

# AN1419: *Bluetooth*<sup>®</sup> LE Electronic Shelf Label



This version of AN1419 has been deprecated. For the latest version, see [docs.silabs.com](https://docs.silabs.com).

\*\*\*\*\*

The Bluetooth Special Interest Group released the Electronic Shelf Label (ESL) profile and service specification at the beginning of 2023. With the essential additions in the Bluetooth Core Specification version 5.4 (e.g., Periodic Advertising with Responses and Encrypted Advertising Data), the ESL specification defines a standardized use of Bluetooth Low Energy (BLE) in electronic shelf labels.

This document introduces the Bluetooth - SoC ESL Tag example project, accompanied by an Access Point host application through examples, which describes the essential features of the BLE ESL. In addition, this document also describes how the ESL network can be configured, through multiple examples. And finally, the document also describes the first steps for how to modify the sample application, extending it to use an additional, or different, display, and using non-volatile memory for storing images.

## KEY POINTS

- Preparing the ESL Tags and access point
- Using the ESL
- Extending the ESL example with an additional display

## 1 Introduction

Chapter [2 Prerequisites](#) provides all prerequisites needed to build, program, and use the Electronic Shelf Label (ESL) Tags and access point (AP).

Chapter [3 Documentation](#) describes where the ESL API documentation and detailed ESL access point documentation can be found.

Chapter [4 Bluetooth ESL](#) provides a detailed introduction to BLE Electronic Shelf Label (ESL) Tags.

Chapter [5 Preparing the ESL Network](#) provides the steps to prepare the ESL access point (Section [5.1 ESL Access Point](#)) and how to build the **Bluetooth – SoC ESL Tag** example project and program it to the target device (Section [5.2 ESL Tags](#)).

Chapter [6 The ESL Network](#) describes how to run the ESL network, both in automatic and manual mode. Section [6.2 ESL AP in Manual Mode](#) focuses on the different states an ESL Tag can be in, and how to do the transitions between these with the access point implementation. All available commands are introduced, most of them with an example.

And finally, chapter [7 Modifying the Bluetooth ESL example](#) provides information on how to extend the example code with an additional display and the relevant configurations and API calls.

## 2 Prerequisites

Requirements:

- A computer with:
  - Latest version of Simplicity Studio v5
  - Python 3 (minimum and preferred version 3.9) [1]
  - Python packages listed on `<sdk_path>/app/bluetooth/example_host/bt_host_esl_ap/requirements.txt`.
- One EFR BLE radio board to be used with access point
- One or more WSTK + radio board sets as ESL Tag (BRD4182A radio board recommended). The ESL example project uses the WSTK LCD screen as the ESL display.

For supported radio boards, see Appendix [A Supported Radio Boards](#).

### 3 Documentation

Full API documentation can be found at [docs.silabs.com](https://docs.silabs.com), or alternatively accessed within Simplicity Studio v5. Select the device in use, browse to the **DOCUMENTATION** tab in Launcher view, and search for "Silicon Labs Bluetooth API Reference Guide".

The detailed access point documentation is in folder:

`<SDK path>/app/bluetooth/example_host/bt_host_esl_ap/readme/`, **file** `readme.md`.



## 4 Bluetooth ESL

### 4.1 Periodic Advertising with Responses

This chapter describes the basics of the Bluetooth LE Periodic Advertising with Responses (PAwR) feature, how it is used in the BLE ESL application, and basics regarding how to form a PAwR network in BLE ESL.

#### 4.1.1 Introduction

Advertising in Bluetooth is commonly used between a Peripheral and a Central device for the Peripheral to indicate its aim to connect with the Central device. As an example, a Bluetooth headset (Peripheral, Advertiser) is advertising Connectable Advertisements and a mobile phone (Central device, Scanner) scans these advertisements. If an advertisement is received, the Central device establishes a Bluetooth Connection with the Peripheral device.

Bluetooth also defines "Connectionless" communication mode where a device advertises non-connectable advertisement packets, which can be received by the scanning device. An example of such an application is, for example, an asset tracking device, which is broadcasting (Broadcaster) the asset-specific identification information. An Observer device receiving these packets then detects that the broadcasting device is in proximity of the scanning device and then receives these broadcasted messages.

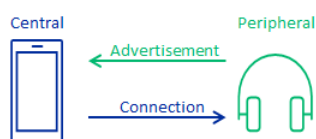
Bluetooth 5.0 defined a Periodic Advertisement feature. Periodic Advertising allows Non-connectable Advertisements to be sent at a fixed interval where advertising data can be sent. One or more Observers can then listen and receive these advertisements. Because the transmissions happen at a fixed interval, the Observer (and Broadcaster) can go to sleep between advertisement events instead of being in receive mode (Scanning) all the time. Periodic Advertisement could be used for example in a sensor network where sensor peripherals are advertising the sensor data periodically and a Central device can receive the sensor data without the need to scan all the time.

PAwR is a BLE extension (released in BT Core 5.4) to Periodic Advertising. In Periodic Advertising, the data communication is limited to be from the Broadcaster to the Observer. In PAwR, the roles of Broadcaster and Observer are swapped. The Central device becomes a Broadcaster, which advertises periodically by keeping up the PAwR train of packets. The Observer becomes a Peripheral device synchronized to the periodic advertisement train. Data from the Central device to the Peripheral device can be sent over the PAwR. The Peripheral device may send data to the Central device by placing the data in the dedicated Response Slot(s). PAwR makes it possible for thousands of Peripheral devices to synchronize to the Central device's periodic advertisement and allows exchange data between the Central device and Peripherals without the need to establish a connection.

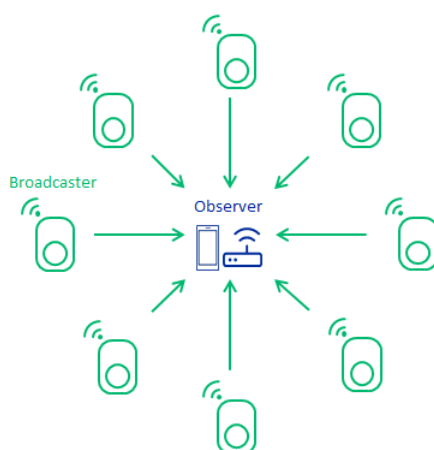
Peripheral devices keep synchronization to the Central device by receiving the PAwR train transmitted by the Central device. At a minimum, the PAwR train transmitted by the Central device consists of an empty Advertising physical channel PDU.

The theoretical maximum payload size in PAwR packet is 251 bytes of Advertising Data. As an example, in the Electronic Shelf Label application (BT SIG defined Profile/Service), the payload is limited to 48 bytes.

#### Advertising for Connection (irregular, unidirectional)



#### One-way "Beaconing" (regular, unidirectional)



#### Periodic Advertising with Responses (regular, bidirectional)

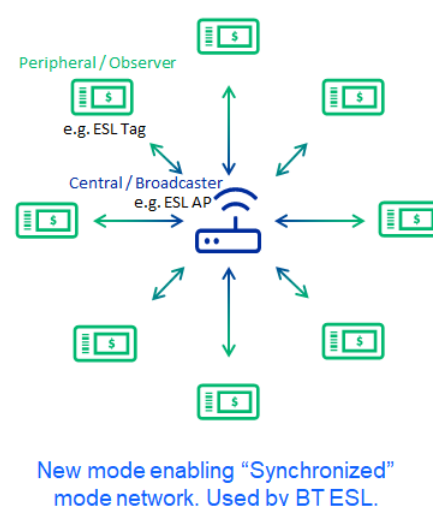
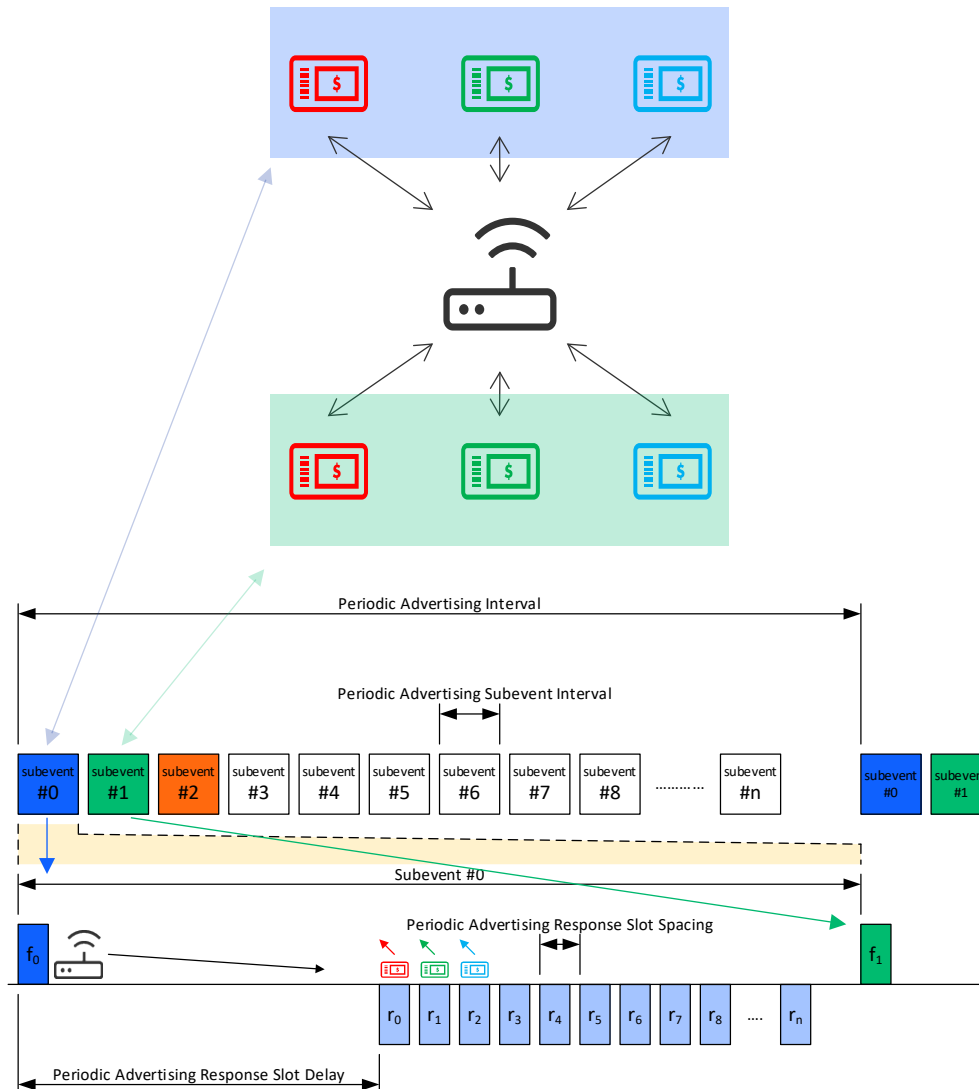


Figure 1. BLE Connection Modes

### 4.1.2 Periodic Advertising with Responses Basics

Periodic Advertising with Responses (PAwR) is kind of a Time Division Multiple Access (TDMA) system, where communication between nodes (single Central device and one or more Peripheral devices) happen in predefined time slots. The fundamental timing period is the "Periodic Advertising Interval" which is the interval of periodic advertisements sent by the Central device. Each Peripheral device receives the periodic advertisement once in the Periodic Advertising Interval to keep synchronization to the PAwR train.

Each Periodic Advertising Interval is divided into 1-128 Subevents and each Subevent to 0-255 Response Slots. In theory  $128 \times 255 = 32,640$  Peripheral devices may be mapped to one PAwR train. The Periodic Advertising Interval can be between 7.5ms to ~82 seconds. The figure below shows the Periodic Advertising Interval divided into Subevents.



**Figure 2. Basics of Periodic Advertising with Responses**

In each Subevent, the Central device transmits one Periodic Advertisement to all Peripherals in that particular Subevent. Each Peripheral has its unique Response Slot where it may send data to the Central device. The allocation of the Response Slots is application-specific. PAwR itself doesn't specify in any means the content of the data or the protocol for how the data should be handled by any of the devices.

### 4.1.3 PAwR Timing

PAwR introduces several timing parameters to define the different intervals and delays. This allows endless possibilities to define the best parameters to meet the system/application bandwidth, latency, and energy consumption requirements.

The following timing parameters exist in PAwR:

- Periodic Advertising Interval (7.5ms - 81.91875sec)
  - Interval of the periodic advertisement train. PAI must be defined so that all the allocated Subevents fit into the PAI.
- Periodic Advertising Subevent Interval (7.5ms - 318.72ms)
  - Periodic Advertising Subevent Interval is the duration of one Subevent. All Subevents have the same duration. One Subevent includes one transmission from the Central device and possible Response Slots from the Peripherals. Periodic Advertising Subevent Interval must be defined so that all allocated Response Slots and the Periodic Advertising Response Slot Delay (see next bullet) fit into one Subevent.
- Periodic Advertising Response Slot Delay (1.25ms - 317.5ms)
  - Periodic Advertising Response Slot Delay is the duration between the Central device transmission to the first Response Slot. Periodic Advertising Response Slot Delay must be defined so that the Central device has enough time to transmit its message and be ready to receive the response in the first Response Slot.
- Response Slot Spacing (0.25ms – 31.875ms)
  - Response Slot Spacing is the duration of one Response Slot. All Response Slots have the same duration. One Response Slot includes one transmission from one Peripheral device to the Central device. Response Slot Spacing must be defined so that the whole Peripheral response fits into the Response Slot including the T\_IFS (Inter Frame Space, min 150us).

Once the timing parameters of the PAwR are defined, the PAwR train is set up and Peripheral devices are synchronized after which the timing parameters cannot be (dynamically) changed.

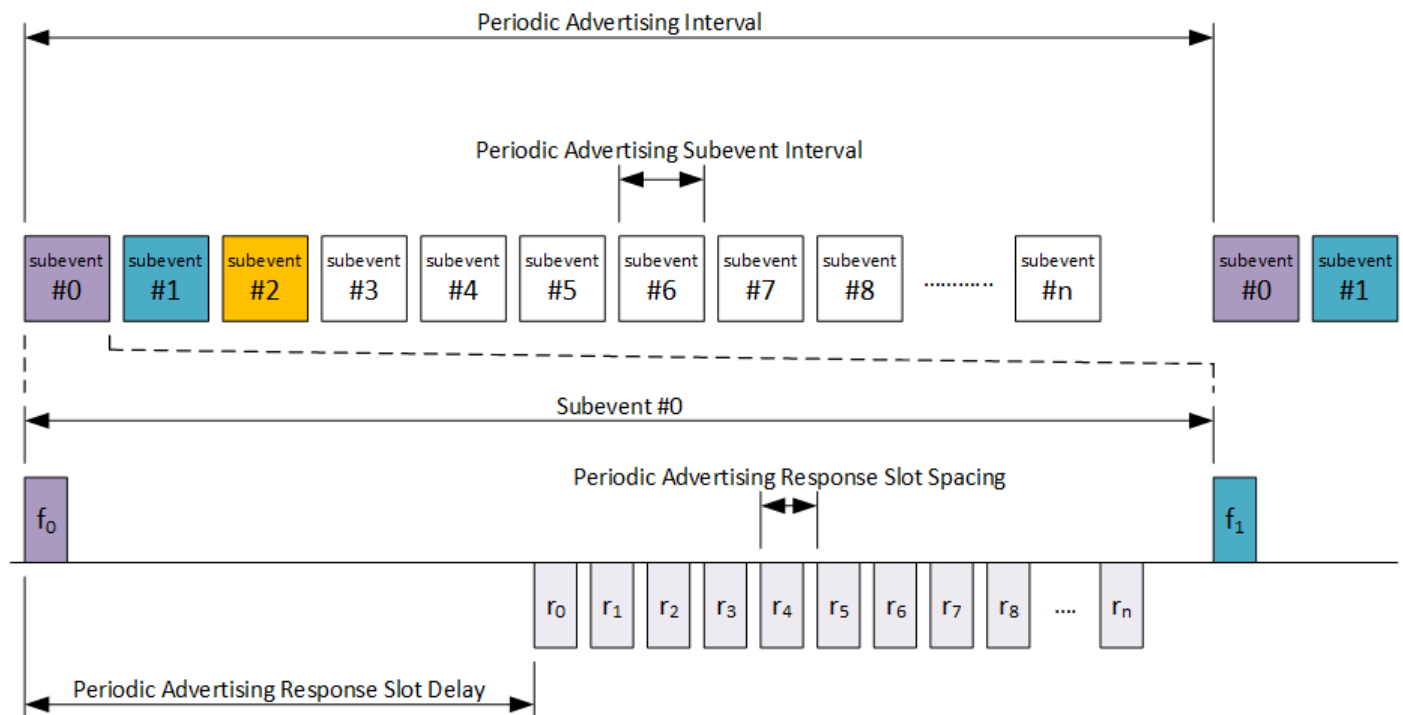


Figure 3. PAwR Timings

### 4.1.4 PAwR in BLE ESL

In BLE ESL, each ESL Tag belongs to one Subevent, in ESL context, called a Group. A maximum of 255 ESL Tags can be mapped to one Group (=Subevent). The ESL Service/Profile determines how the Response Slots are allocated for each ESL Tag. Each ESL doesn't have its own predefined Response Slot; instead, the Response Slot is determined by the order of commands (TLVs, Tag Length Value) sent by the Central device (Access Point). More than one ESL Tag can be addressed in one Subevent. The first addressed command requiring the ESL Tag to respond is responded in the first Response Slot, the second addressed command responded in the second Response Slot, and so on. If there are multiple commands to one ESL Tag in a single AP message, the allocated Response Slot is based on the last addressed command. For example, ESL A receives command in 1<sup>st</sup> and 3<sup>rd</sup> TLV. ESL B receives command in 2<sup>nd</sup> TLV. The ESL A responds in 3<sup>rd</sup> Response Slot and ESL B in 2<sup>nd</sup> Response Slot.

The maximum payload size in BLE ESL Service/Profile is 48 bytes. The minimum command size is 2 bytes. Each ESL Payload starts with Group ID, which is 1 byte. This means that a maximum of 23 ESL Tags can be addressed in one Subevent, meaning that a

maximum of 23 ESL Tags may have a respond in one Subevent. From the PAwR perspective, there may be more ESL Tags mapped to one Subevent than there are Response Slots allocated. In other words, a maximum of 23 Response Slots needs to be allocated to one Subevent because only a maximum of 23 ESL Tags can respond in one Subevent. This is an important factor when determining the PAwR parameters for the ESL Tag network. The maximum command size is 17 bytes, meaning a maximum of two commands fit to one message.

PAwR communication is not the only radio communication happening between the Access Point (Central device) and Peripheral devices (ESL Tags). Enough radio time for the Access Point must be reserved for other BLE activities like Scanning and Connection. This must be taken into account when determining the PAwR parameters. The Periodic Advertising Interval determines the overall latency of the system, because each ESL Tag has the opportunity to be accessed once in a Periodic Advertising Interval. On the other hand, each ESL Tag must listen/receive its dedicated Subevent to keep synchronized to the PAwR train. The Periodic Advertising Interval is then a trade of parameter between the system latency and ESL Tag power consumption. A short Periodic Advertising interval means short latency but higher power consumption and vice versa. Usually, a 1 to 10 second interval is used in similar types of proprietary ESL networks.

Some examples of how to select the PAwR timing parameters and how to allocate the Subevents are introduced below.

## 1000 ESL Tags, 4 Second Periodic Advertising Interval, “Packed”

In this example, 1000 ESL are mapped to minimum time Subevent. 50 Subevents are allocated, meaning 20 ESL Tags are mapped to one Subevent. Periodic Advertising Subevent Interval is calculated to fit 20 Response Slots. Response Slot Duration is defined to fit maximum ESL payload, additional protocol overhead and Inter Frame Space ( $T_{IFS} = 150\mu s$ ) between the Response Slots.

This mapping approach results to one 2.7 second slot for Connections and Scanning.

The access time for one ESL Tag is the maximum Periodic Advertising Interval, 4 seconds. Accessing individually all ESL Tags in the network will take 1 to 10 Periodic Advertising Intervals (4 – 10 seconds).

- Number Subevents
  - 50
- Number of Response Slots (= ESL Tags per Group)
  - 20
- Response Slot Duration
  - 1ms
- Periodic Advertising Subevent Interval
  - 26.25ms
- 5ms Periodic Advertisement Response Slot Delay
  - 20 Response Slots x 1ms = 20ms for Response Slots
  - 1.25ms gap between Subevents
- Periodic Advertising Interval
  - 4 seconds
  - 50 Subevents x 26.25ms = 1.3125 seconds
  - 2.6875 seconds reserved for Connection & Scanning
- Single ESL Access Time
  - 4 seconds
- Total Time to access 1000 ESLs:
  - Minimum command size (2 bytes)
    - $\text{ROUNDUP}(20 / 23) = 1$
    - $\Rightarrow 1 \text{ Advertising Intervals} \times 4 \text{ sec} = 4 \text{ seconds}$
  - Maximum command size (17 bytes)
    - $\text{ROUNDUP}(20 / 2) = 10$
    - $\Rightarrow 10 \text{ Advertising Intervals} \times 4 \text{ sec} = 40 \text{ seconds}$

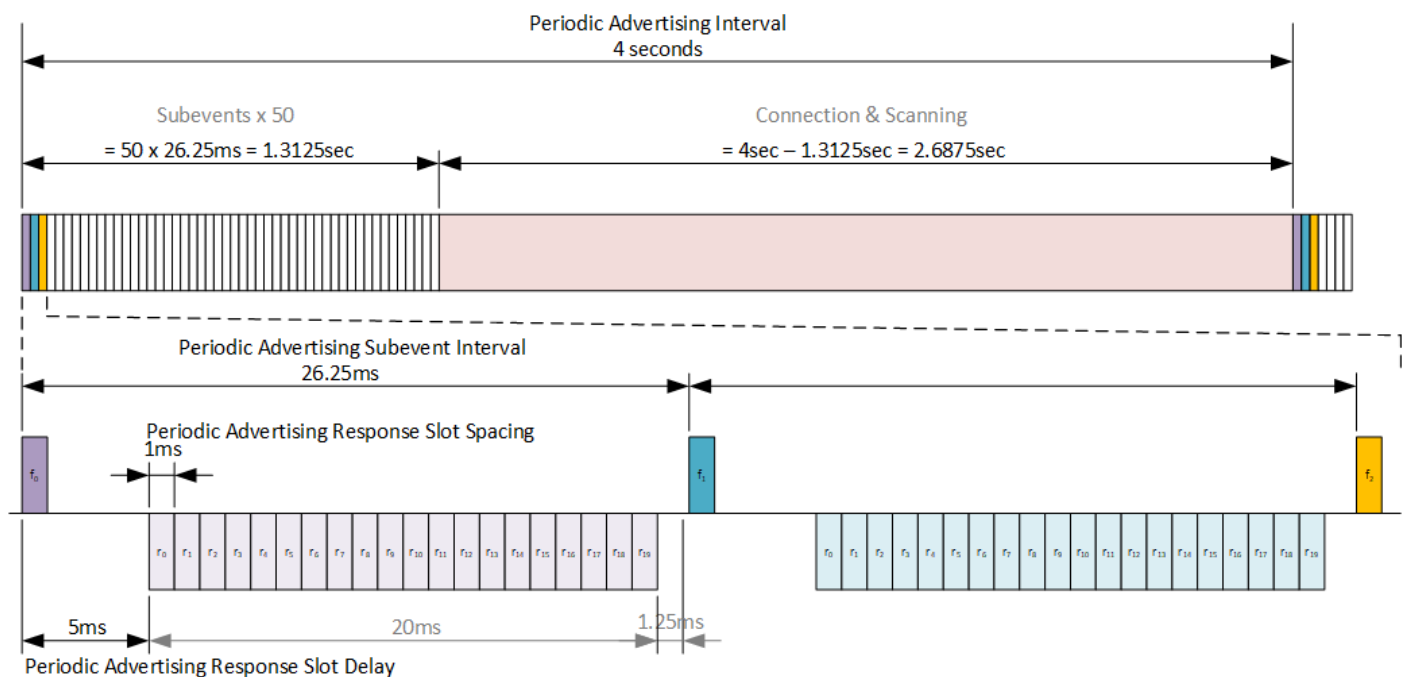


Figure 4. PAwR Timing Example: “Packed”, 1000 ESLs

### 1000 ESL Tags, 4 Second Periodic Advertising Interval, “Evenly Distributed”

In this example, 1000 ESL are mapped to evenly along the Periodic Advertising Interval. 50 Subevents are allocated, meaning 20 ESL Tags are mapped to one Subevent. Periodic Advertising Subevent Interval is calculated by dividing the Periodic Advertising Interval to equal length Subevents. Response Slot Duration is defined to fit maximum ESL payload, additional protocol overhead and Inter Frame Space ( $T_{IFS} = 150\mu s$ ) between the Response Slots.

This mapping approach results to at least 55ms slot for Connections and Scanning every 80ms (Subevent).

The access time for one ESL Tag is the maximum Periodic Advertising Interval, 4 seconds. Accessing individually all ESL Tags in the network will take 1 to 10 Periodic Advertising Intervals (4 – 10 seconds).

- Number Subevents
  - 50
- Number of Response Slots (= ESL Tags per Group)
  - 20
- Periodic Advertising Interval
  - 4 seconds
- Response Slot Duration
  - 1ms
- Periodic Advertising Subevent Interval
- $4000ms / 50 = 80ms$ 
  - 5ms Periodic Advertisement Response Slot Delay
  - 20 Response Slots x 1ms = 20ms for Response Slots
  - 55-75ms gap between Subevents for Connection & Scanning
- Single ESL Access Time
  - 4 seconds
- Total Time to access 1000 ESLs:
  - Minimum command size (2 bytes)
    - $ROUNDUP(20 / 23) = 1$
    - $\Rightarrow 1$  Advertising Intervals x 4 sec = 4 seconds
  - Maximum command size (17 bytes)
    - $ROUNDUP(20 / 2) = 10$
    - $\Rightarrow 10$  Advertising Intervals x 4 sec = 40 seconds

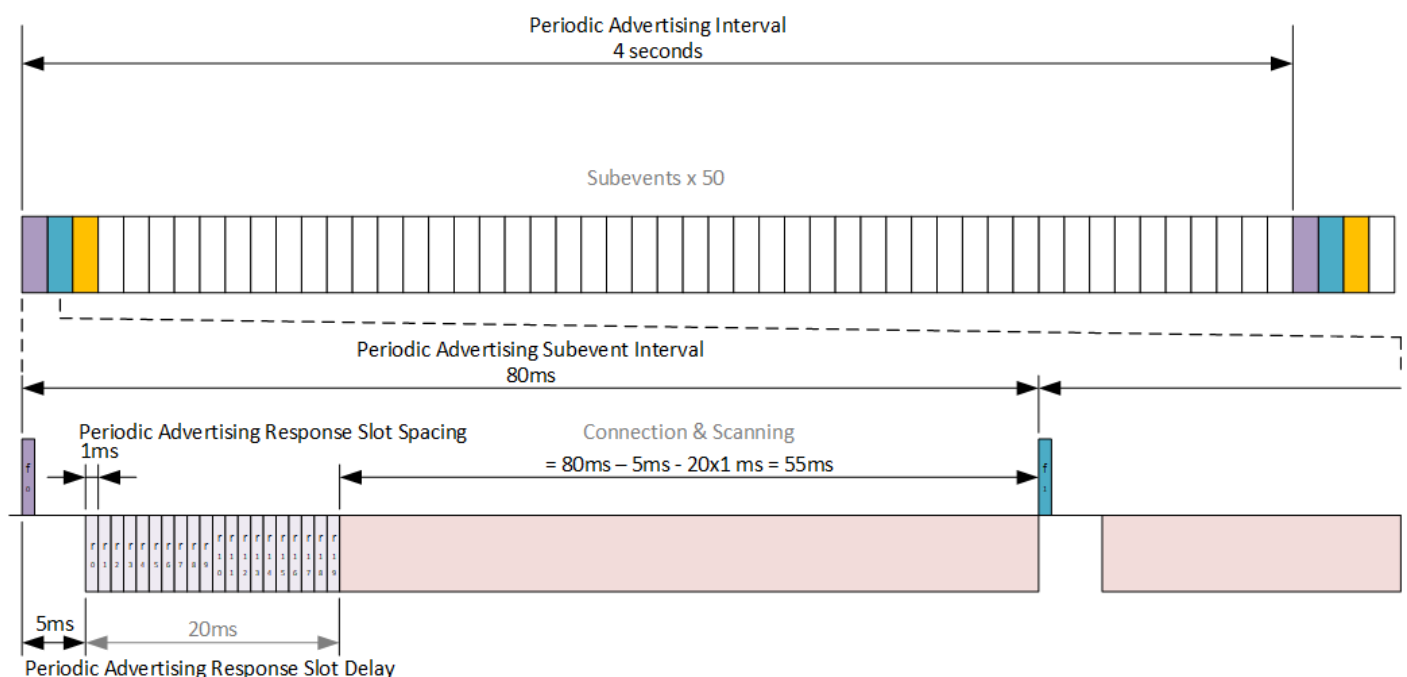


Figure 5. PAwR Timing Example: “Evenly Distributed”, 1000 ESLs

### 10000 ESL Tags, 4 Second Periodic Advertising Interval, “Packed”

In this example, 10000 ESL are mapped to the minimum time Subevent. 50 Subevents are allocated, meaning 200 ESL Tags are mapped to one Subevent. Periodic Advertising Subevent Interval is calculated to fit 23 Response Slots. Response Slot Duration is defined to fit the maximum ESL payload, additional protocol overhead and Inter Frame Space ( $T_{IFS} = 150\mu s$ ) between the Response Slots.

This mapping approach results to one 2.6 second slot for Connections and Scanning.

The access time for one ESL Tag is maximum the Periodic Advertising Interval, 4 seconds. Accessing individually all ESL Tags in the network will take 1 to 10 Periodic Advertising Intervals (4 – 10 seconds).

- Number Subevents
  - 50
- Number of Response Slots
  - 23 (200 ESL Tags per Group)
- Response Slot Duration
  - 1ms
- Periodic Advertising Subevent Interval
  - 28.75ms
- 5ms Periodic Advertisement Response Slot Delay
  - 23 Response Slots x 1ms = 23ms for Response Slots
  - 0.75ms gap between Subevents
- Periodic Advertising Interval
  - 4 seconds
  - 50 Subevents x 28.75ms = 1.4375 seconds
  - 2.5625 seconds reserved for Connection & Scanning
- Single ESL Access Time
  - 4 seconds
- Total Time to access 10000 ESLs:
  - Minimum command size (2 bytes)
    - $\text{ROUNDUP}(200 / 23) = 9$
    - $\Rightarrow 9 \text{ Advertising Intervals} \times 4 \text{ sec} = 36 \text{ seconds}$
  - Maximum command size (17 bytes)
    - $\text{ROUNDUP}(200 / 2) = 100$
    - $\Rightarrow 100 \text{ Advertising Intervals} \times 4 \text{ sec} = 400 \text{ seconds}$

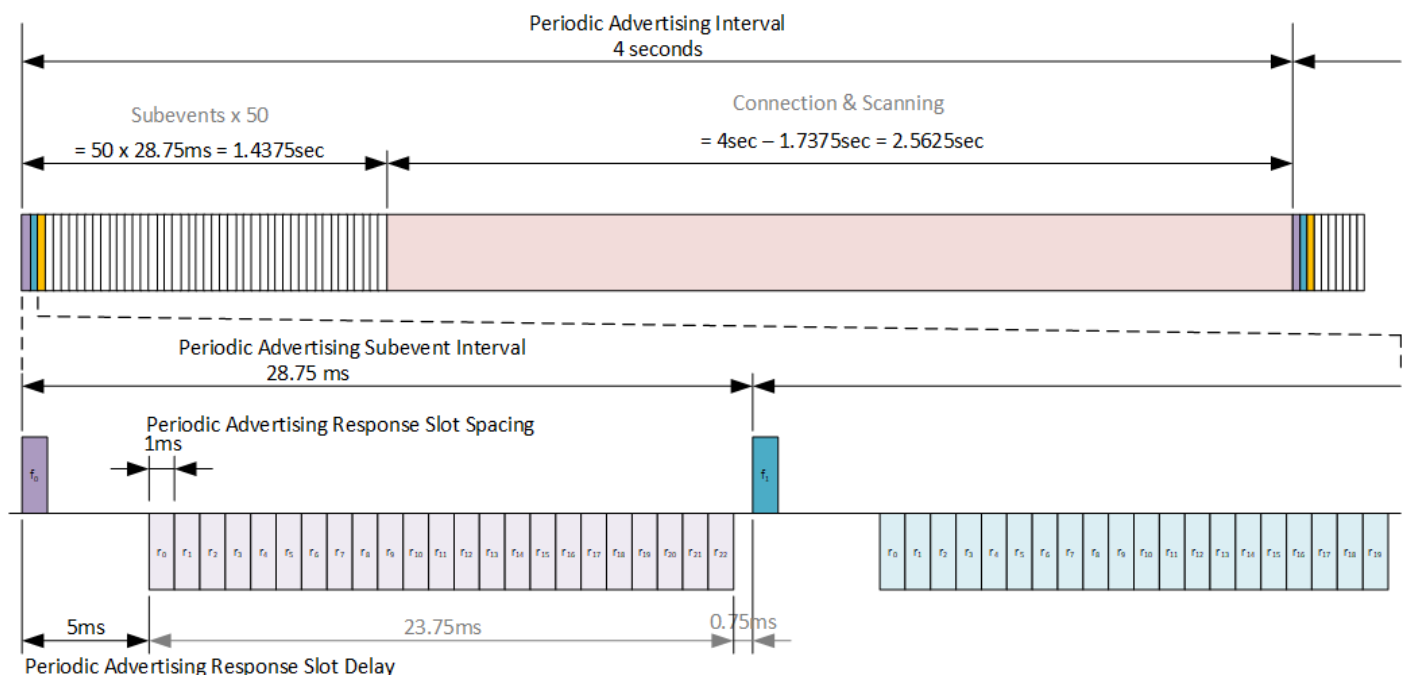


Figure 6. PAwR Timing Example: “Packed”, 10000 ESLs

4.2 Encrypted Advertising Data (EAD)

In addition to PAwR, Bluetooth 5.4 introduced another feature which is essential for the Bluetooth Electronic Shelf Label service: Encrypted Advertising Data (EAD). The EAD is the first standardized method to use encryption for the advertisement data (also with responses), making the communication secure in connectionless modes. Prior to Bluetooth 5.4, encryption was only defined for connection-orientated communication.

The encrypted data will be encapsulated within an Advertising Data structure, as shown in Figure 7 below. Bluetooth uses CCM algorithm to encrypt and authenticate the data, which includes the ESL specific payload, a 40-bit long Randomizer field (i.e., 5 octets of random data), and a 32-bit long Message Integrity Check (MIC) field.

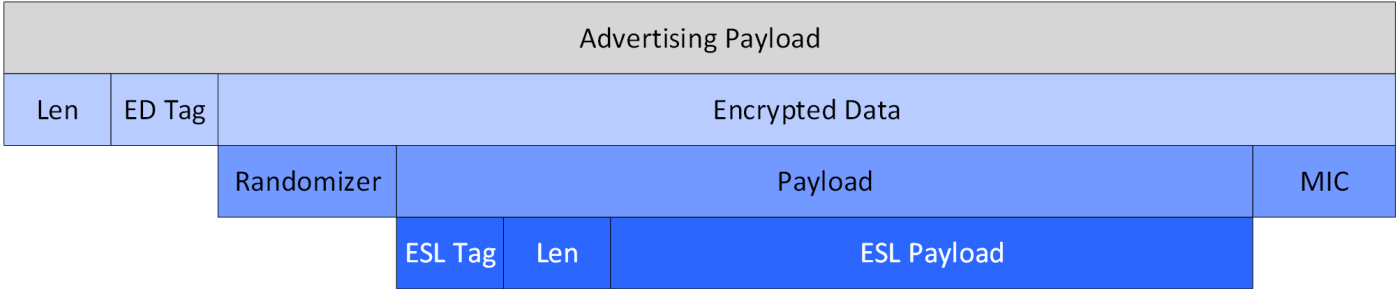


Figure 7. Encrypted Advertising Data Format



## 5 Preparing the ESL Network

### 5.1 ESL Access Point

All the core functionalities of the ESL access point (AP) are implemented as C program, with some additional parts, e.g., the command line interface, implemented with Python. The host program utilizes BGAPI binary protocol to control an EFR32 based radio in Network Co-processor (NCP) mode. Thus, the ESL AP requires an EFR32 based BLE device, and a host (PC) running the access point implementation. For more information about NCP, refer to the application note *AN1259: Using the Silicon Labs Bluetooth® Stack v3.x and Higher in Network Co-Processor Mode* [2].

The access point implementation has been documented in readme file, located in:

```
<SDK>/app/bluetooth/example_host/bt_host_esl_ap/readme/
```

#### 5.1.1 Install Python and Required Packages

Silicon Labs recommends using Python version 3.9, which can be downloaded from [www.python.org](https://www.python.org) [1]. In addition to the Python itself, the access point requires additional packages listed in the file `requirements.txt`, located at `<sdk path>/app/bluetooth/example_host/bt_host_esl_ap/`. The required packages can be installed by using the `requirements.txt` file.

On Linux / Mac, run the following from command line:

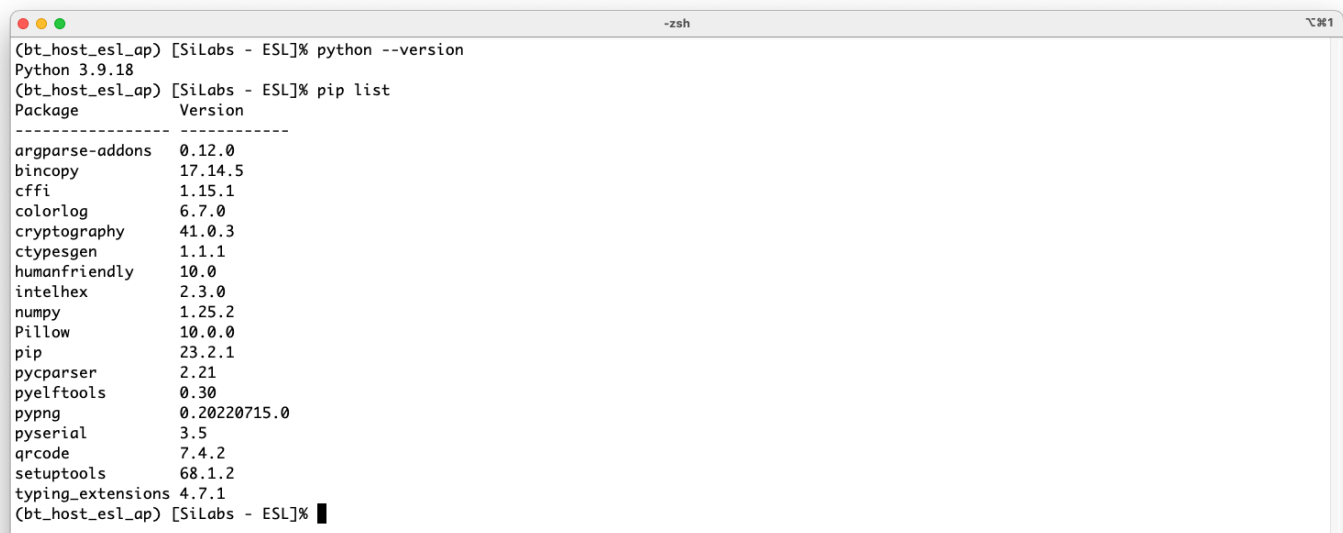
```
python -m pip install -r requirements.txt
```

On Windows, the command is a bit different:

```
py -m pip install -r requirements.txt
```

Note: if Python3 isn't the default Python version, the commands above might require using the format `python3 -m pip install <package>`. For further usage examples, check the pip documentation: [pip.pypa.io/en/stable/](https://pip.pypa.io/en/stable/) [3].

If the pip command is added to the PATH variable, the installed packages can be listed with command `pip list`, or alternatively the installed packages can be listed by using the long command version: `python -m pip list` on Linux/Mac, or `py -m pip list` on Windows.



```
(bt_host_esl_ap) [SiLabs - ESL]% python --version
Python 3.9.18
(bt_host_esl_ap) [SiLabs - ESL]% pip list
Package            Version
-----
argparse-addons    0.12.0
bincopy             17.14.5
cffi                1.15.1
colorlog            6.7.0
cryptography        41.0.3
ctypesgen           1.1.1
humanfriendly       10.0
intelhex            2.3.0
numpy               1.25.2
Pillow              10.0.0
pip                 23.2.1
pyparser            2.21
pyelftools          0.30
pypng               0.20220715.0
pyserial            3.5
qrcode              7.4.2
setuptools          68.1.2
typing_extensions   4.7.1
(bt_host_esl_ap) [SiLabs - ESL]% █
```

Figure 8. Installed Python Packages

#### 5.1.2 Compiling the Access Point Software

After all the necessary Python packages have been installed, the access point host application can be built. Building the C application requires make tool, and a C compiler, e.g., GCC. On Windows, these can be installed using MinGW, and on Linux and Mac, using the package manager.

At the moment, the only supported build environment on Windows is MinGW-64. The makefiles make sure that the gcc is used with a proper prefix (x86\_64-w64-mingw32-). The recommended build environment on Windows is MSYS2.

1. Download and install MSYS2 [4]: [www.msys2.org/](http://www.msys2.org/).
2. Open the Mintty bash. Make sure to start Mingw-w64 64 (mingw64.exe) when launching Mintty. 32-bit versions of MSYS2 will not work.
3. Install additional packages:  

```
pacman -S make mingw-w64-x86_64-gcc mingw-w64-x86_64-python mingw-w64-x86_64-python-pip mingw-w64-x86_64-pkgconf
```
4. Install ctypesgen with pip:  

```
pip install ctypesgen
```

Note: MSYS requires Python and the `ctypesgen` Python package in order to compile the Access Point host software. It is however recommended to run the Access Point by using the native Windows Python environment instead of the Python installation on MSYS2, due to compatibility issues.

To build the project (Mac, Linux, and MSYS2 on Windows):

- Change to the exported project directory:  

```
cd <sdk_path>/app/bluetooth/example_host/bt_host_esl_ap
```
- Build the project using the `make` command.

### 5.1.3 ESL Access Point Hardware

The ESL access point (AP) Python implementation communicates with the ESL tags through a radio interface operating in Network Co-Processor (NCP) mode. Therefore, the EFR device operating as access point has to be programmed with **Bluetooth – NCP ESL Access Point** firmware.

### 5.1.4 Bootloader for ESL AP

The first step is to build a bootloader for the EFR32 device operating as NCP, or AP, device. Select the device from the **Debug Adapters** list, and on the **EXAMPLE PROJECTS & DEMOS** tab, select the **Bootloader – NCP BGAPI UART DFU** example project.

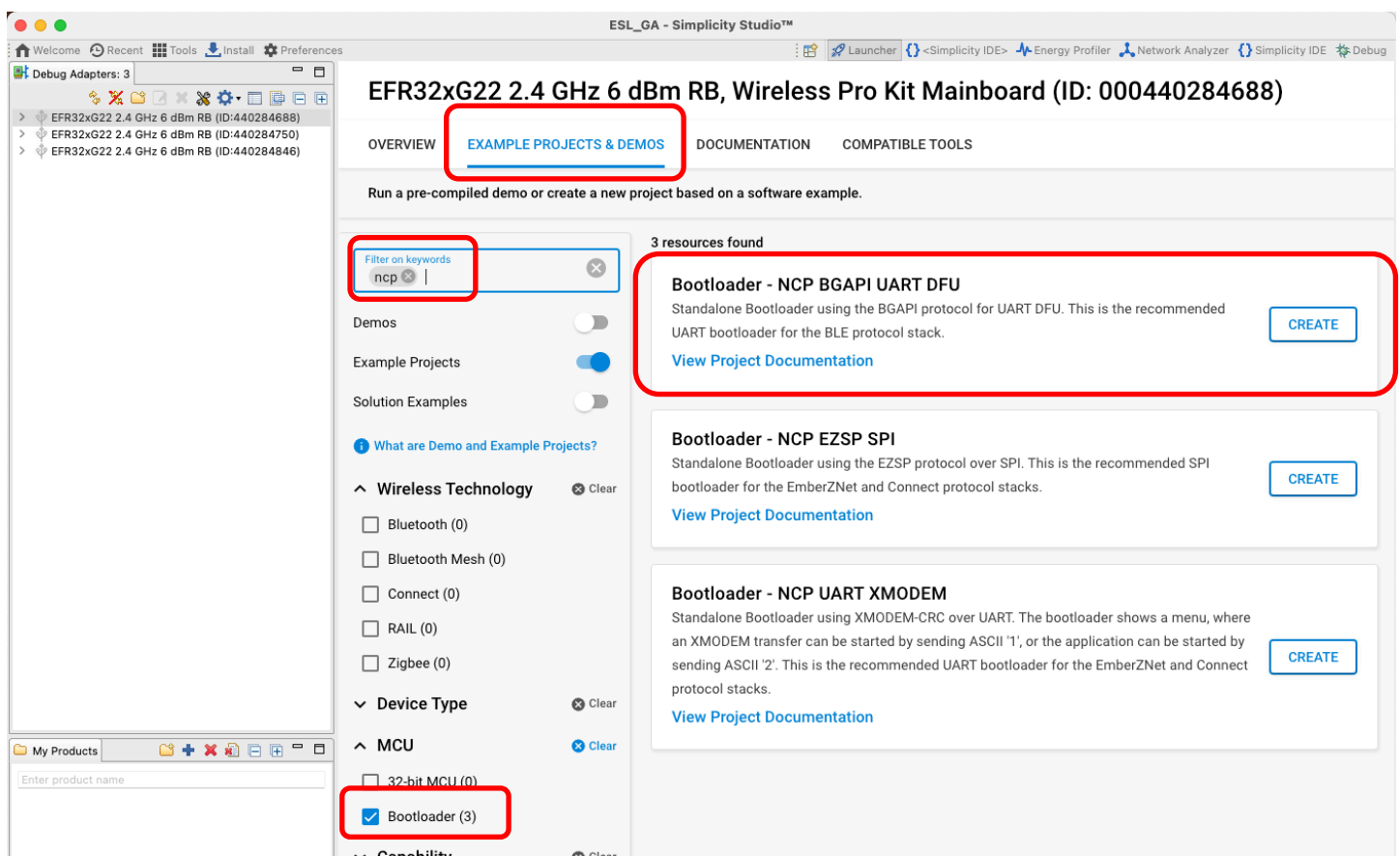


Figure 9. Bootloader – NCP BGAPI UART DFU

Build the **Bootloader - NCP BGAPI UART DFU** project, and program it to the EFR device to be used as an access point. For more information on how to build or program devices using Simplicity Studio v5, see the **Simplicity Studio® 5 User's Guide** on [docs.silabs.com](https://docs.silabs.com).

More information about the bootloaders can be found in document *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher* [5].

### 5.1.5 Access Point Firmware

The next step is to build the actual NCP firmware used by the access point. Select the **EXAMPLE PROJECTS & DEMOS** tab, and from the **Example Projects** list, select the **Bluetooth – NCP ESL Access Point**, and click **CREATE**.

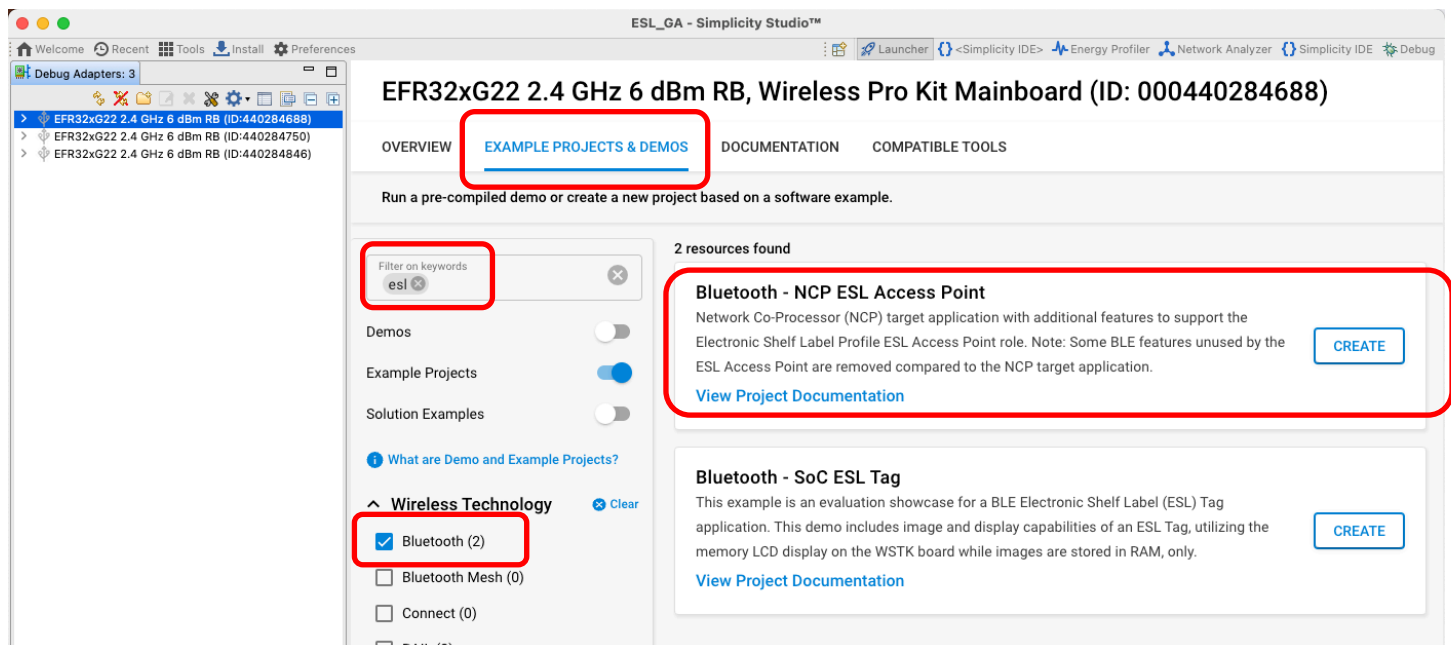


Figure 10. Bluetooth – NCP ESL Access Point

Build the firmware and write the image to the target device to be used as the access point.

Successful access point creation can be verified by executing the AP script (app.py) located in folder <SDK path>/app/bluetooth/example\_host/bt\_host\_esl\_ap/ via command line:

```
Linux:      python app.py <serial device>
            Example: python app.py /dev/ttyACM0

Mac:        python app.py <serial device>
            Example: python app.py /dev/tty.usbmodem0004402847501

Windows:    py -u app.py <serial device>
            Example.: py app.py COM3
```

Note: It is recommended to run the access point script on Windows with option `-u` to have the log streams unbuffered.

```
(bt_host_esl_ap) [SiLabs - ESL]% ./app.py /dev/tty.usbmodem0004402847501
[I] Opened port on posix.
14/Sep 11:24:43.933: LIB - INFO      - [CORE] Resetting NCP target... in esl_lib_init() @ 131:esl_lib_core.c
[I] Start encryption
[I] Communication encrypted
[I] Bluetooth stack booted: v6.1.0-b1357
14/Sep 11:24:44.151: LIB - INFO      - [CORE] Bluetooth type 0 public address: 8C:F6:81:B8:83:66 in esl_lib_core_on_bt_event() @ 221:esl_lib_core.c
14/Sep 11:24:44.209: AP - INFO      - Periodic advertisement started.
14/Sep 11:24:44.223: AP - INFO      - Scanning started.
exit
(bt_host_esl_ap) [SiLabs - ESL]% █
```

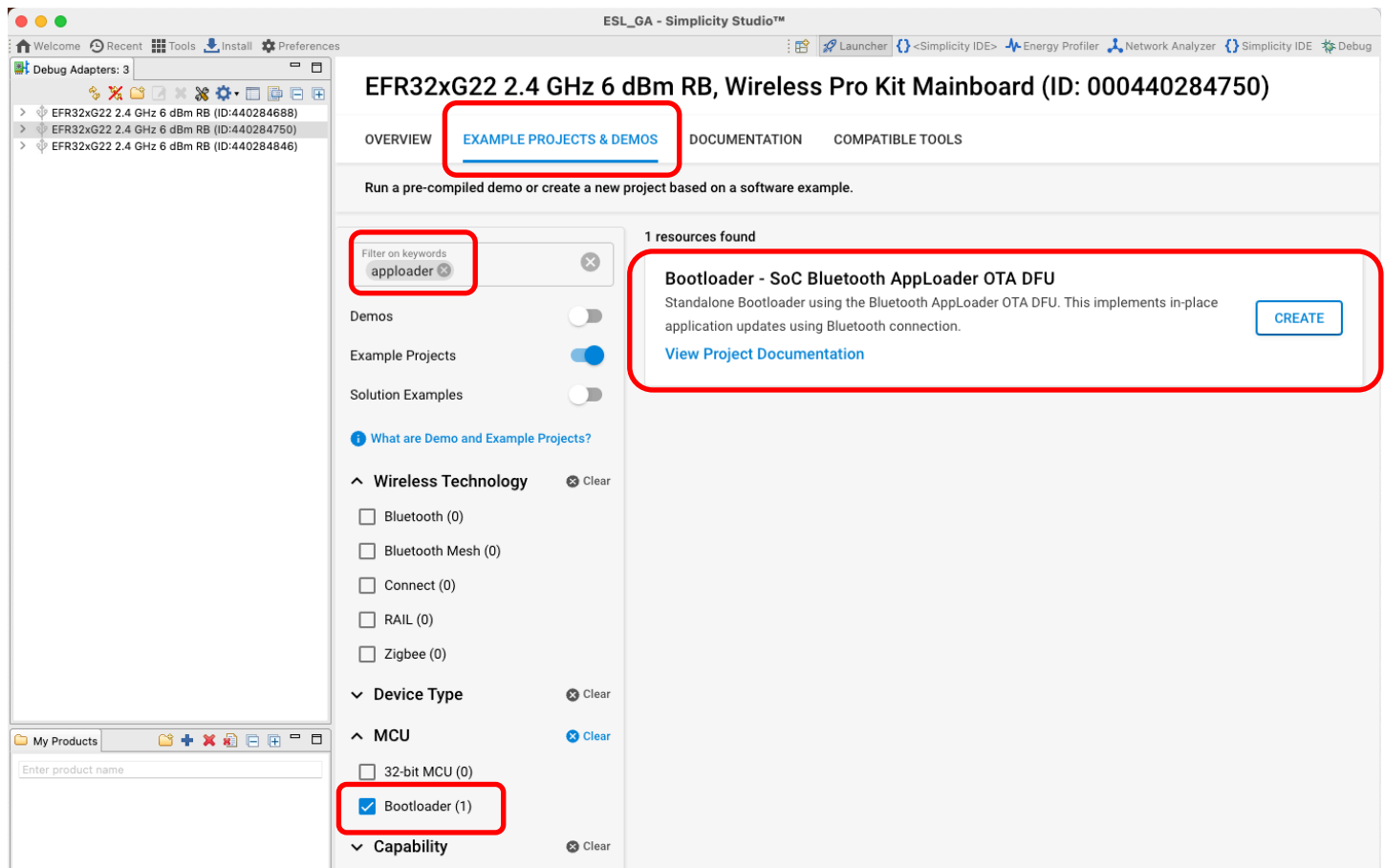
Figure 11. Start the access point with command `python app.py <serial device>`.

## 5.2 ESL Tags

The SDK contains an ESL example project called **Bluetooth – SoC ESL Tag**, which can be easily compiled and programmed to an EFR32 based BLE radio board connected to a Wireless Starter Kit (WSTK) main board. The example project utilizes the WSTK's onboard LCD, LEDs, and sensors and is therefore targeted for the WSTK and a radio board combination. It is however possible to compile the **Bluetooth – SoC ESL Tag** project also to custom hardware, but this will require some additional configuration steps.

### 5.2.1 Bootloader for ESL Tags

Just as with the ESL access point device, the ESL Tags also require a bootloader to work properly. While the ESL AP needed a bootloader targeted for an NCP device, the ESL Tags require an Over-The-Air (OTA) DFU capable bootloader: **Bootloader – SoC Bluetooth AppLoader OTA DFU**. The example project can be created by selecting a correct device from the Debug Adapters list, and filtering the Bootloader example projects with keyword "aploader".



**Figure 12. A bootloader required by the ESL tags can be created from an example project "Bootloader – SoC Bluetooth AppLoader OTA DFU".**

Building the bootloader for the tag and writing the image to the target device follows the same steps as with an access point NCP device.

### 5.2.2 ESL Tag Firmware

To create the example project, first select the device from the Debug Adapters list, and then select the **EXAMPLE PROJECTS & DEMOS** tab. On this view, use the filtering options to show only the Bluetooth Example Projects, and/or use the **Filter on keywords** option to search for the ESL Tag example project, **Bluetooth – SoC ESL Tag**.

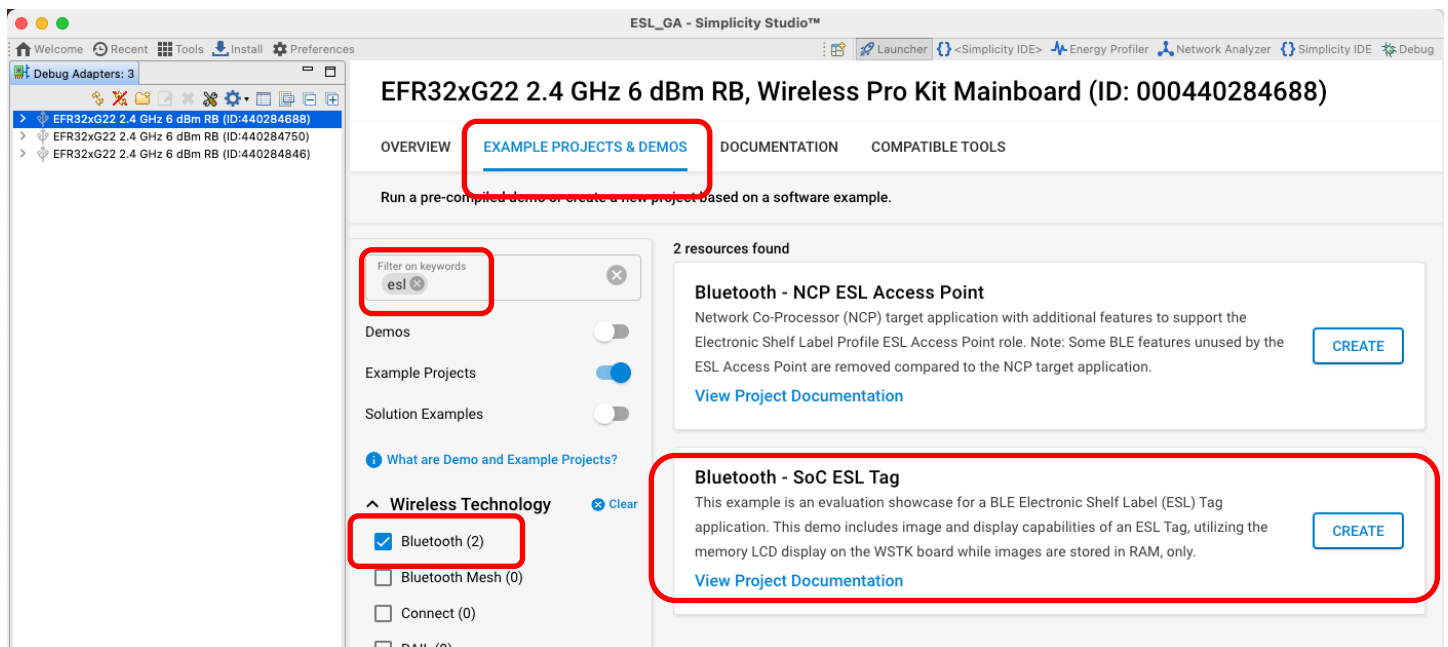


Figure 13. Use the "Filter on keywords" search field to search the "Bluetooth - SoC ESL Tag" example project.

Again, as above, build the project and write the image to the target device(s) to be used as an ESL Tag(s).

After programming the ESL Tag target device(s), the WSTK LCD screen should change, indicating that the device is now running the ESL Tag example project.

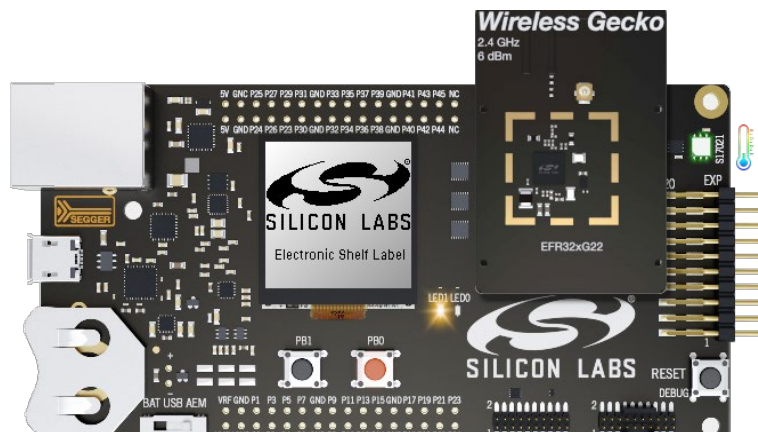


Figure 14. After programming the built "Bluetooth - SoC ESL Tag" project to the EFR32 BLE radio board, the WSTK's LCD screen will indicate the board is now configured to be an ESL tag.

## 6 The ESL Network

The ESL tags on the ESL network can be in one of five different states: Unassociated, Configuring, Synchronized, Updating, and Unsynchronized, as shown in Figure 15. Transitions between states is controlled by the access point, based on the current state of the ESL tag.

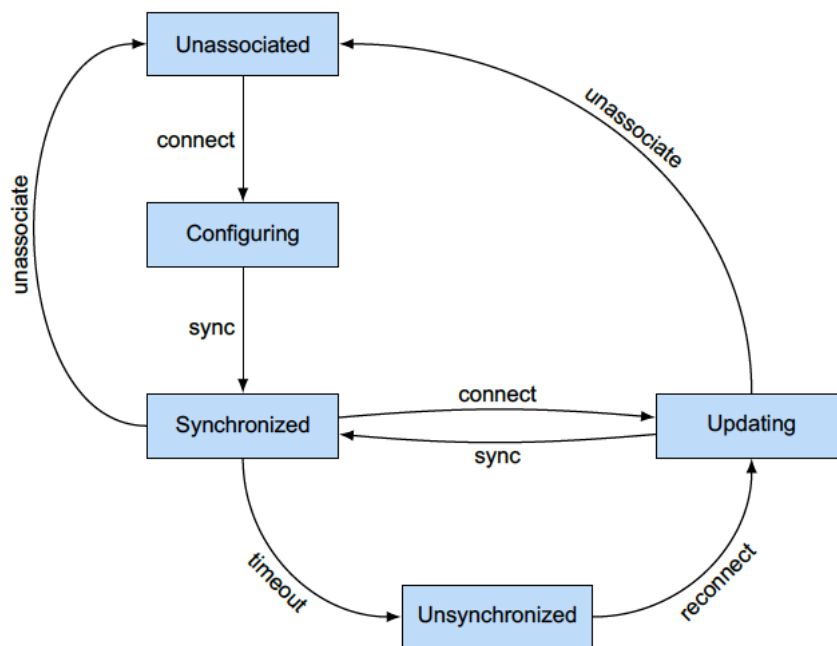


Figure 15. ESL tag state diagram [6]

The ESL access point emulator (<SDK path>/app/bluetooth/example\_host/bt\_host\_esl\_ap/) can be run on two different modes: automatic and manual mode.

### 6.1 ESL AP in Automatic Mode

On automatic mode, the ESL AP starts scanning for ESL tags, and once it finds one, creates a connection to it, reads the GATT database, writes configuration values (ESL ID and group ID, AP sync key, ESL Response key, Nonce, and current time), uploads two random images from the <SDK path>/app/bluetooth/example\_host/bt\_host\_esl\_ap/image/ folder, and then closes the connection. Figure 16 shows the ESL AP state diagram in automatic mode. Once the ESL tag has been configured and the AP has closed the connection, the tag will be in the Synchronized state (see Figure 15). At this point, the ESL tag will also show one of the uploaded, random images on the WSTK LCD screen.

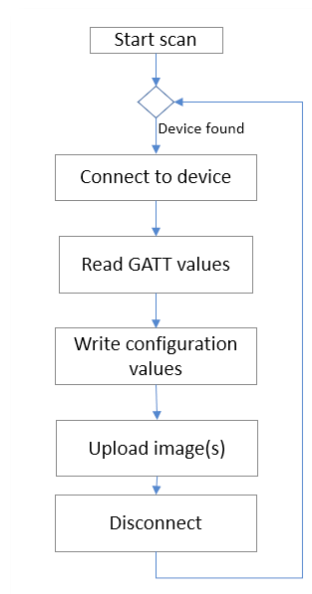


Figure 16. ESL AP Emulator Automatic Mode State Diagram

The access point mode can be switched from automatic to manual and vice versa: `mode manual` will switch the AP to manual mode, and `mode auto` to automatic mode.

Running the AP in automatic mode will verify that both the AP and ESL tag(s) work as expected. If the image on the ESL tag's LCD screen has changed, it has been successfully configured by the AP and is now in the Synchronized state.

The application will be introduced in more details in chapters describing the Manual mode operation.

### 6.1.1 Filter Accept List-based auto connect

Release SiSDK-2024.12 introduced a new addition to the auto mode, called a Filter Accept List-based auto connect. On the previous versions, the access point created a connection to the detected ESL Tags one by one, based on the received advertisements. This created some delays on the onboarding process, since the connection creation process could be initiated only when the previous one has been completed. While the connection initiation process is ongoing, the access point can't make use of the advertisements from other ESL Tags. Using multiple connections mitigates the overall onboarding time.

The new Filter Accept List-based approach brings the connection establishment closer to the BLE controller, as defined by the *Auto Connection Establishment* procedure on the Bluetooth Core Specification. In this approach, the application level adds the ESL Tag device addresses to the Filter Accept List, and the controller autonomously establishes a connection to the devices if the device address matches any of the addresses on the Filter Accept List. This way the BLE stack, and the application layer, continue working normally while the controller is establishing the connections. This speeds up the ESL Tag onboarding process significantly.

The new Filter Accept List-based connection mode is used by default, if the access point is started in auto mode. The connection initiation method can be selected with command `mode auto [{single, list}]`. With the `single` option, the access point will initiate connections to a single ESL Tag at time (as prior to SiSDK-2024.12 release), while the `list` option gives the possibility of utilizing the Filter Accept List and Initiator Filter Policy Core feature.

## 6.2 ESL AP in Manual Mode

To start the AP in manual mode, use option `--cmd`:

```
python app.py --cmd <serial device>
```

The access point can also be started in more verbose mode with switch `-l <level>`, e.g.

```
python app.py /dev/tty.usbmodem0004402847501 -l DEBUG
```

In manual mode, the transitions between tag states are controlled with commands shown in [Figure 17. Commands used to control the ESL tag.](#)

. The commands listed next to the states are the ones you can issue on the specific state.

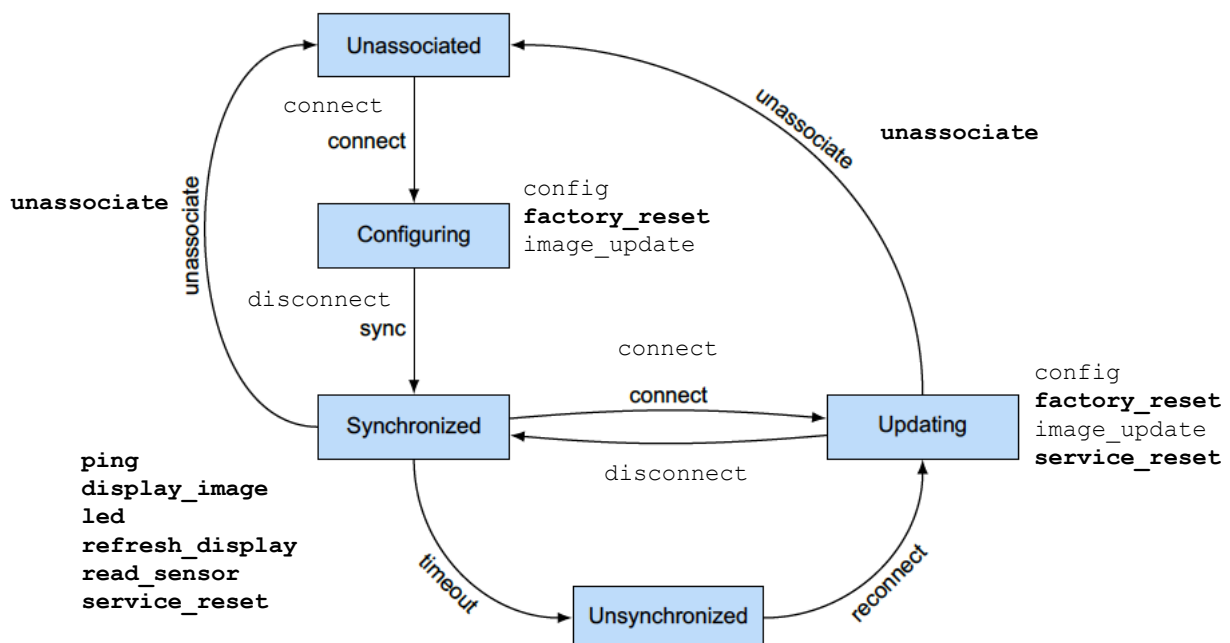


Figure 17. Commands used to control the ESL tag.



**Table 1. ESL Opcodes and Corresponding ESL AP Commands**

Opcode	Procedure	Description	AP Command	Notes
0x00	Ping	Does nothing but solicits a response	ping	—
0x01	Unassociate from AP	Transitions the ESL from the Synchronized state to the Unassociated state	unassociate	—
0x02	Service Reset	Sets the Service Needed flag to False	service_reset	—
0x03	Factory Reset	Requests the ESL to unassociate from the AP and revert to its original state	factory_reset	—
0x04	Update Complete	Requests that the ESL return to the Synchronized state once synchronized	update_complete	Note: Only for testing purposes.
0x10	Read Sensor Data	Requests a response with sensor data, or an indication that data is not yet available	read_sensor	—
0x11	Refresh Display	Refreshes the current displayed image to keep the displayed image fresh on the display	refresh_display	—
0x20	Display Image	Displays a pre-stored image on an ESL display	display_image	—
0x60	Display Timed Image	Displays a prestored image on an ESL display at a specified time	display_image (2)	With options 'time', 'delay', or 'absolute'
0xB0	LED Control	Turns on/off an LED with a color/flashing pattern	led	—
0xF0	LED Timed Control	Turns on/off an LED with a color/flashing pattern at a specified time	led (2)	With options 'time', 'delay' or 'absolute'
0x_F	Vendor-specific Tag	Allows vendors to specify their own commands	N/A	—

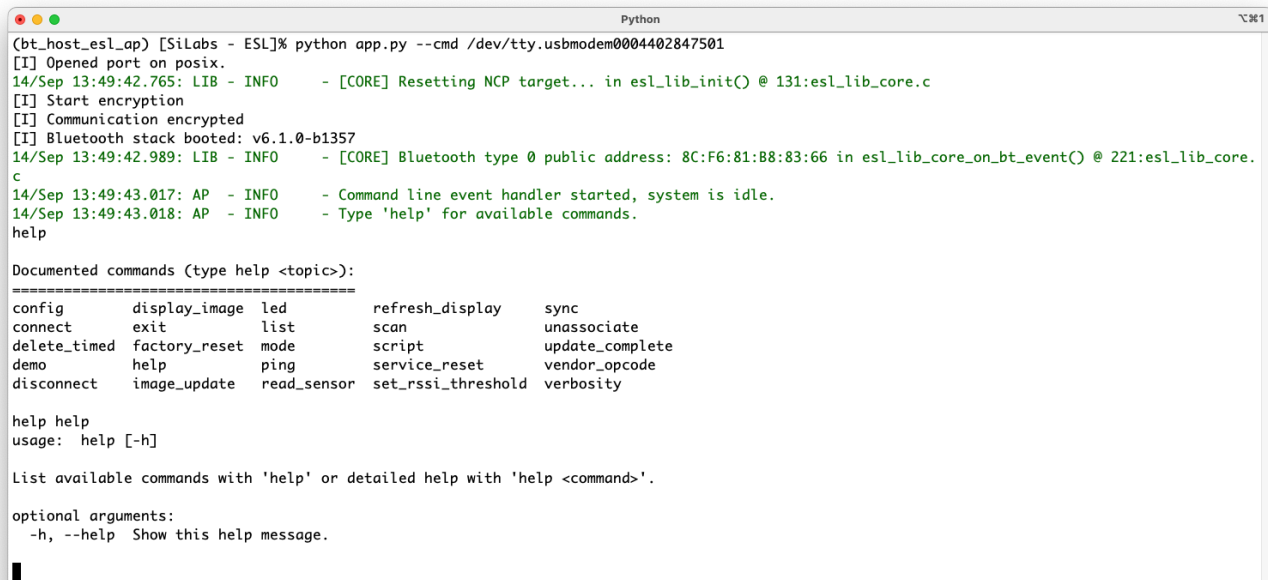
Optional parameters are listed with [ ], while positional paramters are listed without parentheses. A | is used to denote “or” functionality. For example a command `mandatory_param [ opt_param_1 | opt_param_2 ]` has the command itself (`command`), which requires a mandatory parameter (`mandatory_param`). In addition, the command can have optional parameter(s): either `opt_param_1` or `opt_param_2`. The parameters should follow the order as they have been presented on the command help.

In addition to the commands listed in [Figure 17](#), the access point uses a few other commands that are not dependent on the tag state. These are `exit`, `help`, `list`, `mode`, `sync`, `scan`, `script`, and `set_rssi_threshold`.

The `exit` command will terminate the access point application execution.

The `help` command will list all available commands, and `help <topic>` will show the detailed help of that specific command, [Figure 18](#).





```
(bt_host_esl_ap) [SiLabs - ESL]% python app.py --cmd /dev/tty.usbmodem0004402847501
[I] Opened port on posix.
14/Sep 13:49:42.765: LIB - INFO - [CORE] Resetting NCP target... in esl_lib_init() @ 131:esl_lib_core.c
[I] Start encryption
[I] Communication encrypted
[I] Bluetooth stack booted: v6.1.0-b1357
14/Sep 13:49:42.989: LIB - INFO - [CORE] Bluetooth type 0 public address: 8C:F6:81:B8:83:66 in esl_lib_core_on_bt_event() @ 221:esl_lib_core.c
14/Sep 13:49:43.017: AP - INFO - Command line event handler started, system is idle.
14/Sep 13:49:43.018: AP - INFO - Type 'help' for available commands.
help

Documented commands (type help <topic>):
=====
config      display_image  led            refresh_display  sync
connect     exit           list           scan             unassociate
delete_timed factory_reset  mode          script           update_complete
demo        help          ping          service_reset    vendor_opcode
disconnect  image_update  read_sensor   set_rssi_threshold  verbosity

help help
usage: help [-h]

List available commands with 'help' or detailed help with 'help <command>'.

optional arguments:
  -h, --help  Show this help message.
```

**Figure 18.** All available commands can be listed with `help`.

The `list` command can be used to list all the tags the AP can receive. With option `a` (or `advertising`) the AP will list all tags currently advertising (e.g., the ones in Unassociated state), option `s` (or `synchronized`) will list all tags currently on Synchronized state, and option `c` (`connected`) will list all the tags the AP has connected to. In addition to these three most typical states, the tags in the Unsynchronized state can be listed with option `u` (`unsynchronized`). With additional option `-v` (`--verbose`), the AP will list more information of the listed tag, e.g., the AP Sync Key and the ESL Response Key, Display information, etc.

```
list [-h] [--verbose] [--group_id <u7>] state [state ...]
```

As described in Section 6.1 [ESL AP in Automatic Mode](#), the `mode` command can be used to switch between the automatic and manual mode.

```
mode [-h] [{auto,manual}]
```

Command `sync` is used to start, or stop, sending synchronization packets to the ESL tags on network. The optional parameters can be used to set the periodic advertisement interval.

```
sync [-h] [--millis] [--in_max <int>] [--in_min <int>] [--se_count <int>] [--se_interval <int>]
      [--rs_delay <int>] [--rs_spacing <int>] [--rs_count <int>] [{start,stop,config}]
```

See Section 4.1 [Periodic Advertising with Responses](#) for more info about selecting appropriate configuration for your needs.

`scan` command is used to start / stop scanning the ESL tags.

```
scan [-h] [--active] [{start,stop}]
```

The `script` command can be used to record and run scripts containing all possible commands used with the ESL network, making it easy to automate the access point operation.

The `help` command also lists a command `set_rssi_threshold`, which can be used to filter out ESL tags based on their RSSI values.

### 6.2.1 Unassociated State

When an ESL tag is powered, it goes to the Unassociated state, in which the tag will send undirected, connectable advertisement packets. If the ESL AP has been configured to scan (`scan start`), all advertising tags heard by the AP can be listed with command `list a`.

```

Python
help scan
usage: scan [-h] [--active] [{start,stop}]

Start or stop scanning for advertising ESL devices.

positional arguments:
  {start,stop}  Control AP scanning to detect or ignore nearby advertiser ESL devices.

optional arguments:
  -h, --help    Show this help message.
  --active, -a  Start active scan instead of default passive type.

Note: You can obtain the current status of the scanning by omitting the choice.

scan start
14/Sep 13:50:06.238: AP - INFO    - Scanning started.
14/Sep 13:50:06.265: TAG - INFO    - ESL service found at BLE address: 8C:F6:81:B8:82:BE, type 0 with RSSI: -24 dBm
14/Sep 13:50:06.289: TAG - INFO    - ESL service found at BLE address: 8C:F6:81:B8:83:18, type 0 with RSSI: -21 dBm
help list
usage: list [-h] [--verbose] [--group_id <u7>] state [state ...]

List tag information.

positional arguments:
  state

      <advertising, a>:  List advertising tag information
      <blocked, b>:      List blocked (advertising) tag information
      <connected, c>:    List connected tag information
      <synchronized, s>: List synchronized tag information
      <unsynchronized, u>: List unsynchronized tag information

optional arguments:
  -h, --help    Show this help message.
  --verbose, -v List more detailed information
  --group_id <u7>, -g <u7> ESL group ID (optional)

Examples: list a --verbose
          list synchronized -v
Note: To reset the list of advertising and blocked lists you may want to issue a <scan start>
command at any time.

list a
BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
There are 2 advertising tags overall.

list b
There's no blocked tag at all.

list c
There's no connected tag at all.

list s
There's no synchronized tag at all.

list u
There's no unsynchronized tag at all.

```

**Figure 19.** Once the ESL tags have been powered on, they will be in Unassociated state, advertising, as shown by the **list** commands.

For the ESL tags to stay in low power mode and still receive commands from the access point, the access point must send synchronization packages (with optional commands) to the tags. Enabling the synchronization packet transmission is controlled with command `sync start`. The synchronization configuration can be checked, and modified, with parameter `config`.

```

Python
There's no blocked tag at all.
list c
There's no connected tag at all.
list s
There's no synchronized tag at all.
list u
There's no unsynchronized tag at all.
help sync
usage: sync [-h] [--millis] [--in_max <int>] [--in_min <int>] [--se_count <int>] [--se_interval <int>] [--rs_delay <int>] [--rs_spacing <int>]
          [--rs_count <int>]
          [{start,stop,config}]

Start / stop sending synchronization packets.

positional arguments:
  {start,stop,config}    Start/stop sending or config periodic synchronization packets

optional arguments:
  -h, --help            Show this help message.
  --millis, -ms         Specify timing parameters in milliseconds
  --in_max <int>, -max <int> Maximum periodic advertising interval in units of 1.25ms.
  --in_min <int>, -min <int> Minimum periodic advertising interval in units of 1.25ms.
  --se_count <int>, -sc <int> Number of subevents
  --se_interval <int>, -si <int> Subevent interval in units of 1.25ms
  --rs_delay <int>, -rd <int> Response slot delay in units of 1.25ms
  --rs_spacing <int>, -rs <int> Response slot spacing in units of 0.125ms
  --rs_count <int>, -rc <int> Response slot count

Notes: Using the optional '-ms' argument with the 'config' subcommand allows you to specify timing parameters in
milliseconds instead of their natural units, but this may introduce rounding errors. Please also note
that with this option the fractional milliseconds can't be specified precisely.
You can ask for the current status of the PAWR train by omitting the choice.

sync config
14/Sep 13:50:50.243: CLI - INFO    - Current sync parameters:
    Minimum Periodic Interval: 1541.25 ms (1233)
    Maximum Periodic Interval: 1541.25 ms (1233)
    Subevent Count: 28
    Subevent Interval: 55.0 ms (44)
    Response Slot Delay: 37.5 ms (30)
    Response Slot Duration: 0.75 ms (6)
    Response Slot Count: 23

sync start
14/Sep 13:50:54.485: CLI - INFO    - Request Periodic Synchronization Transfer start as follows:
14/Sep 13:50:54.485: CLI - INFO    - Current sync parameters:
    Minimum Periodic Interval: 1541.25 ms (1233)
    Maximum Periodic Interval: 1541.25 ms (1233)
    Subevent Count: 28
    Subevent Interval: 55.0 ms (44)
    Response Slot Delay: 37.5 ms (30)
    Response Slot Duration: 0.75 ms (6)
    Response Slot Count: 23
14/Sep 13:50:54.513: AP - INFO    - Periodic advertisement started.

```

**Figure 20.** Synchronization packet transmission can be enabled with command `sync start`.

## 6.2.2 Configuring State

In the Configuring state, the ESL tag is associated to an ESL network. The Configuring state can be entered only from the Unassociated state by creating a connection to the advertising ESL tag:

```

connect <Bluetooth address>
connect 8C:F6:81:B8:82:BE

```

Once connected, the tag will be listed as connected, but still unassociated to the network.

```

Python
14/Sep 13:50:54.485: CLI - INFO - Current sync parameters:
    Minimum Periodic Interval: 1541.25 ms (1233)
    Maximum Periodic Interval: 1541.25 ms (1233)
    Subevent Count: 28
    Subevent Interval: 55.0 ms (44)
    Response Slot Delay: 37.5 ms (30)
    Response Slot Duration: 0.75 ms (6)
    Response Slot Count: 23
14/Sep 13:50:54.513: AP - INFO - Periodic advertisement started.
help connect
usage: connect [-h] [--group_id <u7>] [--addr_type] [address]

    Connect to one or more ESL devices.

positional arguments:
  address                Bluetooth address (e.g. 'AA:BB:CC:DD:EE:22') in case insensitive format or ESL ID of the tag.

optional arguments:
  -h, --help            Show this help message.
  --group_id <u7>, -g <u7> ESL group ID (optional, default is group 0)
  --addr_type, -t [address_type]: ESL address type (optional), possible values:
                        - public: Public device address (default assumption)
                        - static: Random static device address

Notes: <esl_id> and <group_id> can be used instead of <bt_addr> if ESL is already configured. <address_type>
will be taken into account only if the given <bt_addr> is unknown - otherwise the proper type reported
by the remote device will be used. If the group ID is not given after the ESL ID then the default value
group zero is used. This applies to many commands expecting the group ID as optional parameter.

The 'all' keyword can be used with a special meaning with 'connect' command: it will try to connect to
all advertiser ESLs (within the 'group_id' if it is given or to any advertisers if it isn't) up to the
the maximum number of simultaneous connections supported by the current build of the ESL library and
the attached Network Co-Processor embedded controller.

list a
    BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
    BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
    There are 2 advertising tags overall.

list c
    There's no connected tag at all.

connect 8C:F6:81:B8:82:BE
14/Sep 13:51:18.711: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:82:BE, type 0
14/Sep 13:51:18.711: AP - INFO - Request connecting to: 8C:F6:81:B8:82:BE, type 0 via connectable advertisement
14/Sep 13:51:20.884: TAG - INFO - Reading tag information from address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:51:20.885: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:82:BE, type 0
14/Sep 13:51:21.144: TAG - INFO - Silabs device found - vendor opcodes are not defined

list a
    BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
    There's one advertising tag overall.

list c
    BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
    There's one connected tag overall.

```

**Figure 21. After connection creation, the tag is listed as connected but unassociated.**

Associating a connected ESL tag to the network happens via configuration by the access point. Configuring happens by writing new values to the writable characteristics and reading characteristics related to displays, images, sensors, and LEDs. Configuration is successfully completed after reading and writing the characteristics.

Use command `config` to write all the values at once (parameter `full`), or by using each individual item (parameters `esl_id`, `group_id`, `sync_key`, `response_key`, `nonce`, `time`, `absolute`).

```

config [-h] [--full] [--esl_id <u8>] [--group_id <u7>] [--sync_key] [--response_key] [--time |
    --absolute <u32>] [device]

config --full --esl_id 1

```

It is possible to establish concurrent connections to multiple tags. In such case, the tag has to be addressed with option `[device]`. If there are multiple active connections, the previous `config` command should be given in format

```

config --full --esl_id 1 8C:F6:81:B8:82:BE

```

```

Python
BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
There are 2 advertising tags overall.

list c
    There's no connected tag at all.
connect 8C:F6:81:B8:82:BE
14/Sep 13:51:18.711: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:82:BE, type 0
14/Sep 13:51:18.711: AP - INFO - Request connecting to: 8C:F6:81:B8:82:BE, type 0 via connectable advertisement
14/Sep 13:51:20.884: TAG - INFO - Reading tag information from address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:51:20.885: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:82:BE, type 0
14/Sep 13:51:21.144: TAG - INFO - Silabs device found - vendor opcodes are not defined
list a
    BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
    There's one advertising tag overall.
list c
    BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
    There's one connected tag overall.
help config
usage: config [-h] [--full] [--esl_id <u8>] [--group_id <u7>] [--sync_key] [--response_key] [--time | --absolute <u32>] [device]

    Configure the writable mandatory GATT characteristics of the ESL tag.

positional arguments:
  device                Bluetooth address of the target device (e.g. 'AA:BB:CC:DD:EE:22') in case insensitive format or 'all'

optional arguments:
  -h, --help            Show this help message.
  --full, -f            Configure everything in one step. ESL ID and group can be specified to override default values - see notes.
  --esl_id <u8>, -i <u8> New ESL ID of the connected tag.
  --group_id <u7>, -g <u7> ESL group ID (optional, default is group 0)
  --sync_key, -sk       Set current Access Point Sync Key Material.
  --response_key, -rk   Generate then set new Response Key Material.
  --time, -t            Set current Absolute Time of the ESL Access Point.
  --absolute <u32>, -a <u32> Set custom Absolute Time epoch value - use with care! Mutually exclusive with the 'time' parameter.

Notes: Either the option '--full' or at least one of the other optional parameters shall be given.

    The 'all' keyword can be used to configure a number of connected ESLs, but the ESL ID can't be specified
    in turn, as this would make the command ambiguous.
    However, the same ESL group ID can be specified for multiple connected devices - but use this with care,
    as this command doesn't check against existing ESL configurations, so the network may end up broken!

config --full --esl_id 1
14/Sep 13:51:43.794: AP - INFO - New auto ESL Address: 0x0000
14/Sep 13:51:43.795: AP - INFO - Set ESL ID to 1.
14/Sep 13:51:44.003: TAG - INFO - ESL Tag fully provisioned at address 8C:F6:81:B8:82:BE, type 0
list a
    BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
    There's one advertising tag overall.
list c
    BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
    There's one connected tag overall.

```

Figure 22. Configuring ESL tag

Configuring state is also the state (in addition to Updating state) in which transferring images to the ESL tags is possible.

Transferring images to the connected ESL tag is done with command `image_update`:

```
image_update [-h] [--group_id <u7>] [--label <str>] [--cropfit] [--raw | --display_index <u8>]
             [--cw | --ccw | --flip] image_index imagefile_path  [[address]]
```

```
image_update 0 image/apple.png 1
image_update 1 image/banana.png 1
```

This will add two images to the connected ESL tag: a picture of an apple to image index 0, and a picture of a banana to image index 1.

The **Bluetooth – SoC ESL Tag** example project has been configured to store up to two images. The number of images can be changed via projects Software Components (for more details, see [Section 7.1.2 Image Storage](#)).

```

Python
14/Sep 13:51:43.794: AP - INFO - New auto ESL Address: 0x0000
14/Sep 13:51:43.795: AP - INFO - Set ESL ID to 1.
14/Sep 13:51:44.003: TAG - INFO - ESL Tag fully provisioned at address 8C:F6:81:B8:82:BE, type 0
list a
BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
There's one advertising tag overall.
list c
BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
There's one connected tag overall.
help image_update
usage: image_update [-h] [--group_id <u7>] [--label <str>] [--cropfit] [--raw | --display_index <u8>] [--cw | --ccw | --flip]
                    image_index imagefile_path [[address]]

Update single image on one or more connected Tags.

positional arguments:
  image_index            Image storage index of the ESL tag to be updated.
  imagefile_path         Relative or full path to the selected image file. Use quotation marks if the path contains spaces.
  [address]             Bluetooth address of the target device or ESL ID or 'all' if there are more ESLs connected.

optional arguments:
  -h, --help            Show this help message.
  --group_id <u7>, -g <u7> ESL group ID (optional, default is group 0)
  --label <str>, -l <str> Caption to be written over the image. Use quotation marks if it includes spaces or line breaks.
  --cropfit, -c         Fit the image to the display proportions by cropping.
  --raw, -r            Upload raw image file without any conversion.
  --display_index <u8>, -d <u8> Try auto-conversion image for this display. Mutually exclusive with '--raw' argument.
  --cw, -rr            Clockwise (right) rotation.
  --ccw, -rl           Counter clockwise (left) rotation.
  --flip, -f           Turn the image upside down.

Notes:
  To use space or backslash in the filename or other special characters, such as line break escape
  sequences in the text caption, please enclose these strings in quotes.
  The modifiers like rotation, fitting and labeling are mutually exclusive with raw data input.
  If the group is specified along with the keyword 'all', then only connected devices in the group will be affected.

Examples: image_update 0 ./image/banana.png --label="Line 1\nLine 2"
  Send an image to index 0 on the single connected ESL with two lines of label.

  image_update 1 "/user/home/path with space/img.jpg" all
  Use the 'all' keyword as special address to send the same image to slot 1 on all connected ESLs.

image_update 0 image/apple.png 1
14/Sep 13:52:07.744: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:52:07.757: TAG - INFO - Image file opened: image/apple.png
14/Sep 13:52:07.777: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 0
14/Sep 13:52:08.556: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
image_update 1 image/banana.png 1
14/Sep 13:52:13.093: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:52:13.094: TAG - INFO - Image file opened: image/banana.png
14/Sep 13:52:13.104: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 1
14/Sep 13:52:13.818: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0

```

**Figure 23.** Images can be transferred to ESL tags in Configuring and Updating states with command `image_update`.

Once the ESL tag has been configured, and (optionally) all images are transferred to it, the access point can close the connection and this way change the tag to Synchronized state.

```

disconnect [-h] [--group_id <u7>] [<addr>]
disconnect 1

```



```

--flip, -f          Turn the image upside down.

Notes:      To use space or backslash in the filename or other special characters, such as line break escape
            sequences in the text caption, please enclose these strings in quotes.
            The modifiers like rotation, fitting and and labeling are mutually exclusive with raw data input.
            If the group is specified along with the keyword 'all', then only connected devices in the group will be affected.

Examples: image_update 0 ./image/banana.png --label="Line 1\nLine 2"
            Send an image to index 0 on the single connected ESL with two lines of label.

            image_update 1 "/user/home/path with space/img.jpg" all
            Use the 'all' keyword as special address to send the same image to slot 1 on all connected ESLs.

image_update 0 image/apple.png 1
14/Sep 13:52:07.744: TAG - INFO      - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:52:07.757: TAG - INFO      - Image file opened: image/apple.png
14/Sep 13:52:07.777: AP  - INFO      - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 0
14/Sep 13:52:08.556: TAG - INFO      - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
image_update 1 image/banana.png 1
14/Sep 13:52:13.093: TAG - INFO      - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:52:13.094: TAG - INFO      - Image file opened: image/banana.png
14/Sep 13:52:13.104: AP  - INFO      - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 1
14/Sep 13:52:13.818: TAG - INFO      - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
help disconnect
usage: disconnect [-h] [--group_id <u7>] [<addr>]

            Initiate the Periodic Advertisement Sync Transfer process if PAWR train is
            available then disconnect from an ESL device with the specified address.

positional arguments:
  <addr>              Bluetooth address (e.g. 'AA:BB:CC:DD:EE:22') in case insensitive format or ESL ID of the tag.

optional arguments:
  -h, --help          Show this help message.
  --group_id <u7>, -g <u7>  ESL group ID (optional, default is group 0)

Notes: If no address is specified, the default active connection is closed - if only one exists.

To close more existing connections at once, you can use the 'disconnect all' command.

disconnect 1
14/Sep 13:52:30.332: TAG - INFO      - Command: 0401 written successfully for 8C:F6:81:B8:82:BE, type 0
14/Sep 13:52:34.129: TAG - INFO      - Connection to 8C:F6:81:B8:82:BE, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
list a
BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
There's one advertising tag overall.
list c
There's no connected tag at all.
list s
BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
There's one synchronized tag overall.

```

**Figure 24. Closing connection to the ESL tag will switch it to Synchronized state.**

Now the `connect`, `config`, `image_update`, and `disconnect` steps should be repeated for rest of the ESL tags to be associated to the network, [Figure 25](#) - [Figure 27](#).

First, connection creation to the second tag.

```
connect 8C:F6:81:B8:83:18
```

After connection creation, the second tag is still unassociated to the ESL network. Therefore, the next step is to associate it to the network by configuring it.

```
config --full --esl_id 2
```

```

Python
image_update 1 "/user/home/path with space/img.jpg" all
Use the 'all' keyword as special address to send the same image to slot 1 on all connected ESLs.

image_update 0 image/apple.png 1
14/Sep 13:52:07.744: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:52:07.757: TAG - INFO - Image file opened: image/apple.png
14/Sep 13:52:07.777: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 0
14/Sep 13:52:08.556: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
image_update 1 image/banana.png 1
14/Sep 13:52:13.093: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:52:13.094: TAG - INFO - Image file opened: image/banana.png
14/Sep 13:52:13.104: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 1
14/Sep 13:52:13.818: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
help disconnect
usage: disconnect [-h] [--group_id <u7>] [<addr>]

Initiate the Periodic Advertisement Sync Transfer process if PAWR train is
available then disconnect from an ESL device with the specified address.

positional arguments:
  <addr> Bluetooth address (e.g. 'AA:BB:CC:DD:EE:22') in case insensitive format or ESL ID of the tag.

optional arguments:
  -h, --help Show this help message.
  --group_id <u7>, -g <u7> ESL group ID (optional, default is group 0)

Notes: If no address is specified, the default active connection is closed - if only one exists.

To close more existing connections at once, you can use the 'disconnect all' command.

disconnect 1
14/Sep 13:52:30.332: TAG - INFO - Command: 0401 written successfully for 8C:F6:81:B8:82:BE, type 0
14/Sep 13:52:34.129: TAG - INFO - Connection to 8C:F6:81:B8:82:BE, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
list a
BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
There's one advertising tag overall.
list c
There's no connected tag at all.
list s
BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
There's one synchronized tag overall.
connect 8C:F6:81:B8:83:18
14/Sep 13:52:53.230: AP - INFO - Request connecting to: 8C:F6:81:B8:83:18, type 0 via connectable advertisement
14/Sep 13:52:54.689: TAG - INFO - Reading tag information from address 8C:F6:81:B8:83:18, type 0
14/Sep 13:52:54.690: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:83:18, type 0
14/Sep 13:52:54.922: TAG - INFO - Silabs device found - vendor opcodes are not defined
config --full --esl_id 2
14/Sep 13:53:00.701: AP - INFO - New auto ESL Address: 0x0001
14/Sep 13:53:00.701: AP - INFO - Set ESL ID to 2.
14/Sep 13:53:00.868: TAG - INFO - ESL Tag fully provisioned at address 8C:F6:81:B8:83:18, type 0

```

**Figure 25. Creating Connection to an ELS Tag**

While the second ESL tag is still in the Configuring state, upload images to it.

```

image_update 0 image/bread.png 2
image_update 1 image/chicken.png 2

```



```

Python
image_update 1 image/banana.png 1
14/Sep 13:52:13.093: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:52:13.094: TAG - INFO - Image file opened: image/banana.png
14/Sep 13:52:13.104: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 1
14/Sep 13:52:13.818: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
help disconnect
usage: disconnect [-h] [--group_id <u7>] [<addr>]

Initiate the Periodic Advertisement Sync Transfer process if PAWR train is
available then disconnect from an ESL device with the specified address.

positional arguments:
  <addr> Bluetooth address (e.g. 'AA:BB:CC:DD:EE:22') in case insensitive format or ESL ID of the tag.

optional arguments:
  -h, --help Show this help message.
  --group_id <u7>, -g <u7> ESL group ID (optional, default is group 0)

Notes: If no address is specified, the default active connection is closed - if only one exists.

To close more existing connections at once, you can use the 'disconnect all' command.

disconnect 1
14/Sep 13:52:30.332: TAG - INFO - Command: 0401 written successfully for 8C:F6:81:B8:82:BE, type 0
14/Sep 13:52:34.129: TAG - INFO - Connection to 8C:F6:81:B8:82:BE, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
list a
BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
There's one advertising tag overall.
list c
There's no connected tag at all.
list s
BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
There's one synchronized tag overall.
connect 8C:F6:81:B8:83:18
14/Sep 13:52:53.230: AP - INFO - Request connecting to: 8C:F6:81:B8:83:18, type 0 via connectable advertisement
14/Sep 13:52:54.689: TAG - INFO - Reading tag information from address 8C:F6:81:B8:83:18, type 0
14/Sep 13:52:54.690: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:83:18, type 0
14/Sep 13:52:54.922: TAG - INFO - Silabs device found - vendor opcodes are not defined
config --full --esl_id 2
14/Sep 13:53:00.701: AP - INFO - New auto ESL Address: 0x0001
14/Sep 13:53:00.701: AP - INFO - Set ESL ID to 2.
14/Sep 13:53:00.868: TAG - INFO - ESL Tag fully provisioned at address 8C:F6:81:B8:83:18, type 0
image_update 0 image/bread.png 2
14/Sep 13:53:15.238: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 13:53:15.239: TAG - INFO - Image file opened: image/bread.png
14/Sep 13:53:15.248: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 0
14/Sep 13:53:15.853: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0
image_update 1 image/chicken.png 2
14/Sep 13:53:21.336: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 13:53:21.337: TAG - INFO - Image file opened: image/chicken.png
14/Sep 13:53:21.347: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 1
14/Sep 13:53:22.000: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0

```

**Figure 26.** `image_update` command will transfer images to connected ESL tag.

Once the ESL tag has been associated to the network by configuring it, and it now has all necessary images to be shown on the onboard display, the connection to it can be closed and the tag will switch to the Synchronized state.

```
disconnect 2
```

```

Python
positional arguments:
  <addr>                Bluetooth address (e.g. 'AA:BB:CC:DD:EE:22') in case insensitive format or ESL ID of the tag.

optional arguments:
  -h, --help            Show this help message.
  --group_id <u7>, -g <u7>  ESL group ID (optional, default is group 0)

Notes: If no address is specified, the default active connection is closed - if only one exists.

To close more existing connections at once, you can use the 'disconnect all' command.

disconnect 1
14/Sep 13:52:30.332: TAG - INFO - Command: 0401 written successfully for 8C:F6:81:B8:82:BE, type 0
14/Sep 13:52:34.129: TAG - INFO - Connection to 8C:F6:81:B8:82:BE, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
list a
  BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
  There's one advertising tag overall.
list c
  There's no connected tag at all.
list s
  BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
  There's one synchronized tag overall.
connect 8C:F6:81:B8:83:18
14/Sep 13:52:53.230: AP - INFO - Request connecting to: 8C:F6:81:B8:83:18, type 0 via connectable advertisement
14/Sep 13:52:54.689: TAG - INFO - Reading tag information from address 8C:F6:81:B8:83:18, type 0
14/Sep 13:52:54.690: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:83:18, type 0
14/Sep 13:52:54.922: TAG - INFO - Silabs device found - vendor opcodes are not defined
config --full --esl_id 2
14/Sep 13:53:00.701: AP - INFO - New auto ESL Address: 0x0001
14/Sep 13:53:00.701: AP - INFO - Set ESL ID to 2.
14/Sep 13:53:00.868: TAG - INFO - ESL Tag fully provisioned at address 8C:F6:81:B8:83:18, type 0
image_update 0 image/bread.png 2
14/Sep 13:53:15.238: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 13:53:15.239: TAG - INFO - Image file opened: image/bread.png
14/Sep 13:53:15.248: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 0
14/Sep 13:53:15.853: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0
image_update 1 image/chicken.png 2
14/Sep 13:53:21.336: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 13:53:21.337: TAG - INFO - Image file opened: image/chicken.png
14/Sep 13:53:21.347: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 1
14/Sep 13:53:22.000: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0
disconnect 2
14/Sep 13:53:30.086: TAG - INFO - Command: 0402 written successfully for 8C:F6:81:B8:83:18, type 0
14/Sep 13:53:33.468: TAG - INFO - Connection to 8C:F6:81:B8:83:18, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
list a
  There's no advertising tag at all.
list c
  There's no connected tag at all.
list s
  BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
  BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
  There are 2 synchronized tags overall.

```

**Figure 27.** After configuring the second ESL tag (uploading images) and disconnecting from it, also the second ESL tag will transition to Synchronized state. `list` command with `verbose (v)` option will give more information about the tag, also shown after closing the connection.

## 6.2.3 Synchronized State

In Synchronized state, the ESL tag is in low-power, low-bandwidth communications mode, receiving the synchronization packages (periodic advertisements) from the AP. The communication model is based on PAWR, which enables synchronization and sending small amount of data from the AP to thousands of ESL tags, while also providing a possibility for the tags to respond to the received messages, if directly addressed.

The Synchronized state is considered as the "main" state, in which the tag will spend most of its time. In this state, an access point can request the tag to display an image on the display(s), control the LEDs, and read sensor data from the tag. In addition, transition to the Updating state is controlled by the access point while the tag is in the Synchronized state.

In the Synchronized state, both the AP and ESL Tag must use the Encrypted Advertising Data feature described in chapter 4.2 Encrypted Advertising Data (EAD). For further information, please see the Electronic Shelf Label Profile v1.0 [7].

### Displaying Images

In the Synchronized state, the access point can send control commands to the ESL tag, e.g., to display an image:

```
display_image [-h] [--group_id <u7>] [--time <hh:mm:ss> | --absolute <u32>] [--delay <u32>]
              [--date <YYYY-MM-DD>] esl_id image_index display_index
```

#### Command

```
display_image 1 0 0
```

will display an apple on the WSTK LCD screen (screen index 0) on ESL tag 1, and command

```
display_image --delay 3000 2 0 0
```

will change the image on the WSTK LCD to bread on ESL tag 2 after 3000 ms.

The `display_image` command can also be used to display image on all ESL tags belonging to some group:

```
display_image all 1 0 0
```

Now all ESL tags in group 0 (default) will display image from index 1, on WSTK LCD (display index 0). Thus, the ESL tag 1 will display a picture of a banana, and ESL tag 2 will display a picture of a chicken.

```

Python
14/Sep 13:53:15.239: TAG - INFO - Image file opened: image/bread.png
14/Sep 13:53:15.248: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 0
14/Sep 13:53:15.853: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0
image_update 1 image/chicken.png 2
14/Sep 13:53:21.336: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 13:53:21.337: TAG - INFO - Image file opened: image/chicken.png
14/Sep 13:53:21.347: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 1
14/Sep 13:53:22.000: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0
disconnect 2
14/Sep 13:53:30.086: TAG - INFO - Command: 0402 written successfully for 8C:F6:81:B8:83:18, type 0
14/Sep 13:53:33.468: TAG - INFO - Connection to 8C:F6:81:B8:83:18, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
list a
    There's no advertising tag at all.
list c
    There's no connected tag at all.
list s
    BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
    BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
    There are 2 synchronized tags overall.
help display_image
usage: display_image [-h] [--group_id <u7>] [--time <hh:mm:ss> | --absolute <u32>] [--delay <u32>] [--date <YYYY-MM-DD>]
                    esl_id image_index display_index

    Display tag image.

positional arguments:
  esl_id                ESL ID or all
  image_index            Image index to update.
  display_index          Display index

optional arguments:
  -h, --help            Show this help message.
  --group_id <u7>, -g <u7>    ESL group ID (optional, default is group 0)
  --time <hh:mm:ss>, -t <hh:mm:ss>    Execution time of the command in hour:min:sec format. (optional) Note: If --delay is specified then
                                     it is also added to the calculated value as an additional delay.
  --absolute <u32>, -a <u32>    Execution time of the command in ESL Absolute Time epoch value. Mutually exclusive with timed delay.
  --delay <u32>, -d <u32>    Delay in milliseconds (optional)
  --date <YYYY-MM-DD>, -d <YYYY-MM-DD>    Execution date of the command in ISO-8601 format (optional to time, only).

    Note: Timed display commands with a delay shorter than the actual periodic advertisement interval may be
    rejected on receive by Implausible Absolute Time (0x0C) ESL error response.

display_image 1 0 0
14/Sep 13:54:02.587: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:54:02.588: RSP - INFO - Display State Response received
    Display state: Display_Index: 0 Image_Index: 0 (0x110000)
display_image --delay 3000 2 0 0
14/Sep 13:54:07.629: AP - INFO - Delayed display image command issued at absolute time 267874
14/Sep 13:54:08.749: AP - INFO - Reply in slot 0 from ESL ID 2 in group 0.
14/Sep 13:54:08.749: RSP - INFO - Display State Response received
    Display state: Display_Index: 0 Image_Index: 0 (0x110000)
display_image --group_id 0 all 1 0
14/Sep 13:54:14.402: AP - WARNING - All tags in group 0 will be addressed!

```

**Figure 28. Access point can command tags in synchronized state via Periodic Advertising with Responses (PAwR), e.g. to display an image on the connected display(s).**

The Synchronized state also supports refreshing the (current image on) display:

```
refresh_display [-h] [--group_id <u7>] esl_id display_index
```

## LEDs

The Synchronized state can also be used to control the LED(s) on the ESL tag:

```

led [-h] [--group_id <u7>] [--default] [--pattern <bits>] [--on_period <int[0,3]>]
    [--off_period <int[0,3]>] [--brightness <int[0,3]>] [--color <int[0,3]>] [--repeats <u15>]
    | --duration <u15>] [--index <u8>] [--time <hh:mm:ss> | --absolute <u32>]
    [--date YYYY-MM-DD] [--delay <u32>] {on,off,flash} esl_id

```

For example, command

```
led --index 1 on 1
```

will turn on the LED1 on the WSTK used as ESL tag 1.

As shown by the `help led` output, the LED can be programmed to flash different patterns in addition to just turning it ON/OFF. For example, command

```
led --pattern 10101000111011101110001010100000 --on_period 75 --off_period 75 --brightness 3
--repeats 3 --index 0 flash 2
```

will flash the LED1 as configured with parameter `pattern`.

Note: The **Bluetooth - SoC ESL Tag** example project has only one LED configured (index 0), which controls the LED1 on the WSTK.

```

display_image --delay 3000 2 0 0
14/Sep 13:54:07.629: AP - INFO - Delayed display image command issued at absolute time 267874
14/Sep 13:54:08.749: AP - INFO - Reply in slot 0 from ESL ID 2 in group 0.
14/Sep 13:54:08.749: RSP - INFO - Display State Response received
Display state: Display_Index: 0 Image_Index: 0 (0x10000)
display_image --group_id 0 all 1 0
14/Sep 13:54:14.402: AP - WARNING - All tags in group 0 will be addressed!
help led
usage: led [-h] [--group_id <u7>] [--default] [--pattern <bits>] [--on_period <int[0,3]>] [--off_period <int[0,3]>] [--brightness <int[0,3]>]
        [--color <int[0,3]>] [--repeats <u15> | --duration <u15>] [--index <u8>] [--time <hh:mm:ss> | --absolute <u32>] [--date YYYY-MM-DD]
        [--delay <u32>]
        {on,off,flash} esl_id

Turn on / off or flash an LED utilizing the LED control command.

positional arguments:
  {on,off,flash}        Turn LED on/off or flash the LED
  esl_id                ESL ID or all

optional arguments:
  -h, --help            Show this help message.
  --group_id <u7>, -g <u7>    ESL group ID (optional, default is group 0)
  --default, -d          Restore the default flashing pattern built-in with AP
  --pattern <bits>, -p <bits>    A string containing either '1's or '0's, max length: 40
  --on_period <int[0,3]>, -on <int[0,3]>    Integer value from 1 to 255, meaning 'delay *2ms' for on state bits of the pattern. '0' is prohibited
  --off_period <int[0,3]>, -of <int[0,3]>    Integer value from 1 to 255, meaning 'delay *2ms' for off state bits of the pattern. '0' is prohibited
  --brightness <int[0,3]>, -b <int[0,3]>    Integer value, 4 step brightness from 0 to 3
  --color <int[0,3]>, -c <int[0,3]>        Red, green and blue values - only applies to LED with sRGB type
  --repeats <u15>, -r <u15>            How many times the pattern shall be repeated. Mutually exclusive with 'duration=' param. Value set [1-32767]
  --duration <u15>, -dn <u15>        How many seconds the pattern shall be repeated. Mutually exclusive with 'repeats=' param. Value set [1-32767]
  --index <u8>, -i <u8>                Index of the LED (optional, 0 by default)
  --time <hh:mm:ss>, -t <hh:mm:ss>    Execution time of the command in hour:min:sec format. Note: If <delay_ms> is specified then it is also added to the calculated value as an additional delay
  --absolute <u32>, -a <u32>          Execution time of the command in ESL Absolute Time epoch value. Mutually exclusive with timed delay.
  --date YYYY-MM-DD, -dt YYYY-MM-DD    Execution date of the command in ISO-8601 format (optional to time, only).
  --delay <u32>, -dy <u32>            Delay in milliseconds

Notes: Almost all of the optional led control parameters are "sticky", meaning that the last values are preserved by the AP internally and will be re-used next time, if the given parameter is omitted in the argument list. This doesn't apply on the delay, time and absolute parameters, though.

led --index 0 on 1
14/Sep 13:54:30.328: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:54:30.328: RSP - INFO - Led State Response received
LED State: LED_Index: 0 (0x0100)
led --pattern 10101000111011101110001010100000 --on_period 75 --off_period 75 --brightness 3 --repeats 3 --index 0 flash 2
14/Sep 13:54:39.575: AP - INFO - Reply in slot 0 from ESL ID 2 in group 0.
14/Sep 13:54:39.575: RSP - INFO - Led State Response received
LED State: LED_Index: 0 (0x0100)

```

Figure 29. An ESL tag can have multiple LEDs, and the LEDs support multiple configurations, flashing patterns, etc.

## Sensors

List of sensors supported by an ESL tag can be checked with command `list --verbose s:`

```

argument list. This doesn't apply on the delay, time and absolute parameters, though.

led --index 0 on 1
14/Sep 13:54:30.328: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:54:30.328: RSP - INFO - Led State Response received
LED State: LED_Index: 0 (0x0100)
led --pattern 10101000111011101110001010100000 --on_period 75 --off_period 75 --brightness 3 --repeats 3 --index 0 flash 2
14/Sep 13:54:39.575: AP - INFO - Reply in slot 0 from ESL ID 2 in group 0.
14/Sep 13:54:39.575: RSP - INFO - Led State Response received
LED State: LED_Index: 0 (0x0100)
list --verbose s
BLE Address: 8C:F6:81:B8:82:BE, type 0
ESL Address: 1 (0x0001)
ESL ID: 1
Group ID: 0
AP sync key: **** (Set)
Response key: **** (Set)
Absolute Time: Last time set to 121039
Display: [0] width: 128 height: 128 type: 1
Max image index: 1
Sensors:
[0] Present input voltage
[1] Present device operating temperature
[2] Device firmware version
[3] Date of manufacture
[4] Device operating temperature range specification
[5] Silabs readout counter
[6] Present ambient temperature
[7] Silabs button
LED Info: [0] Colored, current color: RGB(023)
PnP Info: Vendor: Silicon Labs Product_ID: 0x351 Product_version: 0xa00
Last status: Synchronized (0x0002) received at 14/Sep 13:54:30.328
-----
BLE Address: 8C:F6:81:B8:83:18, type 0
ESL Address: 2 (0x0002)
ESL ID: 2
Group ID: 0
AP sync key: **** (Set)
Response key: **** (Set)
Absolute Time: Last time set to 197946
Display: [0] width: 128 height: 128 type: 1
Max image index: 1
Sensors:
[0] Present input voltage
[1] Present device operating temperature
[2] Device firmware version
[3] Date of manufacture
[4] Device operating temperature range specification
[5] Silabs readout counter
[6] Present ambient temperature
[7] Silabs button
LED Info: [0] Colored, current color: RGB(023)
PnP Info: Vendor: Silicon Labs Product_ID: 0x351 Product_version: 0xa00
Last status: Synchronized (0x0002) received at 14/Sep 13:54:39.575
-----
There are 2 synchronized tags overall.

```

**Figure 30. Sensors supported by the ESL tag can be listed with command `list --verbose s`.**

Reading data from the sensors is done with command `read_sensor`, with parameters ESL ID and the sensor index:

```
read_sensor [-h] [--group_id <u7>] esl_id sensor_index
```

The sensors are listed with indexes on descending order with `list --verbose s`, thus reading the current input voltage of an ESL tag 1 is achieved with command `read_sensor 1 0`.



```

Python
Display: [0] width: 128 height: 128 type: 1
Max image index: 1
Sensors: [0] Present input voltage
          [1] Present device operating temperature
          [2] Device firmware version
          [3] Date of manufacture
          [4] Device operating temperature range specification
          [5] Silabs readout counter
          [6] Present ambient temperature
          [7] Silabs button
LED Info: [0] Colored, current color: RGB(023)
PnP Info: Vendor: Silicon Labs Product_ID: 0x351 Product_version: 0xa00
Last status: Synchronized (0x0002) received at 14/Sep 13:54:39.575
-----
There are 2 synchronized tags overall.
help read_sensor
usage: read_sensor [-h] [--group_id <u7>] esl_id sensor_index

Read sensor information.

positional arguments:
  esl_id          ESL ID
  sensor_index    Sensor index.

optional arguments:
  -h, --help            Show this help message.
  --group_id <u7>, -g <u7>  ESL group ID (optional, default is group 0)

read_sensor 1 0
14/Sep 13:55:24.275: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:55:24.276: SEN - INFO - Sensor data received: 0x2e00d200
Sensor type: Present input voltage
3.28125

read_sensor 1 1
14/Sep 13:55:28.895: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:55:28.896: SEN - INFO - Sensor data received: 0x2e01260c
Sensor type: Present device operating temperature
31.1

read_sensor 1 2
14/Sep 13:55:33.527: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:55:33.527: SEN - INFO - Sensor data received: 0x8e0230312e30302e3030
Sensor type: Device firmware version
01.00.00

read_sensor 1 3
14/Sep 13:55:38.147: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:55:38.148: SEN - INFO - Sensor data received: 0x3e03db4a00
Sensor type: Date of manufacture
20 June, 2022

read_sensor 1 4
14/Sep 13:55:42.766: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:55:42.766: SEN - INFO - Sensor data received: 0x4e040000581b
Sensor type: Device operating temperature range specification
[0.0, 70.0]

```

**Figure 31.** Different sensors can be read with command `read_sensor <esl_id> <sensor_index> [g=<group_id>]`. Since the tags (to be read) are in Synchronized state, the read command will be sent via periodic advertisement, and the sensor value will be part of the response.

## 6.2.4 Updating State

To transfer larger data amounts to the ESL tag (e.g. new images), the ESL tag has to transition from the Synchronized state to the Updating state. This occurs by creating a connection to the tag.

```
connect 2
```

Once the connection is created, the access point can transfer new images to the ESL tags currently in the Updating state:

```
image_update 0 image/croissant.png 2
image_update 1 image/donut.png 2
```

```

sensor_index      Sensor index.

optional arguments:
-h, --help          Show this help message.
--group_id <u7>, -g <u7>  ESL group ID (optional, default is group 0)

read_sensor 1 0
14/Sep 13:55:24.275: AP - INFO      - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:55:24.276: SEN - INFO      - Sensor data received: 0x2e00d200
      Sensor type: Present input voltage
      3.28125

read_sensor 1 1
14/Sep 13:55:28.895: AP - INFO      - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:55:28.896: SEN - INFO      - Sensor data received: 0x2e01260c
      Sensor type: Present device operating temperature
      31.1

read_sensor 1 2
14/Sep 13:55:33.527: AP - INFO      - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:55:33.527: SEN - INFO      - Sensor data received: 0x8e0230312e30302e3030
      Sensor type: Device firmware version
      01.00.00

read_sensor 1 3
14/Sep 13:55:38.147: AP - INFO      - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:55:38.148: SEN - INFO      - Sensor data received: 0x3e03db4a00
      Sensor type: Date of manufacture
      20 June, 2022

read_sensor 1 4
14/Sep 13:55:42.766: AP - INFO      - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:55:42.766: SEN - INFO      - Sensor data received: 0x4e040000581b
      Sensor type: Device operating temperature range specification
      [0.0, 70.0]

connect 2
14/Sep 13:55:53.112: AP - INFO      - Bonding LTK found for ESL at 8C:F6:81:B8:83:18, type 0
14/Sep 13:55:53.113: AP - INFO      - Request connecting to: 8C:F6:81:B8:83:18, type 0 over PAwR
14/Sep 13:55:55.525: TAG - INFO      - Tag info already available, skipping discovery for 8C:F6:81:B8:83:18, type 0
14/Sep 13:55:55.525: TAG - INFO      - Already known Tag at BLE address: 8C:F6:81:B8:83:18, type 0

list a
      There's no advertising tag at all.

list c
      BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
      There's one connected tag overall.

list s
      BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
      There's one synchronized tag overall.

image_update 0 image/croissant.png 2
14/Sep 13:56:20.374: TAG - INFO      - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 13:56:20.375: TAG - INFO      - Image file opened: image/croissant.png
14/Sep 13:56:20.384: AP - INFO      - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 0
14/Sep 13:56:21.508: TAG - INFO      - Image sent to device at address 8C:F6:81:B8:83:18, type 0

image_update 1 image/donut.png 2
14/Sep 13:56:25.559: TAG - INFO      - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 13:56:25.559: TAG - INFO      - Image file opened: image/donut.png
14/Sep 13:56:25.568: AP - INFO      - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 1
14/Sep 13:56:26.771: TAG - INFO      - Image sent to device at address 8C:F6:81:B8:83:18, type 0

```

**Figure 32. Creating a connection to an ESL tag in Synchronized state will switch the tag into Updating state.**

And once the data transfer is done, the tag can be switched back to the Synchronized state by the AP closing the connection to:

```
disconnect 2
```

Even though the images on indexes 0 and 1 were updated, the image on the screen hasn't been changed. The new images just uploaded to the tag can be now displayed on the WSTK LCD screen. Command

```
display_image 2 0 0
```

will display a picture of a croissant on the WSTK LCD screen (screen index 0).

```

31.1
read_sensor 1 2
14/Sep 13:55:33.527: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:55:33.527: SEN - INFO - Sensor data received: 0x8e0230312e30302e3030
Sensor type: Device firmware version
01.00.00

read_sensor 1 3
14/Sep 13:55:38.147: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:55:38.148: SEN - INFO - Sensor data received: 0x3e03db4a00
Sensor type: Date of manufacture
20 June, 2022

read_sensor 1 4
14/Sep 13:55:42.766: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:55:42.766: SEN - INFO - Sensor data received: 0x4e040000581b
Sensor type: Device operating temperature range specification
[0.0, 70.0]

connect 2
14/Sep 13:55:53.112: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:83:18, type 0
14/Sep 13:55:53.113: AP - INFO - Request connecting to: 8C:F6:81:B8:83:18, type 0 over PAWR
14/Sep 13:55:55.525: TAG - INFO - Tag info already available, skipping discovery for 8C:F6:81:B8:83:18, type 0
14/Sep 13:55:55.525: TAG - INFO - Already known Tag at BLE address: 8C:F6:81:B8:83:18, type 0

list a
There's no advertising tag at all.

list c
BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
There's one connected tag overall.

list s
BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
There's one synchronized tag overall.

image_update 0 image/croissant.png 2
14/Sep 13:56:20.374: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 13:56:20.375: TAG - INFO - Image file opened: image/croissant.png
14/Sep 13:56:20.384: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 0
14/Sep 13:56:21.508: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0

image_update 1 image/donut.png 2
14/Sep 13:56:25.559: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 13:56:25.559: TAG - INFO - Image file opened: image/donut.png
14/Sep 13:56:25.568: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 1
14/Sep 13:56:26.771: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0

disconnect 2
14/Sep 13:56:35.930: TAG - INFO - Command: 0402 written successfully for 8C:F6:81:B8:83:18, type 0
14/Sep 13:56:36.022: TAG - INFO - Connection to 8C:F6:81:B8:83:18, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED

list a
There's no advertising tag at all.

list c
There's no connected tag at all.

list s
BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
There are 2 synchronized tags overall.

display_image 2 0 0
14/Sep 13:56:53.663: AP - INFO - Reply in slot 0 from ESL ID 2 in group 0.
14/Sep 13:56:53.664: RSP - INFO - Display State Response received
Display state: Display_Index: 0 Image_Index: 0 (0x110000)

```

**Figure 33.** After uploading new images to an ESL tag (in Updating state), the AP will close the connection to the tag, switching it back to synchronized state. Now the access point can again send commands to the tag via periodic advertisements.

The Updating state is also the state from which the ESL tags can be unassociated from the AP / network, or to do a factory reset.

To do the factory reset, first create a connection to switch the tag into Updating state:

```
connect 1
```



```

Python
14/Sep 13:55:42.766: SEN - INFO - Sensor data received: 0x4e040000581b
Sensor type: Device operating temperature range specification
[0.0, 70.0]

connect 2
14/Sep 13:55:53.112: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:83:18, type 0
14/Sep 13:55:53.113: AP - INFO - Request connecting to: 8C:F6:81:B8:83:18, type 0 over PAWR
14/Sep 13:55:55.525: TAG - INFO - Tag info already available, skipping discovery for 8C:F6:81:B8:83:18, type 0
14/Sep 13:55:55.525: TAG - INFO - Already known Tag at BLE address: 8C:F6:81:B8:83:18, type 0
list a
There's no advertising tag at all.
list c
BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
There's one connected tag overall.
list s
BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
There's one synchronized tag overall.
image_update 0 image/croissant.png 2
14/Sep 13:56:20.374: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 13:56:20.375: TAG - INFO - Image file opened: image/croissant.png
14/Sep 13:56:20.384: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 0
14/Sep 13:56:21.508: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0
image_update 1 image/donut.png 2
14/Sep 13:56:25.559: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 13:56:25.559: TAG - INFO - Image file opened: image/donut.png
14/Sep 13:56:25.568: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 1
14/Sep 13:56:26.771: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0
disconnect 2
14/Sep 13:56:35.930: TAG - INFO - Command: 0402 written successfully for 8C:F6:81:B8:83:18, type 0
14/Sep 13:56:36.022: TAG - INFO - Connection to 8C:F6:81:B8:83:18, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
list a
There's no advertising tag at all.
list c
There's no connected tag at all.
list s
BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
There are 2 synchronized tags overall.
display_image 2 0 0
14/Sep 13:56:53.663: AP - INFO - Reply in slot 0 from ESL ID 2 in group 0.
14/Sep 13:56:53.664: RSP - INFO - Display State Response received
Display state: Display_Index: 0 Image_Index: 0 (0x110000)
connect 1
14/Sep 13:57:04.900: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:82:BE, type 0
14/Sep 13:57:04.901: AP - INFO - Request connecting to: 8C:F6:81:B8:82:BE, type 0 over PAWR
14/Sep 13:57:07.591: TAG - INFO - Tag info already available, skipping discovery for 8C:F6:81:B8:82:BE, type 0
14/Sep 13:57:07.591: TAG - INFO - Already known Tag at BLE address: 8C:F6:81:B8:82:BE, type 0
list a
There's no advertising tag at all.
list c
BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
There's one connected tag overall.
list s
BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
There's one synchronized tag overall.

```

**Figure 34.** To do a factory reset, the ESL tag has to be in Updating state. Therefore AP has to set the ESL tag first to advertise, and create connection to it.

Once the connection is created (i.e. the ESL tag is in Updating mode), use command

```
factory_reset [-h] [--group_id <u7>] [--pawr] address
```

to reset the ESL tag:

```
factory_reset 1
```

After issuing the command, the ESL tag clears all the images from its memory, clears all settings, and returns to the Unassociated state. Verify this by listing all advertising tags, as seen in [Figure 35](#). Also, the image on the LCD screen is back to the one shown in [Figure 14](#).

```

Python
14/Sep 13:56:35.930: TAG - INFO - Command: 0402 written successfully for 8C:F6:81:B8:83:18, type 0
14/Sep 13:56:36.022: TAG - INFO - Connection to 8C:F6:81:B8:83:18, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
list a
    There's no advertising tag at all.
list c
    There's no connected tag at all.
list s
    BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
    BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
    There are 2 synchronized tags overall.
display_image 2 0 0
14/Sep 13:56:53.663: AP - INFO - Reply in slot 0 from ESL ID 2 in group 0.
14/Sep 13:56:53.664: RSP - INFO - Display State Response received
    Display state: Display_Index: 0 Image_Index: 0 (0x110000)
connect 1
14/Sep 13:57:04.900: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:82:BE, type 0
14/Sep 13:57:04.901: AP - INFO - Request connecting to: 8C:F6:81:B8:82:BE, type 0 over PAWR
14/Sep 13:57:07.591: TAG - INFO - Tag info already available, skipping discovery for 8C:F6:81:B8:82:BE, type 0
14/Sep 13:57:07.591: TAG - INFO - Already known Tag at BLE address: 8C:F6:81:B8:82:BE, type 0
list a
    There's no advertising tag at all.
list c
    BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
    There's one connected tag overall.
list s
    BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
    There's one synchronized tag overall.
help factory_reset
usage: factory_reset [-h] [--group_id <u7>] [--pawr] address

    Execute factory reset on tag.

positional arguments:
  address                Bluetooth address (e.g. 'AA:BB:CC:DD:EE:22') in case insensitive format or ESL ID of the tag.

optional arguments:
  -h, --help            Show this help message.
  --group_id <u7>, -g <u7>  ESL group ID (optional, default is group 0)
  --pawr, -p            Force command through PAWR sync train even if the addressed ESL is currently connected

    Note: the keyword 'all' can be used as a substitute for the ESL broadcast address (0xff)

factory_reset 1
14/Sep 13:57:33.500: AP - INFO - Delete bonding for blocked tag at address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:57:33.650: TAG - INFO - ESL service found at BLE address: 8C:F6:81:B8:82:BE, type 0 with RSSI: -25 dBm
list a
    BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
    There's one advertising tag overall.
list c
    There's no connected tag at all.
list s
    BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
    There's one synchronized tag overall.

```

**Figure 35.** Once the connection is created (i.e. the ESL tag is in Updating state), a factory reset can be commanded. After factory reset, the tag will be in Unassociated state, sending (undirected, connectable) advertising packages.

ESL tags can also be unassociated from the AP / network, with command `unassociate`:

```
unassociate [-h] [--group_id <u7>] address
```

As shown in [Figure 17](#), tags can be unassociated either from the Synchronized state, or from the Updating state. As an example, use the latter method. To set the ESL tag 2 to the Unassociate state, first create a connection to it (i.e., set the tag to the Updating state):

```
connect 2
```

```

Display state: Display_Index: 0 Image_Index: 0 (0x110000)
connect 1
14/Sep 13:57:04.900: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:82:BE, type 0
14/Sep 13:57:04.901: AP - INFO - Request connecting to: 8C:F6:81:B8:82:BE, type 0 over PAWR
14/Sep 13:57:07.591: TAG - INFO - Tag info already available, skipping discovery for 8C:F6:81:B8:82:BE, type 0
14/Sep 13:57:07.591: TAG - INFO - Already known Tag at BLE address: 8C:F6:81:B8:82:BE, type 0
list a
    There's no advertising tag at all.
list c
    BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
    There's one connected tag overall.
list s
    BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
    There's one synchronized tag overall.
help factory_reset
usage: factory_reset [-h] [--group_id <u7>] [--pawr] address

    Execute factory reset on tag.

positional arguments:
  address                Bluetooth address (e.g. 'AA:BB:CC:DD:EE:22') in case insensitive format or ESL ID of the tag.

optional arguments:
  -h, --help            Show this help message.
  --group_id <u7>, -g <u7>  ESL group ID (optional, default is group 0)
  --pawr, -p            Force command through PAWR sync train even if the addressed ESL is currently connected

    Note: the keyword 'all' can be used as a substitute for the ESL broadcast address (0xff)

factory_reset 1
14/Sep 13:57:33.500: AP - INFO - Delete bonding for blocked tag at address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:57:33.650: TAG - INFO - ESL service found at BLE address: 8C:F6:81:B8:82:BE, type 0 with RSSI: -25 dBm
list a
    BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
    There's one advertising tag overall.
list c
    There's no connected tag at all.
list s
    BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
    There's one synchronized tag overall.
connect 2
14/Sep 13:57:55.729: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:83:18, type 0
14/Sep 13:57:55.729: AP - INFO - Request connecting to: 8C:F6:81:B8:83:18, type 0 over PAWR
14/Sep 13:57:58.848: TAG - INFO - Tag info already available, skipping discovery for 8C:F6:81:B8:83:18, type 0
14/Sep 13:57:58.848: TAG - INFO - Already known Tag at BLE address: 8C:F6:81:B8:83:18, type 0
list a
    BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
    There's one advertising tag overall.
list c
    BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
    There's one connected tag overall.
list s
    There's no synchronized tag at all.

```

**Figure 36.** In order to unassociate a tag from the ESL network, the tag has to be either in Synchronized state, or in Updating state.

Once the tag is in the Updating state (i.e. the access point has created connection to the tag), the `unassociate` command can be issued:

```
unassociate 2
```

After the `unassociate` command, the ESL tag is in the Unassociated state, advertising, as shown in [Figure 37](#). As with `factory_reset` command, `unassociate` will also clear the images and LED states. But this is however controlled by the application; user has an option to keep the images and LED states, if needed, with slight modifications to the application.

```

Python
factory_reset 1
14/Sep 13:57:33.500: AP - INFO - Delete bonding for blocked tag at address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:57:33.650: TAG - INFO - ESL service found at BLE address: 8C:F6:81:B8:82:BE, type 0 with RSSI: -25 dBm
list a
    BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
    There's one advertising tag overall.
list c
    There's no connected tag at all.
list s
    BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
    There's one synchronized tag overall.
connect 2
14/Sep 13:57:55.729: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:83:18, type 0
14/Sep 13:57:55.729: AP - INFO - Request connecting to: 8C:F6:81:B8:83:18, type 0 over PAWR
14/Sep 13:57:58.848: TAG - INFO - Tag info already available, skipping discovery for 8C:F6:81:B8:83:18, type 0
14/Sep 13:57:58.848: TAG - INFO - Already known Tag at BLE address: 8C:F6:81:B8:83:18, type 0
list a
    BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
    There's one advertising tag overall.
list c
    BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
    There's one connected tag overall.
list s
    There's no synchronized tag at all.
help unassociate
usage: unassociate [-h] [--group_id <u7>] address

    Unassociate ESL(s) from AP.

positional arguments:
  address                Bluetooth address (e.g. 'AA:BB:CC:DD:EE:22') in case insensitive format or ESL ID of the tag.

optional arguments:
  -h, --help            Show this help message.
  --group_id <u7>, -g <u7>  ESL group ID (optional, default is group 0)

Note: the keyword 'all' can be used as a substitute for the ESL broadcast address (0xff)

unassociate 2
14/Sep 13:58:22.203: TAG - INFO - Command: 0102 written successfully for 8C:F6:81:B8:83:18, type 0
14/Sep 13:58:22.293: RSP - INFO - Basic State Response received
    Basic state: No Basic State flag is set (0x100000)
14/Sep 13:58:22.293: AP - INFO - Delete bonding for blocked tag at address 8C:F6:81:B8:83:18, type 0
14/Sep 13:58:22.393: TAG - INFO - ESL service found at BLE address: 8C:F6:81:B8:83:18, type 0 with RSSI: -20 dBm
list a
    BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
    BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
    There are 2 advertising tags overall.
list c
    There's no connected tag at all.
list s
    There's no synchronized tag at all.

```

**Figure 37.** ESL tags can also be set to Unassociated state without factory reset, with command `unassociate <ESL ID>`. This will set the ESL tag to unassociated state, but the tag will not clear the images, or e.g. the LED states.

Creating a connection to device `8C:F6:81:B8:82:BE` (factory reset) will set the device to the Configuring state, [Figure 38](#).

```
connect 8C:F6:81:B8:82:BE
```

```

Python
14/Sep 13:57:55.729: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:83:18, type 0
14/Sep 13:57:55.729: AP - INFO - Request connecting to: 8C:F6:81:B8:83:18, type 0 over PAWR
14/Sep 13:57:58.848: TAG - INFO - Tag info already available, skipping discovery for 8C:F6:81:B8:83:18, type 0
14/Sep 13:57:58.848: TAG - INFO - Already known Tag at BLE address: 8C:F6:81:B8:83:18, type 0
list a
    BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
    There's one advertising tag overall.
list c
    BLE Address: 8C:F6:81:B8:83:18, type 0, ESL ID 2 in group 0 (0x0002)
    There's one connected tag overall.
list s
    There's no synchronized tag at all.
help unassociate
usage: unassociate [-h] [--group_id <u7>] address

    Unassociate ESL(s) from AP.

positional arguments:
  address                Bluetooth address (e.g. 'AA:BB:CC:DD:EE:22') in case insensitive format or ESL ID of the tag.

optional arguments:
  -h, --help            Show this help message.
  --group_id <u7>, -g <u7>  ESL group ID (optional, default is group 0)

Note: the keyword 'all' can be used as a substitute for the ESL broadcast address (0xff)

unassociate 2
14/Sep 13:58:22.203: TAG - INFO - Command: 0102 written successfully for 8C:F6:81:B8:83:18, type 0
14/Sep 13:58:22.293: RSP - INFO - Basic State Response received
    Basic state: No Basic State flag is set (0x100000)
14/Sep 13:58:22.293: AP - INFO - Delete bonding for blocked tag at address 8C:F6:81:B8:83:18, type 0
14/Sep 13:58:22.393: TAG - INFO - ESL service found at BLE address: 8C:F6:81:B8:83:18, type 0 with RSSI: -20 dBm
list a
    BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
    BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
    There are 2 advertising tags overall.
list c
    There's no connected tag at all.
list s
    There's no synchronized tag at all.
connect 8C:F6:81:B8:82:BE
14/Sep 13:58:49.877: AP - INFO - Request connecting to: 8C:F6:81:B8:82:BE, type 0 via connectable advertisement
14/Sep 13:58:52.765: TAG - INFO - Reading tag information from address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:58:52.769: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:82:BE, type 0
14/Sep 13:58:53.122: TAG - INFO - Silabs device found - vendor opcodes are not defined
list a
    BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
    There's one advertising tag overall.
list c
    BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
    There's one connected tag overall.
list s
    There's no synchronized tag at all.

```

**Figure 38.** Creating a connection to the ESL tag reset earlier (*factory\_reset*) will set the device to Configuring state.

Since the ESL tag memory was erased by the factory reset, the access point must be configured again, and optionally new images have to be transferred to the tag:

```

config --full --esl_id 1
image_update 0 image/hamburger.png 1
image_update 1 image/pineapple.png 1

```

```

Python
Unassociate ESL(s) from AP.

positional arguments:
  address                Bluetooth address (e.g. 'AA:BB:CC:DD:EE:22') in case insensitive format or ESL ID of the tag.

optional arguments:
  -h, --help            Show this help message.
  --group_id <u7>, -g <u7>  ESL group ID (optional, default is group 0)

Note: the keyword 'all' can be used as a substitute for the ESL broadcast address (0xff)

unassociate 2
14/Sep 13:58:22.203: TAG - INFO - Command: 0102 written successfully for 8C:F6:81:B8:83:18, type 0
14/Sep 13:58:22.293: RSP - INFO - Basic State Response received
Basic state: No Basic State flag is set (0x100000)
14/Sep 13:58:22.293: AP - INFO - Delete bonding for blocked tag at address 8C:F6:81:B8:83:18, type 0
14/Sep 13:58:22.393: TAG - INFO - ESL service found at BLE address: 8C:F6:81:B8:83:18, type 0 with RSSI: -20 dBm
list a
BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
There are 2 advertising tags overall.
list c
There's no connected tag at all.
list s
There's no synchronized tag at all.
connect 8C:F6:81:B8:82:BE
14/Sep 13:58:49.877: AP - INFO - Request connecting to: 8C:F6:81:B8:82:BE, type 0 via connectable advertisement
14/Sep 13:58:52.765: TAG - INFO - Reading tag information from address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:58:52.769: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:82:BE, type 0
14/Sep 13:58:53.122: TAG - INFO - Silabs device found - vendor opcodes are not defined
list a
BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
There's one advertising tag overall.
list c
BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
There's one connected tag overall.
list s
There's no synchronized tag at all.
config --full --esl_id 1
14/Sep 13:59:12.602: AP - INFO - New auto ESL Address: 0x0000
14/Sep 13:59:12.603: AP - INFO - Set ESL ID to 1.
14/Sep 13:59:12.873: TAG - INFO - ESL Tag fully provisioned at address 8C:F6:81:B8:82:BE, type 0
image_update 0 image/hamburger.png 1
14/Sep 13:59:17.037: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:59:17.038: TAG - INFO - Image file opened: image/hamburger.png
14/Sep 13:59:17.047: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 0
14/Sep 13:59:17.967: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
image_update 1 image/pineapple.png 1
14/Sep 13:59:21.064: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:59:21.065: TAG - INFO - Image file opened: image/pineapple.png
14/Sep 13:59:21.074: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 1
14/Sep 13:59:21.992: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0

```

**Figure 39.** After creating the connection, the tag is still considered as Unassociated, and it requires writing the configuration. Once configuration is written, images can be uploaded to the tag.

Closing the connection to the tag will switch the tag to the Synchronized state, and the tag is again ready to receive commands from the AP via periodic advertisements, [Figure 40](#).

```

disconnect 1
display_image 1 0 0

```



```

Python
14/Sep 13:58:22.293: RSP - INFO - Basic State Response received
Basic state: No Basic State flag is set (0x100000)
14/Sep 13:58:22.293: AP - INFO - Delete bonding for blocked tag at address 8C:F6:81:B8:83:18, type 0
14/Sep 13:58:22.393: TAG - INFO - ESL service found at BLE address: 8C:F6:81:B8:83:18, type 0 with RSSI: -20 dBm
list a
BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
There are 2 advertising tags overall.
list c
There's no connected tag at all.
list s
There's no synchronized tag at all.
connect 8C:F6:81:B8:82:BE
14/Sep 13:58:49.877: AP - INFO - Request connecting to: 8C:F6:81:B8:82:BE, type 0 via connectable advertisement
14/Sep 13:58:52.765: TAG - INFO - Reading tag information from address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:58:52.769: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:82:BE, type 0
14/Sep 13:58:53.122: TAG - INFO - Silabs device found - vendor opcodes are not defined
list a
BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
There's one advertising tag overall.
list c
BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
There's one connected tag overall.
list s
There's no synchronized tag at all.
config --full --esl_id 1
14/Sep 13:59:12.602: AP - INFO - New auto ESL Address: 0x0000
14/Sep 13:59:12.603: AP - INFO - Set ESL ID to 1.
14/Sep 13:59:12.873: TAG - INFO - ESL Tag fully provisioned at address 8C:F6:81:B8:82:BE, type 0
image_update 0 image/hamburger.png 1
14/Sep 13:59:17.037: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:59:17.038: TAG - INFO - Image file opened: image/hamburger.png
14/Sep 13:59:17.047: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 0
14/Sep 13:59:17.967: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
image_update 1 image/pineapple.png 1
14/Sep 13:59:21.064: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:59:21.065: TAG - INFO - Image file opened: image/pineapple.png
14/Sep 13:59:21.074: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 1
14/Sep 13:59:21.992: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
disconnect 1
14/Sep 13:59:32.338: TAG - INFO - Command: 0401 written successfully for 8C:F6:81:B8:82:BE, type 0
14/Sep 13:59:35.130: TAG - INFO - Connection to 8C:F6:81:B8:82:BE, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
list a
BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
There's one advertising tag overall.
list c
There's no connected tag at all.
list s
BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
There's one synchronized tag overall.
display_image 1 0 0
14/Sep 13:59:57.072: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:59:57.073: RSP - INFO - Display State Response received
Display state: Display_Index: 0 Image_Index: 0 (0x110000)

```

**Figure 40.** After creating connection to the ESL tag previously reset, the tag will be in Configuring state, and requires to be configured. After reading/writing the GATT characteristics, the AP can transfer images to the tag, and after closing connection, the tag will be in Synchronized mode.

At this point the ESL tag 1 (8C:F6:81:B8:82:BE) is now in Synchronized state, displaying an image of a hamburger on the WSTK LCD display.

The second ESL tag (8C:F6:81:B8:83:18) is still in Unassociated state, advertising.

The ESL tag can be associated to the AP / network by creating a connection to it (the tag will be in Configuring state), and then closing the connection, after which the tag will be in Synchronized mode. (Figure 41):

```

connect 8C:F6:81:B8:83:18
config --full --esl_id 2
image_update 0 image/pizza.png 2
image_update 1 image/steak.png 2

```

```

Python
There's one advertising tag overall.
list c
BLE Address: 8C:F6:81:B8:82:BE, type 0, Unassociated
There's one connected tag overall.
list s
There's no synchronized tag at all.
config --full --esl_id 1
14/Sep 13:59:12.602: AP - INFO - New auto ESL Address: 0x0000
14/Sep 13:59:12.603: AP - INFO - Set ESL ID to 1.
14/Sep 13:59:12.873: TAG - INFO - ESL Tag fully provisioned at address 8C:F6:81:B8:82:BE, type 0
image_update 0 image/hamburger.png 1
14/Sep 13:59:17.037: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:59:17.038: TAG - INFO - Image file opened: image/hamburger.png
14/Sep 13:59:17.074: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 0
14/Sep 13:59:17.967: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
image_update 1 image/pineapple.png 1
14/Sep 13:59:21.064: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:59:21.065: TAG - INFO - Image file opened: image/pineapple.png
14/Sep 13:59:21.074: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 1
14/Sep 13:59:21.992: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
disconnect 1
14/Sep 13:59:32.338: TAG - INFO - Command: 0401 written successfully for 8C:F6:81:B8:82:BE, type 0
14/Sep 13:59:35.130: TAG - INFO - Connection to 8C:F6:81:B8:82:BE, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
list a
BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
There's one advertising tag overall.
list c
There's no connected tag at all.
list s
BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
There's one synchronized tag overall.
display_image 1 0 0
14/Sep 13:59:57.072: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:59:57.073: RSP - INFO - Display State Response received
Display state: Display_Index: 0 Image_Index: 0 (0x110000)
connect 8C:F6:81:B8:83:18
14/Sep 14:00:19.171: AP - INFO - Request connecting to: 8C:F6:81:B8:83:18, type 0 via connectable advertisement
14/Sep 14:00:21.917: TAG - INFO - Reading tag information from address 8C:F6:81:B8:83:18, type 0
14/Sep 14:00:21.917: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:83:18, type 0
14/Sep 14:00:22.361: TAG - INFO - Silabs device found - vendor opcodes are not defined
config --full --esl_id 2
14/Sep 14:00:25.380: AP - INFO - New auto ESL Address: 0x0001
14/Sep 14:00:25.381: AP - INFO - Set ESL ID to 2.
14/Sep 14:00:25.719: TAG - INFO - ESL Tag fully provisioned at address 8C:F6:81:B8:83:18, type 0
image_update 0 image/pizza.png 2
14/Sep 14:00:30.062: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 14:00:30.063: TAG - INFO - Image file opened: image/pizza.png
14/Sep 14:00:30.072: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 0
14/Sep 14:00:31.124: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0
image_update 1 image/steak.png 2
14/Sep 14:00:35.328: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 14:00:35.328: TAG - INFO - Image file opened: image/steak.png
14/Sep 14:00:35.337: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 1
14/Sep 14:00:36.335: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0

```

**Figure 41. Creating a connection to the ESL tag previously unassociated from the AP will set the tag to a Configuring state. The tag requires the AP to configure it in order to associate the tag to the network, but since the tag was unassociated from the network earlier instead of factory reset, all the images uploaded to the tag are still preserved.**

Once the tag is configured, it can be switched to the Synchronized state by closing the connection. After this, the access point can again send commands to the tag via Periodic Advertising with Responses.

```

disconnect 2
display_image 2 0 0

```



```

Python
14/Sep 13:59:17.037: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:59:17.038: TAG - INFO - Image file opened: image/hamburger.png
14/Sep 13:59:17.047: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 0
14/Sep 13:59:17.967: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
image_update 1 image/pineapple.png 1
14/Sep 13:59:21.064: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 13:59:21.065: TAG - INFO - Image file opened: image/pineapple.png
14/Sep 13:59:21.074: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 1
14/Sep 13:59:21.992: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
disconnect 1
14/Sep 13:59:32.338: TAG - INFO - Command: 0401 written successfully for 8C:F6:81:B8:82:BE, type 0
14/Sep 13:59:35.130: TAG - INFO - Connection to 8C:F6:81:B8:82:BE, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
list a
BLE Address: 8C:F6:81:B8:83:18, type 0, Unassociated
There's one advertising tag overall.
list c
There's no connected tag at all.
list s
BLE Address: 8C:F6:81:B8:82:BE, type 0, ESL ID 1 in group 0 (0x0001)
There's one synchronized tag overall.
display_image 1 0 0
14/Sep 13:59:57.072: AP - INFO - Reply in slot 0 from ESL ID 1 in group 0.
14/Sep 13:59:57.073: RSP - INFO - Display State Response received
Display state: Display_Index: 0 Image_Index: 0 (0x110000)
connect 8C:F6:81:B8:83:18
14/Sep 14:00:19.171: AP - INFO - Request connecting to: 8C:F6:81:B8:83:18, type 0 via connectable advertisement
14/Sep 14:00:21.917: TAG - INFO - Reading tag information from address 8C:F6:81:B8:83:18, type 0
14/Sep 14:00:21.917: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:83:18, type 0
14/Sep 14:00:22.361: TAG - INFO - Silabs device found - vendor opcodes are not defined
config --full --esl_id 2
14/Sep 14:00:25.380: AP - INFO - New auto ESL Address: 0x0001
14/Sep 14:00:25.381: AP - INFO - Set ESL ID to 2.
14/Sep 14:00:25.719: TAG - INFO - ESL Tag fully provisioned at address 8C:F6:81:B8:83:18, type 0
image_update 0 image/pizza.png 2
14/Sep 14:00:30.062: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 14:00:30.063: TAG - INFO - Image file opened: image/pizza.png
14/Sep 14:00:30.072: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 0
14/Sep 14:00:31.124: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0
image_update 1 image/steak.png 2
14/Sep 14:00:35.328: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:83:18, type 0
14/Sep 14:00:35.328: TAG - INFO - Image file opened: image/steak.png
14/Sep 14:00:35.337: AP - INFO - Image update started for tag at 8C:F6:81:B8:83:18, type 0 to image slot 1
14/Sep 14:00:36.335: TAG - INFO - Image sent to device at address 8C:F6:81:B8:83:18, type 0
disconnect 2
14/Sep 14:00:50.523: TAG - INFO - Command: 0402 written successfully for 8C:F6:81:B8:83:18, type 0
14/Sep 14:00:53.686: TAG - INFO - Connection to 8C:F6:81:B8:83:18, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
display_image 2 1 0
14/Sep 14:00:57.185: AP - INFO - Reply in slot 0 from ESL ID 2 in group 0.
14/Sep 14:00:57.185: RSP - INFO - Display State Response received
Display state: Display_Index: 0 Image_Index: 1 (0x110001)
display_image 2 0 0
14/Sep 14:01:07.973: AP - INFO - Reply in slot 0 from ESL ID 2 in group 0.
14/Sep 14:01:07.974: RSP - INFO - Display State Response received
Display state: Display_Index: 0 Image_Index: 0 (0x110000)

```

Figure 42. Since the tag 8C:F6:81:B8:83:18 was unassociated from the network instead of factory reset, all the images stored to it were preserved, and therefore the tag can display the images after associating to a network.

## 7 Modifying the Bluetooth ESL examples

### 7.1 Bluetooth – SoC ESL Tag Example

The **Bluetooth – SoC ESL Tag** example project can be further extended, with some of the features requiring only a configuration change, but the current API also supports a more extensive customization, e.g., adding additional displays. The **Bluetooth – SoC ESL Tag** is also a good starting point to for creating an ESL Tag application for a customer hardware.

The ESL API documentation can be found from [docs.silabs.com](https://docs.silabs.com).

#### 7.1.1 Simple Application Scheduler

The **Bluetooth - SoC ESL Tag** example uses a software component **Application > Utility > Simple Application Scheduler**, which requires configuration if the number of displays and/or LEDs changed.

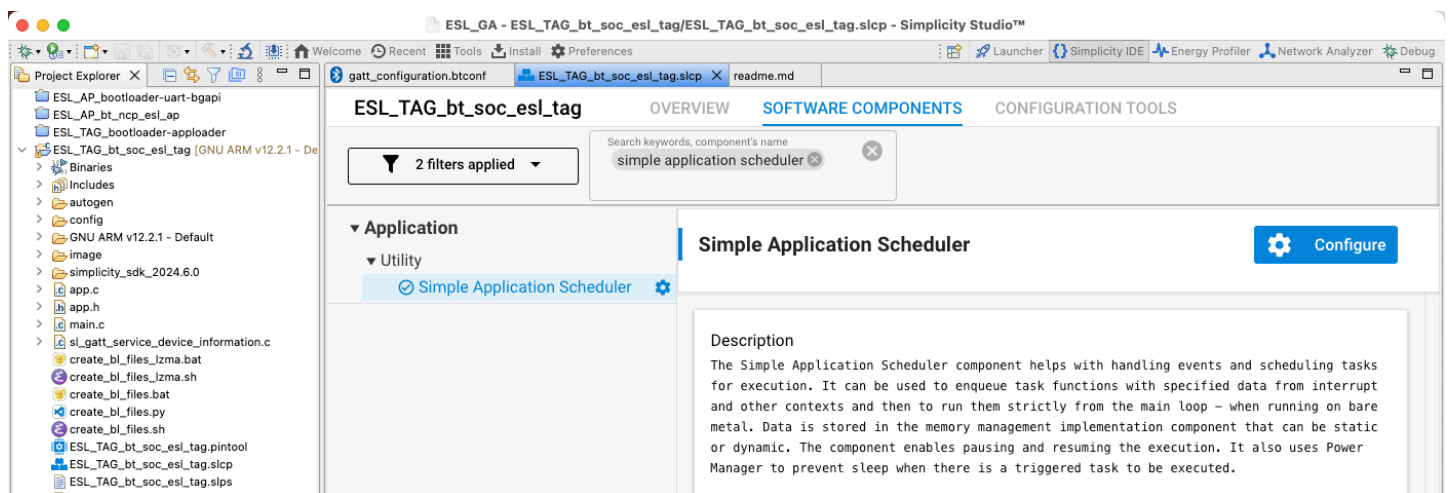


Figure 43. Simple Application Scheduler.

The relevant part here is the **Queue length for static allocation** parameter (Figure 44). The value should be "2 + number of displays + number of LEDs". In case of unmodified **Bluetooth - SoC ESL Tag** example built for a WSTK and a radioboard this value is 4: 2 + one display + one LED.

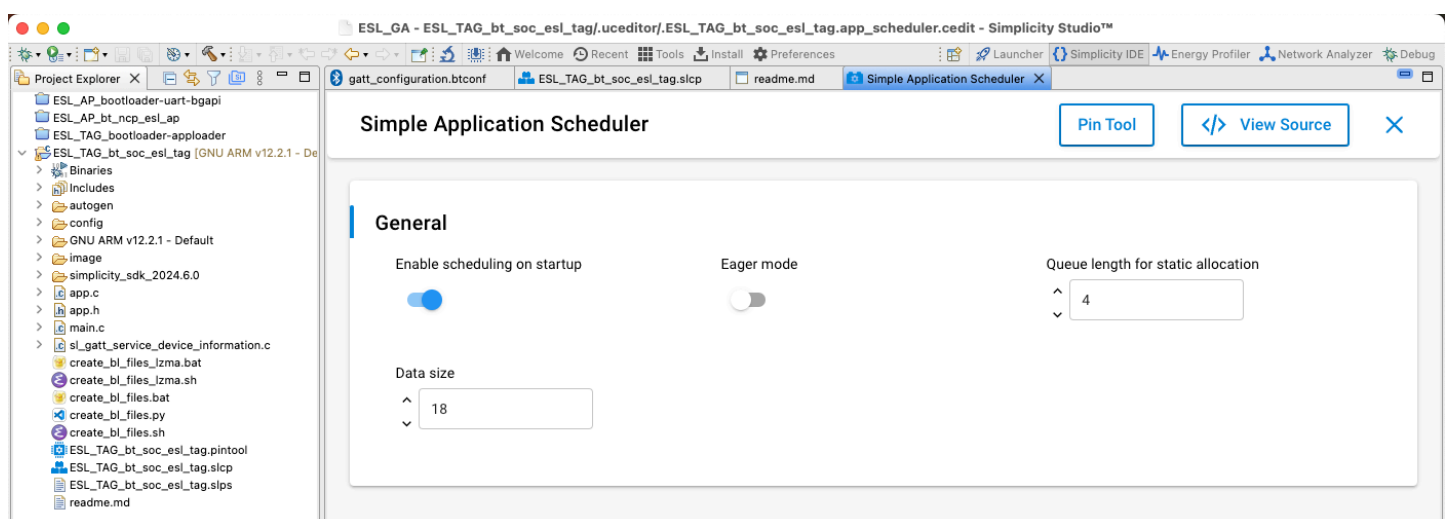


Figure 44. Simple Application Scheduler configuration.

### 7.1.2 Image Storage

Out of the box, the **Bluetooth – SoC ESL Tag** example project can store two separate images to be displayed on the WSTK LCD. The number of images can be configured via **SOFTWARE COMPONENTS**: Select **Bluetooth > Application > GATT Services > ESL Tag RAM Image** (Figure 43), and then click **Configure**. This will open the option for increasing the number of images, and allocating more memory for them, Figure 46.

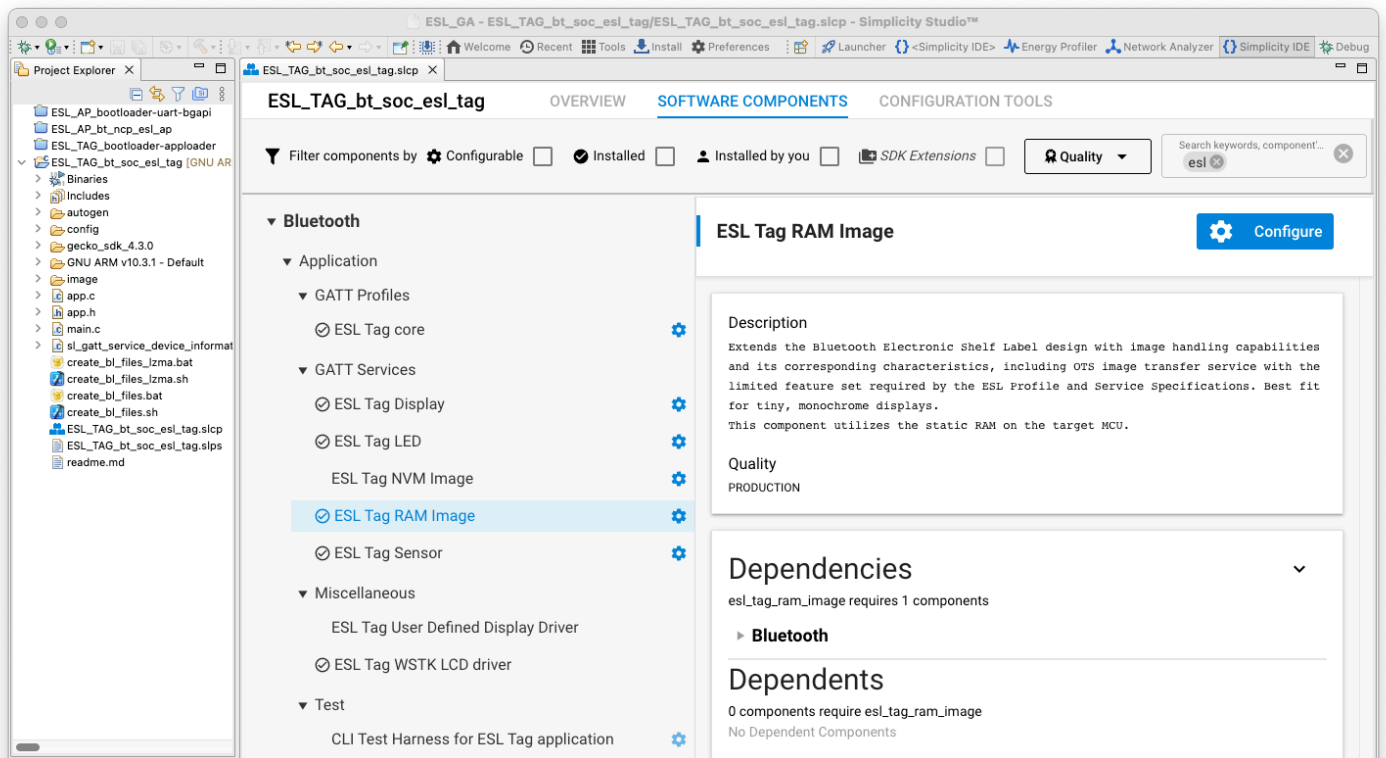


Figure 45. The number of images can be configured via Bluetooth software component: Bluetooth > Application > GATT Services > ESL Tag RAM Image.

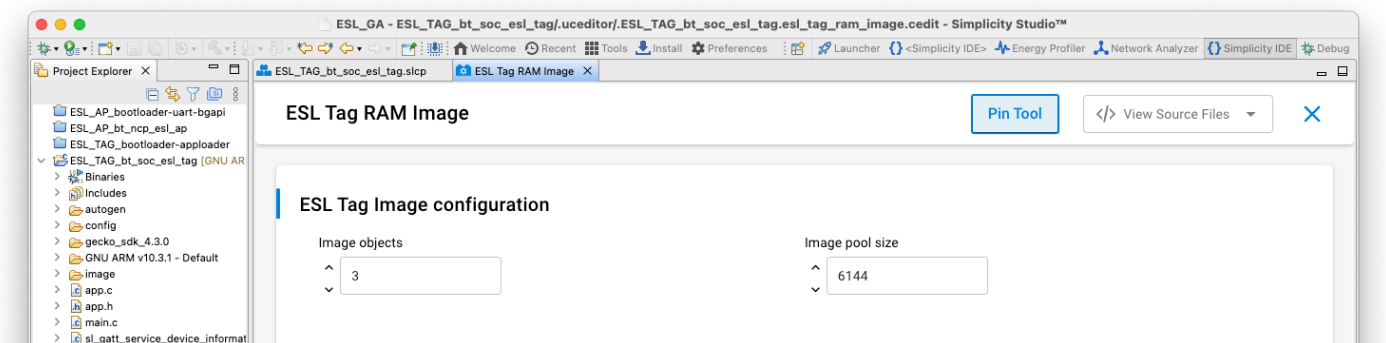
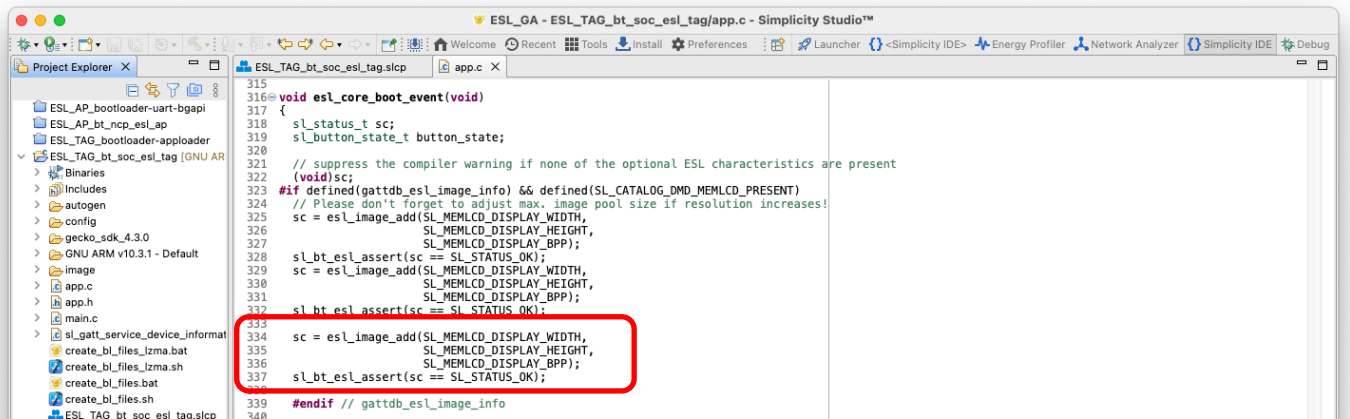


Figure 46. The number of images can be configured via Bluetooth software component: Bluetooth > Application > GATT Services > ESL Tag RAM Image. The Image pool size should be configured accordingly.

The next step is to add the new, additional image storages to the image registry. This happens by calling a function `esl_image_add()`, e.g. on `app.c`, function `esl_core_boot_event()`:



**Figure 47.** For each image, storage configured through Bluetooth > Application > GATT Services > ESL Tag RAM Image must be added to an image registry with function `esl_image_add()`. The third image is added on lines 334 - 337.

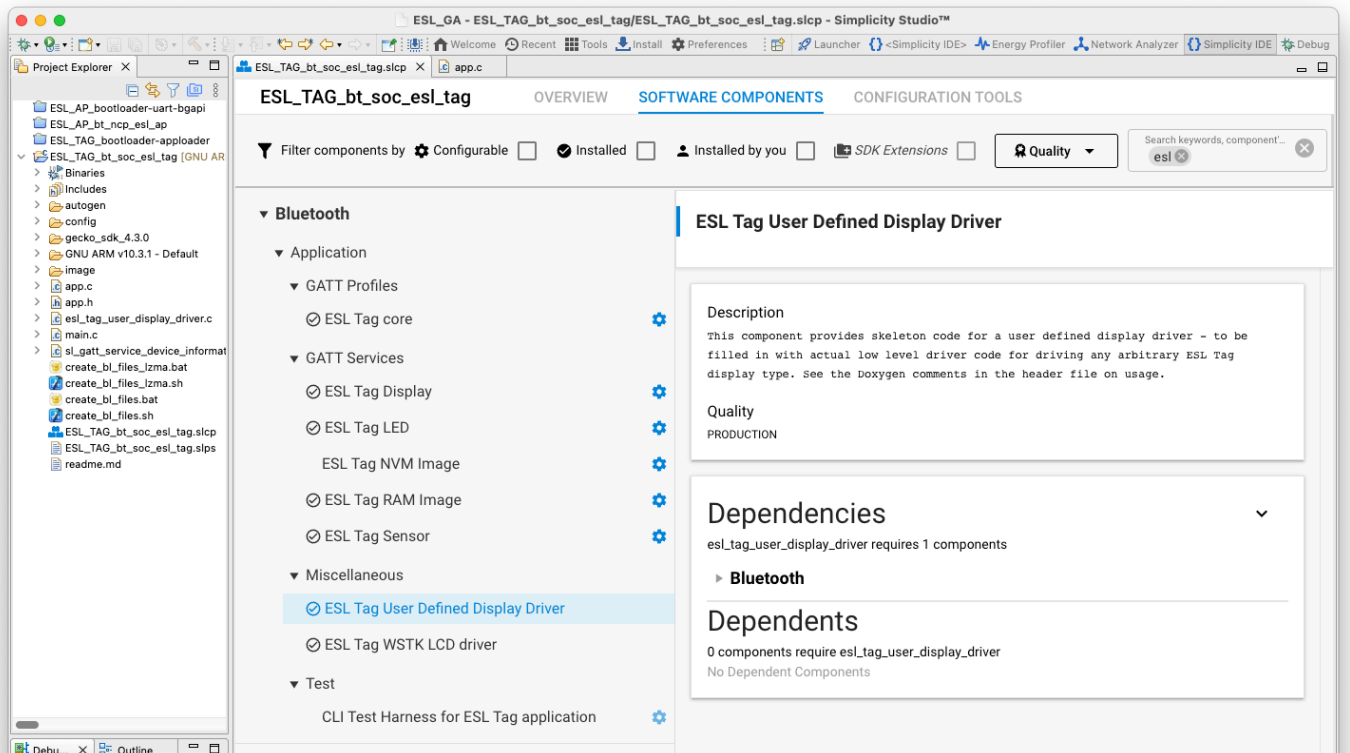
The function parameters are the width and height of the image, and number of bits representing a single pixel of the image.

The images can also be saved to tag's non-volatile memory (NVM). Since the RAM and NVM options are mutually exclusive, the software component Bluetooth > Application > GATT Services > ESL Tag RAM Image must be uninstalled first. Next, install the Bluetooth > Application > GATT Services > ESL Tag NVM Image component, and optionally change the number of images configuration by clicking the **Configure** button. In addition, configure the storage space on NVM3 with the Services > NVM3 > NVM3 Default Config software component.

### 7.1.3 Additional Displays

The **Bluetooth – SoC ESL Tag** example can be extended to support multiple displays, e.g. an electronic paper display (EPD). The example provides an API to add, or register, a new or additional display(s), and a template for the functions in which the display specific driver code should be placed to or called from.

To add an additional display, install the Bluetooth > Application > Miscellaneous > ESL Tag User Defined Display Driver software package, Figure 46.

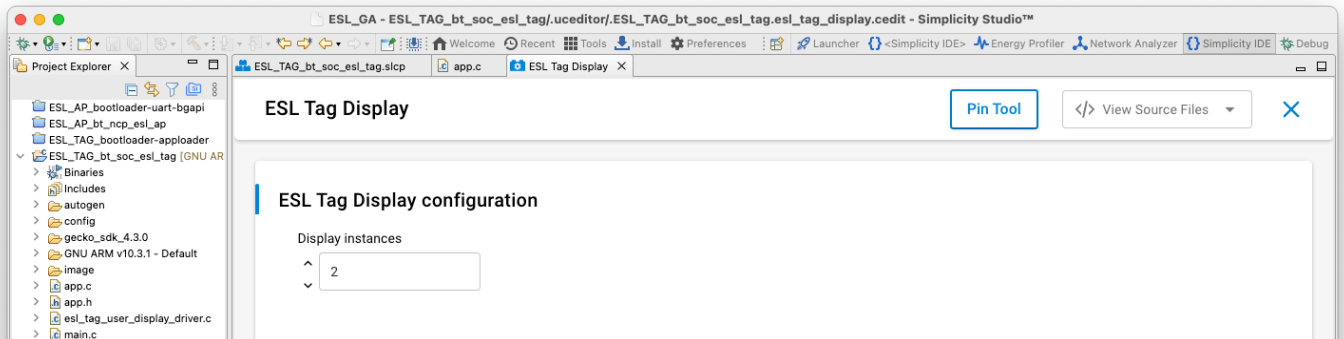


**Figure 48.** To extend the Bluetooth – SoC ESL Tag example project with an additional display (e.g. electronic paper display, EPD), install the Bluetooth > Application > Miscellaneous > ESL Tag User Defined Display Driver component. This will include a file `esl_tag_user_display_driver.c` to the project, including the templates for the required functions.

In addition, in the Bluetooth > Application > GATT Services > ESL Tag Display configuration, increase the Display instances to two (in case of one additional display), Figure 47.

It is also possible to replace the WSTK display with another one. In this case, the ESL Tag User Defined Display Driver should be installed, and the ESL Tag WSTK LCD driver uninstalled.

Also, the configuration for the interface (e.g., SPI or I2C) used by the external display and possible GPIO configuration are required. But being display specific, these won't be covered in this application note. For further information about the interfaces and GPIO configuration, see [docs.silabs.com](https://docs.silabs.com).



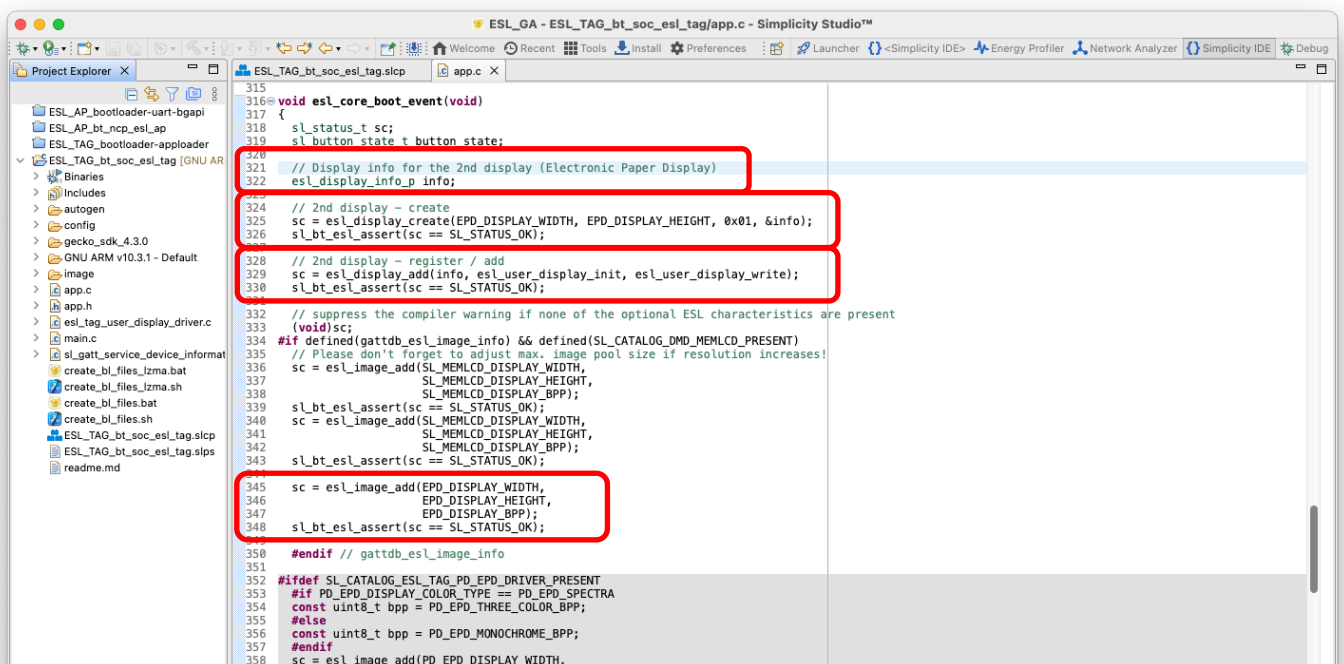
**Figure 49.** The number of display instances can be increased for the second display in Bluetooth > Application > GATT Services > ESL Tag Display software component. This is optional, if the display on WSTK is removed (ESL Tag WSTK LCD driver) or replaced with another one.

To create the display instance, and add it to the display registry, use the API calls `esl_display_create()` (line 325 in Figure 48), and `esl_display_add()` (line 329 in Figure 48). The `esl_display_create()` requires four parameters: the width and height of the display in pixels, the display type defined in Bluetooth SIG Assigned numbers, and a pointer to a `esl_display_info_p` type display info, line 322 in Figure 48.

The `esl_display_add()` requires three parameters: the `esl_display_info_p`, and a function pointers to display initialization function and display write function. The prototypes for the functions `esl_user_display_init()`, Figure 51, and `esl_user_display_write()`, Figure 53, are implemented in `esl_tag_user_display_driver.c`.

Note: If using the functions provided in `esl_tag_user_display_driver.c`, include the header file `esl_tag_user_display_driver.h` in `app.c` (`#include "esl_tag_user_display_driver.h"`).

In addition, a separate storage for images is also needed (line 345 in Figure 48).



**Figure 50.** In addition to software components configuration, the display instance has to be created with function `esl_display_create()`, line 325, and it also has to be added to the display registry with `esl_display_add()`, line 329.



In this example, also the third image configuration on line 345 differs from the previous one; now the third image is being used with the electronic paper display (EPD), while the first two are being displayed on the WSTK LCD.

The optional `esl_user_display_init()` is called at startup and intended for code required to initialize the display on power-on situation.

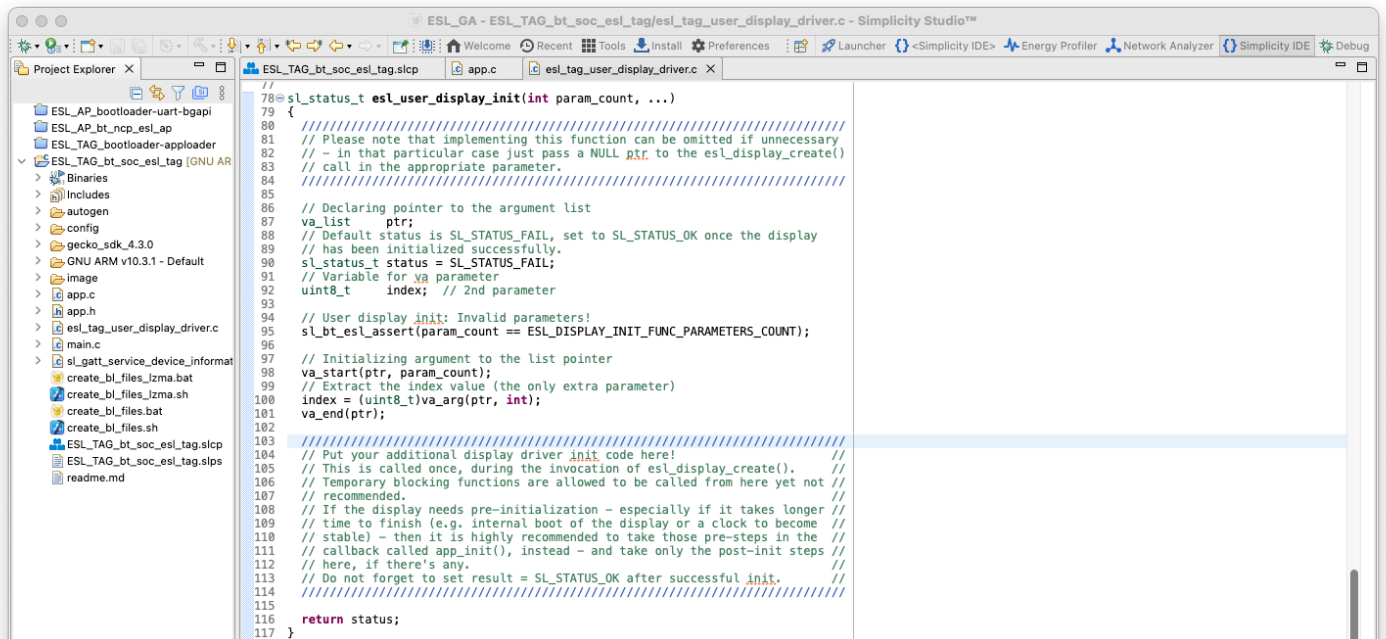


Figure 51. Template for the display initialization function

The only mandatory function to be implemented is the `esl_user_display_write()`. This function is called when the access point requests the ESL tag to display an image on the external display.

The image data can be accessed with an API call `esl_image_get_data()`. The function requires four parameters: the index of the image to be fetched, pointer to an offset variable containing the number of bytes already read (if read in chunks), size of the buffer used, and pointer to the buffer:

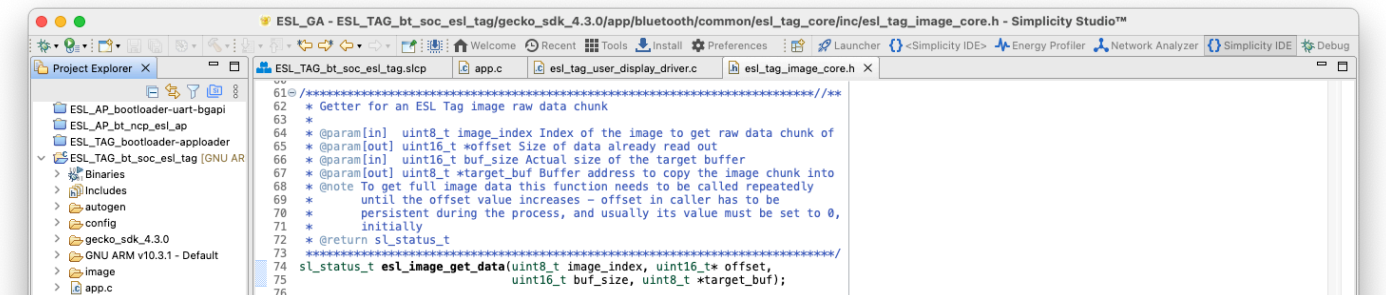


Figure 52. Prototype of the `esl_image_get_data()` function

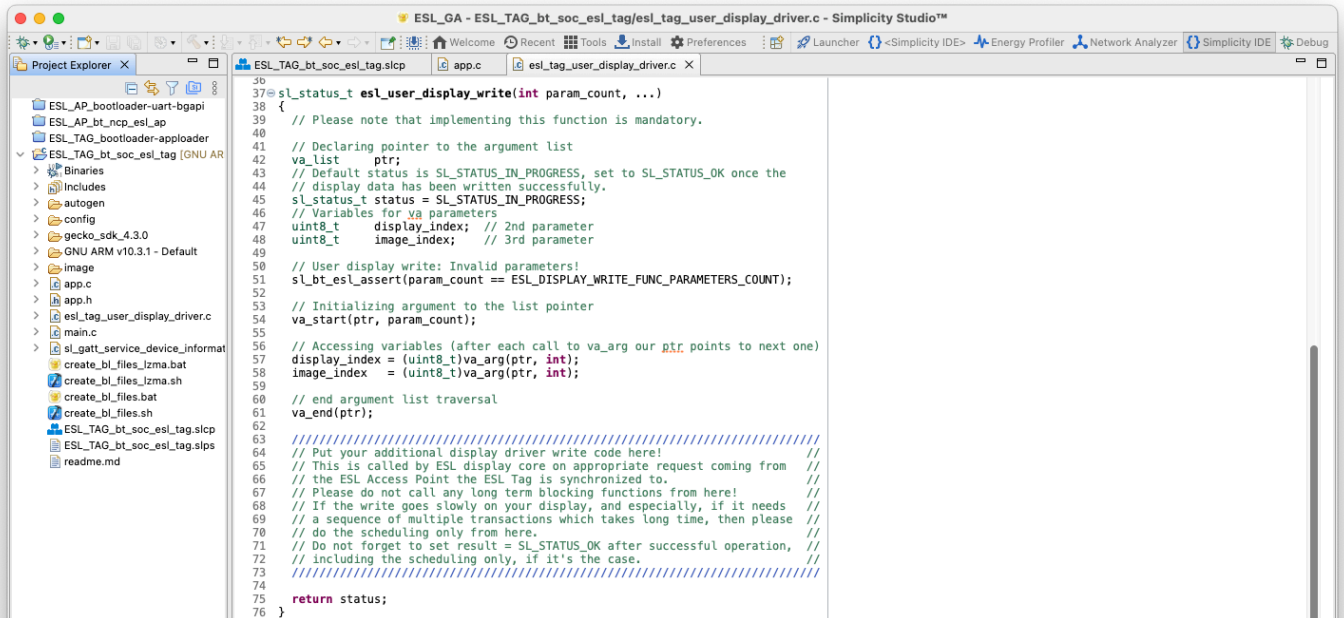


Figure 53. Template for the display write function

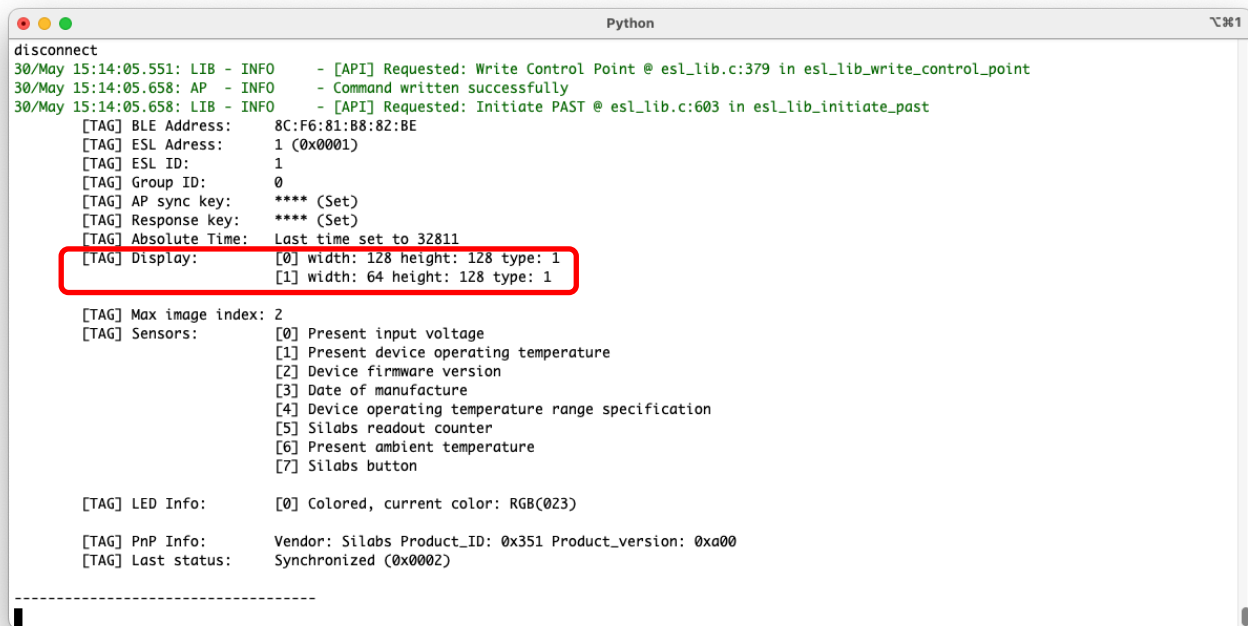


Figure 54. After modifications shown in [Figure 48](#) - [Figure 53](#), the tag will have two displays. The added, second display is now listed with index 1.

## 7.2 ESL AP example

The ESL AP host example software consists of ESL Lib and ESL Key Lib components written in C, and a Python based ESL AP application layer, [Figure 55](#). Both the ESL Lib and ESL Key Lib has a Doxygen style documentation on the header files, describing the interface between the different layers. The source code for ESL Lib and ESL Key Lib are located in `<SDK path>/app/bluetooth/common_host/`.



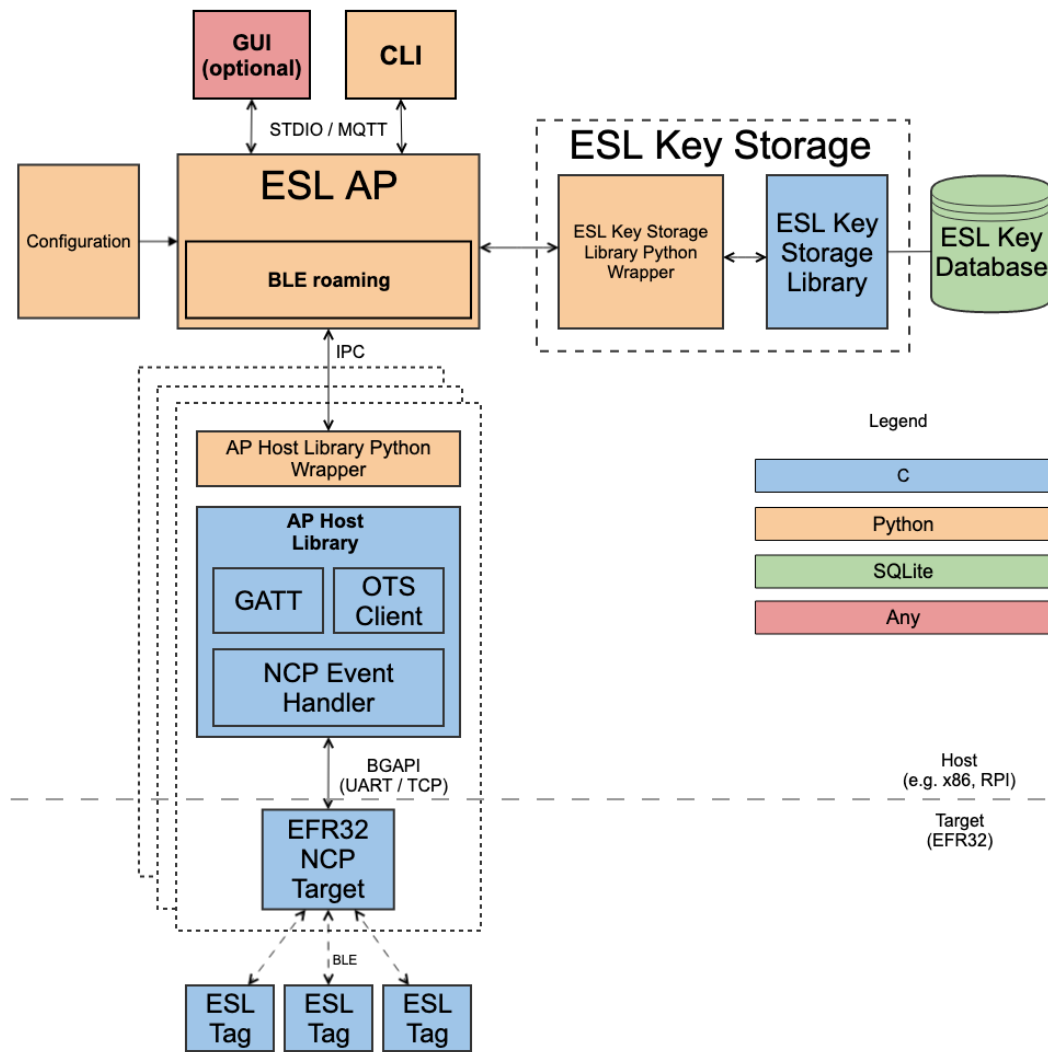


Figure 55. ESL AP logical structure.

## 8 Demo

The access point can also be controlled with a Simplicity Connect mobile application. This requires some additional steps when preparing the ESL tag hardware.

First, build the **Bootloader – SoC Bluetooth AppLoader OTA DFU** and **Bluetooth - SoC ESL Tag** projects normally, as described in chapter 5.2 ESL Tags. Program the bootloader into the ESL tag with Simplicity Studio (or optionally with Simplicity Commander). The compiled application binary will be programmed to the ESL tag by using a separate software called `qrcode_generator.py`, which is in the same folder as the access point (`<SDK path>/app/bluetooth/example_host/bt_host_esl_ap`).

Note: The PATH environment variable must include the path to Simplicity Commander executable, since the `qrcode_generator.py` will use the Simplicity Commander to program the modified application binary to the ESL tag(s).

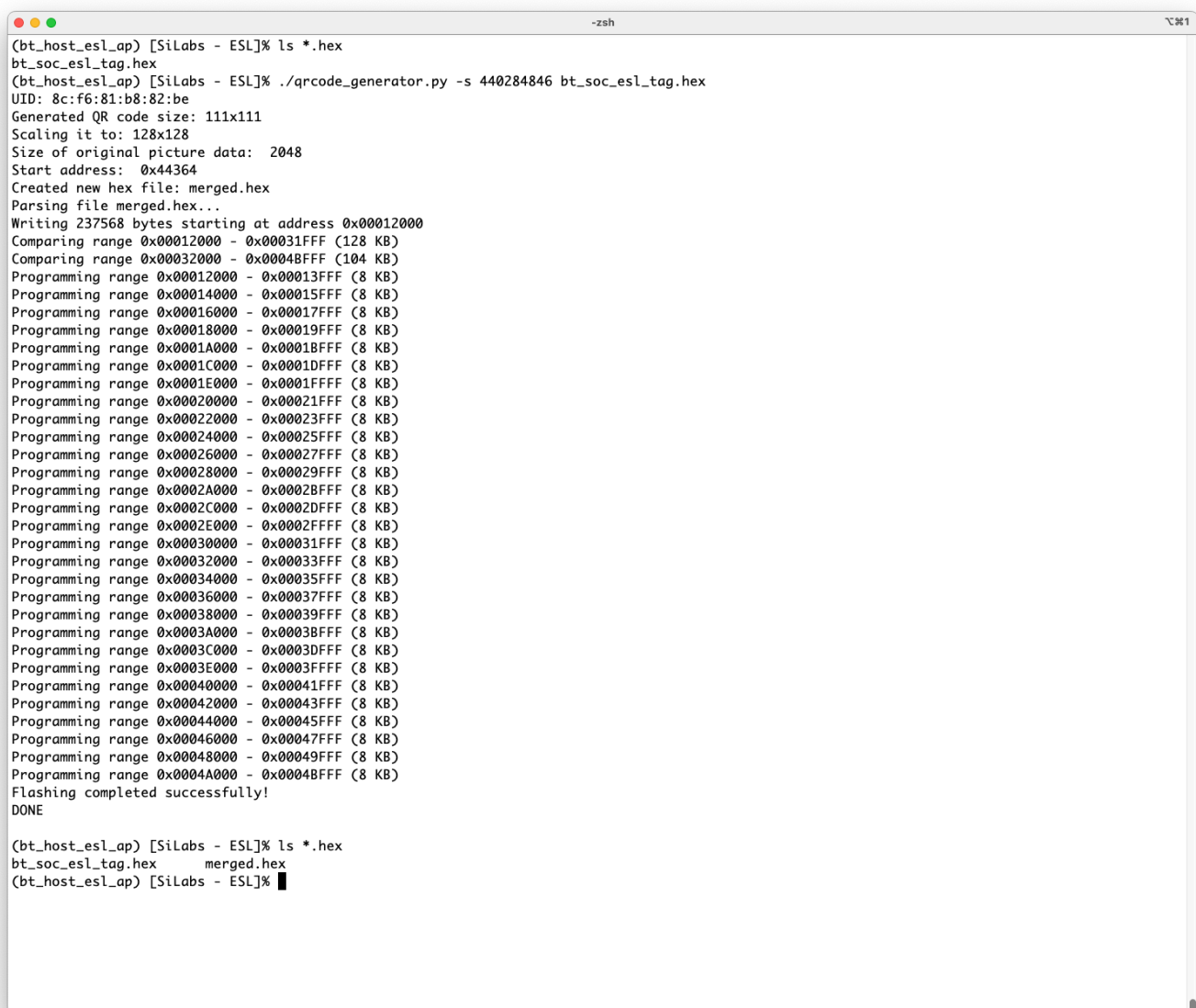
The `qrcode_generator.py` will create a QR code to be used to identify the ESL tag and merge it to the original application binary. Once merged, the `qrcode_generator.py` calls Simplicity Commander and programs the ESL tag.

To program the ESL tag, use the `qrcode_generator.py` and provide the target device and original .hex file as parameters:

```
qrcode_generator.py -s <Serial No> <Path to hex>
```

e.g.

```
./qrcode_generator.py -s 440284846 bt_soc_esl_tag.hex
```



```
(bt_host_esl_ap) [SiLabs - ESL]% ls *.hex
bt_soc_esl_tag.hex
(bt_host_esl_ap) [SiLabs - ESL]% ./qrcode_generator.py -s 440284846 bt_soc_esl_tag.hex
UID: 8c:f6:81:b8:82:be
Generated QR code size: 111x111
Scaling it to: 128x128
Size of original picture data: 2048
Start address: 0x44364
Created new hex file: merged.hex
Parsing file merged.hex...
Writing 237568 bytes starting at address 0x00012000
Comparing range 0x00012000 - 0x00031FFF (128 KB)
Comparing range 0x00032000 - 0x00048FFF (104 KB)
Programming range 0x00012000 - 0x00013FFF (8 KB)
Programming range 0x00014000 - 0x00015FFF (8 KB)
Programming range 0x00016000 - 0x00017FFF (8 KB)
Programming range 0x00018000 - 0x00019FFF (8 KB)
Programming range 0x0001A000 - 0x0001BFFF (8 KB)
Programming range 0x0001C000 - 0x0001DFFF (8 KB)
Programming range 0x0001E000 - 0x0001FFFF (8 KB)
Programming range 0x00020000 - 0x00021FFF (8 KB)
Programming range 0x00022000 - 0x00023FFF (8 KB)
Programming range 0x00024000 - 0x00025FFF (8 KB)
Programming range 0x00026000 - 0x00027FFF (8 KB)
Programming range 0x00028000 - 0x00029FFF (8 KB)
Programming range 0x0002A000 - 0x0002BFFF (8 KB)
Programming range 0x0002C000 - 0x0002DFFF (8 KB)
Programming range 0x0002E000 - 0x0002FFFF (8 KB)
Programming range 0x00030000 - 0x00031FFF (8 KB)
Programming range 0x00032000 - 0x00033FFF (8 KB)
Programming range 0x00034000 - 0x00035FFF (8 KB)
Programming range 0x00036000 - 0x00037FFF (8 KB)
Programming range 0x00038000 - 0x00039FFF (8 KB)
Programming range 0x0003A000 - 0x0003BFFF (8 KB)
Programming range 0x0003C000 - 0x0003DFFF (8 KB)
Programming range 0x0003E000 - 0x0003FFFF (8 KB)
Programming range 0x00040000 - 0x00041FFF (8 KB)
Programming range 0x00042000 - 0x00043FFF (8 KB)
Programming range 0x00044000 - 0x00045FFF (8 KB)
Programming range 0x00046000 - 0x00047FFF (8 KB)
Programming range 0x00048000 - 0x00049FFF (8 KB)
Programming range 0x0004A000 - 0x0004BFFF (8 KB)
Flashing completed successfully!
DONE
(bt_host_esl_ap) [SiLabs - ESL]% ls *.hex
bt_soc_esl_tag.hex      merged.hex
(bt_host_esl_ap) [SiLabs - ESL]% █
```

Figure 56. Programming ELS Tag with image prepared for a demo

Once the `qrcode_generator.py` has programmed the merged binary to the ESL tag (indicated with line "Flashing completed successfully!"), the ESL tag will boot and display a QR code on the WSTK LCD display.



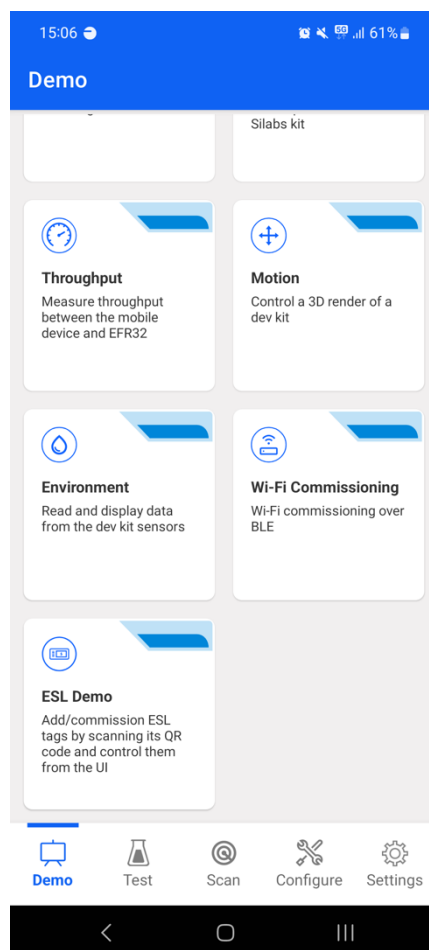
Figure 57. ESL Tag prepared for a demo

In order to use the demo functionality, also the access point has to be started in demo mode (with command line options `--demo -u`), or the demo mode can also be enabled via running access point with command `demo on`.

```
Python
(bt_host_esl_ap) [SiLabs - ESL]% python app.py --demo -u /dev/tty.usbmodem0004402847501
14/Sep 15:06:12.657: AP - WARNING - Starting with NCP encryption disabled!
[I] Opened port on posix.
14/Sep 15:06:12.667: LIB - INFO - [[CORE] Resetting NCP target... in esl_lib_init() @ 131:esl_lib_core.c
[I] Bluetooth stack booted: v6.1.0-b1357
14/Sep 15:06:12.836: LIB - INFO - [[CORE] Bluetooth type 0 public address: 8C:F6:81:B8:83:66 in esl_lib_core_on_bt_event() @ 221:esl_lib_core.c
14/Sep 15:06:12.858: AP - INFO - Command line event handler started, system is idle.
14/Sep 15:06:12.858: AP - INFO - Type 'help' for available commands.
14/Sep 15:06:13.013: AP - INFO - Periodic advertisement started.
14/Sep 15:06:13.025: AP - INFO - Scanning started.
14/Sep 15:06:13.100: TAG - INFO - ESL service found at BLE address: 8C:F6:81:B8:82:BE, type 0 with RSSI: -25 dBm
```

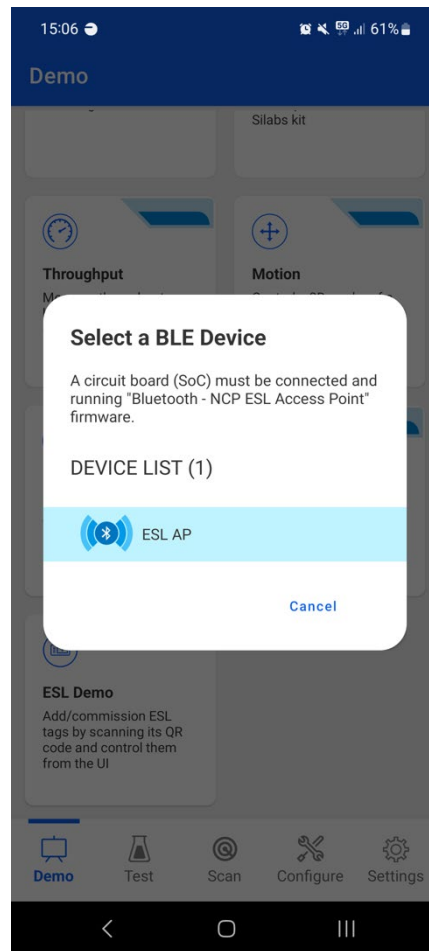
Figure 58. Running ESL access point demo

When the access point is used in demo mode, it advertises its existence. Now the user can connect to the access point with EFR Connect mobile application. Select the Demo tab, and then the ESL Demo application.



**Figure 59. ESL Demo in Simplicity Connect mobile application**

Enabling the Demo feature on the ESL access point will make the access point to advertise, and now the Simplicity Connect mobile application can create a connection to the access point.



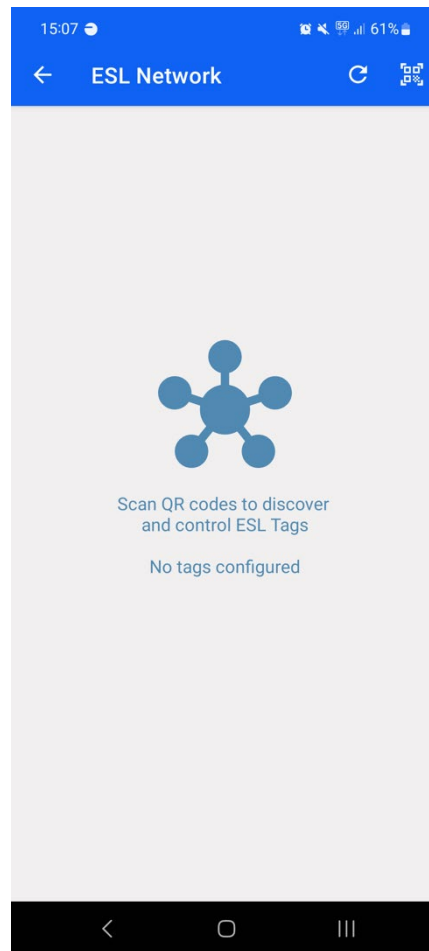
**Figure 60. ESL access point visible in Simplicity Connect mobile application**

Successful connection creation will also be indicated by the access point, as show in Figure 61.

```
Python
(bt_host_esl_ap) [SiLabs - ESL]% python app.py --demo -u /dev/tty.usbmodem0004402847501
14/Sep 15:06:12.657: AP - WARNING - Starting with NCP encryption disabled!
[I] Opened port on posix.
14/Sep 15:06:12.667: LIB - INFO - [[CORE] Resetting NCP target... in esl_lib_init() @ 131:esl_lib_core.c
[I] Bluetooth stack booted: v6.1.0-b1357
14/Sep 15:06:12.836: LIB - INFO - [[CORE] Bluetooth type 0 public address: 8C:F6:81:B8:83:66 in esl_lib_core_on_bt_event() @ 221:esl_lib_core.c
14/Sep 15:06:12.858: AP - INFO - Command line event handler started, system is idle.
14/Sep 15:06:12.858: AP - INFO - Type 'help' for available commands.
14/Sep 15:06:13.013: AP - INFO - Periodic advertisement started.
14/Sep 15:06:13.025: AP - INFO - Scanning started.
14/Sep 15:06:13.100: TAG - INFO - ESL service found at BLE address: 8C:F6:81:B8:82:BE, type 0 with RSSI: -25 dBm
14/Sep 15:07:00.569: AP - INFO - AP control status connected
14/Sep 15:07:01.423: AP - INFO - AP control status subscribed
```

**Figure 61. Simplicity Connect mobile application connected to the ESL access point**

The ESL tags can now be associated to the ESL network by reading the QR code displayed on the WSTK LCD screen with the EFR Connect mobile app. Click the QR code symbol (upper left corner on Figure 62), and read the ESL Tag's QR code.



**Figure 62. Simplicity Connect ready to scan ESL Tag QR codes**

Once the QR has been read, the mobile app will command the access point to connect to the ESL tag in question, and the tag will be in configuring state, Figure 63.

```

Python
(bt_host_esl_ap) [Silabs - ESL]% python app.py --demo -u /dev/tty.usbmodem0004402847501
14/Sep 15:06:12.657: AP - WARNING - Starting with NCP encryption disabled!
[I] Opened port on posix.
14/Sep 15:06:12.667: LIB - INFO - [CORE] Resetting NCP target... in esl_lib_init() @ 131:esl_lib_core.c
[I] Bluetooth stack booted: v6.1.0-b1357
14/Sep 15:06:12.836: LIB - INFO - [CORE] Bluetooth type 0 public address: 8C:F6:81:B8:83:66 in esl_lib_core_on_bt_event() @ 221:esl_lib_core.c
14/Sep 15:06:12.858: AP - INFO - Command line event handler started, system is idle.
14/Sep 15:06:12.858: AP - INFO - Type 'help' for available commands.
14/Sep 15:06:13.013: AP - INFO - Periodic advertisement started.
14/Sep 15:06:13.025: AP - INFO - Scanning started.
14/Sep 15:06:13.100: TAG - INFO - ESL service found at BLE address: 8C:F6:81:B8:82:BE, type 0 with RSSI: -25 dBm
14/Sep 15:07:00.569: AP - INFO - AP control status connected
14/Sep 15:07:01.423: AP - INFO - AP control status subscribed
14/Sep 15:07:21.001: AP - INFO - Command arrived from controller: connect 8c:f6:81:b8:82:be
14/Sep 15:07:21.005: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:82:BE, type 0
14/Sep 15:07:21.006: AP - INFO - Request connecting to: 8C:F6:81:B8:82:BE, type 0 via connectable advertisement
14/Sep 15:07:26.196: TAG - INFO - Reading tag information from address 8C:F6:81:B8:82:BE, type 0
14/Sep 15:07:26.197: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:82:BE, type 0
14/Sep 15:07:26.663: TAG - INFO - Silabs device found - vendor opcodes are not defined

```

**Figure 63. ESL access point after connecting ESL tag via Simplicity Connect mobile application**

On this state, the mobile app will ask if the tag should be configured, Figure 64. If the Configure option is selected, the access point will assign an ESL ID and group for the tag, and read and write all the other necessary configuration parameters, Figure 65.

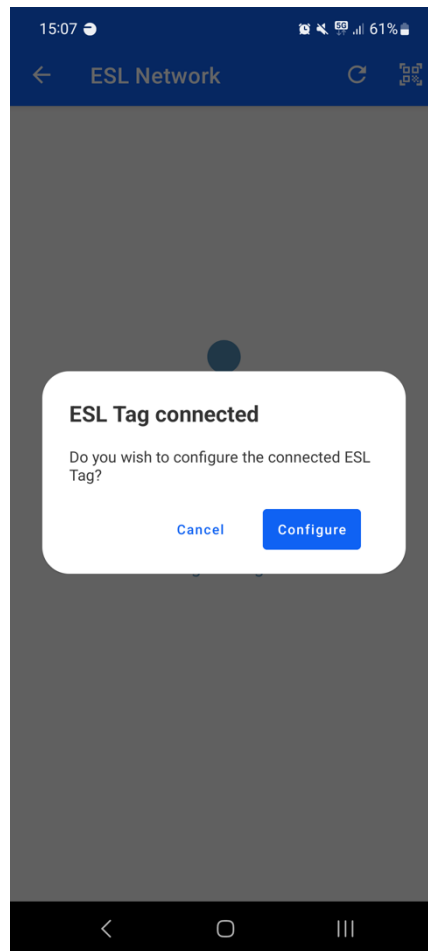


Figure 64. ESL tag connected, ready to be configured

The ESL access point command line interface will also show that the ESL tag has now been configured.

```
Python
(bt_host_esl_ap) [Silabs - ESL]% python app.py --demo -u /dev/tty.usbmodem0004402847501
14/Sep 15:06:12.657: AP - WARNING - Starting with NCP encryption disabled!
[I] Opened port on posix.
14/Sep 15:06:12.667: LIB - INFO - [CORE] Resetting NCP target... in esl_lib_init() @ 131:esl_lib_core.c
[I] Bluetooth stack booted: v6.1.0-b1357
14/Sep 15:06:12.836: LIB - INFO - [CORE] Bluetooth type 0 public address: 8C:F6:81:B8:83:66 in esl_lib_core_on_bt_event() @ 221:esl_lib_core.c
14/Sep 15:06:12.858: AP - INFO - Command line event handler started, system is idle.
14/Sep 15:06:12.858: AP - INFO - Type 'help' for available commands.
14/Sep 15:06:13.013: AP - INFO - Periodic advertisement started.
14/Sep 15:06:13.025: AP - INFO - Scanning started.
14/Sep 15:06:13.100: TAG - INFO - ESL service found at BLE address: 8C:F6:81:B8:82:BE, type 0 with RSSI: -25 dBm
14/Sep 15:07:00.569: AP - INFO - AP control status connected
14/Sep 15:07:01.423: AP - INFO - AP control status subscribed
14/Sep 15:07:21.001: AP - INFO - Command arrived from controller: connect 8c:f6:81:b8:82:be
14/Sep 15:07:21.005: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:82:BE, type 0
14/Sep 15:07:21.006: AP - INFO - Request connecting to: 8C:F6:81:B8:82:BE, type 0 via connectable advertisement
14/Sep 15:07:26.196: TAG - INFO - Reading tag information from address 8C:F6:81:B8:82:BE, type 0
14/Sep 15:07:26.197: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:82:BE, type 0
14/Sep 15:07:26.663: TAG - INFO - Silabs device found - vendor opcodes are not defined
14/Sep 15:07:37.591: AP - INFO - Command arrived from controller: config full
14/Sep 15:07:37.591: AP - INFO - New auto ESL Address: 0x0000
14/Sep 15:07:37.965: TAG - INFO - ESL Tag fully provisioned at address 8C:F6:81:B8:82:BE, type 0
```

Figure 65. ESL Tag configured

In addition, the Simplicity Connect mobile app will also offer an opportunity to upload an image from the mobile to the ESL tag, Figure 66 and Figure 67, and optionally to display the image automatically on the tag.



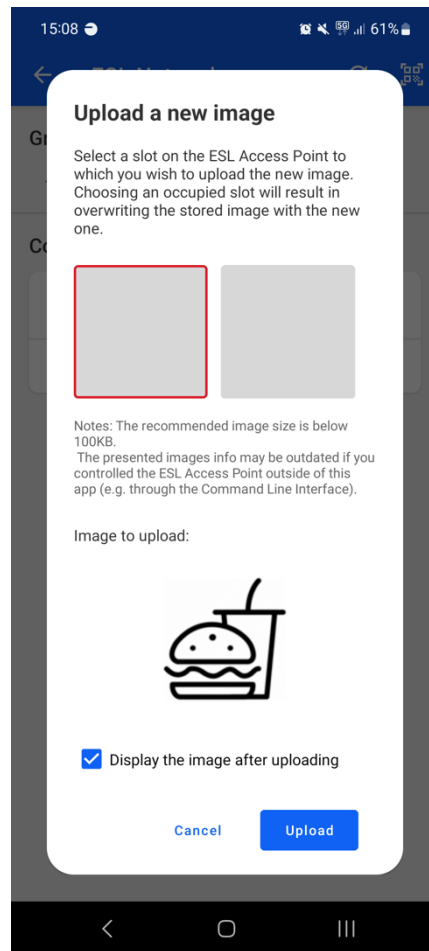


Figure 66. Uploading images to ESL tag via Simplicity Connect mobile application

```
Python
(bt_host_esl_ap) [Silabs - ESL]% python app.py --demo -u /dev/tty.usbmodem0004402847501
14/Sep 15:06:12.657: AP - WARNING - Starting with NCP encryption disabled!
[I] Opened port on posix.
14/Sep 15:06:12.667: LIB - INFO - [CORE] Resetting NCP target... in esl_lib_init() @ 131:esl_lib_core.c
[I] Bluetooth stack booted: v6.1.0-b1357
14/Sep 15:06:12.836: LIB - INFO - [CORE] Bluetooth type 0 public address: 8C:F6:81:B8:83:66 in esl_lib_core_on_bt_event() @ 221:esl_lib_core.c
14/Sep 15:06:12.858: AP - INFO - Command line event handler started, system is idle.
14/Sep 15:06:12.858: AP - INFO - Type 'help' for available commands.
14/Sep 15:06:13.013: AP - INFO - Periodic advertisement started.
14/Sep 15:06:13.025: AP - INFO - Scanning started.
14/Sep 15:06:13.100: TAG - INFO - ESL service found at BLE address: 8C:F6:81:B8:82:BE, type 0 with RSSI: -25 dBm
14/Sep 15:07:00.569: AP - INFO - AP control status connected
14/Sep 15:07:01.423: AP - INFO - AP control status subscribed
14/Sep 15:07:21.001: AP - INFO - Command arrived from controller: connect 8c:f6:81:b8:82:be
14/Sep 15:07:21.005: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:82:BE, type 0
14/Sep 15:07:21.006: AP - INFO - Request connecting to: 8C:F6:81:B8:82:BE, type 0 via connectable advertisement
14/Sep 15:07:26.196: TAG - INFO - Reading tag information from address 8C:F6:81:B8:82:BE, type 0
14/Sep 15:07:26.197: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:82:BE, type 0
14/Sep 15:07:26.663: TAG - INFO - Silabs device found - vendor opcodes are not defined
14/Sep 15:07:37.591: AP - INFO - Command arrived from controller: config full
14/Sep 15:07:37.591: AP - INFO - New auto ESL Address: 0x0000
14/Sep 15:07:37.965: TAG - INFO - ESL Tag fully provisioned at address 8C:F6:81:B8:82:BE, type 0
14/Sep 15:08:09.586: AP - INFO - Command arrived from controller: image_update 0 image.png
14/Sep 15:08:09.587: AP - INFO - Image update command arrived from controller with the following parameters: index 0, filename image.png
14/Sep 15:08:14.052: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 15:08:14.066: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 0
14/Sep 15:08:14.915: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
14/Sep 15:08:15.002: AP - INFO - Command arrived from controller: disconnect
14/Sep 15:08:15.067: TAG - INFO - Command: 0400 written successfully for 8C:F6:81:B8:82:BE, type 0
14/Sep 15:08:18.873: TAG - INFO - Connection to 8C:F6:81:B8:82:BE, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
14/Sep 15:08:18.967: AP - INFO - Command arrived from controller: display_image 0 0 0
14/Sep 15:08:18.969: AP - INFO - Display image command arrived from controller
14/Sep 15:08:19.435: AP - INFO - Reply in slot 0 from ESL ID 0 in group 0.
14/Sep 15:08:19.435: RSP - INFO - Display State Response received
Display state: Display_Index: 0 Image_Index: 0 (0x110000)
```

Figure 67. Image uploaded to the ESL tag

Once the connection to the ESL tag has been closed, the tag can be controlled via Simplicity Connect mobile application. For example, the LED(s) can be turned on, as shown in Figure 68. This will also be indicated on the access point command line interface, Figure 69.

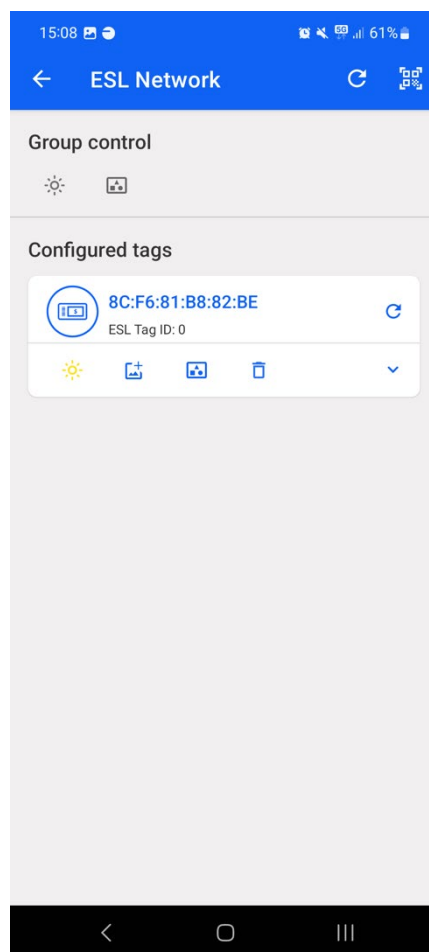


Figure 68. Controlling the ESL tag via Simplicity Connect mobile application

```
(bt_host_esl_ap) [Silabs - ESL]% python app.py --demo -u /dev/tty.usbmodem0004402847501
14/Sep 15:06:12.657: AP - WARNING - Starting with NCP encryption disabled!
[!] Opened port on posix.
14/Sep 15:06:12.667: LIB - INFO - [CORE] Resetting NCP target... in esl_lib_init() @ 131:esl_lib_core.c
[!] Bluetooth stack booted: v6.1.0-b1357
14/Sep 15:06:12.836: LIB - INFO - [CORE] Bluetooth type 0 public address: 8C:F6:81:B8:83:66 in esl_lib_core_on_bt_event() @ 221:esl_lib_core.c
14/Sep 15:06:12.858: AP - INFO - Command line event handler started, system is idle.
14/Sep 15:06:12.858: AP - INFO - Type 'help' for available commands.
14/Sep 15:06:13.013: AP - INFO - Periodic advertisement started.
14/Sep 15:06:13.025: AP - INFO - Scanning started.
14/Sep 15:06:13.100: TAG - INFO - ESL service found at BLE address: 8C:F6:81:B8:82:BE, type 0 with RSSI: -25 dBm
14/Sep 15:07:00.569: AP - INFO - AP control status connected
14/Sep 15:07:01.423: AP - INFO - AP control status subscribed
14/Sep 15:07:21.001: AP - INFO - Command arrived from controller: connect 8c:f6:81:b8:82:be
14/Sep 15:07:21.005: AP - INFO - Bonding LTK found for ESL at 8C:F6:81:B8:82:BE, type 0
14/Sep 15:07:21.006: AP - INFO - Request connecting to: 8C:F6:81:B8:82:BE, type 0 via connectable advertisement
14/Sep 15:07:26.196: TAG - INFO - Reading tag information from address 8C:F6:81:B8:82:BE, type 0
14/Sep 15:07:26.197: TAG - INFO - Registering ESL Tag at BLE address: 8C:F6:81:B8:82:BE, type 0
14/Sep 15:07:26.663: TAG - INFO - Silabs device found - vendor opcodes are not defined
14/Sep 15:07:37.591: AP - INFO - Command arrived from controller: config full
14/Sep 15:07:37.591: AP - INFO - New auto ESL Address: 0x0000
14/Sep 15:07:37.965: TAG - INFO - ESL Tag fully provisioned at address 8C:F6:81:B8:82:BE, type 0
14/Sep 15:08:09.586: AP - INFO - Command arrived from controller: image_update 0 image.png
14/Sep 15:08:09.587: AP - INFO - Image update command arrived from controller with the following parameters: index 0, filename image.png
14/Sep 15:08:14.052: TAG - INFO - Display type matches object type for address 8C:F6:81:B8:82:BE, type 0
14/Sep 15:08:14.066: AP - INFO - Image update started for tag at 8C:F6:81:B8:82:BE, type 0 to image slot 0
14/Sep 15:08:14.915: TAG - INFO - Image sent to device at address 8C:F6:81:B8:82:BE, type 0
14/Sep 15:08:15.002: AP - INFO - Command arrived from controller: disconnect
14/Sep 15:08:15.067: TAG - INFO - Command: 0400 written successfully for 8C:F6:81:B8:82:BE, type 0
14/Sep 15:08:18.873: TAG - INFO - Connection to 8C:F6:81:B8:82:BE, type 0 closed with reason SL_STATUS_BT_CTRL_REMOTE_USER_TERMINATED
14/Sep 15:08:18.967: AP - INFO - Command arrived from controller: display_image 0 0 0
14/Sep 15:08:18.969: AP - INFO - Display image command arrived from controller
14/Sep 15:08:19.435: AP - INFO - Reply in slot 0 from ESL ID 0 in group 0.
14/Sep 15:08:19.435: RSP - INFO - Display State Response received
Display state: Display_Index: 0 Image_Index: 0 (0x110000)
14/Sep 15:08:33.872: AP - INFO - Command arrived from controller: led on 0
14/Sep 15:08:34.851: AP - INFO - Reply in slot 0 from ESL ID 0 in group 0.
14/Sep 15:08:34.851: RSP - INFO - Led State Response received
LED State: LED_Index: 0 (0x0100)
```

Figure 69. LED state change when controlled via Simplicity Connect mobile application

## 9 References

- [1] [www.python.org](http://www.python.org)
- [2] AN1259: Using the Silicon Labs Bluetooth® Stack v3.x and Higher in Network Co-Processor Mode: [www.silabs.com/documents/public/application-notes/an1259-bt-ncp-mode-sdk-v3x.pdf](http://www.silabs.com/documents/public/application-notes/an1259-bt-ncp-mode-sdk-v3x.pdf)
- [3] pip documentation: [pip.pypa.io/en/stable/](http://pip.pypa.io/en/stable/)
- [4] MSYS2 [www.msys2.org](http://www.msys2.org)
- [5] UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher: [www.silabs.com/documents/public/user-guides/ug489-gecko-bootloader-user-guide-gsdk-4.pdf](http://www.silabs.com/documents/public/user-guides/ug489-gecko-bootloader-user-guide-gsdk-4.pdf)
- [6] Electronic Shelf Label Service, Bluetooth Service Specification: [www.bluetooth.com/specifications/specs/electronic-shelf-label-service-1-0/](http://www.bluetooth.com/specifications/specs/electronic-shelf-label-service-1-0/)
- [7] Electronic Shelf Label Profile, Bluetooth Profile Specification: <https://www.bluetooth.com/specifications/specs/electronic-shelf-label-profile-1-0/>

## A. Supported Radio Boards

### Bluetooth ESL Tag Boards

BRD4176A	BRD4187A	BRD4308C
BRD4180A	BRD4187B	BRD4308D
BRD4180B	BRD4187C	BRD4311A
BRD4181A	BRD4188A	BRD4311B
BRD4181B	BRD4188B	BRD4312A
BRD4181C	BRD4195A	BRD4316A
BRD4182A	BRD4196A	BRD4317A
BRD4185A	BRD4197A	BRD4318A
BRD4186A	BRD4198A	BRD4402A
BRD4186B	BRD4308A	
BRD4186C	BRD4308B	

### Bluetooth ESL Access Point Boards

BRD4108A	BRD4191A	BRD4318A
BRD4182A	BRD4310A	BRD4319A
BRD4183A	BRD4311A	BRD4330A
BRD4183B	BRD4311B	BRD4331A
BRD4183C	BRD4312A	BRD4337A
BRD4184A	BRD4314A	BRD4402A
BRD4184B	BRD4316A	BRD4403A
BRD4185A	BRD4317A	

# Simplicity Studio

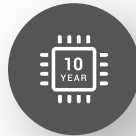
One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!



**IoT Portfolio**  
[www.silabs.com/iot](http://www.silabs.com/iot)



**SW/HW**  
[www.silabs.com/simplicity](http://www.silabs.com/simplicity)



**Quality**  
[www.silabs.com/quality](http://www.silabs.com/quality)



**Support & Community**  
[www.silabs.com/community](http://www.silabs.com/community)

## Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications.

## Trademark Information

Silicon Laboratories Inc., Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Redpine Signals®, WiSeConnect®, n-Link, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, Gecko OS, Gecko OS Studio, Precision32®, Simplicity Studio®, Telegesis, the Telegesis Logo®, USBXpress®, Zentri, the Zentri logo and Zentri DMS, Z-Wave®, and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.



**Silicon Laboratories Inc.**  
400 West Cesar Chavez  
Austin, TX 78701  
USA

**[www.silabs.com](http://www.silabs.com)**