

HT32F67741 BLE Evaluation Guide

Revision History

Version	Date	Descriptions	Owner
0.1	2022-10-14	Initial version	Johnny_Wei

Contents

1	Introduction.....	6
2	HT32F67741 Environment setup.....	6
2.1	Download CM0 library	6
2.2	Download BLE library.....	6
2.3	Compiler code with BLE library.....	6
2.4	Download code	7
3	BLE API.....	7
3.1	BLE Callback function	8
3.2	Register BLE event callback.....	8
3.3	configure BLE interface.....	8
3.4	initial BLE	9
3.5	BLE TX/RX process	9
3.6	Read BLE status.....	9
3.7	Disconnect BLE.....	9
3.8	Modify BLE connection interval	10
3.9	Get BLE connection interval.....	10
3.10	Set BLE device name	10
3.11	Get BLE device name	11
3.12	Set BLE device address	11
3.13	Get BLE device address.....	11
3.14	BLE Advertising enable/disable.....	11
3.15	Get BLE Advertising status is enable or not	12
3.16	Set BLE Advertising interval	12
3.17	Get BLE Advertising interval	12
3.18	Set BLE Advertising data.....	12
3.19	Get BLE Advertising data	13
3.20	Set BLE scan data	13
3.21	Get BLE scan data	13
3.22	Set BLE tx power.....	13
3.23	Get BLE tx power	14
3.24	Set BLE Crystal offset	14
3.25	Get BLE peer device address	14
3.26	Set BLE feature.....	14
3.27	Get BLE data code version	15
3.28	Set BLE power saving mode.....	15
3.29	wakeup BLE	16
3.30	Set BLE baudrate	16

3.31	Get BLE baudrate	16
3.32	Get BLE max baudrate.....	17
3.33	BLE reset	17
3.34	Set BLE WhiteList.....	17
3.35	Get BLE WhiteList	17
3.36	Set BLE Characteristic value	18
3.37	Get value from UUID	18
4	BLE callback example	18

List of figures

Figure 1	6
----------------	---

1 Introduction

This document give a brief description for the following topic.

How to use BLE library on evaluation board

2 HT32F67741 Environment setup

2.1 Download CM0 library

Current M0 version use **HT32_STD_5xxxx_FWLib_V1.1.1_5938**.

How to use M0 latest version description will be update next time.

2.2 Download BLE library

BLE library **HT32_BLE_Library_vXXX** folder structure as following

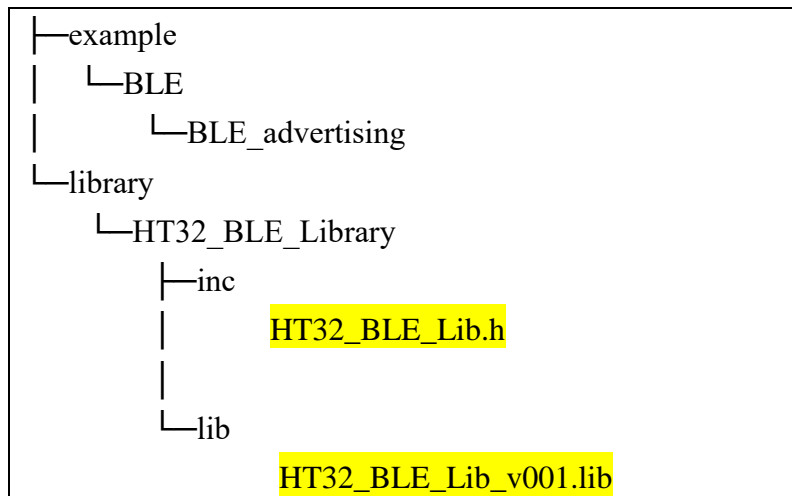


Figure 1

2.3 Compiler code with BLE library

You need copy BLE files to CM0 library before compiler

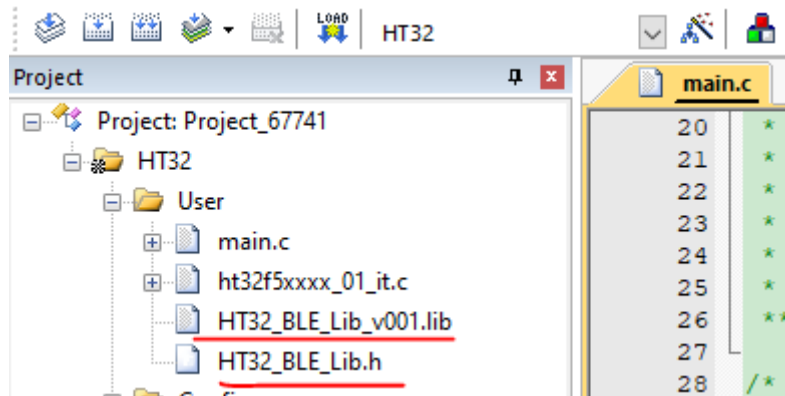
1. copy BLE **example code** to the following path

.\HT32_STD_5xxxx_FWLib_Vx.x.x_xxxx**example**\

2. Copy **HT32_BLE_Library** to the following path

.\ HT32_STD_5xxxx_FWLib_Vx.x.x_xxxx **library**\

3. Add **HT32_BLE_Lib.h** and **HT32_BLE_Lib_v001.lib** to project



Finally, the folder will like the following

.\HT32_STD_5xxxx_FWLib_Vx.x.x_xxxx\example\BLE\ BLE_advertising
.\ HT32_STD_5xxxx_FWLib_Vx.x.x_xxxx \library\HT32_BLE_Library

4. Compiler code

Open MDK_ARMv5\ Project_67741.uvprojx and compiler code

Only support Keil.

2.4 Download code

Download to Evaluation board.

3 BLE API

BLE API use BLE_**Set**xxx and BLE_**Get**xx to modify BLE parameter and get BLE event or parameter.

User can get BLE event from BLE callback function. Different BLE event type may have different data structure. BLE event type and data structure can find in **HT32_BLE_lib.h**. Please refer to **BLE callback example**.

User should do next operation until receive BLE success event.

3.1 BLE Callback function

```
/******  
* Callback function prototype to get BLE event/status  
* @param type          - BLE event type  
* @param evt_status    - BLE event status  
* @param buffer        - pointer to data to be read  
* @param buffer_Len    - length of data  
* @note  
*      each event type may have different data structure  
*****/  
typedef void (*FUNPTR_EvtCallback)(u16 type, en_BLE_St evt_status, u8 *buffer, u8  
buffer_Len);
```

3.2 Register BLE event callback

```
/******  
* @brief  register BLE event callback to receive event/status form BLE. Must be setup by the  
application  
* @param  funptr:   BLE callback function pointer  
* @retval None  
*****/  
void BLE_registerEvtCallback(FUNPTR_EvtCallback funptr);
```

3.3 configure BLE interface

```
/******  
* @brief  configure BLE interface  
* @param  buadrate:  BLE baudrate speed  
* @retval None  
*****/  
void BLE_InterfaceConfigure(u32 buadrate);
```


3.4 initial BLE

```
/******  
 * @brief  initial BLE  
 * @retval BLE state  
*****/  
en_BLE_St BLE_Init(void);
```

3.5 BLE TX/RX process

```
/******  
 * @brief  BLE TX/RX process routine.  
 * @retval None  
 *  
*****/  
void BLE_Routine(void);
```

3.6 Read BLE status

```
/******  
 * @brief  read BLE current status  
 * @retval BLE state  
*****/  
en_BLE_St BLE_ReadStatus(void);
```

3.7 Disconnect BLE

```
/******  
 * @brief  Disconnect BLE current link  
 * @retval BLE state  
*****/  
en_BLE_St BLE_Disconnect(void);
```

3.8 Modify BLE connection interval

```
/******  
 * @brief  modify BLE connection interval  
 * @param  min:  Minimum connection interval(unit 1.25ms)  
 * @param  max:  Maximum connection interval(unit 1.25ms)  
 * @retval BLE state  
 * @note valid range=7.5ms~4s(0x0006~0x0C80)  
*****/  
en_BLE_St BLE_SetConnectInterval(u16 min,u16 max);
```

3.9 Get BLE connection interval

```
/******  
 * @brief  get BLE connection interval  
 * @retval BLE state  
 * @note  
*****/  
en_BLE_St BLE_GetConnectInterval(void);
```

3.10 Set BLE device name

```
/******  
 * @brief  set BLE device name  
 * @param  name    - device name buffer  
 * @param  length  - device name length  
 * @retval BLE state  
 * @note length must less than or equal to 31 byte  
*****/  
en_BLE_St BLE_SetDeviceName(u8 *name, u8 length);
```

3.11 Get BLE device name

```
/******  
 * @brief  get BLE device name  
 * @retval BLE state  
 * @note  
*****/  
en_BLE_St BLE_GetDeviceName(void);
```

3.12 Set BLE device address

```
/******  
 * @brief  set BLE device address  
 * @param  addr  - device address buffer(6 byte)  
 * @param  type  - BLE Address Type, default use 0x0 (Static address)  
 * @retval BLE state  
 * @note BLE Address Type only support Static address  
*****/  
en_BLE_St BLE_SetDeviceAddress(u8 *addr,u8 type);
```

3.13 Get BLE device address

```
/******  
 * @brief  get BLE device address  
 * @retval BLE state  
 * @note  
*****/  
en_BLE_St BLE_GetDeviceAddress(void);
```

3.14 BLE Advertising enable/disable

```
/******  
 * @brief  BLE Advertising enable/disable  
 * @retval BLE state  
 * @note  
*****/  
en_BLE_St BLE_SetAdvertising(ControlStatus ctrl);
```

3.15 Get BLE Advertising status is enable or not

```
/******  
 * @brief  get BLE Advertising status is enable or not  
 * @retval BLE state  
 * @note  
*****/  
en_BLE_St BLE_GetAdvertisingStatus(void);
```

3.16 Set BLE Advertising interval

```
/******  
 * @brief  set BLE Advertising interval  
 * @param  min   - Minimum Advertising interval (unit=0.625ms)  
 * @param  max   - Maximum Advertising interval (unit=0.625ms)  
 * @retval BLE state  
 * @note valid range=20ms~10.24s(0x0020~0x4000)  
*****/  
en_BLE_St BLE_SetAdvertisingInterval(u16 min,u16 max);
```

3.17 Get BLE Advertising interval

```
/******  
 * @brief  get BLE Advertising interval  
 * @retval BLE state  
 * @note  
*****/  
en_BLE_St BLE_GetAdvertisingInterval(void);
```

3.18 Set BLE Advertising data

```
/******  
 * @brief  set BLE Advertising data  
 * @param  advdata - Advertising data  
 * @param  length  - Advertising data length  
 * @retval BLE state  
 * @note length must less than or equal to 31 byte  
*****/  
en_BLE_St BLE_SetAdvertisingData(u8 *advdata,u8 length);
```

3.19 Get BLE Advertising data

```
/******  
 * @brief  get BLE Advertising data  
 * @retval BLE state  
 * @note  
 *****/  
en_BLE_St BLE_GetAdvertisingData(void);
```

3.20 Set BLE scan data

```
/******  
 * @brief  set BLE scan data  
 * @param  scandata - BLE scan data  
 * @param  length    - BLE scan data length  
 * @retval BLE state  
 * @note length must less than or equal to 31 byte  
 *****/  
en_BLE_St BLE_SetScanResponseData(u8 *scandata,u8 length);
```

3.21 Get BLE scan data

```
/******  
 * @brief  get BLE scan data  
 * @retval BLE state  
 *****/  
en_BLE_St BLE_GetScanResponseData(void);
```

3.22 Set BLE tx power

```
/******  
 * @brief  set BLE tx power  
 * @param  pwr - BLE tx power  
 * @retval BLE state  
 * @note valid range from 0x00~0x0F(max), default is 0xC  
 *****/  
en_BLE_St BLE_SetTxPower(u8 pwr);
```

3.23 Get BLE tx power

```
/******  
 * @brief  get BLE tx power  
 * @retval BLE state  
 *****/  
en_BLE_St BLE_GetTxPower(void);
```

3.24 Set BLE Crystal offset

```
/******  
 * @brief  set BLE Crystal offset  
 * @param  cl - BLE Crystal offset  
 * @note valid range from 0x00~0x0F(max), default is 0x4  
 *****/  
en_BLE_St BLE_SetCrystalCload(u8 cl);
```

3.25 Get BLE peer device address

```
/******  
 * @brief  get BLE peer device address  
 * @retval BLE state  
 *****/  
en_BLE_St BLE_GetPeerBTAddr(void);
```

3.26 Set BLE feature

```
/******  
 * @brief  set BLE feature  
 * @param  flag : BLE feature flag  
           b[2]  - advDataNotAutoAppendBTName(Not auto append BTName in  
advertising data)  
           b[4]  - swParaUpdate(Save swPara in Flash (auto setup swParaErase=1)  
           b[5]  - swParaErase(Erase swPara in Flash)  
           b[7]  - autoSendSatus(send BLE status when change)  
           b[9]  - rfCalibDone(send status after calibration finish)
```

b[12] - external32k(1=use external 32k, 0=use internal 32k)
b[25] - rfCalibForce(Force calibration at next RF turned off->on)
others reserved.range from b[0]~b[31]

* @retval BLE state

*****/

en_BLE_St BLE_SetFeature(FeatureFlag flag);

3.27 Get BLE data code version

* @brief get BLE data code version

* @retval BLE state

*****/

en_BLE_St BLE_GetVersion(void);

3.28 Set BLE power saving mode

* @brief set BLE power saving mode

* @param mode : BLE power saving mode

0 : OP_NORMAL , Running between Sleep and wake-up

1 : OP_DEEPSLEEP ,Running between Sleep, Deep Sleep and wake-up, must
send dummy bytes to wake up BLE

2 : OP_POWERDOWN , Keep in Deep Sleep, wake-up will force to reset, all
setting in RAM reset to default value

* @retval BLE state

*****/

en_BLE_St BLE_SetPowerSaving(u8 mode);

* @brief set BLE to power saving mode. BLE can also send dummy bytes to wake up Host

* @param mode : BLE power saving mode

0 - OP_NORMAL , Running between Sleep and wake-up

1 - OP_DEEPSLEEP ,Running between Sleep, Deep Sleep and wake-up, must
send dummy bytes to wake up BLE

2 - OP_POWERDOWN , Keep in Deep Sleep, wake-up will force to reset, all
setting in RAM reset to default value

```

* @param  enableWakeup      : enable BLE wakeup host
* @param  wakeupEvtDelayTime : Delay between dummy wake-up and event(unit=ms)
* @param  wakeupByteCnt     : Dummy wake-up byte number
* @retval BLE state
* @note :Due to BLE scheduling, may be delay more than setting

```

```

/*****

```

```

en_BLE_St BLE_SetPowerSavingSlaveWakeup(u8 mode,ControlStatus enableWakeup,u8
wakeupEvtDelayTime,u8 wakeupByteCnt);

```

3.29 wakeup BLE

```

/*****

```

```

* @brief  wakeup BLE by dummy wakeup
* @param wakeupByteCnt - byte count of dummy wakeup
* @retval BLE state
* @note max wakeupByteCnt is 8

```

```

/*****

```

```

en_BLE_St BLE_DummyWakeup(u8 wakeupByteCnt)

```

3.30 Set BLE baudrate

```

/*****

```

```

* @brief  set BLE baudrate
* @param baudrate
* @retval BLE state

```

```

/*****

```

```

en_BLE_St BLE_SetBaudRate(u32 baudrate);

```

3.31 Get BLE baudrate

```

/*****

```

```

* @brief get BLE baudrate
* @retval BLE state

```

```

/*****

```

```

en_BLE_St BLE_GetBaudRate(void);

```


3.32 Get BLE max baudrate

```
/******  
 * @brief get BLE max baudrate  
 * @retval BLE state  
 *****/  
en_BLE_St BLE_GetMaxBaudRate(void);
```

3.33 BLE reset

```
/******  
 * @brief BLE reset  
 * @retval BLE state  
 *****/  
en_BLE_St BLE_SoftwareReset(void);
```

3.34 Set BLE WhiteList

```
/******  
 * @brief set BLE WhiteList  
 * @param eraseBeforeSet : Remove all whitelists before setup  
 * @param addr - BLE address(LSB -> MSB)  
 * @param mask - BLE address mask(mask will do "AND action" with BLE address)  
 * @retval BLE state  
 * @note  
     If the peer device uses random address, the function may be useless  
     Max storage 4 pair whitelist address+addressMask. The 5th pair will be wraparound to  
the 1st pair  
     Ex: BtAddr=0x112233445566, BtAddrMask=0x00FFFFFFFF, then the device with  
address between 0x002233445566 to 0xFF2233445566 can be connected to the BLE.  
 *****/  
en_BLE_St BLE_SetWhiteList(ControlStatus eraseBeforeSet, u8 *addr, u8 *mask);
```

3.35 Get BLE WhiteList

```

/*****
 * @brief get BLE WhiteList
 * @retval BLE state
 *****/
en_BLE_St BLE_GetWhiteList(void);

```

3.36 Set BLE Characteristic value

```

/*****
 * @brief set BLE Characteristic value and Characteristic notificaion
 * @param Characteristic : Characteristic UUID
 * @param serviceFlag : service operation
 *                        0x00 - BLE_ServiceMessage : set Characteristic value
 *                        0x20 - BLE_ServiceProperties : Characteristic notify
 * @param buffer : Characteristic value
 * @param length : Characteristic value length
 * @retval BLE state
 * @note
 *****/
en_BLE_St BLE_SetCharacteristicValue(u16 Characteristic,u8 serviceFlag,u8 *buffer,u8 length);

```

3.37 Get value from UUID

```

/*****
 * @brief get value of UUID
 * @param UUID : UUID can be service UUID or Characteristic UUID
 * @retval BLE state
 * @note
 *****/
en_BLE_St BLE_GetUUIDValue(u16 UUID);

```

4 BLE callback example

The following shows how to parse buffer from BLE callback with different BLE type.

```
typedef void (*FUNPTR_EvtCallback)(u16 type, en_BLE_St evt_status, u8 *buffer, u8
buffer_Len);
```

API name	type	Buffer(hex)	buffer_Len	
BLE_ReadStatus	0x0000	10010000	0x04	
<p>In this example the buffer content is 0x00000110.</p> <p>((tSTATUS *)(buffer))->b. CalibDone = 1</p> <p>((tSTATUS *)(buffer))->b. Ext32k = 1</p> <p>The status is calibration done and External 32.768k existed</p>				
BLE_GetConnectInterval	0x0004	2400	0x02	
<p>buffer is 0x0024 = 36</p> <p>The interval is 36 * 1.25 = 45ms</p>				
BLE_GetDeviceName	0x0005	486F6C74656B	0x06	
<p>buffer is 0x486F6C74656B</p> <p>Device name is Holtek</p>				
BLE_GetDeviceAddress	0x0006	6984C044AC0400	0x07	
<p>tDEV_ADDR devAddr;</p> <p>The address get by</p> <p>memcpy(&(devAddr.addr),((tDEV_ADDR *)buffer)->addr,6);</p> <p>The address type</p> <p>devAddr_type=((tDEV_ADDR *)buffer)->type;</p> <p>address type 0x00 is Static address</p>				
BLE_GetAdvertisingStatus	0x0007	01	0x01	
<p>Buffer is 0x01</p> <p>Advertising is enable</p>				
BLE_GetAdvertisingInterval	0x0008	2000200007	0x05	
<p>tADV_INTV devAdvIntv;</p> <p>memcpy(&(devAdvIntv),buffer,buffer_Len);</p> <p>adv min : 0x0020</p> <p>adv max : 0x0020</p> <p>adv channel : 0x07 means advertising in ch37, ch38, ch39</p>				
BLE_GetAdvertisingData	0x0009	020106 0BFF FFFF486F6C74656B3031	0x17	

		0709486F6C74656B		
Buffer Format is length + type + value Example: 02:length=0x02 01: type=flag 06: value				
BLE_GetScanResponseData	0x000A	0B03001801180A180F18F0FF	0x0C	
Refer to BLE_GetAdvertisingData				
BLE_GetTxPower	0x000B	0C	0x01	
The power is 0x0C				
BLE_GetPeerBTAddr	0X000F	FFD3E1B74A6801	0x07	
tDEV_ADDR peerDevAddr The address get by <pre>memcpy(&(peerDevAddr.addr),((tDEV_ADDR *)buffer)->addr,6);</pre> The address type <pre>devAddr_type=((tDEV_ADDR *)buffer)->type;</pre> address type 0x01 is Random address				
BLE_GetVersion	0x0020	202204181607	0x06	
Date code Version is 202204181607				
BLE_GetBaudRate	0x0026	00C20100	0x04	
Baudrate is 0x0001C200(115200) bps				
BLE_GetMaxBaudRate	0x0027	00D43000	0x04	
Baudrate is 0x0030D400(3200000) bps				
BLE_GetWhiteList	0X002A	112233445566FFFFFFFFFFFF	0x0C	
tWHITE_LIST devWhitelist; <pre>memcpy(&(devWhitelist.addr),((tWHITE_LIST *)buffer)->addr,6);</pre> <pre>memcpy(&(devWhitelist.mask),((tWHITE_LIST *)buffer)->mask,6);</pre> Whitelist addr:0x665544332211 Whitelist mask:0xFFFFFFFFFFFF				
BLE_GetUUIDValue	0X2A29	484F4C54454B31322020202020202020	0x10	
ManufacturerNameString UUID is 0x2A29 Buffer is 0x484F4C54454B31322020202020202020("HOLTEK12") 0x20 is space				

--