使用 Apollo3 SDK 合入与自建 Profile 指南

Version 0.1

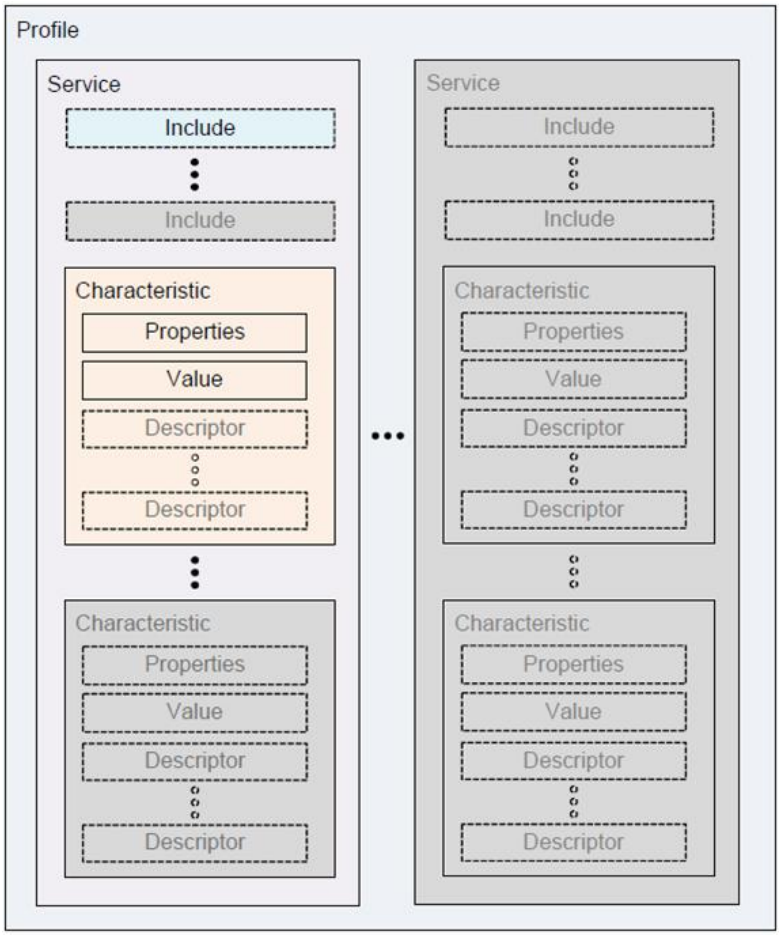
目录

1. Profile 概念介绍

1.1. 什么是 Profile

Profile 是蓝牙应用中定义的 GATT 属性配置文件. 该配置文件以 GATT 功能为基础，描述了用例、角色和一般性能。服务集合了封装设备组件性能的其他服务的特征和关系。这还包括在属性服务器中所用服务的层次结构、特征和属性。

Profile 文件的层次结果如下图:



层次结构的最顶层为配置文件，由完成用例时必需的一个或多个服务组成。一个服务由特征和对其他服务的参考组成。特征包括类型（表现为 UUID）、值、一组指示特征所支持操作的属性以及一组与安全相关的权限。特征还可能包括一个或多个描述符—与所拥有特征相关的元数据或配置标志。

GATT 对这些服务分组，以封装设备的组件性能并描述基于 GATT 功能的用例、角色和一般性能。这一框架定义了服务的规程、格式及特征，其中包括发现、读取、写入、通知和指示特征以及配置特征广播。

1.2. Profile 规格

蓝牙 SIG 组织已经对一些应用进行了规范性的规格定义. 在蓝牙 SIG 官方页面可以找到现有已定义好的 Profile 与服务规格, 网址如下:

https://www.bluetooth.com/specifications/gatt

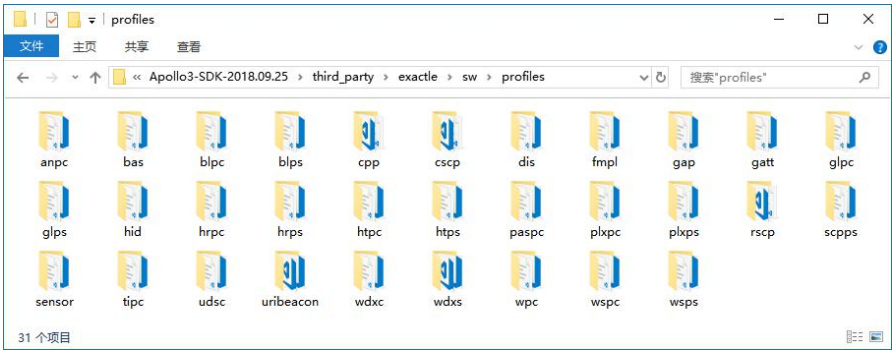
在该网址我们能找到诸如心率\血压\血糖\计步\自行车等等一系列的标准规范.当用户的应用设备需要这些服务时(譬如一台带有心率服务的手环),就可以直接利用这些规范代入到用户的蓝牙设计中,从而实现了跨平台标准化的快捷设计.

2. SDK 样例说明

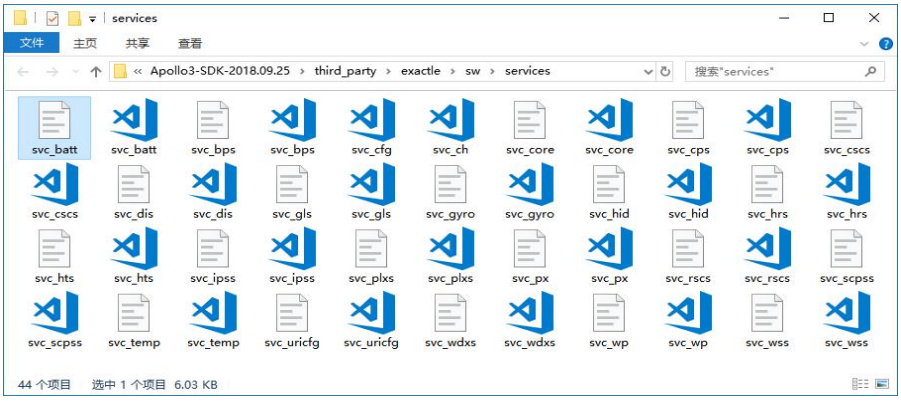
2.1. SDK 中 profile 的文件系统

在 Apollo3 SDK 中,分别将标准 Profile 与自定义 Profile 放置在不同的目录下以示区分.

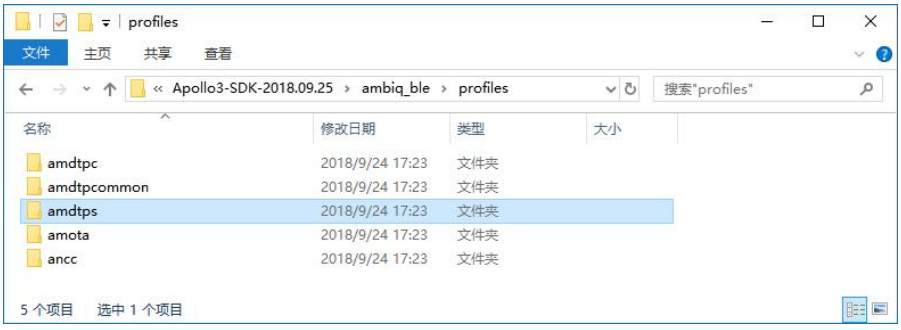
标准 Profile 目录



标准 Profile 下的服务目录

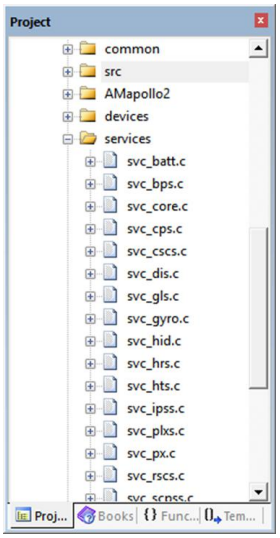


自定义 Profile 目录



自定义下的 Profile 服务目录

在项目文件中,我们可以直接调用现有的标准服务进入项目中



或者自建私有服务

2.2. SDK 中 profile 的例程分析

以 Freertos_fit 工程为例,在这个工程中,fit_main.c 作为 profile 配置文件加入了心率以及步速与电量服务.

以下代码枚举了各服务的事件入口

```
/*! WSF message event enumeration */
enum
{
  FIT_HR_TIMER_IND = FIT_MSG_START,          /*! Heart rate measurement timer expired */
  FIT_BATT_TIMER_IND,                        /*! Battery measurement timer expired */
  FIT_RUNNING_TIMER_IND                      /*! Running speed and cadence measurement timer expired */
};
```

以下代码将服务属性加入到蓝牙配置中:

```
/**************************************************************************************************
  Client Characteristic Configuration Descriptors
**************************************************************************************************/

/*! enumeration of client characteristic configuration descriptors */
enum
{
  FIT_GATT_SC_CCC_IDX,                       /*! GATT service, service changed characteristic */
  FIT_HRS_HRM_CCC_IDX,                       /*! Heart rate service, heart rate monitor characteristic */
  FIT_BATT_LVL_CCC_IDX,                      /*! Battery service, battery level characteristic */
  FIT_RSCS_SM_CCC_IDX,                       /*! Runninc speed and cadence measurement characteristic */
  FIT_NUM_CCC_IDX
};

/*! client characteristic configuration descriptors settings, indexed by above enumeration */
static const attsCccSet_t fitCccSet[FIT_NUM_CCC_IDX] =
{
  /* cccd handle               value range                  security level */
  {GATT_SC_CH_CCC_HDL,         ATT_CLIENT_CFG_INDICATE,     DM_SEC_LEVEL_NONE},     /*
FIT_GATT_SC_CCC_IDX */
  {HRS_HRM_CH_CCC_HDL,         ATT_CLIENT_CFG_NOTIFY,       DM_SEC_LEVEL_NONE},     /*
```

```
FIT_HRS_HRM_CCC_IDX */
  {BATT_LVL_CH_CCC_HDL,          ATT_CLIENT_CFG_NOTIFY,          DM_SEC_LEVEL_NONE},          /*
FIT_BATT_LVL_CCC_IDX */
  {RSCS_RSM_CH_CCC_HDL,          ATT_CLIENT_CFG_NOTIFY,          DM_SEC_LEVEL_NONE}           /*
FIT_RSCS_SM_CCC_IDX */
};
```

在 FitStart 函数下添加注册服务:

```
void FitStart(void)
{
  /* Register for stack callbacks */
  DmRegister(fitDmCback);
  DmConnRegister(DM_CLIENT_ID_APP, fitDmCback);
  AttRegister(fitAttCback);
  AttConnRegister(AppServerConnCback);
  AttsCccRegister(FIT_NUM_CCC_IDX, (attsCccSet_t *) fitCccSet, fitCccCback);

  /* Register for app framework callbacks */
  AppUiBtnRegister(fitBtnCback);

  /* Initialize attribute server database */
  SvcCoreAddGroup();
  //注册心率服务
  SvcHrsCbackRegister(NULL, HrpsWriteCback);
  SvcHrsAddGroup();
  //注册设备信息服务
  SvcDisAddGroup();
  //注册电池服务
  SvcBattCbackRegister(BasReadCback, NULL);
  SvcBattAddGroup();
  //注册步速服务
  SvcRscsAddGroup();

  /* Set running speed and cadence features */
  RscpsSetFeatures(RSCS_ALL_FEATURES);

  /* Reset the device */
  DmDevReset();
}
```

在 FitHandlerInit 函数中加入事件处理句柄:

```
/*************************************************************************************************/
/*!
 *  \fn         FitHandlerInit
 *
 *  \brief    Application handler init function called during system initialization.
 *
 *  \param    handlerID    WSF handler ID.
 *
 *  \return None.
 */
/*************************************************************************************************/
void FitHandlerInit(wsfHandlerId_t handlerId)
{
  APP_TRACE_INFO0("FitHandlerInit");

  /* store handler ID */
  fitHandlerId = handlerId;

  /* Set configuration pointers */
  pAppAdvCfg = (appAdvCfg_t *) &fitAdvCfg;
  pAppSlaveCfg = (appSlaveCfg_t *) &fitSlaveCfg;
  pAppSecCfg = (appSecCfg_t *) &fitSecCfg;
  pAppUpdateCfg = (appUpdateCfg_t *) &fitUpdateCfg;

  /* Initialize application framework */
  AppSlaveInit();

  /* Set stack configuration pointers */
  pSmpCfg = (smpCfg_t *) &fitSmpCfg;
```

```
    /* initialize heart rate profile sensor */
    HrpsInit(handlerId, (hrpsCfg_t *) &fitHrpsCfg);
    HrpsSetFlags(CH_HRM_FLAGS_VALUE_8BIT | CH_HRM_FLAGS_ENERGY_EXP);

    /* initialize battery service server */
    BasInit(handlerId, (basCfg_t *) &fitBasCfg);
}
```

当事件发生时将信息传递给 fitProcMsg 进行总成处理:

```
static void fitProcMsg(fitMsg_t *pMsg)
{
    uint8_t uiEvent = APP_UI_NONE;

    switch(pMsg->hdr.event)
    {
        //到达计步速时间进行步速计算及上传
      case FIT_RUNNING_TIMER_IND:
        fitSendRunningSpeedMeasurement((dmConnId_t)pMsg->ccc.hdr.param);
        break;
      //到达计心率时间进行心率测算及上传
      case FIT_HR_TIMER_IND:
        HrpsProcMsg(&pMsg->hdr);
        break;
      //到达计电量时间进行电量测算及上传
      case FIT_BATT_TIMER_IND:
        BasProcMsg(&pMsg->hdr);
        break;
      //发送属性信息给上位机确认
      case ATTS_HANDLE_VALUE_CNF:
        HrpsProcMsg(&pMsg->hdr);
        BasProcMsg(&pMsg->hdr);
        break;
      //服务属性参数变更进行处理
      case ATTS_CCC_STATE_IND:
        fitProcCccState(pMsg);
        break;
      //其它事件
      case ATT_MTU_UPDATE_IND:
        APP_TRACE_INFO1("Negotiated MTU %d", ((attEvt_t *)pMsg)->mtu);
        break;
       ...
```

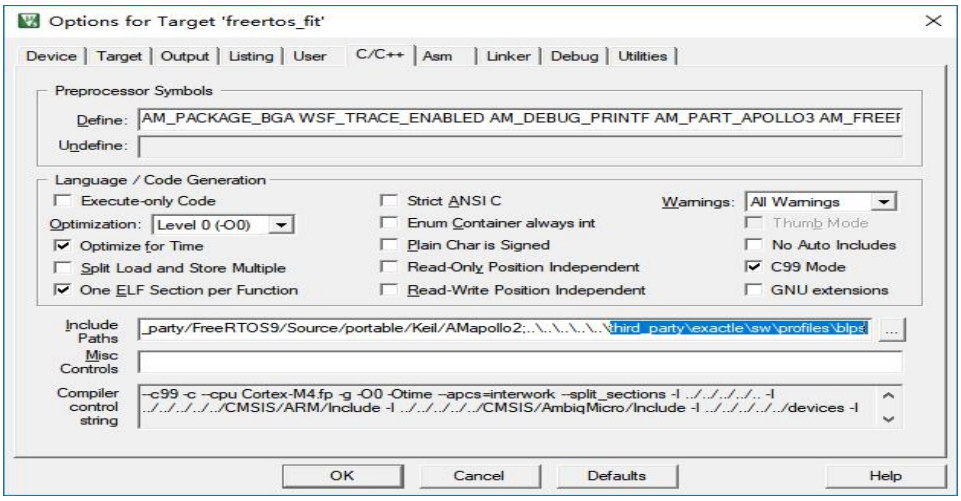
3. 加入标准服务

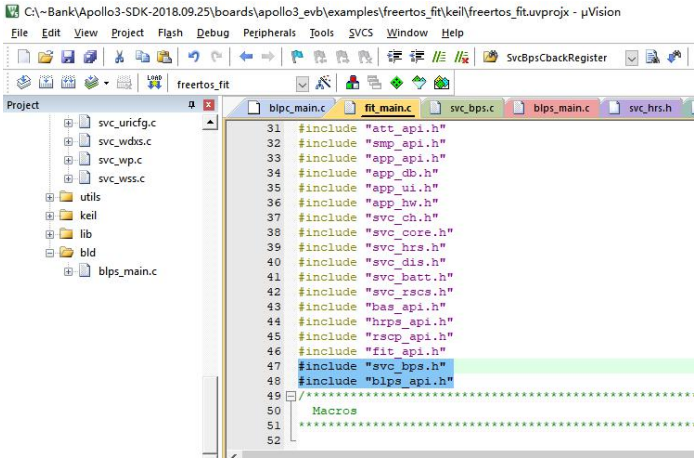
我们以将血压计服务加入到 fit profile 为例来看如何将 SDK 中已有的标准 profile 合入到现有工程中.

3.1. 项目配置与文件导入

血压计 Profile 属于蓝牙 SIG 组织已经定义好的标准设备,在 SDK 中标准 Profile 目录下可以找到该文件位置并将其加入到项目中:

在项目文件中加入该 Profile 文件并包含其路径.在 fit_main 的主 profile 内加入 api 和 svc 引用.





3.2. 在 profile 中添加服务

加入服务与属性：

```
/*! enumeration of client characteristic configuration descriptors */
enum
{
  FIT_GATT_SC_CCC_IDX,                    /*! GATT service, service changed characteristic */
  FIT_HRS_HRM_CCC_IDX,                    /*! Heart rate service, heart rate monitor characteristic */
  FIT_BATT_LVL_CCC_IDX,                   /*! Battery service, battery level characteristic */
  FIT_RSCS_SM_CCC_IDX,                    /*! Runninc speed and cadence measurement characteristic */
FIT_BPS_SM_CCC_IDX,
  FIT_NUM_CCC_IDX
};

/*! client characteristic configuration descriptors settings, indexed by above enumeration */
static const attsCccSet_t fitCccSet[FIT_NUM_CCC_IDX] =
{
  /* cccd handle              value range                  security level */
  {GATT_SC_CH_CCC_HDL,        ATT_CLIENT_CFG_INDICATE,     DM_SEC_LEVEL_NONE},     /*
FIT_GATT_SC_CCC_IDX */
  {HRS_HRM_CH_CCC_HDL,        ATT_CLIENT_CFG_NOTIFY,       DM_SEC_LEVEL_NONE},     /*
FIT_HRS_HRM_CCC_IDX */
  {BATT_LVL_CH_CCC_HDL,       ATT_CLIENT_CFG_NOTIFY,       DM_SEC_LEVEL_NONE},     /*
FIT_BATT_LVL_CCC_IDX */
  {RSCS_RSM_CH_CCC_HDL,       ATT_CLIENT_CFG_NOTIFY,       DM_SEC_LEVEL_NONE},     /*
FIT_RSCS_SM_CCC_IDX */
 {BPS_BPM_CH_CCC_HDL,         ATT_CLIENT_CFG_NOTIFY,       DM_SEC_LEVEL_NONE},     /*
FIT_HRS_HRM_CCC_IDX */
};
```

在 FitHandlerInit 函数中加入血压计句柄并初始化：

```
void FitHandlerInit(wsfHandlerId_t handlerId)
{
```

```
  ...
BlpsInit(handlerId, (blpsCfg_t *) &fitBlpsCfg);
  ...
}
```

在 FitStart 函数注册血压计服务：

```
void FitStart(void)
{
  ...
SvcBpsCbackRegister(NULL,NULL);
SvcBpsAddGroup();
  ...
}
```

3.3. 在 profile 中添加事件处理

在枚举事件中加入血压计事件：

```
/*! WSF message event enumeration */
enum
{
  FIT_HR_TIMER_IND = FIT_MSG_START,          /*! Heart rate measurement timer expired */
  FIT_BATT_TIMER_IND,                        /*! Battery measurement timer expired */
  FIT_RUNNING_TIMER_IND,                     /*! Running speed and cadence measurement timer expired */
 FIT_BPS_TIMER_IND,
};
```

在参数配置加入时间参数：

```
static const blpsCfg_t fitBlpsCfg =
{
  100         /*! Measurement timer expiration period in ms */
};
```

在 fitProcCccState 函数加入属性事件响应

```
static void fitProcCccState(fitMsg_t *pMsg)
{
  ...
 /* handle BPM level CCC */
  if (pMsg->ccc.idx == BPS_BPM_CH_CCC_HDL)
  {
    if (pMsg->ccc.value == ATT_CLIENT_CFG_NOTIFY)
    {
            BlpsMeasStartt((dmConnId_t) pMsg->ccc.hdr.param, FIT_BPS_TIMER_IND, FIT_BPS_SM_CCC_IDX);
    }
    else
    {
            BlpsMeasStop();
    }
    return;
  }
...
}
```

在 fitClose 事件加入停止函数

```
static void fitClose(fitMsg_t *pMsg)
{
 ....
BlpsMeasStop();
 ...
}
```

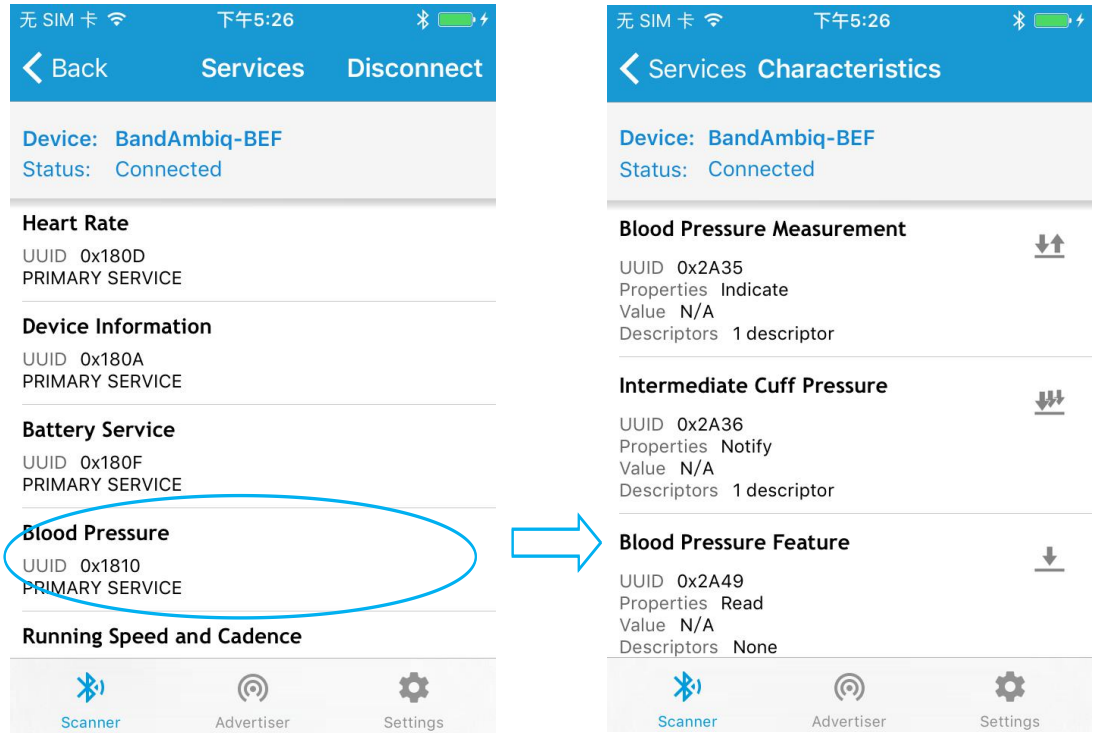
在事件总成中加入对应操作：

```
static void fitProcMsg(fitMsg_t *pMsg)
{
  uint8_t uiEvent = APP_UI_NONE;

  switch(pMsg->hdr.event)
  {
    ...
      case FIT_BPS_TIMER_IND:
            BlpsProcMsg(&pMsg->hdr);
            break;
```

```
        case ATTS_HANDLE_VALUE_CNF:
            HrpsProcMsg(&pMsg->hdr);
            BasProcMsg(&pMsg->hdr);
            BlpsProcMsg(&pMsg->hdr);
            break;
...
```

3.4. 运行效果：

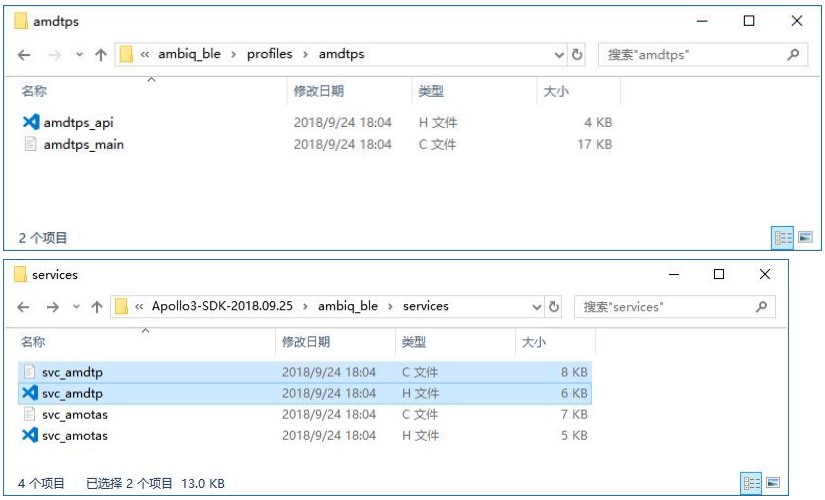


4. 加入私有服务

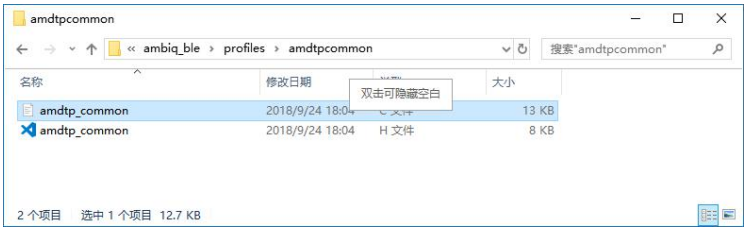
除了加入标准的蓝牙服务外，我们同样可以自定义服务并加入到 Profile 中.

4.1. 使用私有 profile 文档

为了不至于混淆项目中各个服务与属性的引用，我们应该独立一个 svc 服务的文档用于私有服务. 在本例中，我们使用 amdtps 例程中的自定义服务作为私有服务. 作为自定义的私有服务，在 SDK 中位于以下位置：

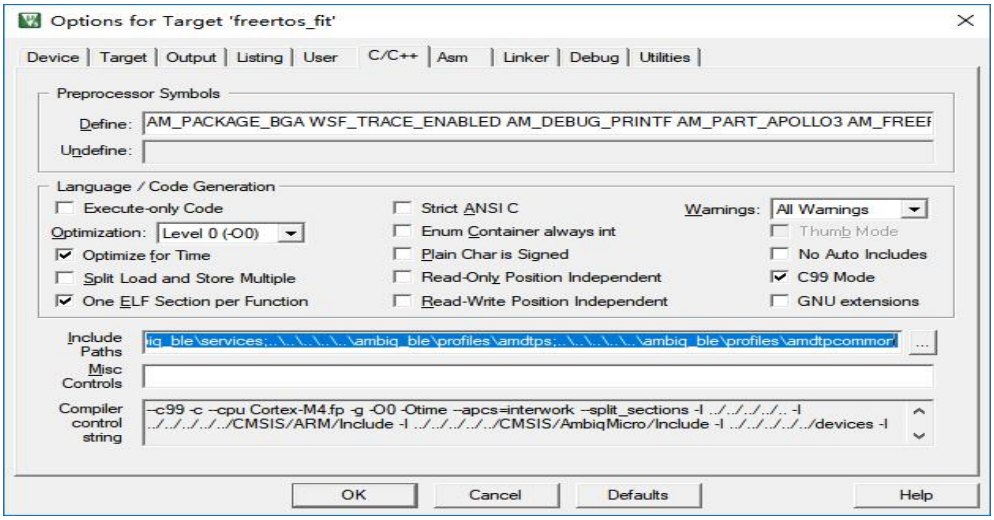


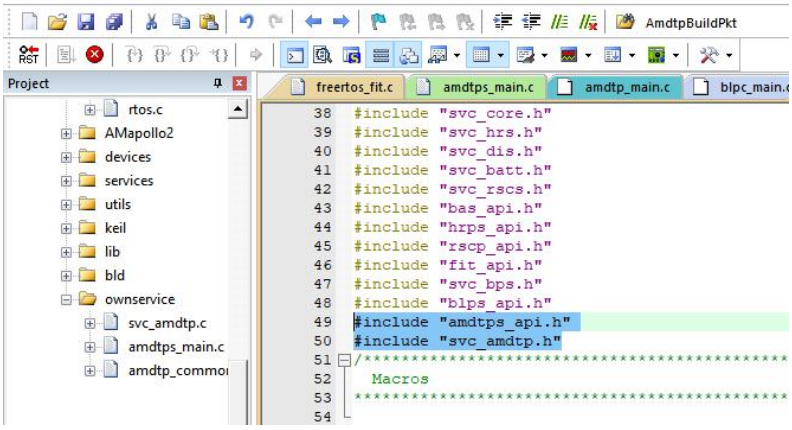
以及该服务中需要用到的数据打包等算法工具:



4.2. 引入到当前项目的 Profile 中

加入文档进入项目与编译路径





参照 amdpt_main.c 文件加入事件:

```
enum
{
  FIT_HR_TIMER_IND = FIT_MSG_START,          /*! Heart rate measurement timer expired */
  FIT_BATT_TIMER_IND,                        /*! Battery measurement timer expired */
  FIT_RUNNING_TIMER_IND,                     /*! Running speed and cadence measurement timer expired */
FIT_BPS_TIMER_IND,
AMDTP_TIMER_IND,
};
```

参数配置:

```
/*! AMDTPS configuration */
static const AmdtpsCfg_t amdtpAmdtpsCfg =
{
    0
};
```

属性与服务索引:

```
/*! enumeration of client characteristic configuration descriptors */
enum
{
  FIT_GATT_SC_CCC_IDX,                    /*! GATT service, service changed characteristic */
  FIT_HRS_HRM_CCC_IDX,                    /*! Heart rate service, heart rate monitor characteristic */
  FIT_BATT_LVL_CCC_IDX,                   /*! Battery service, battery level characteristic */
  FIT_RSCS_SM_CCC_IDX,                    /*! Runninc speed and cadence measurement characteristic */
FIT_BPS_SM_CCC_IDX,
AMDTP_GATT_SC_CCC_IDX,                    /*! GATT service, service changed characteristic */
  AMDTP_AMDTPS_TX_CCC_IDX,                /*! AMDTP service, tx characteristic */
  AMDTP_AMDTPS_RX_ACK_CCC_IDX,            /*! AMDTP service, rx ack characteristic */
  FIT_NUM_CCC_IDX
};

/*! client characteristic configuration descriptors settings, indexed by above enumeration */
static const attsCccSet_t fitCccSet[FIT_NUM_CCC_IDX] =
{
  /* cccd handle              value range                 security level */
  {GATT_SC_CH_CCC_HDL,        ATT_CLIENT_CFG_INDICATE,    DM_SEC_LEVEL_NONE},     /*
FIT_GATT_SC_CCC_IDX */
  {HRS_HRM_CH_CCC_HDL,        ATT_CLIENT_CFG_NOTIFY,      DM_SEC_LEVEL_NONE},     /*
FIT_HRS_HRM_CCC_IDX */
  {BATT_LVL_CH_CCC_HDL,       ATT_CLIENT_CFG_NOTIFY,      DM_SEC_LEVEL_NONE},     /*
FIT_BATT_LVL_CCC_IDX */
  {RSCS_RSM_CH_CCC_HDL,       ATT_CLIENT_CFG_NOTIFY,      DM_SEC_LEVEL_NONE},     /*
FIT_RSCS_SM_CCC_IDX */
 {BPS_BPM_CH_CCC_HDL,         ATT_CLIENT_CFG_NOTIFY,      DM_SEC_LEVEL_NONE},     /*
FIT_HRS_HRM_CCC_IDX */
 {AMDTPS_TX_CH_CCC_HDL,       ATT_CLIENT_CFG_NOTIFY,      DM_SEC_LEVEL_NONE},     /*
AMDTP_AMDTPS_TX_CCC_IDX */
  {AMDTPS_ACK_CH_CCC_HDL,     ATT_CLIENT_CFG_NOTIFY,      DM_SEC_LEVEL_NONE},     /*
AMDTP_AMDTPS_RX_ACK_CCC_IDX */
};
```

属性事件响应

```
static void fitProcCccState(fitMsg_t *pMsg)
{
  ...
  /* AMDTPS TX CCC */
  if (pMsg->ccc.idx == AMDTP_AMDTPS_TX_CCC_IDX)
  {
    if (pMsg->ccc.value == ATT_CLIENT_CFG_NOTIFY)
    {
      // notify enabled
      amdtps_start((dmConnId_t)pMsg->ccc.hdr.param, AMDTP_TIMER_IND, AMDTP_AMDTPS_TX_CCC_IDX);
      // AppSlaveSecurityeq(1);
      // if(bPairingCompleted == true)
      {
#if defined(AMDTPS_TXTEST)
        counter = 0;
        AmdtpsSendTestData(); //fixme
#endif
      }
    }
    else
    {
      // notify disabled
       amdtps_stop((dmConnId_t)pMsg->ccc.hdr.param);
    }
    return;
  }
...
}
```

事件总成:

```
static void fitProcMsg(fitMsg_t *pMsg)
{
  uint8_t uiEvent = APP_UI_NONE;

  switch(pMsg->hdr.event)
```

```
    {
      ...
        case AMDTP_TIMER_IND:
        amdtps_proc_msg(&pMsg->hdr);
        break;
          ...
        case ATTS_HANDLE_VALUE_CNF:
        HrpsProcMsg(&pMsg->hdr);
        BasProcMsg(&pMsg->hdr);
        BlpsProcMsg(&pMsg->hdr);
          amdtps_proc_msg(&pMsg->hdr);
        break;
...
```

句柄初始化:

```
void FitHandlerInit(wsfHandlerId_t handlerId)
{
  ...
   /* initialize amdtp service server */
   amdtps_init(handlerId, (AmdtpsCfg_t *) &amdtpAmdtpsCfg, amdtpDtpRecvCback, amdtpDtpTransCback);
   ...
}
```

数据收发的回调函数:

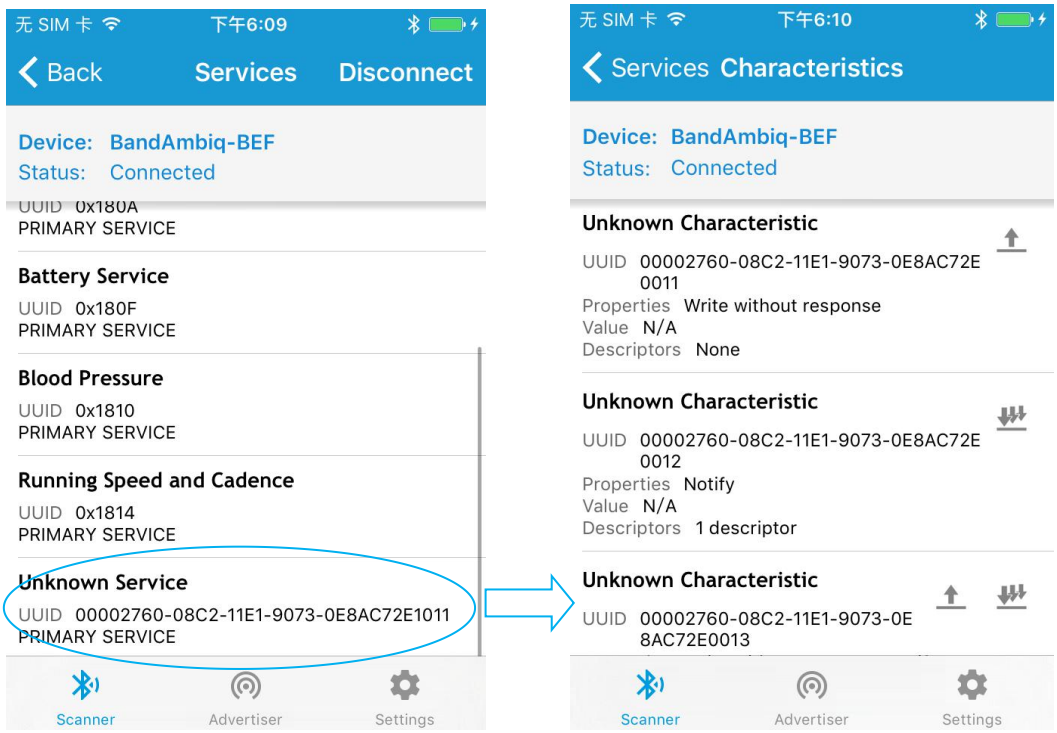
```
void amdtpDtpRecvCback(uint8_t * buf, uint16_t len)
{
//内容移除
}

void amdtpDtpTransCback(eAmdtpStatus_t status)
{
   //内容移除
}
```

注册服务与回调函数:

```
void FitStart(void)
{
   ...
   SvcAmdtpsCbackRegister(NULL, amdtps_write_cback);
   SvcAmdtpsAddGroup();
...
}
```

4.3. 运行效果



5. 其它

版本记录：

Date	Revision History	Reviser
2018-12-27	V0.1 draft created	Jacky he