BM7701-00-1 Application Note

D/N: AN0614EN

Introduction

The BM7701-00-1 is a Bluetooth Low Energy transceiver module based on Holtek's BC7701 device and BLE transparent transmission. The module can be used to wirelessly control external devices. It supports bidirectional data transmission and is suitable for lighting products, healthcare products and home appliances as well as many others. This application note will introduce how to use a Holtek 32-bit MCU, the HT32F52352, and an 8-bit MCU, the HT66F2370, to control the BM7701-00-1 module to modify Bluetooth parameters and transmit data using a smartphone App. This methodology can implement functions to send out data using keys and to display the received status using LEDs. By using an example for data transmission between the Bluetooth module and a smartphone App, it can demonstrate the Bluetooth module's ability for data transmission and reception, thus assisting users in its usage with a range of electronic products.

Functional Description

Communication Interface

The BM7701-00-1 module is controlled using a UART interface, which is a universal serial data bus for asynchronous communication. This can implement full-duplex transmission and reception. The BM7701-00-1 provides two UART communication interfaces, UART1 and UART2. The UART1 interface will be used throughout this application note.

Pin No.	Pin Name	Functional Description
1	UART1_TX	BLE UART1_TX serial data output
2	UART1_RX	BLE UART1_RX serial data input
3	UART2_TX	BLE UART2_TX serial data output
4	UART2_RX	BLE UART2_RX serial data input

Table 1. BM7701-00-1 UART Interface

When the RST_N pin is pulled from low to high or a power-on reset occurs, the TX/RX pin will remain in the I/O mode for an initial 60ms, here the BM7701-00-1 can select a UART Baud Rate according to the I/O condition. After this, the TX/RX pin state will change back from the I/O mode to the UART mode. This is shown in Table 2.

TX/RX pin will remain in an I/O mode after a power-on reset for 60ms	BM7701-00-1 TX=1 (UART1/2_TX)	BM7701-00-1 TX=0 (UART1/2_TX)
BM7701-00-1 RX=1 (UART1/2_RX)	Baud Rate=115200 - default	Baud Rate=9600

Table 2. UART Baud Rate Selection

UART Format

The UART data format should meet the following requirements:

1. Baud rate: 1953~115200bps - BM7701-00-1
2. Data bit: 8 bits
3. Parity check: No
4. Stop bit: 1 bit

Figure 1. UART Sequence Diagram

Command/Event Format

The BM7701-00-1 supports three types of commands, including CmdType1, CmdType2 and CmdType3. CmdType1 are HCI defined in the Bluetooth specification. CmdType2 support access to the BLE Services. CmdType3 support setting up the BM7701-00-1 internal parameters, including Tx Power, crystal load, advertising enable/disable, advertising interval, connection interval and so on. When the MCU sends a command to the BM7701-00-1 using the UART interface, the BM7701-00-1 will respond with an Event when the BM7701-00-1 successfully identifies the command. Refer to the "BLE_API"document for details. This application only uses CmdType2 and CmdType3 whose formats are shown as follows.

Description	Header (1-byte)	Length (1-byte)	CmdFlag/EvtStatus (1-byte)	Type (2-byte)	Value (n-byte)
Initiator → Responder	0x77	xx	CmdFlag: b[0]: 1: Force Evt to read action format b[3:1]: N/A b[7:4]: 1: CmdFlag_supported 2: CmdFlag_notifyIndicate	UUID (Universal Unique Identifier) Profile: 0x18xx Service: 0x18xx Characteristics: 0x2Axx Descriptor: 0x29xx	Depends on Type
Responder → Initiator	0x78		EvtStatus: [3:0]: ErrorCode [7:4]: 1: CmdFlag_supported 2: CmdFlag_notifyIndicate		

Table 3. CmdType2 Format

Description	Header (1-byte)	Length (1-byte)	CmdFlag/EvtStatus (1-byte, "b" denoted as bit)	Type (2-byte)	Value (n-byte)
Initiator → Responder	0x77	xx	CmdFlag: b[0:1]: BLE responses read Evt b[7:1]: Force Evt to read action format	As follows	Depends on Type
Responder → Initiator	0x78		EvtStatus: b[3:0]: ErrorCode b[7:4]: N/A		

Table 4. CmdType3 Format

Example	
MCU→77 04 00 0B00 08→BM7701-00-1 MCU←78 03 00 0B00←BM7701-00-1	Set RF Tx power =0x08 (CmdType3)
MCU→77 04 00 0700 01→BM7701-00-1 MCU←78 03 00 0700←BM7701-00-1	Start advertising (CmdType3)
MCU→77 08 20 F2FF 01 02 03 04 05→BM7701-00-1 MCU←78 03 20 F2FF←BM7701-00-1	Send 0x0102030405 to phone (CmdType2)
MCU←78 08 00 F2FF 01 02 03 04 05←BM7701-00-1	Receive 0x0102030405 from phone (CmdType2)

Table 5. Examples

User Guide

This chapter will introduce the environment preparation, operating instructions, hardware description, software description, module API function introduction, Bluetooth parameter modification and other information. This will help users can quickly grasp the BM7701-00-1 module basic operation and system architecture using the application examples.

Environment Preparation

Hardware

- HT32: Bluetooth Module - BM7701-00-1, Patch Board - BCT-7701-001, Development Board - BCE-GenTrx32-002 which is used with the HT32F52352 MCU
- HT8: Bluetooth Module - BM7701-00-1, Patch Board - BCT-7701-001, Development Board - BCE-GenTrx8-001 which is used with the HT66F2370 MCU

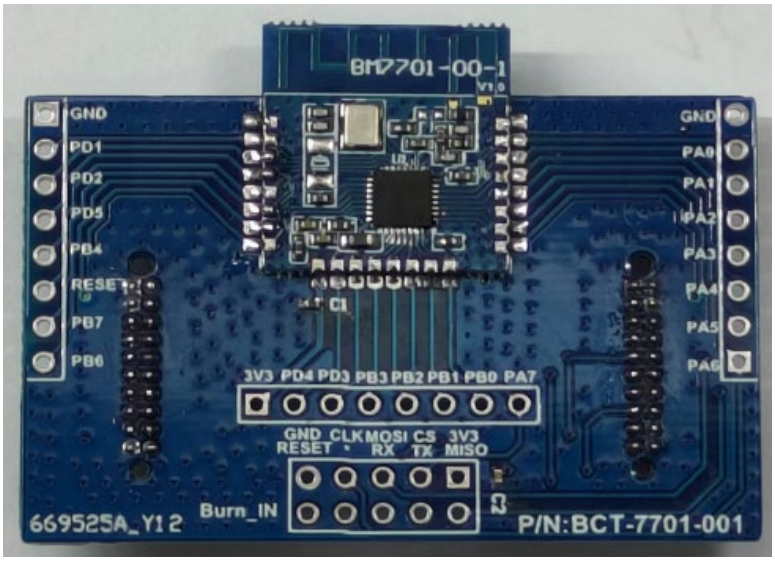

Figure 2. Module + Patch Board

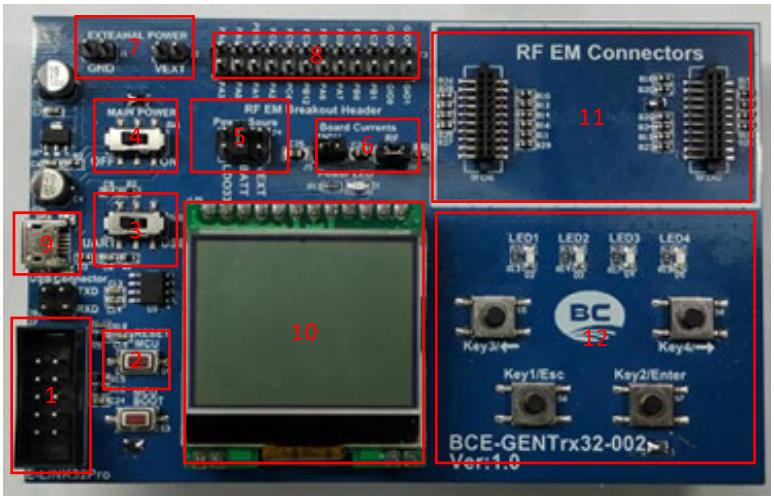

Figure 3. Development Board - BCE-GENTrx32-002

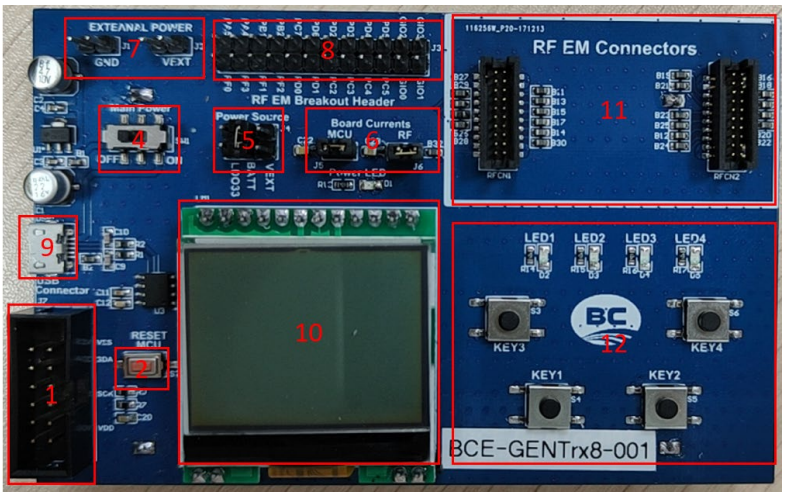

Figure 4. Development Board - BCE-GENTrx8-001

1. JTAG interface, which can be used with IDE interface to emulate and download programs
2. System reset key
3. UART/USB interface selection - only the BCE-GENTrx32-002 development board has this option
4. Main power switch, when in the left position this means OFF and ON when in the right position.
5. System power selection. If the jumper is on the left side LDO33 position, this means that power is sourced from the USB interface. If the jumper is in the middle BATT position, this means that the power supply is sourced from the battery holder, which is on the back of the board and which uses two 1.5V AA batteries. If the jumper is located on the right side VEXT position, this means that power is sourced from the external power contacts. Note that if external power is used, the voltage must not exceed 3.6V.
6. MCU and BM7701-00-1 module board current detection point.
7. External power supply contacts
8. MCU I/O ports
9. Micro USB interface. This can be as a system power input. The power supply jumper should be placed on the LDO33 position.
10. The liquid crystal display supports 128×64 pixels. Refer to the example program in this application note for usage information and also the display control library.
11. The RF module interface is the connection point for the RF transmitter/receiver. In this example, the BCT-7701-001+BM7701-00-1 is used.
12. 4 LEDs and 4 keys. This example is used to display data receiving status and transmitted data.

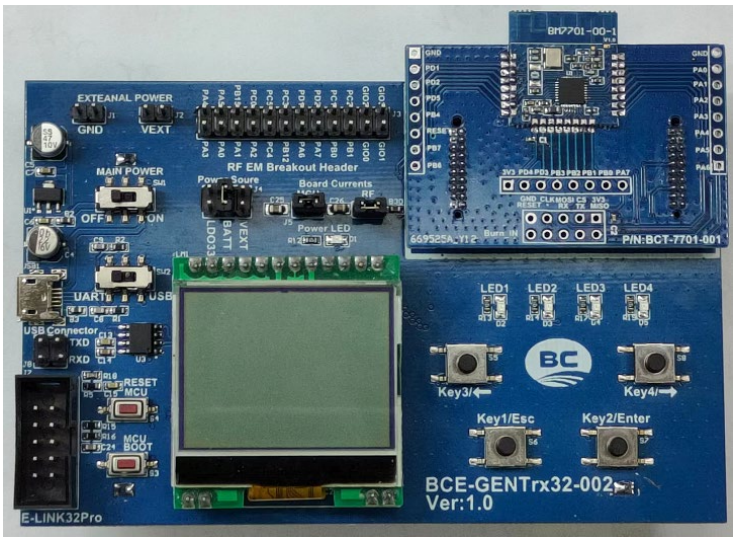

Figure 5. Complete Assembly - HT32

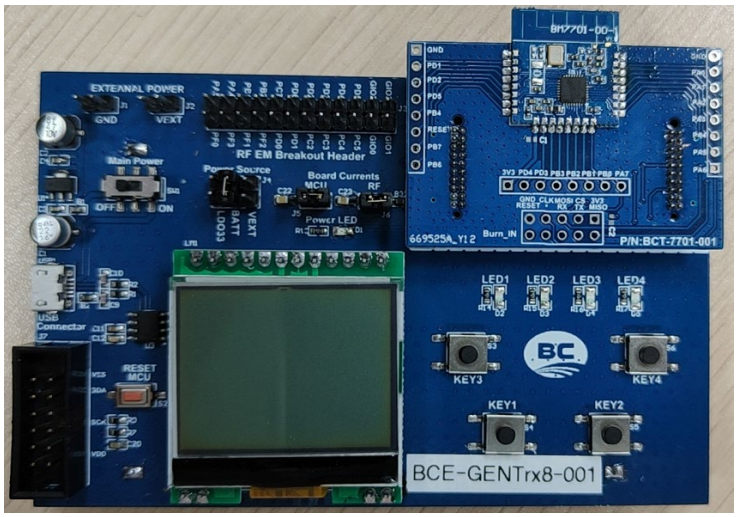

Figure 6. Complete Assembly - HT8

Software: Users need to download the BLEDemo App on their smartphone, the links for this are shown in the following figure.

Figure 7. Android link – Left and iOS link - Right

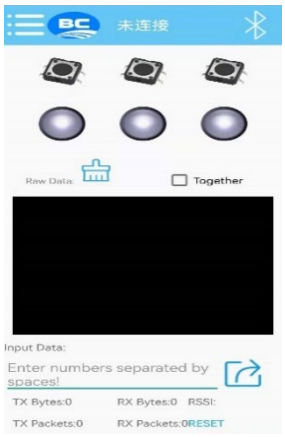

Figure 8. UI after Installation

Operating Instructions

1. Use the elink-32 Pro to program the application example to the HT32F52352 (HT32) on the development board BCE-GENTrx32-002.

   Use the elink to program the application example to the HT66F2370 (HT8) on the development board BCE-GENTrx8-001.

2. Turn on the power switch of the development board. After it is powered-on, the power LED will be on and the LCD screen will display.
3. After the boot screen has displayed for about one second, the LCD will display the status interface until the “PWR ON OK” message is displayed. This means that the Bluetooth has been configured successfully and the advertising has been enabled.
4. Click on the top right of smartphone App to scan for Bluetooth devices and select the name "BM7701" to connect to. When the Bluetooth device has connected successfully, the connection icon in the upper right of the App will turn red.
5. Development board key operation
   - When KEY1 is pressed: The LED1 icon on the App is on
   - When KEY2 is pressed: The LED2 icon on the App is on
   - When KEY3 is pressed: The LED3 icon on the App is on
6. App operation
   - When clicking the “KEY1” icon: LED1 is on
   - When clicking the “KEY2” icon: LED2 is on
   - When clicking the “KEY3” icon: LED3 is on

Hardware Description

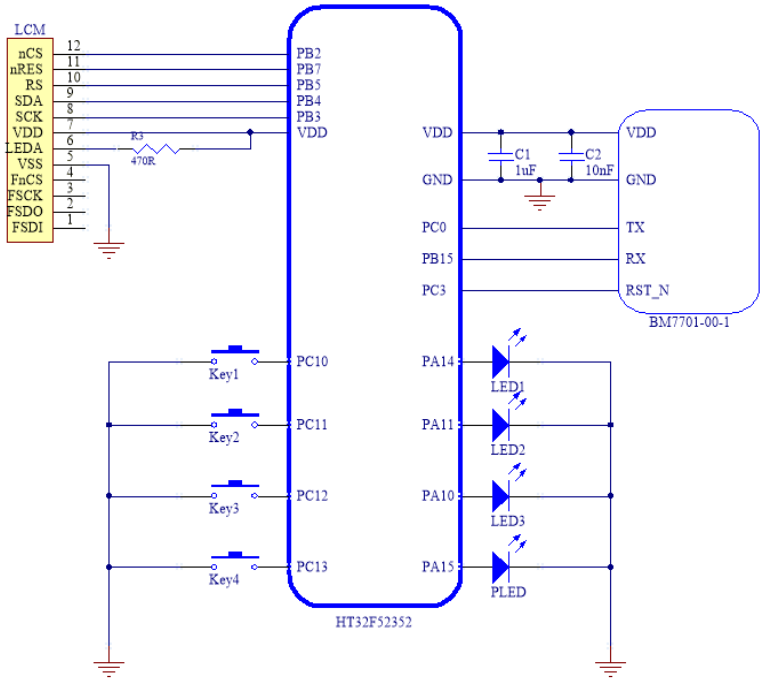

Figure 9. Application Circuit - HT32

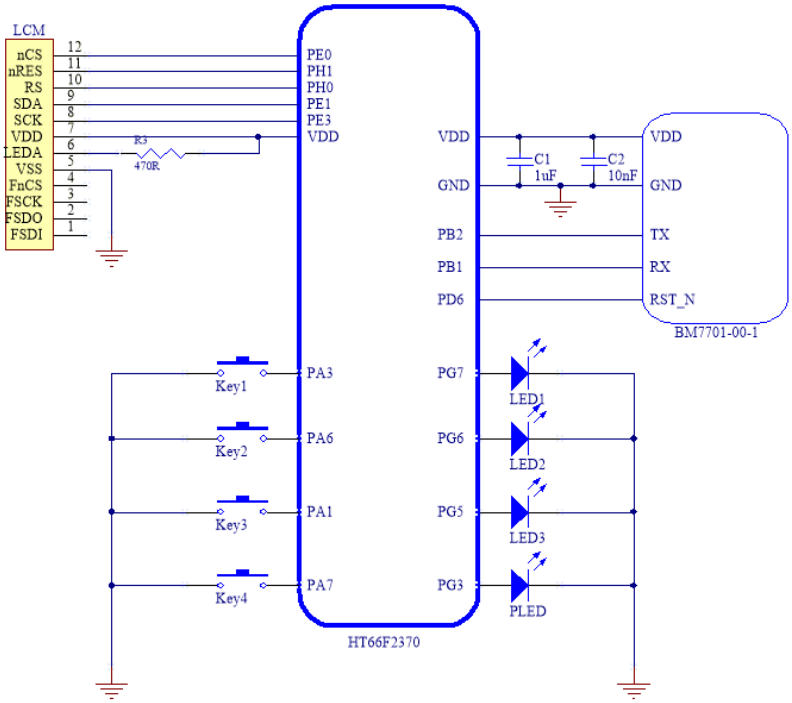

Figure 10. Application Circuit - HT8

Software Description

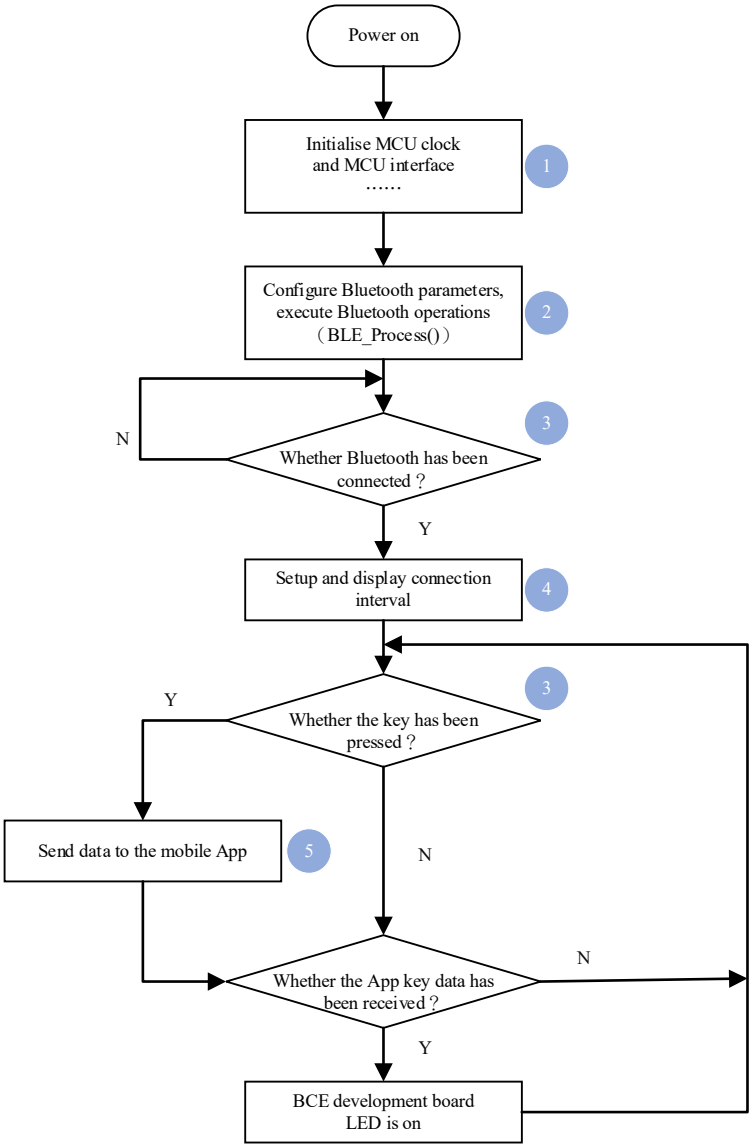

Figure 11. Software Flowchart

- First initialise the MCU clock and then initialise the peripheral devices such as the LEDs, the LCDs and the keys.
- The MCU sends a command to the BM7701-00-1 to configure the Bluetooth parameters and enable advertising.
- The smartphone App is successfully connected with BM7701-00-1 Bluetooth.
- When the MCU detects that the key on the development board has been pressed, the MCU will send a command to the BM7701-00-1 and the BM7701-00-1 will then send data to the smartphone App.
- The smartphone App will send data to the BM7701-00-1 when it has received the command. The MCU will parse the data Event received by the BM7701-00-1 and illuminate the corresponding LED on the development board.

1. Initialise Update Functions
   - BC7701_InterfaceConfigure(_UART_BAUDRATE_): Configure the port states of the MCU I/O which communicate with the BM7701-00-1
   - BC7701_RESET_CLR(): The BM7701-00-1 RST_N pin is pulled low
   - BC7701_RESET_SET(): The BM7701-00-1 RST_N pin is pulled high
   - BC7701_HardwareBaudRateDefault(_UART_BAUDRATE_): Initialise the BM7701-00-1 baud rate using the communication pin state
   - BC7701_HardwareBaudRateRelease(): Complete the BM7701-00-1 baud rate initialisation
2. Setup Bluetooth Parameters
   - BC7701_SendBCIReadPackage(BCI_VERSION, 0x80): Check version (HT32)

     BC7701_TransmitCmdPackage(BCI_VERSION, 0x80): Check version (HT8)
   - BC7701_SendBCIReadPackage(BCI_DEV_ADDRESS, 0x00): Obtain address (HT32)

     BC7701_ TransmitCmdPackage(BCI_DEV_ADDRESS, 0x00): Obtain address (HT8)
   - BC7701_SetBaudRate(), BC7701_SetTxPower()…: Setup the BM7701-00-1 baud rate and Tx power etc. It is required to enable the corresponding options (HT32).

     BC7701_SetBaudRate(),BC7701_TransmitPackageConst((tBCI_PACKAGE *)&BLE_SetTxPower)…: Setup the BM7701-00-1 baud rate and Tx power etc. It is required to enable the corresponding options (HT8).
   - BC7701_SetupFeatureFlag(FEATURE_SET, FEATURE_STATUS_EVENT): Setup the BM7701-00-1 to send data when the Bluetooth communication state has changed - HT32

     BC7701_TransmitPackageConst((tBCI_PACKAGE *)&BLE_SetFeatureState): Setup the BM7701-00-1 to send data when the Bluetooth communication state has changed - HT8
   - BC7701_AdvertisingControl(ENABLE): Setup the BM7701-00-1 to enable advertising - HT32

     BC7701_TransmitPackageConst((tBCI_PACKAGE *)&BC7701_AdvertiseEnable): Setup the BM7701-00-1 to enable advertising - HT8
3. Enter the Deep Sleep Mode
   - BC7701_SetOperateMode(OP_DEEPSLEEP,ENABLE,_MASTER_WUW_VALUE_,_MASTER_WUT_VALUE_): Setup the BM7701-00-1 to enter the Deep Sleep mode and MCU external wake-up signal - HT32

     BC7701_TransmitPackageConst((tBCI_PACKAGE *)&BC7701_OperateSleep): Setup the BM7701-00-1 to enter the Deep Sleep mode and MCU external wake-up signal - HT8
4. Setup Connection Interval
   - BC7701_DummyWakeup():Wake-up the BM7701-00-1 from the Sleep mode
   - BC7701_ConnectIntervalModify(u16 opcode, u16 min, u16 max): Setup the BM7701-00-1 connection interval - HT32

     BC7701_TransmitPackageConst((tBCI_PACKAGE *)BLE_CmdIndex): Setup the BM7701-00-1 connection interval - HT8

5. Send Data
   - BC7701_DummyWakeup(): Wake-up the BM7701-00-1 from the Sleep mode
   - BC7701_SendBCIPackage(NotifyFFF1, BCI_ServiceProperties, len, pbuf): Setup the BM7701-00-1 to send wireless data to the smartphone App.

Module API Function Introduction

1. The following table shows the BM7701-00-1 API command functions and related functions. Refer to the BC7701.c file for the application example program - HT32.

API Prototype	Functional Description
void BC7701_InterfaceConfigure(u8 br)	Configure the port states of the MCU I/O which communicate with the BM7701-00-1
void BC7701_UARTConfigure(u8 br)	Configure the MCU communication pin to be in the UART mode and setup the baud rate
void BC7701_RESET_SET(void)	BM7701-00-1 RST_N pin is pulled high
void BC7701_RESET_CLR(void)	BM7701-00-1 RST_N pin is pulled low
void BC7701_HardwareBaudRateDefault(u8 br)	Initialise the BM7701-00-1 baud rate using the communication pin state
void BC7701_HardwareBaudRateRelease(void)	Complete the BM7701-00-1 baud rate initialisation
void BC7701_HardwareReset(void)	Initialise Bluetooth parameters using a hardware reset
void *BC7701_SoftwareReset(void)	Setup the “Software Reset” command to initialise the Bluetooth parameters. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_SoftwareResetKeep(void)	Setup the “Software Reset” command to retain the RAM Bluetooth parameters. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_SendHCIPackage(u16 opcode,u8 len,u8 *pbuf)	Setup the HCI command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_SendBCIPackage(u16 opcode,u8 flag,u8 len,u8 *pbuf)	Setup the BCI command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_SetDeviceName(u8 leng,u8 *name)	Setup the “BM7701-00-1 device name” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_SetDeviceAddress(u8 *adr,u8 type)	Setup the “BM7701-00-1 device address” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_ConnectIntervalModify(u16 opcode,u16 min,u16 max)	Setup the "BM7701-00-1 connection interval" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_SetAdvertisingData(u8 mode,u8 leng,u8 *advdata)	Setup the “BM7701-00-1 advertising data” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_SetScanResponseData(u8 leng,u8 *sdata)	Setup the “BM7701-00-1 scan response data” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_AdvertisingInterval(u16 min,u16 max,u8 chmap)	Setup the “BM7701-00-1 advertising interval” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_AdvertisingControl(ControlStatus ctrl)	Setup the “BM7701-00-1 advertising enable/disable” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_SetTxPower(u8 pwr)	Setup the “BM7701-00-1 Tx power” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.

API Prototype	Functional Description
void *BC7701_SetCrystalCload(u8 cc)	Setup the "BM7701-00-1 16MHz crystal load capacitor value, C_{Load}" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_SetupFeatureFlag(u8 md,FeatureFlag sff)	Setup the "BM7701-00-1 Feature" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_SetOperateMode(u8 omd,ControlStatus ctrl,u8 wuw,u8 wut)	Setup the "BM7701-00-1 Sleep mode and MCU external wake-up signal" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_SetWhiteList(ControlStatus erase,u8 *adr,u8 *mask)	Setup the "BM7701-00-1 whitelist" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void *BC7701_SetBaudRate(u8 br)	Setup the "BM7701-00-1 baud rate" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
bool BC7701_DummyWakeup(void)	Wake-up the BM7701-00-1 from the Sleep mode
bool BC7701_TransmitPackage(void *pbuf)	Store the command packet into send queue
bool BC7701_ReadTransmitEmpty(void)	Obtain status of the data send queue
bool BC7701_ReadReceiveEmpty(void)	Obtain status of the data receive queue
void *BC7701_ReadReceivePackage(void)	Read the BM7701-00-1 received data
void BC7701_WriteReceivePackage(void)	Write the received data index to the receive queue
void BC7701_ReceiveParserPackage(void)	Parse the BM7701-00-1 received data

Table 6. Module API Command Functions - HT32

API Name	void BC7701_InterfaceConfigure(u8 br)
Function	Configure the port states of the MCU I/O which communicate with the BM7701-00-1
Input Parameter	Baud rate: BAUD_RATE_9600 / BAUD_RATE_14400 / BAUD_RATE_19200 / BAUD_RATE_38400 / BAUD_RATE_57600 / BAUD_RATE_115200
Output Parameter	—
Program Description	In the example program, this API function is executed to configure the port states of the MCU I/O which communicate with the BM7701-00-1. Configure the MCU reset control pin as AFIO mode. This function can call the BC7701_UARTConfigure(u8 br) to configure the MCU communication pin to be in the UART mode and setup the MCU baud rate.

API Name	void BC7701_UARTConfigure(u8 br)
Function	Configure the MCU communication pin to be in the UART mode and setup the MCU baud rate
Input Parameter	Baud rate: BAUD_RATE_9600 / BAUD_RATE_14400 / BAUD_RATE_19200 / BAUD_RATE_38400 / BAUD_RATE_57600 / BAUD_RATE_115200
Output Parameter	—
Program Description	In the example program, this API function is executed to configure the MCU communication pin to be in the UART mode and setup the MCU baud rate.

API Name	void BC7701_RESET_SET(void)
Function	The BM7701-00-1 RST_N pin is pulled high
Input Parameter	—
Output Parameter	—
Program Description	In the example program, this API function is executed to pull the BM7701-00-1 RST_N pin high.

API Name	void BC7701_RESET_CLR(void)
Function	The BM7701-00-1 RST_N pin is pulled low
Input Parameter	—
Output Parameter	—
Program Description	In the example program, this API function is executed to pull the BM7701-00-1 RST_N pin low

API Name	void BC7701_HardwareBaudRateDefault(u8 br)
Function	Initialise the BM7701-00-1 baud rate using the communication pin state - refer to Table 2
Input Parameter	Baud rate: BAUD_RATE_9600/ BAUD_RATE_115200
Output Parameter	—
Program Description	In the example program, this API function is executed to initialise the BM7701-00-1 baud rate using the communication pin state according to the communication interface definition. This API function is valid only when executed immediately after a hardware reset occurs. This API function can be executed to change the communication pin state from the UART mode to the I/O mode. After the command execution is completed, the API function BC7701_HardwareBaudRateRelease() must be called to release the communication pins to the UART mode.

API Name	void BC7701_HardwareBaudRateRelease(void)
Function	Complete the BM7701-00-1 baud rate initialisation
Input Parameter	—
Output Parameter	—
Program Description	In the example program, this API function is executed to complete the BM7701-00-1 baud rate initialisation. The premise of executing this function is that BC7701_ HardwareBaudRateDefault(u8 br) has been executed. It will then wait 60ms for the BM7701-00-1 to complete its power-on reset before executing this function.

API Name	void BC7701_HardwareReset(void)
Function	Initialise Bluetooth parameters using a hardware reset – the BM7701-00-1 RST_N pin is pulled low)
Input Parameter	—
Output Parameter	—
Program Description	In the example program, when this API function has been executed, the BM7701-00-1 hardware reset will occur and the Bluetooth parameters stored in the RAM will be cleared. After this function is executed, the BM7701-00-1 needs to wait at least 60ms before the next communication. This function is equivalent to *BC7701_SoftwareReset(void)

API Name	void *BC7701_SoftwareReset(void)
Function	Setup the “Software Reset” command to initialise the Bluetooth parameters. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter	—
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the “software reset” command. When this command has been executed successfully, the BM7701-00-1 software reset will occur and the Bluetooth parameters stored in RAM will be cleared. After the BM7701-00-1 responds with an Event, it needs to wait 60ms before the next communication. This command is equivalent to BC7701_HardwareReset(void)。 Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_SoftwareResetKeep(void)
Function	Setup the “Software Reset” command to retain the RAM Bluetooth parameters. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter	—
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the "software reset" command - not Initial Parameter. After the command has been executed successfully, the BM7701-00-1 software reset will occur however the Bluetooth parameters in the RAM will be retained. After the BM7701-00-1 responds with an Event, it needs to wait 3ms before the next communication. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_SendHCIPackage(u16 opcode,u8 len,u8 *pbuf)
Function	Setup the HCI command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	HCI opcode - refer to the "BLE_API" document
Input Parameter 2	Data length
Input Parameter 3	Buffer for storing data
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the HCI command. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_SendBCIPackage(u16 opcode,u8 flag,u8 len,u8 *pbuf)
Function	Setup the “BCI - BLE Controller Interface” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	BCI opcode - refer to the "BLE_API" document
Input Parameter 2	BCI flags - refer to the "BLE_API" document
Input Parameter 3	Data length
Input Parameter 4	Buffer for storing data
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the BCI (BLE Controller Interface) command. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_SetDeviceName(u8 leng,u8 *name)
Function	Setup the “BM7701-00-1 device name” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Data length: ≤31- byte
Input Parameter 2	Buffer for storing the name data
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the “BM7701-00-1 device name” command. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_SetDeviceAddress(u8 *adr,u8 type)
Function	Setup the “BM7701-00-1 device address” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	6-byte buffer for storing the Bluetooth address
Input Parameter 2	Bluetooth address type: STATIC_ADDRESS/RANDOM_ADDRESS
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the “BM7701-00-1 device address” command. Input Parameter 2: 1. When the Input Parameter 2 is equal to STATIC_ADDRESS, the Bluetooth address type is setup to be a static address. 2. When the Input Parameter 2 is equal to RANDOM_ADDRESS, the Bluetooth address type is setup to be a random address Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_ConnectIntervalModify(u16 opcode,u16 min,u16 max)
Function	Setup the "BM7701-00-1 connection interval" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Opcode: BCI_CONN_INTV / BCI_CONN_INTV1 - refer to the “BLE_API” document
Input Parameter 2	Minimum connection interval, step: 1.25, valid range: 7.5ms~4s
Input Parameter 3	Maximum connection interval, step: 1.25, valid range: 7.5ms~4s. This parameter is valid only when the opcode is BCI_CONN_INTV1.
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the"BM7701-00-1 connection interval" command. Input Parameter 1: 1. When the Input Parameter 1 is equal to BCI_CONN_INTV, only Input Parameter 2 is valid and will setup a unique connection interval. 2. When the Input Parameter 1 is equal to BCI_CONN_INTV1, the Input Parameter 2 and the Input Parameter 3 are both valid. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_SetAdvertisingData(u8 mode,u8 leng,u8 *advdata)
Function	Setup the “BM7701-00-1 advertising data” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Advertising mode: ADV_JOIN_NAME / ADV_UNJOIN_NAME - refer to the “BLE_API” document
Input Parameter 2	Advertising data length: ≤31-bytes
Input Parameter 3	Buffer for storing advertising data
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the “BM7701-00-1 advertising data” command. Input Parameter 1: 1. When the Input Parameter 1 is equal to ADV_JOIN_NAME, the Bluetooth device name will be automatically added to the advertising data, provided that the advertising data is less than or equal to 31-bytes. 2. When the Input Parameter 1 is equal to ADV_UNJOIN_NAME, the advertising data will not be added to the Bluetooth device name. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_SetScanResponseData(u8 leng,u8 *sdata)
Function	Setup the "BM7701-00-1 scan response data" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Scan response data length: ≤31-bytes
Input Parameter 2	Buffer for storing the scan response data
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 scan response data" command. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_AdvertisingInterval(u16 min,u16 max,u8 chmap)
Function	Setup the "BM7701-00-1 advertising interval" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Minimum advertising interval, step: 0.625, range: 20ms ~ 10.24s
Input Parameter 2	Maximum advertising interval, step: 0.625, range: 20ms~10.24s
Input Parameter 3	Advertising channel
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 advertising interval" command. Input Parameter 3: 1. When the Input Parameter 3 is equal to 0B xxx1 (B[0]=1), setup the advertising channel 37 - 2402MHz. 2. When the Input Parameter 3 is equal to 0B xx1x (B[1]=1), setup the advertising channel 38 - 2426MHz. 3. When the Input Parameter 3 is equal to 0B x1xx (B[2]=1), setup the advertising channel 39 - 2480MHz. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_AdvertisingControl(ControlStatus ctrl)
Function	Setup the "BM7701-00-1 advertising enable/disable" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter	ENABLE/DISABLE
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 advertising enable/disable" command. Input Parameter: 1. When the Input Parameter is equal to ENABLE, the Bluetooth advertising function is enabled. 2. When the Input Parameter is equal to DISABLE, the Bluetooth advertising function is disabled. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_SetTxPower(u8 pwr)			
Function	Setup the "BM7701-00-1 Tx power" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.			
Input Parameter	0~15			
Output Parameter	Command packet pointer			
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 Tx power" command. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.			
Input Parameter	0	5	10	15
Tx Power(dbm)	-32.5	-11.5	0.5	3.5

Note: The Tx power value is for reference only. The specific value should be based on the actual obtained module data.

API Name	void *BC7701_SetCrystalCload(u8 cc)			
Function	Setup the "BM7701-00-1 16MHz crystal load capacitor value, C_{Load}" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.			
Input Parameter	0~15			
Output Parameter	Command packet pointer			
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 16MHz crystal load capacitor value, C_{Load}" command. It is recommended that the BM7701-00-1 uses 04 as an Input Parameter. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.			
Input Parameter	0	5	10	15
Crystal Frequency(MHz)	15.99985	16.00004	16.00031	16.00074

Note: The crystal oscillator frequency is for reference only. The specific value should be based on the actual obtained module data.

API Name	void *BC7701_SetupFeatureFlag(u8 md,FeatureFlag sff)
Function	Setup the "BM7701-00-1 Feature" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Data update mode: FEATURE_DIR / FEATURE_SET / FEATURE_CLR
Input Parameter 2	Mode: FEATURE_NO_APPED_NAME / FEATURE_PARAM_UPDATE / FEATURE_PARAM_ERASE / FEATURE_STATUS_EVENT / FEATURE_EXTERNAL32K / FEATURE_FORCE_CALIB / FEATURE_CK32K_OUTPUT - refer to the "BLE_API" document
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 Feature" command - refer to the "BLE_API" document Input Parameter 1: 1. When the Input Parameter 1 is equal to FEATURE_DIR, the Input Parameter 2 will directly overwrite the previous data. 2. When the Input Parameter 1 is equal to FEATURE_SET, the Input Parameter 2 will execute an "OR" operation with the previous data. 3. When the Input Parameter 1 is equal to FEATURE_CLR, the Input Parameter 2 will execute an "AND" operation with the previous data. Input Parameter 2: 1. When the Input Parameter 2 is equal to FEATURE_NO_APPED_NAME, the advertising data will not be added to the Bluetooth device name. 2. When the Input Parameter 2 is equal to FEATURE_PARAM_UPDATE, the Bluetooth parameters in the BM7701-00-1 RAM will be written to the Flash memory. 3. When the Input Parameter 2 is equal to FEATURE_PARAM_ERASE, the Bluetooth parameters in the BM7701-00-1 Flash memory will be deleted and the default values will be restored.

API Name	void *BC7701_SetupFeatureFlag(u8 md,FeatureFlag sff)
	4. When the Input Parameter 2 is equal to FEATURE_STATUS_EVENT, the BM7701-00-1 will automatically send a Status Event when the communication state has changed. 5. When the Input Parameter 2 is equal to FEATURE_EXTERNAL32K, the external 32.768kHz crystal oscillator will be enabled. 6. When the Input Parameter 2 is equal to FEATURE_FORCE_CALIB, a calibration will be forced when the next software reset occurs. 7. When the Input Parameter 2 is equal to FEATURE_CK32K_OUTPUT, the BM7701-00-1 UART2_TX(PB6) will output a 32kHz square wave. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_SetOperateMode(u8 omd,ControlStatus ctrl,u8 wuw,u8 wut)
Function	Setup the “BM7701-00-1 Sleep mode and MCU external wake-up signal” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Mode: OP_NORMAL / OP_DEEPSLEEP / OP_POWERDOWN
Input Parameter 2	MCU Sleep mode: ENABLE / DISABLE
Input Parameter 3	MCU external wake-up signal width: 0~8 byte
Input Parameter 4	MCU external wake-up signal delay time: 0~20ms
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the “BM7701-00-1 Sleep mode and MCU external wake-up signal” command - refer to the “BLE_API” document Input Parameter 1: 1. When the Input Parameter 1 is equal to OP_NORMAL, the BM7701-00-1 enters Normal mode. The BM7701-00-1 operates in Normal mode by default. 2. When the Input Parameter 1 is equal to OP_DEEPSLEEP, the BM7701-00-1 enters Deep Sleep mode. The BM7701-00-1 can be woken up to Normal mode using the API function, BC7701_DummyWakeup(void). 3. When the Input Parameter 1 is equal to OP_POWERDOWN, BM7701-00-1 enters Power Down mode. The BM7701-00-1 can be woken up to Normal mode using the API function, BC7701_DummyWakeup(void). Input Parameter 2: 1. When the Input Parameter 2 is equal to ENABLE, the MCU will enters Sleep mode. The BM7701-00-1 will be setup to send a wake-up signal to wake up the MCU. 2. When the Input Parameter 2 is equal to DISABLE, the MCU will not enter Sleep mode. It is not necessary to setup the BM7701-00-1 to send a wake-up signal. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_SetWhiteList(ControlStatus erase,u8 *adr,u8 *mask)
Function	Setup the “BM7701-00-1 whitelist addresses” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Erase previous whitelist addresses: ENABLE/DISABLE
Input Parameter 2	6-byte buffer for storing whitelist addresses
Input Parameter 3	6-byte buffer for storing the address mask
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the “BM7701-00-1 whitelist” command. Ex: Whitelist address=0x112233445566, address mask=0xFFFFFFFFFF00. Only the Bluetooth addresses ranged from 0x112233445500 to 0x1122334455FF can be connected. If the address mask are all written 0, all Bluetooth addresses can be connected. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void *BC7701_SetBaudRate(u8 br)
Function	Setup the "BM7701-00-1 baud rate" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter	Baud rate: BAUD_RATE_9600 / BAUD_RATE_14400 / BAUD_RATE_19200 / BAUD_RATE_38400 / BAUD_RATE_57600 / BAUD_RATE_115200
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 baud rate" command. After this command has been executed and the BM7701-00-1 has responded with Event, it needs to wait about 1ms before the MCU will change the UART baud rate and execute the next communication transmission. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	bool BC7701_DummyWakeup(void)
Function	Wake-up the BM7701-00-1 from the Sleep mode
Input Parameter	—
Output Parameter	TRUE: "Dummy wakeup" command has been sent successfully FALSE: "Dummy wakeup" command has failed to be sent
Program Description	In the example program, this API function is executed to wake-up the BM7701-00-1 from the Sleep mode.

API Name	bool BC7701_TransmitPackage(void *pbuf)
Function	Store the command packet into send queue. Refer to the "Command/Event Format" for more information about the package format
Input Parameter	Buffer for storing packet data
Output Parameter	TRUE: The send queue has been placed successfully FALSE: The send queue has been full
Program Description	In the example program, when this API function is executed, the MCU places the packets in the send queue. The packets in the send queue will be transmitted to the BM7701-00-1 in sequence.

API Name	bool BC7701_ReadTransmitEmpty(void)
Function	Obtain status of the data send queue
Input Parameter	—
Output Parameter	TRUE: The sent queue is empty FALSE: The sent queue is not empty
Program Description	In the example program, this API function is executed to obtain status of the data send queue.

API Name	bool BC7701_ReadReceiveEmpty(void)
Function	Obtain status of the data receive queue
Input Parameter	—
Output Parameter	TRUE: The sent queue is empty FALSE: The sent queue is not empty
Program Description	In the example program, this API function is executed to obtain status of the data receive queue.

API Name	void *BC7701_ReadReceivePackage(void)
Function	Read the BM7701-00-1 received data
Input Parameter	—
Output Parameter	Packet pointer
Program Description	In the example program, this API function is executed to obtain the data placed in the receive queue. If the output is NULL pointer, this means no data.

API Name	void BC7701_WriteReceivePackage(void)
Function	Write the received data index to the receive queue
Input Parameter	—
Output Parameter	—
Program Description	In the example program, this API function is executed to write the receive data index to the receive queue.

API Name	void BC7701_ReceiveParserPackage(void)
Function	Parse the BM7701-00-1 received data
Input Parameter	—
Output Parameter	—
Program Description	In the example program, this API function is executed to parse the received BM7701-00-1 data, convert the serial data into packet data and place it in the receive queue.

2. The following table shows the BM7701-00-1 module API command functions and some important macro definitions. Refer to the BC7701.c and BC7701.h files - HT8 for the application example program.

API Prototype	Functional Description
void BC7701_InterfaceConfigure(u8 br)	Configure the port states of the MCU I/O which communicate with the BM7701-00-1
void BC7701_UARTConfigure(u8 br)	Configure the MCU communication pin as UART mode and setup the MCU baud rate
void BC7701_UARTWakeUpCtrl(u8 ctrl)	Setup the MCU UART wake-up function
void BC7701_HardwareBaudRateDefault(u8 br)	Initialise the BM7701-00-1 baud rate using the communication pin state
void BC7701_HardwareBaudRateRelease(void)	Complete the BM7701-00-1 baud rate initialisation
void BC7701_SoftwareReset(void)	Send the “BM7701-00-1 Software Reset” command to initialise the Bluetooth parameters. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_SendHCIPackage(u16 opcode, u8 len,u8 *pbuf)	Setup the HCI command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_SendBCIPackage(u16 opcode, u8 flag,u8 len,u8 *pbuf)	Setup the BCI command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_SetDeviceName(u8 leng,u8 *name)	Setup the “BM7701-00-1 device name” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_SetDeviceAddress(u8 *adr,u8 type)	Setup the “BM7701-00-1 device address” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_ConnectIntervalModify(u16 opcode,u16 min,u16 max)	Setup the "BM7701-00-1 connection interval" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_SetAdvertisingData(u8 mode, u8 leng,u8 *advdata)	Setup the “BM7701-00-1 advertising data” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_SetScanResponseData(u8 leng,u8 *sdata)	Setup the “BM7701-00-1 scan response data” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_AdvertisingInterval(u16 min,u16 max,u8 chmap)	Setup the “BM7701-00-1 advertising interval” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_AdvertisingControl(u8 ctrl)	Setup the “BM7701-00-1 advertising enable/disable” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_SetTxPower(u8 pwr)	Setup the “BM7701-00-1 Tx power” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_SetCrystalCload(u8 cc)	Setup the “BM7701-00-1 16MHz crystal load capacitor value, C_{Load}” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.

API Prototype	Functional Description
void BC7701_SetupFeatureFlag(u8 md, FeatureFlag sff)	Setup the “BM7701-00-1 Feature” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_SetOperateMode(u8 omd,u8 wue, u8 wuw,u8 wut)	Setup the “BM7701-00-1 Sleep mode and MCU external wake-up signal” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_SetWhiteList(u8 erase,u8 *adr,u8 *mask)	Setup the “BM7701-00-1 whitelist addresses” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_SetBaudRate(u8 br)	Setup the “BM7701-00-1 baud rate” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
void BC7701_DummyWakeup(u8 leng)	Wake-up the BM7701-00-1 from the Sleep mode
void BC7701_TransmitCmdPackage(u16 opcode,u8 flag)	Send the BCI "READ" command to BM7701-00-1
void BC7701_TransmitPackage(u8 length)	Send a TransmitData[] packet to the BM7701-00-1
Void BC7701_TransmitPackageConst (tBCI_PACKAGE *pkg)	Send a BCI packet data to BM7701-00-1
u8 BC7701_PackageParserProcess(void)	Parse the BM7701-00-1 received data
#define BC7701_RESET_SET()	The BM7701-00-1 RST_N pin is pulled high
#define BC7701_RESET_CLR()	The BM7701-00-1 RST_N pin is pulled low

Table 7. Module API Command Functions - HT8

API Name	void BC7701_InterfaceConfigure(u8 br)
Function	Configure the port states of the MCU I/O which communicate with the BM7701-00-1
Input Parameter	Baud rate: BAUD_RATE_4800 / BAUD_RATE_9600 / BAUD_RATE_19200 / BAUD_RATE_25000 / BAUD_RATE_50000 / BAUD_RATE_62500 / BAUD_RATE_115200
Output Parameter	—
Program Description	In the example program, this API function is executed to configure the port states of the MCU I/O which communicate with the BM7701-00-1. Configure the MCU reset control pin to be in the I/O mode and the MCU communication pin to be in the UART mode. This function can call the BC7701_UARTConfigure(u8 br) to setup MCU UART parameters.

API Name	void BC7701_UARTConfigure(u8 br)
Function	Setup the MCU UART parameters
Input Parameter	Baud rate: BAUD_RATE_4800 / BAUD_RATE_9600 / BAUD_RATE_19200 / BAUD_RATE_25000 / BAUD_RATE_50000 / BAUD_RATE_62500 / BAUD_RATE_115200
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the MCU UART parameters, including the MCU baud rate.

API Name	void BC7701_UARTWakeUpCtrl(u8 ctrl)
Function	Setup the MCU UART wake-up function
Input Parameter	ENABLE / DISABLE
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the MCU UART wake-up function. Input Parameter 1: 1. When the Input Parameter 1 is equal to ENABLE, the MCU UART wake-up function is enabled. 2. When the Input Parameter 1 is equal to DISABLE, the MCU UART wake-up function is disabled.

API Name	void BC7701_HardwareBaudRateDefault(u8 br)
Function	Initialise the BM7701-00-1 baud rate using the communication pin state - refer to Table 2
Input Parameter	Baud rate: BAUD_RATE_9600/ BAUD_RATE_115200
Output Parameter	—
Program Description	In the example program, this API function is executed to initialise the BM7701-00-1 baud rate using the communication pin state according to the definition of the communication interface. This API function is valid only when executed immediately after a hardware reset occurs. This API function can be executed to change the communication pin state from the UART mode to the I/O mode. After the command execution has completed, the API function BC7701_HardwareBaudRateRelease() must be called to release the communication pins to the UART mode.

API Name	void BC7701_HardwareBaudRateRelease(void)
Function	Complete the BM7701-00-1 baud rate initialisation
Input Parameter	—
Output Parameter	—
Program Description	In the example program, this API function is executed to complete the BM7701-00-1 baud rate initialisation. The premise of executing this function is that BC7701_ HardwareBaudRateDefault(u8 br) has been executed. Then it will wait 60ms for the BM7701-00-1 power-on reset before executing this function.

API Name	void BC7701_SoftwareReset(void)
Function	Setup the “BM7701-00-1 Software Reset” command to initialise the Bluetooth parameters. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter	—
Output Parameter	—
Program Description	In the example program, this API function is executed to send the “BM7701-00-1 software reset” command. The Bluetooth parameters stored in RAM will be cleared. After the BM7701-00-1 responds with an Event, it needs to wait 60ms before the next communication.

API Name	void BC7701_SendHCIPackage(u16 opcode,u8 len,u8 *pbuf)
Function	Setup the HCI command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	HCI opcode - refer to the "BLE_API" document
Input Parameter 2	Data length
Input Parameter 3	Buffer for storing data
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the HCI command and store it to TransmitData[].

API Name	void BC7701_SendBCIPackage(u16 opcode,u8 flag,u8 len,u8 *pbuf)
Function	Setup the “BCI - BLE Controller Interface” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	BCI opcode - refer to the "BLE_API" document
Input Parameter 2	BCI flags - refer to the "BLE_API" document
Input Parameter 3	Data length
Input Parameter 4	Buffer for storing data
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the BCI command and store this to TransmitData[].

API Name	void BC7701_SetDeviceName(u8 leng,u8 *name)
Function	Setup the “BM7701-00-1 device name” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Data length: ≤31-bytes
Input Parameter 2	Buffer for storing the name data
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the “BM7701-00-1 device name” command and store this command data to TransmitData[].

API Name	void BC7701_SetDeviceAddress(u8 *adr,u8 type)
Function	Setup the “BM7701-00-1 device address” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	6-byte buffer for storing the Bluetooth address
Input Parameter 2	Bluetooth address type: STATIC_ADDRESS/RANDOM_ADDRESS
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the “BM7701-00-1 device address” command and store this command data to TransmitData[]. Input Parameter 2: 1. When the Input Parameter 2 is equal to STATIC_ADDRESS, the Bluetooth address type is setup to be a static address. 2. When the Input Parameter 2 is equal to RANDOM_ADDRESS, the Bluetooth address type is setup to be a random address.

API Name	void BC7701_ConnectIntervalModify(u16 opcode,u16 min,u16 max)
Function	Setup the "BM7701-00-1 connection interval" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Opcode: BCI_CONN_INTV / BCI_CONN_INTV1 - refer to the “BLE_API” document
Input Parameter 2	Minimum connection interval, step: 1.25, valid range: 7.5ms~4s
Input Parameter 3	Maximum connection interval, step: 1.25, valid range: 7.5ms~4s. This parameter is valid only when the opcode is BCI_CONN_INTV1.
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the"BM7701-00-1 connection interval" command and store this command data to TransmitData[]. Input Parameter 1: 1. When the Input Parameter 1 is equal to BCI_CONN_INTV, only Input Parameter 2 is valid and setup a unique connection interval. 2. When the Input Parameter 1 is equal to BCI_CONN_INTV1, the Input Parameter 2 and the Input Parameter 3 are both valid.

API Name	void BC7701_SetAdvertisingData(u8 mode,u8 leng,u8 *advdata)
Function	Setup the “BM7701-00-1 advertising data” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Advertising mode: UPDAE_AUTO_NAME / DEFAULT_NAME / DEFAULT_EMPTY / UPDAE_NOAO_NAME - refer to the “BLE_API” document
Input Parameter 2	Advertising data length: ≤31-bytes
Input Parameter 3	Buffer for storing advertising data
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the “BM7701-00-1 advertising data” command and store this command data to TransmitData[]. Input Parameter 1: 1. If the Input Parameter 1 is equal to DEFAULT_NAME, the advertising data uses a default value. 2. When the Input Parameter 1 is equal to UPDAE_AUTO_NAME, the Bluetooth device name will be automatically added to the advertising data, provided that the advertising data is less than or equal to 31 bytes. 3. If the Input Parameter 1 is equal to DEFAULT_EMPTY, the advertising data is empty. 4. If the Input Parameter 1 is equal to UPDAE_NOAO_NAME, the advertising data will not be added to the Bluetooth device name.

API Name	void BC7701_SetScanResponseData(u8 leng,u8 *sdata)
Function	Setup the "BM7701-00-1 scan response data" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Scan response data length: ≤31-bytes
Input Parameter 2	Buffer for storing scan response data
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 scan response data" command and store this command data to TransmitData[]. Output Parameter: 1. When the Output Parameter is equal to NULL, the API function has failed to configure. 2. When the Output Parameter is not equal to NULL, the API function has been configured successfully.

API Name	void BC7701_AdvertisingInterval(u16 min,u16 max,u8 chmap)
Function	Setup the "BM7701-00-1 advertising interval" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Minimum advertising interval, step: 0.625, range: 20ms ~ 10.24s
Input Parameter 2	Minimum advertising interval, step: 0.625, range: 20ms ~ 10.24s
Input Parameter 3	Advertising channel
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 advertising interval" command and store this command data to TransmitData[]. Input Parameter 3: 1. When the Input Parameter 3 is equal to 0B xxx1 (B[0]=1), setup the advertising channel 37 - 2402MHz. 2. When the Input Parameter 3 is equal to 0B xx1x (B[1]=1), setup the advertising channel 38 - 2426MHz. 3. When the Input Parameter 3 is equal to 0B x1xx (B[2]=1), setup the advertising channel 39 - 2480MHz.

API Name	void BC7701_AdvertisingControl(ControlStatus ctrl)
Function	Setup the "BM7701-00-1 advertising enable/disable" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter	ENABLE/DISABLE
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 advertising enable/disable" command and store this command data to TransmitData[]. Input Parameter: 1. When the Input Parameter is equal to ENABLE, the Bluetooth advertising function is enabled. 2. When the Input Parameter is equal to DISABLE, the Bluetooth advertising function is disabled.

API Name	void BC7701_SetTxPower(u8 pwr)			
Function	Setup the "BM7701-00-1 Tx power" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.			
Input Parameter	0~15			
Output Parameter	—			
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 Tx power" command and store this command data to TransmitData[].			
Input Parameter	0	5	10	15
Tx Power(dbm)	-32.5	-11.5	0.5	3.5

Note: The Tx power value is for reference only. The specific value should be based on the actual obtained module data.

API Name	void BC7701_SetCrystalCload(u8 cc)			
Function	Setup the "BM7701-00-1 16MHz crystal load capacitor value, C_{Load}" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.			
Input Parameter	0~15			
Output Parameter	—			
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 16MHz crystal load capacitor value, C_{Load}" command and store this command data to TransmitData[]. The BM7701-00-1 is recommended to use 04 as Input Parameter.			
Input Parameter	0	5	10	15
Crystal Frequency(MHz)	15.99985	16.00004	16.00031	16.00074

Note: The crystal oscillator frequency is for reference only. The specific value should be based on the actual obtained module data.

API Name	void BC7701_SetupFeatureFlag(u8 md,FeatureFlag sff)
Function	Setup the "BM7701-00-1 Feature" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Data update mode: FEATURE_DIR / FEATURE_SET / FEATURE_CLR
Input Parameter 2	Mode: FEATURE_NO_APPED_NAME / FEATURE_PARAM_UPDATE / FEATURE_PARAM_ERASE / FEATURE_STATUS_EVENT / FEATURE_EXTERNAL32K / FEATURE_FORCE_CALIB / FEATURE_CK32K_OUTPUT - refer to the "BLE_API" document
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 Feature" command and store this command data to TransmitData[] - refer to the"BLE_API" document Input Parameter 1: 1. When the Input Parameter 1 is equal to FEATURE_DIR, the Input Parameter 2 will directly overwrite the previous data. 2. When the Input Parameter 1 is equal to FEATURE_SET, the Input Parameter 2 will execute an "OR" operation with the previous data. 3. When the Input Parameter 1 is equal to FEATURE_CLR, the Input Parameter 2 will execute an "AND" operation with the previous data. Input Parameter 2: 1. When the Input Parameter 2 is equal to FEATURE_NO_APPED_NAME, the advertising data will not be added to the Bluetooth device name. 2. When the Input Parameter 2 is equal to FEATURE_PARAM_UPDATE, the Bluetooth parameters in the BM7701-00-1 RAM will be written to Flash memory. 3. When the Input Parameter 2 is equal to FEATURE_PARAM_ERASE, the Bluetooth parameters in the BM7701-00-1 Flash memory will be erased and restore the default values. 4. When the Input Parameter 2 is equal to FEATURE_STATUS_EVENT, the BM7701-00-1 will automatically send a Status Event when the communication state has changed. 5. When the Input Parameter 2 is equal to FEATURE_EXTERNAL32K, the external 32.768kHz crystal oscillator will be enabled. 6. When the Input Parameter 2 is equal to FEATURE_FORCE_CALIB, the calibration will be forced when the next software reset occurs. 7. When the Input Parameter 2 is equal to FEATURE_CK32K_OUTPUT, the BM7701-00-1 UART2_TX(PB6) will output a 32kHz square wave.

API Name	void BC7701_SetOperateMode(u8 omd,u8 wue,u8 wuw,u8 wut)
Function	Setup the "BM7701-00-1 Sleep mode and MCU external wake-up signal" command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Mode: OP_NORMAL / OP_DEEPSLEEP / OP_POWERDOWN
Input Parameter 2	MCU Sleep mode: ENABLE / DISABLE
Input Parameter 3	MCU external wake-up signal width: 0~8 bytes
Input Parameter 4	MCU external wake-up signal delay time: 0~20ms
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the "BM7701-00-1 Sleep mode and MCU external wake-up signal" command and store this command data to TransmitData[] - refer to the "BLE_API" document Input Parameter 1: 1. When the Input Parameter 1 is equal to OP_NORMAL, the BM7701-00-1 enters Normal mode. The BM7701-00-1 operates in the Normal mode by default.

API Name	void BC7701_SetOperateMode(u8 omd,u8 wue,u8 wuw,u8 wut)
	2. When the Input Parameter 1 is equal to OP_DEEPSLEEP, the BM7701-00-1 enters Deep Sleep mode. The BM7701-00-1 can be woken up to the Normal mode using the API function, BC7701_DummyWakeup(u8 leng). 3. When the Input Parameter 1 is equal to OP_POWERDOWN, the BM7701-00-1 enters the Power Down mode. The BM7701-00-1 can be woken up to the Normal mode using the API function, BC7701_DummyWakeup(u8 leng). Input Parameter 2: 1. When the Input Parameter 2 is equal to ENABLE, the MCU will enter the Sleep mode. The BM7701-00-1 will be setup to send a wake-up signal to wake up the MCU. 2. When Input Parameter 2 is equal to DISABLE, the MCU will not enter the Sleep mode. It is not necessary to setup the BM7701-00-1 to send a wake-up signal.

API Name	void BC7701_SetWhiteList(ControlStatus erase,u8 *adr,u8 *mask)
Function	Setup the “BM7701-00-1 whitelist addresses” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter 1	Erase the previous whitelist addresses: ENABLE/DISABLE
Input Parameter 2	6-byte buffer for storing whitelist addresses
Input Parameter 3	6-byte buffer for storing the address mask
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the “BM7701-00-1 whitelist addresses” command and store this command data to TransmitData[]. Ex: Whitelist address=0x112233445566, address mask=0xFFFFFFFFFF00. Only the Bluetooth addresses ranging from 0x112233445500 to 0x1122334455FF can be connected. If the address masks are all written be 0, all Bluetooth addresses can be connected.

API Name	void BC7701_SetBaudRate(u8 br)
Function	Setup the “BM7701-00-1 baud rate” command. This will need to be transmitted by calling the BC7701_TransmitPackage function.
Input Parameter	Baud rate: BAUD_RATE_4800 / BAUD_RATE_9600 / BAUD_RATE_19200 / BAUD_RATE_25000 / BAUD_RATE_50000 / BAUD_RATE_62500 / BAUD_RATE_115200
Output Parameter	—
Program Description	In the example program, this API function is executed to setup the “BM7701-00-1 baud rate” command and store this command data to TransmitData[]. The BM7701-00-1 will respond with an EVENT, it then needs to wait about 1ms before the MCU will change the UART baud rate and execute the next communication transmission.

API Name	void BC7701_DummyWakeup(u8 leng)
Function	Wake-up the BM7701-00-1 from the Sleep mode
Input Parameter	Dummy wakeup signal(0x00) used byte
Output Parameter	—
Program Description	In the example program, this API function is executed to wake-up the BM7701-00-1 from the Sleep mode. It is recommended that the Input Parameter be set to 1.

API Name	void BC7701_TransmitCmdPackage(u16 opcode,u8 flag)
Function	Send the BCI “READ” command to the BM7701-00-1
Input Parameter 1	BCI opcode - refer to the "BLE_API" document
Input Parameter 2	BCI flags - refer to the "BLE_API" document
Output Parameter	—
Program Description	In the example program, when this API function is executed, the MCU will send a BCI "READ" command to BM7701-00-1.

API Name	void BC7701_TransmitPackage(u8 length)
Function	Send TransmitData[] packet to BM7701-00-1
Input Parameter	Sent packet length
Output Parameter	—
Program Description	In the example program, when this API function is executed, the MCU will send the TransmitData[] data to BM7701-00-1. If the TransmitData[0] detects that it is either HCI or BCI type, this Input Parameter will be ignored, and the sent packet length will be equal to TransmitData[1]+2 (the packet length of the HCI or BCI command).

API Name	void BC7701_TransmitPackageConst(tBCI_PACKAGE *pkg)
Function	Send the BCI packet data to the BM7701-00-1
Input Parameter	Buffer for storing the BCI packet data
Output Parameter	—
Program Description	In the example program, when this API function is executed, the MCU will send packet data to the BM7701-00-1.

API Name	u8 BC7701_PackageParserProcess(void)
Function	Parse the EVENT received by BM7701-00-1
Input Parameter	—
Output Parameter	TRUE / FALSE
Program Description	In the example program, this API function is executed to parse the EVENT received by BM7701-00-1. Output Parameter: 1. If the Output Parameter is equal to TRUE, the received EVENT is valid. 2. If the Output Parameter is equal to FALSE, the received EVENT is invalid.

API Name	#define BC7701_RESET_SET()
Function	The BM7701-00-1 RST_N pin is pulled high
Program Description	In the example program, this macro definition is used to pull the BM7701-00-1 RST_N pin high

API Name	#define BC7701_RESET_CLR()
Function	The BM7701-00-1 RST_N pin is pulled low
Program Description	In the example program, this macro definition is used to pull the BM7701-00-1 RST_N pin low

Bluetooth Parameter Modification - HT32

1. Open the bleprocess.h file and click “Configuration Wizard” to enter the "Bluetooth Parameters" setting page, as shown in Figure 12. Each parameter option will be described in detail in the next section.

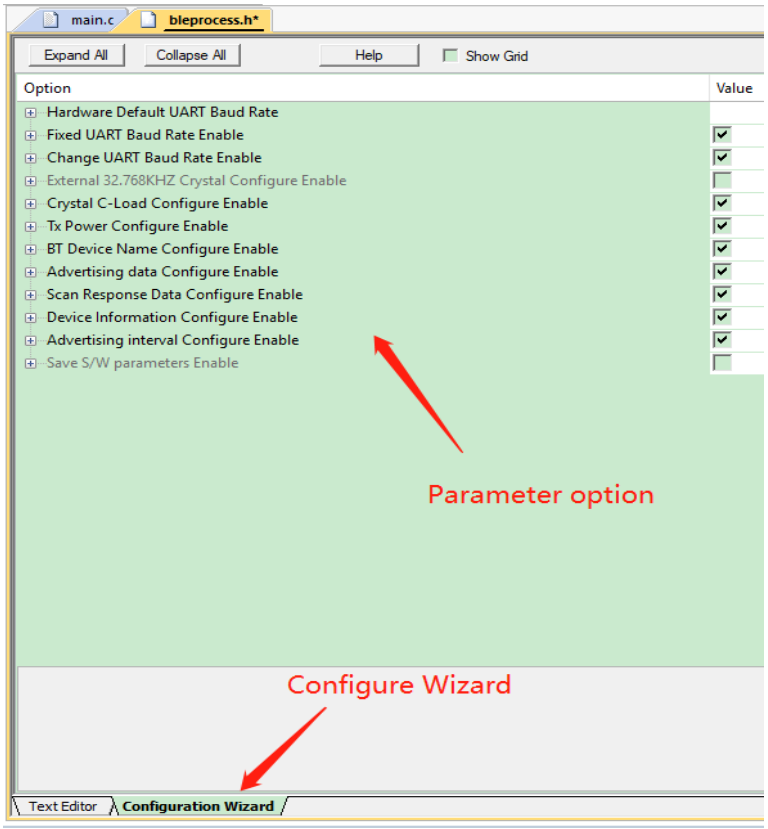

Figure 12

- Hardware Default UART Baud Rate: Setup the BM7701-00-1 default baud rate after a power-on reset for 60ms. Only two baud rates, 9600bps or 115200bps can be selected, as shown in Figure 13. This is a hardware method to modify the BM7701-00-1 baud rate. Refer to Table 2.

Figure 13

- Fixed UART Baud Rate Enable: Click to enable "Setup the MCU default baud rate" and click the additional item “Baud Rate” to select a baud rate, as shown in Figure 14.
  - ➢ Disable: Automatically setup the MCU default baud rate equal to the BM7701-00-1 default baud rate - select 9600bps or 115200bps.
  - ➢ Enable: The user needs to manually select the MCU default baud rate.

  Ex: The BM7701-00-1 stores a new baud rate parameter to the Flash memory by clicking “Save S/W parameters Enable”. Refer to "Parameter setting 12 points" for details. For example, when a baud rate of 57600bps is stored to the Flash memory, the BM7701 -00-1

default baud rate will be equal to 57600bps after each power-on reset for 60ms. The user should enable this option and select the MCU default baud rate to be equal to 57600bps, otherwise the MCU and the BM7701-00-1 cannot communicate normally.

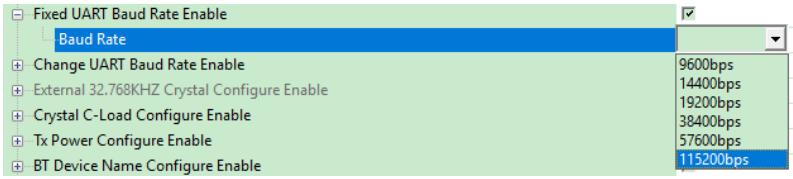

Figure 14

- Change the UART Baud Rate Enable: Click to enable “Change UART Baud Rate", click the additional item “Baud Rate” to select a baud rate, as shown in Figure 15. This option can be used to setup the baud rate of the BM7701-00-1 and the MCU at the same time. It will only be executed after the default baud rate of MCU and BM7701-00-1 has been setup and are the same (EX: The MCU first sends a “Change UART Baud Rate ” command to the BM7701-00- 1 and then the MCU will setup the baud rate of the communication pin to be the same as that of the BM7701-00-1).

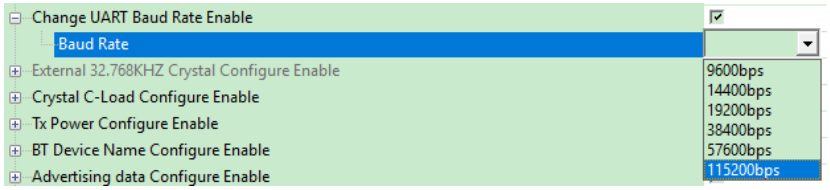

Figure 15

- External 32.768kHz Crystal Configure Enable: Click to enable "External 32.768kHz Crystal", as shown in Figure 16. Some modules include an external 32.768kHz crystal oscillator, users can enable this option. It should be disabled by default in the application example program. Note that if the BM7701-00-1 module does not have an external 32.768kHz crystal oscillator, enabling this option will cause the operation to fail.

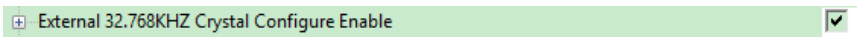

Figure 16

- Crystal C-Load Configure Enable: Click to enable "Change external 16MHz crystal oscillator CLoad value" and click the additional item “Crystal C-Load” to enter a value which ranges from 0 to 15, as shown in Figure 17. A value of 4 is recommended for the BM7701-00-1 module.

Figure 17

Input Parameter	0	5	10	15
Crystal Frequency(MHz)	15.99985	16.00004	16.00031	16.00074

Note: The crystal oscillator frequency is for reference only, the specific value should be based on the actual obtained module data.

Table 8

- Tx Power Configure Enable: Click to enable "Change Tx Power value" and click the additional item “Tx Power” to enter a value ranging from 0 to 15, as shown in Figure 18. Users can adjust this value according to the transmission distance or power consumption.

Figure 18

Input Parameter	0	5	10	15
Tx Power(dbm)	-32.5	-11.5	0.5	3.5

Note: The Tx power value is for reference only, the specific value should be based on the actual obtained module data.

Table 9

- BT Device Name Configure Enable: Click to enable "Change BT Device Name", as shown in Figure 19. The Bluetooth name after successful change is "BM7701". Users can also customise the Bluetooth name on the bleprocess.c page. Here, find the red box position shown in Figure 20 which can be changed, the maximum length is 31-bytes.

Figure 19

```
24 #if (_BDNAME_CFG_ENABLE_ == 1)
25 /* set BT device name */
26 uc8 BLE_DeviceName[] = {'B','M','7','7','0','1'};
27 #endif
28
```

Figure 20

- Advertising data Configure Enable: Click to enable "Change advertising data", as shown in Figure 21. Users can also customise the advertising data on the bleprocess.c page. Here, find the red box position shown in Figure 22, the maximum length is 31-bytes.

Figure 21

```
29 #if (_ADV_DATA_CFG_ENABLE_ == 1)
30 /* set Advertising Data */
31 uc8 BLE_AdvData[] =
32 {
33   /* flag */
34   2,0x01,0x06,
35   /* Manufacturer Specific Data */
36   11,0xFF,0xFF,0xFF,
37   'B','e','s','t','C','o','m','m'
38 };
39 #endif
```

Figure 22

- Scan Response Data Configure Enable: Click to enable "Chane Scan Response Data", as shown in Figure 23. Users can also customise the scan response data on the bleprocess.c page. Here, find the red box position shown in Figure 24, the maximum length is 31-bytes.

Figure 23

```
#if (_SCAN_DATA_CFG_ENABLE_ == 1)
/* Set Scan Response Data */
uc8 BLE_ScanData[] =
{
  /* complete List of 16-bit Service Class UUIDs */
  11,0x03,0x00,0x18,0x01,0x18,0x0A,0x18,0x0F,0x18,0xF0,0xFF
};
#endif
```

Figure 24

- Advertising interval Configure Enable: Click to enable "Change advertising interval", click the additional item to enter a value, as shown in Figure 25. It should be noted that Min Interval should be less than or equal to Max Interval.

Figure 25

- Device Information Configure Enable: Click to enable "Change Device Information", as shown in Figure 26. Users can also customise device information on the bleprocess.c page. Here, find the red box position in Figure 27 which can be changed.

Figure 26

```
#if ( DEV_INF_CFG_ENABLE  == 1)
uc8 BLE_DevInfSysID[] = { 0x37,0x37,0x30,0x31 };
uc8 BLE_DevInfModelNumber[] = {'B','M','7','7','0','1'};    //max 16 byte
uc8 BLE_DevInfSerialNumber[] = {'1','.','0','0'};
uc8 BLE_DevInfFirmwareRevs[] = {'1','.','0','0'};
uc8 BLE_DevInfHardwareRevs[] = {'1','.','0','0'};
uc8 BLE_DevInfSofewareRevs[] = {'1','.','0','0'};
uc8 BLE_DevInfManufacturer[] = {'B','e','s','t','C','o','m','m'};   //max 16 byte
uc8 BLE_DevInfIEEE11073[] = {0x00,0x00,0x00,0x00};
uc8 BLE_DevInfPnPID[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00};
#endif
```

Figure 27

- Save S/W parameters Enable: Click to enable "Store parameters". The Bluetooth parameters in "Parameter Setting 3~11" will be stored to the BM7701-00-1 Flash memory, as shown in Figure 28. The premise is that the corresponding setting options are enabled. The BM7701-00-1 Bluetooth parameters setup using command will be temporarily stored in the RAM. After the option is enabled, the Bluetooth parameters in the RAM will be stored in the Flash memory. The BM7701-00-1 will automatically use the Bluetooth parameters in the Flash memory after a power-on reset occurs.

Figure 28

2. Open the main.c file and click “Configuration Wizard” to setup the related parameters for "Data transmission" and "MCU sleep mode".

- Change Connect Interval Enable: Click to enable "Change BM7701-00-1 Connection Interval", as shown in Figure 29. Click the additional item to enter a connection interval value.

Figure 29

- MCU sleep mode Enable / disable: Click to enable "MCU sleep mode". Click the additional item to enter a time. As shown in Figure 30, when this option is enabled, if the BM7701-00-1 does not transmit data with a connected device (smartphone) within 500ms, the MCU will actively enter the Sleep mode.

Figure 30

- Wake Up Signal Configure: Click the additional item to select "Wake up Signal Width(byte)" and " Wake up Signal delay time(ms)". As shown in Figure 31, when the MCU needs to wake up from the Sleep mode, the BM7701-00-1 can be setup to send 2 bytes, composed of 0x00, to wake-up the MCU and then wait for 8ms to transmit the received data Event to the MCU. The wake-up signal width and delay time is determined by the MCU sleep mode setting and wake-up stabilisation time. The user can adjust it according to the actual situation.

Figure 31

- In order to allow users to be clear about the difference between baud rate options and the impact on the Flash storage, the following example shows how to select different baud rates. Unless otherwise specified, the baud rates refer to those of the MCU and the BM7701-00-1.

Example 1: Select a baud rate of 9600bps or 115200bps. The operating steps are shown in Table 10.

➢ The baud rate is 9600bps or 115200bps after each power-on reset for 60ms.

Hardware Default UART Baud Rate	Fixed UART Baud Rate Enable	Change UART Baud Rate Enable	Save S/W Parameters Enable
(9600bps or 115200bps)	Disable	Disable	Disable

Table 10

Example 2: Select an arbitrary baud rate. For example, when a baud rate of 57600bps is selected, the baud rate will be not stored to the BM7701-00-1 Flash memory. The operation steps are shown in Table 11.

➢ The baud rate is 9600bps or 115200bps after each power-on reset for 60ms.

➢ When the “Change UART Baud Rate” Command is executed, the baud rate will be changed to 57600bps.

- The above operations will be executed sequentially after each power-on reset.

Hardware Default UART Baud Rate	Fixed UART Baud Rate Enable	Change UART Baud Rate Enable	Save S/W Parameters Enable
(9600bps or 115200bps)	Disable	Enable (57600bps)	Disable

Table 11

Example 3: Select an arbitrary baud rate. For example, when a baud rate of 57600bps is selected, the baud rate will be stored to the BM7701-00-1 Flash memory. This should execute the programming operation twice. The operating steps are shown in Table 12.

- First programming
  - After programming, the default baud rate is 9600bps or 115200bps for the first power-on reset after 60ms.
  - When a "Change UART Baud Rate" Command is executed, the baud rate will be changed to 57600bps.
  - Store the baud rate to the BM7701-00-1 Flash memory.
  - Then, the BM7701-00-1 baud rate is always 57600bps after each power-on reset. When the MCU baud rate is 9600bps or 115200bps, the MCU and BM7701-00-1 cannot communicate normally.
  - It is required to execute a second programming.
- Second programming
  - The MCU and BM7701-00-1 can communicate normally (click to "Fixed UART Baud Rate Enable" to change the MCU default baud rate to 57600bps).

	Hardware Default UART Baud Rate	Fixed UART Baud Rate Enable	Change UART Baud Rate Enable	Save S/W Parameters Enable
First programming	(9600bps or 115200bps)	Disable	Enable (57600bps)	Enable
Second programming	(9600/115200)	Enable (57600bps)	Enable (57600bps)	Enable

Table 12

Bluetooth Parameter Modification - HT8

1. Open the bleprocess.h file and find the location as shown in Figure 32 to setup the "Bluetooth Parameters". Each parameter option will be described in detail in the next section.

```
/*------ Change UART Baud Rate Enable/Disable ---------------------*/
#define _CHG_BAUD_RATE_ENABLE_      1
//#define   _CHANGE_BAUDRATE_               BAUD_RATE_19200
//#define   _CHANGE_BAUDRATE_               BAUD_RATE_25000
//#define   _CHANGE_BAUDRATE_               BAUD_RATE_50000
#define     _CHANGE_BAUDRATE_               BAUD_RATE_115200
/*------ Change Crystal C-Load Enable/Disable ---------------------*/
#define _CLOAD_CFG_ENABLE_          1
#define     _CLOAD_VALUE_               0x4
/*------ Change TX Power Enable/Disable ---------------------------*/
#define _TXPWR_CFG_ENABLE_          1
#define     _TX_POWER_VALUE_            10
/*------ Configure BT Device Name Enable/Disable ------------------*/
#define _BDNAME_CFG_ENABLE_         1
/*------ Configure Advertising data Enable/Disable ----------------*/
#define _ADV_DATA_CFG_ENABLE_       1
/*------ Configure Scan Response Data Enable/Disable --------------*/
#define _SCAN_DATA_CFG_ENABLE_      0
/*------ Configure Advertising interval Enable/Disable ------------*/
#define _ADV_INTV_CFG_ENABLE_       1
/*------ Min Interval(ms)<10-40000> -------------------------------*/
#define     _ADV_INTV_MIN_VALUE_        (1000)
/*------ Max Interval(ms)<10-40000> -------------------------------*/
#define     _ADV_INTV_MAX_VALUE_        (1000)
/*------ Configure Feature setup ----------------------------------*/
//#define   FEATURE_STATUS_SETUP        (FEATURE_STATUS_EVENT+FEATURE_EXTERNAL32K)      /* external 32.768K */
#define FEATURE_STATUS_SETUP        (FEATURE_STATUS_EVENT)
/*------ Power On finish BC7701 to Sleep mode Enable/Disable -----*/
#define _PWRON_FINISH_SLEEP_            0
```

Figure 32

- Change UART Baud Rate
  - #define _CHG_BAUD_RATE_ENABLE_ 1: Enable “Change UART Baud Rate”
  - #define _CHG_BAUD_RATE_ENABLE_ 0: Disable “Change UART Baud Rate”
  - #define _CHANGE_BAUDRATE_ BAUD_RATE_115200: The baud rate is setup to 115200bps. Users can also define other baud rates according to their requirements.

```
/*------ Change UART Baud Rate Enable/Disable --------------------*/
#define _CHG_BAUD_RATE_ENABLE_      1
//#define    _CHANGE_BAUDRATE_              BAUD_RATE_19200
//#define    _CHANGE_BAUDRATE_              BAUD_RATE_25000
//#define    _CHANGE_BAUDRATE_              BAUD_RATE_50000
#define      _CHANGE_BAUDRATE_              BAUD_RATE_115200
```

Figure 33

- Change Crystal C-Load
  - #define _CLOAD_CFG_ENABLE_ 1: Enable “Change Crystal C-Load”
  - #define _CLOAD_CFG_ENABLE_ 0: Disable “Change Crystal C-Load”
  - #define _CLOAD_VALUE_ 04: The external 16MHz crystal oscillator C_{Load} value is setup to 04.

```
/*------ Change Crystal C-Load Enable/Disable --------------------*/
#define _CLOAD_CFG_ENABLE_          1
#define      _CLOAD_VALUE_                04
```

Figure 34

Input Parameter	0	5	10	15
Crystal Frequency(MHz)	15.99985	16.00004	16.00031	16.00074

Note: The crystal oscillator frequency is for reference only. The specific value should be based on the actual obtained module data.

Table 13

- Change Tx Power
  - #define _TXPWR_CFG_ENABLE_ 1: Enable “Change Tx Power”
  - #define _TXPWR_CFG_ENABLE_ 0: Disable “Change Tx Power”
  - #define _TX_POWER_VALUE_ 10: The Tx power value is setup to 10

```
/*------ Change TX Power Enable/Disable --------------------------*/
#define _TXPWR_CFG_ENABLE_          1
#define      _TX_POWER_VALUE_             10
```

Figure 35

Input Parameter	0	5	10	15
Tx Power(dbm)	-32.5	-11.5	0.5	3.5

Note: The Tx power value is for reference only. The specific value should be based on the actual obtained module data.

Table 14

- Configure BT Device Name
  - #define _BDNAME_CFG_ENABLE_ 1: Enable “Change BT Device Name”
  - #define _BDNAME_CFG_ENABLE_ 0: Disable “Change BT Device Name”

```
/*------ Configure BT Device Name Enable/Disable ----------------*/
#define _BDNAME_CFG_ENABLE_         1
```

Figure 36

Users can also open the bleprocess.c page and customise the Bluetooth device name by finding the red box position as shown in Figure 37. Note that the length is less than or equal to 31-bytes.

```
/*--- set BT device name ---*/
#if (_BDNAME_CFG_ENABLE_ == 1)
const tBCI_PACKAGE  BLE_SetDeviceName =
{
    BCI_CMD_PKG,                //head type
    3+6,                            //length
    0x00,                           //Flag
    BCI_DEV_NAME,               //opcode
    {'B','M','7','7','0','1'}          //parameter
};
#endif
```

Figure 37

- Configure Advertising data
  - #define _ADV_DATA_CFG_ENABLE_ 1: Enable “Change advertising data”
  - #define _ADV_DATA_CFG_ENABLE_ 0: Disable “Change advertising data”

```
/*------ Configure Advertising data Enable/Disable --------------*/
#define _ADV_DATA_CFG_ENABLE_       1
```

Figure 38

Users can also open the bleprocess.c page and customise the Bluetooth advertising data by finding the red box as shown in Figure 37. Note that the length is less than or equal to 31-bytes.

```
/*--- set Advertising Data ---*/
#if (_ADV_DATA_CFG_ENABLE_ == 1)
const tBCI_PACKAGE  BLE_SetAdvData =
{
    BCI_CMD_PKG,                //head type
    3+15,                           //length
    UPDAE_AUTO_NAME,            //Flag=auto add device name
    BCI_ADV_DATA,               //opcode
    {
        /* flag */
        2,0x01,0x06,
        /* Manufacturer Specific Data */
        11,0xFF,0xFF,0xFF,
        'B','e','s','t','C','o','m','m'
    },
};
#endif
```

Figure 39

- Configure Scan Response Data
  - #define _SCAN_DATA_CFG_ENABLE_ 1: Enable “Change scan response data”
  - #define _SCAN_DATA_CFG_ENABLE_ 0: Disable “Change scan response data”

```
/*------ Configure Scan Response Data Enable/Disable ------------*/
#define _SCAN_DATA_CFG_ENABLE_      0
```

Figure 40

Users can also open the bleprocess.c page and customise the scan response data by finding the red box position as shown in Figure 41. Note that the length is less than or equal to 31 -bytes.

```
/*--- Set Scan Response Data ---*/
#if (_SCAN_DATA_CFG_ENABLE_ == 1)
const tBCI_PACKAGE  BLE_SetScanData =
{
    BCI_CMD_PKG,                //head type
    3+12,                           //length
    0x00,                           //Flag
    BCI_SCAN_DATA,              //opcode
    {
      /* complete List of 16-bit Service Class UUIDs */
      11,0x03,0x00,0x18,0x01,0x18,0x0A,0x18,0x0F,0x18,0xF0,0xFF
    }
};
#endif
```

Figure 41

- Configure Advertising interval Enable/Disable
  - #define _ADV_INTV_CFG_ENABLE_ 1: Enable “Change advertising interval”
  - #define _ADV_INTV_CFG_ENABLE_ 0: Disable “Change advertising interval”
  - #define _ADV_INTV_MIN_VALUE_ (1000): The minimum advertising interval is seup to 1000ms
  - #define _ADV_INTV_MAX_VALUE_ (1000): The maximum advertising interval is setup to 1000ms

```
#define _ADV_INTV_CFG_ENABLE_       1
/*------ Min Interval(ms)<10-40000> ------------------------------*/
#define     _ADV_INTV_MIN_VALUE_        (1000)
/*------ Max Interval(ms)<10-40000> ------------------------------*/
#define     _ADV_INTV_MAX_VALUE_        (1000)
```

Note: The MIN_VALUE is no greater than MAX_VALUE

Figure 42

- Configure Feature setup
  - #define FEATURE_STATUS_SETUP (FEATURE_STATUS_EVENT+FEATURE_EXTERNAL32K): Enable "The BM7701-00-1 will automatically send a Status Event and enable the external 32.768KHz crystal when the communication state has changed."
  - #define FEATURE_STATUS_SETUP (FEATURE_STATUS_EVENT): Enable "The BM7701-00-1 will automatically send a Status Event when the communication state has changed."

```
/*----- Configure Feature setup ---------------------------------*/
//#define   FEATURE_STATUS_SETUP        (FEATURE_STATUS_EVENT+FEATURE_EXTERNAL32K)      /* external 32.768K */
#define FEATURE_STATUS_SETUP        (FEATURE_STATUS_EVENT)
```

Figure 43

- Power On finish BC7701 to Sleep mode Enable/Disable
  - #define _PWRON_FINISH_SLEEP_ 0: When the parameter has been configured successfully, the MCU will not enter the Sleep mode.
  - #define _PWRON_FINISH_SLEEP_ 1: When the parameter has been configured successfully, the MCU will enter the Sleep mode.

```
/*------ Power On finish BC7701 to Sleep mode Enable/Disable -----*/
#define _PWRON_FINISH_SLEEP_            0
```

Figure 44

2. Open the main.c file, find the location as shown in the following figure and setup the relevant parameters for "Data Transmission" and "MCU Sleep mode".

- Change Connect Interval
  - #define _CHG_CONNECT_INTV_ 1: Enable “Change Connection Interval”
  - #define _CHG_CONNECT_INTV_ 0: Disable “Change Connection Interval”
  - #define _CNNT_INTV_MIN_VALUE_ (30): The minimum connect interval is seup to 30ms
  - #define _CNNT_INTV_MAX_VALUE_ (30): The maximum connect interval is setup to be 30ms

```
64  #define _CHG_CONNECT_INTV_          1
65  /*------ Min Interval(ms)<10-40000> ------------------------------*/
66  #define     _CNNT_INTV_MIN_VALUE_       (30)
67  /*------ Max Interval(ms)<10-40000> ------------------------------*/
68  #define     _CNNT_INTV_MAX_VALUE_       (30)
```

Figure 45

- Configure MCU sleep mode
  - #define _MCU_SLEEP_ENABLE_ 1: Enable “MCU sleep mode”
  - #define _MCU_SLEEP_ENABLE_ 0: Disable “MCU sleep mode”
  - #define _SLEEP_DELAY_TIMER_ (500/2): If there is no data transmission with the connected device (smartphone) for a time of 250ms, the MCU will enter Sleep mode.
  - #define _MASTER_WUW_VALUE_ (1)

    #define _MASTER_WUT_VALUE_ (4)
- As shown in Figure 46, when the MCU needs to wake up from the Sleep mode, the BM7701-00-1 will be setup to send a byte with data 0x00 to wake-up the MCU and then wait for 4ms to transmit the received data Event to the MCU. The wake-up signal width and delay time is determined by the MCU sleep mode setting and wake-up stabilisation time, the user can adjust it according to the actual situation.

```
87  /* MCU sleep mode Enable/disable    */
88  #define _MCU_SLEEP_ENABLE_          1
89  /* Enter deep sleep delay timer(ms)<10-5000>    */
90  #define     _SLEEP_DELAY_TIMER_     (500/2)
91  /* Wake up Signal Width(byte)<1-8>  */
92  #define     _MASTER_WUW_VALUE_          (1)
93  /* Wake up Signal delay time(ms)<0-20>  */
94  #define     _MASTER_WUT_VALUE_          (4)
```

Figure 46

Conclusion

This application note has introduced the commonly used API function commands for the BM7701-00-1 module. It has guided users on how to use the HT8 and HT32 MCUs to configure the BM7701-00-1 module Bluetooth parameters and transmit data with the smartphone App. The module operating method and the bidirectional data transmission ability have also been demonstrated in this example, helping users to quickly use in their related electronic products, such as household appliances, medical products, etc.

Reference File

Reference files: BM7701-00-1 Datasheet, BLE_API and BLE_Service.

For more information, consult the Holtek official website: www.holtek.com.

Revision and Modification Information

Date	Author	Issue	Modification Information
2022.06.23	阮義展	V1.00	First Version

Disclaimer

All information, trademarks, logos, graphics, videos, audio clips, links and other items appearing on this website ('Information') are for reference only and is subject to change at any time without prior notice and at the discretion of Holtek Semiconductor Inc. and its related companies (hereinafter 'Holtek', 'the company', 'us', 'we' or 'our'). Whilst Holtek endeavors to ensure the accuracy of the Information on this website, no express or implied warranty is given by Holtek to the accuracy of the Information. Holtek shall bear no responsibility for any incorrectness or leakage.

Holtek shall not be liable for any damages (including but not limited to computer virus, system problems or data loss) whatsoever arising in using or in connection with the use of this website by any party. There may be links in this area, which allow you to visit the websites of other companies. These websites are not controlled by Holtek. Holtek will bear no responsibility and no guarantee to whatsoever Information displayed at such sites. Hyperlinks to other websites are at your own risk.

Limitation of Liability

In no event shall Holtek Limited be liable to any other party for any loss or damage whatsoever or howsoever caused directly or indirectly in connection with your access to or use of this website, the content thereon or any goods, materials or services.

Governing Law

The Disclaimer contained in the website shall be governed by and interpreted in accordance with the laws of the Republic of China. Users will submit to the non-exclusive jurisdiction of the Republic of China courts.

Update of Disclaimer

Holtek reserves the right to update the Disclaimer at any time with or without prior notice, all changes are effective immediately upon posting to the website