GL-Zigbee SDK User Guide

Version	Changed Item	Author	Date
1.0.0	First version	Feng.He	2020.06.16
2.1.0	Change code architecture、Change API & add zdo API	Feng.He	2020.11.25
2.1.1	Add new APIs	Feng.He	2020.12.04
2.2.0	Add UART support	Feng.He	2020.12.10
2.2.1	Optimized APIs	Feng.He	2020.12.18
2.2.2	Optimized APIs	Feng.He	2020.12.25

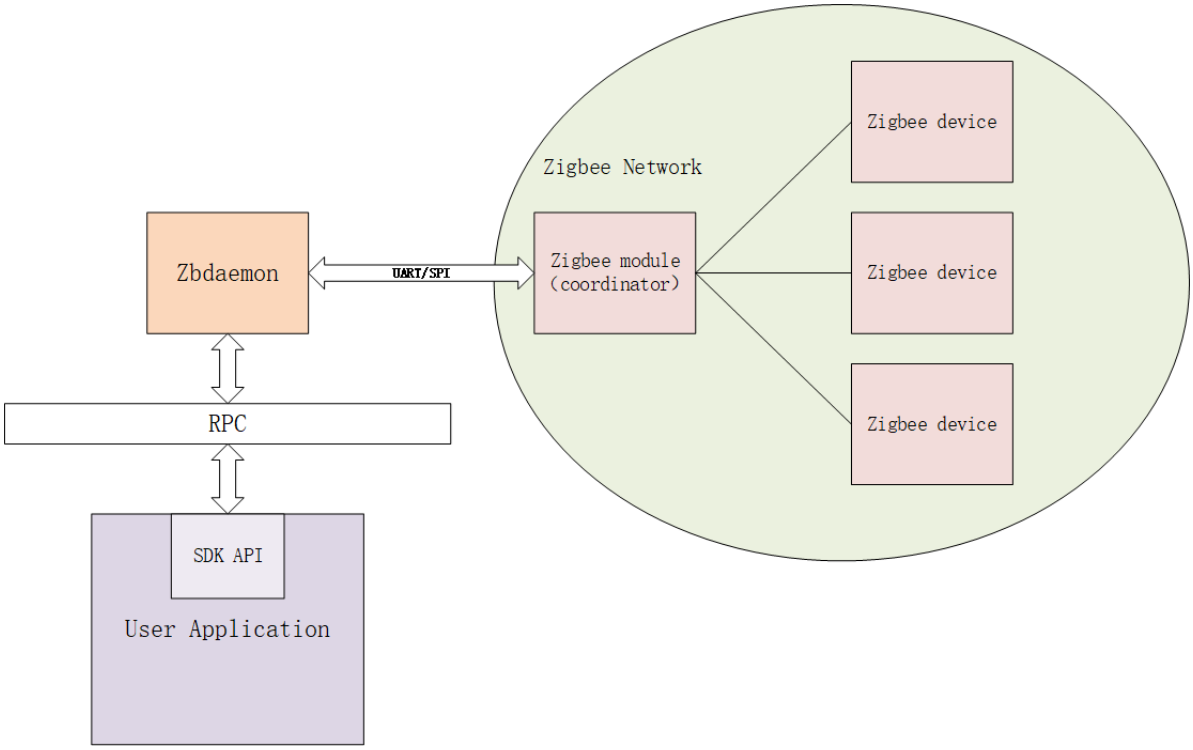
1. Description

1.1 What's GL-Zigbee SDK

GL-Zigbee SDK is developed and provided by GL-iNet Technology as a part of the zigbee solution. The SDK encapsulates the communication between the gateway and the zigbee module, allowing third-party developers to focus on the business layer rather than the communication layer.

There are two parts of GL-Zigbee SDK, zbdaemon and gl-zb-api. Zbdaemon is used to communicate with modules and process zigbee protocol stack. Gl-zb-api encapsulates the communication between the user application and the zbdaemon. To maintain the stability of zbdaemon, we separated it from the user project. So developers only need to call the APIs, no need to maintain zbdaemon.

The overall structure is as follows:



1.2 How to test easily

Zbtool is a cmd line tool for debug gl-zigbee module which bases on gl-zb-api. You can use it for quick controning and managing zigbee network and devices.

```
$ opkg update
$ opkg install gl-zbtool_2.1.0_ipq806x.ipk
```

For information on how to use zbtool, users can refer to the “Zbtool User Guide”.

1.3 Download

```
$ git clone https://github.com/gl-inet/gl-zigbee-sdk.git
```

1.4 Directory structure of gl-zigbee_sdk

```
|-- Makefile
|
|-- LICENSE
|
|-- VERSION_FILE
|
|-- README
|
|-- doc                                 # document
|   |-- Zbtool User Guide
|   |-- GL-Zigbee SDK User Guide
|
|-- files
|   |-- gl-zbtool.init                  # configuration file
|
|-- src
    |   |-- components
    |   |   |-- dev_mgr
```

```
|       |-- log
|       |-- thread
|
|-- daemon                          # zb daemon
|   |-- daemon
|   |-- zbdriver
|       |--silabs                   # silabs SDK
|
|-- include                         # header file
|
|-- lib                             # zb api lib
|   |-- libglzbapi.h
|   |-- libglzbapi.c
|
|-- project                         # user application file
|   |-- demo.c                      # demo file
|
|-- tool
|   |-- cli.c                       # debug tool - zbtool
|
|-- Makefile
```

1.5 Adapt the hardware interface

We support both SPI and Uart hardware interface.

You can switch interface drivers by modifying the `src/Makefile` before compiling.

```
#src/Makefile

MODULE_INTERFACE = SPI
# MODULE_INTERFACE = UART
```

2. API References

2.1 glzb_init()

```
GL_RET glzb_init(void);
```

Summary: Initialize the zigbee.

Parameters: Void

Return Values: Check return code

2.2 glzb_free()

```
GL_RET glzb_free(void);
```

Summary: End of the program and release the resource.

Parameters: Void

Return Values: Check return code

Note: This API has been abandoned and is no longer in use.

2.3 glzb_get_sdk_ver()

```
GL_RET glzb_get_sdk_ver(char* version);
```

Summary: End of the program and release the resource.

Parameters: Pointer to string for version

Return Values: Check return code

2.4 glzb_subscribe()

```
GL_RET glzb_subscribe(int timeout);
```

Summary: Enable callback functions(see in gl_zb_register_cb()), subscribe module message.

Parameters:

parameter	description
timeout	The time of subscribe, unit is millisecond.

During the timeout period, this API is blocked. If the timeout is set to 0, it will always subscribe.

Return Values: Check return code

2.5 glzb_unsubscribe()

```
GL_RET glzb_unsubscribe(void);
```

Summary: Disable callback functions(see in gl_zb_register_cb()), unsubscribe module message.

Parameters: Void

Return Values: Check return code

Note: This API has been abandoned and is no longer in use.

2.6 glzb_register_cb()

```
GL_RET glzb_register_cb(glzb_cbs_s *cb);
```

Summary: Register user callback functions

Parameters: glzb_cbs_s *cb

```
typedef struct {
    int (*z3_zcl_report_cb)(glzb_zcl_repo_s* zcl_p);
    int (*z3_zdo_report_cb)(glzb_zdo_repo_s* zdo_p);
    int (*z3_dev_manage_cb)(glzb_desc_s *dev);
} glzb_cbs_s;
```

Return Values: Check return code

2.6.1 int (*z3_zcl_report_cb)(glzb_zcl_repo_s* zcl_p)

Summary: Receive zcl message reports from the module. User can get report message and use it in this callback. This callback will be called when module receive a zcl data(report or response).

Parameters: glzb_zcl_repo_s* zcl_p

```
typedef struct {
    uint16_t short_id;
    uint16_t profile_id;
    uint16_t cluster_id;
    uint8_t src_endpoint;
    uint8_t dst_endpoint;
    uint8_t cmd_type;
    uint8_t cmd_id;
    uint16_t msg_length;
    uint8_t *message;
} glzb_zcl_repo_s
```

parameter	description
short_id	Short ID of report deivce
profile_id	Profile ID of this message
cluster_id	Cluster ID of this message
src_endpoint	Source endpoint of this message
dst_endpoint	Destination endpoint of this message
cmd_type	0: global zcl cmd; 1: specific zcl cmd
cmd_id	ZCl command ID of this message
msg_length	The length of data
message	Data

Return Values: Define by user.

2.6.2 int (*z3_zdo_report_cb)(glzb_zdo_repo_s* zdo_p)

Summary: Receive zdo message reports from the module. User can get report message and use it in this callback. This callback will be called when module receive a zdo data(report or response).

Parameters: glzb_zdo_repo_s* zdo_p

```
typedef struct {
    uint16_t short_id;
    uint16_t profile_id;
    uint16_t cluster_id;
    uint16_t msg_length;
    uint8_t *message;
} glzb_zdo_repo_s;
```

parameter	description
short_id	Short ID of report deivce
profile_id	Profile ID of this message
cluster_id	Cluster ID of this message
msg_length	The length of data
message	Data

Return Values: Define by user.

2.6.3 int (*z3_dev_manage_cb)(glzb_desc_s *dev)

Summary: Receive zigbee device message reports from the module. User can get report message and use it in this callback. This callback will be called when a zigbee device join, rejoin or left the network.

Parameters: glzb_desc_s *dev

```
typedef struct {
    char eui64[DEVICE_MAC_LEN+1];
    uint16_t short_id;
    uint16_t parent_node_id;
    glzb_dev_update status;
    glzb_Join_decision decision;
} glzb_desc_s;
```

parameter	description
eui64	Eui64 of new device
short _id	Short ID of new device
parent_node_id	Short ID of the parent of new device
status	Status of the Update Device message
decision	The decision made by the Trust Center when a node attempts to join.

Return Values: Define by user.

2.7 glzb_get_module_msg()

```
GL_RET glzb_get_module_msg(glzb_module_ver_s* status);
```

Summary: Get module software information.

Parameters: glzb_module_ver_s* status

```
typedef struct {
    char mac[DEVICE_MAC_LEN+1];
    uint16_t build;
    uint8_t major;
    uint8_t minor;
    uint8_t patch;
    uint8_t special;
    uint8_t type;
} glzb_module_ver_s;
```

Return Values: Check return code

2.8 glzb_get_nwk_status()

```
GL_RET glzb_get_nwk_status(glzb_nwk_status_para_s* status);
```

Summary: Get current network status.

Parameters: glzb_nwk_status_para_s* status

```
typedef struct {
    glzb_nwk_status_e nwk_status;
    glzb_node_type_e node_type;
    char extended_pan_id[EXPENDED_PAN_ID_LEN+1];
    uint16_t pan_id;
    uint8_t radio_tx_power;
    uint8_t radio_channel;
    glzb_join_method_e join_method;
    uint16_t nwk_manager_id;
    uint8_t nwk_update_id;
} glzb_nwk_status_para_s;
```

parameter	description
nwk_status	State of device
node_type	Type of device in current network
extended_pan_id	Extended pan ID of current network
pan_id	Pan ID of current network
radio_tx_power	Radio TX power
radio_channel	Radio channel
join_method	The type of method used for joining.
nwk_manager_id	The ID of the network manager in the current network.
nwk_update_id	The value of the ZigBee nwkUpdateId known by the stack.

Return Values: Check return code

2.9 glzb_create_nwk()

```
GL_RET glzb_create_nwk(uint16_t pan_id, uint8_t channel, uint8_t tx_power);
```

Summary: Create a new zigbee network(as coordinator).

Parameters:

parameter	description
pan_id	Pan ID of network
channel	Channel of network
tx_power	Radio TX power(dbm)

The tx power can be set to 0-255, but the actual tx power is limited by the chip, so setting the tx power beyond the upper limit will not take effect.

Return Values: Check return code

2.10 glzb_leave_nwk()

```
GL_RET glzb_leave_nwk(void);
```

Summary: leave the network.

Parameters: Void

Return Values: Check return code

2.11 glzb_allow_dev_join()

```
GL_RET glzb_allow_dev_join(int limit_time);
```

Summary: open the network, allow new device join in.

Parameters:

parameter	description
limit_time	The time of open network(0-255)(s)

Return Values: Check return code

2.12 glzb_delete_dev()

```
GL_RET glzb_delete_dev(char* mac, uint16_t short_id);
```

Summary: Remove device from zigbee network.

Parameters:

parameter	description
mac	Eui64 of target device
short_id	Short ID of target device

Return Values: Check return code

2.13 glzb_init_dev_tab()

```
glzb_dev_table_s* glzb_init_dev_tab(void);
```

Summary: Initialize the pointer to device table.

Parameters: Void.

Return Values: Pointer to device table.

Note: This API is used with glzb_free_dev_tab().

2.14 glzb_get_dev_tab()

```
GL_RET glzb_get_dev_tab(glzb_dev_table_s *table);
```

Summary: Get current child/neighbor device table. A child device is usually a zigbee-end_device or zigbee-sleepy_end_device mounted on the current device. A neighbor device is usually a zigbee-router_device.

Note: if an end-device mounted on other router, it will not show in child table.

Parameters: glzb_dev_table_s *table

```
typedef struct glzb_child_tab_node{
    char eui64[DEVICE_MAC_LEN+1];
    glzb_node_type_e type;
    uint16_t short_id;
    uint8_t phy;
    uint8_t power;
    uint8_t timeout;
    struct glzb_child_tab_node* next;
} glzb_child_tab_s;

typedef struct glzb_neighbor_tab_node{
    uint16_t short_id;
    uint8_t average_lqi;
    uint8_t in_cost;
    uint8_t out_cost;
    uint8_t age;
    char eui64[DEVICE_MAC_LEN+1];
    struct glzb_neighbor_tab_node* next;
} glzb_neighbor_tab_s;

typedef struct {
    int child_num;
    glzb_child_tab_s *child_table_header;
    int neighbor_num;
```

```
    glzb_neighbor_tab_s *neighbor_table_header;
} glzb_dev_table_s;
```

Return Values: Check return code

2.15 glzb_free_dev_tab()

```
GL_RET glzb_free_dev_tab(glzb_dev_table_s *table);
```

Summary: Free the pointer to device table.

Parameters: glzb_dev_table_s *table

Return Values: Check return code

Note: This API is used with glzb_init_dev_tab().

2.16 glzb_send_zcl_cmd()

```
GL_RET glzb_send_zcl_cmd(glzb_aps_s *frame);
```

Summary: Create and send a zcl command.

Parameters: glzb_aps_s *frame

```
typedef struct {
    char mac[DEVICE_MAC_LEN+1];
    uint16_t short_id;
    uint16_t profile_id;
    uint16_t cluster_id;
    uint16_t group_id;
    uint8_t src_endpoint;
    uint8_t dst_endpoint;
    uint8_t cmd_type;           //0: global zcl cmd; 1: specific zcl cmd
    uint8_t cmd_id;
    uint8_t frame_type;         //0: unicast; 1: multicast; 2: broadcast
    uint16_t msg_length;
    uint8_t *message;
} glzb_aps_s;
```

parameter	description
mac	Eui64 of target device
short_id	Short ID of target device
profile_id	Profile ID of this message
cluster_id	Cluster ID of this message
src_endpoint	Source endpoint of this message
dst_endpoint	Destination endpoint of this message
cmd_type	0: global zcl cmd; 1: specific zcl cmd
cmd_id	ZCl command ID of this message
frame_type	0: unicast; 1: multicast; 2: broadcast
msg_length	The length of data
message	Data

Return Values: Check return code

2.17 glzb_send_node_desc_req()

```
GL_RET glzb_send_node_desc_req(uint16_t target);
```

Summary: Send node descriptor request to target device.

Parameters:

parameter	description
target	Short ID of target device

Return Values: Check return code

2.18 glzb_send_power_desc_req()

```
GL_RET glzb_send_power_desc_req(uint16_t target);
```

Summary: Send power descriptor request to target device.

Parameters:

parameter	description
target	Short ID of target device

Return Values: Check return code

2.19 glzb_send_active_eps_req()

```
GL_RET glzb_send_active_eps_req(uint16_t target);
```

Summary: Send active endpoint request to target device.

Parameters:

parameter	description
target	Short ID of target device

Return Values: Check return code

2.20 glzb_send_simple_desc_req()

```
GL_RET glzb_send_simple_desc_req(uint16_t target, uint8_t targetEndpoint);
```

Summary: Send simple descriptor request to target device.

Parameters:

parameter	description
target	Short ID of target device
targetEndpoint	endpoint on the target device where the simple descriptor request will be sent

Return Values: Check return code

2.21 glzb_send_lqi_tab_req()

```
GL_RET glzb_send_lqi_tab_req(uint16_t target, uint8_t startIndex);
```

Summary: Send lqi table request to target device.

Parameters:

parameter	description
target	Short ID of target device
startIndex	starting index into table query

Return Values: Check return code

2.22 glzb_send_routing_tab_req()

```
GL_RET glzb_send_routing_tab_req(uint16_t target, uint8_t startIndex);
```

Summary: Send routing table request to target device.

Parameters:

parameter	description
target	Short ID of target device
startIndex	starting index into table query

Return Values: Check return code

2.23 glzb_send_binding_tab_req()

```
GL_RET glzb_send_binding_tab_req(uint16_t target, uint8_t startIndex);
```

Summary: Send binding table request to target device.

Parameters:

parameter	description
target	Short ID of target device
startIndex	starting index into table query

Return Values: Check return code

2.24 glzb_send_dev_bind_req()

```
GL_RET glzb_send_dev_bind_req(glzb_bind_req_para_s* bind_para);
```

Summary: Sendzigbee device bind request to target device.

Parameters: glzb_bind_req_para_s* bind_para

```
typedef struct {
    uint16_t target;
    uint16_t bind_cluster_id;
    uint8_t source[8];
    uint8_t sourceEndpoint;
    uint16_t clusterId;
    uint8_t type;
    uint8_t destination[8];
    uint16_t groupAddress;
    uint8_t destinationEndpoint;
} glzb_bind_req_para_s;
```

parameter	description
target	Short ID of target device
bind_cluster_id	Bind cluster ID
source	The source EUI64 of the binding
sourceEndpoint	The source endpoint of the binding
clusterId	The cluster ID to bind
type	The type of bind request
destination	The destination EUI64 of the binding
groupAddress	The group address in the binding if use group bind
destinationEndpoint	The destination endpoint of the binding

```
Bind cluster ID:
- #define BIND_REQUEST                       0x0021
- #define UNBIND_REQUEST                     0x0022

Type of bind request:
- #define UNICAST_BINDING                      0x03
- #define UNICAST_MANY_TO_ONE_BINDING          0x83
- #define MULTICAST_BINDING                    0x01
```

Return Values: Check return code

2.25 glzb_send_match_desc_req()

```
GL_RET glzb_send_match_desc_req(uint16_t target, uint16_t profile, uint8_t
inCount,uint8_t outCount, uint16_t *inClusters, uint16_t *outClusters);
```

Summary: Send match descriptors request to target device.

Parameters:

parameter	description
target	Short ID of target device
profile	Profile id for the match descriptor request
inCount	Num of in clusters
outCount	Num of out clusters
inClusters	Array of in clusters
outClusters	Array of out clusters

Return Values: Check return code

3. Return Code

```
typedef uint16_t GL_RET;
```

0x0000 - 0x00FF Error code defined by GL-iNet

0x0100 - 0xFFFF Error code defined by zigbee module manufacturer

Description	Error code defined by GL-iNet
GL_SUCCESS	0x0000
GL_UNKNOW_ERR	0x0001
GL_PARAM_ERR	0x0002
GL_UBUS_CONNECT_ERR	0x0011
GL_UBUS_LOOKUP_ERR	0x0012
GL_UBUS_SUBSCRIBE_ERR	0x0013
GL_UBUS_INVOKE_ERR	0x0014
GL_UBUS_REGISTER_ERR	0x0015
GL_UBUS_CALL_STR_ERR	0x0016
GL_UBUS_JSON_PARSE_ERR	0x0017