客户端多连接

两个以上KW36

1 导出temp\_coll的freertos工程

2 修改app\_preinclude.h，作如下修改

**#define** gAppMaxConnections\_c 8

**#define** gTmrStackTimers\_c (6 + gAppMaxConnections\_c)

不使用低功耗，则

**#define** cPWR\_UsePowerDownMode 0

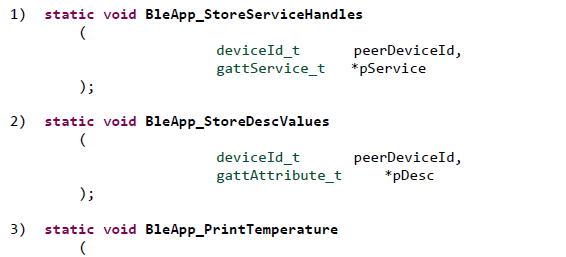
3 修改temperature\_collector.c

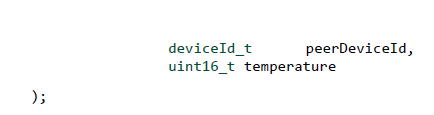
3.1 修改**static** appPeerInfo\_t mPeerInformation

改为**static** appPeerInfo\_t mPeerInformation[gAppMaxConnections\_c];

3.2 定义全局变量uint8\_t mActiveConnections = 0;

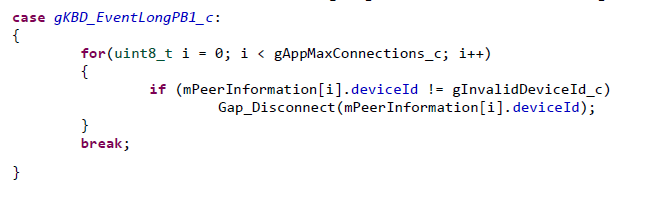
3.3 修改以下函数





3.4 注释static void DisconnectTimerCallback(void\* pParam);

3.5 找到BleApp\_HandleKeys函数，作如下修改



3.6 找到BleApp\_Config函数，修改mPeerInformation

**for**(uint8\_t i = 0; i < gAppMaxConnections\_c; i++)

{

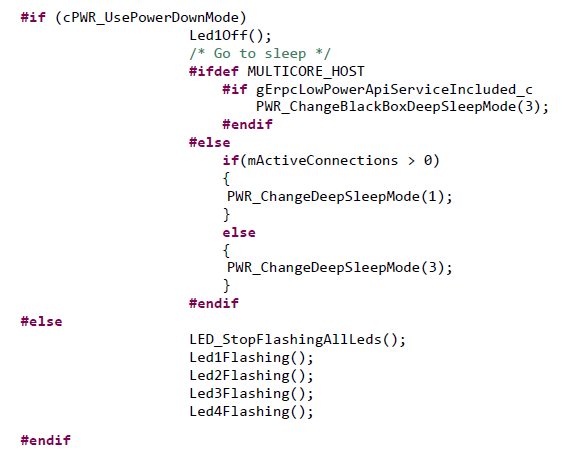
mPeerInformation[i].appState = *mAppIdle\_c*;

}

3.7 找到BleApp\_ScanningCallback

在case为*gScanStateChanged\_c下*

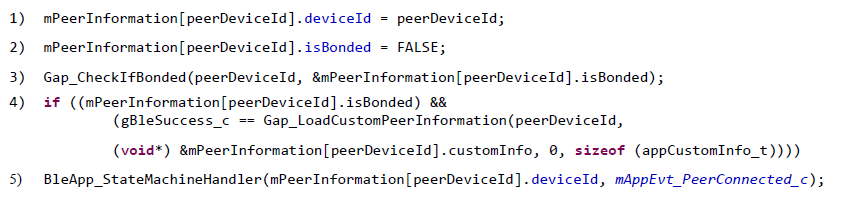
修改为



3.8 找到BleApp\_ConnectionCallback

3.8.1 在case为*gConnEvtConnected\_c*

修改如下



在该case结束最后几行



3.8.2 在case为*gConnEvtDisconnected\_c*

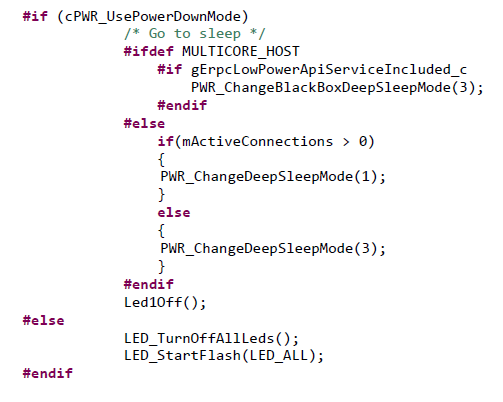
修改如下



在case最后几行加



找到低功耗代码作如下修改



3.8.3 在case为gConnEvtPairingComplete\_c

修改为

BleApp\_StateMachineHandler(mPeerInformation[peerDeviceId].deviceId, mAppEvt\_PairingComplete\_c);

3.8.4 在case为gConnEvtEncryptionChanged\_c

修改如下

**if**( gBleSuccess\_c != BleApp\_ConfigureNotifications(peerDeviceId) )

3.9 找到BleApp\_ServiceDiscoveryCallback

在case为 gServiceDiscovered\_c 作如下修改

**case** *gServiceDiscovered\_c*:

{

BleApp\_StoreServiceHandles(peerDeviceId,pEvent->eventData.pService);

}

**break**;

3.10 找到BleApp\_StoreServiceHandles

添加参数

**static** **void** **BleApp\_StoreServiceHandles**

(

deviceId\_t peerDeviceId,

gattService\_t \*pService

);

3.10.1 修改如下代码

mPeerInformation[peerDeviceId].customInfo.tempClientConfig.hService= pService->startHandle;

mPeerInformation[peerDeviceId].customInfo.tempClientConfig.hTemperature = pService->aCharacteristics[i].value.handle;

3.10.2 找到case为 gBleSig\_CharPresFormatDescriptor\_d

将mPeerInformation改为mPeerInformation[peerDeviceId]

3.10.3 case为gBleSig\_CCCD\_d

将mPeerInformation改为mPeerInformation[peerDeviceId]

3.11 找到BleApp\_StoreDescValues

做如上操作

3.12 找到BleApp\_PrintTemperature

做如上操作

3.13 找到BleApp\_GattClientCallback，将函数修改

BleApp\_StoreDescValues(serverDeviceId ,mpCharProcBuffer);

3.14 找到BleApp\_GattNotificationCallback，修改如下



3.15 找到BleApp\_StateMachineHandler

修改为**switch** (mPeerInformation[peerDeviceId].appState)

3.15.1 在case为mAppIdle\_c将mPeerInformation改为mPeerInformation[peerDeviceId]

3.15.2 在case为mAppExchangeMtu\_c，做如上相同操作

3.15.3 在case为mAppServiceDisc\_c，做如上相同操作

3.15.4 在case为mAppReadDescriptor\_c，做如上修改，同时修改

if( gBleSuccess\_c != BleApp\_ConfigureNotifications(peerDeviceId) )

3.15.5 在case为mAppRunning\_c，修改

(**void**)Gap\_SaveCustomPeerInformation(mPeerInformation[peerDeviceId].deviceId,

(**void** \*)&mPeerInformation[peerDeviceId].customInfo, 0,

**sizeof**(appCustomInfo\_t));

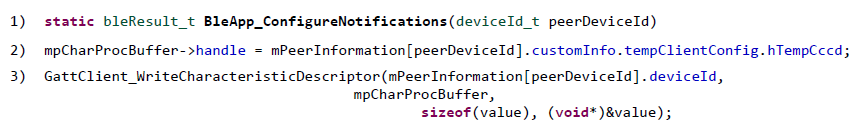
然后注释掉TMR\_StartLowPowerTimer函数

再修改

**if**(*gBleSuccess\_c* != BleApp\_ConfigureNotifications(peerDeviceId))

3.16 找到BleApp\_ConfigureNotifications

修改如下

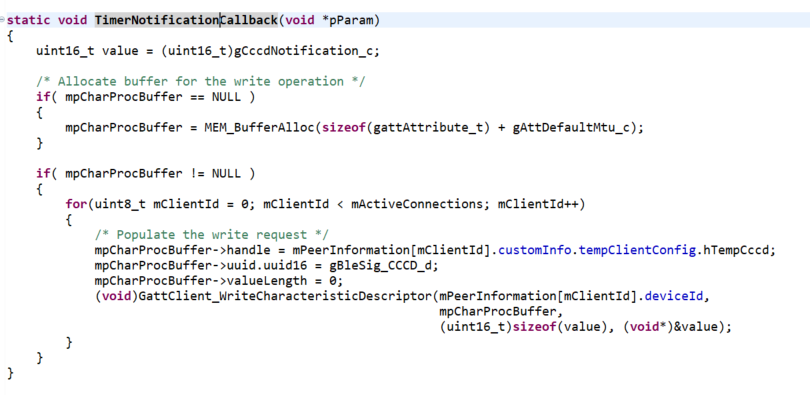


3.17 注释掉DisconnectTimerCallback所有实现

4 一个板子烧temp\_sense，一个烧该程序，当该程序连接成功后，按下sw2，可以再次扫描其他广播，并连接。其他实验过程与server相似。该程序大部分都在改

mPeerInformation为mPeerInformation[peerDeviceId]如果编译有问题，大概率是有些地方没有做这样修改。

如果想要不断收数据，则和之前server做一样操作，要分配定时器，但是要修改定时器回调函数

****

结果如下，数字是随机生成的

