1 在gatt\_uuid128.h 自定义UUID，使用宏定义UUID128（服务名/特征名，128位uuid）

UUID128(uuid\_service\_adcTst,0x01,0x03,0x07,0x09,0x0b,0x0d,0x0f,0x02,0x03,0x01,0x04,0x43,0x23,0x32,0x33,0x78)

UUID128(uuid\_characteristic\_adcTst,0x01,0x03,0x07,0x0a,0x0b,0x0d,0x0f,0x02,0x04,0x01,0x04,0x93,0x23,0x32,0x33,0x79)

名字+16字节uuid

2 gatt\_db.h定义服务

PRIMARY\_SERVICE\_UUID128(service\_device\_adc, uuid\_service\_adcTst)

CHARACTERISTIC\_UUID128(char\_adc\_value,uuid\_characteristic\_adcTst,(gGattCharPropRead\_c | gGattCharPropNotify\_c | gGattCharPropWrite\_c))

VALUE\_UUID128(value\_adc\_value, uuid\_characteristic\_adcTst, gPermissionFlagReadable\_c | gPermissionFlagWritable\_c ,1, 0x00)

CCCD(cccd\_adc)

3 在profiles下创建一个新文件夹，放自定义服务，在文件夹下放两个文件，xxx\_interface.h和xxx\_services.c

4 在interface.h自定义数据结构，类似

typedef struct psConfig\_tag  
{  
    uint16\_t    serviceHandle;                 /\*!<Service handle \*/  
    uint8\_t     potentiometerValue;            /\*!<Input report field \*/  
} psConfig\_t;

声明两个开启服务的函数，类似

bleResult\_t Ps\_Start(psConfig\_t \*pServiceConfig);  
bleResult\_t Ps\_Stop(psConfig\_t \*pServiceConfig);

两个注册客户端注册服务，类似

bleResult\_t Ps\_Subscribe(deviceId\_t clientDeviceId);  
bleResult\_t Ps\_Unsubscribe();

用于读写更新特征的函数，类似

bleResult\_t Ps\_RecordPotentiometerMeasurement (uint16\_t serviceHandle, uint8\_t newPotentiometerValue);

5 service.c要实现

一个静态变量用于存储ID

static deviceId\_t mPs\_SubscribedClientId;

实现以上函数

bleResult\_t Ps\_Start (psConfig\_t \*pServiceConfig)  
{      
    /\* Clear subscibed clien ID (if any) \*/  
    mPs\_SubscribedClientId = gInvalidDeviceId\_c;  
      
    /\* Set the initial value defined in pServiceConfig to the characteristic values \*/  
    return Ps\_RecordPotentiometerMeasurement (pServiceConfig->serviceHandle,   
                                             pServiceConfig->potentiometerValue);  
}  
  
bleResult\_t Ps\_Stop (psConfig\_t \*pServiceConfig)  
{  
  /\* Unsubscribe current client \*/  
    return Ps\_Unsubscribe();  
}  
  
bleResult\_t Ps\_Subscribe(deviceId\_t deviceId)  
{  
   /\* Subscribe a new client to this service \*/  
    mPs\_SubscribedClientId = deviceId;  
  
    return gBleSuccess\_c;  
}  
  
bleResult\_t Ps\_Unsubscribe()  
{  
   /\* Clear current subscribed client ID \*/  
    mPs\_SubscribedClientId = gInvalidDeviceId\_c;  
    return gBleSuccess\_c;  
}

这函数的 特征uuid变量是定义在uuid128里的，但是我实验时候发现，直接填进函数里会报未定义错误，后来使用了extern解决了问题

bleResult\_t Ps\_RecordPotentiometerMeasurement (uint16\_t serviceHandle, uint8\_t newPotentiometerValue)  
{  
    uint16\_t  handle;  
    bleResult\_t result;  
  
    /\* Get handle of Potentiometer characteristic \*/  
    result = GattDb\_FindCharValueHandleInService(serviceHandle,  
        gBleUuidType128\_c, (bleUuid\_t\*)&potentiometerCharacteristicUuid128, &handle);  
  
    if (result != gBleSuccess\_c)  
        return result;  
  
    /\* Update characteristic value \*/  
    result = GattDb\_WriteAttribute(handle, sizeof(uint8\_t), (uint8\_t\*)&newPotentiometerValue);  
  
    if (result != gBleSuccess\_c)  
        return result;  
  
    Ps\_SendPotentiometerMeasurementNotification(handle);  
  
    return gBleSuccess\_c;  
}

实现发送提醒的函数

static void Ps\_SendPotentiometerMeasurementNotification  
(  
  uint16\_t handle  
)  
{  
    uint16\_t  hCccd;  
    bool\_t isNotificationActive;  
  
    /\* Get handle of CCCD \*/  
    if (GattDb\_FindCccdHandleForCharValueHandle(handle, &hCccd) != gBleSuccess\_c)  
        return;  
  
    if (gBleSuccess\_c == Gap\_CheckNotificationStatus  
        (mPs\_SubscribedClientId, hCccd, &isNotificationActive) &&  
        TRUE == isNotificationActive)  
    {  
        GattServer\_SendNotification(mPs\_SubscribedClientId, handle);  
    }  
}

6 在文件里，放置服务数据的地方，定义自定义的数据，类似

Static psConfig\_t psServiceConfig = {service\_potentiometer,0};

这个service\_potentiometer就是UUID定义的主服务

7 在BleApp\_Config() 里开启服务

Ps\_Start(&psServiceConfig);

8 在BleApp\_ConnectionCallback, 在gConnEvtConnected\_c这个状态下，调用

Ps\_Subscribe(peerDeviceId); 注册ID，开启对应服务定时器，实现定时器回调函数

# 9 如果需要处理，Notifications和写请求

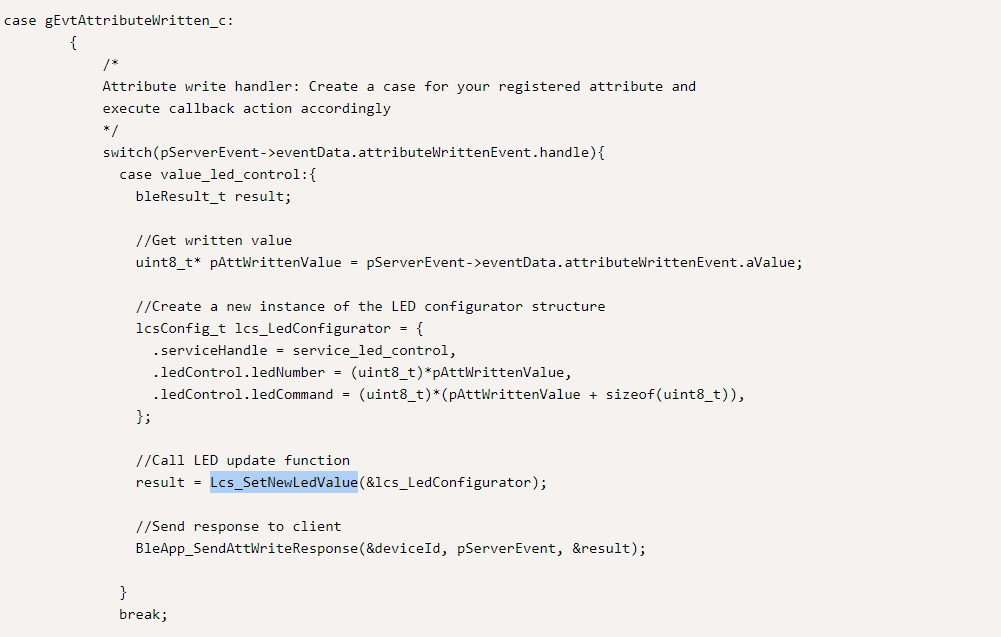
在BleApp\_GattServerCallback里的ccd写情况下，开启定时器，定时发送数据

写请求则要先用GattServer\_RegisterHandlesForWriteNotifications来注册函数

uint16\_t notifiableHandleArray[] = {value\_led\_control, value\_buzzer, value\_accelerometer\_scale, value\_controller\_command, value\_controller\_configuration};  
    uint8\_t notifiableHandleCount = sizeof(notifiableHandleArray)/2;  
    bleResult\_t initializationResult = GattServer\_RegisterHandlesForWriteNotifications(notifiableHandleCount, (uint16\_t\*)&notifiableHandleArray[0]);

在BleApp\_GattServerCallback里的属性写情况下，

同时要用GattServer\_RegisterHandlesForWriteNotifications来注册句柄，cpHandles放了VALUE值的句柄，需要在里面添加VALUE对应的第一个名字



实现sendAttWrite…

static void BleApp\_SendAttWriteResponse (deviceId\_t\* pDeviceId, gattServerEvent\_t\* pGattServerEvent, bleResult\_t\* pResult){  
  attErrorCode\_t attErrorCode;  
    
  // Determine response to send (OK or Error)  
  if(\*pResult == gBleSuccess\_c)  
    attErrorCode = gAttErrCodeNoError\_c;  
  else{  
    attErrorCode = (attErrorCode\_t)(\*pResult & 0x00FF);  
  }  
  // Send response to client    
  GattServer\_SendAttributeWrittenStatus(\*pDeviceId, pGattServerEvent->eventData.attributeWrittenEvent.handle, attErrorCode);  
}

按下app里的notify按钮则会触发cccd写事件 gEvtCharacteristicCccdWritten\_c

启用配对验证在app\_preinclude.h定义为1

/\*! Enable/disable use of bonding capability \*/

#define gAppUseBonding\_d 1

/\*! Enable/disable use of pairing procedure \*/

#define gAppUsePairing\_d 1

密钥定义

#define gPasskeyValue\_c 999999