BLZ User Guide

Version 1.2

2024-03-19

Revision History

Revision	**Date**	**Details**
1.0	2024-01-02	Version 1.0: first release.
1.0	2024-01-18	1. reorg the blz uart implementation, app layer must overridden the uart operations. 2. add ota-server demo in host application. 3. add ncp backup and restore demo.
1.0	2024-02-07	1. add CLI command to dump source route table. 2. add CLI command to send ZCL request.
1.1	2024-02-20	Add NCP OTA.
1.2	2024-03-19	1.Add command to to get ncp version, dump neighbor table, route table. 2. Change source route table size to uint16_t.

Table of Contents

1. BouffaloLab Zigbee Serial Protocol

BLZ is BouffaloLab’s Zigbee serial protocol used by host application (HOST) to interact with BouffaloLab’s Zigbee network Co-Processor (NCP).

1.1 Frame Format

The BLZ frame is composed of a Start Byte, a Frame Control Byte, a Sequence Byte, a 2-byte Frame ID, a Data Field with 0-n bytes depending on Frame ID, a 2-byte CRC, and a Stop Byte.

Byte No.	**1**	**2**	**3**	**4**	**5**	**0-n**	**n+5**	**n+6**	**n+7**
Description	Start	Frame Control	Sequence	Frame ID		Data	CRC		Stop

Table 1.1 Frame Format

1.1.1 Start Byte

The Start Byte, 0x42, is first byte of a frame.

1.1.2 Frame Control Byte

Currently 2-bits are reserved, other 6 bits are reserved for further usage.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Debug	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	ReTx

Table 1.2 Frame Control Format

- Bit7: debug mode, only for BLZ development.
- Bit0: indicates the this frame is a re-transmission frame, due to no ACK is received in the sender side. To ensure the transmission reliability, each BLZ data frame (except ACK, ERROR, RESET and RESET ACK frames) requires an acknowledgment from remote peer, the default acknowledgment timeout is 3 seconds and the re-transmission threshold is 3 times. Once the threshold is reached, NCP or Host will enter ERROR state and a reset or reconnect operation is required in order to recover the communication.

1.1.3 Sequence Byte

Bit7	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**
Reserved	Rx Sequence			Reserved	Tx Sequence		

Table 1.3 Rx/Tx Sequence Format

Rx Sequence is the last frame received from remote peer, and it serves as acknowledgment to the remote peer in the next frame sent to remote peer.

Tx Sequence is the last frame sent to remote peer, and will be used by remote peer as Rx Sequence to acknowledge this tx frame.

1.1.4 Frame ID

The 2-bytes Frame ID is in little-endian mode, refer to the “Frames” section for different Frame ID definitions.

1.1.5 Data

The Data field is depend on different Frame IDs, refer to the details along the different Frame ID definitions.

1.1.6 CRC

The 2-bytes CRC is in big-endian mode, and is computed on all bytes in a frame preceding the CRC field, excluding the start byte and bytes added by byte stuffing. The standard CRC-16/CCITT-FALSE (x^16 + x^12 + x^2 + 1) hash function is used and is initialized to 0xFFFF.

1.1.7 Stop Byte

The Stop Byte, 0x4C, marks the end of a frame.

1.2 Reserved Bytes and Byte Stuffing

1.2.1 Reserved Bytes

The following table lists the reserved bytes that used in this serial protocol.

Byte	**Description**
0x42	Start Byte: the first byte of a frame.
0x4C	End Byte: the end byte of a frame.
0x07	Escape Byte: escape reserved byte to allow send frame data contains reserved byte values, known as “byte stuffing”

Table 1.4 Reserved Bytes

1.2.2 Byte Stuffing

Step 1: send escape byte.
Step 2: XOR the byte to be sent with **0x10** and send this modified byte.

2. Frames

2.1 BLZ_FRAME_ID_ACK

Acknowledge to the received frame, no Data field in this frame. ACK frame doesn't require a subsequent acknowledgment.

2.2 BLZ_FRAME_ID_ERROR

As an error notification to the remote peer, e.g. from NCP to Host, after Host received this notification, Host can inform the NCP to reset, in order to recover the communication. ERROR frame doesn't require a subsequent acknowledgment.

Command Parameters:

Name	**Type**	**Description**
Error Code	uint8_t	Error code

2.3 BLZ_FRAME_ID_RESET

Used by Host to inform NCP to reset. RESET frame doesn't require a subsequent acknowledgment.

2.4 BLZ_FRAME_ID_RESET_ACK

Used by NCP to inform Host that NCP is ready, then Host can start normal operations. RESET_ACK frame doesn't require a subsequent acknowledgment.

2.5 BLZ_FRAME_ID_GET_VALUE

Reads a value from NCP.

Command Parameters:

Name	**Type**	**Description**
Value Id	uint8_t	Indicates which value to read.

Response Parameters:

Name	**Type**	**Description**
Status	uint8_t	Indicates whether the operation is success or not.
Value Length	uint8_t	Valid only if Status=BLZ_SUSS.
Value	bytes	Valid only if Status=BLZ_SUSS.

2.6 BLZ_FRAME_ID_SET_VALUE

TBD

2.7 BLZ_FRAME_ID_ADD_ENDPOINT

Add a dynamic endpoint to NCP.

Command Parameters:

Name	**Type**	**Description**
Endpoint Id	uint8_t	Endpoint Id.
Profile Id	uint16_t	Profile Id.
Device Id	uint16_t	Device Id.
Endpoint Flags	uint8_t	Reserved.
Input Cluster Count	uint8_t	Server side cluster count.
Output Cluster Count	uint8_t	Client side cluster count.
Input Cluster List	uint16_t[]	Server cluster list.
Output Cluster List	uint16_t[]	Client cluster list.

Response Parameters:

Name	**Type**	**Description**
Status	uint8_t	Indicates whether the operation is success or not.

2.8 BLZ_FRAME_ID_FORM_NETWORK

Form a Zigbee network with specified network parameters.

Command Parameters:

Name	**Type**	**Description**
Extended Pan Id	uint64_t	Extended Pan Id. The value of NCP’s MAC address will be used if set this parameter to 0.
Pan Id	uint16_t	Pan Id.
Channel	uint8_t	Zigbee network channel, from 11 to 26.

Response Parameters:

Name	**Type**	**Description**
Status	uint8_t	Indicates whether the operation is success or not.

2.9 BLZ_FRAME_ID_LEAVE_NETWORK

Inform NCP to reset its network.

Command Parameters:

No parameter required.

Response Parameters:

Name	**Type**	**Description**
Status	uint8_t	Indicates whether the operation is success or not.

2.10 BLZ_FRAME_ID_PERMIT_JOINING

Open or close Zigbee network.

Command Parameters:

Name	Type	Description
Duration	uint8_t	Network open duration. Will close network if duration is 0.

Response Parameters:

Name	Type	Description
Status	uint8_t	Indicates whether the operation is success or not.

2.11 BLZ_FRAME_ID_GET_NETWORK_PARAMETERS

Get network informations from NCP.

Command Parameters:

No parameter required.

Response Parameters:

Name	Type	Description
Status	uint8_t	Indicates whether the operation is success or not.
Node Type	uint8_t	Fixed to 0x00 (Zigbee Coordinator).
Extended Pan Id	uint64_t	Extended Pan Id.
Pan Id	uint16_t	Pan Id.
Tx Power	uint8_t	Tx power, default value is 14dBm.
Channel	uint8_t	Current network channel.
Network Manager	uint16_t	Network manager address.
Network Update Id	uint8_t	Network Update Id.
Channel Mask	uint32_t	Channel Mask.

2.12 BLZ_FRAME_ID_GET_NWK_SECURITY_INFOS

Get network security related information from NCP.

Command Parameters:

No parameter required.

Response Parameters:

Name	Type	Description
Status	uint8_t	Indicates whether the operation is success or not.
Network Key	uint8_t[16]	Current Zigbee network key.
Network Outgoing frame counter	uint32_t	Current network outgoing frame counter.
Network Key Sequence	uint8_t	Network key sequence number.

2.13 BLZ_FRAME_ID_NETWORK_INIT

Resumes NCP’s network.

Command Parameters:

No parameter required.

Response Parameters:

Name	**Type**	**Description**
Status	uint8_t	Indicates whether the operation is success or not. If NCP is not in a network (e.g. doesn’t form a network yet), this operation will return fail.

2.14 BLZ_FRAME_ID_STACK_STATUS_CALLBACK

NCP’s Zigbee stack status callback to Host.

Command Parameters:

Name	**Type**	**Description**
Stack Status	uint8_t	NCP_NETWORK_UP (0x00): NCP resumed its network. NCP_NETWORK_DOWN (0xFF): NCP left its network. Others value indicates a error stack status.

Response Parameters:

No response to this frame.

2.15 BLZ_FRAME_ID_DEVICE_JOIN_CALLBACK

NCP’s Zigbee stack status callback to Host.

Command Parameters:

Name	**Type**	**Description**
Extended Address	uint64_t	Device’s mac address.
NodeId	uint16_t	Device’s network address.
Status	uint8_t	0x00: unsecured join or association join. 0x01: rejoin. 0x03: leave.

Response Parameters:

No response to this frame.

2.16 BLZ_FRAME_ID_SEND_APS_DATA

Send APS layer message from Host to NCP.

Command Parameters:

Name	**Type**	**Description**

Message Type	uint8_t	BLZ_MSG_TYPE_INDIRECT = 0x00,
		BLZ_MSG_TYPE_UNICAST = 0x01,
		BLZ_MSG_TYPE_MULTICAST = 0x02,
		BLZ_MSG_TYPE_BROADCAST = 0x03,

Destination Address	uint16_t	Address of destination.
Profile Id	uint16_t	Profile Id.
Cluster Id	uint16_t	Cluster Id.
Source Endpoint	uint8_t	Endpoint in NCP.
Description Endpoint	uint8_t	Endpoint of destination.

Tx Options	uint8_t	ZB_APS_TX_OPTIONS_SEC_EN_TRANS = 0x01,
		ZB_APS_TX_OPTIONS_USE_NWK_KEY = 0x02,
		ZB_APS_TX_OPTIONS_ACK_TRANS = 0x04,

Radius	uint8_t	Max of hops.
Message Tag	uint32_t	A value used to different requests in subsequent APS confirm.
Asdu length	uint8_t	APS payload length.
Asdu	uint8_t[]	APS payload.

Response Parameters:

Name	**Type**	**Description**
Status	uint8_t	Indicates whether the operation is success or not.

2.17 BLZ_FRAME_ID_APS_DATA_CONFIRM

APS Message sent confirm from NCP to Host.

Command Parameters:

Name	**Type**	**Description**
Profile Id	uint16_t	Profile Id.
Cluster Id	uint16_t	Cluster Id.
Destination Address	uint16_t	Address of APS message destination.
Source Endpoint	uint8_t	Source endpoint of APS message.
Destination Endpoint	uint8_t	Destination endpoint of APS message.

Message Type	uint16_t	BLZ_MSG_TYPE_INDIRECT = 0x00,
		BLZ_MSG_TYPE_UNICAST = 0x01,
		BLZ_MSG_TYPE_MULTICAST = 0x02,
		BLZ_MSG_TYPE_BROADCAST = 0x03,

Status	uint8_t	APS confirm status.
Message Tag	uint32_t	Message Tag passed to APS message.

Response Parameters:

No response to this frame.

2.17 BLZ_FRAME_ID_APS_DATA_INDICATION

APS Message received from NCP to Host.

Command Parameters:

Name	**Type**	**Description**
Profile Id	uint16_t	Profile Id.
Cluster Id	uint16_t	Cluster Id.
Source Address	uint16_t	Address of APS message sent from.
Destination Address	uint16_t	Address of APS message sent to.
Source Endpoint	uint8_t	Source endpoint.
Destination Endpoint	uint8_t	Destination endpoint.

Message Type	uint16_t	BLZ_MSG_TYPE_INDIRECT = 0x00,
		BLZ_MSG_TYPE_UNICAST = 0x01,
		BLZ_MSG_TYPE_MULTICAST = 0x02,
		BLZ_MSG_TYPE_BROADCAST = 0x03,

Last Hop LQI	uint8_t	Link quality from the device that NCP directly received this message from.
Last Hop RSSI	int8_t	RSSI when NCP received this message.
Asdu Length	uint8_t	APS payload length.
Asdu	uint8_t[]	APS payload

Response Parameters:

No response to this frame.

2.18 BLZ_FRAME_ID_GET_SOURCE_ROUTE_TABLE_COUNT

Get the total number of entries in source routing table.

Command Parameters:

No parameter required.

Response Parameters:

Name	**Type**	**Description**
Status	uint8_t	
Count	uint16_t	

2.19 BLZ_FRAME_ID_GET_SOURCE_ROUTE_TABLE_ENTRY

Get a source route table entry.

Command Parameters:

Name	**Type**	**Description**
Index	uint16_t	Index of source route table entry.

Response Parameters:

Name	**Type**	**Description**
Status	uint8_t	Status
NetworkAddress	uint16_t	Zigbee device’s Network address.
RelayCount	uint8_t	Number of relay devices to above network address.
RelayList	uint16_t	Relay devices.

2.20 BLZ_FRAME_ID_GET_ROUTE_TABLE_COUNT

Get the total number of entries in routing table.

Command Parameters:

No parameter required.

Response Parameters:

Name	**Type**	**Description**
Status	uint8_t	
Count	uint16_t	

2.21 BLZ_FRAME_ID_GET_ROUTE_TABLE_ENTRY

Get a route table entry.

Command Parameters:

Name	**Type**	**Description**
Index	uint16_t	Index of route table entry.

Response Parameters:

Name	**Type**	**Description**
Status	uint8_t	Status
DestinationAddress	uint16_t	
NextHop	uint16_t	
Status	uint8_t	Refer to “enum _zbRouteStatusAndFlags”.

2.22 BLZ_FRAME_ID_GET_NEIGHBOR_TABLE_COUNT

Get the total number of entries in neighbor table.

Command Parameters:

No parameter required.

Response Parameters:

Name	**Type**	**Description**
Status	uint8_t	
Count	uint16_t	

2.23 BLZ_FRAME_ID_GET_NEIGHBOR_TABLE_ENTRY

Get a neighbor table entry.

Command Parameters:

Name	**Type**	**Description**
Index	uint16_t	Index of neighbor table.

Response Parameters:

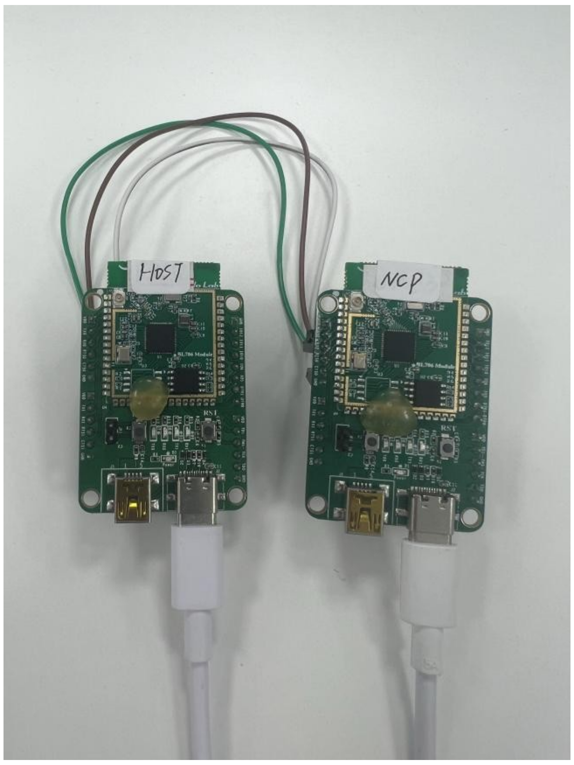
Name	**Type**	**Description**
Status	uint8_t	Status
Extended Address	uint64_t	
Network Address	uint16_t	
DeviceType	uint8_t	0x00: Zigbee coordinator 0x01: Zigbee router 0x02: Zigbee end device
RxOnWhenIdle	uint8_t	0x00: Receiver is off 0x01: Receiver is on
LinkQuality	uint8_t	
LinkCost	uint8_t	
OutgoingCost	uint8_t	
Age	uint8_t	

3. NCP and Host Demo Application Usage.

This section introduces how to setup/build/run the NCP and Host demo applications in the SDK.

3.1 Setup and Build

3.1.1 Hardware



Two BL706 EVB boards with both UART0 and UART1 enabled are used, as HOST and NCP respectively.

UART0 is used for the communication between HOST and NCP. UART1 is used for debugging and CLI. The USB Port in both HOST and NCP is using UART1. The 2M (2000000) baudrate is used in both NCP and HOST applications.

The following tables shows the default GPIO configuration for UART0 and UART1.

ID	TX	RX
UART0	14	15
UART1	18	19

3.1.2 Software

Application path: customer_app/bl702_demo_ncp_host/.

Build ncp application: ./genncp

Build host application: ./genhost

Flash to evb board: please use the provided dts file which enabled both UART0 and UART1.

3.2 Run

After flashing, reboot NCP first, then reboot Host. After Host device powered on, it will try connecting to NCP by sending RESET request to NCP and you can see the following print in NCP's serial terminal.

```
[BLZ]rx frameCtrl=0x00, frmSeq=0x01, rxSeq=0, txSeq=1, frameId=0003
[BLZ]curRx=6, curTx=4, frmReTx=0
[BLZ]blz_post_event: eventId=1
[NCP]Reboot NCP!
```

3.2.1 List CLI commands

In Host's serial terminal, send "help" command, and available CLI commands will be listed.

```
====User Commands====
zb_info : ncp informations
zb_nwk_init : start network
zb_nwk_form : form network
zb_nwk_leave : form network
zb_nwk_open : open permit join
zb_dump_source_route_table: dump source route table
zb_zdp_active_ep_req : send active endpoint request
zb_zdp_simple_desc_req : send simple discriptor request
zb_zcl_read_attr : send zcl read attribute request
zb_zcl_onoff : send zcl on/off/toggle request
```

3.2.2 Form network

In Host's serial terminal, send "zb_nwk_form 20 0 0xabd3" command, the following key prints indicate NCP has successfully formed a network:

```
[2024-01-10 19:45:15.190]: zb_nwk_form
[Host]Wrong number of args
[Host]Usage: zb_nwk_form <channel> <extPanId> <panId>
[Host] channel - <uint8_t>: zigbee network channel 11-26.
[Host] extPanId - <uint64_t>: extended pan id, mac address will be used if set
to 0.
[Host] panId - <uin16_t>: network pan id.

# zb_nwk_form 20 0 0xabd3
[Host]syncmsg_start
[BLZ]tx enqueue
...
[Host]form network: success.
...
[BLZ]rx frameCtrl=0x00, frmSeq=0x17, rxSeq=1, txSeq=7, frameId=0035
...
[Host]NCP_NETWORK_UP
```

3.2.3 Get NCP information

In Host’s serial terminal, send “zb_info”, then NCP’s mac address, version, network related information will be listed:

```
[2024-01-10 19:51:49.478]: zb_info
[Host]syncmsg_start
[BLZ]tx enqueue
...
[Host]Get network security: success.
[Host]ncp mac: 0xb4e8420004b30000
[Host]ncp version: 1.6.40-1793
[Host]nodeType=0x00, extPanId=0xb4e8420004b30000, panId=0xabd3
[Host]txPower=14, channel=20
[Host]nwkManager=0x0000, nwkUpdateId=0, channelMask=0x00100000
[Host]nwk key: 2e 85 e2 43 ef 14 db f6 4b 90 d3 d0 0f 79 be 42
[Host]nwk frame counter: 143408
[Host]nwk key seq: 0
```

3.2.4 ZCL read attribute

In Host’s serial terminal, send “zb_zcl_read_attr”, the required parameters will be promoted, below is an example to read the “On/Off” attribute from a light device, in which, you can see how to send APS layer messages, as well as how to process the received APS confirm and APS data indication (e.g. ZCL Read Attribute Response in this case):

```
zb_zcl_read_attr
[Host]Wrong number of args
[Host]Usage: zb_zcl_read_attr <msgType> <dstAddr> <srcEp> <dstEp> <manufCode>
<clusterId> <attributeId>
[Host] msgType - <uint8_t> 0: indirect via binding table, 1: unicast, 2:
multicast; 3. broadcast
[Host] dstAddr - <uint16_t> dest device's short addr, valid only if dstAddrMode
is not 0.
[Host] srcEp - <uint8_t> source endpoint.
[Host] dstEp - <uint8_t> dest endpoint.
[Host] manufCode - <uint16_t> manufacture id (neither 0x0000 nor 0xFFFF) used to
read manufacture's private cluster attribute.
[Host] clusterId - <uint16_t> cluster id.
[Host] attributeId - <uint16_t> attribute id.

# zb_zcl_read_attr 1 0x6382 1 1 0x0000 0x0006 0x0000
[Host]syncmsg_start
[BLZ]tx enqueue
...
[BLZ]rx frameCtrl=0x00, frmSeq=0x06, rxSeq=0, txSeq=6, frameId=0051
[Host]APS confirm: status=0x00, dstAddr=0x6382, dstEp=0x01, tag=2
...
[BLZ]rx frameCtrl=0x00, frmSeq=0x17, rxSeq=1, txSeq=7, frameId=0052
[Host]APS data from: srcAddr=0x6382, srcEp=0x01, profileId=0x0104,
clusterId=0x0006, len=8
[Host]APS payload: 18 02 01 00 00 00 10 00
```

3.2.5 ZCL send request

This is a general CLI command to send various ZCL request frames.

```
[HOST]Usage: zb_zcl_command <msgType> <dstAddr> <srcEp> <dstEp> <manufCode>
<clusterId> <zclFrameType> <direction> <cmdId> <payloadLen> <payload>
[HOST] msgType - <uint8_t> 0: indirect via binding table, 1: unicast, 2:
multicast; 3. broadcast
[HOST] dstAddr - <uint16_t> dest device's short addr, valid only if dstAddrMode
is not 0.
[HOST] srcEp - <uint8_t> source endpoint.
[HOST] dstEp - <uint8_t> dest endpoint.
[HOST] manufCode - <uint16_t> manufacture id (neither 0x0000 nor 0xFFFF) used to
read manufacture's private cluster attribute.
[HOST] clusterId - <uint16_t> cluster id.
[HOST] attributeId - <uint16_t> attribute id.
[HOST] zclFrameType - <uint8_t> zcl general frame type (0x00) or cluster
specific frame (0x01)
[HOST] direction - <uint8_t> 0x00: from client to server, 0x01: from server to
client
[HOST] cmdId - <uint8_t>: command id
[HOST] payloadLen - <uint8_t>: command payload length
[HOST] payload - <uint8_t[]>: command payload, in bytes string, required if
payload_len > 0.
```

Examples:

- read manufacture name attribute (0x0004) from basic cluster
  zb_zcl_command 1 0x6676 1 1 0x0000 0x0000 0x00 0x00 0x00 0x02 0400

- send unicast toggle command
  zb_zcl_command 1 0x6676 1 1 0x0000 0x0006 0x01 0x00 0x02 0x00

- send unicast move current-hue to 0
  zb_zcl_command 1 0x6676 1 1 0x0000 0x0300 0x01 0x00 0x00 4 00000000

- send unicast step-hue command (stepMode=Up, StepSize=1, transitionTime=0)
  zb_zcl_command 1 0x6676 1 1 0x0000 0x0300 0x01 0x00 0x02 4 01010000

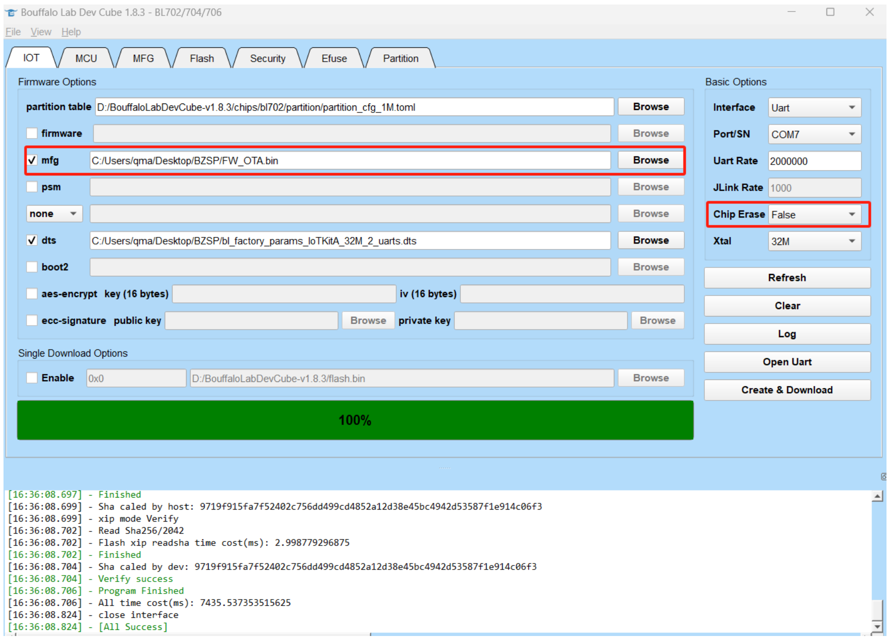
- read the current-hue attribute
  zb_zcl_command 1 0x6676 1 1 0x0000 0x0300 0x00 0x00 0x00 0x02 0000

3.2 Zigbee Device OTA

This section describes how to use the demo HOST and NCP applications to upgrade Zigbee devices. The HOST application serves as OTA Server, while NCP just forwards OTA requests and responses between HOST and Zigbee device (OTA Client).

3.2.1 Upload OTA upgrade image

The demo HOST application saves/reads upgrade image to/from its internal flash (the "mfg" partition). As below, Bouffolab Dev Cube can be used to upload the OTA upgrade image to HOST:



partition table: D:/BouffaloLabDevCube-v1.8.3/chips/bl702/partition/partition_cfg_1M.toml
firmware: [ ]
mfg: [✓] C:/Users/qma/Desktop/BZSP/FW_OTA.bin
psm: [ ]
none
dts: [✓] C:/Users/qma/Desktop/BZSP/bl_factory_params_IoTKitA_32M_2_uarts.dts
boot2: [ ]
aes-encrypt: [ ] key (16 bytes) iv (16 bytes)
ecc-signature: [ ] public key private key
Enable: [ ] 0x0 D:/BouffaloLabDevCube-v1.8.3/flash.bin

Interface: Uart
Port/SN: COM7
Uart Rate: 2000000
JLink Rate: 1000
Chip Erase: False
Xtal: 32M

```
[16:36:08.697] - Finished
[16:36:08.699] - Sha caled by host: 9719f915fa7f52402c756dd499cd4852a12d38e45bc4942d53587f1e914c06f3
[16:36:08.699] - xip mode Verify
[16:36:08.702] - Read Sha256/2042
[16:36:08.702] - Flash xip readsha time cost(ms): 2.998779296875
[16:36:08.702] - Finished
[16:36:08.704] - Sha caled by dev: 9719f915fa7f52402c756dd499cd4852a12d38e45bc4942d53587f1e914c06f3
[16:36:08.704] - Verify success
[16:36:08.706] - Program Finished
[16:36:08.706] - All time cost(ms): 7435.537353515625
[16:36:08.824] - close interface
[16:36:08.824] - [All Success]
```

Two differences, compare to the normal firmware flashing:

- Check the "mfg" check box and browse to select the OTA upgrade file (may save as .bin format first).
- The "Chip Erase" option should be set to false, since we are only writing the "mfg" partition.

For how to generate the OTA upgrade image, it is out of the scope of this document.

3.2.2 Load OTA upgrade image

In HOST, using the "zb_ota_load_image" CLI command to load OTA upgrade image information from flash to RAM, before starting the OTA image transmission.

```
[2024-01-18 09:20:23.553]: zb_ota_load_image
[MTD] >>>>>> Hanlde info Dump >>>>>>
name FW
id 0
offset 0x00093000(602112)
size 0x00066000(408Kbytes)
xip_addr 0x00000000
[MTD] <<<<<< Hanlde info End <<<<<<
Starting OTA. OTA size is 417792
[OTA] activeIndex is 0, use OTA address=00093000
[HOST]OTA image info:
[HOST]Manufacturer ID: 0x130d
[HOST]Image Type ID: 0x0000
[HOST]Version: 0x00000002
```

```
[HOST]Header String: test
[HOST]Size: 344098
```

3.2.3 Image Notify

In HOST, use “zb_ota_image_notify” CLI command to notify client Zigbee device to start the upgrade.

```
zb_ota_image_notify
[HOST]Wrong number of args
[HOST]Usage: zb_ota_image_notify <dstAddr> <srcEp> <dstEp> <queryJitter>
[HOST] dstAddr - <uint16_t> dest short address
[HOST] srcEp - <uint8_t> source endpoint of ota server.
[HOST] dstEp - <uint8_t> ota client's endpoint.
[HOST] queryJitter - <uint8_t> query jitter.

# zb_ota_image_notify 0x90FA 1 1 0
...
[HOST]APS data from: srcAddr=0x90fa, srcEp=0x01, profileId=0x0104,
clusterId=0x0019, len=14
[HOST]APS payload: 01 00 01 01 0d 13 00 00 01 00 00 00 00 00
[HOST]blzProcessZclCommand
[HOST]incoming zcl frame: frameCtrl=0x01, seqNum=0, cmdId=1
[HOST]processQueryNextImageRequest: srcAddr=90fa
[HOST]Send QueryNextImageResponse: found matched image
[HOST]blzSendApsDataRequest: dstAddr=0x90fa, asdulen=16
```

3.2.4 Image Transmission

Normally, in the HOST’s serial terminal, you can see the continuous “ImageBlockRequest” and “ImageBlockReponse” which indicate the image transmissions, and finally “UpgradeEndResponse” indicates the transmission is finished.

```
[HOST]host rx dequeue
[HOST]APS data from: srcAddr=0x90fa, srcEp=0x01, profileId=0x0104,
clusterId=0x0019, len=19
[HOST]APS payload: 01 7a 03 02 0d 13 00 00 02 00 00 00 42 1a 00 00 40 00 00
[HOST]blzProcessZclCommand
[HOST]incoming zcl frame: frameCtrl=0x01, seqNum=122, cmdId=3
[HOST]processImageBlockRequest: srcAddr=90fa
[HOST]block fileOffset=6722
[HOST]syncmsg_start
[HOST]Recv frame: frameId=0038
[HOST]syncmsg_on_received: response received
[HOST]host rx enqueue
[HOST]get nwk payload limit: success.
[HOST]nwk payload limit=88
[HOST]syncmsg_release
[HOST]max free aps payload length: 77
[HOST]block size=63
[HOST]Send ImageBlockResponse: status=0x00, block size=63
[HOST]blzSendApsDataRequest: dstAddr=0x90fa, asdulen=80

…
[HOST]host rx dequeue
[HOST]APS data from: srcAddr=0x90fa, srcEp=0x01, profileId=0x0104,
clusterId=0x0019, len=12
[HOST]APS payload: 01 a4 06 00 0d 13 00 00 02 00 00 00
[HOST]blzProcessZclCommand
[HOST]incoming zcl frame: frameCtrl=0x01, seqNum=164, cmdId=6
```

```
[HOST]processUpgradeEndRequest: srcAddr=90fa, status=0
[HOST]Send UpgradeEndResponse
[HOST]blzSendApsDataRequest: dstAddr=0x90fa, asdulen=19
```

3.3 NCP Backup & Restore

In the HOST application, it demos how to backup NCP and restore the NCP to its previous network. Two CLI commands: “`zb_ncp_backup`” and “`zb_ncp_restore`”.

The required data to backup and restore is listed below:

- NCP’s MAC address.
- Network Extended PanId, PanId, Channel.
- Network key and outgoing nwk frame counter.
- Global trust center link key and outgoing frame counter.
- Device unique trust center link keys.

3.4 NCP Upgrade

This section describes how to upgrade NCP's firmware using the demo HOST application. Different from the normal Zigbee device OTA, NCP needs to enter the firmware downloading mode (via bootstrap pin and reset pin) and then write the new firmware to NCP via UART.

NCP upgrade flow:

1) HOST controls NCP to enter downloading mode by pulling up the bootstrap pin and keep it, then pull down & up the reset pin.
2) HOST stops the BLZ communication.
3) HOST handshakes with NCP's Bootrom and downloads the "eflash_loader" program to NCP and run it.
4) HOST communicates with "eflash_loader", request "eflash_loader" to erase necessary flash sectors and write new firmware's data to NCP's flash.
5) Verify the new firmware by checking the image hash.
6) HOST restarts the BLZ communication.
7) Reboot NCP.

Notes:

1. the upgrade image for NCP is also saved in the HOST's internal flash, refer to 3.2.1 for more details.
2. to generate NCP's upgrade image, the FW_OTA.bin or the FW_OTA.bin.hash file SHALL be used and the compressed version SHOULD NOT be used.
3. For host application, please use the provided partition table file (partition_cfg_1M_host.toml) which has larger "MFG" partition.

In the HOST application, the "zb_ncp_ota" CLI command demos the NCP upgrade progress.

4 Steps to create NCP OTA image:

1. build ncp firmware

```
$ ./genncp
```

2. generate the FW_OTA.bin.hash which includes boot header and image hash.

```
$ bflb-iot-tool --chipname=BL702 --firmware=./build_ncp/bl702_demo_ncp_host.bin --build --ota ./ncp_ota/
```

3. generate NCP ota image which include OTA header used by Zigbee.

```
$ ./ota-image-builder --bin-file ncp_ota/FW_OTA.bin.hash --ota-file NCP_OTA_IMAGE_v2.bin --manuf-id 0x130d --image-type 0x0000 --image-version 0x00000002 --min-hw-ver 0x0000 --max-hw-ver 0xfff0 --header-string test
```

4. upload above ota image to HOST device.