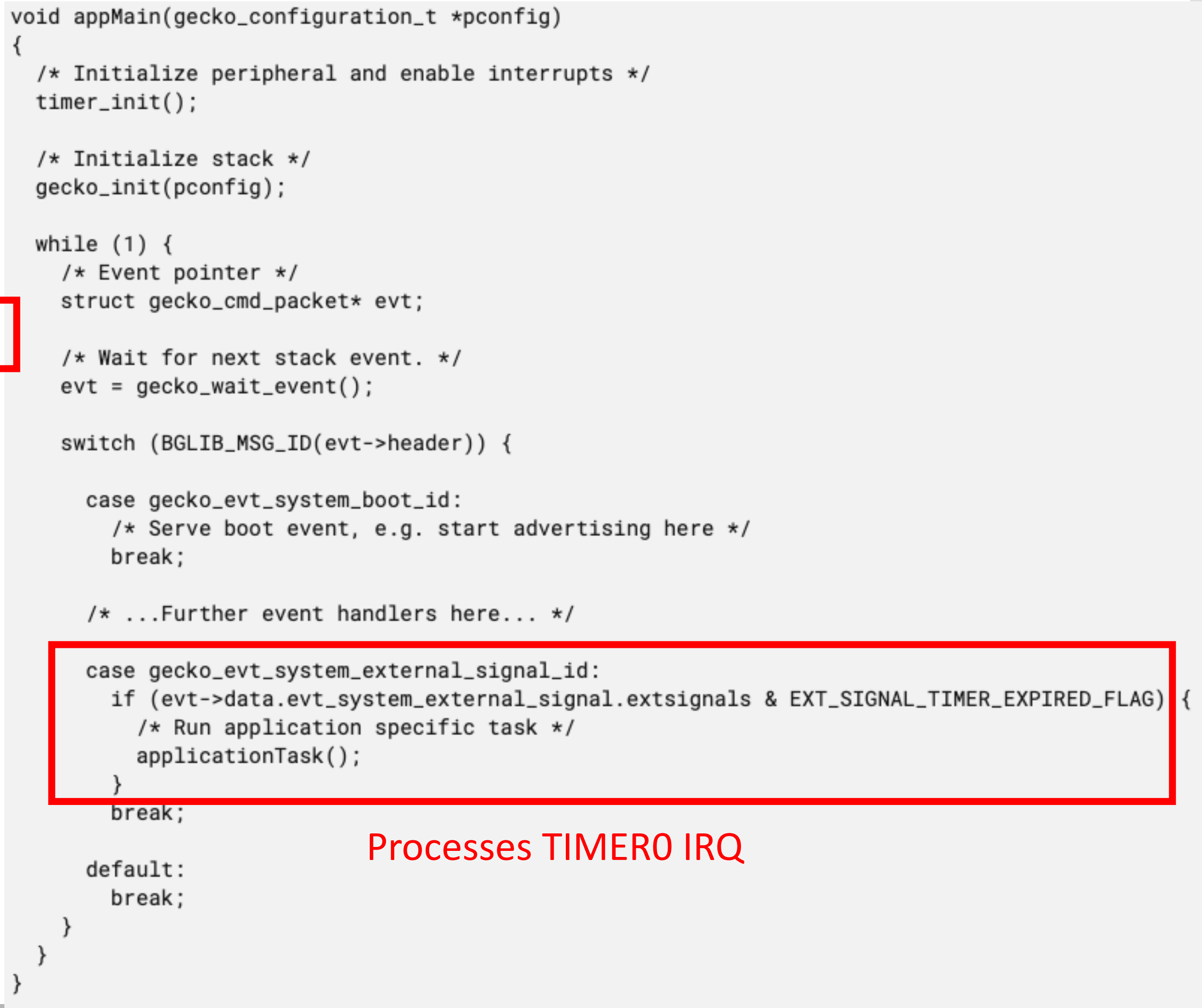
Managing a Hardware IRQ



```
/* external signal flags */
#define EXT_SIGNAL_TIMER_EXPIRED_FLAG  0x01

/* Interrupt handler */
void TIMER0_IRQHandler(void)
{
  /* Send external signal to Bluetooth stack */
  gecko_external_signal(EXT_SIGNAL_TIMER_EXPIRED_FLAG);

  /* Clear flag for TIMER0 overflow interrupt */
  TIMER_IntClear(TIMER0, TIMER_IF_OF);
}
```

TIMER0 IRQ

Adds TIMER0 IRQ to Stack External Event Handler



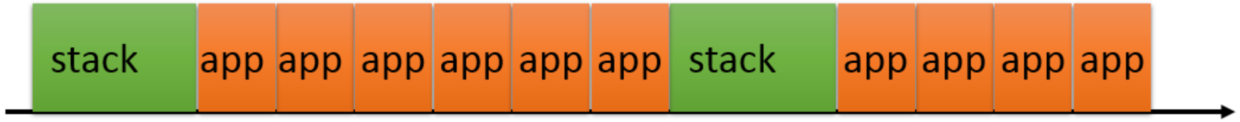
```
void appMain(gecko_configuration_t *pconfig)
{
  /* Initialize peripheral and enable interrupts */
  timer_init();

  /* Initialize stack */
  gecko_init(pconfig);

  while (1) {
    /* Event pointer */
    struct gecko_cmd_packet* evt;

    /* Wait for next stack event. */
    evt = gecko_wait_event();

    switch (BGLIB_MSG_ID(evt->header)) {

      case gecko_evt_system_boot_id:
        /* Serve boot event, e.g. start advertising here */
        break;

      /* ...Further event handlers here... */

      case gecko_evt_system_external_signal_id:
        if (evt->data.evt_system_external_signal.extsignals & EXT_SIGNAL_TIMER_EXPIRED_FLAG) {
          /* Run application specific task */
          applicationTask();
        }
        break;

      default:
        break;
    }
  }
}
```

Processes TIMER0 IRQ

Non-Blocking Event Listener

- The gecko_peek_event() function processes the internal message queue until an event is received or all the messages are processed.

- Advantages:
  - Flexibility for timing critical applications
  - Application tasks Run Continuously



- Disadvantages:
  - Power managed by the application
  - Prior to sleep Interrupts are disabled

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

Documents / User Guides

Main source of Documents - docs.silabs.com

Bluetooth Software API - https://www.silabs.com/documents/public/reference-manuals/bluetooth-api-reference.pdf

UG136: Silicon Labs *Bluetooth* ® C Application Developer's Guide - https://www.silabs.com/documents/public/user-guides/ug136-ble-c-soc-dev-guide.pdf

Qsg139: Getting started with Bluetooth - https://www.silabs.com/documents/public/quick-start-guides/qsg139-getting-started-with-bluetooth.pdf

AN1042: Using the Silicon Labs Bluetooth® Stack in Network Co-Processor Mode - https://www.silabs.com/documents/public/application-notes/an1042-bt-ncp-mode.pdf

Silicon Labs GitHub - https://github.com/siliconlabs

Hardware Peripheral Examples - https://github.com/SiliconLabs/peripheral_examples

BG22 Virtual Workshop

Hands-on Virtual Bluetooth Workshop
Develop a Secure IoT Device on a Thunderboard Kit
SILICON LABS

Learn how to develop and deploy more powerful, efficient, and secure IoT products with your own BG22 Thunderboard – free for all registrants!

New Sessions Open for June

10:00AM –11:30 AM CST - T, W, Th

(Other sessions available for Asia Pacific and Europe)

Register today! https://www.silabs.com/about-us/events/virtual-bluetooth-workshop

w/

works with

BY SILICON LABS

SEPTEMBER 9–10, 2020 | VIRTUAL

workswith.silabs.com

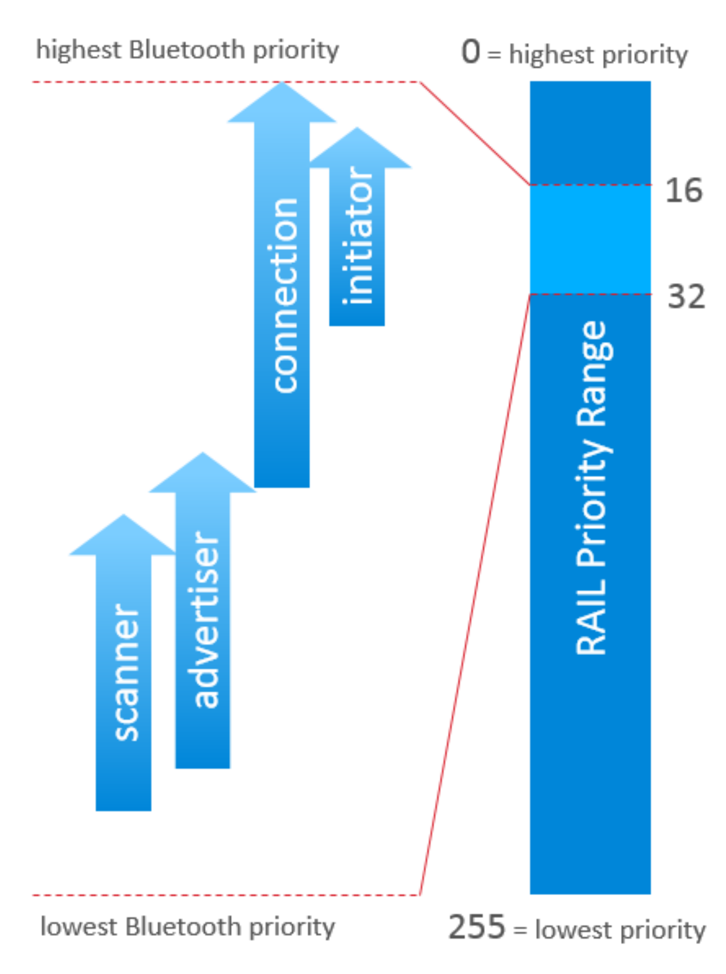
TWO DAYS OF TECHNICAL TRAINING FROM BEGINNER TO ADVANCED

Q & A Session

Backup Slides

Stack priorities

```
gecko_bluetooth_ll_priorities priorities = { 191, 143,  //scan_min, scan_max
                                             175, 127,  //adv_min,  adv_max
                                             135,   0,  //conn_min, conn_max
                                             55,  15,   //init_min, init_max
                                             175,       //threshold_coex
                                             16,        //rail_mapping_offset
                                             16,        //rail_mapping_range
                                             0,         //afh_scan_interval
                                             4,4        //adv_step, scan_step
                                           };

// Gecko configuration parameters (see gecko_configuration.h)
static const gecko_configuration_t config = {
  //...
  .bluetooth.linklayer_priorities = &priorities,
};
```



Source: Silicon Labs KBA_BT_0407: Bluetooth Radio Task Priorities

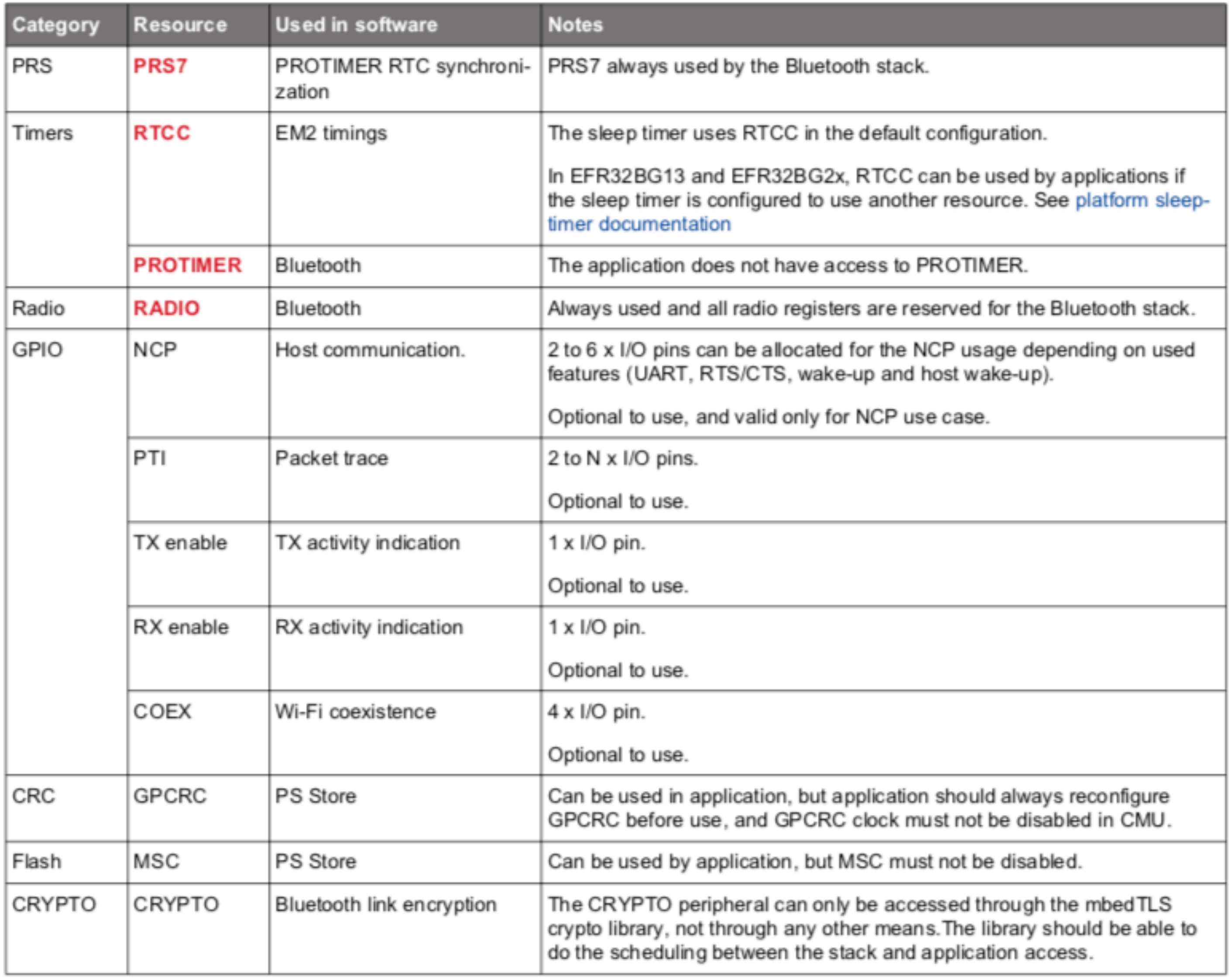
Blocking Event Listener

- The gecko_wait_event() is an implementation of a blocking wait function, which waits for events to emerge to the event queue and returns them to the event handler.

- Advantages:
  - It manages the sleep most efficiently and automatically.
  - It manages all the system timing keeping connections in sync

- Disadvantages:
  - Hardware Interrupts and external events need to be passed to the main event loop and to be processed within that loop
  - Applications can not run continuously

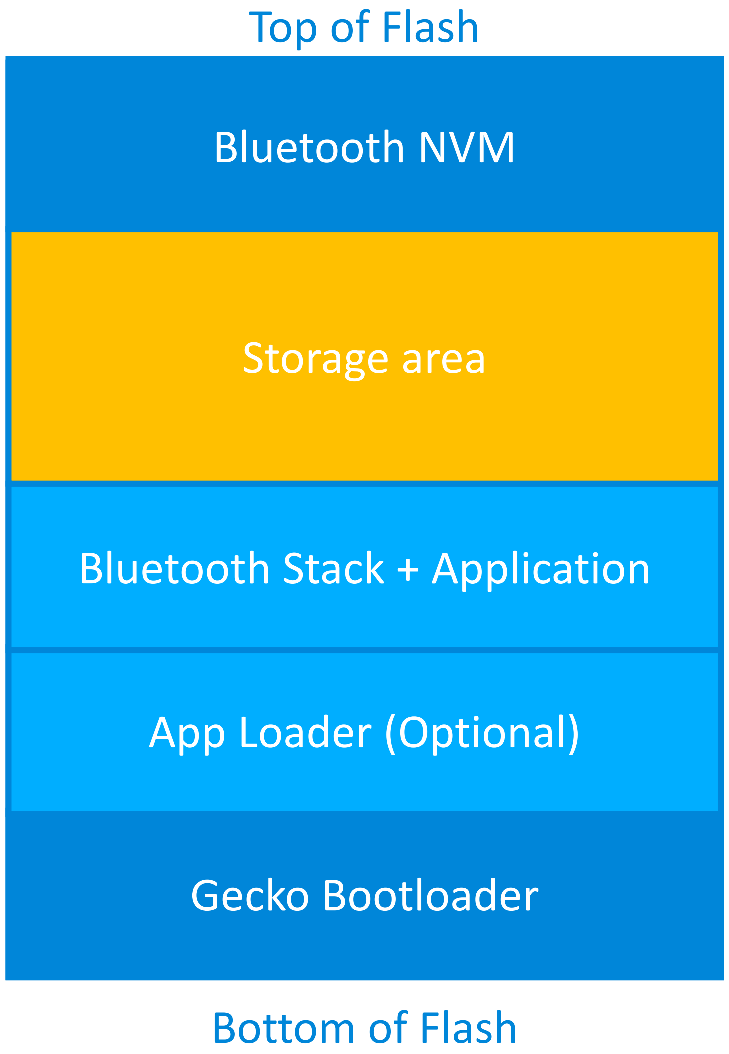
Bluetooth Stack required resources

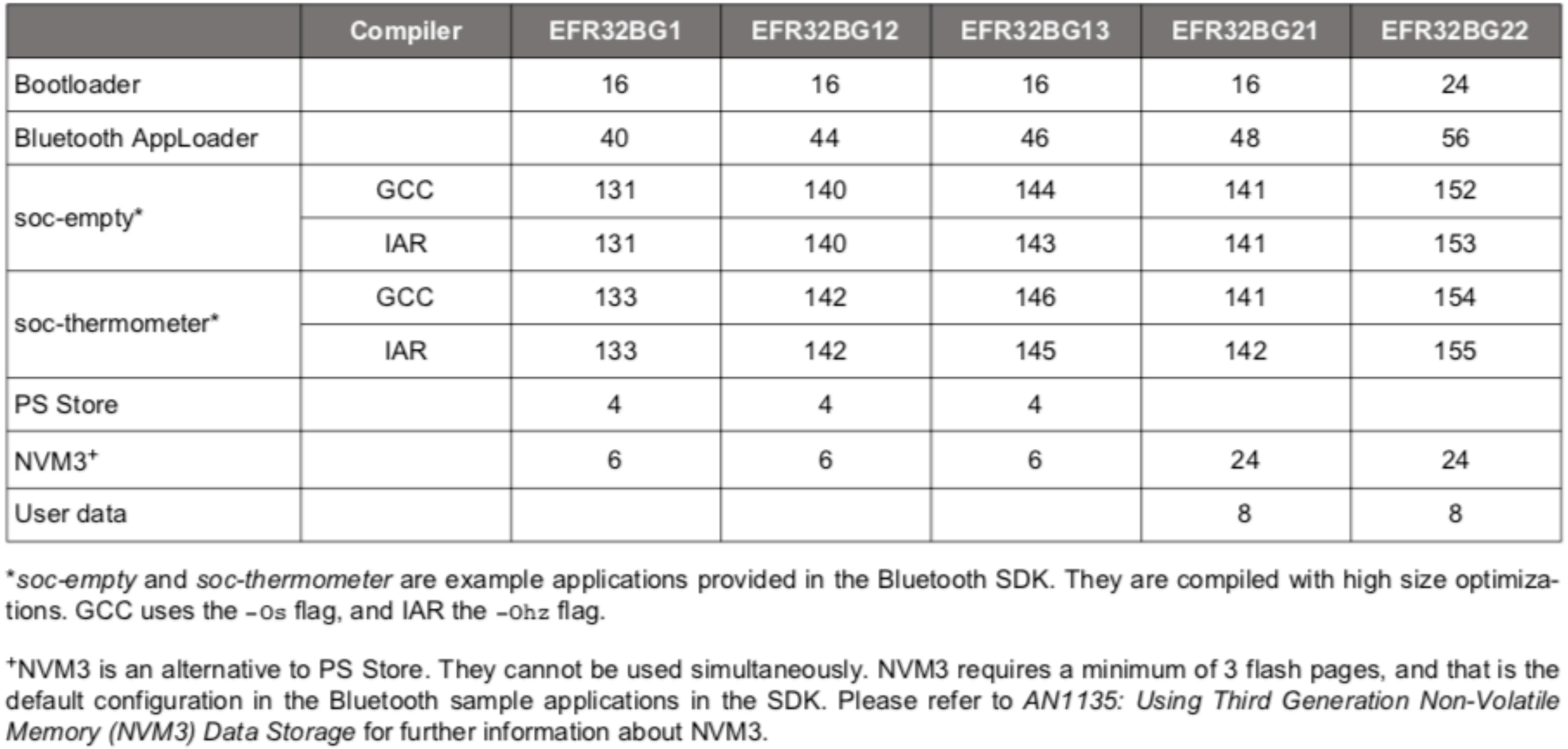


Category	Resource	Used in software	Notes
PRS	PRS7	PROTIMER RTC synchronization	PRS7 always used by the Bluetooth stack.
Timers	RTCC	EM2 timings	The sleep timer uses RTCC in the default configuration. In EFR32BG13 and EFR32BG2x, RTCC can be used by applications if the sleep timer is configured to use another resource. See platform sleep-timer documentation
	PROTIMER	Bluetooth	The application does not have access to PROTIMER.
Radio	RADIO	Bluetooth	Always used and all radio registers are reserved for the Bluetooth stack.
GPIO	NCP	Host communication.	2 to 6 x I/O pins can be allocated for the NCP usage depending on used features (UART, RTS/CTS, wake-up and host wake-up). Optional to use, and valid only for NCP use case.
	PTI	Packet trace	2 to N x I/O pins. Optional to use.
	TX enable	TX activity indication	1 x I/O pin. Optional to use.
	RX enable	RX activity indication	1 x I/O pin. Optional to use.
	COEX	Wi-Fi coexistence	4 x I/O pin. Optional to use.
CRC	GPCRC	PS Store	Can be used in application, but application should always reconfigure GPCRC before use, and GPCRC clock must not be disabled in CMU.
Flash	MSC	PS Store	Can be used by application, but MSC must not be disabled.
CRYPTO	CRYPTO	Bluetooth link encryption	The CRYPTO peripheral can only be accessed through the mbedTLS crypto library, not through any other means.The library should be able to do the scheduling between the stack and application access.

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

Device Memory organization and Flash Usage





	Compiler	EFR32BG1	EFR32BG12	EFR32BG13	EFR32BG21	EFR32BG22
Bootloader		16	16	16	16	24
Bluetooth AppLoader		40	44	46	48	56
soc-empty*	GCC	131	140	144	141	152
	IAR	131	140	143	141	153
soc-thermometer*	GCC	133	142	146	141	154
	IAR	133	142	145	142	155
PS Store		4	4	4		
NVM3⁺		6	6	6	24	24
User data					8	8

* *soc-empty* and *soc-thermometer* are example applications provided in the Bluetooth SDK. They are compiled with high size optimizations. GCC uses the `-Os` flag, and IAR the `-Ohz` flag.

⁺NVM3 is an alternative to PS Store. They cannot be used simultaneously. NVM3 requires a minimum of 3 flash pages, and that is the default configuration in the Bluetooth sample applications in the SDK. Please refer to *AN1135: Using Third Generation Non-Volatile Memory (NVM3) Data Storage* for further information about NVM3.

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