700X & 750X & 880E API 文檔

本文文件是 700X & 750X & 880E 的应用程序开发接口（Application Programming Interface）。

文文件修改记录

版本	日期	修改內容	修改人
1.0	**2019/09/04**	初版	**Sandy Lin**
1.1	**2019/12/27**	修改 **Samling rate** 及相关程序代码说明	**Sam Hsu**

目录

常数定义

名称	值	说明
BT_HEADER	1	BLE 接收测量结果
BT_STANDBY	2	待机模式
BT_MEASURE	3	BLE 接收测量数据
BT_DOWNLOAD	**4**	**下载**
BT_DOWNLOAD_WAIT	0	下载结束进行等待
BT_DOWNLOAD_HEADER	2	下载 Device 记录信息
BT_DOWNLOAD_RAWD	3	下载 ECG rawdata
BT_DOWNLOAD_U1_COUNT	5	读取 User1 笔数
BT_DOWNLOAD_U2_COUNT	6	读取 User2 笔数
BT_SETUP	**5**	**设定 Device (时间)**
BT_SETUP_700X	2	设定 Device (时间)
BT_ERASE_FLASH	**6**	**删除记录**
BT_ERASE_USER1	3	删除 User1 记录
BT_ERASE_USER2	4	删除 User2 记录
BT_CONFIG_INFO	**7**	**读取 Device 信息**
BT_CONFIG_INFO_DEVICE	1	读取 Device ID
BT_CONFIG_INFO_SETTING	2	读取 Device FW_version
Realtime(ChannelNo)		
CH_HR	70	ECG 测量读取心率
CH_MMHG	71	BP 测量读取 mmhg
CH_ECG	73	ECG rawdata(心电图)目前 Realtime 125 个资料为 1 秒，收到值为 109 为 1mV
CH_AUTOSCALE	74	调整放大 (心电图) 1、2 为一倍;3 为两倍;4 为 3 倍 (此功能非必要)

蓝牙特征值

Service UUID - Characteristic UUID
A000-2a36 数据传输(读
-2a37 数据传输(写

蓝芽指令

取得 User 1 或 User2 笔数 (880E 只有 User1)

取得 User 1 或 User 2 笔数
主要 **Function - public static void** download_user(**int** user)

function 传入参数：

- **user：** 取得数据之用户

BLE 传送指令：

```
// user = 1:
    -   data2[1] = BT_DOWNLOAD;
    -   data2[2] = BT_DOWNLOAD_U1_COUNT;
// user = 2:
    -   data2[1] = BT_DOWNLOAD;
    -   data2[2] = BT_DOWNLOAD_U2_COUNT;
mBluetoothLeService.writeCharacteristic(characteristic2, data2);
```

BLE 传回：

```
于 DataList functionBLE(final int[] value)
{
    format = value[1];
    bpCmd = value[2];
    if (format == BT_DOWNLOAD) {
        int recordsCount1, recordsCount2;
        if (bpCmd == BT_DOWNLOAD_U1_COUNT) {
            recordsCount1 = value[3];    //User 1 笔数
            recordsCount2 = 0;
        }
        else if (bpCmd == BT_DOWNLOAD_U2_COUNT) {
            recordsCount1 = 0;
            recordsCount2 = value[4];    //User 2 笔数
        }
    }
}
```

取得 User 1 或 User2 资料(880E 只有 User1)

取得 User 1 或 User 2 资料

主要 **Function –** 取得笔数后直接传送此指令

BLE 传送指令：

```
//判断数据笔数大于 0，则传送取得数据指令
data2[1] = BT_DOWNLOAD;
data2[2] = BT_DOWNLOAD_HEADER;
data2[3] = (byte) headerSeq;     // headerSeq:  从 0 开始  为第一笔数据
mBluetoothLeService.writeCharacteristic(characteristic2, data2);
```

BLE 传回：

```
于 DataList functionBLE(final int[] value) {
    format = value[1];
    bpCmd = value[2];
    if (bpCmd == BT_DOWNLOAD_HEADER) {
        seq= value [4];
        year= value [5];    //年
        month= value [6];    //月
        day= value [7];    //日
        hour= value [8];    //时
        minute= value [9];    //分
        second= value [10];    //秒
        UserMode= value [11]    //目前下载的 user
        HeartRate= value [13]//心率
        HighBloodPressure = value [14] + value [15] * 256;    //SYS
        LowBloodPressure = value [16] + value [17] * 256;    //DIA
        if (HighBloodPressure == 0 && LowBloodPressure == 0)
            AnalysisType = TYPE_ECG;          //ECG 量测模式
        else
            AnalysisType = TYPE_BP;          //BP 量测模式
        WHOIndicate= value [18];
        value [19]& 0x01 ==0x01 //  显示  OK
        value [19]& 0x02 ==0x02 //    ECG noise  ，  HR 显示’EE’
        value [19]& 0x04 ==0x04 // ECG rhythm
        value [19]& 0x08 ==0x08 //ECG wave
        value [19]& 0x10 ==0x10 // ECG pause
        value [19]& 0x20 ==0x20 // ECG  心率 fast
        value [19]& 0x40 ==0x40 // ECG  心率 slow
        value [19]& 0x80 ==0x80 // BP  有 AF
    }
}
```

取得 ECG rawdata

取得 ECG rawdata
主要 **Function – public static void** download_file(RecordList header) {}

function 传入参数：

- **header：欲下载资料的 header**

BLE 传送指令：

```
UserMode = (byte) header.UserMode;
headerSeq = (byte) header.Seq;
downloadBufSeq = 1;
data2[1] = BT_DOWNLOAD;
data2[2] = BT_DOWNLOAD_RAWD;
data2[3] = UserMode; //要下载的 userMode, 0=user1, 1=user2
data2[4] = headerSeq; //要下载的 seq
data2[5] = downloadBufSeq; //从 1 开始传值，收到 256byte 传送下一个，收满 12* 256 + 68 * 125 停止
mBluetoothLeService.writeCharacteristic(characteristic2, data2);
```

BLE 传回：

```
于 DataList functionBLE(final int[] value) {
    format = value[1]; // BT_DOWNLOAD
    bpCmd = value[2]; // BT_DOWNLOAD_RAWD
    if (bpCmd == BT_DOWNLOAD_RAWD) {
```

```
int Cmd = value[3];

if (Cmd < 15) {

    for (int i = 0; i < 16; i++)

        FlashBuffer[Cmd * 16 + i] = (byte) value[i + 4];

}

 else if (Cmd == 15) {
     for (int i = 0; i < 16; i++)
         FlashBuffer[Cmd * 16 + i] = (byte) value[i + 4];

     if (currentSize < 12 * 256 + 68 * 125) {
         for (int i = 0; i < 256; i++) {
             rawDataBuf[currentSize++] = FlashBuffer[i];
         }
     }

      if (currentSize >= 12*256+68*125)   {

           data2[1] = BT_DOWNLOAD;

           data2[2] = BT_DOWNLOAD_WAIT;
               mBluetoothLeService.writeCharacteristic(characteristic2, data2);

           //下载 Rawdata 完成，转换成 ECG 数据

           for (int i = 0, j = 0; i < (12 * 256 + 68 * 125) / 2 ; i++, j += 2) {

                 iFirstByte = (short) (0x00FF & ((short) rawDataBuf[j]));

                 iSecondByte = (short) (0x00FF & ((short) rawDataBuf[j + 1]));

                 rawData[i] = (short) (iFirstByte << 8 | iSecondByte);

            }

     }

     else {

       //  传送下一个

           data2 [1] = BT_DOWNLOAD;

           data2 [2] = BT_DOWNLOAD_RAWD;

           data2 [3] = (byte) UserMode;

           data2 [4] = (byte) Seq;
```

```
                              data2 [5] = (byte) ++downloadBufSeq;

                              mBluetoothLeService.writeCharacteristic(characteristic2, data);

                      ]

              }

        }
}
```

取得装置信息

取得装置信息
主要 Function – public static void get_info() {}

```
BLE 传送指令:

//取得 DeviceID
    data[1] = BT_CONFIG_INFO;

    data[2] = BT_CONFIG_INFO_DEVICE;

    mBluetoothLeService.writeCharacteristic(characteristic2, data);

//取得 FirmwareVersion

    data[1] = BT_CONFIG_INFO;

    data[2] = BT_CONFIG_INFO_SETTING;
    mBluetoothLeService.writeCharacteristic(characteristic2, data);


BLE  传回:
于 DataList functionBLE(final int[] value) {

   format= value [1];     // BT_CONFIG_INFO;

   bpCmd= value [2] ;

   if (format == BT_CONFIG_INFO) {

        if (bpCmd == BT_CONFIG_INFO_DEVICE) {

               int Device ID = value[12] * 256 * 256 * 256 + value [13] * 256 * 256 + value [14] * 256 + value [15];

               String DID= String.format("%08x", DeviceID)

               //继续取得 Firmware Version

               data[1] = BT_CONFIG_INFO;

               data[2] = BT_CONFIG_INFO_SETTING;
```

```
                mBluetoothLeService.writeCharacteristic(characteristic2, data);

        }

        else if (bpCmd == BT_CONFIG_INFO_SETTING) {

                FirmwareVersion = (value[3] * 256 + value[4]);

        }

}
```

同步装置时间

同步装置时间
主要 Function – **public static void** time_setting() {}

```
BLE 传送指令：
     data[1] = BT_SETUP;

     data[2] = BT_SETUP_700X;

     data[3] = year;        //年，2019 年则设定为 19

     data[4] = month;     //月

     data[5] = day;        //日

     data[6] = hour;       //时

     data[7] = minute;   //分

     data[8] = second;   //秒

    mBluetoothLeService.writeCharacteristic(characteristic2, data);

BLE 传回：
于 DataList functionBLE(final int[] value) {

   format= value[1];        // BT_SETUP;

   if(format = BT_SETUP)

   {

     /" 时间设定成功"/

   }

}
```

回到待机模式(进行 Realtime 测量)

回到待机模式(Realtime)
主要 Function – **public static void** standby() {}
BLE 传送指令： **data[1] = BT_STANDBY** mBluetoothLeService.writeCharacteristic(characteristic2, data);

Realtime 测量

Realtime 测量

BLE 传回：

```
于 DataList functionBLE(final int[] value) {

    format= value [1];
    if (format == BT_MEASURE) {
        int j = 0;
        for (int i = 4; i < 20; ) {
            ChannelNo = value[i++] & 0xff;
            ChannelMSB = value[i++] & 0xff;
            ChannelData = (value[i++] & 0xff) * 256;
            ChannelData = ChannelData + (value[i++] & 0xff);
            if (ChannelNo == CH_HR) {//HR
                HeartRate = ChannelMSB * 256 * 256 + ChannelData;
            }
            else if (ChannelNo == CH_MMHG) {//BP
                bpDiastolic = ChannelMSB * 256 * 256 + ChannelData;
            }
            else if (ChannelNo == CH_AUTOSCALE) {
                ecgSize = ChannelMSB * 256 * 256 + ChannelData;
            }
            else if (ChannelNo == CH_ECG) { //ecg data
                MDRawData = ChannelMSB * 256 * 256 + ChannelData;
                if (ecgCount < ecg_rawData.length) {   //125*34，共 34 秒
                    ecg_rawData[ecgCount] = (short) (MDRawData);
                    ecgCount++;
                    if (ecgCount > 125 * 3) { //从第 3 秒开始画
```

```
                                displayData[displayCount] = ecg_rawData[ecgCount - 1];
                                displayCount++;
                            }
                        }
                    }
                }
            }
}
```

测量结果

测量结果

```
BLE 传回:
于 DataList functionBLE(final int[] value) {
    format= value [1];
    if (format == BT_HEADER) {
          seq= value [4];
          year= value [5];   //年
          month= value [6];   //月
          day= value [7];   //日
          hour= value [8];   //时
          minute= value [9];   //分
          second= value [10];   //秒
          UserMode= value [11]//目前的 user
          HeartRate= value [13]//心率
          if (value [14] != 0 && value [16] != 0)
               AnalysisType = TYPE_BP;     //血压量测模式
          else
               AnalysisType = TYPE_ECG;     //ECG 量测模式
          HighBloodPressure = value [14] + value [15] * 256; //SYS
          HighBloodPressure = HighBloodPressure & 0x00FF //转正十六进制数字显示
          LowBloodPressure = value [16] + value [17] * 256; //DIA
          LowBloodPressure = HighBloodPressure & 0x00FF //转正十六进制数字显示
```

```
        WHOIndicate= value [18];

        value [19]& 0x01 ==0x01 //  显示  OK

        value [19]& 0x02 ==0x02 // ECG noise  ，  HR 显示’EE’

        value [19]& 0x04 ==0x04 // ECG rhythm

        value [19]& 0x08 ==0x08 //ECG wave

        value [19]& 0x10 ==0x10 // ECG pause

        value [19]& 0x20 ==0x20 // ECG  心率 fast

        value [19]& 0x40 ==0x40 // ECG  心率 slow

        value [19]& 0x80 ==0x80 // BP  有 AF

    }

}
```

删除 User 数据(880E 只有 User1)

删除 User 数据
主要 Function – public static void delete_user(int user) {}

```
BLE 传送指令：
/*删除 User1 数据*/
    strArray[1] = BT_ERASE_FLASH;
    strArray[2] = BT_ERASE_USER1;    //删除 User 1 数据
/*删除 User2 数据*/
    strArray[1] = BT_ERASE_FLASH;

    strArray[2] = BT_ERASE_USER2;    //删除 User 2 数据

BLE  传回：
于 DataList functionBLE(final int[] value) {

    format= value [1];

    if (format == BT_ERASE_FLASH) {

        //删除成功

    }

}
```