Program Guide

AT32F403+MG126 编程指南

Revision History:

Rev. No.	History	Issue Date	Remark
1.0	Initial issue	July 15, 2019	release

Important Notice:
MACROGIGA reserves the right to make changes to its products or to discontinue any integrated circuit product or service without notice. MACROGIGA integrated circuit products are not designed, intended, authorized, or warranted to be suitable for use in life-support applications, devices or systems or other critical applications. Use in such applications is done at the sole discretion of the customer. MACROGIGA will not warrant the use of its devices in such applications.

目录

1. 概述

本文是使用 AT32F403+MG126 芯片实现 BLE 应用的编程指导。

2. 软件架构

软件代码工程的目录和文件结构如下：

software\

M:.

```
├─Libraries
│  ├─AT32F4xx_StdPeriph_Driver
│  │  ├─inc
│  │  └─src
│  └─CMSIS
│      ├─CM4
│      ├─Documentation
│      ├─DSP_Lib
│      └─Lib
└─Project
    ├─AT32_Board
    └─AT_START_F403
        ├─Examples
        │  ├─ADC
        │  ├─BKP
        │  ├─BLE
        │  │  └─MDK_v5
        │  ├─CAN
        │  ├─CortexM4
        │  ├─CRC
        │  ├─DAC
```

```
│   ├─DMA
│   ├─EXTI
│   ├─FLASH
│   ├─FreeRTOS
│   ├─GPIO
│   ├─I2C
│   ├─I2S
│   ├─IWDG
│   ├─NVIC
│   ├─PWR
│   ├─RCC
│   ├─RTC
│   ├─SDIO
│   ├─SPI
│   ├─SRAM
│   ├─SYS_TICK
│   ├─TMR
│   ├─WWDG
│   └─XMC
└─Templates
```

其中 BLE 目录包含了 BLE 协议栈接口文件 mg_api.h 和 BLE 库文件。应用入口文件为 main.c，应用实现代码在 app.c。

3. 应用开发说明

BLE 应用开发对 mcu 的一般要求是：

1） 系统时钟至少 48MHz

2） 硬件 SPI，时钟至少 6MHz，不超过 10MHz

3） Flash 至少 16KB，SRAM 4KB，根据应用情况调整

4） 请注意配置好 StackSize，推荐配置 2KB

在 mcu SDK 的基础上，BLE 应用开发需要实现

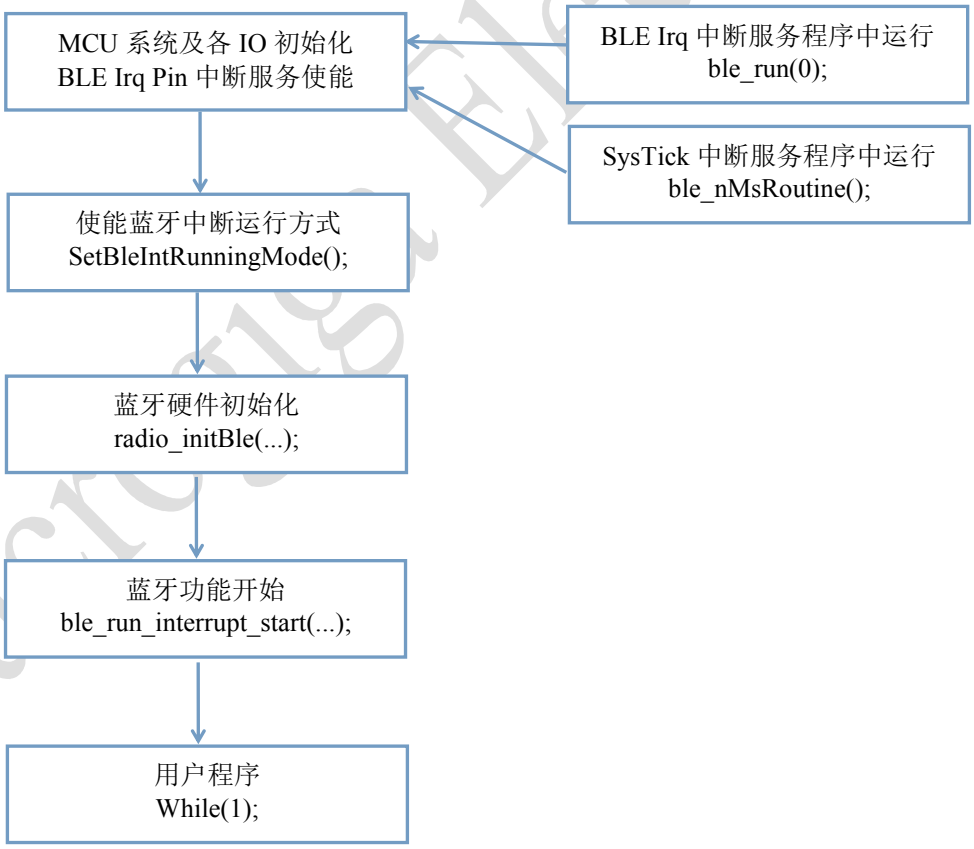
1） 与蓝牙协议相关的接口函数，具体见文件 app.c

2） 系统相关的函数：获取系统 tick 值（单位 1ms）：GetSysTickCount（）

3.1 中断服务程序方式运行模式

BLE 库以系统中断服务程序方式运行，适合于实现用户某功能需要占用较长 CPU 时间但可以被任意打断的应用场景。需要用到两个中断服务程序：一个是蓝牙 IRQ 中断 pin 对应的中断，另一个是实现 SysTick 对应的中断。IRQ 对应的中断服务程序用以运行蓝牙协议，优先级不高于 SysTick，但是针对其他 IO 中断来说需要有较高的优先级。

中断服务程序方式运行的软件架构如下图所示。



低功耗接口为 unsigned char ble_run_interrupt_McuCanSleep(void)

低功耗功能实现示例：

```
void IrqMcuGotoSleepAndWakeup(void)
{
    if(ble_run_interrupt_McuCanSleep())
    {
        //to do low power mode
    }
}
```

此函数可以在 main()函数的用户程序循环调用。

4. BLE 特征值及双向数据通信操作方法

用户如需调整 BLE 数据交互的特征值、服务及数据的收发，可按照如下的几个步骤进行调整（代码在文件 app_xxx.c 中）：

服务（service）及特征值定义

如图 1 所示，用户需要自行分配句柄（递增，不得重复），数据结构定义如下

```
typedef struct ble_character16{
    u16 type16;          //type2
    u16 handle_rec;      //handle
    u8  characterInfo[5];//property1 - handle2 - uuid2
    u8  uuid128_idx;     //0xff means uuid16,other is idx of uuid128
}BLE_CHAR;

typedef struct ble_UUID128{
    u8  uuid128[16];//uuid128 string: little endian
}BLE_UUID128;
```

图 1 特征值定义数据结构

图 1 中 type16 为 database 每个记录的类型，具体取值根据蓝牙规范定义。宏定义 TYPE_CHAR 为特征值（character），宏定义 TYPE_CFG 为 client configuration，宏定义 TYPE_INFO 为特征值描述信息；handle_rec 为对应记录的句柄；characterInfo 保存了对应特征

值的属性（property1）、句柄（handle2）及 uuid（uuid2），其中 handle2 及 uuid2 为 16 bit 小端格式；uuid128_idx 表示 uuid2 的格式，如该值为 UUID16_FORMAT 则表示 uuid2 为 16bit 格式，反之则表示 uuid2 为 128bit 的 uuid 信息对应的索引值，该索引值对应于 AttUuid128List 的内容索引。uuid128 为小端格式保存。


```
//
///STEP0:Character declare
//
const BLE_CHAR AttCharList[] = {
// ======  gatt =====  Do NOT Change!!
    {TYPE_CHAR,0x03,ATT_CHAR_PROP_RD, 0x04,0,0x00,0x2a,UUID16_FORMAT},//name
    //05-06 reserved
// ======  device info =====    Do NOT Change if using the default!!!
    {TYPE_CHAR,0x08,ATT_CHAR_PROP_RD, 0x09,0,0x29,0x2a,UUID16_FORMAT},//manufacture
    {TYPE_CHAR,0x0a,ATT_CHAR_PROP_RD, 0x0b,0,0x26,0x2a,UUID16_FORMAT},//firmware version
    {TYPE_CHAR,0x0e,ATT_CHAR_PROP_RD, 0x0f,0,0x28,0x2a,UUID16_FORMAT},//sw version
// ======  LED service or other services added here =====  User defined
    {TYPE_CHAR,0x11,ATT_CHAR_PROP_W|ATT_CHAR_PROP_RD|ATT_CHAR_PROP_NTF, 0x12,0, 0,0, 1/*uuid128-idx1*/ },//status
    {TYPE_CFG, 0x13,ATT_CHAR_PROP_RD|ATT_CHAR_PROP_W},//cfg
    {TYPE_CHAR,0x14,ATT_CHAR_PROP_W|ATT_CHAR_PROP_RD,                  0x15,0, 0,0, 2/*uuid128-idx2*/ },//cmd
    {TYPE_CHAR,0x17,ATT_CHAR_PROP_W|ATT_CHAR_PROP_RD,                  0x18,0, 0,0, 3/*uuid128-idx3*/ },//OTA
    {TYPE_INFO, 0x19,ATT_CHAR_PROP_RD},//description,"Mg OTA"
};

const BLE_UUID128 AttUuid128List[] = {
    {0x10,0x19,0x0d,0xc,0xb,0xa,9,8,7,6,5,4,3,2,1,0}, //idx0,little endian, service uuid
    {0x11,0x19,0x0d,0xc,0xb,0xa,9,8,7,6,5,4,3,2,1,0}, //idx1,little endian, character status uuid
    {0x12,0x19,0x0d,0xc,0xb,0xa,9,8,7,6,5,4,3,2,1,0}, //idx2,little endian, character cmd uuid
    {0x13,0x19,0x0d,0xc,0xb,0xa,9,8,7,6,5,4,3,2,1,0}, //idx3,little endian, character OTA uuid
};
```

图 2 服务器及特征值数据库定义示意图

图 2 给出了具体的定义示例，示例中包含三个 Service 配置，句柄分别对应的范围为 1-6，7-15 及 16-25，其中 7-15 为 Device Info Service，16-25 为用户自定义 Service。句柄 0x0004 为 Device Name 特征值，句柄 0x0009 为 Manufacture Info 特征值，句柄 0x000b 为 Firmware version 特征值，句柄 0x000f 为软件版本特征值。句柄 0x0012、0x0015 和 0x0018 为用户自定义特征值。

如果客户需要自行定义 Service，可按需更改接口如下函数的实现

```
void att_server_rdByGrType( u8 pdu_type,
                            u8 attOpcode,
                            u16 st_hd,
                            u16 end_hd,
                            u16 att_type );
```

图 3 给出了调用 Service 的例子。缺省情况下如果客户对 Device Info 不做特别修改，可直接调用缺省函数 att_server_rdByGrTypeRspDeviceInfo(pdu_type)即可。

自定义 service 的调用方式可参考图 3 中红色框内的方式，对应的调用接口为

```
void att_server_rdByGrTypeRspPrimaryService( u8 pdu_type,
                                             u16 start_hd,
                                             u16 end_hd,
                                             u8*uuid,
                                             u8 uuidlen);
```

其中 start_hd 与 end_hd 为对应 Service handle 取值范围，uuid 为字符串，对应的长度由 uuidlen 给出。

```
////////////////////////////////////////////////////////////////////////
///STEP1:Service declare
// read by type request handle, primary service declare implementation
void att_server_rdByGrType( u8 pdu_type, u8 attOpcode, u16 st_hd, u16 end_hd, u16 att_type )
{
 //!!!!!!!!  hard code for gap and gatt, make sure here is 100% matched with database:[AttCharList] !!!!!

    if((att_type == GATT_PRIMARY_SERVICE_UUID) && (st_hd == 1))//hard code for device info service
    {
        att_server_rdByGrTypeRspDeviceInfo(pdu_type);//using the default device info service
        //apply user defined (device info)service example
        //{
        //    u8 t[] = {0xa,0x18};
        //    att_server_rdByGrTypeRspPrimaryService(pdu_type,0x7,0xf,(u8*)(t),2);
        //}
        return;
    }

    //hard code
    else if((att_type == GATT_PRIMARY_SERVICE_UUID) && (st_hd <= 0x10)) //usr's service
    {
        att_server_rdByGrTypeRspPrimaryService(pdu_type,0x10,0x19,(u8*)(AttUuid128List[0].uuid128),16);
        return;
    }
    //other service added here if any
    //to do....

    ///error handle
    att_notFd( pdu_type, attOpcode, st_hd );
}
```

用户自定义Service代码示例

图 3 自定义 service 调用方式示意图

数据读写

BLE 的数据读写就是对应特征值（句柄）的读写操作，协议对应的接口函数为：

【读操作】

```
void server_rd_rsp( u8 attOpcode,
                    u16 att_hd,
                    u8 pdu_type);
```

其中 att_hd 为手机通过 BLE 希望读取特征值的句柄，应答数据通过如下接口反馈

```
void att_server_rd( u8 pdu_type,
                    u8 attOpcode,
                    u16 att_hd,
                    u8* attValue,
                    u8 datalen );
```

其中 attValue 为应答的数据指针，数据长度为 datalen。注意数据最长不得超过 20 字节。

【写操作】

```
void ser_write_rsp( u8 pdu_type,
                    u8 attOpcode,
                    u16 att_hd,
                    u8* attValue,
                    u8 valueLen_w);
```

其中 att_hd 为从手机 BLE 传（写）过来数据对应的特征值的句柄，数据内容保存在变量 attValue 中，数据长度为 valueLen_w。

【Notify 数据发送操作】

Notify 通过如下接口进行

```
u8 sconn_notifydata(u8* data, u8 len);
```

其中 data 为需要发送到手机的数据，长度由 len 指定。原则上数据长度可以超过 20 字节，协议会自动拆包发送，该函数返回实际发送的数据长度。

需要注意的是，Notify 接口没有指定对应的句柄，如果用户定义并使用多个 Notify 的特征值，需要在发送数据前通过调用如下接口指定对应的句柄

```
void set_notifyhandle(u16 hd);
```

或者直接修改变量 u16 cur_notifyhandle 即可。

回调函数 void gatt_user_send_notify_data_callback(void)可用于蓝牙模块端主动发送数据，代码实现方式推荐使用循环缓冲区的形式。

【DeviceName】

用户可以根据需要修改设备名称。缺省情况下，设备名称由宏 DeviceInfo 定义，BLE 协议栈内会通过接口 u8* getDeviceInfoData(u8* len)获取该信息，用户可根据实际情况重新实现该函数（app.c 文件内）。

【软件版本信息】

用户可根据需要修改软件的版本信息。缺省情况下，宏 SOFTWARE_INFO 定义了软件版本信息。

5. 注意事项

1、所有接口函数不得阻塞调用

2、函数 att_server_rd(...)每次调用发送的数据长度不得超过 20 字节

3、函数 sconn_notifydata(...)只能在协议主循环体内调用，函数不可重入，可以发送多于 20 字节的数据，协议会自动分包发送，每个分包长度最大为 20 字节。推荐一次发送的数据尽量不超过 3 个分包。

6. 硬件/系统通信/低功耗接口

配合蓝牙协议栈，需要系统应用实现 SPI 通信、定时器等接口。主要函数包括：

1、 unsigned int GetSysTickCount(void); 获取毫秒定时器累积值，用于计时等功能。

2、 Char IsIrqEnabled(void) ; 判断 IRQ 信号是否产生中断（低电平为中断有效）。

3、 unsigned char SPI_ReadBuf(unsigned char reg, unsigned char *pBuf, unsigned char len);

4、 unsigned char SPI_WriteBuf(unsigned char reg, unsigned char const *pBuf, unsigned char len);

另外，对于低功耗应用，蓝牙协议栈已经预留了接口供系统编程实现。主要函数包括：

void IrqMcuGotoSleepAndWakeup(void); mcu 进入低功耗的操作

7. 与蓝牙应用相关的函数接口

为便于蓝牙差异化功能的灵活实现，蓝牙协议内设置了若干接口并以回调函数的方式由应用层 porting 实现，所有回调函数不得阻塞调用，具体函数的实现可参考 SDK 发布包中对应代码的实现示例。回调函数主要包括

如下：

1、void gatt_user_send_notify_data_callback(void);　蓝牙连接成功后协议在空闲的时侯会调用本回调函数

2、void UsrProcCallback(void); 蓝牙协议 peripheral 角色时会周期性回调本函数，无论是在广播状态还是连接状态

3、void ser_prepare_write(unsigned short handle,

unsigned char* attValue, unsigned short attValueLen, unsigned short att_offset);

4、void ser_execute_write(void);

以上两个函数为队列写数据回调函数。

5、unsigned char* getDeviceInfoData(unsigned char* len);

本函数是 GATT 中设备名称获取的回调函数。

6、void att_server_rdByGrType(unsigned char pdu_type, unsigned char attOpcode,

unsigned short st_hd, unsigned short end_hd, unsigned short att_type);

蓝牙 GATT 查询服务的回调函数

7、void ser_write_rsp(unsigned char pdu_type/*reserved*/, unsigned char attOpcode/*reserved*/,

unsigned short att_hd, unsigned char* attValue/*app data pointer*/,

unsigned char valueLen_w/*app data size*/);

蓝牙 GATT 写操作回调函数

8、void server_rd_rsp(unsigned char attOpcode, unsigned short attHandle, unsigned char pdu_type);

蓝牙 GATT 读操作回调函数，对 rd_req 的回应

9、void server_blob_rd_rsp(u8 attOpcode, u16 attHandle, u8 dataHdrP,u16 offset);

蓝牙 GATT 读操作回调函数，对 blob_rd_req 的回应

10、void ConnectStausUpdate(unsigned char IsConnectedFlag);

蓝牙连接状态更新回调函数。上电运行后会调用一次，IsConnectedFlag=0；进入连接状态会调用一次，IsConnectedFlag=1；断开连接进入广播状态会调用一次，IsConnectedFlag=0。

8. FAQ

1. 如何设置广播间隔？

A：有两种方法 1）在调用接口 ble_run_interrupt_start（）中由参数指定；2）调用接口函数 ble_set_interval()进行设置。

2. 如何设置发射功率？

A：在芯片初始化接口函数 radio_initBle（）中由参数指定。

3. 如何修改广播内容及设备名字（Device Name）？

A：广播内容可通过接口函数 ble_set_adv_data()设置。如果仅仅需要修改设备名字，也可通过 app.c 文件中的宏定义 DeviceInfo 进行修改。

4. 如何获取蓝牙地址（MAC）？

A：芯片接口初始化函数 radio_initBle（）输出参数给出。

5. 如何自定义特征值？

A：通过 app.c 文件中直接给特征值变量 AttCharList 赋值定义即可。

6. 如何自定义 Service？

A：step1 通过 app.c 文件中直接分配 Service 及对应的特征值；

step2 通过 app.c 文件中在回调函数 att_server_rdByGrType（）内应答 Service 句柄范围及 UUID。

7. 如何用特征值显示用户自己的软件版本信息？

A：通过 app.c 文件回调函数 server_rd_rsp（）内对软件版本特征值应答用户自己的版本信息。

8. 如何获取连接状态？

A：通过 app.c 的回调函数 ConnectStausUpdate 的输入参数获得，零表示断开，非零表示连接。

9. 如何通过蓝牙发送数据？

A：首先需要通过手机 App 端将对应特征值的 Notify 使能，然后在 app.c 文件中回调函数

gatt_user_send_notify_data_callback（）中调用接口函数 sconn_notifydata（）实现，但不得阻塞调用。

10. 如何接收蓝牙收到的数据？

A：蓝牙收到的数据会通过回调函数 ser_write_rsp（）中对应特征值通知，用户可按需保存。

11. 低功耗处理情况如何？

A：可以根据不同的应用来确定低功耗模式。

对于灯控应用，可以使用 SLEEP 低功耗模式，不能使用 STOP 和 STANDBY 模式，否则 PWM 输出会停止；

对于数据透传应用，可以使用 SLEEP 或者 STOP 低功耗模式；

对于超时关机的应用，可以使用 STANDBY 低功耗模式。这种低功耗模式下的电流消耗约 2uA，但是只能通过 PA0 引脚的上升沿唤醒，或者通过 NRST 引脚进行复位唤醒。

12. 如何实现按键检测功能等用户自己的逻辑代码？

A：为了保证蓝牙功能的正常工作，用户不得长时间阻塞运行，耗时较长的操作不要放在 BLE 接口函数中，建议在用户任务中进行。以按键检测为例，按键检测功能包括：1）检测；2）按键功能执行。

检测：可以在用户任务循环中扫描按键对应 IO 状态。

执行：按键对应的功能执行内容如果要调用 BLE 接口函数，建议在蓝牙的回调函数 UsrProcCallback() 中执行。

13. 队列写数据回调函数如何使用？

A：用户（或 App）在手机端对某个具有写属性的特征值进行写操作，手机蓝牙驱动会根据写操作数据量的多少会相应触发两类不同的写操作回调函数。

当写操作数据量小于等于 20 字节，触发 ser_write_rsp(...)回调函数；

当写操作数据量大于 20 字节，手机蓝牙底层会自动拆分数据并触发队列写数据回调函数，每个被拆分的数据段会依次触发 ser_prepare_write(...)，该函数携带了数据段所在的偏移量及数据长度。当所有数据片段都传输完成后会触发 ser_execute_write()回调函数，用户需要准备足够长的 Buffer 来保存接收到的数据。队列写的机制在结束之前是无法知道本次队列传输总的数据长度的，缓冲 Buffer 的长度应由具体应用交互协议确定（不是蓝牙协议确定）。通常队列写的操作遵循如下的步骤：

ser_prepare_write(...) ： slice1，offset=0，slice1_length = L1

ser_prepare_write(...) ： slice2，offset=L1，slice1_length = L2

ser_prepare_write(...) ： slice3，offset=L1+L2，slice1_length = L3

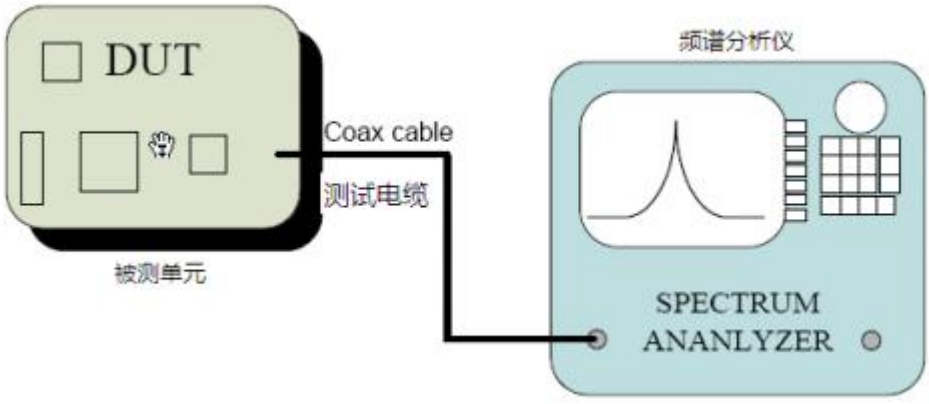
....

ser_execute_write() 结束。

14、如何调整晶体频偏？

A：RF 使用 16MHz 晶体，不同规格晶体的频偏是不同的，需要在实际 PCB 上进行测量和调整。通过载波输出可以测量 RF 的射频输出功率，也可以用来调节晶体的负载电容，使晶体工作在准确的频率上。

射频发射信号通过产品的天线耦合到频谱分析仪。测试装置连接如下：



以 2480MHz 载波发射为例，频谱分析仪设置：

a, 中心频率设置（Center Frequency）为 2480MHz
b, 扫频宽度（Span）为 1~10M
c, 最大信号输入幅度（Amplitude）应大于被测产品射频输出功率 3~10 dB
d, 分辨率带宽（RBW）为 30K~300KHz（一般自动调整）

在频谱分析仪上可以观察到载波的中心频率和最高幅度。

- 通过调节晶体的负载电容来调节载波中心频率接近 2480MHz（-50KHz，+50KHz）
- 通过调节天线匹配电路使载波幅度在配置范围内

在 16MHz 晶体和 PCB 保持不变的情况下，以上测试只需做一次，确定合适的负载电容值和天线匹配参数。

9. 参考文件

1. Bluetooth SIG core-specification v5.0 https://www.bluetooth.com/zh-cn/specifications/bluetooth-core-specification

2. Bluetooth SIG GATT https://www.bluetooth.com/zh-cn/specifications/gatt

3. BLE 接口函数说明 v1.0