

MG32x02z

BLE Library User Guide

Version: 1.12

List of Contents

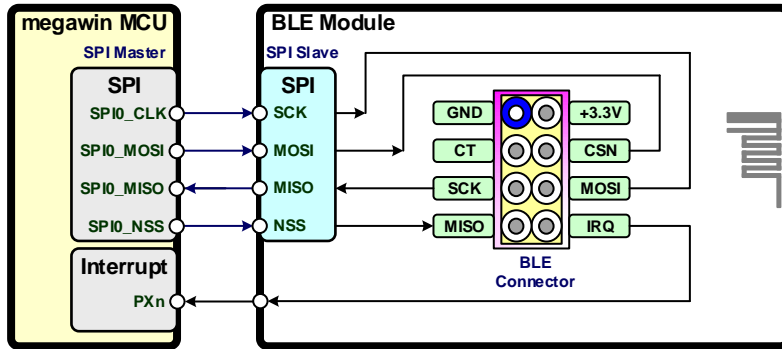
List of Contents	2
1. Hardware Connection	4
1.1. BLE to MCU SPI Connection	4
1.2. BLE Module Picture	4
1.3. Real BLE to MCU Connection	4
2. BLE_SPI.c	6
2.1. Overview	6
2.2. Function List	6
2.3. Prototype Description	7
2.3.1. SPI0_Init	7
2.3.2. SPI_WriteBuf	7
2.3.3. SPI_ReadBuf	7
3. BSP.c	8
3.1. Overview	8
3.2. Function List	8
3.3. Prototype Description	9
3.3.1. BLE_EXIC_Init	9
3.3.2. EXINT1_IRQHandler	9
3.3.3. BLE_URT0_Init	9
3.3.4. IWDT_Init	9
3.3.5. BLE_SPI0_Init	10
3.3.6. BLE_LED_Init	10
3.3.7. LED_Flash	10
3.3.8. BSP_Init	10
3.3.9. IsIrqEnabled	10
3.3.10. IrqMcuGotoSleepAndWakeup	11
3.3.11. GetSysTickCount	11
3.3.12. SysTick_Handler	11
4. retarget.c	12
4.1. Overview	12
4.2. Function List	12
4.3. Prototype Description	13
4.3.1. URT_IRQHandler	13
4.3.2. moduleOutData	13
4.3.3. CheckComPortInData	13
4.3.4. UsrProcCallback	13
5. BLE MG126 Library	14
5.1. Overview	14
5.2. List of interface functions	14
5.3. Description of the interface function	15
5.3.1. radio_initBle	15
5.3.2. radio_initBle_TO	15
5.3.3. ble_run_interrupt_start	15
5.3.4. SetBleIntRunningMode	15

5.3.5. SetLePinCode.....	15
5.3.6. ble_set_adv_type.....	16
5.3.7. ble_set_adv_data.....	16
5.3.8. ble_set_adv_rsp_data.....	16
5.3.9. ble_set_name	16
5.3.10. ble_set_interval.....	16
5.3.11. radio_standby	17
5.3.12. radio_resume	17
5.3.13. ble_set_wakeupdly	17
5.3.14. ble_set_adv_enableFlag.....	17
5.3.15. ble_set_role	17
5.3.16. ble_disconnect.....	17
5.3.17. ble_disc.....	18
5.3.18. ble_set_feature_supported	18
5.3.19. ble_set_md_enable.....	18
5.3.20. get_ble_version.....	18
5.3.21. GetFirmwareInfo	18
5.3.22. ser_write_rsp_pkt.....	18
5.3.23. att_notFd.....	19
5.3.24. att_ErrorFd_eCode	19
5.3.25. att_server_rdyByGrTypeRspDeviceInfo.....	20
5.3.26. att_server_rdyByGrTypeRspPrimaryService	20
5.3.27. att_server_rd.....	20
5.3.28. sconn_notifydata.....	20
5.3.29. sconn_indicationdata	21
5.3.30. SIG_ConnParaUpdateReq.....	21
5.3.31. sconn_GetConnInterval	21
5.3.32. GetRssiData.....	21
5.3.33. radio_setBleAddr	21
5.3.34. SetFixAdvChannel	22
5.3.35. test_carrier.....	22
5.3.36. test_SRRCCarrier	22
5.3.37. test_PRBS9	22
5.3.38. test_RX.....	22
5.4. Breaking Service Program Mode.....	23
5.4.1. Main program.....	23
5.4.2. SysTick IRQ handle	23
5.4.3. For BLE IRQ handler.....	24
6. Revision History	25

1. Hardware Connection

1.1. BLE to MCU SPI Connection

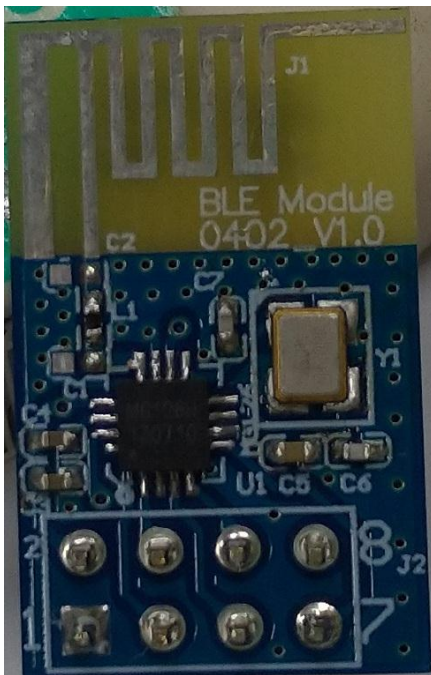
The following diagram is showing the BLE to MCU SPI connection.



<Note> : 1. PXn ~ GPIO pin, PX={PA, PB, PC, PD}, n= pin index

1.2. BLE Module Picture

The following picture is showing BLE module.

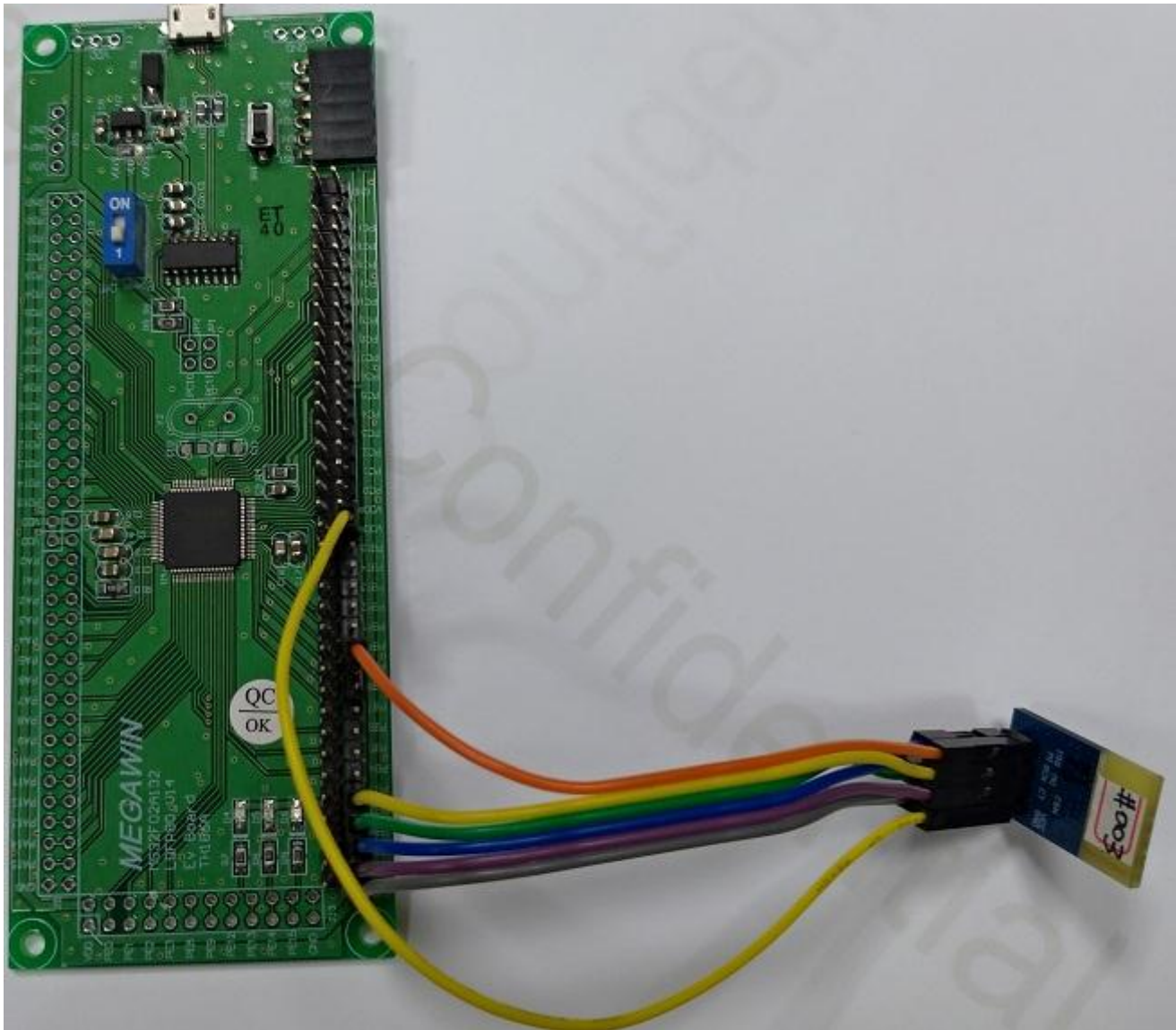


Note:

1. BLE module operation voltage is 1.9 ~3.6V.
2. CT pin is test pin (not connect).

1.3. Real BLE to MCU Connection

The following picture is showing real BLE to MCU connection.



2. BLE_SPI.c

2.1. Overview

This article is a description of the function of the BLE_SPI.c protocol, which is suitable for BLE application development. SPI initializes SPI read and write.

2.2. Function List

The application interface functions are defined in the file BLE_SPI.c, and the test interface functions are defined in BLE_SPI.c, which mainly include the following functions:

```
void SPI0_Init(void)
unsigned char SPI_WriteBuf(unsigned char reg, unsigned char const *pBuf, unsigned char len)
unsigned char SPI_ReadBuf(unsigned char reg, unsigned char *pBuf, unsigned char len)
```

2.3. Prototype Description

2.3.1. SPI0_Init

Function prototype: void SPI0_Init (void)

	Description
Brief	SPI0 / URTx(SPI mode) module initial
Parameter 1	None
Return	None

2.3.2. SPI_WriteBuf

Function prototype: unsigned char SPI_WriteBuf (unsigned char reg, unsigned char const *pBuf, unsigned char len)

	Description
Brief	SPI0 / URTx data transmit to BLE
Parameter1	reg, BLE command
Parameter 2	pBuf, Write buffer index
Parameter 3	len, data length 0 ~ 255 bytes
Return	None

2.3.3. SPI_ReadBuf

Function prototype: unsigned char SPI_ReadBuf (unsigned char reg, unsigned char *pBuf, unsigned char len)

	Description
Brief	SPI0 / URTx data receive
Parameter 1	reg, BLE command
Parameter 2	pBuf, Read buffer index
Parameter 3	len, data length 0 ~ 255 bytes
Return	None

3. BSP.c

3.1. Overview

This article is a description of the function of the BSP.c protocol, which is suitable for BLE application development. This file including the module EXIC, URT0, SPI/URTx, IWDt used to initialize BLE, and provides SysTick and EXINT1 interrupt services.

3.2. Function List

The application interface functions are defined in the file BSP.c, and the test interface functions are defined in BSP.c, which mainly include the following functions:

```
void BLE_EXIC_Init(void)
void EXINT1_IRQHandler(void)
void BLE_URT0_Init(void)
void IWDt_Init(void)
void BLE_SPI0_Init(void)
void BLE_LED_Init(void)
void LED_Flash(void)
void BSP_Init(void)
char IsIrqEnabled(void)
void IrqMcuGotoSleepAndWakeup(void)
unsigned int GetSysTickCount(void)
void SysTick_Handler(void)
```


3.3. Prototype Description

3.3.1. BLE_EXIC_Init

Function prototype: void BLE_EXIC_Init (void)

	Description
Brief	EXIC module initial for BLE
Parameter 1	None
Return	None

3.3.2. EXINT1_IRQHandler

Function prototype: void EXINT1_IRQHandler (void)

	Description
Brief	EXINT1 IRQ Handler
Parameter 1	None
Return	None

3.3.3. BLE_URT0_Init

Function prototype: void BLE_URT0_Init (void)

	Description
Brief	UART0 module initial for BLE
Parameter 1	None
Return	None

3.3.4. IWDT_Init

Function prototype: void IWDT_Init (void)

	Description
Brief	IWDT module initial
Parameter 1	None
Return	None

3.3.5. BLE_SPI0_Init

Function prototype: void BLE_SPI0_Init (void)

	Description
Brief	SPI0 / URTx module and used pins initial for BLE
Parameter 1	None
Return	None

Note: PB0 AFS GPB0 (Software NSS), PB1 AFS SPI0_MISO, PB2 AFS SPI0_CLK and PB3 AFS SPI0_MOSI.

3.3.6. BLE_LED_Init

Function prototype: void BLE_LED_Init (void)

	Description
Brief	LED module initial for BLE
Parameter 1	None
Return	None

3.3.7. LED_Flash

Function prototype: void LED_Flash (void)

	Description
Brief	LED flash
Parameter 1	None
Return	None

3.3.8. BSP_Init

Function prototype: void BSP_Init(void)

	Description
Brief	BSP initial
Parameter 1	None
Return	None

3.3.9. IsIrqEnabled

Function prototype: char IsIrqEnabled (void)

	Description
Brief	Is EXIT1 trigger
Parameter 1	None
Return	None

3.3.10. IrqMcuGotoSleepAndWakeup

Function prototype: void IrqMcuGotoSleepAndWakeup (void)

	Description
Brief	IRQ MCU go to sleep and wait trigger wakeup
Parameter 1	None
Return	None

3.3.11. GetSysTickCount

Function prototype: unsigned int GetSysTickCount(void)

	Description
Brief	Get sys tick count
Parameter 1	None
Return	None

3.3.12. SysTick_Handler

Function prototype: void SysTick_Handler(void)

	Description
Brief	SysTick Handler
Parameter 1	None
Return	None

4. retarget.c

4.1. Overview

This article is a description of the function of the retarget.c protocol, which is suitable for BLE application development. Contains URT0 interrupt service program, and provides BLE Data Buffer URT0 data mutual transfer, regular IWDG count refresh, LED flashing...

4.2. Function List

The application interface functions are defined in the file retarget.c, and the test interface functions are defined in retarget.c, which mainly include the following functions:

```
void URT_IRQHandler (void)
void moduleOutData (u8 *data, u8 len)
void CheckComPortInData (void)
void UsrProcCallback (void)
```

4.3. Prototype Description

4.3.1. URT_IRQHandler

Function prototype: void URT0_IRQHandler (void)

	Description
Brief	URT0 IRQ TX/RX service
Parameter 1	None
Return	None

4.3.2. moduleOutData

Function prototype: void moduleOutData(u8*data, u8 len)

	Description
Brief	Move data to TX buffer
Parameter 1	data, Data start index
Parameter 2	Len, Move data length
Return	None

4.3.3. CheckComPortInData

Function prototype: void CheckComPortInData(void)

	Description
Brief	Check con port input data
Parameter 1	None
Return	None

4.3.4. UsrProcCallback

Function prototype: void UsrProcCallback(void)

	Description
Brief	Refresh IWDT, switch LED flash, check com port input data and URT0 transfer data.
Parameter 1	None
Return	None

5. BLE MG126 Library

5.1. Overview

This article is a description of the MG BLE protocol stack interface function and is suitable for BLE application development of MG chips.

5.2. List of interface functions

The application interface function definition is in `mg_api.h`, and the test interface function definition is `mg_test_api.h`, which mainly includes the following functions:

- 1) `void radio_initBle(unsigned char txpwr, unsigned char**addr);`
- 2) `unsigned char radio_initBle_TO(unsigned char txpwr, unsigned char**addr, unsigned short ms_timeout);`
- 3) `void ble_run_interrupt_start(unsigned short adv_interval);`
- 4) `void SetBleIntRunningMode(void);`
- 5) `void SetLePinCode(unsigned char *PinCode/*6 0~9 digitals*/);`
- 6) `void ble_set_adv_type(unsigned char type);`
- 7) `void ble_set_adv_data(unsigned char* adv, unsigned char len);`
- 8) `void ble_set_adv_rsp_data(unsigned char* rsp, unsigned char len);`
- 9) `void ble_set_name(unsigned char* name, unsigned char len);`
- 10) `void ble_set_interval(unsigned short interval);`
- 11) `void radio_standby(void);`
- 12) `void radio_resume(void);`
- 13) `unsigned char ble_set_wakeupdly(unsigned short counter);`
- 14) `void ble_set_adv_enableFlag(char sEnableFlag);`
- 15) `unsigned char ble_set_role(unsigned char role_new, unsigned short scan_window);`
- 16) `void ble_disconnect(void);`
- 17) `unsigned char ble_disc(unsigned char reason);`
- 18) `void ble_set_feature_supported(unsigned int feature_local);`
- 19) `void ble_set_md_enable(unsigned char enable_flag);`
- 20) `unsigned char *get_ble_version(void);`
- 21) `unsigned char *GetFirmwareInfo(void);`
- 22) `void ser_write_rsp_pkt(unsigned char pdu_type);`
- 23) `void att_notFd(unsigned char pdu_type, unsigned char attOpcode, unsigned short attHd);`
- 24) `void att_ErrorFd_ecode(unsigned char pdu_type, unsigned char attOpcode, unsigned short attHd, unsigned char errCode);`
- 25) `void att_server_rdByGrTypeRspDeviceInfo(unsigned char pdu_type);`
- 26) `void att_server_rdByGrTypeRspPrimaryService(unsigned char pdu_type, unsigned short start_hd, unsigned short end_hd, unsigned char*uuid, unsigned char uuidlen);`
- 27) `void att_server_rd(unsigned char pdu_type, unsigned char attOpcode, unsigned short att_hd, unsigned char* attValue, unsigned char datalen);`
- 28) `unsigned char sconn_notifydata(unsigned char* data, unsigned char len);`
- 29) `unsigned char sconn_indicationdata(unsigned char* data, unsigned char len);`
- 30) `void SIG_ConnParaUpdateReq(unsigned short IntervalMin, unsigned short IntervalMax, unsigned short SlaveLatency, unsigned short TimeoutMultiplier);`
- 31) `unsigned short sconn_GetConnInterval(void);`
- 32) `Unsigned char GetRssiData(void);`
- 33) `void radio_setBleAddr(u8 addr[6]);`
- 34) `void SetFixAdvChannel(unsigned char isFixCh37Flag);`
- 35) `void test_carrier(unsigned char freq, unsigned char txpwr);`
- 36) `void test_SRRCCarrier(unsigned char freq, unsigned char txpwr);`
- 37) `void test_PRBS9(unsigned char freq, unsigned char txpwr);`
- 38) `void test_RX(unsigned char freq);`

5.3. Description of the interface function

5.3.1. radio_initBle

Function prototype: `radio_initBle (unsigned char txpwr, unsigned char**addr/*Output*/);`

Function brief: Initialize bluetooth chip and Bluetooth protocol stack

Input parameters:

txpwr: This parameter is used to initialize bluetooth chip transmit power, with desirable values such as TXPWR_0DBM, TXPWR_3DBM, and so on.

Output parameter:

addr: This parameter returns Bluetooth MAC address information, 6 bytes long.

Return value: None

Note: Call after SetBleIntRunningMode.

5.3.2. radio_initBle_TO

Function prototype: `unsigned char radio_initBle_TO (unsigned char txpwr, unsigned char**addr, unsigned short ms_timeout);`

Function brief: Initialize bluetooth chip and Bluetooth protocol stack

Input parameters:

txpwr: This parameter is used to initialize bluetooth chip transmit power, with desirable values such as TXPWR_0DBM, TXPWR_3DBM, , and so on.

ms_timeout: This parameter is used for bluetooth chip initialization timeout in milliseconds, with a recommended value of 10 to 50.

Output parameter:

addr: This parameter returns Bluetooth MAC address information, 6 bytes long.

Return value: 0 indicates that initialization failed, and non-0 indicates that initialization was successful.

Note: Call after SetBleIntRunningMode.

5.3.3. ble_run_interrupt_start

Function prototype: `void ble_run_interrupt_start (unsigned short adv_interval);`

Function brief: Run the Bluetooth protocol as an interrupt service program

Input parameters:

adv_interval: in units of 0.625ms, such as 160 for advertising intervals of 100ms.

Output parameters: None

Return value: None

Note: Refer to the programming guide "**Breaking Service Program Mode**".

5.3.4. SetBleIntRunningMode

Function prototype: `void SetBleIntRunningMode (void);`

Function brief: Set how to interrupt the service program

Input parameters: None

Output parameters: None

Return value: None

Note: Refer to the programming guide "**Breaking Service Program Mode**".

5.3.5. SetLePinCode

Function prototype: `void SetLePinCode (unsigned char* PinCode /*6 0 ~ 9 digitals*/);`

Function: Set the Bluetooth pairing PIN

Input parameters:

PinCode: 6 numbers from 0 to 9.

Output parameters: None

Return value: None

Note: Call before ble_run_interrupt_start is called. The default PIN is 000000, which can be called if the PIN

needs to be changed.

5.3.6. ble_set_adv_type

Function prototype: ble_set_adv_type (unsigned char type);

Function brief: Set up BLE advertising Adv PDU Header

Input parameters:

type: 0-adv_ind, 2-adv_nonconn_ind, default 0x80.

Output parameters: None

Return value: None

Note: This function can be called at any time, advertising data content immediately effective.

5.3.7. ble_set_adv_data

Function prototype: ble_set_adv_data (unsigned char*adv, unsigned char len);

Function brief: Set the BLE advertising data

Input parameters:

adv: advertising data pointer.

len: advertising data length.

Output parameters: None

Return value: None

Note: This function can be called at any time, advertising data content immediately effective.

5.3.8. ble_set_adv_rsp_data

Function prototype: ble_set_adv_rsp_data (unsigned char*rsp, unsigned char len);

Function brief: Set the BLE advertising scan answer data

Input parameters:

rsp: advertising scan answer data pointer.

len: advertising scan answer data length.

Output parameters: None

Return value: None

Note:1) This function can be called at any time, advertising data content immediately effective.

2)Calling this function causes the function ble_set_name() to fail, but the function getDeviceInfoData() in the app .c is still valid.

5.3.9. ble_set_name

Function prototype: ble_set_name (unsigned char*name, unsigned char len);

Function brief: Set the name contents of the BLE advertising answer package

Input parameters:

name: name data pointer.

len: name data length.

Output parameters: None

Return value: None

Note:1) Thisfunction can be called at any time, the data content immediately effective.

2)Calling this function will only change the data content of the advertising scan answer, which needs to be modified synchronously in order to be consistent with the GATT counterpart

The .c deviceInfo in the app, refer to .c implementation of the function updateDeviceInfoData() in the app.

5.3.10. ble_set_interval

Function prototype: void ble_set_interval (un shortsinged interval);

Function brief: Set the BLE advertising interval

Input parameters:

interval: advertising interval, range:0x0020 to 0x4000, in 0.625ms.

Output parameters: None

Return value: None

Note: This function can be called at any time, the data content immediately effective.

5.3.11. radio_standby

Function prototype: void radio_standby (void);

Function brief: Bluetooth chip enters standby, in the most power-saving state

Input parameters: None

Output parameters: None

Return value: None

Note: This function is typically called before the MCU enters standby, or it can also be entered in `UsrProcCallback` or `ble /b30> Sleep` is called after.

The system consumes approximately 2uA when it enters standby.

5.3.12. radio_resume

Function prototype: void radio_resume (void);

Function brief: Bluetooth chip recovers from standby

Input parameters: None

Output parameters: None

Return value: None

Note: This function can be called when the radio needs to be restored after it enters standby.

5.3.13. ble_set_wakeupdly

Function prototype: unsigned char ble_set_wakeupdly (unsigned short counter);

Function brief: Set the mcu wake-up time. mcu uses low power consumption, irq interrupt wake-up has a delay, and this interface needs to be called to set wake-up delay

room. /b30> Mcu does not use low-power mode and does not require a wake-up time.

Input parameter:

counter: mcu wake-up delay time in 1/64ms. Range: 0x0000 to 0x0040.

Output parameters: None

Return value: 0 - fail; 1 - success

Note: This function is called `radio_initBle` the function.

5.3.14. ble_set_adv_enableFlag

Function prototype: ble_set_adv_enableFlag (char sEnableFlag);

Function brief: Set whether the BLE is advertising or not

Input parameters:

sEnableFlag: 1 -- Run advertising ; 0 -- stop advertising .

Output parameters: None

Return value: None

Note: This function can be called at any time, immediately effective.

5.3.15. ble_set_role

Function prototype: unsigned char ble_set_role (unsigned char role_new, unsigned short scan_window);

Function brief: Set the ble role to perm(0) or central(/b18> 1) . If it is central, set the scan window.

Input parameters:

role_new: 0 -- perity; 1 - central. _____ is perity

scan_window: a single- advertising channel scan window, in 0.625ms. Range: 0x0004 to 0x4000.

Output parameters: None

Return value: 0 - fail; 1 - success

Note: This function is called only in a non-connected state, and the default for the role is perity if it is not called. scan_window parameter is only at the central corner

is the length of the scan window for a single advertising channel. The scan starts with 37 advertising channels.

5.3.16. ble_disconnect

Function prototype: void ble_disconnect (void);

Function brief: Disconnect an existing connection

Input parameters: None

Output parameters: None

Return value: None

Note: This function can be called at any time, immediately effective.

5.3.17. ble_disc

Function prototype: unsigned char ble_disc (unsigned char reason);

Function brief: Disconnect an existing connection

Input parameters:

reason: reason for disconnection (please fill in the BLE protocol).

Output parameters: None

Return value: 0 - Failed;

Note: This function can be called at any time, immediately effective.

5.3.18. ble_set_feature_supported

Function prototype: void ble_set_feature_supported (unsigned int feature_local);

Function brief: Set the feature of the BLE(32bit).

Input parameters:

feature_local: THIS-supported LE feature.

Output parameters: None

Return value: None

Note: This function can be called at any time, in the connection establishment process.

5.3.19. ble_set_md_enable

Function prototype: void ble_set_md_enable (char sEnableFlag);

Function brief: Set whether the BLE receive supports more data

Input parameters:

sEnableFlag: 1-- more data is supported; more data.

Output parameters: None

Return value: None

Note: This function can be called at any time, after the connection effective. This function is typically used for OTAs to speed up OTAs. This letter is not called

The number default does not support more data.

5.3.20. get_ble_version

Function prototype: unsigned char *get_ble_version (void);

Function brief: Gets the Bluetooth protocol stack version information string

Input parameters: None

Output parameters: None

Return value: Bluetooth stack version information string

5.3.21. GetFirmwareInfo

Function prototype: unsigned char *GetFirmwareInfo (void);

Function brief: Gets the Bluetooth baseband version information string

Input parameters: None

Output parameters: None

Return value: Bluetooth baseband version information string

5.3.22. ser_write_rsp_pkt

Function prototype: void ser_write_rsp_pkt (unsigned char pdu_type);

Function brief: A response function after writing a value with a Write With Response property feature

Input parameters:

pdu_type: pdu type parameters, direct reference callback functions ser_write_rsp parameters in the table.

Output parameters: None

Return value: None

Note: For feature values that need to be written, failure to answer will result in a disconnection of the connection.

5.3.23. att_notFd

Function prototype: att_notFd (unsigned char pdu_type, unsigned char attOpcode, unsigned short attHd);

Function brief: An answer function that operates on an invalid feature value (or a feature value that is not defined).

Input parameters:

pdu_type: pdu type parameters. Refer directly to the corresponding parametersser_write_rsp, att_server_rdyByGrType, server_rd_rsp callback function.

attOpcode: action type parameters. Refer directly to the corresponding parametersser_write_rsp, att_server_rdyByGrType, server_rd_rsp callback function.

attHd: corresponds to the characteristic value handle value. Refer directly to the corresponding parametersser_write_rsp, att_server_rdyByGrType, server_rd_rsp callback function.

Output parameters: None

Return value: None

Note: Any operation that does not have an invalid feature value requires an answer to this function, which can be called as a default.

5.3.24. att_ErrorFd_eCode

Function prototype: att_ErrorFd_eCode (unsigned char pdu_type, unsigned char attOpcode, unsigned short attHd, unsigned char errorCode);

Function brief: An answer function for an invalid operation

Input parameters:

pdu_type: pdu type parameters. Refer directly to the corresponding parametersser_write_rsp, att_server_rdyByGrType, server_rd_rsp callback function.

attOpcode: action type parameters. Refer directly to the corresponding parametersser_write_rsp, att_server_rdyByGrType, server_rd_rsp callback function.

attHd: corresponds to the characteristic value handle value. Refer directly to the corresponding parametersser_write_rsp, att_server_rdyByGrType, server_rd_rsp callback function.

ErrorCode error , refer to ATT Error Code

Output parameters: None

Return value: None

ATT Error Code define:

#define ATT_ERR_INVALID_HANDLE 0x01 ◦

#define ATT_ERR_READ_NOT_PERMITTED 0x02 ◦

#define ATT_ERR_WRITE_NOT_PERMITTED 0x03 ◦

#define ATT_ERR_INVALID_PDU 0x04 ◦

#define ATT_ERR_INSUFFICIENT_AUTHEN 0x05 ◦

#define ATT_ERR_UNSUPPORTED_REQ 0x06 ◦

#define ATT_ERR_INVALID_OFFSET 0x07 ◦

#define ATT_ERR_INSUFFICIENT_AUTHOR 0x08 ◦

#define ATT_ERR_PREPARE_QUEUE_FULL 0x09 ◦

#define ATT_ERR_ATTR_NOT_FOUND 0x0a ◦

#define ATT_ERR_ATTR_NOT_LONG 0x0b ◦

#define ATT_ERR_INSUFFICIENT_KEY_SIZE 0x0c ◦

#define ATT_ERR_INVALID_VALUE_SIZE 0x0d ◦

#define ATT_ERR_UNLIKELY 0x0e ◦

```
#define ATT_ERR_INSUFFICIENT_ENCRYPT 0x0f ◦  
#define ATT_ERR_UNSUPPORTED_GRP_TYPE 0x10 ◦  
#define ATT_ERR_INSUFFICIENT_RESOURCES 0x11 ◦
```

5.3.25. att_server_rdyByGrTypeRspDeviceInfo

Function prototype: void att_server_rdyByGrTypeRspDeviceInfo (unsigned char pdu_type);

Function features: Answers to the default Device Info content can call this interface function

Input parameters: pdu type parameters, direct reference callback functions att_server_rdyByGrType parameters in the table

Output parameters: None

Return value: None

Note: This interface function can be called directly if the user is using the release package code directly.

5.3.26. att_server_rdyByGrTypeRspPrimaryService

Function prototype: void att_server_rdyByGrTypeRspPrimaryService (unsigned char pdu_type, unsigned short start_hd, unsigned short end_hd, unsigned char* uuid, unsigned char uuidlen);

Function brief: Answers a query for Primary Service, and the user populates the corresponding data with handles and UUIDs that are actually defined by the characteristic value

Input parameters:

pdu_type: pdu type parameters, refer directly to the callback function att_server_rdyByGrType parameters in the table.

start_hd: a service corresponds to the starting handle value.

end_hd: the end handle value for a service.

uuid: a UUID string (Hex value) for a service, such as 0x180A /b18> Represented as 0x0a,0x18.

uuidlen: the length of a service corresponding to the UUID string.

Output parameters: None

Return value: None

Note: The corresponding parameters need to be populated strictly in accordance with the feature value definition.

5.3.27. att_server_rd

Function prototype: void att_server_rd (unsigned char pdu_type, unsigned char attOpcode, unsigned short att_hd, unsigned char* attValue, unsigned char datalen);

Function brief: Reads the value of a feature value.

Input parameters:

pdu_type: pdu type parameters and refer directly to the callback function server_rd_rsp parameters in the table.

attOpcode: The value corresponding to the operation refers directly to the callback function server_rd_rsp parameters in the operation.

att_hd: the handle value corresponding to the feature value, refers directly to the callback function server_rd_rsp the corresponding argument in

attValue: The value string pointer corresponding to the feature value.

datalen: The length of the feature value string.

Output parameters: None

Return value: None

Note: The corresponding feature value content needs to be treated as an answer content on demand, and can be answered if the corresponding feature value content is att_notFd().

5.3.28. sconn_notifydata

Function prototype: unsigned char sconn_notifydata (unsigned char* data, unsigned char len);

Function brief: Send data via Bluetooth

Input parameters:

data: needs to send a data pointer

len: the data length.

Output parameters: None

Return value: The number of bytes of data that were successfully sent

Note: This interface function automatically unpacks and sends data based on system caching conditions, but must not block loop calls. You need to acknowledge before calling cur_notifyhandle is correct.

5.3.29. sconn_indicationdata

Function prototype: unsigned char sconn_indicationdata (unsigned char* data, unsigned char len);

Function brief: Send data via Bluetooth

Input parameters:

data: needs to send a data pointer.

len: the data length.

Output parameters: None

Return value: The number of bytes of data that were successfully sent

Note: This interface function automatically unpacks and sends data based on system caching conditions, but must not block loop calls. An acknowledgement is required before the call can be made cur_notifyhandle is correct.

5.3.30. SIG_ConnParaUpdateReq

Function prototype: void SIG_ConnParaUpdateReq (un shortsinged IntervalMin, unsigned short IntervalMax, unsigned short SlaveLatency, unsigned short TimeoutMultiplier);

Function brief: In the connected state, try to use the emission interval within the user's expected range. /b30> The final launch interval is determined by the central device.

Input parameters:

IntervalMin: Minimum emission interval, in 1.25ms (Ranges 6 to 3200).

IntervalMax: Maximum emission interval, 1.25ms (Ranges 6 to 3200).

SlaveLatency: Response delay, ranges from 0 to 500, and less than (TheDionTimeout / (IntervalMax)2)) -1.

TimeoutMultiplier: Connection break time. /b30> Unit 10ms (Range 10 to 3200).

Output parameters: None

Return value: None

Note: SlaveLatency is recommended for a range of 0-5.

5.3.31. sconn_GetConnInterval

Function prototype: unsigned short sconn_GetConnInterval (void);

Function brief: Get the current bluetooth connection using the transmit interval value, in 1.25ms.

Input parameters: None

Output parameters: None

Return value: The interval between the current Bluetooth connections.

5.3.32. GetRssiData

Function prototype: unsigned char GetRssiData (void);

Function brief: Call this function when receiving advertising packages or packets, and you can get the raw data for receiving signal strength. Depending on the number of RSSI referenced value, using this data to compare the strength of the received signal.

Input parameters: None

Output parameters: None

Return value: Receives a relative value of signal strength, such as 1dB 191 is 190 large.

5.3.33. radio_setBleAddr

Function prototype: void radio_setBleAddr (unsigned char addr[6]);

Function brief: Set the customer's own Bluetooth device address.

Input parameters:

addr: Bluetooth device address, 6 bytes.

Output parameters: None

Return value: None

Note: Called after the protocol stack is initialized.

5.3.34. SetFixAdvChannel

Function prototype: void SetFixAdvChannel (unsigned isFixCh37Flag);

Function brief: During the debugging phase, the protocol stack is fixed to the 37-channel advertising , and the protocol stack defaults to advertising on all channels in order to facilitate air grabs.

Input parameters:

isFixCh37Flag: Set whether to advertising on channel 37 only.

Output parameters: None

Return value: None

Note: For debugging only, SetFixAdvChannel (1) is called after the protocol stack is initialized when debugging is required.

5.3.35. test_carrier

Function prototype: test_carrier (unsigned char freq, unsigned char txpwr);

Function brief: Test the actual carrier transmit power, the protocol stack will be fixed at the 2400-freq frequency point to send carrier, the ideal transmit power reference txpwr.

Input parameters:

freq: Carrier center frequency is (2400-freq) MHz.

txpwr: 0x43 corresponds to the ideal RF transmit power of 0dBm.

Output parameters: None

Return value: None

Note: For carrier and crystal frequency bias testing. After the stack is initialized, call test_carrier (80, 0x43), and then while (1) waits in a loop.

5.3.36. test_SRRCCarrier

Function prototype: test_SRRCCarrier (unsigned char freq, unsigned char txpwr);

Function brief: For SRRRC test - carrier.

Input parameters:

freq: Carrier center frequency is (2400-freq) MHz.

txpwr: 0x43 corresponds to the ideal RF transmit power of 0dBm.

Output parameters: None

Return value: None

Note: For SRRRC fixed frequency carrier emission tests. Called after the stack is initialized, , then while(1) waits in a loop.

5.3.37. test_PRBS9

Function prototype: test_PRBS9 (unsigned char freq, unsigned char txpwr);

Function brief: For SRRRC test - PRBS modulation signal emission

Input parameters:

freq: Carrier center frequency is (2400-freq) MHz

txpwr: 0x43 corresponds to the ideal RF transmit power of 0dBm

Output parameters: None

Return value: None

Note: For SRRRC modulation signal emission tests. Called after the stack is initialized, , then while(1) waits in a loop.

5.3.38. test_RX

Function prototype: void test_RX (unsigned char freq);

Function brief: Used for - to put the chip in a receiving state

Input parameters:

freq: Carrier center frequency is (2400-freq) MHz.

Output parameters: None

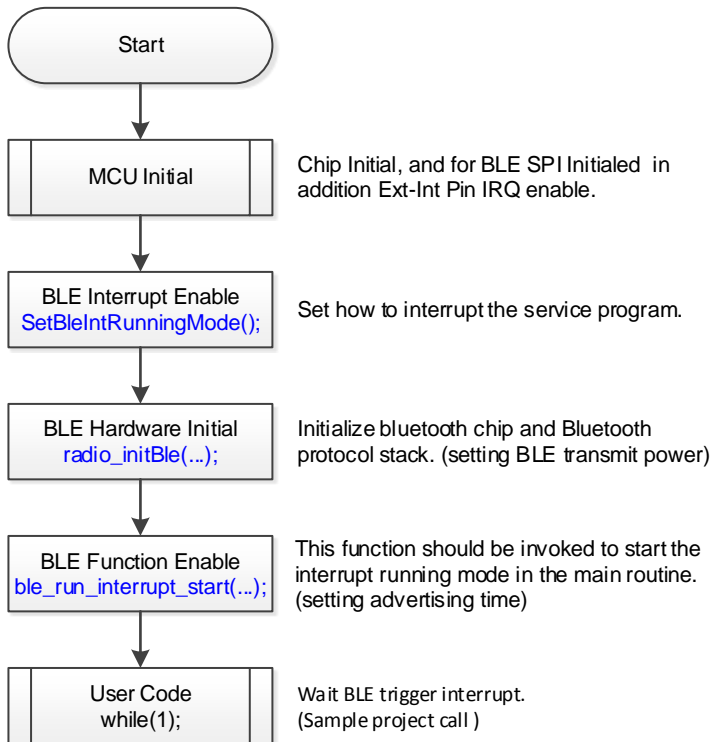
Return value: None

Note: Used to receive tests and do not care about data. Called after the stack is initialized, , then while(1) waits in a loop.

5.4. Breaking Service Program Mode

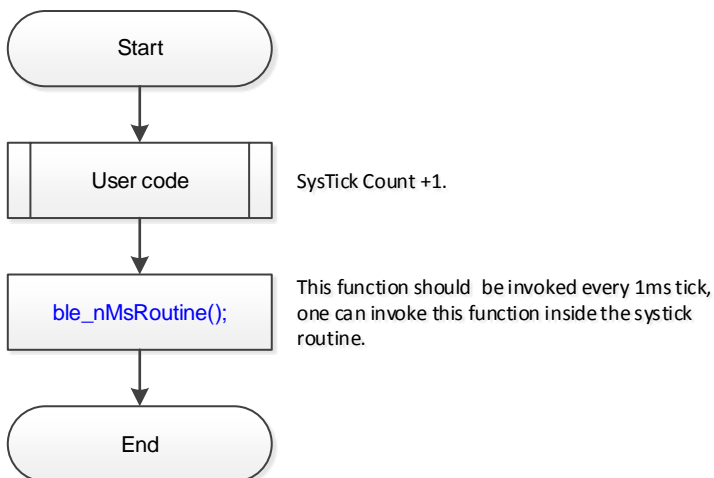
5.4.1. Main program

The following flowchart is showing about main code.



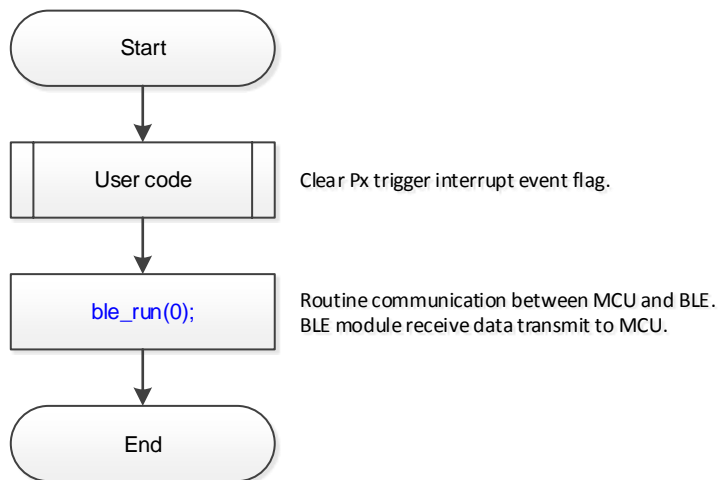
5.4.2. SysTick IRQ handle

The following flowchart is showing about SysTick_Handfer code description.



5.4.3. For BLE IRQ handler

The following flowchart is showing about EXINTx_IRQHandler code description.



6. Revision History

Revision V1.10 (2020_1211)		Chapter
1	English version	
2	Add "BLE to MCU SPI Connection" data.	1
3	Add "Breaking Service Program Mode" data.	5.4.
Revision V1.10 (2020_0731)		Chapter
1	Initial version	
Revision V1.11 (2020_0731)		Chapter
1	Change the name of the first chapter description file	1
2	Remove "BLE_CSC_Init" in 2.3.1 BSP.c, this is redundant code.	2.3.1
3	Update the overview content of BLE_SPI.c, BSP.c and retarget.c.	2.1 、 3.1 、 4.1
Revision V1.12 (2021_0713)		Chapter
1	Update List of Contents.	
2	Rename "BLE_SPI0_Library.lib" to "BLE_SPI.c" at "Overview" of "BLE_SPI.c" support	2.1
3	Rename "BLE_SPI0_Library.lib" to "BLE_SPI.c" at "Prototype Description" of "BLE_SPI.c".	2.2
4	Update "SPI0_Init" support SPI0 / URTx.	2.3.1
5	Update "SPI_WriteBuf" support SPI0 / URTx.	2.3.2
6	Update "SPI_ReadBuf" support SPI0 / URTx.	2.3.3
7	Update "BLE_SPI0_Init" initial SPI0 / URTx and used pins.	3.3.5