AIC ble api

1. 用户初始化操作

```
app_ble_env_init();
app_ble_register_callbak(&ble_app_callbacks);
app_ble_addr_init(ble_addr, GAPM_STATIC_ADDR);
app_b1e_set_dev_name(APP_DFLT_DEVICE_NAME, AP P_DFLT_DEVICE_NAME_LEN);
app_b1e_set_adv_data(adv_data, build_adv_data());
init_adv_params();
```

2. 系统初始化完成后会自动开始adv

3. adv api

- stop adv: `app_ble_adv_stop` (停止adv)
- update adv interval: `app_ble_adv_param_update` (如果adv正在进行，需要先停止adv,再更改参数，改完参数后会自动开始adv)
- update adv data : `app_ble_adv_data_update` (如果adv正在进行，无需停止adv)
- restart adv: `app_b1e_adv_start` (停止adv后再次开始adv)

4. connection api

- disconnect: `app_ble_disconnect` (断开当前连接)
- update con param: `app_ble_update_con_params` (更新连接参数)

5. ble enable/disable

- ble enable: `ble_task_init`
- ble disable: `ble_task_deinit`

6. batt set/get

- get bstt level: `app_batt_get_lvl` (获取本地电池信息,batt_min, batt_max是工作的最小和最大电压，需要根据实际需求修改batt_min,batt_max)
- end batt level : `app_batt_send_lvl` (通过ble发送电池信息)

7. app_callbacks

- app init: `app_on_init` (profile对应的app init函数)
- app_add_svc: `app_on_add_svc` (添加profile service instance)
- pp_enable_profile: `app_on_enable_prf` (使能profile)
- app_connection: `app_on_connection` (连接建立时的callbak函数)
- app_disconnect: `app_on_disconnect` (断开连接时的callbak函数)
- app_con_param_update: `app_on_update_params_request` (收到连接参数改变时的callback函数)
- app_adv_status: `app_on_adv_status` (adv操作(set adv data/param, enable/disable adv)完成时的callback函数)

8. MSG

app.c ble通用应用方法实现，包括adv, connection
app_task.c 协议栈发给应用层消息处理
`KE_MSG_HANDLER_TAB(appm)` 中的msg是协议栈发给应用层的消息,需要关注的msg的含义分别如下：

- `KE_MSG_DEFAULT_HANDLER`: 协议栈发给profile对应APP层的消息，需要对应的profile中的msg handler来处理。
- `GAPM_PROFILE_ADDED_IND`: profile添加完成
- `GAPM_ACTIVITY_CREATE_IND`:adv创建完成
- `GAPM_ACTIVITY_STOP_IND`: adv停止
- `GAPM_CMP_EVT`: 一些通用设置操作完成，比如：set adv data/param等
- `GAPM_CONNECTION_REQ_IND`: 连接建立完成
- `GAPC_APRAM_UPDATA_REQ_IND`: 连接参数更新
- `GAPC_DISCONNECT_IND`: 连接断开

profile.c profile创建，销毁等
profile_task.c 是协议栈和用户的桥梁，与协议栈和profile_app通过MSG进行数据及状态交互
app_profile.c 与profile_task.c通过MSG进行交互，处理用户数据

9. 添加自定义profile

需要创建的文件分别是：**pfofile.c** 及 **pfofile_task.c**, **app_profile.c**及对应的.h文件。(`可参考smartconfig部分`)

profile:

profile.c负责profile的创建，销毁等操作
profie_task.c负责协议栈和用户层的数据转发，处理。
app_pfofile.c负责处理数据

- pfofile.c:
  需要关注结构体 `prf_task_cbs`

  ```
  /// Profile task callbacks.
  struct prf_task_cbs
  {
      /// Initialization callback
      prf_init_fnct    init;  //init profile: create att database, init
  profile task handler
      /// Destroy profile callback
      prf_destroy_fnct destroy;  //free memory
      /// Connection callback
      prf_create_fnct  create;   //connection callback
      /// Disconnection callbac
      prf_cleanup_fnct cleanup;  //Disconnection callback
  };
  ```

  需要实现pfofile创建销毁等对应的回调函数，如下所示：
  `profile_init`
  `profile_destory`
  `profile_create`
  `profile_cleanup`
  定义 `profile_itf`

接口函数 `profile_prf_its_get` ，并添加到 `prf_itf_get` 函数中
添加属性表 `profile_att_db`

- pfofile.h
  添加服务及特征值的定义
- profile_task.c
  实现消息处理相关的函数

用户层：

- app_profile.c
  主要添加函数如下：
  `app_profile_init` 初始化 `app_profile_env` (需要把此函数添加到 `ble_user_app_init_cb` 函数中)
  `app_profile_add_profile` 发送 MSG(`GAPM_PROFILE_TASK_ADD_CMD`) 给协议栈，添加profile(需要把此函数添加到 `app_add_svc_func_list` 中)
  `app_profile_enable_prf` 发送 MSG(`PROFILE_ENABLE_REQ`) 给profile task，使能pfofile(需要把此函数添加到 `ble_user_enable_prf_cb` 函数中)
  定义消息处理函数，处理profile发来的msg、data
  定义用户数据发送接收处理函数，处理数据
- 其他头文件及编译
  rwip_task.h中添加 `TASK_ID_PROFILE`
  rwprf_config.h 中添加宏控制： `BLE_PROFILE`
  btdm文件夹中的**config/includelist.txt sourcelist.txt**中添加profile及app_profile到编译目录。

自定义profile应用例程audtransmit（可用于客户参考，自行添加自定义profile）：

- audtransmit例程分为client端和server端：

  Client：

  audtransmitc.c：　　//btdm\ble\ble_profiles\audtransmit\audtransmitc\src

  audtransmitc.h

  audtransmitc_task.c：　　//btdm\ble\ble_profiles\audtransmit\audtransmitc\src

  audtransmitc_task.h

  app_audtransmitc.c:　　//btdm\ble\ble_app\app_audtransmitc

  app_audtransmitc.h

  Server:

  audtransmits.c:　　//btdm\ble\ble_profiles\audtransmit\audtransmits\src

  audtransmits.h

  audtransmits_task.c　　//btdm\ble\ble_profiles\audtransmit\audtransmits\src

  audtransmits_task.h

  app_audtransmits.c　　//btdm\ble\ble_app\app_audtransmits

  app_audtransmits.h

  并由应用层文件调用基础接口，配合其他模块实现与audio模块的结合使用：

  app_ble_audtransmit.c　　//modules\apps\src\ble

  app_ble_audtransmit.h

  app_ble_only.c　　//modules\apps\src\ble

  app_ble_only.h

- Server:
  - audtransmits.c提供Attributes Database以及协议栈注册/注销接口，并将自定义的UUID等database注册到协议栈ATT描述中，用于其他设备检索。

    ```
    const struct prf_task_cbs audtransmits_itf =
    {
       (prf_init_fnct) audtransmits_init,
       audtransmits_destroy,
       audtransmits_create,
       audtransmits_cleanup,
    };

    const struct prf_task_cbs* audtransmits_prf_itf_get(void)
    {
      return &audtransmits_itf;
    }
    ```

    audtransmits_prf_itf_get需要添加在**prf_user.c**中的prf_itf_get函数中，结合在rwip_task.h注册的TASK ID使用。

    自定义添加时需注意对应task的ID以及状态值。
  - audtransmits_task.c提供作为server端时，创建对应task之后，收到client端相关gatt指令write/read等操作时的处理函数以及作为server发送notification和indication等相关基础指令。
  - app_audtransmits.c 作为对上面两个profile文件的应用，提供向kernel注册task的调用流程，以及从app task（app_audtransmits.c 注册在app task）向audtransmits task（profile task）发送指令以及接收数据等用户层面API接口。

    另外提供应用层数据处理的注册函数，方便与其他模块配合使用。
  - app_ble_only.c作为对app_audtransmits.c中API接口与协议栈的接入流程使用。
  - app_ble_audtransmit.c作为应用层注册audio模块与app_audtransmits.c中cmd、data等接口对接以及实现应用逻辑。
- Cilent：

  与Server端接口类似。实现GATT协议中Cilent端支持的检索服务、write（cmd）、read，接收indication以及notification等相关操作。应用层也与Server端类似，app_ble_only.c实现接入协议栈，app_ble_audtransmit.c实现多模块对接。