BM67C741-1 Application Note

D/N: AN0624EN

Introduction

The BM67C741-1 is a Bluetooth Low Energy transceiver module based on Holtek's HT32F67741 device for BLE transparent transmission. While the HT32F67741 device is designed based on a BLE SoC and a 32-bit MCU, they can communicate using a UART interface. 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 the BM67C741-1 to configure Bluetooth parameters and transmit data using a smartphone. 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 to develop their Bluetooth electronic products.

Functional Description

Communication Interface

The BM67C741-1 module is based on a BLE SoC and a 32-bit MCU which can communicate using the UART interface. The UART interface is a universal serial data bus for asynchronous communication which can implement full-duplex transmission and reception. The communication serial port connection method between the BLE device and the internal MCU is shown as follows. Note that the communication pins are unbonded on the module.

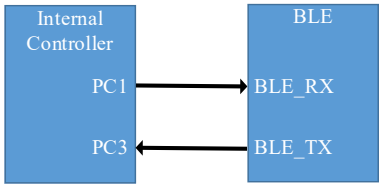

Figure 1. UART Connection Diagram

When the nRST_B pin is pulled from low to high or the BLE experiences a power-on reset, the BLE_TRX or TEST_0/1 pin will remain in the I/O mode for an initial 60ms. At this point the BLE device will select a UART Baud Rate according to the I/O condition. After this, the BLE_TRX or TEST_0/1 pin state will change back from the I/O mode to the UART TX/RX mode. This is shown in Table 1. If a different baud rate is required, users should send commands to the module with an initial baud rate after the device has been powered-on.

The TX/RX pin remains in an I/O mode after the BLE device experiences a power-on reset for 60ms	BLE_TX=1 (TEST_1=1)	BLE_TX=0 (TEST_1=0)
BLE_RX=1 (TEST_0=1)	Baud Rate = 115200 (default)	Baud Rate = 9600

Table 1. UART Baud Rate Selection

UART Format

The UART data format should meet the following requirements:

1. Baud rate: 1953bps~115200bps (BLE)
2. Data bit: 8 bits
3. Parity check: No
4. Stop bit: 1 bit

Figure 2. UART Sequence Diagram

Command/Event Format

The BLE supports three types of commands, including CmdType1, CmdType2 and CmdType3. CmdType1 are HCI defined in the Bluetooth specification. CmdType2 supports access to the BLE Services. CmdType3 supports setting up the BLE internal parameters, including Tx Power, crystal load, advertising enable/disable, advertising interval, connection interval, etc. When the internal MCU sends a command to the BLE using the UART interface, the BLE will respond with an Event when the BLE 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	Depend on Type
Responder → Initiator	0x78		EvtStatus: [3:0]: ErrorCode [7:4]: 1: CmdFlag_supported 2: CmdFlag_notifyIndicate		

Table 2. 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	Depend on Type
Responder → Initiator	0x78		EvtStatus: b[3:0]: ErrorCode b[7:4]: N/A		

Table 3. CmdType2 Format

Example	
Internal MCU→77 04 00 0B00 08→BLE Internal MCU←78 03 00 0B00←BLE	Set RF Tx power =0x08 (CmdType3)
Internal MCU→77 04 00 0700 01→BLE Internal MCU←78 03 00 0700←BLE	Start advertising (CmdType3)
Internal MCU→77 08 20 F1FF 01 02 03 04 05→BLE Internal MCU←78 03 20 F1FF←BLE	Send 0x0102030405 to phone (CmdType2)
Internal MCU←78 08 00 F2FF 01 02 03 04 05←BLE	Receive 0x0102030405 from phone (CmdType2)

Table 4. 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 to quickly grasp the BM67C741-1 module basic operation and system architecture using the application examples.

Environment Preparation

Hardware: Bluetooth Module BM67C741-1, Patch Board BCT-67C741-001, Development Board BCE-GENTRX0-001.

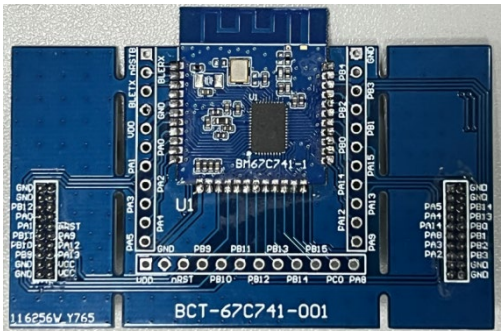

Figure 3. Module + Patch Board

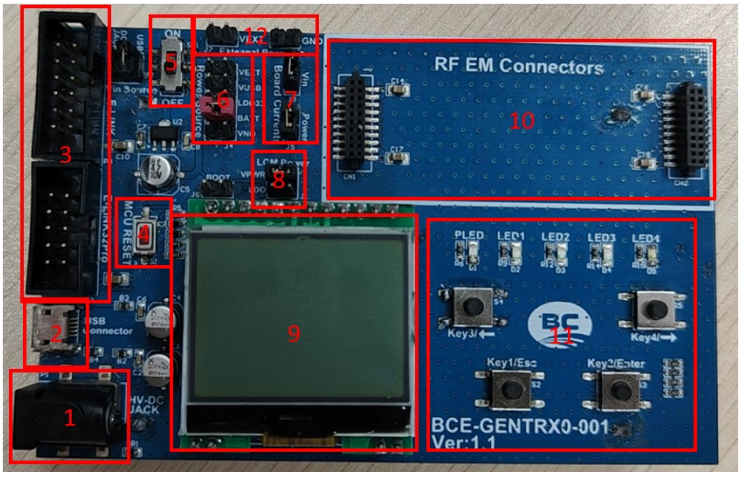

Figure 4. Development Board

1. DC power socket, the voltage here must not exceed 12V.

2. Micro USB interface. This can be used as a system power input. Here the power supply jumper should be placed on the LDO33 position.
3. JTAG interface, which can be used with IDE interface to emulate and downlown programs.
4. System reset key.
5. Main power switch, OFF when in the down position and ON when in the up position.
6. System power selection. If the jumper is on the LDO33 position, this means that power is sourced from the USB interface. If the jumper is on the BATT position, this means that the power supply is sourced from the battery holder. This is located on the back of the board and uses two 1.5V AA batteries. If the jumper is located on the 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.
7. Development board and BM67C741-1 module board current detection point.
8. LCM power selection.
9. The lipuid crystal display supports 128×64 pixels. Refer to the example program in this application note for usage information and also for the display control library.
10. The RF module interface is the connection point for the RF transmitter/receiver. In this example, the BCT-67C741-001+ BM67C741-1 is used.
11. 5 LEDs and 4 keys. This example is used to display the power-on status, data receiving status and transmitted data.
12. External power supply contacts.

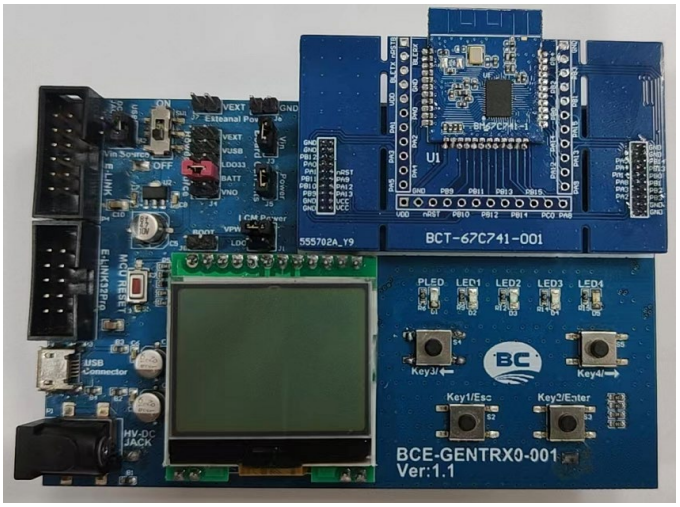

Figure 5. Complete Assembly

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

Figure 6. Android Link – Left and iOS Link – Right

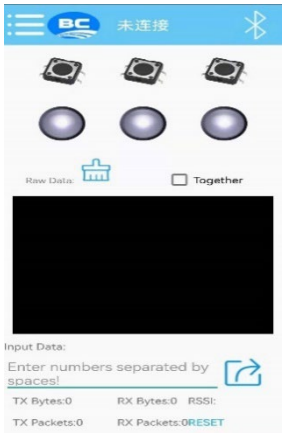

Figure 7. UI after Installation

Operating Description

1. Use the elink-32 Pro to program the application example to the BM67C741-1.
2. Turn on the the development board power switch. 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 upper right of smartphone App to scan for Bluetooth devices and select the name “BM67C741-1” to connect to. When the Bluetooth device has connected successfully, the connection icon in the upper right of the App will change to 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 KEY1 icon is pressed: The LED1 is on
   - When KEY2 icon is pressed: The LED2 is on
   - When KEY3 icon is pressed: The LED3 is on

Hardware Description

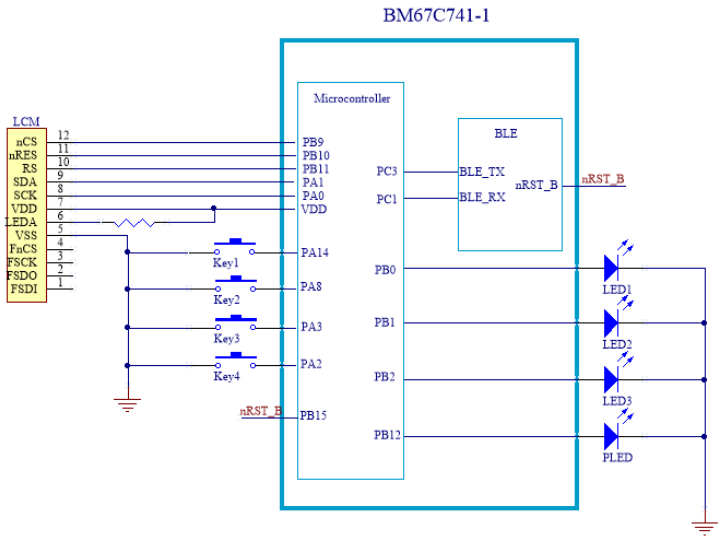

Figure 8. Application Circuit

Software Description

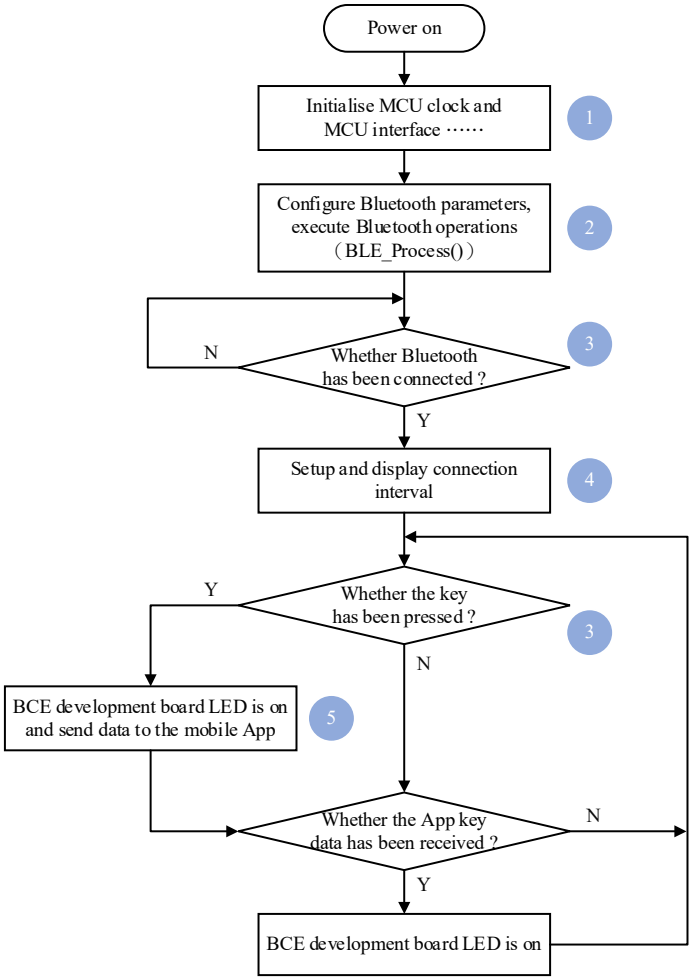

Figure 9. Software Flowchart

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

① Initialise Update Function

- BCI_InterfaceConfigure(_UART_BAUDRATE_): Configure the I/O port states between the internal MCU and the BLE
- BCI _RESET_CLR(): The BLE nRST_B pin is polled low
- BCI _RESET_SET(): The BLE nRST_B pin is polled high
- BCI _HardwareBaudRateDefault(_UART_BAUDRATE_): Initialise the BLE baud rate using the communication pin state
- BCI _HardwareBaudRateRelease(): Complete the BLE baud rate initialisation

② Setup Bluetooth Parameters

- BCI _SendBCIReadPackage(BCI_VERSION, 0x80): Check version
- BCI _SendBCIReadPackage(BCI_DEV_ADDRESS, 0x00): Obtain address
- BCI_SetBaudRate(), BCI_SetTxPower()….: Setup the BLE baud rate and Tx power etc.
- BCI _SetupFeatureFlag(FEATURE_SET, FEATURE_STATUS_EVENT): Setup the BLE to send data when the Bluetooth communication state has changed
- BCI _AdvertisingControl(ENABLE): Setup the BLE to enable advertising

③ Enter the Deep Sleep Mode

- BCI_SetOperateMode(OP_DEEPSLEEP,ENABLE,_MASTER_WUW_VALUE_,_MASTER_WUT_VALUE_): Setup the BLE to enter the Deep Sleep mode if there is no operation for a long time and setup the external wake-up signal for the internal MCU

④ Setup Connnection Interval

- BCI _DummyWakeup(): Wake-up the BLE from the Sleep mode
- BCI _ConnectIntervalModify(u16 opcode, u16 min, u16 max): Setup the BLE connnection interval

⑤ Send Data

- BCI _DummyWakeup(): Wake-up the BLE from the Sleep mode
- BCI _SendBCIPackage(NotifyFFF1, BCI_ServiceProperties, len, pbuf): Setup the BLE to send wireless data to the smartphone App

Module API Function Introduction

The following table shows the BM67C741-1 API command functions and related functions. Refer to the BLE.c file for the application example program.

API Prototype	Functional Description
void BCI_InterfaceConfigure(u8 br)	Configure the I/O port states between the internal MCU and BLE
void BCI _UARTConfigure(u8 br)	Configure the UART data format of the internal MCU communication pin
void BCI _RESET_SET(void)	nRST_B pin is pulled high
void BCI _RESET_CLR(void)	nRST_B pin is pulled low
void BCI _HardwareBaudRateDefault(u8 br)	Initialise the BLE baud rate using the communication pin state
void BCI _HardwareBaudRateRelease(void)	Complete the BLE baud rate initialisation
void BCI _HardwareReset(void)	BLE hardware reset, Initialise Bluetooth parameters
void * BCI _SoftwareReset(void)	Setup the “Software Reset” command to initialise the Bluetooth parameters. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _SoftwareResetKeep(void)	Setup the “Software Reset” command to retain the RAM Bluetooth parameters. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _SendHCIPackage(u16 opcode,u8 len,u8 *pbuf)	Setup the HCI command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _SendBCIPackage(u16 opcode,u8 flag,u8 len,u8 *pbuf)	Setup the BCI command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _SetDeviceName(u8 leng,u8 *name)	Setup the “BLE Bluetooth device name” command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _SetDeviceAddress(u8 *adr,u8 type)	Setup the “BLE Bluetooth address” command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _ConnectIntervalModify(u16 opcode,u16 min,u16 max)	Setup the “BLE connection interval” command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _SetAdvertisingData(u8 mode,u8 leng,u8 *advdata)	Setup the “BLE advertising data” command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _SetScanResponseData(u8 leng,u8 *sdata)	Setup the “BLE scan response data” command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _AdvertisingInterval(u16 min,u16 max,u8 chmap)	Setup the “BLE advertising interval” command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _AdvertisingControl(ControlStatus ctrl)	Setup the “BLE advertising enable/disable” command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _SetTxPower(u8 pwr)	Setup the “BLE Tx power” command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _SetCrystalCload(u8 cc)	Setup the “BM67C741-1 16MHz external crystal load capacitor value, C_{LOAD}” command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _SetupFeatureFlag(u8 md,FeatureFlag sff)	Setup the “BLE Feature” command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _SetOperateMode(u8 omd,ControlStatus ctrl,u8 wuw,u8 wut)	Setup the “BLE Sleep mode and the internal MCU external wake-up signal” command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _SetWhiteList(ControlStatus erase,u8 *adr,u8 *mask)	Setup the “BLE whitelist” command. This will need to be transmitted by the BCI _TransmitPackage( )
void * BCI _SetBaudRate(u8 br)	Setup the “BLE baud rate” command. This will need to be transmitted by the BCI _TransmitPackage( )
bool BCI _DummyWakeup(void)	Wake-up the BLE from the Sleep mode

API Prototype	Functional Description
bool BCI _TransmitPackage(void *pbuf)	Store the command packet into the send queue
bool BCI _ReadTransmitEmpty(void)	Obtain the data send queue status
bool BCI _ReadReceiveEmpty(void)	Obtain the data receive queue status
void * BCI _ReadReceivePackage(void)	Read the BLE received data
void BCI _WriteReceivePackage(void)	Write the received data index to the receive queue
void BCI _ReceiveParserPackage(void)	Parse the BLE received data

Table 5. Module API Command Functions

API Name	void BCI_InterfaceConfigure(u8 br)
API Function	Setup the I/O port states between the internal MCU and BLE
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 setup the I/O port states between the internal MCU and the BLE. Setup the internal MCU nRST_B control pin to be in the AFIO mode. This function can call the BCI_UARTConfigure(u8 br) to set the internal MCU communication pin to be in the UART data format.

API Name	void BCI_UARTConfigure(u8 br)
API Function	Setup the internal MCU communication pin to be in the UART data format
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 set the internal MCU communication pin to be in the UART data format

API Name	void BCI_RESET_SET(void)
API Function	The nRST_B pin is pulled high
Input Parameter	—
Output Parameter	—
Program Description	In the example program, this API function is executed to pull the BLE nRST_B pin high

API Name	void BCI_RESET_CLR(void)
API Function	The nRST_B pin is pulled low
Input Parameter	—
Output Parameter	—
Program Description	In the example program, this API function is executed to pull the BLE nRST_B pin low

API Name	void BCI_HardwareBaudRateDefault(u8 br)
API Function	Initialise the BLE 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 BLE 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 BCI_HardwareBaudRateRelease() must be called to release the communication pins to the UART mode

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

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

API Name	void *BCI_SoftwareReset(void)
API Function	Setup the "Software Reset" command to initialise the Bluetooth parameters. This will need to be transmitted by the BCI _TransmitPackage( )
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, a BLE software reset will occur and the Bluetooth parameters stored in RAM will be cleared. After the BLE responds with an Event, it needs to wait 60ms before the next communication. This command is equivalent to BCI_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 *BCI_SoftwareResetKeep(void)
API Function	Setup the "Software Reset" command to retain the RAM Bluetooth parameters. This will need to be transmitted by the BCI _TransmitPackage( )
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 BLE software reset will occur however the Bluetooth parameters in the RAM will be retained. After the BLE 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 *BCI_SendHCIPackage(u16 opcode,u8 len,u8 *pbuf)
API Function	Setup the HCI command. This will need to be transmitted by the BCI _TransmitPackage( )
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 *BCI_SendBCIPackage(u16 opcode,u8 flag,u8 len,u8 *pbuf)
API Function	Setup the "BCI - BLE Controller Interface" command. This will need to be transmitted by the BCI _TransmitPackage( )
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 *BCI_SetDeviceName(u8 leng,u8 *name)
API Function	Setup the “BLE Bluetooth device name” command. This will need to be transmitted by the BCI _TransmitPackage( )
Input Parameter 1	Data length: ≤31-bytes
Input Parameter 2	Buffer for storing data
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the “BLE Bluetooth 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 *BCI_SetDeviceAddress(u8 *adr,u8 type)
API Function	Setup the “BLE Bluetooth address” command. This will need to be transmitted by the BCI _TransmitPackage( )
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 “BLE Bluetooth 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 *BCI_ConnectIntervalModify(u16 opcode,u16 min,u16 max)
API Function	Setup the “BLE connection interval” command. This will need to be transmitted by the BCI _TransmitPackage( )
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 “BLE connection interval” command. Input Parameter 1: 1. When the Input Parameter 1 is equal to BCI_CONN_INTV, only Input Parameter 2 is valid which 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 *BCI_SetAdvertisingData(u8 mode,u8 leng,u8 *advdata)
API Function	Setup the "BLE advertising data" command. This will need to be transmitted by the BCI _TransmitPackage( )
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 "BLE 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 *BCI_SetScanResponseData(u8 leng,u8 *sdata)
API Function	Setup the "BLE scan response data" command. This will need to be transmitted by the BCI _TransmitPackage( )
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 "BLE 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 *BCI_AdvertisingInterval(u16 min,u16 max,u8 chmap)
API Function	Setup the "BLE advertising interval" command. This will need to be transmitted by the BCI _TransmitPackage( )
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 "BLE 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 *BCI_AdvertisingControl(ControlStatus ctrl)
API Function	Setup the “BLE advertising enable/disable” command. This will need to be transmitted by the BCI _TransmitPackage( )
Input Parameter	ENABLE/DISABLE
Output Parameter	Command packet pointer
Program Description	In the example program, this API function is executed to setup the “BLE 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 *BCI_SetTxPower(u8 pwr)			
API Function	Setup the “BLE Tx power” command. This will need to be transmitted by the BCI _TransmitPackage( )			
Input Parameter	0~15			
Output Parameter	Command packet pointer			
Program Description	In the example program, this API function is executed to setup the “BLE 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)	-29.5	-9.7	0.6	3.7

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

API Name	void *BCI_SetCrystalCload(u8 cc)			
API Function	Setup the “BM67C741-1 16MHz crystal load capacitor value, C_{Load}” command. This will need to be transmitted by the BCI _TransmitPackage( )			
Input Parameter	0~15			
Output Parameter	Command packet pointer			
Program Description	In the example program, this API function is executed to setup the “BLE 16MHz crystal load capacitor value, C_{Load}” command. It is recommended that the BLE 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)	15.99971	16.00009	16.00076	16.00272

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

API Name	void *BCI_SetupFeatureFlag(u8 md,FeatureFlag sff)
API Function	Setup the “BLE Feature” command. This will need to be transmitted by the BCI _TransmitPackage( )
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 “BLE 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 BLE 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 BLE Flash memory will be deleted and the default values will be restored. 4. When the Input Parameter 2 is equal to FEATURE_STATUS_EVENT, the BLE 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 BLE 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 *BCI_SetOperateMode(u8 omd,ControlStatus ctrl,u8 wuw,u8 wut)
API Function	Setup the “BLE Sleep mode and internal MCU external wake-up signal” command. This will need to be transmitted by the BCI _TransmitPackage( )
Input Parameter 1	Mode: OP_NORMAL / OP_DEEPSLEEP / OP_POWERDOWN
Input Parameter 2	Internal MCU Sleep mode: ENABLE / DISABLE
Input Parameter 3	Internal MCU external wake-up signal width: 0~8 byte
Input Parameter 4	Internal 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 “BLE Sleep mode and internal 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 BLE enters Normal mode. The BLE operates in the Normal mode by default. 2. When the Input Parameter 1 is equal to OP_DEEPSLEEP, the BLE enters the Deep Sleep mode. The BLE can be woken up to the Normal mode using the API function, BCI_DummyWakeup(void). 3. When the Input Parameter 1 is equal to OP_POWERDOWN, BLE enters the Power Down mode. The BLE can be woken up to the Normal mode using the API function, BCI_DummyWakeup(void). Input Parameter 2: 1. When the Input Parameter 2 is equal to ENABLE, the internal MCU will enter the Sleep mode. The BLE will be setup to send a wake-up signal to wake up the internal MCU. 2. When the Input Parameter 2 is equal to DISABLE, the internal MCU will not enter the Sleep mode. It is not necessary to setup the BLE 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 *BCI_SetWhiteList(ControlStatus erase,u8 *adr,u8 *mask)
API Function	Setup the “BLE whitelist addresses” command. This will need to be transmitted by the BCI _TransmitPackage( )
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 “BLE whitelist” command. Ex: Whitelist address=0x112233445566, address mask=0xFFFFFFFFFF00. Only the Bluetooth addresses ranging from 0x112233445500 to 0x1122334455FF can be connected. If the address mask are all written to be 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 *BCI_SetBaudRate(u8 br)
API Function	Setup the “BLE baud rate” command. This will need to be transmitted by the BCI _TransmitPackage( )
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 “BLE baud rate” command. After this command has been executed and the BLE has responded with Event, it needs to wait about 1ms before the internal 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 BCI_DummyWakeup(void)
API Function	Wake-up the BLE 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 BLE from the Sleep mode.

API Name	bool BCI_TransmitPackage(void *pbuf)
API 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 is full
Program Description	In the example program, when this API function is executed, the internal MCU places the packets into the send queue. The packets in the send queue will be transmitted to the BLE in sequence.

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

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

API Name	void BCI_WriteReceivePackage(void)
API 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 into the receive queue.

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

Bluetooth Parameter Modification

1. Open the bleprocess.h file and click “Configuration Wizard” to enter the “Bluetooth Parameters” setting page, as shown in Figure 10. Each parameter option will be described in detail in the next section.

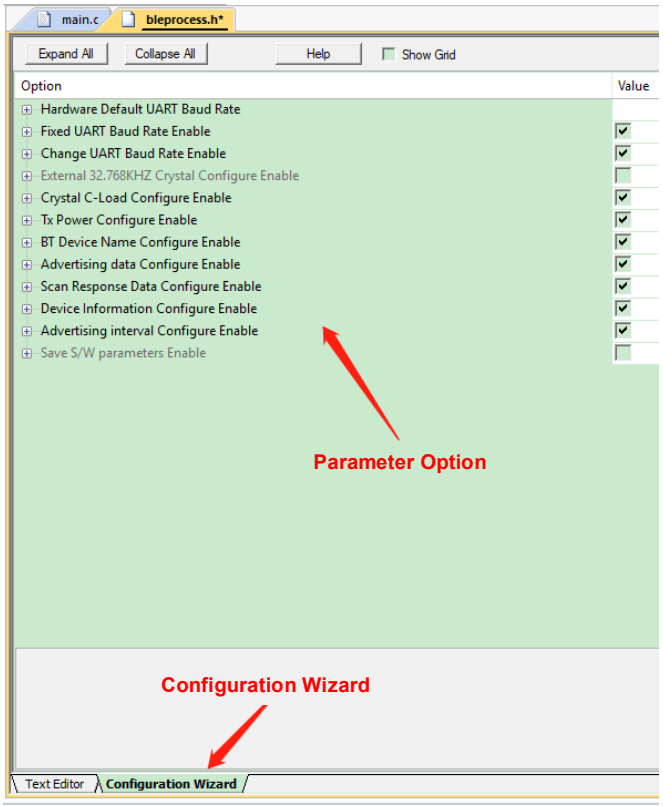

Figure 10

- Hardware Default UART Baud Rate: Setup the BLE default baud rate after a power-on reset of 60ms. Only two baud rates, 9600bps or 115200bps can be selected, as shown in Figure 11. This is the hardware method which is used to modify the BLE baud rate. Refer to Table 2.

Figure 11

- Fixed UART Baud Rate Enable: Click to enable "Setup the internal MCU default baud rate" and click the additional item "Baud Rate" to select a baud rate, as shown in Figure 12.
  - ➢ Disable: Automatically setup the internal MCU default baud rate equal to the BLE default baud rate - select 9600bps or 115200bps.
  - ➢ Enable: The user needs to manually select the internal MCU default baud rate.

  Ex: The BLE 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 BLE default baud rate will be equal to 57600bps after each power-on reset of 60ms. The user should enable this option and select the internal MCU default baud rate to be equal to 57600bps, otherwise the internal MCU and the BLE will not be able to communicate.

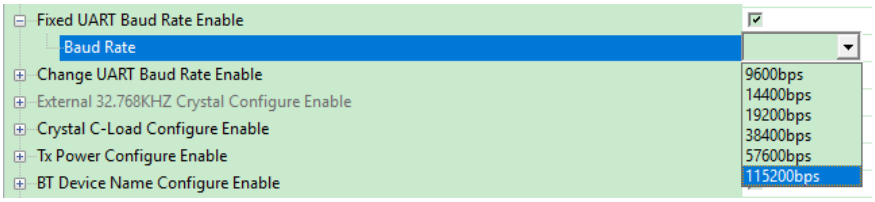

Figure 12

- 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 13. This option can be used to setup the BLE and internal MCU baud rate at the same time. It will only be executed after the default baud rate of the internal MCU and BLE has been setup and are the same.

  Ex: The internal MCU first sends a "Change UART Baud Rate" command to the BLE and then the internal MCU will setup the communication pin baud rate to be the same as that of the BLE.

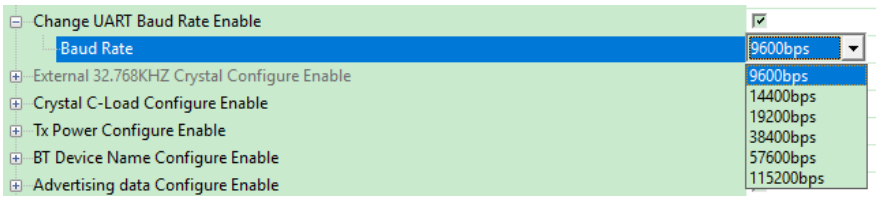

Figure 13

- External 32.768kHz Crystal Configure Enable: Click to enable "External 32.768kHz Crystal", as shown in Figure 14. Some modules include an external 32.768kHz crystal oscillator which can be enabled. It should be disabled by default in the application example program. Note that if the BM67C741-1 module does not have an external 32.768kHz crystal oscillator, enabling this option will cause the operation to fail.

Figure 14

- Crystal C-Load Configure Enable: Click to enable "Change external 16MHz crystal oscillator C_{Load} value" and click the additional item "Crystal C-Load" to enter a value which ranges from 0 to 15, as shown in Figure 15. A value of 4 is recommended for the BM67C741-1 module.

Figure 15

Input Parameter	0	5	10	15
Crystal Frequency(MHz)	15.99971	16.00009	16.00076	16.00272

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

Table 6

- 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 16. Users can adjust this value according to the transmission distance or power consumption.

Figure 16

Input Parameter	0	5	10	15
Tx Power(dbm)	-29.5	-9.7	0.6	3.7

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

Table 7

- BT Device Name Configure Enable: Click to enable "Change BT Device Name", as shown in Figure 17. The Bluetooth name after successful change is "BM67C741-1". Users can also customise the Bluetooth name on the bleprocess.c page. Here, find the red box position shown in Figure 18 which can then be changed, the maximum length is 31-bytes

Figure 17

```
24 #if (_BDNAME_CFG_ENABLE_ == 1)
25 /* set BT device name */
26 uc8 BLE_DeviceName[] = {'B','M','6','7','C','7','4','1','-','1'};
27 #endif
28
```

Figure 18

- Advertising data Configure Enable: Click to enable “Change advertising data”, as shown in Figure 19. Users can also customise the advertising data on the bleprocess.c page. Here, find the red box position shown in Figure 20, the maximum length is 31-bytes.

Figure 19

```
#if (_ADV_DATA_CFG_ENABLE_ == 1)
/* set Advertising Data */
uc8 BLE_AdvData[] =
{
  /* flag */
  2,0x01,0x06,
  /* Manufacturer Specific Data */
  11,0xFF,0xFF,0xFF,
  'B','e','s','t','C','o','m','m'
};
#endif
```

Figure 20

- Scan Response Data Configure Enable: Click to enable “Chane Scan Response Data”, as shown in Figure 21. Users can also customise the scan response data on the bleprocess.c page. Here, find the red box position shown in Figure 22, the maximum length is 31-bytes.

Figure 21

```
#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 22

- Advertising interval Configure Enable: Click to enable “Change advertising interval”, click the additional item to enter a value, as shown in Figure 23. It should be noted that Min Interval should be less than or equal to Max Interval.

Figure 23

- Device Information Configure Enable: Click to enable “Change Device Information”, as shown in Figure 24. Users can also customise device information on the bleprocess.c page. Here, find the red box position in Figure 25 which can be changed.

Figure 24

```
#if (_DEV_INF_CFG_ENABLE_ == 1)
uc8 BLE_DevInfSysID[] = { 0x37,0x37,0x30,0x31 };
uc8 BLE_DevInfModelNumber[] = {'B','M','6','7','C','7','4','1','-','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 25

- Save S/W parameters Enable: Click to enable "Store parameters". The Bluetooth parameters in "Parameter Setting 3~11" will be stored to the BLE Flash memory as shown in Figure 26. The premise is that the corresponding setting options are enabled. The BLE Bluetooth parameters setup using commands 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 BLE will automatically use the Bluetooth parameters in the Flash memory after a power-on reset occurs.

Figure 26

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 BLE Connection Interval", as shown in Figure 27. Click the additional item to enter a connection interval value.

Figure 27

- MCU sleep mode Enable / disable: Click to enable "MCU sleep mode". Click the additional item to enter a time. As shown in Figure 28, when this option is enabled, if the BLE does not transmit data with a connected device (smartphone) within 500ms, the internal MCU will actively enter the Sleep mode.

Figure 28

- 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 29, when the internal MCU needs to wake up from the Sleep mode, the BLE can be setup to send 2 bytes, composed of 0x00, to wake-up the internal MCU. It will then wait for 8ms to transmit the received data Event to the internal MCU. The wake-up signal width and delay time is determined by the internal MCU sleep mode setting and wake-up stabilisation time. The user can adjust it according to the actual situation.

Figure 29

- In order to clarify 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 internal MCU and the BLE.

  Example 1: Select a baud rate of 9600bps or 115200bps. The operating steps are shown in Table 8.

  - The baud rate is 9600bps or 115200bps after each power-on reset of 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 8

  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 BLE Flash memory. The operation steps are shown in Table 9.

  - The baud rate is 9600bps or 115200bps after each power-on reset of 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 9

  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 BLE Flash memory. This should execute the programming operation twice. The operating steps are shown in Table 10.

  - 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 BLE Flash memory.
    - Then, the BLE baud rate is always 57600bps after each power-on reset. When the internal MCU baud rate is 9600bps or 115200bps, the internal MCU and BLE will not be able to communicate.
    - It is required to execute a second programming.
  - Second programming
    - The internal MCU and BLE can communicate normally (click to "Fixed UART Baud Rate Enable" to change the internal 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 10

Conclusion

This application note has introduced the commonly used API function commands for the BM67C741-1 module. It has guided users on how to use the internal MCU to configure the BLE 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 them in their related electronic products, such as household appliances, medical products, etc.

Reference File

Reference files: BM67C741-1 Datasheet, BLE_API, BLE_Service.

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

Revision and Modification Information

Data	Author	Issue	Modification Information
2022.07.01	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.