Telink

Telink Semiconductor (Shanghai) Co., Ltd.

IoT Wireless Connectivity Solutions

Telink Zigbee Overview

Lecturer: Bin YANG（bin.yang@telink-semi.com）

1. Introduction of chip

Telink Zigbee chip series

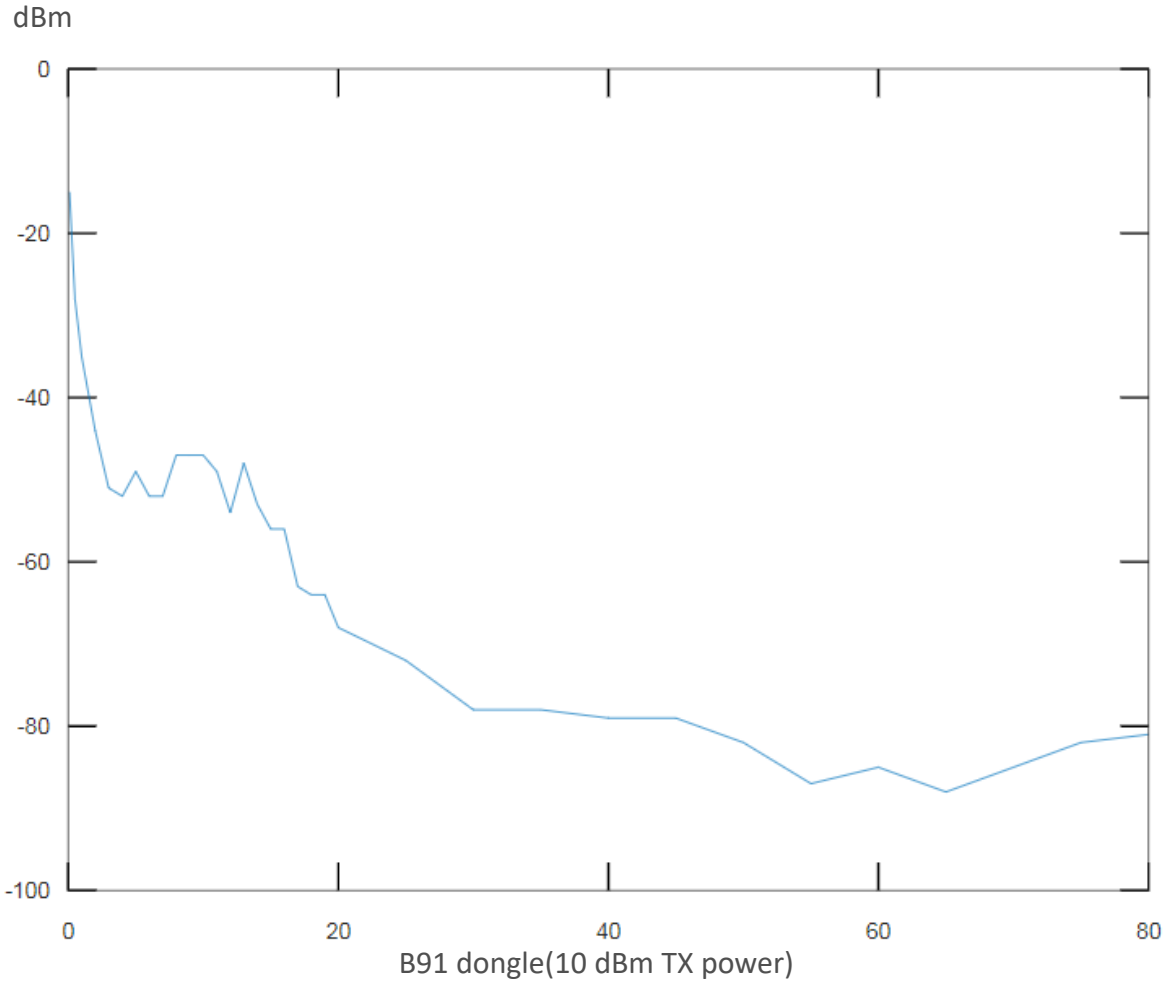
- TLSR8
  - 8258 - Multimode ULP: Zigbee 3.0 + Bluetooth 5.0
    - 64K SRAM(32K with retention); 512K/1M FLASH
    - TX: up to 10dBm; RX: -99.5dBm@802.15.4 250kbps
    - RX: 5.3mA; TX: 4.8mA@0dBm
  - 8278 - Multimode ULP: Zigbee 3.0 + Bluetooth 5.1
    - 64K SRAM(32K with retention); 1M FLASH
    - TX: up to 10dBm; RX: -99.5dBm@802.15.4 250kbps
    - HW accelerator for ECC
    - RX: 4.6mA(DCDC), 9.1mA(LDO)

      TX: 4.9mA(DCDC), 9.5mA(LDO) @0dBm

- TLSR9
  - 9518 – Multimode ULP, RISC-V IoT: Zigbee 3.0 + Bluetooth 5.2
    - 32-bit RISC-V MCU
    - 256K SRAM(64K with retention)
    - 1M/2M Flash
    - TX: up to 10dBm; RX: -99.5dBm@802.15.4 250kbps
    - Max. 96M operating frequency
    - DSP instruction set, floating-point unit
    - 1.8 - 5.5V
    - RX: 5.2mA(DCDC); TX: 5.3mA(DCDC)@0dBm

Telink Zigbee distance vs RSSI

- Relationship between distance and RSSI
  - 11 channel
  - Office environment

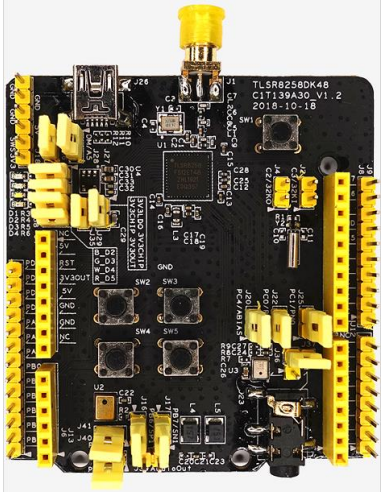


2. Development environment

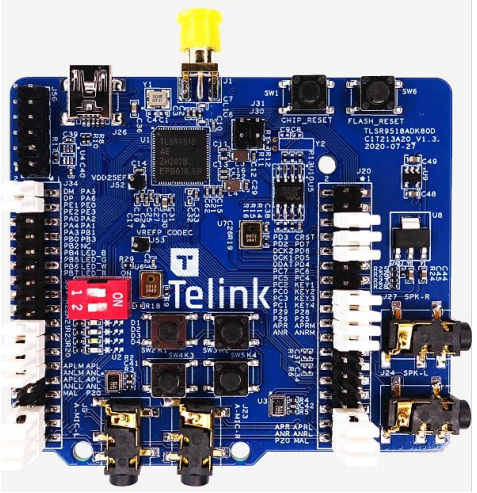
Hardware

➢ Development boards

- EVK Board



8258 Development Board



B91 Development Board

- USB Dongle

8258 Dongle

B91 Dongle

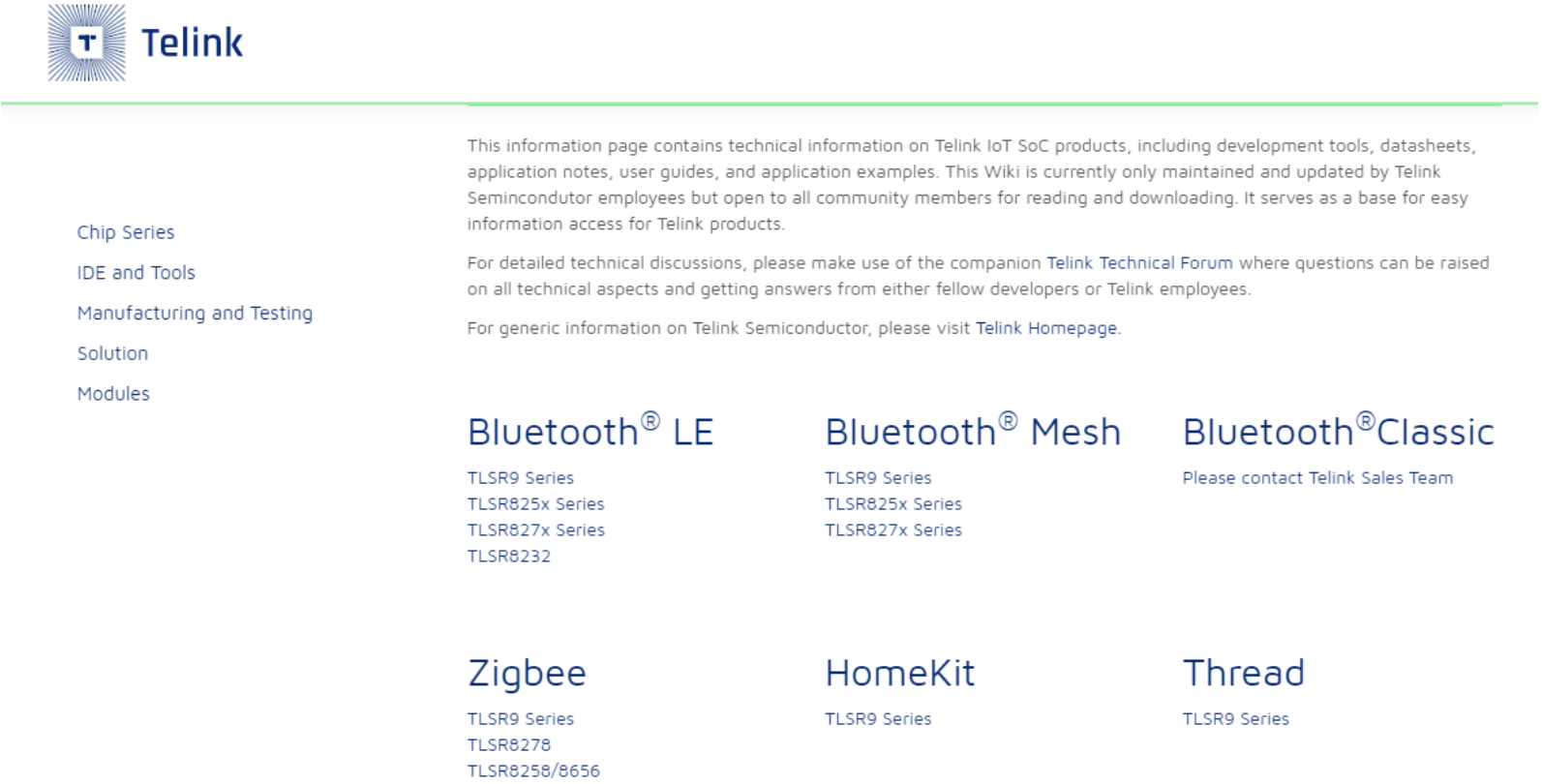
Hardware

➢ Burning EVK

- Connect to PC via USB, the connection is successful after the LED if off.
  - Driver-free
- Connect to DUT using DuPont wires
  - 3V3 -> 3V3(DUT)
  - SWM -> SWS(DUT)
  - GND -> GND(DUT)

Software

- Download link: wiki.telink-semi.cn
  - Integrated Development Environment (IDE)
    - IDE for TLSR8 Chips
    - IDE for TLSR9 Chips
  - Software Development Kit (SDK)
    - V3.6.x
  - Burning and Debugging Tool (BDT)
    - Burning and Debugging Tool



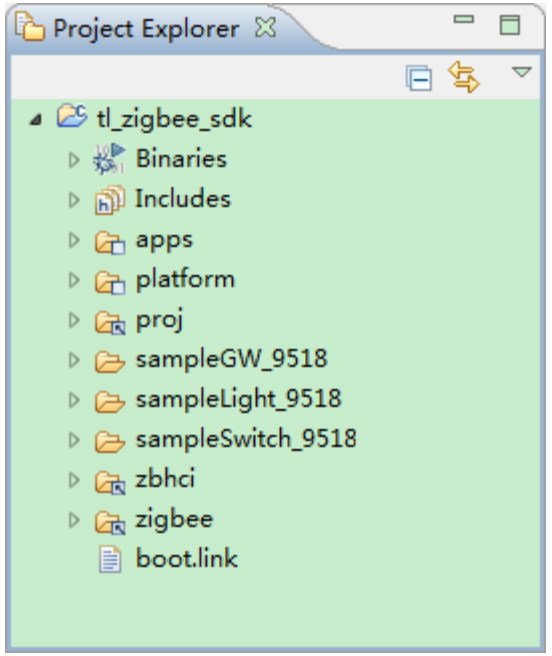
Telink Zigbee SDK

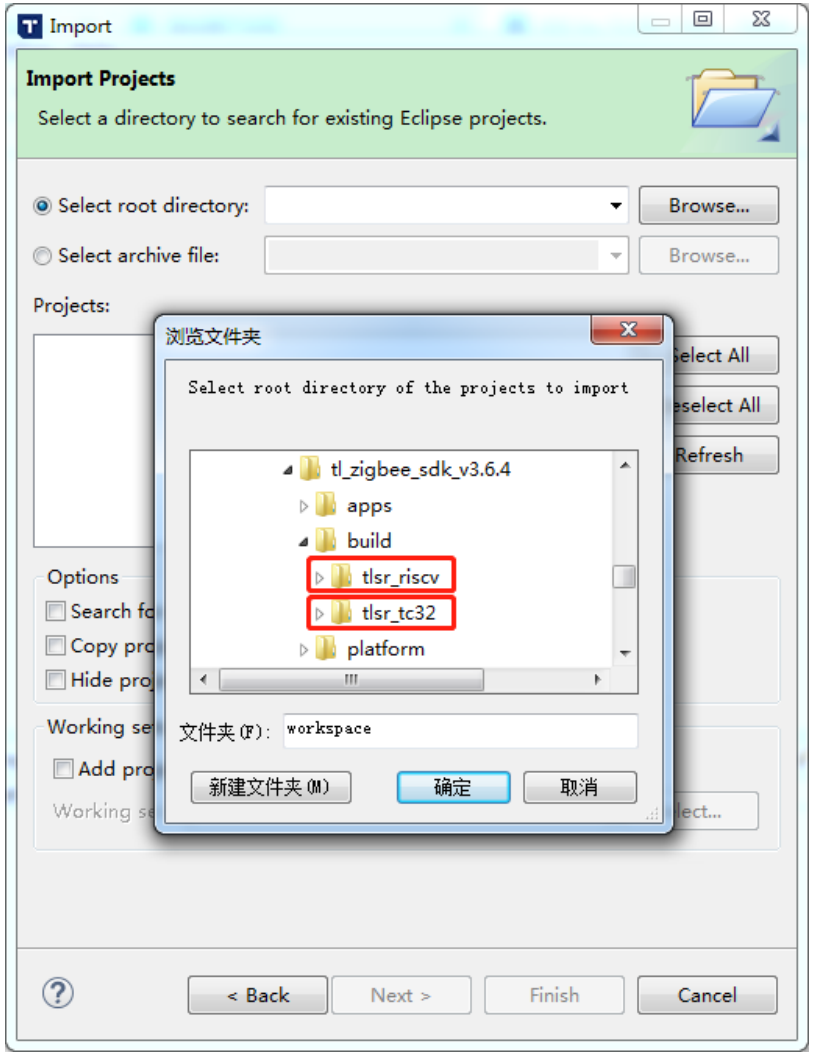
➢ Project import

- Open IDE
- On menu, File->Import->Existing Projects into Workspace
- select tl_zigbee_sdk/build
  - Select tlsr_tc32 for TLSR8
  - Select tlsr_riscv for TLSR9

➢ Directory structure

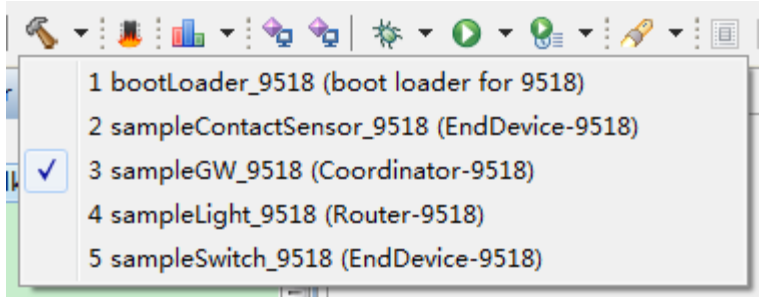
- /apps: application directory
- /platform: platform directory
- /proj: project directory
- /zigbee: protocol stack directory
- /zbhci: hci command processing directory

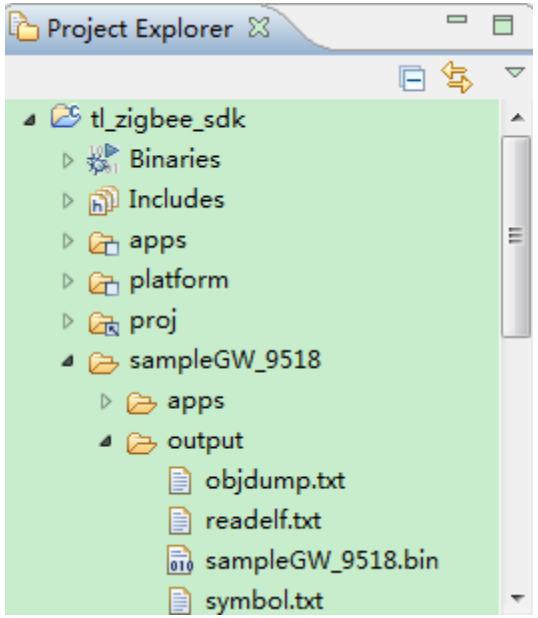




Telink Zigbee SDK

- Project compilation
  - sampleGW_xx
  - sampleLight_xx
  - sampleSwitch_xx
  - sampleContactSensor_xx
- Compilation output file
  - Burning file (xx.bin)
  - Mapping file (xx.lst / objdump.txt)

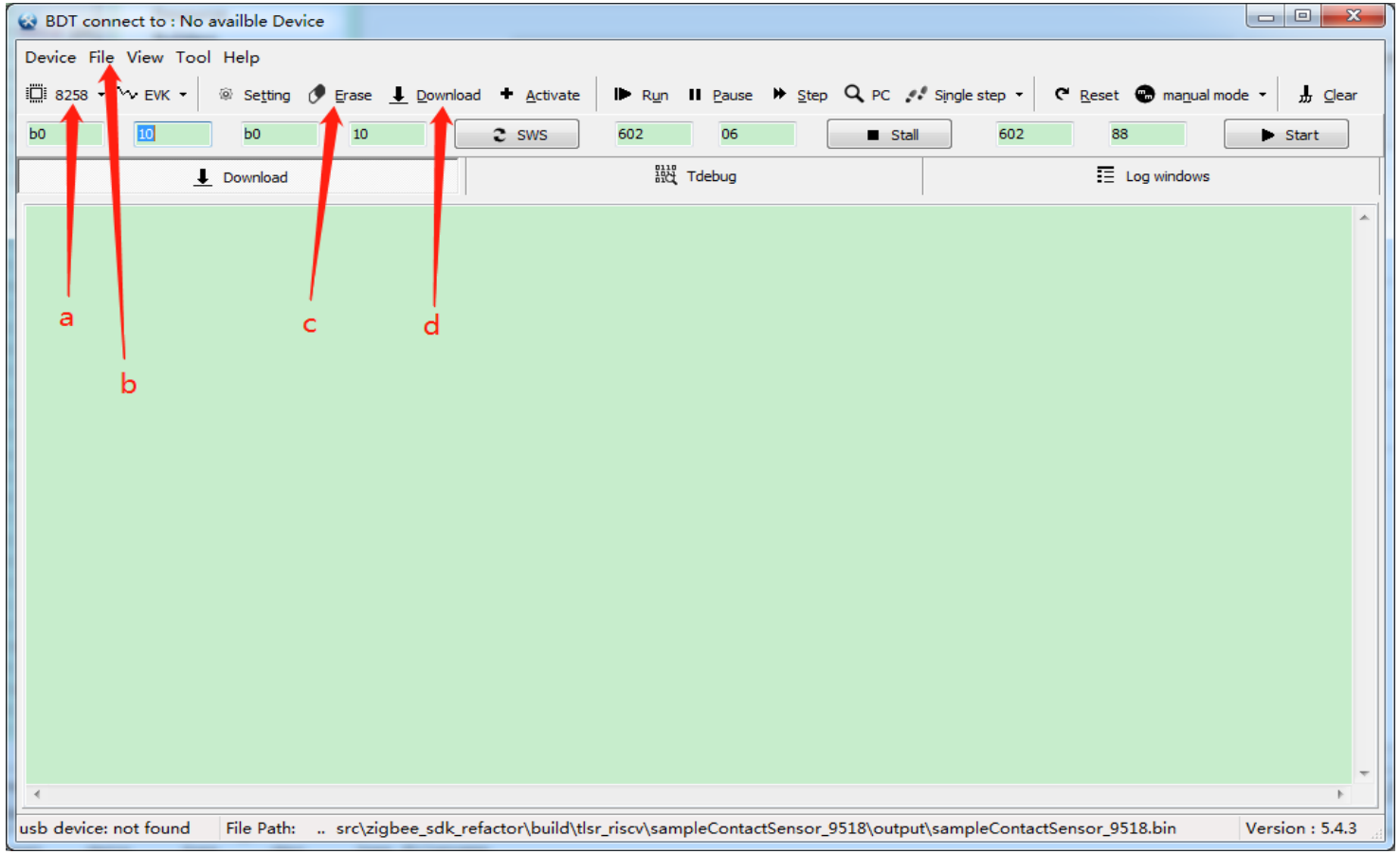




Telink Zigbee SDK

➢ Firmware burning

- Single wire connection
- Burn the firmware using BDT tool
  - a. Select chip type
  - b. Select firmware
  - c. Erase FLASH
  - d. Download firmware



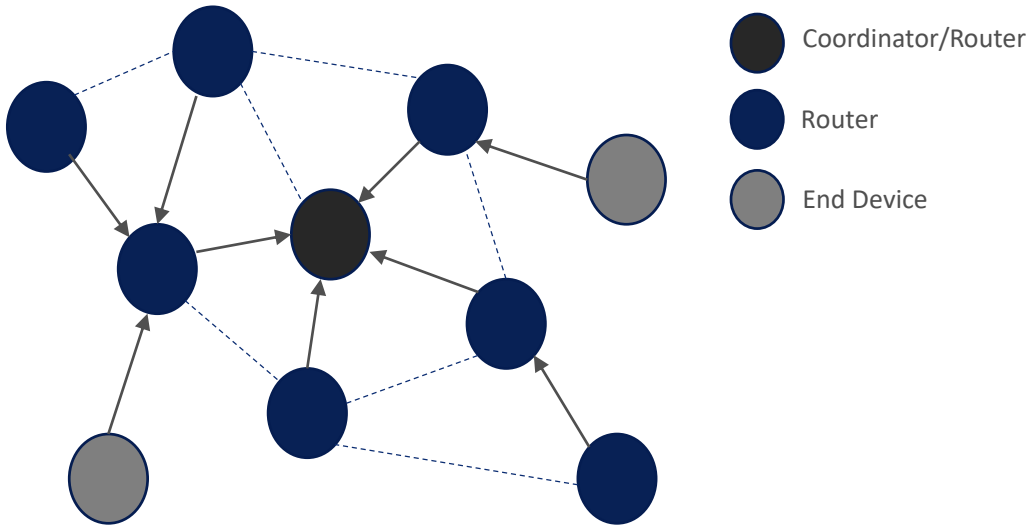
3. SDK development

Basic concepts – Device types

- Coordinator
  - Create central network
  - Trust Center
  - Network Manager
  - Routing function
- Router
  - Create Distribute network
  - Routing function
- End Device
  - Lower power consumption

Basic concepts – Network types

- Network topology
  - Mesh network
  - End devices send and receive data through the parent node
- Network type
  - Centralized network
    - Network created by the coordinator
  - Distributed network
    - Network created by routing node
    - Network created by TouchLink method



Basic concepts – Protocol specifications

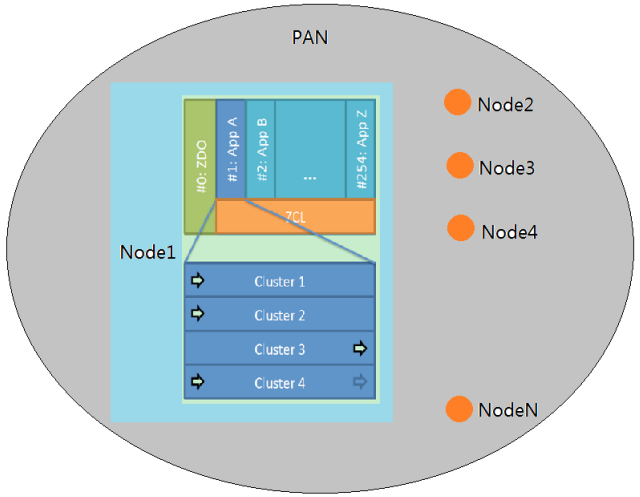
- **PAN ID**: Personal Area Network ID, a Zigbee network has only one PAN ID.
- **Channel**: The operating channel N of Zigbee is 11~26, and the relationship between operating frequency and channel is: 2405+(N-11)*5 MHz.
- **Node**: A physical device with a globally unique IEEE address and a unique short address in that PAN ID network after accessing to the network.
- **Endpoint**: Application port, a Node can contain more than one Endpoint.

  0 : ZDO port.

  1~240 : APP port.

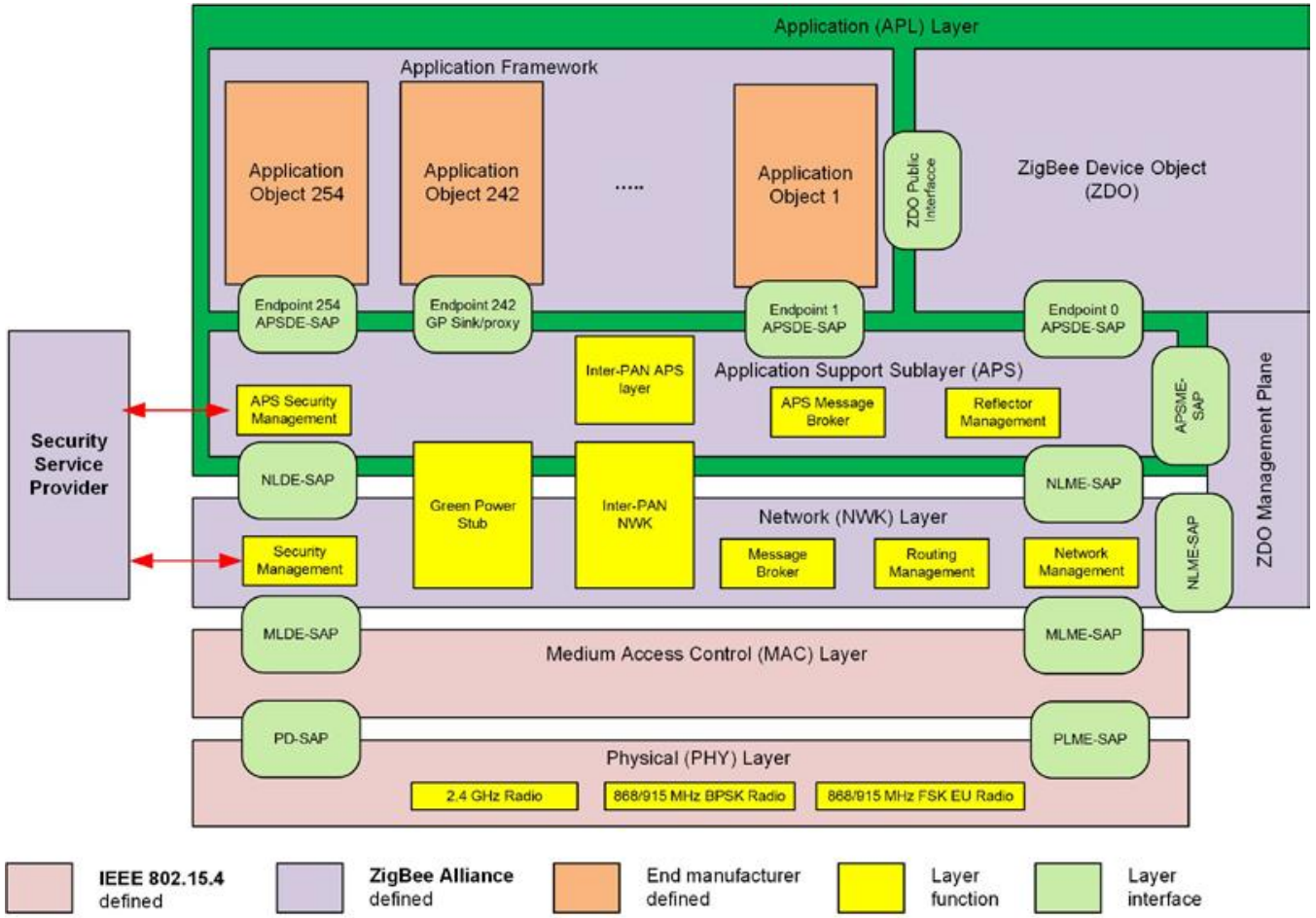
  240~254: port used by protocol, not for APP. For example, 242 is used for Green Power.

  255 : Broadcast port.
- **Cluster:** A specific application is composed of several different clusters, thus accomplishing different behaviors. ZCL defines a set of behavioral specifications that clusters need to follow.
- **Attribute:** ZCL defines not only the value of the attribute, but also the permission to operate on the attribute value.



Basic concepts – Protocol framework

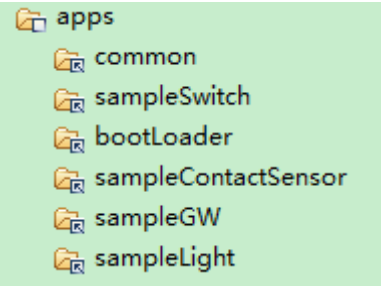
- Protocol framework
  - AF
  - APS
  - ZDO
  - NWK
  - MAC
  - PHY



Directory structure

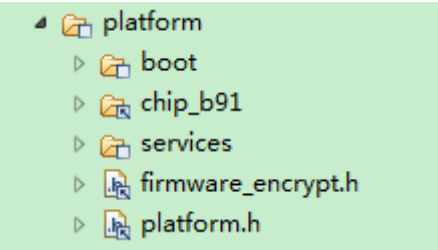
➢ Application directory

- /apps
  - /common: common function directory
    - main.c: main function
    - comm_cfg.h: version definition and operation mode configuration
  - /sampleGW: Gateway application sample
  - /sampleLight: Light application sample
  - /sampleSwitch: Switch application sample
  - /sampleContactSensor: Door magnet application sample
  - /bootLoader: bootloader

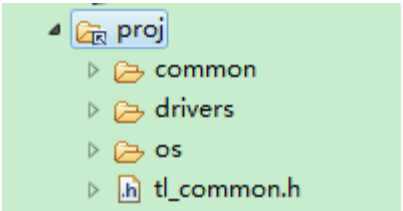


Directory structure

- Platform driver directory
  - /platform
    - /boot: boot file
    - /chip_xxxx: chip driver file
    - /services: interrupt service function file
    - platform.h: header file
    - firmware_encrypt.h: encryption solution based on FLASH UID

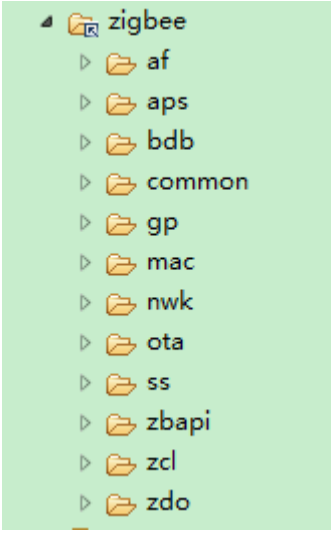


- Project directory
  - /proj
    - /common: common function file, such as strings, claims, and etc.
    - /drivers: abstract layer driver file
    - /os: task management file
    - tl_common.h: header file



Directory structure

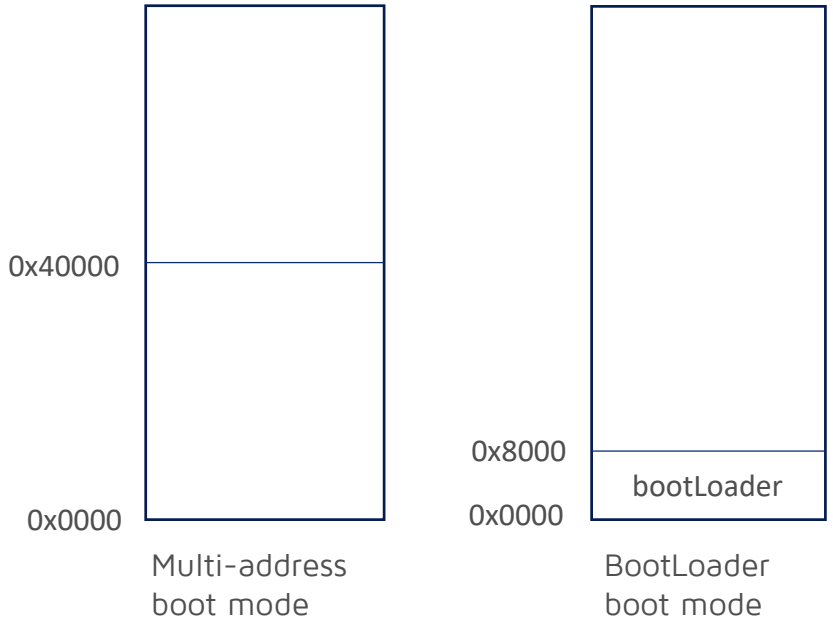
- Zigbee protocol stack directory
  - /zigbee
    - /af: application framework layer file
    - /aps: application support sub-layer file
    - /bdb: Basic Device Behavior file
    - /common: protocol configuration file
    - /gp: Green Power file
    - /mac: Media Access Control layer file
    - /nwk: network layer file
    - /ota: Over The Air upgrade file
    - /ss: Security Service file
    - /zbapi: Zigbee common interface file
    - /zcl: ZCL layer file
    - /zdo: device object layer file



Development guide – Boot mode

➢ Operation mode selection

- Supports two operation modes
  - Multi-address boot mode (0x0, 0x40000)
    - ◆ Advantages: fast boot; no need to transfer image again after OTA
    - ◆ Disadvantages: image can only be located at address 0x0 or 0x40000, which will cause discontinuity in flash space allocation
  - BootLoader boot mode
    - ◆ Advantages: image location can be defined at will which is highly flexible
    - ◆ Disadvantages: consumes bootloader code space; slow boot, needs to transfer image after successful OTA
- Modification method (comm_cfg.h)
  - #define BOOT_LOADER_MODE 0//1

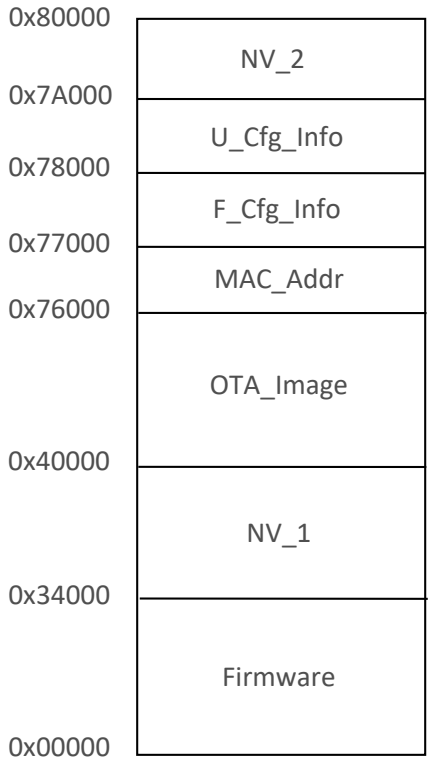


Development guide – Hardware selection

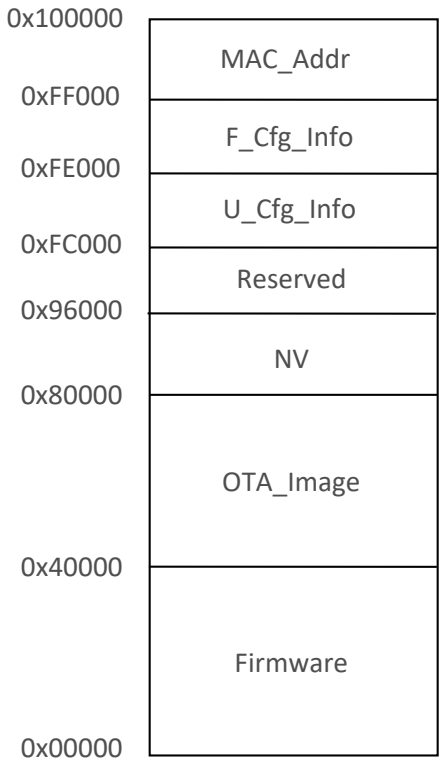
- Chip type selection
  - Modification method (version_cfg.h)
    - #define CHIP_TYPE TLSR_9518
- Target board selection
  - Modification method (app_cfg.h)
    - #define BOARD BOARD_9518_DONGLE// BOARD_9518_EVK

Development guide – Flash allocation

- Multi-address boot mode Flash allocation



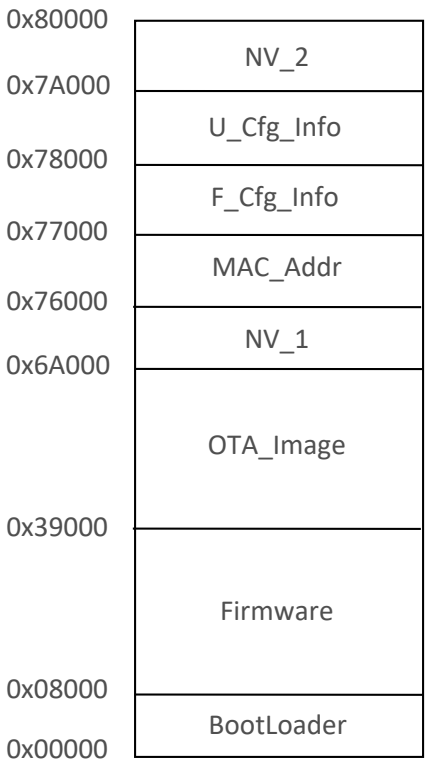
512k Flash



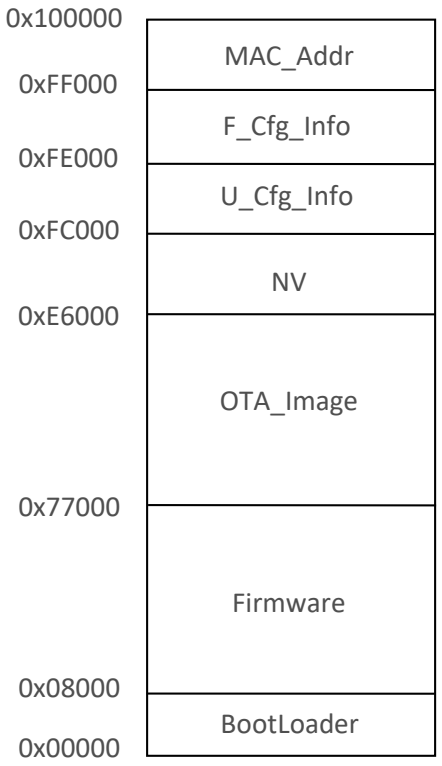
1M Flash

Development guide – Flash allocation

➢ bootLoader boot mode Flash allocation



512k Flash



1M Flash

Development guide – Flash allocation

➢ Flash allocation description

- MAC_Addr: The MAC address is stored in the starting 8 bytes of this area, it is pre-written when the chip is out of factory, please erase it with careful consideration.

  The system will check the MAC address information after boot, and if it is found to be 0xFF for 8 bytes, a MAC address will be generated randomly to backfill this area.

- F_Cfg_Info: Factory configuration parameter information.

  The chip will be pre-written with some calibration parameter information at the factory, such as frequency bias calibration, ADC calibration, etc. Please erase it with careful consideration.

- U_Cfg_Info: Information about user configuration parameters. For example, install code and factory reset flags are stored in this area.

- NV(NV_1,NV_2): Network information storage area from which the network parameters are read and restored after reboot.

- Firmware and OTA_Image: firmware storage area.

- BootLoader: it is used to store the bootloader code when using bootloader boot mode.

Development guide – Printout

➢ Print and debug

We use GPIO to simulate the TX function of serial port and use printf() function to realize printout. Users can change to the appropriate I/O port at will, without taking up hardware serial port resources.

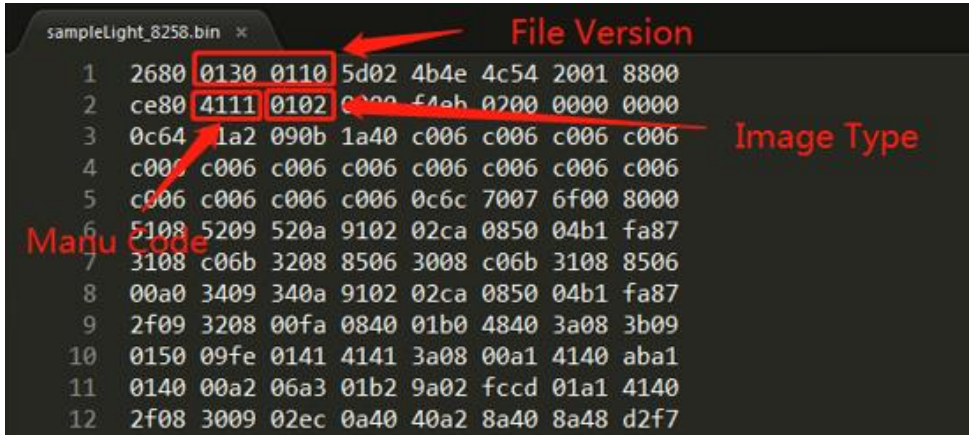
- Print enable (app_cfg.h)
  - #define UART_PRINTF_MODE 1
- Print port configuration (board_xx.h)
  - #define DEBUG_INFO_TX_PIN GPIO_PC4
  - #define BAUDRATE 1000000//1M

Development guide – Version management

➢ APP version management

There is a version_cfg.h file in each demo directory to manage the app version, which can effectively prevent the risk of being bricked when upgrading due to unmatched firmware and chip during OTA.

- Manufacturer code
  - #define MANUFACTURER_CODE_TELINK 0x1141
- Firmware type
  - #define IMAGE_TYPE ((CHIP_TYPE << 8) | IMAGE_TYPE_SWITCH)
- File version
  - #define FILE_VERSION ((APP_RELEASE << 24) | (APP_BUILD << 16) | (STACK_RELEASE << 8) | STACK_BUILD)



Development guide – Storage management

➢ Dynamic memory management

- Memory request
  - ev_buf_allocate()
- Memory release
  - ev_buf_free()
- The memory pool management implemented using number arrays supports memory requests up to 142 bytes in length by default.

Users can modify the relevant configuration of the memory pool (ev_buffer.h) by themselves.

Development guide – Storage management

➢ NV management

- NV_MODULE
  - Each module occupies 2 or (2*n) sectors (1 sector = 4k flash)
  - It uses append-write method until one sector is full, moves the valid information to another sector, and then erases the contents of the previous sector.
- NV_ITEM
  - Each item is a data storage unit
  - A module can consist of many entries
- Format
  - sector info + item index + item content
- API interface
  - nv_flashWriteNew()
  - nv_flashReadNew()

```
/*****************************************************************************************************
 * Store zigbee information in flash.
 *      Module ID                |           512K Flash                |           1M Flash                |
 * -----------------------------|-----------------------------------|-----------------------------------|
 * NV_MODULE_ZB_INFO            |       0x34000 - 0x36000           |       0x80000 - 0x82000           |
 * NV_MODULE_ADDRESS_TABLE      |       0x36000 - 0x38000           |       0x82000 - 0x84000           |
 * NV_MODULE_APS                |       0x38000 - 0x3a000           |       0x84000 - 0x86000           |
 * NV_MODULE_ZCL                |       0x3a000 - 0x3c000           |       0x86000 - 0x88000           |
 * NV_MODULE_NWK_FRAME_COUNT    |       0x3c000 - 0x3e000           |       0x88000 - 0x8a000           |
 * NV_MODULE_OTA                |       0x3e000 - 0x40000           |       0x8a000 - 0x8c000           |
 * NV_MODULE_APP                |       0x7a000 - 0x7c000           |       0x8c000 - 0x8e000           |
 * NV_MODULE_KEYPAIR            |       0x7c000 - 0x80000           |       0x8e000 - 0x96000           |
 *                              |   *16K - can store 127 nodes      |   *32K - can store 302 nodes      |
 * NV_MAX_MODULS
 */
```

Development guide – Task management

- Single task
  - API interface
    - Register task: TL_SCHEDULE_TASK(tl_zb_callback_t func, void *arg)
  - Notes on use
    - Execute only once, no priority
    - Avoid overflowing task queues due to pressing too many tasks at once
- Standing task
  - API interface
    - Register task: ev_on_poll(ev_poll_e e, ev_poll_callback_t cb)
    - Enable task: ev_enable_poll(ev_poll_e e, ev_poll_callback_t cb)
    - Suspend task: ev_disable_poll(ev_poll_e e, ev_poll_callback_t cb)
  - After starts, it keeps executing in the main loop.

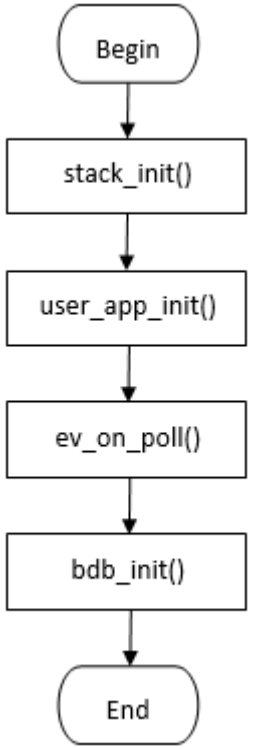
Development guide – Task management

➢ Software timed task

- API interface
  - Register task: TL_ZB_TIMER_SCHEDULE(ev_timer_callback_t func, void *arg, u32 t_ms)
  - Cancel task: TL_ZB_TIMER_CANCEL(ev_timer_event_t **evt)
- Notes on use
  - Avoid using TL_ZB_TIMER_CANCEL() in the interrupt function
  - In the callback function
    - Do not use TL_ZB_TIMER_CANCEL() to cancel its own task
    - The return value is less than 0 means the task will be executed only once and will be logged out automatically after execution.
    - The return value is equal to 0 means that after the task is executed, the timed task is still started using the time parameter registered.
    - The return value greater than 0 means that the task is executed and the return value is used as a new time parameter to start the timed task.

Development guide – Initialization process

- user_init(): Application layer initialization process
  - stack_init()
    - zb_init(): protocol stack initialization
    - zb_zdoCbRegister(): register protocol stack callback function
  - user_app_init()
    - af_endpointRegister(): register port descriptor information and message handling callback
    - zcl_init(): Initialize ZCL basic command
    - zcl_register(): register the cluster processing functions required by the application layer
  - ev_on_poll(): register user polling event
  - bdb_init(): initialize and start the BDB process



Development guide – Initialization process

- void zb_zdoCbRegister(zdo_appIndCb_t *cb):

  Register protocol stack callback functions

  - Callback functions
    - zdpStartDevCnfCb
    - zdpResetCnfCb
    - zdpDevAnnounceIndCb
    - zdpLeaveIndCb
    - zdpLeaveCnfCb
    - zdpNwkUpdateIndCb
    - zdpPermitJoinIndCb
    - zdoNlmeSyncCnfCb
    - zdoTcJoinIndCb

```
const zdo_appIndCb_t appCbLst = {
    bdb_zdoStartDevCnf,
    NULL,
    sampleGW_devAnnHandler,
    sampleGW_leaveIndHandler,
    sampleGW_leaveCnfHandler,
    sampleGW_nwkUpdateIndicateHandler,
    NULL,
    NULL,
    sampleGW_tcJoinIndHandler,
};
```

Development guide – Initialization process

- af_endpointRegister(): register port descriptor information and message handling callback
  - u8 ep, application port
  - af_simple_descriptor_t *simple_desc, port descriptor information
  - af_endpoint_cb_t rx_cb, the callback function for receiving messages, usually registered as zcl_rx_handler()
  - af_dataCnf_cb_t cnf_cb, the confirmation callback function for sending messages

```
typedef struct{
    u16 app_profile_id;         //APP profile ID specifies the profile which supported on this EP.
    u16 app_dev_id;             //APP DEV ID specifies the device description supported on this EP.

    u8  endpoint;               //end-point num of the simple descriptor 1 ~ 240
    u8  app_dev_ver:4;          //APP DEV version specifies the version of the device description supported
    u8  reserved:4;             //Reserved
    u8  app_in_cluster_count;   //The number of input clusters supported on this EP
    u8  app_out_cluster_count;  //The number of output clusters supported on this EP

    u16 *app_in_cluster_lst;    //Input cluster list address
    u16 *app_out_cluster_lst;   //Output cluster list address
}af_simple_descriptor_t;
```

Development guide – Initialization process

- zcl_register(): register the cluster processing functions required by the application layer
  - u8 endpoint, application port
  - u8 clusterNum, the number of supported cluster
  - zcl_specClusterInfo_t *info, registry of clusters and attributes
    - u16 clusterId, cluster ID
    - u16 manuCode, manufacturer code
    - u16 attrNum, the number of attributes supported by the cluster
    - const zclAttrInfo *attrTbl, attribute table
    - cluster_registerFunc_t func, pointer of cluster registration function
    - cluster_forAppCb_t appCb, callback function of cluster message

```
typedef struct {
    u16 clusterId;
    u16 manuCode;
    u16 attrNum;
    const zclAttrInfo_t      *attrTbl;
    cluster_registerFunc_t   clusterRegisterFunc;
    cluster_forAppCb_t       clusterAppCb;
} zcl_specClusterInfo_t;
```

Development guide – Initialization process

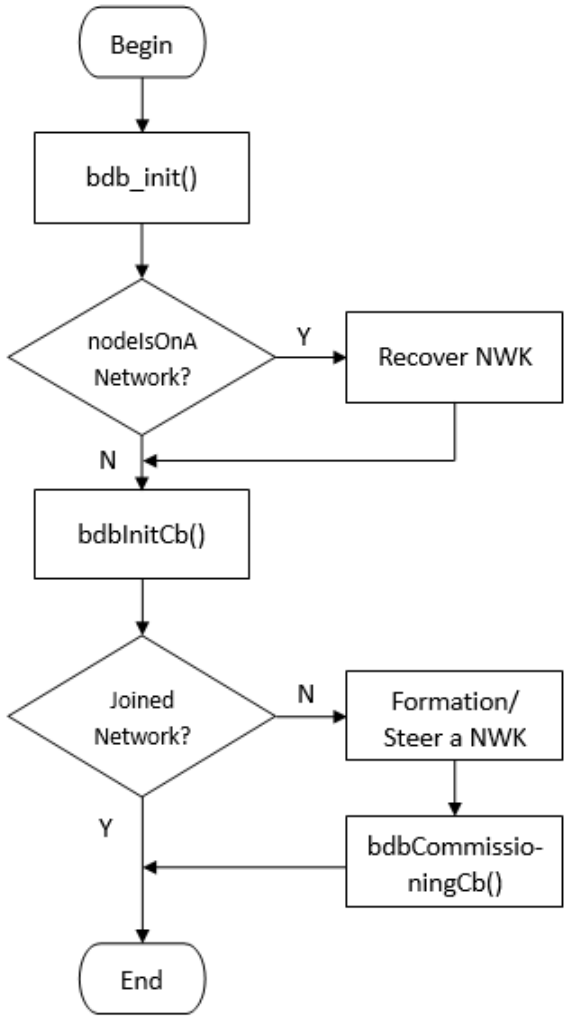
- zcl_specClusterInfo_t *info, registry of clusters and attributes

```
/**
 *  @brief Definition for simple GW ZCL specific cluster
 */
zcl_specClusterInfo_t g_sampleGwClusterList[] =
{
    {ZCL_CLUSTER_GEN_BASIC,                     ZCL_BASIC_ATTR_NUM,     basic_attrTbl,      zcl_basic_register,     sampleGW_basicCb},
    {ZCL_CLUSTER_GEN_IDENTIFY,                  ZCL_IDENTIFY_ATTR_NUM,  identify_attrTbl,   zcl_identify_register,  sampleGW_identifyCb},
    {ZCL_CLUSTER_GEN_GROUPS,                    0,                      NULL,               zcl_group_register,     sampleGW_groupCb},
    {ZCL_CLUSTER_GEN_SCENES,                    0,                      NULL,               zcl_scene_register,     sampleGW_sceneCb},
#ifdef ZCL_DOOR_LOCK
    {ZCL_CLUSTER_CLOSURES_DOOR_LOCK,            0,                      NULL,               zcl_doorLock_register,  &sampleGW_doorLockCb},
#endif
#ifdef ZCL_TEMPERATURE_MEASUREMENT
    {ZCL_CLUSTER_MS_TEMPERATURE_MEASUREMENT,    0,                      NULL,               zcl_temperature_measurement_register,   NULL},
#endif
#ifdef ZCL_IAS_ZONE
    {ZCL_CLUSTER_SS_IAS_ZONE,                   0,                      NULL,               zcl_iasZone_register,   &sampleGW_iasZoneCb},
#endif
#ifdef ZCL_POLL_CTRL
    {ZCL_CLUSTER_GEN_POLL_CONTROL,              0,                      NULL,               zcl_pollCtrl_register,  &sampleGW_pollCtrlCb},
#endif
};
```

Development guide – BDB process

➢ BDB process

- bdb_init(): initialization checks whether joined a network
  - If yes, first restore the network, then call back to bdbInitCb()
  - If not, call back to bdbInitCb(), users determine the next action
- bdbInitCb(): users obtain the results of bdb initialization by this function and determine the next actions, such as creating a network or joining a network.
  - bdb_networkFormationStart(): create a network
  - bdb_networkSteerStart(): search and join a network
- bdbCommissioningCb(): users obtain the results of Commissioning by this function.



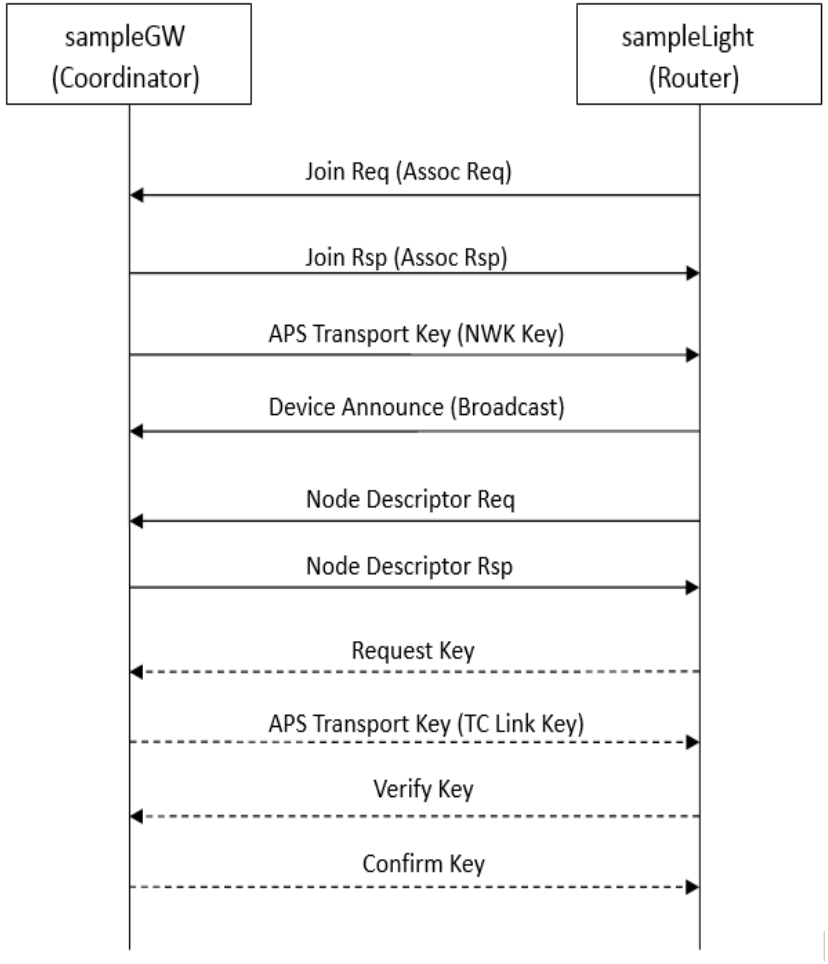
Development guide – Network security

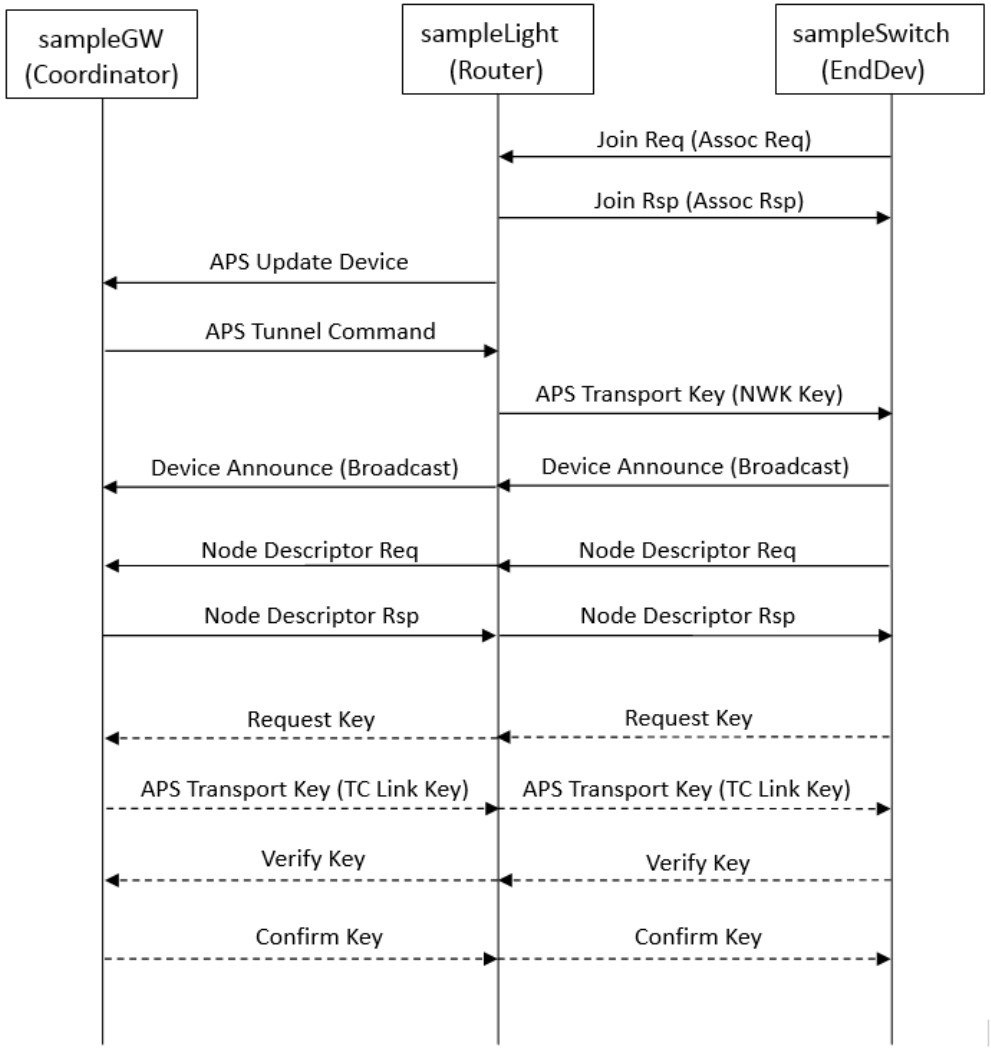
➢ Network security

- Default TCLK: The Global Trust Center Link Key defined by the Alliance
  - {5a6967426565416c6c69616e63653039}
- Install Code: The Unique Trust Center Link Key derived by MMO-Hash algorithm
  - zb_pre_install_code_load(), the node reads the install code from Flash
  - zb_pre_install_code_store(), the gateway invokes this interface to write the MAC address and install code of the node to the gateway
- Pre-configured NWK Key: pre-configured network key
  - zb_preConfigNwkKey()

Development guide – Network access process

- Flowchart of network access





Development guide – Data sending and receiving

Data sending

- Use ZCL command interface
  - read/write/report and other basic commands
  - cluster command
- Use AF Data sending interface
  - af_dataSend()
    - u8 srcEp, source port
    - epInfo_t *pDstEpInfo, target port information
    - u16 clusterId, cluster ID
    - u16 payloadLen, data length
    - u8 *payload, message address
    - u8 *apsCnt, the aps counter of the current message

Data receiving

- Use af_endpointRegister() to register the message receiving and handling function
  - zcl_rx_handler(void *pData)

```
typedef struct apsdeDataInd_s{
    aps_data_ind_t indInfo;
    u16 asduLen;
    u8  asdu[];
}apsdeDataInd_t;
```

Development guide – Low power management

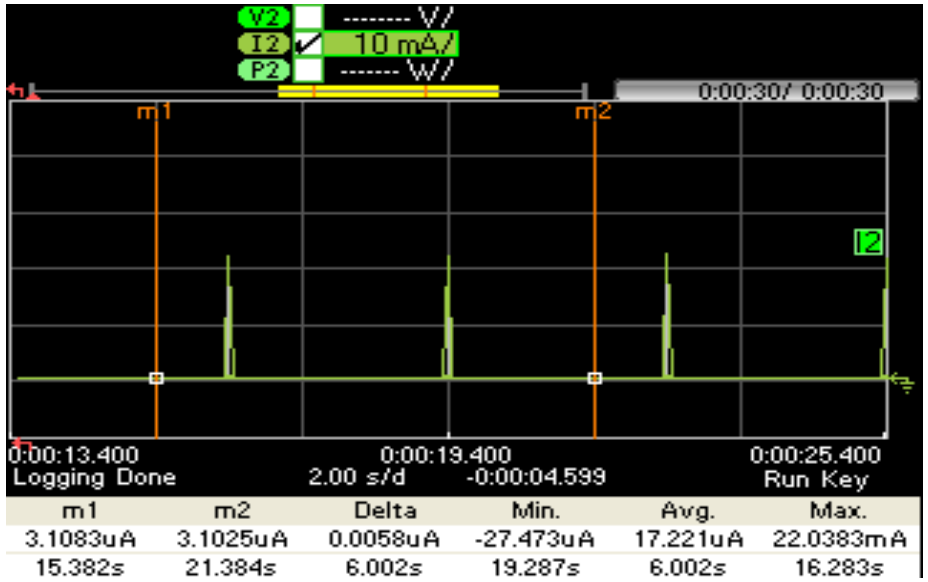
➢ Lower power management

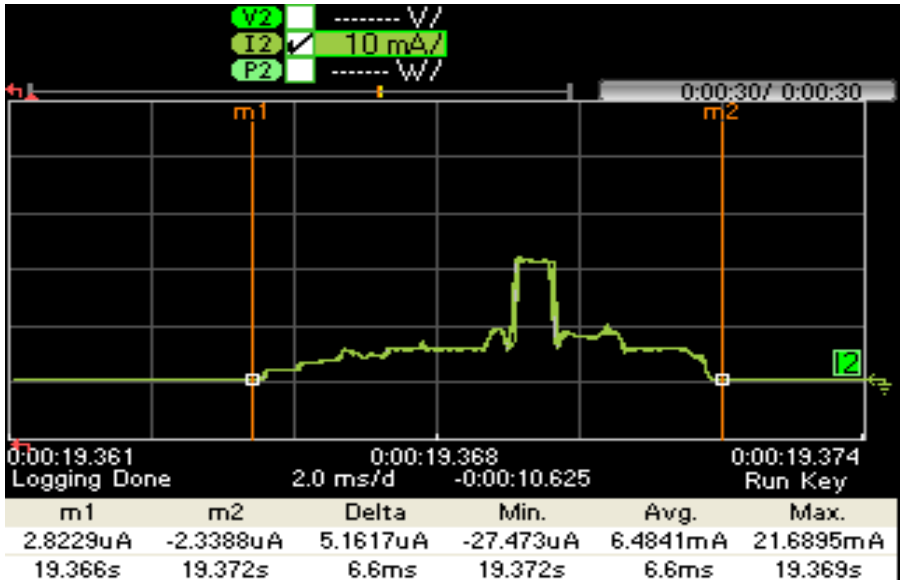
- Hibernation interface
  - drv_pm_lowPowerEnter()
- Two-level hibernation
  - suspend/deep with retention mode (with timed tasks)
  - deep mode (without timed tasks)
- Wake-up method
  - Timer wake-up
    - ◆ Use with software timed tasks
  - Button wake-up
    - ◆ drv_pm_wakeupPinConfig()

Development guide – Low power management

- 8258 dongle, deep sleep with 32k ram retention

poll rate(s)	1	3	5	10
sampling time(s)	6	6	15	20
wakeup time(ms)	6.9	6.6	6.6	7
avg current(uA)	48.45	17.22	12.04	7.64

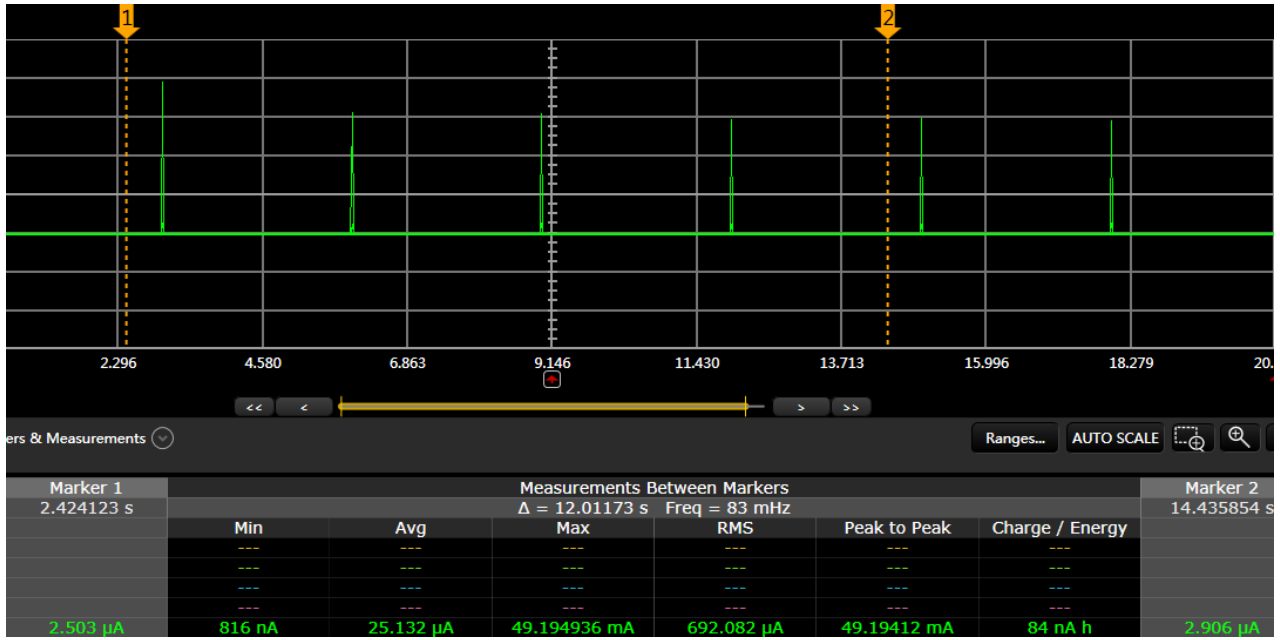


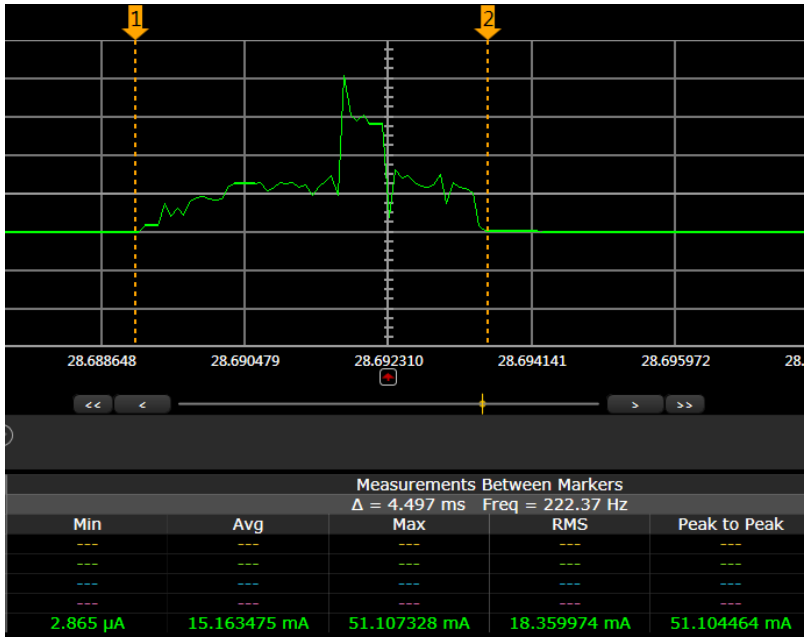


Development guide – Low power management

➢ B91 dongle, deep sleep with 64k ram retention

poll rate(s)	1	3	10
sampling time(s)	6	12	20
wakeup time(ms)	4.2	4.5	4.5
avg current(uA)	64.8	25.1	10.1





Development guide – Network parameter configuration

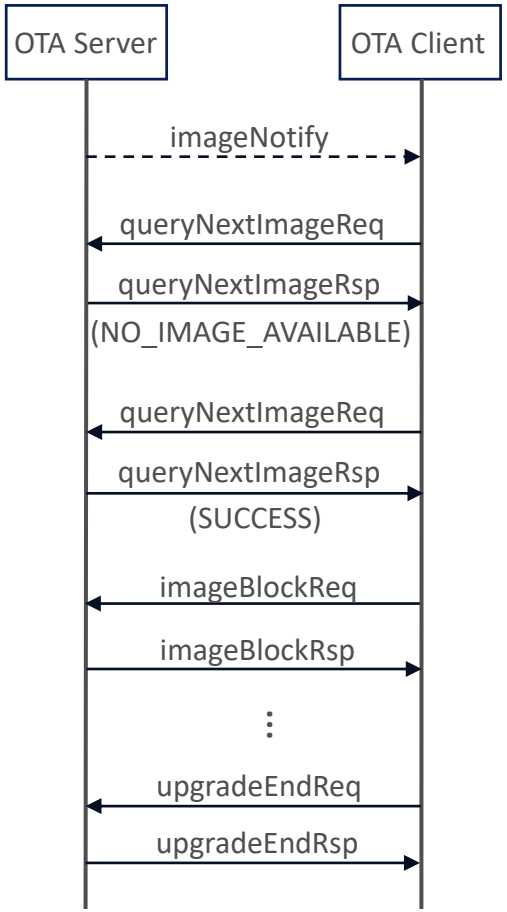
➢ Network parameter configuration

- APS_BINDING_TABLE_NUM, the number of binding tables
- APS_GROUP_TABLE_NUM, the number of grouped tables
- TL_ZB_NWK_ADDR_MAP_NUM, the number of address mapping tables
- TL_ZB_NEIGHBOR_TABLE_NUM, the number of neighbor tables
- ROUTING_TABLE_NUM, the number of routing tables
- NWK_ROUTE_RECORD_TABLE_NUM, the number of routing record tables
- NWK_BRC_TRANSTBL_NUM, the number of broadcast tables
- NWK_ENDDEV_TIMEOUT_DEFAULT, end device keep-alive timeout time
- ZB_MAC_RX_ON_WHEN_IDLE, the state of the RF receiver when the end device is idle
- ZB_NWK_LINK_STATUS_PEROID_DEFAULT, the cycle time of the link status command

Development guide – OTA

➢ OTA upgrade

- OTA initialization
  - OTA device type: Server/Client
  - ota_init()
- Client starts OTA timing queries
  - ota_queryStart(u16 seconds)
- Server initiates OTA notifications
  - zcl_ota_imageNotifyCmdSend()
- OTA condition check
  - Manufacturer Code
  - Image Type
  - File Version



Development guide – OTA

- OTA Image
  - Contains OTA Header, a ZCL OTA compliant zigbee file for over-the-air upgrades to remote devices
  - Extension: AES-128 encryption
- Generating method
  - Copy "sampleLight_9518.bin" to the "zigbee_ota_tool_v2.2.exe" folder
  - Code conversion
    - ◆ Double-click "zigbee_ota_tool_v2.2.exe" and follow the instructions
    - ◆ Or use the command line "./zigbee_ota_tool_v2.2.exe [arg1] [arg2]"

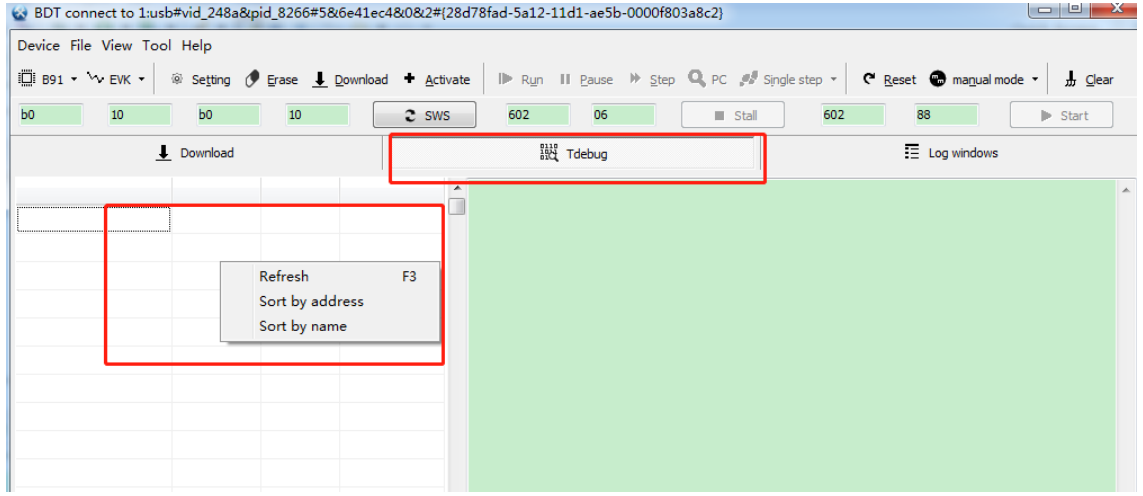
    [arg1]: file name, e.g. "sampleLight_9518.bin"

    [arg2]: no parameter means no encryption; input this parameter means use this parameter to encrypt, input format like "00112233445566778899AABBCCDDEEFF"
- Output file
  - 1141-0201-10013001-sampleLight_9518.zigbee

4. Debugging method

Debugging method

- Use printf() function to print
  - printf("test: %d, %x\r\n", val_1, val_2)
- Use BDT tool to view the data in RAM directly
  - Click "Tdebug"
  - Make sure the current file is the same as the firmware the chip is running
  - In the left side of display area, right-click -> "Refresh"
  - The BDT tool will automatically parse all global variables according to the mapping file (xx.lst / objdump.txt) and display the variable names and values in the display bar.
  - It should be noted that when the device is in hibernation, the data in RAM is not available to access.



Variable Name	Addr	Len	Value
g_bdbCommissionSet	02308	18	...
g_gpCtx	0231c	23	...
g_zbDemoBdbCb	02334	16	...
g_zcl_basicAttrs	02344	54	...
gpEpCbs	0237c	12	...
myUartDriver	02388	9	...
sampleGW_otaInfo	02394	12	...
zclGpAttr_gpLinkKey	023a0	16	...
g_zllTouchLink	023b0	73	...
ndt.lto_priv.185	023fc	13	...
pmcd	0240c	41	...
dGpStubHandle	02435	1	00000000
flash_cnt	02436	1	00000000

5. Demo test

Demo test

➢ Preliminary preparation

Here is an example of TLSR9518.

If you use other chip model, you can modify CHIP_TYPE and BOARD definition and compile to generate the corresponding firmware.

- Development boards
  - USB Dongle, as Gateway (Coordinator)
  - EVK Board 1, as Light (Router)
  - EVK Board 2, as Switch (End Device)
- Firmware
  - sampleGW_9518_dongle.bin
  - sampleLight_9518_evk.bin
  - sampleSwitch_9518_evk.bin

Demo test

Burning connection

- USB Dongle
  - J56
    - 3V3
    - SWS
    - GND
- EVK Board
  - J51
    - VBAT (jumper cap: remove during burning, insert back after burning)
  - J56
    - SWS
    - GND

Demo test

➢ Create network

- Power up Gateway node (USB Dongle), the red LED is steady on.
  - If it is a new device, a network will be created and permit join on (180s) will be turned on to allow the device to join the network and the green LED will be on.
  - If it is a device that has already established a network, the network will be restored and the state of permit join can be turned on or off by pressing key SW7 (green LED on means status is on, green LED off means status is off).

➢ Join network (under the condition that Gateway's permit join is on, as Gateway's green LED is on)

- Power up Light node (EVK Board 1), the red LED is on.
  - If it is a new device, the network join will be started automatically and the green LED is on after successful join.
  - If it is a device that has been in the network, the network will be restored.
- Power up Switch (EVK Board 2)
  - If it is a new device, the network join will be started automatically and the green LED will blink after successful join.
  - If it is a device that has been in the network, the network will be restored.

Demo test

➢ Button control function

- Gateway
  - SW2: turn on or off the broadcast toggle commend with the period of 1 seconds. (If Light node has joined network successfully, its red LED will be on and off accordingly)
  - SW7: turn on or off permitting device to join network. (permit join, green LED on means this switch is on, green LED off means this switch is off)
- Light
  - SW2: short press to switch the status of light (red LED); long press for 5 seconds to leave network, the red LED flashes three times after successfully off-network.
  - SW3: turn on or off permitting device to join network. (permit join, green LED on means this switch is on, green LED off means this switch is off)
- Switch
  - SW2: short press, broadcast toggle command. (If Light node has joined network successfully, its red LED will be on and off accordingly) Long press for 5 seconds to leave network, the red LED flashes three times after successfully off-network.
  - SW3: broadcast move to level command. (If Light node has joined network successfully, its red LED brightness will change accordingly)

Application extensions

- Dual-mode applications
  - Zigbee + BLE (concurrent mode)
    - Time Division Multiplexing (TDM) RF module for real-time switching of two modes
    - Supports Zigbee Coordinator, Router and End Device (low power device)
    - Supports BLE Master and Slave
  - Zigbee + SIG Mesh (switch mode)
    - For un-networked device, actively detect current valid network (Zigbee or SIG Mesh) and then join the network
    - For networked device, keep the original network state until factory new reset
- OS support
  - freeRTOS