

---

# **Application Note : Telink Zigbee SDK Developer Manual**

AN-19052900-E1

**Ver 1.0.0**

---

**2019/6/27**

## **Brief:**

This document is the guide for Telink Zigbee SDK.



**TELINK SEMICONDUCTOR**

**Published by**  
**Telink Semiconductor**

**Bldg 3, 1500 Zuchongzhi Rd,  
Zhangjiang Hi-Tech Park, Shanghai, China**

**© Telink Semiconductor**  
**All Right Reserved**

### **Legal Disclaimer**

This document is provided as-is. Telink Semiconductor reserves the right to make improvements without further notice to this document or any products herein. This document may contain technical inaccuracies or typographical errors. Telink Semiconductor disclaims any and all liability for any errors, inaccuracies or incompleteness contained herein.

Copyright (c) 2019 Telink Semiconductor (Shanghai) Ltd, Co.

### **Information:**

For further information on the technology, product and business term, please contact Telink Semiconductor Company ([www.telink-semi.com](http://www.telink-semi.com)).

For sales or technical support, please send email to the address of:

[telinknsales@telink-semi.com](mailto:telinknsales@telink-semi.com)

[telinknsupport@telink-semi.com](mailto:telinknsupport@telink-semi.com)

**Revision History**

Version	Major Changes	Date	Author
1.0.0	Initial release	2019/6	YB, WJZ, Cynthia

## Table of contents

1	Overview.....	6
1.1	Zigbee brief .....	6
1.1.1	Device type.....	6
1.1.2	Network type .....	6
1.1.3	Basic terminologies .....	7
1.2	Telink Zigbee SDK brief .....	10
1.3	Software development environment for Telink Zigbee SDK .....	10
1.4	HW platform supported by Telink Zigbee SDK .....	10
2	Telink SDK Development Environment .....	11
2.1	Install SDK .....	11
2.1.1	Import project .....	11
2.1.2	Project directory structure .....	13
2.1.3	Compile existing project option.....	14
2.1.4	Add new project.....	15
2.1.5	Introduction to project configuration.....	17
2.2	Image file burning .....	21
2.2.1	Burning operation .....	21
2.2.2	Flash address allocation.....	23
3	Software architecture.....	24
3.1	\tl_zigbee_sdk\platform .....	24
3.2	Directory for common functions .....	25
3.3	Zigbee stack directory.....	26
3.4	Directory architecture for APP layer.....	27
3.4.1	\common.....	27
3.4.2	\sampleXxxx .....	27
4	Development Guide.....	28
4.1	Print debugging.....	28

4.1.1	UART print.....	28
4.1.2	USB Print .....	29
4.2	Select target board .....	29
4.3	Development flow.....	30
4.3.1	APP-layer initialization (user_init) .....	30
4.3.2	BDB initialization (bdb_init) .....	32
4.3.3	BDB Commissioning .....	33
4.3.4	Data interaction .....	33
4.3.5	Dynamic memory management .....	35
4.3.6	Task scheduling .....	36
4.3.7	Software timer task.....	36
4.3.8	Sleep and wakeup .....	40
4.3.9	GPIO interrupt.....	41
4.3.10	Configure network parameters .....	41
4.4	Common APIs.....	42
4.4.1	Network Management.....	42
4.4.2	Bind Management .....	43
4.4.3	ZDP commands .....	43
4.4.4	Other commands .....	44
4.5	OTA.....	45
4.5.1	OTA initialization.....	45
4.5.2	OTA Server .....	46
4.5.3	OTA Client .....	46
5	HCI Interface for Extended Functions.....	47
5.1	Control flow chart .....	47
5.2	Command frame format .....	48
5.3	Acknowledge format.....	48
5.3.1	Message Type .....	48
5.3.2	Payload.....	48

5.4	BDB command .....	49
5.4.1	Message Type .....	49
5.4.2	Payload.....	49
5.5	Network management command.....	51
5.5.1	Message Type (Host).....	51
5.5.2	Payload (Host).....	52
5.5.3	Message Type (Slave).....	58
5.5.4	Payload (Slave).....	59
5.6	ZCL Cluster command .....	64
5.6.1	ZCL command header format .....	64
5.6.2	General cluster command.....	65
5.6.3	Basic cluster command .....	69
5.6.4	Group cluster command .....	70
5.6.5	Identify cluster command .....	73
5.6.6	On/Off cluster command .....	74
5.6.7	Level cluster command .....	75
5.6.8	Scene cluster command .....	78
5.6.9	OTA cluster command .....	83
6	Appendix: Pro R21 Certificate issued by Zigbee Alliance .....	84

## 1 Overview

### 1.1 Zigbee brief

#### 1.1.1 Device type

Zigbee 3.0 supports the device types below:

- ✧ **Router:** Device with routing function
- ✧ **Coordinator:** Router with network management capability
- ✧ **End Device:** Device with low power sleep mode support but without routing function. ED cannot directly communicate with other nodes in the network, which means its parent node must be used to implement the communication.
- ✧ **GPD (Green Power Device):** Low power passive device. It won't be introduced in this document.

#### 1.1.2 Network type

As per creator type, Zigbee3.0 network, which is one Mesh network, supports two network types: Central network, and Distribute network. The two network types have difference on link key in the network as well as the maintenance of nwk key.

- ✧ **Central network:** Generally it's a network built and managed by Coordinator acting as Trust Center.
- ✧ **Distribute network:** This type of network is built by a node with Router function.

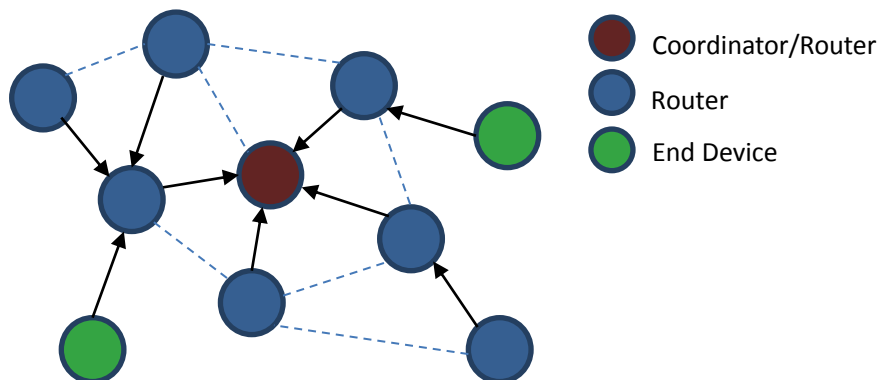


Figure 1 Mesh network

### 1.1.3 Basic terminologies

#### 1) PAN ID: Personal Area Network ID

PAN ID is used to identify an established network. The node that formed this network is responsible for the assignment of PAN ID by means of direct appointing or random generation. But note that active scan must be used to avoid PAN ID collision.

#### 2) Channel: Frequency point/channel

Frequency points allowed by Zigbee to work at (2.4G):  $2405 + (N-11)*5$  (mHz), (N = 11~26)

Once connected to a network, the working mode with single channel is adopted. (If current channel has interference, the Frequency agility mechanism will be adopted to hop to a channel with less interference.)

#### 3) Node: A physical device

Each node has a unique 64-bit MAC address, and after forming or joining in a network, it also has a unique 16-bit short network address in the PAN. The short address can be directly used in subsequent data transfer.

#### 4) Endpoint: Port corresponding to specific Apps

A Node can contain multiple applications (multiple ports). Applications in APP layer are operations based on different ports.

Port number range: 0~255. The Zigbee Alliance has specific rules for the usage of Port number.

- ✧ 0: This Port number is necessary for each Node and it's used for ZDO (Zigbee Device Object).
- ✧ 1~240: These Port numbers are used for user-level applications, i.e. ports that can be randomly used during application development.



- ✧ 241~254: These Port numbers have specialized purpose as per the rule of Zigbee Alliance, e.g. 242 is only used for Green Power.
- ✧ 255: When using the Endpoint 255 to send data, the data will be sent to all application ports via advertising.

## 5) Clusters:

A specific APP (Port) consists of a group of clusters to implement different behaviors. ZCL (Zigbee Cluster Library) specifies a set of behavior rules for clusters.

## 6) Attributes:

A Cluster consists of Attributes and Commands.

## 7) BDB: Base Device Behavior

The BDB specifies a set of rules for the behavior to enable node, form a network, join in a network, manage key, and reset node.

Generally speaking, object operated by the APP layer can be regarded as certain Attribute of certain Cluster on certain Endpoint. The figure below shows the corresponding relationship:

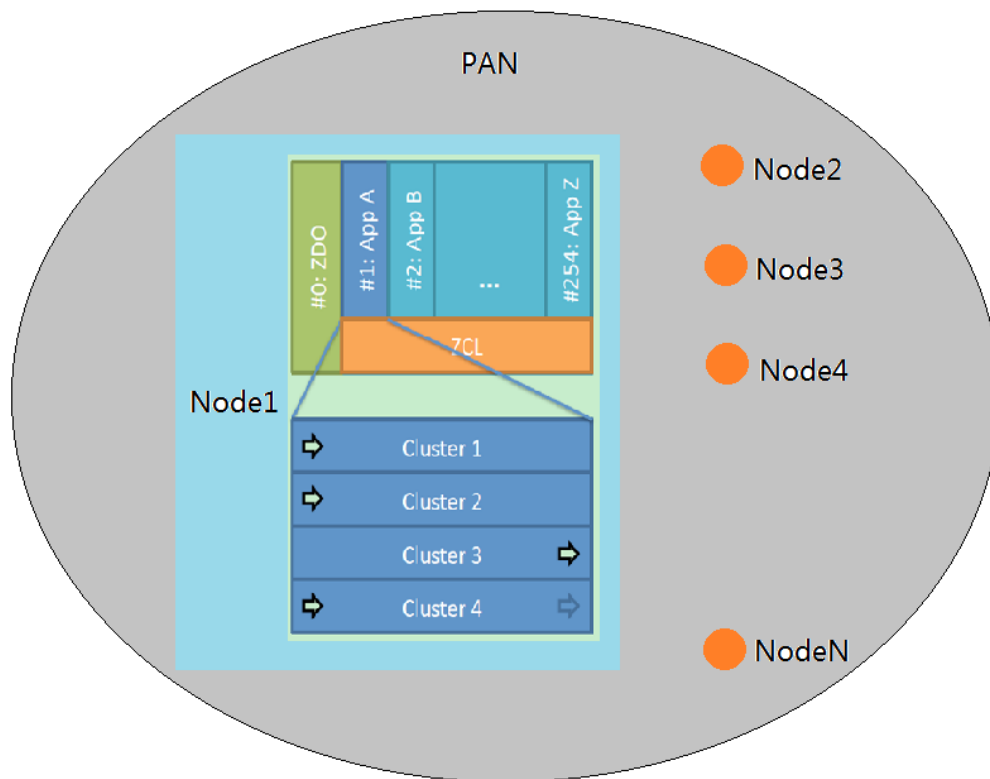


Figure 2 Relationship chart for ED, Cluster and Attribute

## 1.2 Telink Zigbee SDK brief

Telink Zigbee SDK, a set of zigbee protocol stack developed on the basis of Zigbee PRO specification, has passed Zigbee Pro R21 certification of Zigbee Alliance Platform, supports Pro R22 and comply with zigbee3.0 specification.

## 1.3 Software development environment for Telink Zigbee SDK

### 1) Essential SW tools (Website to download the tools: <http://www.telink-semi.cn/>)

- ✧ Integrated Development Environment: Telink IDE
- ✧ Download and debugging tool: Telink download tool
- ✧ OTA code conversion tool: tl\_ota\_tool

### 2) Auxiliary tool to capture and analyze packet (User can download or purchase the tool as needed.)

- ✧ TI Packet Sniffer
- ✧ Ubiqua

### 3) Software development kit (website to download the SDK: <http://www.telink-semi.cn/>)

- ✧ tl\_zigbee\_sdk\_v3.x.x.x.zip

### 4) Auxiliary control software on PC side (ZGC)

## 1.4 HW platform supported by Telink Zigbee SDK

- ✧ 8269: 8269 EVK Board & 8269 USB Dongle
- ✧ 8258: 8258 EVK Board & 8258 USB Dongle

## 2 Telink SDK Development Environment

### 2.1 Install SDK

#### 2.1.1 Import project

- 1) Install Telink IDE.
- 2) Start the IDE, and successively enter the interface “File” -> “Import” -> “Existing Projects into Workspace”.
- 3) Select the directory containing the “tl\_zigbee\_sdk\_v3.x.x.x”, and then click the “OK” button.

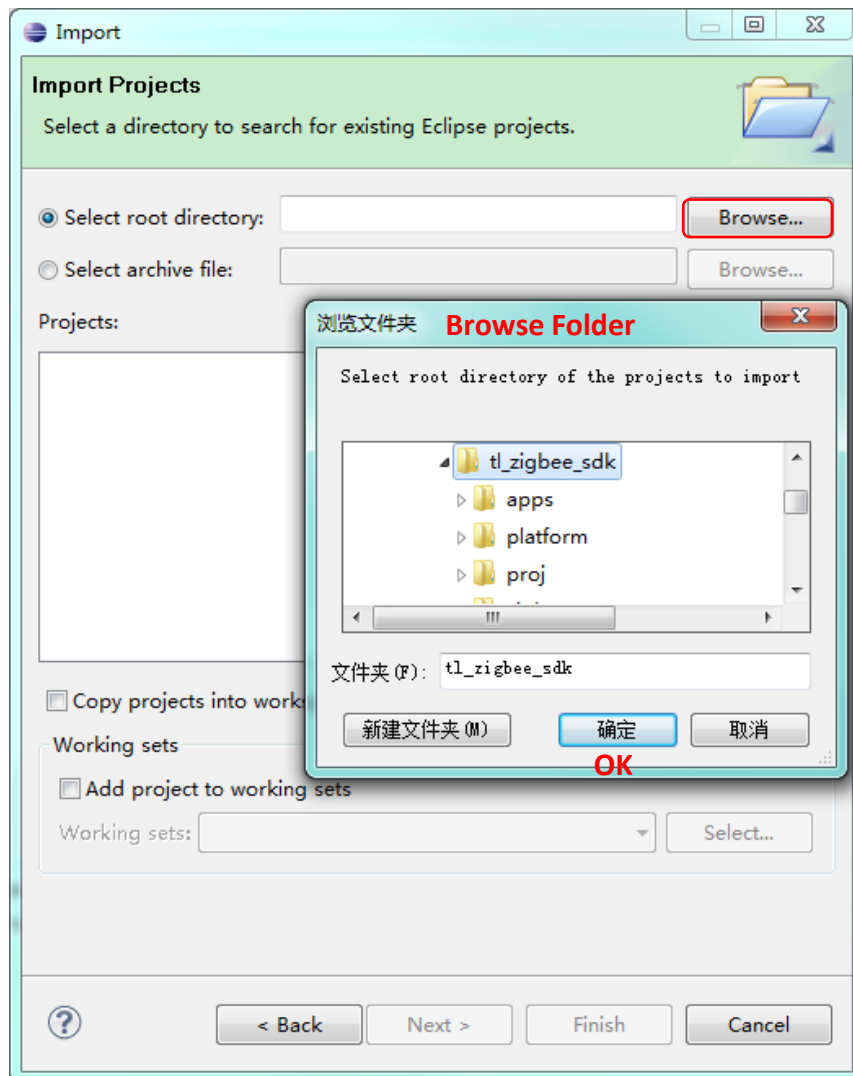


Figure 3 Select project file to be imported

- 4) Click the “Finish” button to finish project importing.

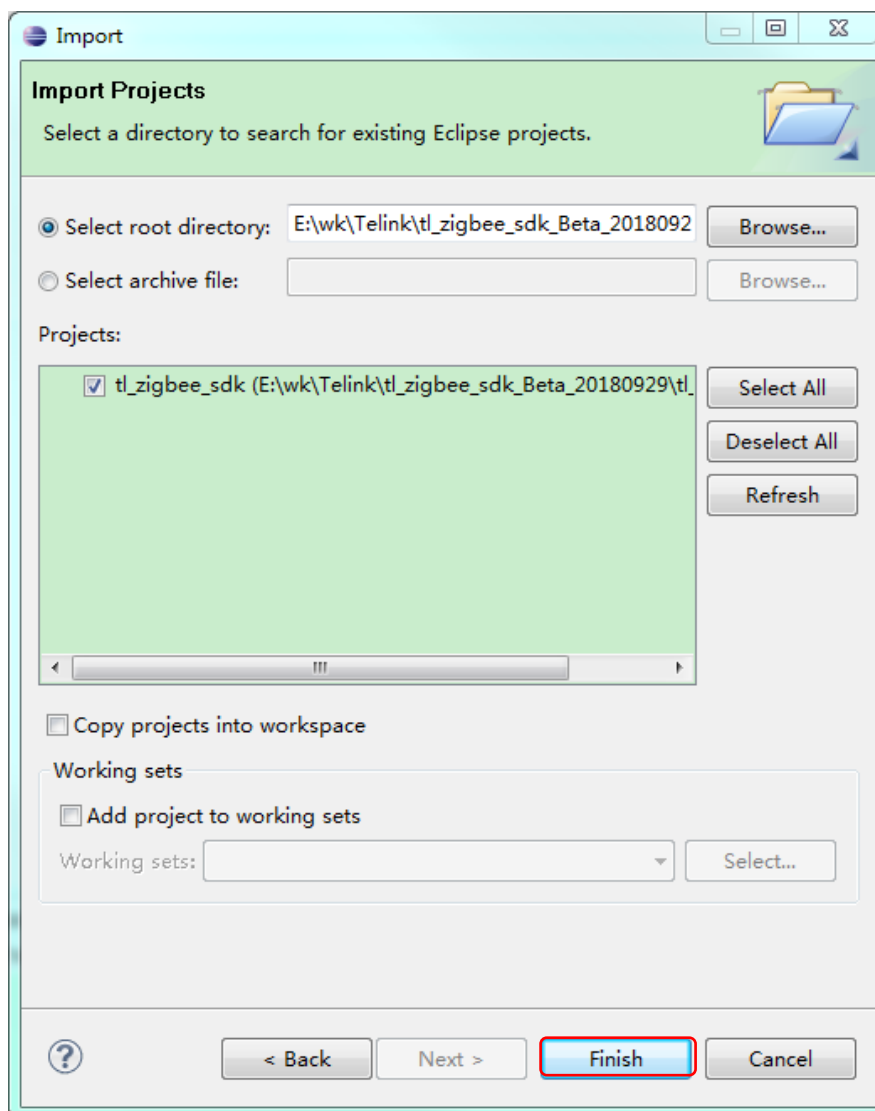


Figure 4 Finish project importing

## 2.1.2 Project directory structure

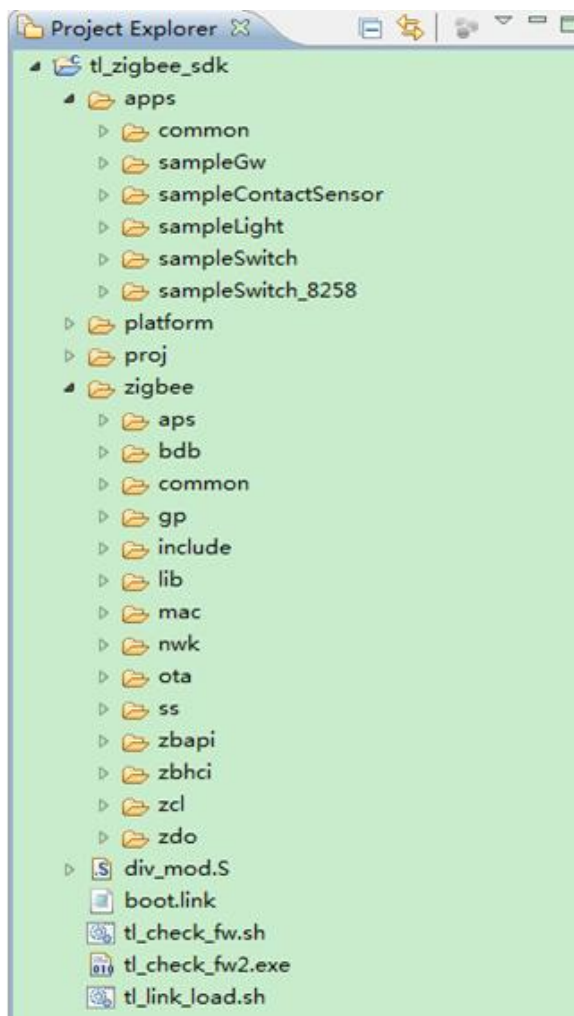
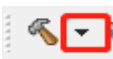


Figure 5 Project directory structure

- ✧ **../apps**: user project directory
- ✧ **../platform**: stack operation platform directory
- ✧ **../proj**: directory for common code part, e.g. driver, buffer management, software timer, string processing
- ✧ **../zigbee**: stack-related directory

### 2.1.3 Compile existing project option

Click the drop-down icon  to show all available compiling options. Select the project sample to be compiled and wait until the compiling is finished.

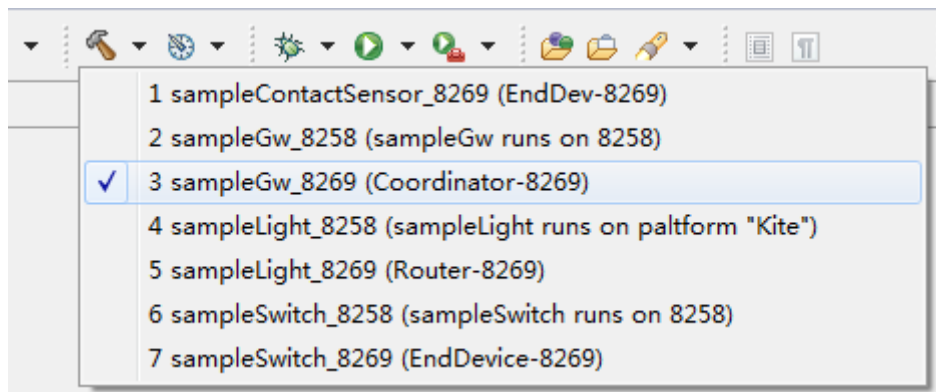
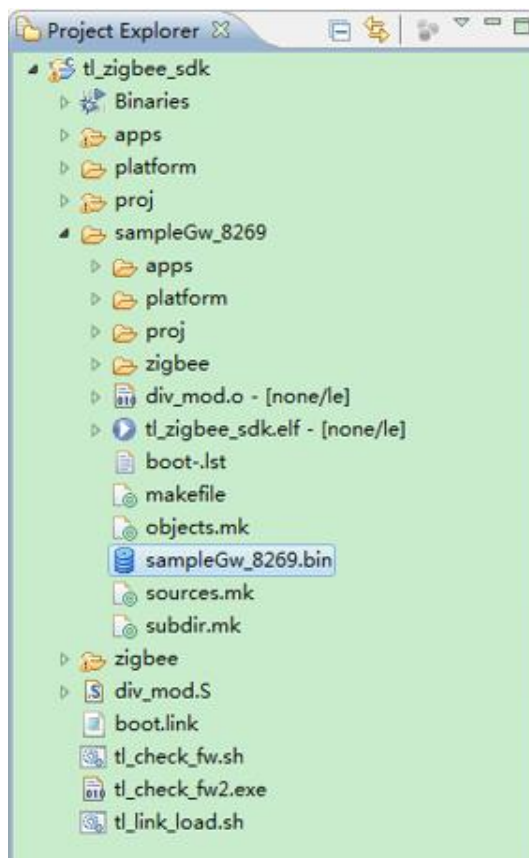


Figure 6 Select and compile project sample

After compiling is finished, the “Project Explorer” window will show the project folder (e.g. sampleGw\_8269) including firmware binary file (e.g. “sampleGw\_8269.bin”).



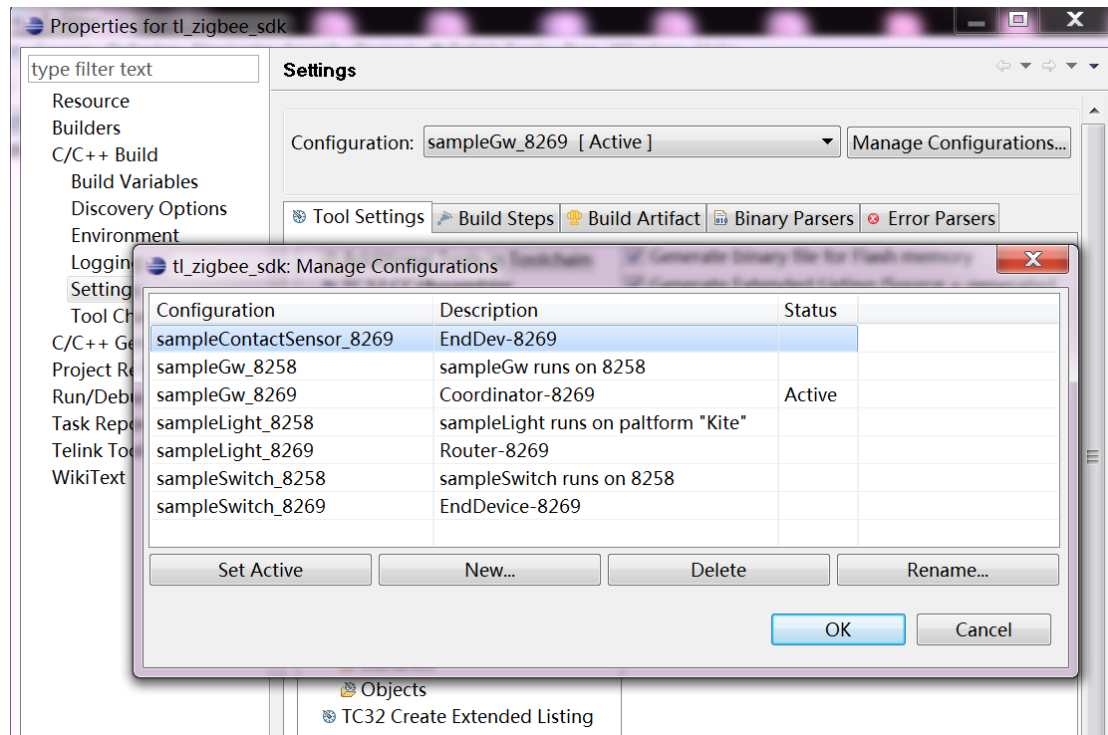
## 2.1.4 Add new project

The SDK only provides some simple project demos as well as compiling options as listed in Figure 6.

User can add application project and compiling option as needed.

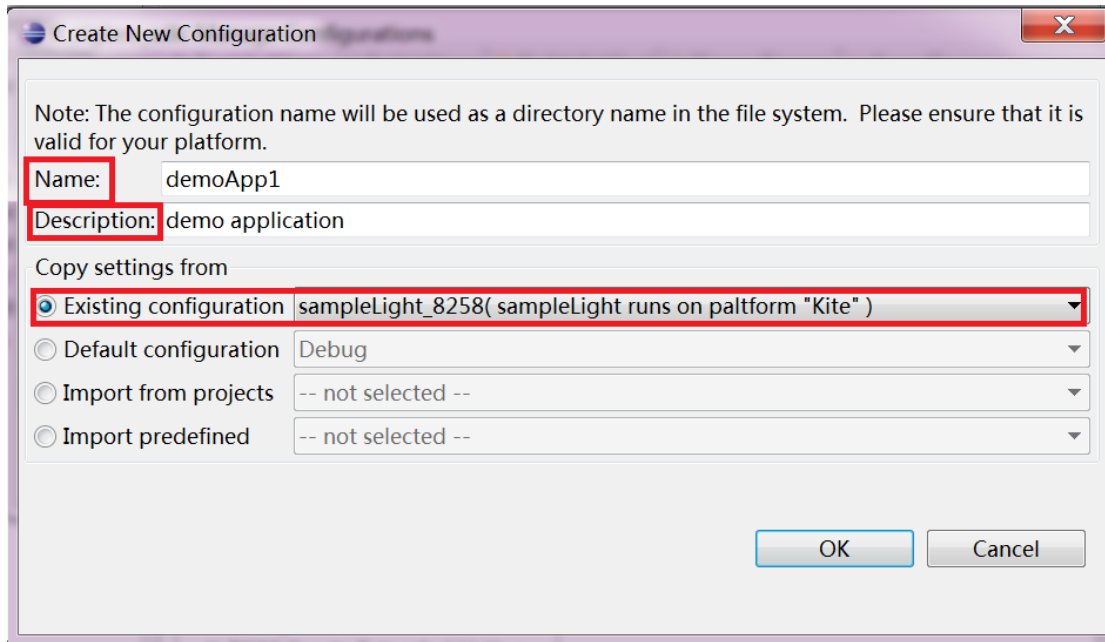
Following shows the operation steps:

- 1) Step 1: Project Explorer -> tl\_zigbee\_sdk -> Properties -> Manage Configurations





2) Step 2: Click the “New” button to open the window below.



- ✧ Name: Project name
- ✧ Description: Brief project description
- ✧ Copy settings from: It's recommended to select “Existing configuration”, so as to simplify the configuration process.

Following shows the guide to select “Existing configuration”:

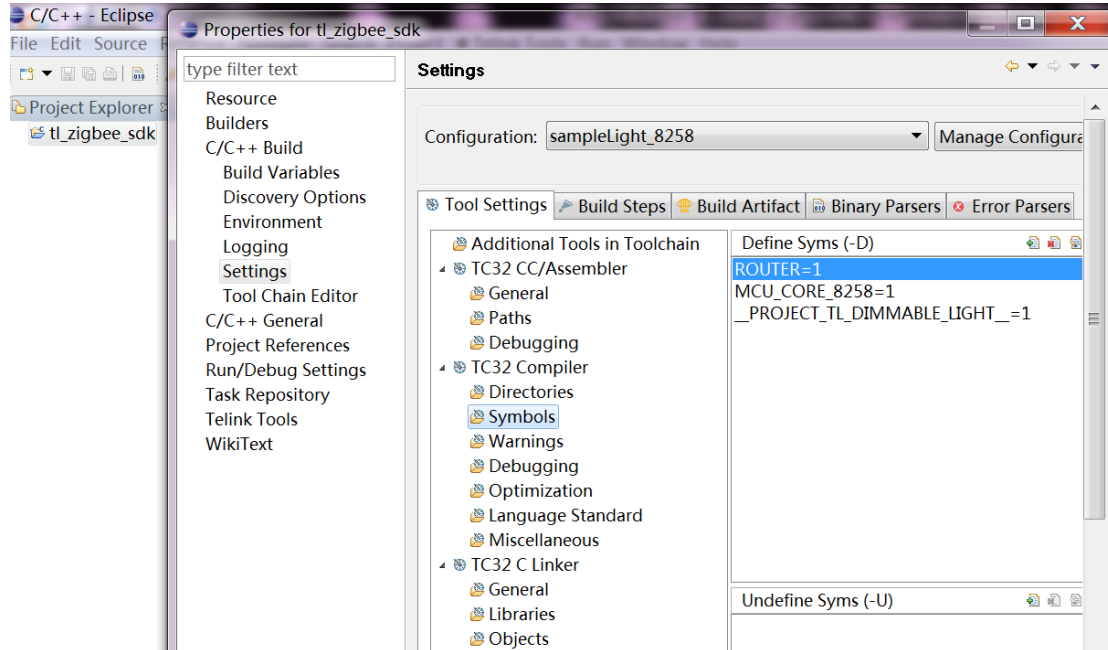
- ✧ Corresponding to 8269 platform or 8258 platform, select “xxx\_8269” and “xxx\_8258” respectively.
- ✧ Corresponding to project of Gateway, Router device, or End Device, select “sampleGw\_xxxx”, “sampleLight\_xxxx”, and “sampleSwitch\_xxxx” respectively.

For example, suppose user needs to develop a light project with routing function based on 8258. As per the guide, the configuration of the project “sampleLight\_8258” should be selected.

For the method to modify project configuration, please refer to section 2.1.5.

## 2.1.5 Introduction to project configuration

Successively enter: Project Explorer -> tl\_zigbee\_sdk -> Properties -> Settings  
(take the project “sampleLight\_8258” for example)



The “Tool Settings” contains some pre-defined configurations for current project:

### 1) Pre-definition for device type

-DROUTER=1: It indicates the project is a device with router function.

Following shows the device setting for coordinator and end device:

-DEND\_DEVICE=1: It indicates the project is a device with end device function.

-DCOORDINATOR=1: It indicates the project is a device with Coordinator function.

### 2) Platform selection

✧ 8269 platform: -DMCU\_CORE\_826x=1, -DCHIP\_8269=1

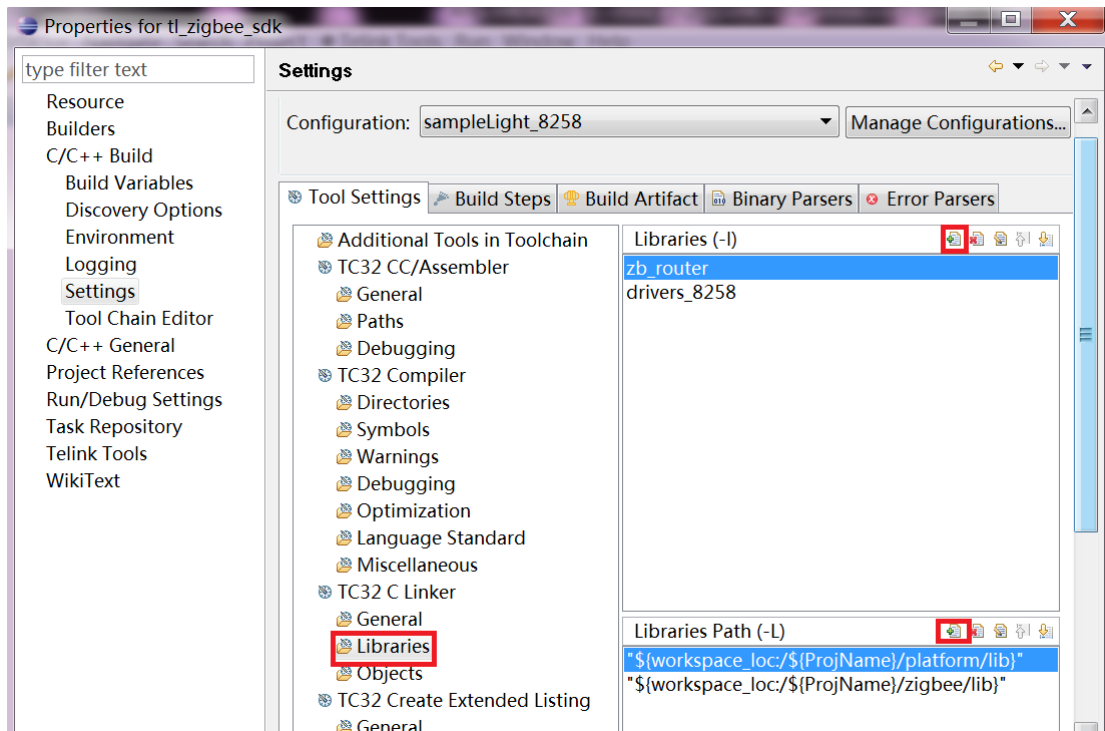
The startup code “cstartup\_826x.S” is contained by the folder “\tl\_zigbee\_sdk\platform\chip\_826x”.

✧ 8258 platform: -DMCU\_CORE\_8258=1

The startup code “cstartup\_8258.S” and “cstartup\_8258\_ed.S” are contained by the folder “\tl\_zigbee\_sdk\platform\chip\_8258”.

- For device with low power function, to enable 32kB RAM retention for fast wakeup, the 8258 will use the “cstartup\_8258\_ed.S” as the startup code.  
(Note: If RAM size to be used exceeds 32kB, user needs to modify the startup code.)
- For device without low power function, the 8258 can directly use the “cstartup\_8258.S” as the startup code to disable retention function.

### 3) Lib file link



Current SDK supports two types of “.lib” file: zigbee stack library, and platform driver library.

✧ zigbee stack library: libzb\_router.a, libzb\_coordinator.a, libzb\_ed.a

Available in the folder “\tl\_zigbee\_sdk\zigbee\lib”.

✧ Platform driver library: libdrivers\_826x.a, libdrivers\_8258.a

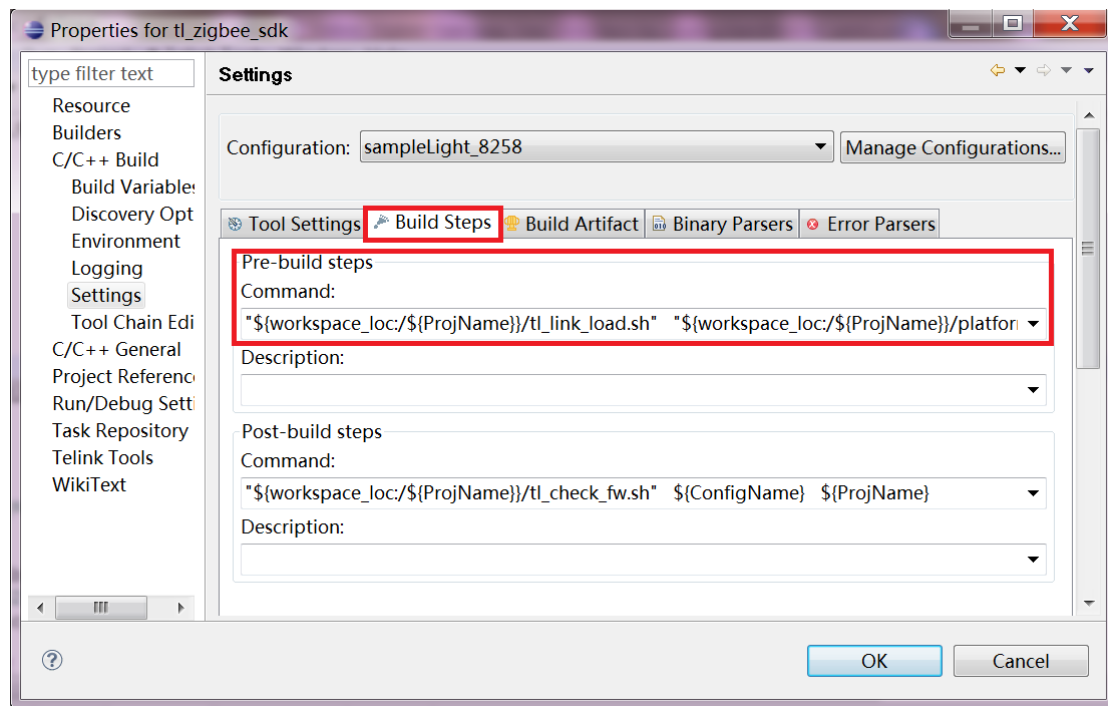
Available in the folder “\tl\_zigbee\_sdk\platform\lib”.

#### 4) Link file

Based on actual application demand, user can adjust linker file as per the selected platform and memory requirement.

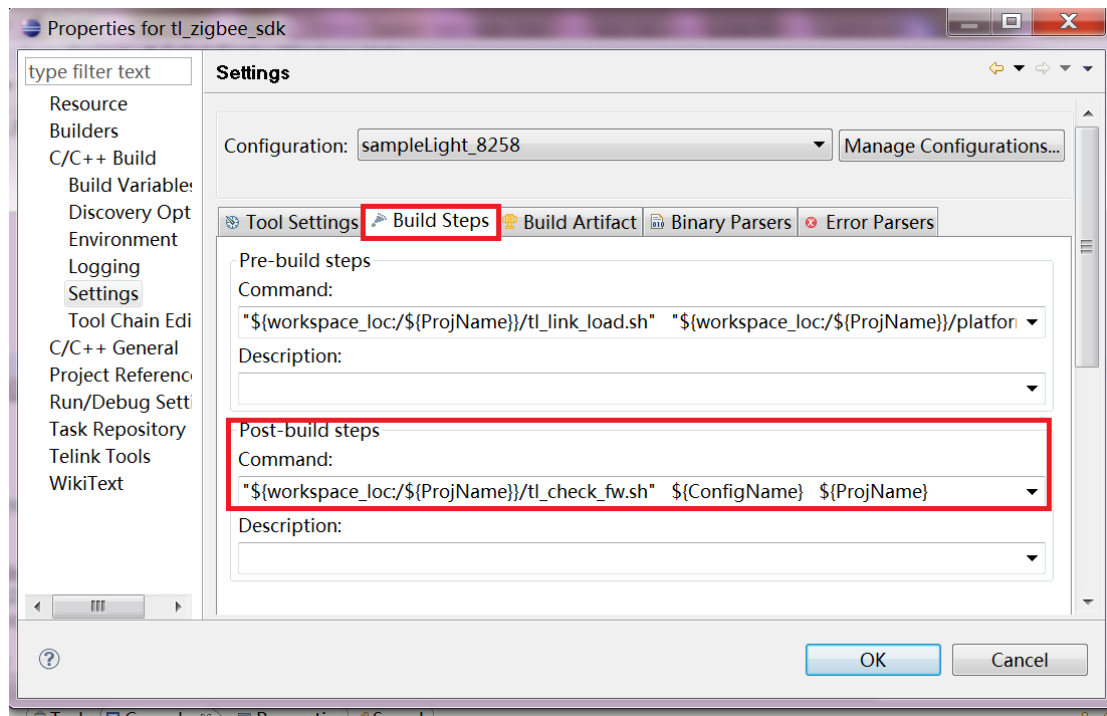
Current SDK supplies default linker file for both 826x platform and 8258 platform: boot\_826x.link, and boot\_8258.link (available in the folder “\tl\_zigbee\_sdk\platform\bootlink”).

As shown below, linker file to be used can be selected by invoking the script “tl\_link\_load.sh” in the folder “\tl\_zigbee\_sdk” during pre-building.



## 5) .image file check

To ensure the reliability of file downloading, the script “tl\_check\_fw.sh” in the folder “\tl\_zigbee\_sdk” can be used to add the check field in the generated image file. During FW downloading or OTA process, the check field will implement matching verification, so as to determine whether to update image.



## 2.2 Image file burning

### 2.2.1 Burning operation

1) Connect hardware:

Use a mini USB cable to connect Telink burning EVK (TLSR8266BR56) with PC USB; the indicating lights on the EVK will blink once to indicate normal connection.

Then use three DuPont cables to connect the VCC, GND and SWM of the EVK with the VCC, GND and SWS of the target board, respectively.



Figure 7 Connection chart for target board, burning EVK and PC

- 2) Use “Telink download tool.exe” on PC side to download target firmware file into target board.
  - a) Select chip part number: e.g. 8258.
  - b) Click “File”->“Open”, and select the firmware to be burned.
  - c) Click the “Erase” button to erase the entire 512kB Flash of the target board by default.
  - d) Click the “Download” button to start firmware burning.

For detailed usage guide of the tool, please refer to the corresponding user guide document.

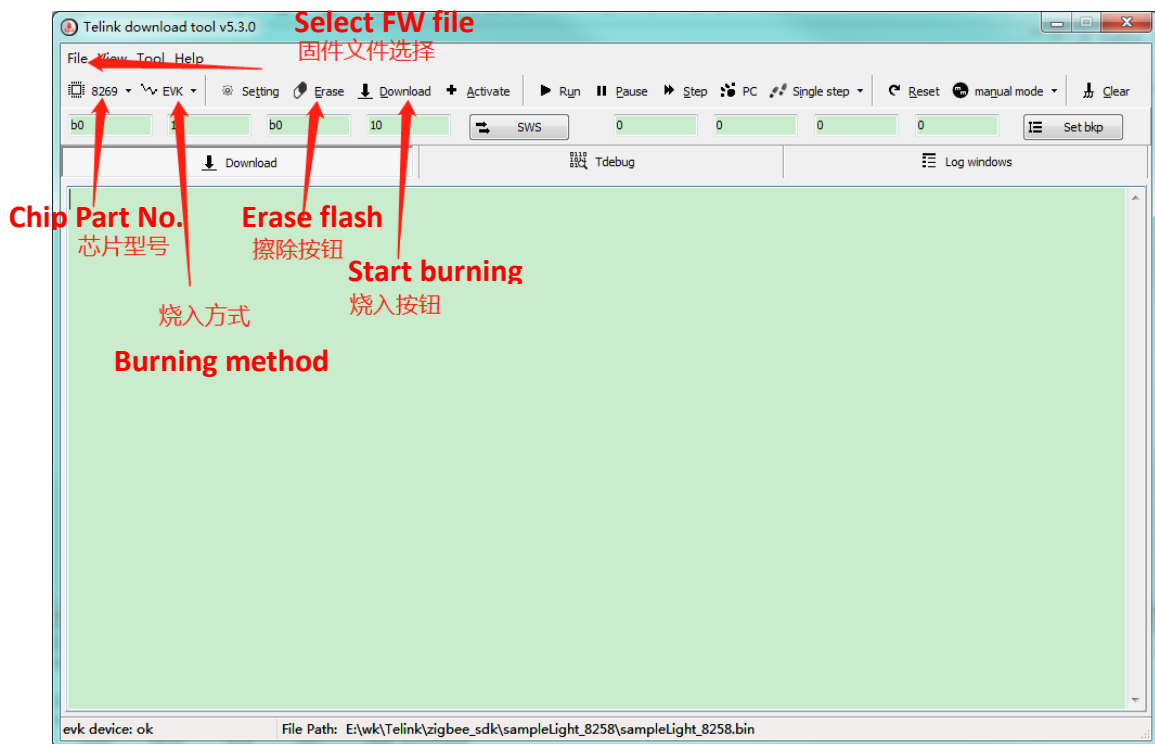


Figure 8 Download firmware

## 2.2.2 Flash address allocation

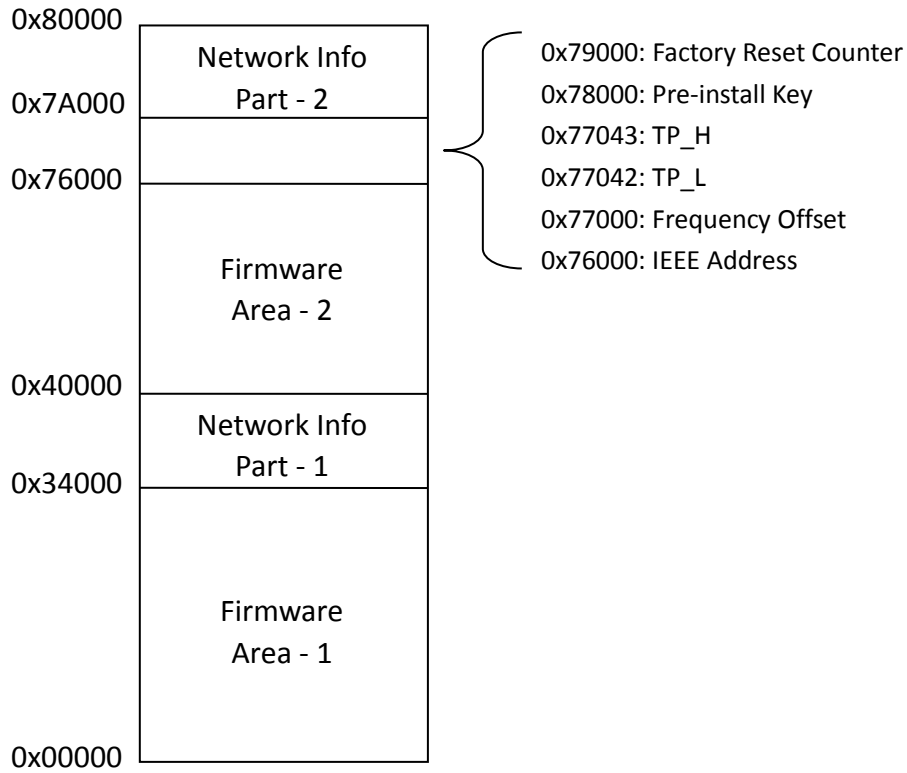


Figure 9 Flash space allocation in Zigbee SDK

### 1) MAC address:

MAC address information is stored in the 8 bytes starting from flash address 0x76000.

After system booting, MAC address info will be checked. If it's 0xFFFFFFFFFFFFFFFF, a MAC address will be generated randomly.

### 2) Network information storage:

After a node joins in the network, network information will be stored into flash, including 48kB space starting from 0x34000 and 24kB space starting from 0x7a000.

After FW update or power cycle, the network info won't be lost.



### 3 Software architecture

This section will further introduce folders and files of Telink Zigbee SDK listed in section **Project directory structure**.

#### 3.1 \tl\_zigbee\_sdk\platform

Current SDK supports 826x platform and 8258 platform. Other HW platforms are to be added in following versions.

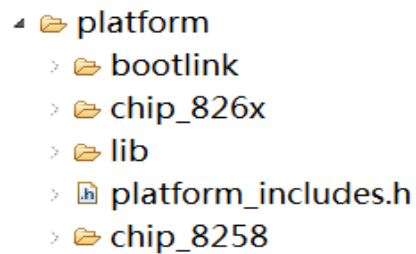


Figure 10 Platform directory

- ✧ \bootlink: linker file
- ✧ \chip\_826x: header file of HW module driver and startup code for 826x platform
- ✧ \chip\_8258: header file of HW module driver and startup code for 8258 platform
- ✧ \lib: driver library file

## 3.2 Directory for common functions

```
└─ proj
   ├── common
   ├── config
   ├── drivers
   ├── os
   └── tl_common.h
```

- ✧ \common: The folder contains some common functions, e.g. functions to process string, link table, and printing.
- ✧ \config: select the user\_config file
- ✧ \drivers: driver abstraction layer
- ✧ \os: task event, buffer management function

### 3.3 Zigbee stack directory

- └─ zigbee
  - > aps
  - > bdb
  - > common
  - > gp
  - > include
  - > lib
  - > mac
  - > nwk
  - > ota
  - > ss
  - > zbapi
  - > zbhci
  - > zcl
  - > zdo

In this directory, most are given in the form of “.lib” file, while source code of ZCL and zbhci closely related to PHY layer and APP layer are public to user.

- ✧ PHY related files: \zigbee\mac\mac\_phy.c, \zigbee\mac\mac\_phy.h, \zigbee\mac\mac\_radio\_826x.h, \zigbee\mac\mac\_radio\_8258.h
- ✧ ZCL: cluster libraries
- ✧ zbhci: Communication interface between Zigbee and Host. Currently supported HCI interface includes UART and USB CDC. It's extensible in the following versions.

### 3.4 Directory architecture for APP layer

All development samples of the APP layer are available in the folder “\tl\_zigbee\_sdk\apps”.

```
└─ apps
   └─ common
   └─ sampleGw
   └─ sampleContactSensor
   └─ sampleLight
   └─ sampleSwitch
   └─ sampleSwitch_8258
```

#### 3.4.1 \common

Common functions shared by projects:

- ✧ factory\_reset.c/.h: Implement device reset via power-down operation
- ✧ module\_test.c: Test each HW or SW module used in the SDK. It's only used for verification during development.
- ✧ pm\_interface.c/.h: Low power management function
- ✧ user\_config.h: Select different application configurations.
- ✧ main.c: Main function entry to implement module initialization as well as scheduling of the whole system. By invoking “user\_init()” in application directory (e.g. sampleGw, sampleLight), corresponding application can be enabled.

#### 3.4.2 \sampleXxxx

User can add his own application project directory, and implement project configuration as per the guide in section **2.1.4 Add new project**.

Application entry function “user\_init()” can be used to initialize corresponding HW and SW function as per user's requirement (e.g. GPIO/PWM/UART/USB function on HW, whether to enable HCI interface on SW), and then enable Zigbee network function.

## 4 Development Guide

### 4.1 Print debugging

#### 4.1.1 UART print

To avoid waster of HW UART resource, UART print function is supplied by using a GPIO to simulate UART TX, and the GPIO used for UART print is also modifiable by user.

Following shows the configuration method.

##### 1) Enable/Mask UART print function

```
#define UART_PRINTF_MODE 1
```

##### 2) Configure GPIO for UART print

```
#define DEBUG_INFO_TX_PIN GPIO_PC4//print
#define PC4_OUTPUT_ENABLE 1
#define PC4_INPUT_ENABLE 0
```

##### 3) Configure baud rate

```
#ifndef MCU_CORE_8258
#define BAUDRATE 1000000//1M
#define BIT_INTERVAL (16*1000*1000 / BAUDRATE)
#else
#define BAUDRATE 2000000//2M
#define BIT_INTERVAL (CLOCK_SYS_CLOCK_HZ / BAUDRATE)
#endif
```

### 4.1.2 USB Print

USB print function is supplied for platforms with HW USB support by virtue of Telink BDT tool.

Following shows the configuration method.

#### 1) Enable/Mask USB print function

```
#define USB_PRINTF_MODE 1
```

#### 2) Configure USB

```
#if USB_PRINTF_MODE
/* Enable USB Port */
gpio_set_func(GPIO_PA5, AS_USB);
gpio_set_func(GPIO_PA6, AS_USB);
usb_dp_pullup_en(TRUE);
#endif
```

## 4.2 Select target board

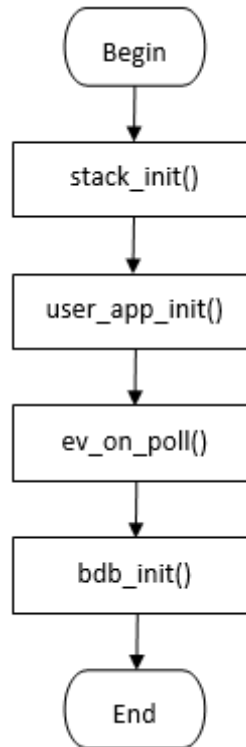
Demos in Telink Zigbee SDK are executed on EVK board or USB Dongle. User can change target board in the “app\_cfg.h”.

```
#define BOARD BOARD_8258_DONGLE
```

After the target board is selected, the configuration corresponding to the selection will be loaded during pre-compiling. The configuration file, “board\_xx.h” in the project folder, implements pre-definition based on IO usage of different HW boards. (Note: The configuration won’t take effect unless the “gpio\_init()” function is invoked during system initialization.)

## 4.3 Development flow

### 4.3.1 APP-layer initialization (user\_init)



#### 1) `stack_init()`

- ✧ Invoke the “`zb_init()`” to initialize the zigbee protocol stack.
- ✧ Invoke the register function “`zb_zdoCbRegister()`” to obtain related information during network operation.

## 2) user\_app\_init()

- ✧ Invoke the function “af\_endpointRegister()” to register basic description information of device. The registry is shown as below:

```
const af_simple_descriptor_t simpleDesc =
{
    /* Application profile identifier */
    /* Device identifier */
    /* Endpoint */
    /* Application device version */
    /* Reserved */
    /* Application input cluster count */
    /* Application output cluster count */
    /* Application input cluster list */
    /* Application output cluster list */
}
```

- ✧ Invoke the function “zcl\_init()” to register basic command processing supported by the ZCL layer.
- ✧ Invoke the function “zcl\_register()” to register Endpoint, Clusters and Attributes that the APP layer needs to support, as well as corresponding command callback processing functions. The registry is shown as below:

```
zcl_specClusterInfo_t g_sampleClusterList =
{
    {
        /* Cluster ID */
        /* Attributes Total Number */
        /* Attribute Table */
        /* ZCL Register Function Name */
        /* Cluster Command Callback Function */
    },
    .
    .
    .
}
```



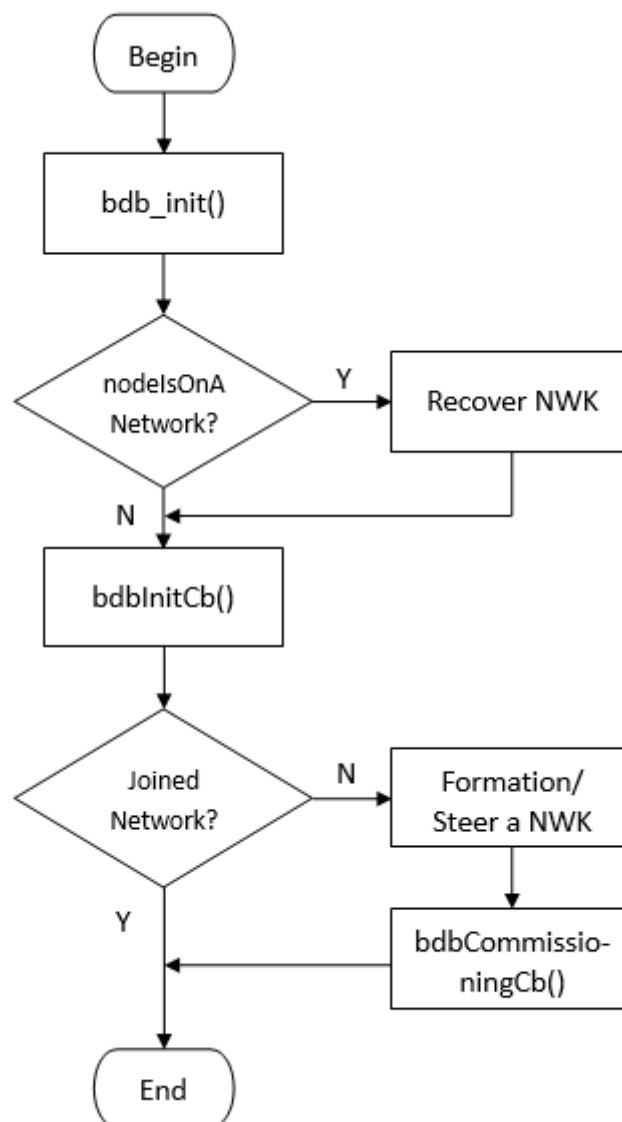
### 3) ev\_on\_poll()

Register user polling event.

### 4) bdb\_init()

Initialize the behavior for device to join in a network. After initialization is completed, callback the function “zbdemo\_bdbInitCb()”.

#### 4.3.2 BDB initialization (bdb\_init)



After BDB initialization, the callback function “bdbInitCb” registered by the APP layer will be invoked, and the upper-layer user will determine subsequent BDB Commissioning behavior.

### 4.3.3 BDB Commissioning

BDB Commissioning supports the behaviors below:

- ✧ Network steering: Search and join in a network.
- ✧ Network formation: New device, Router or Coordinator, forms a Distribute network or a Central network respectively.
- ✧ Finding & Binding: Search matched function after joining a network and bind the method to send data.
- ✧ Touch Link: Connected to a network by touch link.

After executing BDB Commissioning, the system will invoke the function “bdbCommissioningCb” to return the commissioning result to user, so that the user can determine subsequent operation.

For example: If an End Device returns a state of BDB\_COMMISSION\_STA\_NO\_NETWORK after executing BDB Commissioning, user can determine whether to enter suspend or continue attempting to search and join a network.

### 4.3.4 Data interaction

After a network is established successfully, nodes within the same network can transmit and receive data with each other.

User can use standard ZCL to implement data interaction, or directly adopt related APIs to customize data Tx and Rx processing.

#### 4.3.4.1 Use ZCL layer to implement data interaction

The Zigbee Alliance specifies basic commands of ZCL standard, including Read, Write, Report, and Configure Report, as well as processing for Clusters, Attributes and Commands for different applications.

Telink Zigbee SDK implements most part of ZCL, while user can directly invoke the source code or extend as per his own need. The Demos in the SDK adopt the ZCL method to interact data (Please refer to “ZigBee Cluster Library Specification”).

Take the “sampleLight” to illustrate the implementation method:

- 1) Register the callback function for ZCL Rx processing, i.e. “zcl\_rx\_handler”.

```
/* Register endPoint */
af_endpointRegister(SAMPLE_LIGHT_ENDPOINT,
                    (af_simple_descriptor_t*)&sampleLight_simpleDesc,
                    zcl_rx_handler,
                    NULL);
```

- 2) Register the callback function for ZCL basic command processing, i.e. “zclProcessIncomingMsg”.

```
/* Register Incoming ZCL Foundation command/response messages */
zcl_init(sampleLight_zclProcessIncomingMsg);
```

- 3) Register the callback function table for command processing corresponding to Clusters that user needs to support, i.e. the registry referred to in section 4.3.1.

```
/* Register ZCL specific cluster information */
zcl_register (SAMPLE_LIGHT_ENDPOINT,
              SAMPLELIGHT_CB_CLUSTER_NUM,
              g_sampleLightClusterList);
```

#### 4.3.4.2 Use AF layer to implement data interaction

If not using standard ZCL method, user can also register customized Rx processing function, and invoke the function “af\_dataSend” supplied by the AF layer to send data to other device. Indeed the ZCL specification is the secondary encapsulation for the AF-layer data processing.

Following shows the implementation method:

- 1) Register the callback function for data Rx processing, i.e. “user\_rx\_handler”.

```
/* Register endPoint */
af_endpointRegister(SAMPLE_LIGHT_ENDPOINT,
                   (af_simple_descriptor_t*)&sampleLight_simpleDesc,
                   user_rx_handler,
                   NULL);
```

- 2) Invoke the “af\_dataSend” to send data.

```
u8 af_dataSend( u8 srcEp,
                epInfo_t *pDstEpInfo,
                u16 clusterId,
                u8 cmdPldLen,
                u8 *cmdPld,
                u8 *seqNo);
```

#### 4.3.5 Dynamic memory management

Telink Zigbee SDK supplies the interfaces for dynamic memory allocation and release which user can directly invoke during development.

- 1) Apply for memory (“size” can support up to 142 bytes)

```
u8 *ev_buf_allocate(u16 size);
```

- 2) Release memory:

```
buf_sts_t ev_buf_free(u8 *pBuf);
```

#### 4.3.6 Task scheduling

By reasonable usage of task scheduling, internal stack resource waste of chip can be avoided effectively.

```
/**
 * @brief      push a task to task list
 *
 * @param[in]  func - the callback of the event
 *
 * @param[in]  arg - the parameter to the callback
 *
 * @return     the status
 */
u8 tl_zbTaskPost(tl_zb_callback_t func, void *arg);
#define TL_SCHEDULE_TASK  tl_zbTaskPost
```

#### 4.3.7 Software timer task

To help user implement timer task without the requirement of high precision, Telink Zigbee SDK supplies the interfaces to register and cancel a software timer task.

#### 4.3.7.1 Register SW timer task

```
/**
 * @brief      push timer task to task list
 *
 * @param[in]  func - the callback of the timer event
 *
 * @param[in]  arg - the parameter to the callback
 *
 * @param      cycle - the timer interval
 *
 * @return     the status
 */
ev_time_event_t *tl_zbTimerTaskPost(ev_timer_callback_t func,
                                     void *arg,
                                     u32 t_us);

#define TL_ZB_TIMER_SCHEDULE      tl_zbTimerTaskPost
```

#### 4.3.7.2 Cancel task

```
/**
 * @brief      cancel timer task from task list
 *
 * @param[in]  te - the pointer to the timer event pointer
 *
 * @return     the status
 */
u8 tl_zbTimerTaskCancel(ev_time_event_t **te);

#define TL_ZB_TIMER_CANCEL      tl_zbTimerTaskCancel
```

#### 4.3.7.3 Usage example

When the “VK\_SW1” button is pressed, a 10s timer task is started or canceled. When the timing duration expires, an advertising On/Off Toggle command is executed. After the execution ends, exit the task.

```
ev_time_event_t *brc_toggleEvt = NULL;

s32 brc_toggleCb(void *arg)
{
    epInfo_t dstEpInfo;
    TL_SETSTRUCTCONTENT(dstEpInfo, 0);

    dstEpInfo.dstAddrMode = APS_SHORT_DSTADDR_WITHEP;
    dstEpInfo.dstEp = SAMPLE_GW_ENDPOINT;
    dstEpInfo.dstAddr.shortAddr = 0xffff;
    dstEpInfo.profileId = HA_PROFILE_ID;
    dstEpInfo.txOptions = 0;
    dstEpInfo.radius = 0;

    zcl_onOff_toggleCmd(SAMPLE_GW_ENDPOINT, &dstEpInfo, FALSE);

    brc_toggleEvt = NULL;
    return -1;
}

void buttonShortPressed(u8 btNum)
{
    if(btNum == VK_SW1){
        if(!brc_toggleEvt){
            brc_toggleEvt = TL_ZB_TIMER_SCHEDULE(brc_toggleCb,
                                                    NULL,
                                                    10 * 1000 * 1000);
        }else{
            TL_ZB_TIMER_CANCEL(&brc_toggleEvt);
        }
    }
}
```

#### 4.3.7.4 Notes

- 1) Three cases for the return value of timer task callback function:
  - a) If the return value is less than 0, the task will be deleted automatically after execution.
  - b) If the return value equals 0, the “t\_us” specified when registering the task will be used as the timing period.
  - c) If the return value is larger than 0, the return value will be used as new timing period (unit: us).

- 2) Maximum timing value

Telink Zigbee SDK supplies a readable System Timer Tick, which is 32-bit long System Timer counter. The timer tick value will be incremented automatically by 1 for each clock cycle.

Since System Timer clock of the 826x and 8x5x are derived from different sources, System Timer for the 8269 and 8258 are 32M and 16M respectively, maximum timing value is also different.

- ✧ For the 8269, the maximum timing value is  $(1/16)\mu s \times (2^{32}) = 268s$ , which means System Timer Tick loops with the period of 268s.
- ✧ For the 8258, the maximum timing value is  $(1/32)\mu s \times (2^{32}) = 134s$ , which means System Timer Tick loops with the period of 134s.

Thus, time set in user timer task must be lower than the maximum timing value. If the timing value needs to exceed the maximum allowed value, the nesting method should be adopted to do counting processing in callback function.



#### 4.3.8 Sleep and wakeup

Telink Zigbee SDK supplies related low power management function, and it supports wakeup triggered by key press and timer. The “sampleSwitch” and “sampleSwitch\_8258” provide the usage method.

##### 1) Configure wakeup pin

```
/**
 * @brief Definition for wakeup source and level for PM
 */
pm_pinCfg_t g_switchPmCfg[] =
{
    {BUTTON1, PM_WAKEUP_LEVEL},
    {BUTTON2, PM_WAKEUP_LEVEL},
};

pm_wakeupPinConfig(g_switchPmCfg,
                   sizeof(g_switchPmCfg)/sizeof(pm_pinCfg_t));
```

##### 2) Select sleep mode and wakeup source

Suspend mode:

```
pm_suspendEnter(PLATFORM_WAKEUP_TIMER | PLATFORM_WAKEUP_PAD,
                zb_getPollRate());
```

Deep sleep mode:

```
pm_deepSleepEnter(PLATFORM_WAKEUP_TIMER|PLATFORM_WAKEUP_PAD,
                  zb_getPollRate());
```

#### 4.3.9 GPIO interrupt

The SDK also provides GPIO interrupt handler functions for users.

Both the 826x and 8x5x series can support up to three external GPIO interrupts simultaneously with interrupt modes of GPIO\_IRQ\_MODE, GPIO\_IRQ\_RISC0\_MODE and GPIO\_IRQ\_RISC1\_MODE.

##### 1) Register interrupt service function

```
int drv_gpio_irq_conf(gpio_irq_mode_t mode,  
                     unsigned int pin,  
                     enum gpio_pol polarity,  
                     irq_callback gpio_irq_callback);
```

##### 2) Enable interrupt pins

```
int drv_gpio_irq_en(unsigned int pin);  
int drv_gpio_irq_risc0_en(unsigned int pin);  
int drv_gpio_irq_risc1_en(unsigned int pin);
```

#### 4.3.10 Configure network parameters

User can modify network parameter configurations in the “/zigbee/common/zb\_config.c” as per his own need.

##### 1) TL\_ZB\_NWK\_ADDR\_MAP\_NUM

The maximum number of address mapping tables supported by node.

##### 2) TL\_ZB\_NEIGHBOR\_TABLE\_NUM

The number of neighbor tables supported by node.

##### 3) ROUTING\_TABLE\_NUM

The number of AODV routing tables supported by node.

4) `NWK_ROUTE_RECORD_TABLE_NUM`

The number of MTO routing record tables supported by node.

5) `APS_BINDING_TABLE_NUM`

The number of binding tables supported by node.

6) `APS_GROUP_TABLE_NUM`

The number of groups supported by node.

7) `const u8 nwkKeyDefalut[]`

Default value of network key.

8) `const tl_zb_mac_pib macPibDefault`

Basic parameter information of the MAC layer.

9) `const nwk_nib_t nib_default`

Basic parameter information of the NWK layer.

## 4.4 Common APIs

### 4.4.1 Network Management

1) `zb_mgmtPermitJoinReq`

The control command to permit joining in a network.

2) `zb_mgmtLeaveReq`

The control command to leave a network.

3) `zb_mgmtLqiReq`

Obtain the LQI information of the neighbor device associated with the target device.

4) zb\_mgmtBindReq

Search for binding table of the target device.

5) zb\_mgmtNwkUpdateReq

Update network parameters or request information of the operation environment.

#### **4.4.2 Bind Management**

1) zb\_zdoBindUnbindReq

Binding and unbinding request command.

#### **4.4.3 ZDP commands**

1) zb\_zdoNwkAddrReq

The command to request for the target short address.

2) zb\_zdoIeeeAddrReq

The command to request for the target IEEE address.

3) zb\_zdoSimpleDescReq

The command to request for simple descriptor of the target.

4) zb\_zdoNodeDescReq

The command to request for node description information of the target.

5) zb\_zdoPowerDescReq

The command to request for power description information of the target.

6) `zb_zdoActiveEpReq`

The command to request for information of the target's active port.

7) `zb_zdoMatchDescReq`

The command to request for matching description information of the target.

#### 4.4.4 Other commands

1) `ota_queryStart`

Start OTA query function (unit: minute).

2) `zb_setPollRate`

Set polling rate of End Device (unit: ms).

3) `zb_factoryReset`

Factory reset setting. Leave command will be advertised before reset.

4) `zb_isDeviceJoinedNwk`

Check whether a node has already joined a network.

5) `zb_apsExtPanidSet`

Set Extend Pan ID.

6) `zb_getMacAssocPermit`

Obtain local permission state for device to join in a network.

7) `zb_joinAFixedNetwork`

Directly join in certain network.

## 4.5 OTA

The 826x and 8x5x family support flash multi-address booting from 0x00000 and 0x40000. This features is used by Telink Zigbee SDK to implement OTA function.

As shown in **section 2.2.2**, two areas area-1 and area-2 are assigned for firmware, and firmware size should not exceed 208k.

Suppose the firmware in the area-1 (area-2) is being executed, during one OTA upgrade, new firmware data will be stored into the area-2 (area-1). After finishing the OTA process and passing the verification, the system is rebooted to execute the firmware in the area-2 (area-1).

### 4.5.1 OTA initialization

```
typedef enum{
    OTA_TYPE_CLIENT,
    OTA_TYPE_SERVER
}ota_type_e;

void ota_init( ota_type_e type,
              af_simple_descriptor_t *simpleDesc,
              ota_preamble_t *otaPreamble, //current running fw info
              ota_callBack_t *cb );
```

OTA device includes Server and Client. User needs to notice service type during OTA initialization.

Generally, the device whose firmware is to be upgraded is the Client, while the device which supplies new firmware is the Server.

#### 4.5.2 OTA Server

The OTA Server needs to write the new firmware prepared for the Client into another Firmware area for OTA use.

For example: If the Server's firmware is in Firmware area-1, the OTA image of the target device including OTA Header (please see "Zigbee Cluster Library Specification") can be stored in Firmware area-2 temporarily. For the method to generate OTA image, please refer to Telink Zigbee Demo User Guide.

After the OTA Server invokes the function "ota\_init", the information of the OTA image will be loaded automatically.

#### 4.5.3 OTA Client

```
void ota_queryStart(u8 period);
```

Set OTA query period for OTA Client (unit: minute).

After the OTA Client invokes the function "ota\_init", the previous OTA progress information will be reloaded automatically. If joining in a network successfully, it will continue the previous OTA request after the "period" duration expires.

## 5 HCI Interface for Extended Functions

Telink Zigbee SDK supplies HCI interface to simplify extended application development for user. For the implementation details, please refer to “./zigbee/zbhci”.

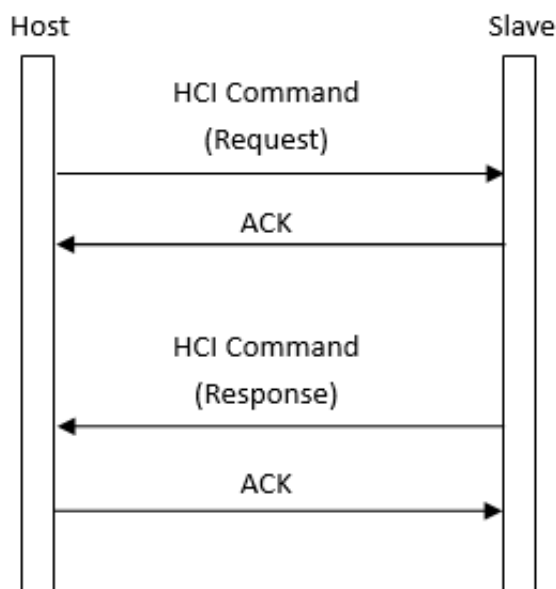
Current HCI interface supports three methods including UART, USB CDC and USB Print, corresponding to the macros below:

UART	→	#define ZBHCI_UART	1
USB CDC	→	#define ZBHCI_USB_CDC	1
USB Print	→	#define ZBHCI_USB_PRINT	1

To use certain method, user only needs to set the corresponding macro to 1.

For example, if the demo “sampleGW” needs to use the HCI function of the UART method, please set the macro “ZBHCI\_UART” to 1.

### 5.1 Control flow chart





## 5.2 Command frame format

Header	Message Type	Message Length	Checksum	Payload	Tail
1 Byte	2 Bytes	2 Bytes	1 Byte	Variable	1 Byte

Field	Description
Header	Header flag, shall be 0x55.
Message Type	Message Type, big-endian.
Message Length	Payload length, big-endian.
Checksum	The checksum.
Payload	Payload.
Tail	Tail flag, shall be 0xAA.

Note: The big-endian mode is for all the fields of the HCI commands.

## 5.3 Acknowledge format

### 5.3.1 Message Type

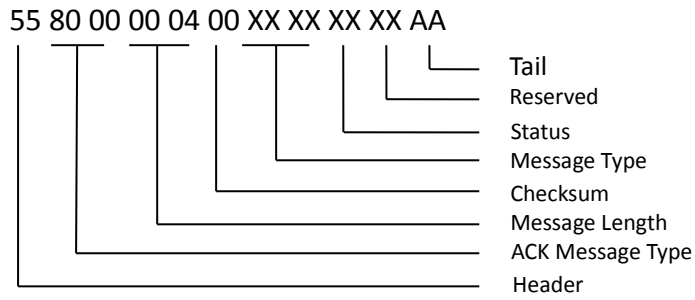
Message Type	Value
ZBHCI_CMD_ACKNOWLEDGE	0x8000

### 5.3.2 Payload

Message Type	Status	Reserved
2 Bytes	1 Byte	1 Byte

Name	Description
Message Type	HCI command message type.
Status	0 = Success; 1 = Wrong parameter; 2 = Unsupported command; 3 = Busy; 4 = No memory.
Reserved	0

Example:



## 5.4 BDB command

### 5.4.1 Message Type

Message Type	Value
ZBHCI_CMD_BDB_COMMISSION_FORMATION	0x0001
ZBHCI_CMD_BDB_COMMISSION_STEER	0x0002
ZBHCI_CMD_BDB_COMMISSION_TOUCHLINK	0x0003
ZBHCI_CMD_BDB_COMMISSION_FINDBIND	0x0004
ZBHCI_CMD_BDB_FACTORY_RESET	0x0005
ZBHCI_CMD_BDB_PRE_INSTALL_CODE	0x0006
ZBHCI_CMD_BDB_CHANNEL_SET	0x0007

### 5.4.2 Payload

#### 1) ZBHCI\_CMD\_BDB\_COMMISSION\_FORMATION

No payload.

Example: 55 00 01 00 00 00 AA

#### 2) ZBHCI\_CMD\_BDB\_COMMISSION\_STEER

No payload.

Example: 55 00 02 00 00 00 AA

### 3) ZBHCI\_CMD\_BDB\_COMMISSION\_TOUCHLINK

Role
1 Byte

Name	Description
Role	1 = Touch link initiator; 2 = Touch link target.

Example: 55 00 03 00 01 00 XX AA

### 4) ZBHCI\_CMD\_BDB\_COMMISSION\_FINDBIND

Role
1 Byte

Name	Description
Role	1 = Find & Bind initiator; 2 = Find & Bind target.

Example: 55 00 04 00 01 00 XX AA

### 5) ZBHCI\_CMD\_BDB\_FACTORY\_RESET

No payload.

Example: 55 00 05 00 00 00 AA

### 6) ZBHCI\_CMD\_BDB\_PRE\_INSTALL\_CODE

devAddr	uniqueLinkKey
8 Bytes	16 Bytes

Name	Description
devAddr	The IEEE address of the pre-configured device.
uniqueLinkKey	The link key for the device.

Example: 55 00 06 00 18 00 XX[8 Bytes] XX[16 Bytes] AA

## 7) ZBHCI\_CMD\_BDB\_CHANNEL\_SET

<b>channelIdx</b>
1 Byte

Name	Description
channelIdx	11 ~ 26

Example: 55 00 07 00 01 00 XX AA

## 5.5 Network management command

### 5.5.1 Message Type (Host)

Message Type	Value
ZBHCI_CMD_DISCOVERY_NWK_ADDR_REQ	0x0010
ZBHCI_CMD_DISCOVERY_IEEE_ADDR_REQ	0x0011
ZBHCI_CMD_DISCOVERY_NODE_DESC_REQ	0x0012
ZBHCI_CMD_DISCOVERY_SIMPLE_DESC_REQ	0x0013
ZBHCI_CMD_DISCOVERY_MATCH_DESC_REQ	0x0014
ZBHCI_CMD_DISCOVERY_ACTIVE_EP_REQ	0x0015
ZBHCI_CMD_DISCOVERY_LEAVE_REQ	0x0016
ZBHCI_CMD_BIND_REQ	0x0020
ZBHCI_CMD_UNBIND_REQ	0x0021
ZBHCI_CMD_MGMT_LQI_REQ	0x0030
ZBHCI_CMD_MGMT_BIND_REQ	0x0031
ZBHCI_CMD_MGMT_LEAVE_REQ	0x0032
ZBHCI_CMD_MGMT_DIRECT_JOIN_REQ	0x0033
ZBHCI_CMD_MGMT_PERMIT_JOIN_REQ	0x0034
ZBHCI_CMD_MGMT_NWK_UPDATE_REQ	0x0035
ZBHCI_CMD_NODES_JOINED_GET_REQ	0x0040
ZBHCI_CMD_NODES_TOGGLE_TEST_REQ	0x0041

## 5.5.2 Payload (Host)

### 5.5.2.1 ZBHCI\_CMD\_DISCOVERY\_NWK\_ADDR\_REQ

<b>dstAddr</b>	<b>ieeeAddr</b>	<b>reqType</b>	<b>startIdx</b>
2 Bytes	8 Bytes	1 Byte	1 Byte

<b>Name</b>	<b>Description</b>
dstAddr	The destination address to which this command will be sent, shall be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.
ieeeAddr	The IEEE address to be matched by the remote device.
reqType	Request type for this command: 0x00 – Single device response; 0x01 – Extended response; 0x02 ~ 0xFF – Reserved.
startIdx	If the request type for this command is extended response, the startIdx provides the starting index for the requested elements of the associated device list.

Example: 55 00 10 00 0C 00 XX[2 Bytes] XX[8 Bytes] XX XX AA

### 5.5.2.2 ZBHCI\_CMD\_DISCOVERY\_IEEE\_ADDR\_REQ

<b>dstAddr</b>	<b>nwkAddrOfInterest</b>	<b>reqType</b>	<b>startIdx</b>
2 Bytes	2 Bytes	1 Byte	1 Byte

<b>Name</b>	<b>Description</b>
dstAddr	The destination address to which this command will be sent, shall be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.
nwkAddrOfInterest	The NWK address that is used for IEEE address mapping.
reqType	Request type for this command: 0x00 – Single device response; 0x01 – Extended response; 0x02 ~ 0xFF – Reserved.
startIdx	If the request type for this command is extended response, the startIdx provides the starting index for the requested elements of the associated device list.

Example: 55 00 11 00 06 00 XX[2 Bytes] XX[2 Bytes] XX XX AA

### 5.5.2.3 ZBHCI\_CMD\_DISCOVERY\_NODE\_DESC\_REQ

<b>dstAddr</b>	<b>nwkAddrOfInterest</b>
2 Bytes	2 Bytes

<b>Name</b>	<b>Description</b>
dstAddr	The destination address to which this command will be sent, shall be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.
nwkAddrOfInterest	NWK address of the target.

Example: 55 00 12 00 04 00 XX[2 Bytes] XX[2 Bytes] AA

### 5.5.2.4 ZBHCI\_CMD\_DISCOVERY\_SIMPLE\_DESC\_REQ

<b>dstAddr</b>	<b>nwkAddrOfInterest</b>	<b>endpoint</b>
2 Bytes	2 Bytes	1 Byte

<b>Name</b>	<b>Description</b>
dstAddr	The destination address to which this command will be sent, shall be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.
nwkAddrOfInterest	NWK address of the target.
endpoint	The endpoint on the destination.

Example: 55 00 13 00 05 00 XX[2 Bytes] XX[2 Bytes] XX AA

### 5.5.2.5 ZBHCI\_CMD\_DISCOVERY\_MATCH\_DESC\_REQ

<b>dstAddr</b>	<b>nwkAddrOfInterest</b>	<b>profileID</b>	<b>numInClusters</b>	<b>numOutClusters</b>	<b>clusterList</b>
2 Bytes	2 Bytes	2 Bytes	1 Byte	1 Byte	n Bytes

<b>Name</b>	<b>Description</b>
dstAddr	The destination address to which this command will be sent, shall be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.
nwkAddrOfInterest	NWK address of the target.
profileID	Profile ID to be matched at the destination.
numInClusters	The number of input clusters provided for matching within the inClusterList.
numOutClusters	The number of output clusters provided for matching within the outClusterList.

Name	Description
clusterList	inClusterList + outClusterList. List of input cluster IDs to be used for matching, the inClusterList is the desired list to be matched by the Remote Device (the elements of the inClusterList are the supported output clusters of the Local Device). List of output cluster IDs to be used for matching, the outClusterList is the desired list to be matched by the Remote Device (the elements of the outClusterList are the supported input clusters of the Local Device).

Example:

55 00 14 msgLenH msgLenL 00 XX[2 Bytes] XX[2 Bytes] XX[2 Bytes] XX XX  
XX[n Bytes] AA

#### 5.5.2.6 ZBHCI\_CMD\_DISCOVERY\_ACTIVE\_EP\_REQ

<b>dstAddr</b>	<b>nwkAddrOfInterest</b>
2 Bytes	2 Bytes

Name	Description
dstAddr	The destination address to which this command will be sent, shall be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.
nwkAddrOfInterest	NWK address of the target.

Example: 55 00 15 00 04 00 XX[2 Bytes] XX[2 Bytes] AA

#### 5.5.2.7 ZBHCI\_CMD\_DISCOVERY\_LEAVE\_REQ

<b>devAddr</b>	<b>rejoin</b>	<b>removeChildren</b>
8 Bytes	1 Byte	1 Byte

Name	Description
devAddr	The IEEE address of the device to be removed or NULL if the device removes itself.
rejoin	TRUE if the device is being asked to leave from the current parent and requested to rejoin the network. Otherwise, the parameter has a value of FALSE.
removeChildren	TRUE if the device being asked to leave the network is also being asked to remove its child device, if any. Otherwise, it has a value of FALSE.

Example: 55 00 16 00 0A 00 XX[8 Bytes] XX XX AA

### 5.5.2.8 ZBHCI\_CMD\_BIND\_REQ / ZBHCI\_CMD\_UNBIND\_REQ

srcIEEEAddr	srcEndpoint	clusterID	dstAddrMode	dstAddr	dstEndpoint
8 Bytes	1 Byte	2 Byte	1 Byte	2 or 8 Bytes	0 or 1 Byte

Name	Description
srcIEEEAddr	The IEEE address for the source.
srcEndpoint	The source endpoint for the binding entry.
clusterID	The identifier of the cluster on the source device that is bound to the destination.
dstAddrMode	The addressing mode for the destination address used in this command. 0x00 – Reserved; 0x01 – 16-bit group address for dstAddr and dstEndpoint not present; 0x02 – Reserved; 0x03 – 64-bit extended address for dstAddr and dstEndpoint present; 0x04 ~ 0xFF – Reserved.
dstAddr	The destination address for the binding entry.
dstEndpoint	Shall be present only if the dstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.

Example:

ZBHCI\_CMD\_BIND\_REQ

55 00 20 msgLenH msgLenL 00 XX[8 Bytes] XX XX[2 Bytes] XX XX[2 or 8 Bytes]  
XX[0 or 1 Bytes] AA

ZBHCI\_CMD\_UNBIND\_REQ

55 00 21 msgLenH msgLenL 00 XX[8 Bytes] XX XX[2 Bytes] XX XX[2 or 8 Bytes]  
XX[0 or 1 Bytes] AA

### 5.5.2.9 ZBHCI\_CMD\_MGMT\_LQI\_REQ

dstAddr	startIdx
2 Bytes	1 Byte

Name	Description
dstAddr	The destination address to which this command will be sent, shall be unicast.
startIdx	Starting index for the requested elements of the neighbor table.

Example: 55 00 30 00 03 00 XX[2 Bytes] XX AA



#### 5.5.2.10 ZBHCI\_CMD\_MGMT\_BIND\_REQ

<b>dstAddr</b>	<b>startIdx</b>
2 Bytes	1 Byte

Name	Description
dstAddr	The destination address to which this command will be sent, shall be unicast.
startIdx	Starting index for the requested elements of the binding table.

Example: 55 00 31 00 03 00 XX[2 Bytes] XX AA

#### 5.5.2.11 ZBHCI\_CMD\_MGMT\_LEAVE\_REQ

<b>dstAddr</b>	<b>devAddr</b>	<b>rejoin</b>	<b>removeChildren</b>
2 Bytes	8 Bytes	1 Byte	1 Byte

Name	Description
dstAddr	The destination address to which this command will be sent, shall be unicast.
devAddr	The IEEE address of the device to be removed or NULL if the device removes itself.
rejoin	TRUE if the device is being asked to leave from the current parent and requested to rejoin the network. Otherwise, the parameter has a value of FALSE.
removeChildren	TRUE if the device being asked to leave the network is also being asked to remove its child device, if any. Otherwise, it has a value of FALSE.

Example: 55 00 32 00 0C 00 XX[2 Bytes] XX[8 Bytes] XX XX AA

### 5.5.2.12 ZBHCI\_CMD\_MGMT\_PERMIT\_JOIN\_REQ

<b>dstAddr</b>	<b>permitDuration</b>	<b>TC_significance</b>
2 Bytes	1 Byte	1 Byte

<b>Name</b>	<b>Description</b>
dstAddr	The destination address to which this command will be sent.
permitDuration	The length of time in seconds during which the coordinator or router will allow associations. The value 0x00 and 0xff indicate that permission is disabled or enabled, respectively, without a specified limit.
TC_significance	This field shall always have a value of 1, indicating a request to change the Trust Center policy. If a frame is received with a value of 0, it shall be treated as having a value of 1.

Example: 55 00 34 00 04 00 XX[2 Bytes] XX XX AA

### 5.5.2.13 ZBHCI\_CMD\_MGMT\_NWK\_UPDATE\_REQ

<b>dstAddr</b>	<b>nwkManagerAddr</b>	<b>scanChannels</b>	<b>scanDuration</b>	<b>scanCount/ nwkUpdateId</b>
2 Bytes	2 Bytes	4 Bytes	1 Byte	1 Byte

<b>Name</b>	<b>Description</b>
dstAddr	The destination address to which this command will be sent.
nwkManagerAddr	This field shall be present only if the scanDuration is set to 0xff, and, when present, it indicates the NWK address for the device with the Network Manager bit set in its Node Descriptor.
scanChannels	Channel bit mask, 32-bit field structure.
scanDuration	A value used to calculate the length of time to spend on scanning each channel. 0x00 ~ 0x05 or 0xfe or 0xff.
scanCount	This field represents the number of energy scans to be conducted and reported. This field shall be present only if the scanDuration is within the range of 0x00 to 0x05.
nwkUpdateId	The value of the nwkUpdateId contained in this request. This value is set by the Network Channel Manager prior to sending the message. This field shall only be present if the scanDuration is 0xfe or 0xff. If the scanDuration is 0xff, then the value in the nwkUpdateId shall be ignored.

Example: 55 00 35 00 0A 00 XX[2 Bytes] XX[2 Bytes] XX[4 Bytes] XX XX AA

#### 5.5.2.14 ZBHCI\_CMD\_NODES\_JOINED\_GET\_REQ

startIdx
1 Byte

Name	Description
startIdx	Starting index for the requested elements of the joined node list.

Example: 55 00 40 00 01 00 XX AA

#### 5.5.2.15 ZBHCI\_CMD\_NODES\_TOGGLE\_TEST\_REQ

on/off	timerInterval
1 Byte	1 Byte

Name	Description
on/off	On/off command.
timerInterval	The timer interval of the data transmission. Unit: millisecond.

Example: 55 00 41 00 02 00 XX XX AA

### 5.5.3 Message Type (Slave)

Message Type	Value
ZBHCI_CMD_DISCOVERY_NWK_ADDR_RSP	0x8010
ZBHCI_CMD_DISCOVERY_IEEE_ADDR_RSP	0x8011
ZBHCI_CMD_DISCOVERY_NODE_DESC_RSP	0x8012
ZBHCI_CMD_DISCOVERY_SIMPLE_DESC_RSP	0x8013
ZBHCI_CMD_DISCOVERY_MATCH_DESC_RSP	0x8014
ZBHCI_CMD_DISCOVERY_ACTIVE_EP_RSP	0x8015
ZBHCI_CMD_BIND_RSP	0x8020
ZBHCI_CMD_UNBIND_RSP	0x8021
ZBHCI_CMD_MGMT_LQI_RSP	0x8030
ZBHCI_CMD_MGMT_BIND_RSP	0x8031
ZBHCI_CMD_MGMT_LEAVE_RSP	0x8032
ZBHCI_CMD_MGMT_DIRECT_JOIN_RSP	0x8033
ZBHCI_CMD_MGMT_PERMIT_JOIN_RSP	0x8034
ZBHCI_CMD_MGMT_NWK_UPDATE_RSP	0x8035

Message Type	Value
ZBHCI_CMD_NODES_JOINED_GET_RSP	0x8040
ZBHCI_CMD_NODES_TOGGLE_TEST_RSP	0x8041
ZBHCI_CMD_NODES_DEV_ANNCIE_IND	0x8043

#### 5.5.4 Payload (Slave)

##### 5.5.4.1 ZBHCI\_CMD\_DISCOVERY\_NWK\_ADDR\_RSP

seqNum	status	ieeeAddr	nwkAddr	numAssocDev	startIdx	assocDevList
1 Byte	1 Byte	8 Bytes	2 Bytes	0 or 1 Byte	0 or 1 Byte	0 or n Bytes

Name	Description
seqNum	ZDP transaction sequence number.
status	The status of the request command.
ieeeAddr	64-bit address for the Remote Device.
nwkAddr	16-bit address for the Remote Device.
numAssocDev	Count of the number of 16-bit short address to follow.
startIdx	Starting index into the list of associated devices for this report.
assocDevList	The list of associated devices.

Example: 55 80 10 msgLenH msgLenL 00 XX XX XX[8 Bytes] XX[2 Bytes] {XX XX XX[n Bytes]} AA

##### 5.5.4.2 ZBHCI\_CMD\_DISCOVERY\_IEEE\_ADDR\_RSP

seqNum	status	ieeeAddr	nwkAddr	numAssocDev	startIdx	assocDevList
1 Byte	1 Byte	8 Bytes	2 Bytes	0 or 1 Byte	0 or 1 Byte	0 or n Bytes

Name	Description
seqNum	ZDP transaction sequence number.
status	The status of the request command.
ieeeAddr	64-bit address for the Remote Device.
nwkAddr	16-bit address for the Remote Device.
numAssocDev	Count of the number of 16-bit short address to follow.
startIdx	Starting index into the list of associated devices for this report.
assocDevList	The list of associated devices.

Example: 55 80 11 msgLenH msgLenL 00 XX XX XX[8 Bytes] XX[2 Bytes] {XX XX XX[n Bytes]} AA

#### 5.5.4.3 ZBHCI\_CMD\_DISCOVERY\_NODE\_DESC\_RSP

seqNum	status	nwkAddrOfInterest	nodeDesc
1 Byte	1 Byte	2 Bytes	0 or n Bytes

Name	Description
seqNum	ZDP transaction sequence number.
status	The status of the request command.
nwkAddrOfInterest	NWK address for the request.
nodeDesc	This field shall only be included in the frame if the status field is SUCCESS.

Example: 55 80 12 msgLenH msgLenL 00 XX XX XX[2 Bytes] {XX[n Bytes]} AA

#### 5.5.4.4 ZBHCI\_CMD\_DISCOVERY\_SIMPLE\_DESC\_RSP

seqNum	status	nwkAddrOfInterest	length	simpleDesc
1 Byte	1 Byte	2 Bytes	1 Byte	n Bytes

Name	Description
seqNum	ZDP transaction sequence number.
status	The status of the request command.
nwkAddrOfInterest	NWK address for the request.
length	The length of simple description.
simpleDesc	This field shall only be included in the frame if the status field is SUCCESS.

Example: 55 80 13 msgLenH msgLenL 00 XX XX XX[2 Bytes] XX {XX[n Bytes]} AA

#### 5.5.4.5 ZBHCI\_CMD\_DISCOVERY\_MATCH\_DESC\_RSP

seqNum	status	nwkAddrOfInterest	matchLen	matchList
1 Byte	1 Byte	2 Bytes	1 Byte	n Bytes

Name	Description
seqNum	ZDP transaction sequence number.
status	The status of the request command.
nwkAddrOfInterest	NWK address for the request.
matchLen	The count of endpoints on the Remote Device that match the request criteria.
matchList	List of bytes each of which represents an 8-bit endpoint.

Example: 55 80 14 msgLenH msgLenL 00 XX XX XX[2 Bytes] XX {XX[n Bytes]} AA

#### 5.5.4.6 ZBHCI\_CMD\_DISCOVERY\_ACTIVE\_EP\_RSP

seqNum	status	nwkAddrOfInterest	activeEpCount	epList
1 Byte	1 Byte	2 Bytes	1 Byte	n Bytes

Name	Description
seqNum	ZDP transaction sequence number.
status	The status of the request command.
nwkAddrOfInterest	NWK address for the request.
activeEpCount	The count of active endpoints.
epList	List of active endpoints.

Example: 55 80 15 msgLenH msgLenL 00 XX XX XX[2 Bytes] XX {XX[n Bytes]} AA

#### 5.5.4.7 ZBHCI\_CMD\_BIND\_RSP/ ZBHCI\_CMD\_UNBIND\_RSP

seqNum	status
1 Byte	1 Byte

Name	Description
seqNum	ZDP transaction sequence number.
status	The status of the request command.

Example:

ZBHCI\_CMD\_BIND\_RSP

55 80 20 00 02 00 XX XX AA

ZBHCI\_CMD\_UNBIND\_RSP

55 80 21 00 02 00 XX XX AA

#### 5.5.4.8 ZBHCI\_CMD\_MGMT\_LQI\_RSP

seqNum	status	neighborTabEntries	startIdx	neighborTabListCount	neighborTabList
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	n Bytes

Name	Description
seqNum	ZDP transaction sequence number.
status	The status of the request command.
neighborTabEntries	Total number of Neighbor Table entries within the Remote Device.
startIdx	Starting index within the Neighbor Table to begin reporting for the neighborTabList.
neighborTabListCount	Number of Neighbor Table entries included within neighborTabList.
neighborTabList	A list of descriptors, beginning with the startIdx element and continuing for neighborTabListCount.

Example: 55 80 30 msgLenH msgLenL 00 XX XX XX XX XX {XX[n Bytes]} AA

#### 5.5.4.9 ZBHCI\_CMD\_MGMT\_BIND\_RSP

seqNum	status	bindingTabEntries	startIdx	bindingTabListCount	bindingTabList
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	n Bytes

Name	Description
seqNum	ZDP transaction sequence number.
status	The status of the request command.
bindingTabEntries	Total number of Binding Table entries within the Remote Device.
startIdx	Starting index within the Binding Table to begin reporting for the bindingTabList.
bindingTabListCount	Number of Binding Table entries included within bindingTabList.
bindingTabList	A list of descriptors, beginning with the startIdx element and continuing for bindingTabListCount.

Example: 55 80 31 msgLenH msgLenL 00 XX XX XX XX XX {XX[n Bytes]} AA

#### 5.5.4.10 ZBHCI\_CMD\_MGMT\_LEAVE\_RSP

seqNum	status
1 Byte	1 Byte

Name	Description
seqNum	ZDP transaction sequence number.
status	The status of the request command.

Example: 55 80 32 00 02 00 XX XX AA

#### 5.5.4.11 ZBHCI\_CMD\_MGMT\_PERMIT\_JOIN\_RSP

seqNum	status
1 Byte	1 Byte

Name	Description
seqNum	ZDP transaction sequence number.
status	The status of the request command.

Example: 55 80 34 00 02 00 XX XX AA

#### 5.5.4.12 ZBHCI\_CMD\_NODES\_JOINED\_GET\_RSP

status	totalCnt	startIdx	listCount	macAddrList
1 Byte	1 Byte	1 Byte	1 Byte	n Bytes

Name	Description
status	The status of the request command.
totalCnt	The total count of the joined nodes.
startIdx	Starting index within the mac address list.
listCount	The count of the MAC address list in the current packet.
macAddrList	The MAC address list in the current packet.

Example: 55 80 40 msgLenH msgLenL 00 XX XX XX XX {XX[n Bytes]} AA

#### 5.5.4.13 ZBHCI\_CMD\_NODES\_TOGGLE\_TEST\_RSP

No payload.

Example: 55 80 41 00 00 00 AA



#### 5.5.4.14 ZBHCI\_CMD\_NODES\_DEV\_ANNCE\_IND

<b>nwkAddr</b>	<b>ieeeAddr</b>	<b>capability</b>
2 Bytes	8 Bytes	1 Byte

<b>Name</b>	<b>Description</b>
nwkAddr	NWK address of the joined device.
ieeeAddr	IEEE address of the joined device.
capability	Capability of the joined device.

Example: 55 80 43 00 0B 00 XX[2 Bytes] XX[8 Bytes] XX AA

### 5.6 ZCL Cluster command

#### 5.6.1 ZCL command header format

ZCLCmdHdr:

<b>dstAddrMode</b>	<b>dstAddr</b>	<b>srcEp</b>	<b>dstEp</b>
1 Byte	0/2/8 Bytes	1 Byte	1 Byte

<b>Name</b>	<b>Description</b>
dstAddrMode	Destination address mode: 0 – without destination address and endpoint, for binding; 1 – with group address; 2 – with destination short address and endpoint; 3 – with destination IEEE address and endpoint.
dstAddr	Destination address for this command.
srcEp	Source endpoint.
dstEp	Destination endpoint if dstAddrMode is 2 or 3.

## 5.6.2 General cluster command

### 5.6.2.1 Message Type (Host)

Message Type	Value
ZBHCI_CMD_ZCL_ATTR_READ	0x0100
ZBHCI_CMD_ZCL_ATTR_WRITE	0x0101
ZBHCI_CMD_ZCL_CONFIG_REPORT	0x0102
ZBHCI_CMD_ZCL_READ_REPORT_CFG	0x0103

### 5.6.2.2 Payload (Host)

#### 5.6.2.2.1 ZBHCI\_CMD\_ZCL\_ATTR\_READ

ZCLCmdHdr	direction	clusterID	attrNum	attrList
n Bytes	1 Byte	2 Bytes	1 Byte	n Bytes

attrList:

attrID[0]	attrID[1]	...	attrID[n]
2 Bytes	2 Bytes	...	2 Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
direction	0 – Client to server; 1 – Server to client.
clusterID	Cluster identifier.
attrNum	The number of attributes to be read.
attrList	The list of the attribute IDs to be read.

Example:

55 01 00 msgLenH msgLenL 00 XX[n Bytes] XX XX[2 Bytes] XX {XX[n Bytes]} AA

#### 5.6.2.2.2 ZBHCI\_CMD\_ZCL\_ATTR\_WRITE

ZCLCmdHdr	direction	clusterID	attrNum	attrList[0]	...	attrList[n]
n Bytes	1 Byte	2 Bytes	1 Byte	n Bytes	...	n Bytes

attrList:

attrID	dataType	attrData
2 Bytes	1 Byte	n Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
direction	0 – Client to server; 1 – Server to client.
clusterID	Cluster identifier.
attrNum	The number of attributes to be written.
attrList	The list of the attributes to be written.

Example: 55 01 01 msgLenH msgLenL 00 XX[n Bytes] XX XX[2 Bytes] XX {XX[n Bytes] ...} AA

#### 5.6.2.2.3 ZBHCI\_CMD\_ZCL\_CONFIG\_REPORT

ZCLCmdHdr	direction	clusterID	attrNum	attrList[0]	...	attrList[n]
n Bytes	1 Byte	2 Bytes	1 Byte	n Bytes	...	n Bytes

attrList:

Report Direction	attrID	dataType	minRep Interval	maxRep Interval	reportable Change	timeout Period
1 Byte	2 Bytes	1 Byte	2 Bytes	2 Bytes	n Bytes	2 Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
direction	0 – Client to server; 1 – Server to client.
clusterID	Cluster identifier.
attrNum	The number of attributes to be configured.
attrList	The list of the attributes to be configured.

Example:

55 01 02 msgLenH msgLenL 00 XX[n Bytes] XX XX[2 Bytes] XX {XX[n Bytes] ...} AA

#### 5.6.2.2.4 ZBHCI\_CMD\_ZCL\_READ\_REPORT\_CFG

ZCLCmdHdr	direction	clusterID	attrNum	attrList[0]	...	attrList[n]
n Bytes	1 Byte	2 Bytes	1 Byte	n Bytes	...	n Bytes

attrList:

reportDirection	attrID
1 Byte	2 Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
direction	0 – Client to server; 1 – Server to client.
clusterID	Cluster identifier.
attrNum	The number of attributes' configuration to be read.
attrList	The list of the attributes to be read.

Example:

55 01 03 msgLenH msgLenL 00 XX[n Bytes] XX XX[2 Bytes] XX {XX[n Bytes] ...} AA

#### 5.6.2.3 Message Type (Slave)

Message Type	Value
ZBHCI_CMD_ZCL_ATTR_READ_RSP	0x8100
ZBHCI_CMD_ZCL_ATTR_WRITE_RSP	0x8101
ZBHCI_CMD_ZCL_CONFIG_REPORT_RSP	0x8102
ZBHCI_CMD_ZCL_READ_REPORT_CFG_RSP	0x8103
ZBHCI_CMD_ZCL_REPORT_MSG_RCV	0x8104

#### 5.6.2.4 Payload (Slave)

##### 5.6.2.4.1 ZBHCI\_CMD\_ZCL\_ATTR\_READ\_RSP

attrNum	attrList[0]	...	attrList[n]
1 Byte	n Bytes	...	n Bytes

attrList:

attrID	status	dataType	data
2 Bytes	1 Byte	1 Byte	n Bytes

Name	Description
attrNum	The number of attributes to be read.
attrList	The list of the attributes to be read.

Example: 55 81 00 msgLenH msgLenL 00 XX {XX[n Bytes] ...} AA

#### 5.6.2.4.2 ZBHCI\_CMD\_ZCL\_ATTR\_WRITE\_RSP

attrNum	attrList[0]	...	attrList[n]
1 Byte	n Bytes	...	n Bytes

attrList:

status	attrID
1 Byte	2 Bytes

Name	Description
attrNum	The number of attributes to be written.
attrList	The list of the attributes to be written.

Example: 55 81 01 msgLenH msgLenL 00 XX {XX[n Bytes] ...} AA

#### 5.6.2.4.3 ZBHCI\_CMD\_ZCL\_CONFIG\_REPORT\_RSP

attrNum	attrList[0]	...	attrList[n]
1 Byte	n Bytes	...	n Bytes

attrList:

status	reportDirection	attrID
1 Byte	1 Byte	2 Bytes

Name	Description
attrNum	The number of attributes' reporting to be configured.
attrList	The list of the attributes to be configured.

Example: 55 81 02 msgLenH msgLenL 00 XX {XX[n Bytes] ...} AA

#### 5.6.2.4.4 ZBHCI\_CMD\_ZCL\_READ\_REPORT\_CFG\_RSP

attrNum	attrList[0]	...	attrList[n]
1 Byte	n Bytes	...	n Bytes

attrList:

status	Report Direction	attrID	dataType	minRep Interval	maxRep Interval	Reportable Change	timeout Period
1 Byte	1 Byte	2 Bytes	1 Byte	2 Bytes	2 Bytes	n Bytes	2 Bytes

Name	Description
attrNum	The number of attributes' reporting to be read.
attrList	The list of the attributes to be read.

Example: 55 81 03 msgLenH msgLenL 00 XX {XX[n Bytes] ...} AA

#### 5.6.2.4.5 ZBHCI\_CMD\_ZCL\_REPORT\_MSG\_RCV

srcAddr	srcEp	attrNum	attrList[0]	...	attrList[n]
2 Bytes	1 Byte	1 Byte	n Bytes	...	n Bytes

attrList:

attrID	dataType	data
2 Bytes	1 Byte	n Bytes

Name	Description
srcAddr	The source address of the reporting message.
srcEp	The source endpoint of the reporting message.
attrNum	The number of attributes' reporting message to be received.
attrList	The list of the attributes' reporting message to be received.

Example: 55 81 04 msgLenH msgLenL 00 XX[2 Bytes] XX XX {XX[n Bytes] ...} AA

### 5.6.3 Basic cluster command

#### 5.6.3.1 Message Type (Host)

Message Type	Value
ZBHCI_CMD_ZCL_BASIC_RESET	0x0110

#### 5.6.3.2 Payload (Host)

ZCLCmdHdr
n Bytes

Example: 55 01 10 msgLenH msgLenL 00 XX[n Bytes] AA

## 5.6.4 Group cluster command

### 5.6.4.1 Message Type (Host)

Message Type	Value
ZBHCI_CMD_ZCL_GROUP_ADD	0x0120
ZBHCI_CMD_ZCL_GROUP_VIEW	0x0121
ZBHCI_CMD_ZCL_GROUP_GET_MEMBERSHIP	0x0122
ZBHCI_CMD_ZCL_GROUP_REMOVE	0x0123
ZBHCI_CMD_ZCL_GROUP_REMOVE_ALL	0x0124
ZBHCI_CMD_ZCL_GROUP_ADD_IF_IDENTIFYING	0x0125

### 5.6.4.2 Payload (Host)

#### 5.6.4.2.1 ZBHCI\_CMD\_ZCL\_GROUP\_ADD

ZCLCmdHdr	groupId	groupName
n Bytes	2 Bytes	n Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
groupId	Group identifier.
groupName	Group name, character string.

Example:

55 01 20 msgLenH msgLenL 00 XX[n Bytes] XX[2 Bytes] {XX[n Bytes]} AA

#### 5.6.4.2.2 ZBHCI\_CMD\_ZCL\_GROUP\_VIEW

ZCLCmdHdr	groupId
n Bytes	2 Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
groupId	Group identifier.

Example: 55 01 21 msgLenH msgLenL 00 XX[n Bytes] XX[2 Bytes] AA

#### 5.6.4.2.3 ZBHCI\_CMD\_ZCL\_GROUP\_GET\_MEMBERSHIP

ZCLCmdHdr	groupCount	groupList
n Bytes	1 Byte	n Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
groupCount	Group count.
groupList	Group list.

Example: 55 01 22 msgLenH msgLenL 00 XX[n Bytes] XX XX[n Bytes] AA

#### 5.6.4.2.4 ZBHCI\_CMD\_ZCL\_GROUP\_REMOVE

ZCLCmdHdr	groupId
n Bytes	2 Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
groupId	Group identifier.

Example: 55 01 23 msgLenH msgLenL 00 XX[n Bytes] XX[2 Bytes] AA

#### 5.6.4.2.5 ZBHCI\_CMD\_ZCL\_GROUP\_REMOVE\_ALL

ZCLCmdHdr
n Bytes

Example: 55 01 24 msgLenH msgLenL 00 XX[n Bytes] AA

#### 5.6.4.2.6 ZBHCI\_CMD\_ZCL\_GROUP\_ADD\_IF\_IDENTIFYING

ZCLCmdHdr	groupId	groupName
n Bytes	2 Bytes	n Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
groupId	Group identifier.
groupName	Group name, character string.

Example:

55 01 25 msgLenH msgLenL 00 XX[n Bytes] XX[2 Bytes] {XX[n Bytes]} AA



### 5.6.4.3 Message Type (Slave)

Message Type	Value
ZBHCI_CMD_ZCL_GROUP_ADD_RSP	0x8120
ZBHCI_CMD_ZCL_GROUP_VIEW_RSP	0x8121
ZBHCI_CMD_ZCL_GROUP_GET_MEMBERSHIP_RSP	0x8122
ZBHCI_CMD_ZCL_GROUP_REMOVE_RSP	0x8123

### 5.6.4.4 Payload (Slave)

#### 5.6.4.4.1 ZBHCI\_CMD\_ZCL\_GROUP\_ADD\_RSP

status	groupId
1 Byte	2 Bytes

Name	Description
status	The status field is set to SUCCESS, DUPLICATE_EXISTS, or INSUFFICIENT_SPACE as appropriate.
groupId	Group identifier.

Example: 55 81 20 00 03 00 XX XX[2 Bytes] AA

#### 5.6.4.4.2 ZBHCI\_CMD\_ZCL\_GROUP\_VIEW\_RSP

status	groupId	groupName
1 Byte	2 Bytes	n Bytes

Name	Description
status	The status field is set to SUCCESS, or NOT_FOUND as appropriate.
groupId	Group identifier.
groupName	Group name, character string.

Example: 55 81 21 msgLenH msgLenL 00 XX XX[2 Bytes] XX[n Bytes] AA

#### 5.6.4.4.3 ZBHCI\_CMD\_ZCL\_GROUP\_GET\_MEMBERSHIP\_RSP

capability	groupCount	groupList
1 Byte	1 Byte	n Bytes

Name	Description
capability	The remaining capability of the group table of the device.
groupCount	The number of groups contained in the group list field.
groupName	The list of groupId in the group list field.

Example: 55 81 22 msgLenH msgLenL 00 XX XX XX[n Bytes] AA

#### 5.6.4.4.4 ZBHCI\_CMD\_ZCL\_GROUP\_REMOVE\_RSP

status	groupId
1 Byte	2 Bytes

Name	Description
status	The status field is set to SUCCESS, DUPLICATE_EXISTS, or INSUFFICIENT_SPACE as appropriate.
groupId	Group identifier.

Example: 55 81 23 00 03 00 XX XX[2 Bytes] AA

### 5.6.5 Identify cluster command

#### 5.6.5.1 Message Type (Host)

Message Type	Value
ZBHCI_CMD_ZCL_IDENTIFY	0x0130
ZBHCI_CMD_ZCL_IDENTIFY_QUERY	0x0131

#### 5.6.5.2 Payload (Host)

##### 5.6.5.2.1 ZBHCI\_CMD\_ZCL\_IDENTIFY

ZCLCmdHdr	identifyTime
n Bytes	2 Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
identifyTime	Unsigned 16-bit integer.

Example: 55 01 30 msgLenH msgLenL 00 XX[n Bytes] XX[2 Bytes] AA

### 5.6.5.2.2 ZBHCI\_CMD\_ZCL\_IDENTIFY\_QUERY

ZCLCmdHdr
n Bytes

Example: 55 01 31 msgLenH msgLenL 00 XX[n Bytes] AA

### 5.6.5.3 Message Type (Slave)

Message Type	Value
ZBHCI_CMD_ZCL_IDENTIFY_QUERY_RSP	0x8131

### 5.6.5.4 Payload (Slave)

#### 5.6.5.4.1 ZBHCI\_CMD\_ZCL\_IDENTIFY\_QUERY\_RSP

shortAddr	srcEp	timeout
2 Bytes	1 Byte	2 Bytes

Name	Description
shortAddr	The short address of the device.
srcEp	The source endpoint of the device.
timeout	The remaining time.

Example: 55 81 31 00 05 00 XX[2 Bytes] XX XX[2 Bytes] AA

### 5.6.6 On/Off cluster command

#### 5.6.6.1 Message Type (Host)

Message Type	Value
ZBHCI_CMD_ZCL_ONOFF_ON	0x0140
ZBHCI_CMD_ZCL_ONOFF_OFF	0x0141
ZBHCI_CMD_ZCL_ONOFF_TOGGLE	0x0142

#### 5.6.6.2 Payload (Host)

ZCLCmdHdr
n Bytes

Example:

ZBHCI\_CMD\_ZCL\_ONOFF\_ON

55 01 40 msgLenH msgLenL 00 XX[n Bytes] AA

ZBHCI\_CMD\_ZCL\_ONOFF\_OFF

55 01 41 msgLenH msgLenL 00 XX[n Bytes] AA

ZBHCI\_CMD\_ZCL\_ONOFF\_TOGGLE

55 01 42 msgLenH msgLenL 00 XX[n Bytes] AA

## 5.6.7 Level cluster command

### 5.6.7.1 Message Type (Host)

Message Type	Value
ZBHCI_CMD_ZCL_LEVEL_MOVE2LEVEL	0x0150
ZBHCI_CMD_ZCL_LEVEL_MOVE	0x0151
ZBHCI_CMD_ZCL_LEVEL_STEP	0x0152
ZBHCI_CMD_ZCL_LEVEL_STOP	0x0153
ZBHCI_CMD_ZCL_LEVEL_MOVE2LEVEL_WITHONOFF	0x0154
ZBHCI_CMD_ZCL_LEVEL_MOVE_WITHONOFF	0x0155
ZBHCI_CMD_ZCL_LEVEL_STEP_WITHONOFF	0x0156
ZBHCI_CMD_ZCL_LEVEL_STOP_WITHONOFF	0x0157

### 5.6.7.2 Payload (Host)

#### 5.6.7.2.1 ZBHCI\_CMD\_ZCL\_LEVEL\_MOVE2LEVEL

ZCLCmdHdr	level	transTime
n Bytes	1 Byte	2 Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
level	Level.
transTime	Transition time, 1/10ths of a second.

Example: 55 01 50 msgLenH msgLenL 00 XX[n Bytes] XX XX[2 Bytes] AA

#### 5.6.7.2.2 ZBHCI\_CMD\_ZCL\_LEVEL\_MOVE

ZCLCmdHdr	mode	rate
n Bytes	1 Byte	1 Byte

Name	Description
ZCLCmdHdr	ZCL command header.
mode	Move mode. 0x00 – Up; 0x01 – Down.
rate	The rate field specifies the rate of movement in units per second.

Example: 55 01 51 msgLenH msgLenL 00 XX[n Bytes] XX XX AA

#### 5.6.7.2.3 ZBHCI\_CMD\_ZCL\_LEVEL\_STEP

ZCLCmdHdr	mode	stepSize	transTime
n Bytes	1 Byte	1 Byte	2 Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
mode	Step mode. 0x00 – Up; 0x01 – Down.
stepSize	A step is a change in the current level of ‘step size’ units.
transTime	The transition time field specifies the time that shall be taken to perform the step, in 1/10ths of a second.

Example: 55 01 52 msgLenH msgLenL 00 XX[n Bytes] XX XX XX[2 Bytes] AA

#### 5.6.7.2.4 ZBHCI\_CMD\_ZCL\_LEVEL\_STOP

ZCLCmdHdr
n Bytes

Example: 55 01 53 msgLenH msgLenL 00 XX[n Bytes] AA

#### 5.6.7.2.5 ZBHCI\_CMD\_ZCL\_LEVEL\_MOVE2LEVEL\_WITHONOFF

ZCLCmdHdr	level	transTime
n Bytes	1 Byte	2 Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
level	Level.
transTime	Transition time, 1/10ths of a second.

Example: 55 01 54 msgLenH msgLenL 00 XX[n Bytes] XX XX[2 Bytes] AA

#### 5.6.7.2.6 ZBHCI\_CMD\_ZCL\_LEVEL\_MOVE\_WITHONOFF

ZCLCmdHdr	mode	rate
n Bytes	1 Byte	1 Byte

Name	Description
ZCLCmdHdr	ZCL command header.
mode	Move mode. 0x00 – Up; 0x01 – Down.
rate	The rate field specifies the rate of movement in units per second.

Example: 55 01 55 msgLenH msgLenL 00 XX[n Bytes] XX XX AA

#### 5.6.7.2.7 ZBHCI\_CMD\_ZCL\_LEVEL\_STEP\_WITHONOFF

ZCLCmdHdr	mode	stepSize	transTime
n Bytes	1 Byte	1 Byte	2 Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
mode	Step mode. 0x00 – Up; 0x01 – Down.
stepSize	A step is a change in the current level of 'step size' units.
transTime	The transition time field specifies the time that shall be taken to perform the step, in 1/10ths of a second.

Example: 55 01 56 msgLenH msgLenL 00 XX[n Bytes] XX XX XX[2 Bytes] AA

### 5.6.7.2.8 ZBHCI\_CMD\_ZCL\_LEVEL\_STOP\_WITHONOFF

ZCLCmdHdr
n Bytes

Example: 55 01 57 msgLenH msgLenL 00 XX[n Bytes] AA

## 5.6.8 Scene cluster command

### 5.6.8.1 Message Type (Host)

Message Type	Value
ZBHCI_CMD_ZCL_SCENE_ADD	0x0160
ZBHCI_CMD_ZCL_SCENE_VIEW	0x0161
ZBHCI_CMD_ZCL_SCENE_REMOVE	0x0162
ZBHCI_CMD_ZCL_SCENE_REMOVE_ALL	0x0163
ZBHCI_CMD_ZCL_SCENE_STORE	0x0164
ZBHCI_CMD_ZCL_SCENE_RECALL	0x0165
ZBHCI_CMD_ZCL_SCENE_GET_MEMBERSHIP	0x0166

### 5.6.8.2 Payload (Host)

#### 5.6.8.2.1 ZBHCI\_CMD\_ZCL\_SCENE\_ADD

ZCLCmdHdr	groupId	sceneId	transTime	scene NameLen	scene Name	extField Len	extField Sets
n Bytes	2 Bytes	1 Byte	2 Byte	1 Byte	n Bytes	1 Byte	n Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
groupId	The group ID for which this scene applies.
sceneId	The identifier, unique within this group, which is used to identify this scene.
transTime	The amount of time, in seconds, it will take for the device to change from its current state to the requested scene.
sceneNameLen	Length of scene name.
sceneName	Scene name, char string.
extFieldLen	Length of extFieldSets field.
extFieldSets	The sum of all such defines a scene.

Example:

55 01 60 msgLenH msgLenL 00 XX[n Bytes] XX[2 Bytes] XX XX[2 Bytes] XX {XX[n Bytes]} XX {XX[n Bytes]} AA

#### 5.6.8.2.2 ZBHCI\_CMD\_ZCL\_SCENE\_VIEW

ZCLCmdHdr	groupid	sceneId
n Bytes	2 Bytes	1 Byte

Name	Description
ZCLCmdHdr	ZCL command header.
groupid	The group ID for which this scene applies.
sceneId	The identifier, unique within this group, which is used to identify this scene.

Example: 55 01 61 msgLenH msgLenL 00 XX[n Bytes] XX[2 Bytes] XX AA

#### 5.6.8.2.3 ZBHCI\_CMD\_ZCL\_SCENE\_REMOVE

ZCLCmdHdr	groupid	sceneId
n Bytes	2 Bytes	1 Byte

Name	Description
ZCLCmdHdr	ZCL command header.
groupid	The group ID for which this scene applies.
sceneId	The identifier, unique within this group, which is used to identify this scene.

Example: 55 01 62 msgLenH msgLenL 00 XX[n Bytes] XX[2 Bytes] XX AA

#### 5.6.8.2.4 ZBHCI\_CMD\_ZCL\_SCENE\_REMOVE\_ALL

ZCLCmdHdr	groupid
n Bytes	2 Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
groupid	The group ID for which this scene applies.

Example: 55 01 63 msgLenH msgLenL 00 XX[n Bytes] XX[2 Bytes] AA



#### 5.6.8.2.5 ZBHCI\_CMD\_ZCL\_SCENE\_STORE

ZCLCmdHdr	groupid	sceneId
n Bytes	2 Bytes	1 Byte

Name	Description
ZCLCmdHdr	ZCL command header.
groupid	The group ID for which this scene applies.
sceneId	The identifier, unique within this group, which is used to identify this scene.

Example: 55 01 64 msgLenH msgLenL 00 XX[n Bytes] XX[2 Bytes] XX AA

#### 5.6.8.2.6 ZBHCI\_CMD\_ZCL\_SCENE\_RECALL

ZCLCmdHdr	groupid	sceneId
n Bytes	2 Bytes	1 Byte

Name	Description
ZCLCmdHdr	ZCL command header.
groupid	The group ID for which this scene applies.
sceneId	The identifier, unique within this group, which is used to identify this scene.

Example: 55 01 65 msgLenH msgLenL 00 XX[n Bytes] XX[2 Bytes] XX AA

#### 5.6.8.2.7 ZBHCI\_CMD\_ZCL\_SCENE\_GET\_MEMBERSHIP

ZCLCmdHdr	groupid
n Bytes	2 Bytes

Name	Description
ZCLCmdHdr	ZCL command header.
groupid	The group ID for which this scene applies.

Example: 55 01 66 msgLenH msgLenL 00 XX[n Bytes] XX[2 Bytes] AA

### 5.6.8.3 Message Type (Slave)

Message Type	Value
ZBHCI_CMD_ZCL_SCENE_ADD_RSP	0x8160
ZBHCI_CMD_ZCL_SCENE_VIEW_RSP	0x8161
ZBHCI_CMD_ZCL_SCENE_REMOVE_RSP	0x8162
ZBHCI_CMD_ZCL_SCENE_REMOVE_ALL_RSP	0x8163
ZBHCI_CMD_ZCL_SCENE_STORE_RSP	0x8164
ZBHCI_CMD_ZCL_SCENE_GET_MEMBERSHIP_RSP	0x8166

### 5.6.8.4 Payload (Slave)

#### 5.6.8.4.1 ZBHCI\_CMD\_ZCL\_SCENE\_ADD\_RSP

status	groupId	sceneId
1 Byte	2 Bytes	1 Byte

Name	Description
status	SUCCESS, INSUFFICIENT_SPACE or INVALID_FIELD (the group is not present in the group table).
groupId	The group ID for which this scene applies.
sceneId	The identifier, unique within this group, which is used to identify this scene.

Example: 55 81 60 00 04 00 XX XX[2 Bytes] XX AA

#### 5.6.8.4.2 ZBHCI\_CMD\_ZCL\_SCENE\_VIEW\_RSP

status	groupId	sceneId	transTime	sceneName	extFieldSets
1 Byte	2 Bytes	1 Byte	2 Bytes	n Bytes	n Bytes

Name	Description
status	SUCCESS, NOT_FOUND (the scene is not present in the scene table) or INVALID_FIELD (the group is not present in the group table).
groupId	The group ID for which this scene applies.
sceneId	The identifier, unique within this group, which is used to identify this scene.
transTime	Transition time copied from scene table entry.
sceneName	Scene name copied from scene table entry. First byte is the length of the scene name.
extFieldSets	Extension field sets copied from scene table entry. First byte is the length of the extension field sets.

Example:

55 81 61 msgLenH msgLenL 00 XX XX[2 Bytes] XX XX[2 Bytes] XX[n Bytes]  
XX[n Bytes] AA

#### 5.6.8.4.3 ZBHCI\_CMD\_ZCL\_SCENE\_REMOVE\_RSP

status	groupid	sceneld
1 Byte	2 Bytes	1 Byte

Name	Description
status	SUCCESS, NOT_FOUND (the scene is not present in the scene table) or INVALID_FIELD (the group is not present in the group table).
groupid	The group ID for which this scene applies.
sceneld	The identifier, unique within this group, which is used to identify this scene.

Example: 55 81 62 00 04 00 XX XX[2 Bytes] XX AA

#### 5.6.8.4.4 ZBHCI\_CMD\_ZCL\_SCENE\_REMOVE\_ALL\_RSP

status	groupid
1 Byte	2 Bytes

Name	Description
status	SUCCESS or INVALID_FIELD (the group is not present in the group table).
groupid	The group ID for which this scene applies.

Example: 55 81 63 00 03 00 XX XX[2 Bytes] AA

#### 5.6.8.4.5 ZBHCI\_CMD\_ZCL\_SCENE\_STORE\_RSP

status	groupid	sceneld
1 Byte	2 Bytes	1 Byte

Name	Description
status	SUCCESS, INSUFFICIENT_SPACE or INVALID_FIELD (the group is not present in the group table).
groupid	The group ID for which this scene applies.
sceneld	The identifier, unique within this group, which is used to identify this scene.

Example: 55 81 64 00 04 00 XX XX[2 Bytes] XX AA

#### 5.6.8.4.6 ZBHCI\_CMD\_ZCL\_SCENE\_GET\_MEMBERSHIP\_RSP

status	capability	groupId	sceneCnt	sceneList
1 Byte	1 Byte	2 Bytes	1 Byte	n Bytes

Name	Description
status	SUCCESS or INVALID_FIELD (the group is not present in the group table).
capability	Contain the remaining capacity of the scene table of the device.
groupId	The group ID for which this scene applies.
sceneCnt	The number of scenes contained in the scene list field.
sceneList	Contain the identifiers of all the scenes in the scene table with the corresponding Group ID.

Example: 55 81 66 msgLenH msgLenL 00 XX XX XX[2 Bytes] XX XX[n Bytes] AA

### 5.6.9 OTA cluster command

#### 5.6.9.1 Message Type (Host)

Message Type	Value
ZBHCI_CMD_ZCL_OTA_IMAGE_NOTIFY	0x0190

#### 5.6.9.2 Payload (Host)

##### 5.6.9.2.1 ZBHCI\_CMD\_ZCL\_OTA\_IMAGE\_NOTIFY

ZCLCmdHdr	payloadType	queryJitter
n Bytes	1 Byte	1 Byte

Name	Description
ZCLCmdHdr	ZCL command header.
payloadType	0x00 – Query jitter; 0x01 – Query jitter and manufacturer code; 0x02 – Query jitter, manufacturer code, and image type; 0x03 - Query jitter, manufacturer code, image type, and new file version.
queryJitter	By using the parameter, it prevents a single notification of a new OTA upgrade image from flooding the upgrade server with requests from clients.

Example: 55 01 90 msgLenH msgLenL 00 XX[n Bytes] XX XX AA

## 6 Appendix: Pro R21 Certificate issued by Zigbee Alliance



**CERTIFICATE**

**ZIGBEE COMPLIANT PLATFORM**

The Zigbee Alliance is pleased to congratulate TeLink Micro LLC on the completion of the Zigbee Certification Program testing of the following product:

Type of Device	Zigbee Compliant Platform
Manufacturer	Zigbee PRO Feature Set (2015)
Model Identification	TeLink Micro LLC
Firmware Version	Telink ZB3.0 Platform
Hardware Version	Telink Zigbee Stack 1.0
Certification Date	TL8R82xx/TL8R86xx v1.1
Certification ID Number	February 7, 2018
	ZIG18002ZCP27119-24

This Certificate serves to confirm that the above-mentioned product has passed all relevant tests in conjunction with the Zigbee Certification Program and is deemed compliant to it. The manufacturer has been granted the right to use the following term and all associated logos:



The usage of this term is limited to the described device and does not encompass any changes, firmware upgrades or subsequent versions and models after the listed test date.

All usage guidelines for the Zigbee Logo also apply to the term above.

We congratulate you on the completion of the program!

Sincerely,

Zigbee Alliance

February 7, 2018

Date



Signature

508 Second Street, Suite 206 ■ Davis, CA 95616 ■ USA ■ Phone +1.530.564.4565 ■ [www.zigbee.org](http://www.zigbee.org)