Migration from Cordio to Mindtree stack

1. Overview

Starting from Ambiq Suite SDK v1.2.12, Ambiq Micro is offering Mindtree's Ethermind BLE Host Stack while continuing to support ARM Cordio stack. This document outlines the steps needed for customers to migrate their existing projects from Cordio stack to the Ethermind stack. Additional support for Ambiq products can be found at http://ambiqmicro.com/support/. Ambiq strongly recommends that new customers start with the Ethermind stack as the ARM Cordio stack will not include new BLE examples and features in future releases.

2. Turn on Bluetooth

2.1 Steps for Cordio stack: (refer to freertos_amota example)

- HciDrvRadioBoot() should be called first to configure BLE controller (e.g. EM9304) by applying any applicable patches before Bluetooth host stack communication.
- exactle_stack_init() from radio_task.c is called to initialize the Cordio stack. exactle_stack_init() does initialization of WSF timer, various modules of Cordio stack (e.g. master or slave role related modules) and also profile/application layer handler which handles profile/application specific events, callback, etc. Note that exactle_stack_init() doesn't need to be called again when turning on Bluetooth again after it's turned off before.

2.2 Steps Ethermind stack: (refer to mindtree_amota example)

- appl_init() is called from mindtree_main.c. appl_init() is a must-have implementation that is linked by Ethermind stack during firmware build, it should be implemented in user's application layer or appl.c located under third_party\ethermind\appl\profile_appl. Ethermind stack provides a single framework or middleware for peripheral or slave based projects, all the .c and .h files under the folder third_party\ethermind\appl\profile_appl are part of the framework or middleware that is responsible for interacting with Ethermind core stack through the stack APIs and handling events from the stack. Internally BT_ethermind_init() is called first and then BT_bluetooth_on() is called to turn on Bluetooth.
- Configuring EM9304 and initializing SPI driver will be triggered with calling appl_init(). The SPI transport layer for Ethermind will HCI SPI read and write threads interacting with Bluetooth controller (e.g. em9304).

3. Turn off Bluetooth

3.1 Steps for Cordio stack:

- First, make sure to disable any radio activity if any first, e.g. advertising, scanning/connecting and any connection, AND wait for Cordio stack to complete disabling radio activity through Cordio stack event, e.g. DM_ADV_STOP_IND, DM_SCAN_STOP_IND and DM_CONN_CLOSE_IND.
- HciDrvRadioShutdown() then can be called to de-initialize SPI transport layer and put Bluetooth controller (e.g. em9304) into deep sleep mode or de-assert mode.

3.2 Steps for Ethermind stack:

- First, make sure to disable any radio activity if any first, e.g. advertising, scanning/connecting and any connection, AND wait for Ethermind stack to complete disabling radio activity through the callback of HCI events.

- BT_bluetooth_off() then can be called to de-initialize SPI transport layer and put Bluetooth controller (e.g. em9304) into deep sleep mode or de-assert mode.

4. Turn on Bluetooth at runtime

4.1 Steps for Cordio stack:

- Assuming Bluetooth is off through the steps of the above 3.1 section;
- Call HciDrvRadioBoot() to configure EM9304 with SPI interface
- Then call DmDevReset() to trigger initialization of Cordio internal variables and state machines.

4.2 Steps for Ethermind stack:

- Assuming Bluetooth is off through the steps of the above 3.2 section;
- Call BT_bluetooth_on() to initialize SPI transport layer and put Bluetooth controller (e.g. em9304) into normal functional mode.

5. GATT service definition

5.1 Cordio stack GATT service definition:

- Refer to svc_amotas.c and svc_amtoas.h under the folder ambiq_ble\services of SDK for custom GATT service definition.

5.2 Ethermind stack GATT service definition:

- Refer to gatt_db.c and gatt_db.h under the folder ambiq_ble\profile_appl\amota of SDK for custom GATT service definition.
- Refer to EtherMind_GATT_DB_Customization.pdf (which is included in the SDK) for detailed information about GATT service customization.

6. Application Layer Interaction with Bluetooth Host Stack

6.1 Application Layer Interaction with Cordio Stack:

- In Cordio stack, application layer registers a few callback functions with different modules of Cordio stack, e.g. DM (device manager), ATT module, see the below code snippet from amota_main.c.

```
/* Register for stack callbacks */

DmRegister(amotaDmCback);

DmConnRegister(DM_CLIENT_ID_APP, amotaDmCback);

AttRegister(amotaAttCback);

AttConnRegister(AppServerConnCback);

AttsCccRegister(AMOTA_NUM_CCC_IDX, (attsCccSet_t *) amotaCccSet, amotaCccCback);
```

- Those callback functions then processes the incoming events from Cordio stack.
- The basic GAP APIs exposed to application for advertising, etc is defined in app_api.h, the following global variables are referenced by general app_slave and app_master modules which are the foundations for peripheral and central role device respectively, they must be declared by application layer:

```
/*! Configuration pointer for advertising */
extern appAdvCfg_t *pAppAdvCfg;


/*! Configuration pointer for extended advertising */
extern appExtAdvCfg_t *pAppExtAdvCfg;


/*! Configuration pointer for slave */
extern appSlaveCfg_t *pAppSlaveCfg;


/*! Configuration pointer for master */
extern appMasterCfg_t *pAppMasterCfg;


/*! Configuration pointer for extended master */
extern appExtMasterCfg_t *pAppExtMasterCfg;


/*! Configuration pointer for security */
extern appSecCfg_t *pAppSecCfg;


/*! Configuration pointer for connection parameter update */
extern appUpdateCfg_t *pAppUpdateCfg;


/*! Configuration pointer for discovery */
extern appDiscCfg_t *pAppDiscCfg;


/*! Configuration for application */
```

```
extern appCfg_t *pAppCfg;


/*! Configuration pointer for incoming request actions on master */
extern appReqActCfg_t *pAppMasterReqActCfg;


/*! Configurable pointer for incoming request actions on slave */
extern appReqActCfg_t *pAppSlaveReqActCfg;
```

6.2 Application Layer Interaction with Ethermind Stack:

- Ethermind stack provides a common framework or middleware that's suitable for peripheral (or slave) project like wearable band;
- The framework or middleware's implementation is located in the folder third_party\ethermind\appl\profile_appl;
- The framework by default will call the following predefined APIs:
  - #define APPL_PROFILE_INIT(...)
  - #define APPL_PROFILE_CONNECT(x)
  - #define APPL_SEND_MEASUREMENT(x)
  - #define APPL_PROFILE_DISCONNECT_HANDLER(x)
  - #define GATT_DB_PROFILE_HANDLER
  - #define APPL_PROFILE_HVN_NTF_COMPLETE_HANDLER(x, y, z)
  - #define APPL_PROFILE_HVN_IND_COMPLETE_HANDLER(x, y, z)
  - #define APPL_PROFILE_MTU_UPDT_COMPLETE_HANDLER(x, y)
- Custom profile need to implement APPL_PROFILE_INIT for initializing Profiles and services application(s). APPL_PROFILE_INIT will be called from the framework during Ethermind stack initialization.
- Custom profile need to implement APPL_PROFILE_CONNECT for handling connection complete event
- Custom profile may implement APPL_SEND_MEASUREMENT for notification operation and call BT_start_timer() to setup notification interval.
- Custom profile need to implement APPL_PROFILE_DISCONNECT_HANDLER for handling disconnection complete event
- Custom profile need to implement GATT_DB_PROFILE_HANDLER for handling Characteristic Read/Write events & Characteristic Client Configuration Write event. There's a single entry point for GATT profile handler, gatt_char_handler() which is from gatt_db_pl.c under third_party\ethermind\os\freertos\, for new profile, it need to add its corresponding handler to gatt_char_handler(), like the gatt_db_amotas_handler() below:

```
#ifdef ANS
   return gatt_db_ans_handler (handle,param);
#elif defined AMOTAS
   return gatt_db_amotas_handler (handle,param);
```

```
#elif defined AMDTPS
   return gatt_db_amdtps_handler (handle,param);
```

- Custom profile might implement APPL_PROFILE_HVN_NTF_COMPLETE_HANDLER to get notified when a GATT notification is transmitted.
- Custom profile might implement APPL_PROFILE_HVN_IND_COMPLETE_HANDLER to get notified when a GATT indication is acknowledged from peer device.
- Custom profile might implement APPL_PROFILE_MTU_UPDT_COMPLETE_HANDLER to update newly negotiated MTU.
- The basic GAP APIs exposed to application for advertising, etc is defined in appl_gap.h and implemented in appl_gap.c, the following global variables are referenced by appl_fsm_handler and appl_gap module which are the foundations for peripheral and central role device respectively, they must be declared by application layer:

```
extern APPL_GAP_ADV_INFO    appl_gap_adv_table;
extern APPL_GAP_SCAN_INFO   appl_gap_scan_table;
extern APPL_GAP_CONN_INFO   appl_gap_conn_table;
```

Contact Information

Address Ambiq Micro, Inc.
6500 River Place Blvd.
Building 7, Suite 200
Austin, TX 78730
Phone +1 (512) 879-2850
Website http://www.ambiqmicro.com
General Information info@ambiqmicro.com
Sales sales@ambiqmicro.com
Technical Support support@ambiqmicro.com

Legal Information and Disclaimers

AMBIQ MICRO INTENDS FOR THE CONTENT CONTAINED IN THE DOCUMENT TO BE ACCURATE AND RELIABLE. THIS CONTENT MAY, HOWEVER, CONTAIN TECHNICAL INACCURACIES, TYPOGRAPHICAL ERRORS OR OTHER MISTAKES. AMBIQ MICRO MAY MAKE CORRECTIONS OR OTHER CHANGES TO THIS CONTENT AT ANY TIME. AMBIQ MICRO AND ITS SUPPLIERS RESERVE THE RIGHT TO MAKE CORRECTIONS, MODIFICATIONS, ENHANCEMENTS, IMPROVEMENTS AND OTHER CHANGES TO ITS PRODUCTS, PROGRAMS AND SERVICES AT ANY TIME OR TO DISCONTINUE ANY PRODUCTS, PROGRAMS, OR SERVICES WITHOUT NOTICE.

THE CONTENT IN THIS DOCUMENT IS PROVIDED "AS IS". AMBIQ MICRO AND ITS RESPECTIVE SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THIS CONTENT FOR ANY PURPOSE AND DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS CONTENT, INCLUDING BUT NOT LIMITED TO, ALL IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHT.

AMBIQ MICRO DOES NOT WARRANT OR REPRESENT THAT ANY LICENSE, EITHER EXPRESS OR IMPLIED, IS GRANTED UNDER ANY PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT OF AMBIQ MICRO COVERING OR RELATING TO THIS CONTENT OR ANY COMBINATION, MACHINE, OR PROCESS TO WHICH THIS CONTENT RELATE OR WITH WHICH THIS CONTENT MAY BE USED.

USE OF THE INFORMATION IN THIS DOCUMENT MAY REQUIRE A LICENSE FROM A THIRD PARTY UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF THAT THIRD PARTY, OR A LICENSE FROM AMBIQ MICRO UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF AMBIQ MICRO.

INFORMATION IN THIS DOCUMENT IS PROVIDED SOLELY TO ENABLE SYSTEM AND SOFTWARE IMPLEMENTERS TO USE AMBIQ MICRO PRODUCTS. THERE ARE NO EXPRESS OR IMPLIED COPYRIGHT LICENSES GRANTED HEREUNDER TO DESIGN OR FABRICATE ANY INTEGRATED CIRCUITS OR INTEGRATED CIRCUITS BASED ON THE INFORMATION IN THIS DOCUMENT. AMBIQ MICRO RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN. AMBIQ MICRO MAKES NO WARRANTY, REPRESENTATION OR GUARANTEE REGARDING THE SUITABILITY OF ITS PRODUCTS FOR ANY PARTICULAR PURPOSE, NOR DOES AMBIQ MICRO ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT, AND SPECIFICALLY DISCLAIMS ANY AND ALL LIABILITY, INCLUDING WITHOUT LIMITATION CONSEQUENTIAL OR INCIDENTAL DAMAGES. “TYPICAL” PARAMETERS WHICH MAY BE PROVIDED IN AMBIQ MICRO DATA SHEETS AND/OR SPECIFICATIONS CAN AND DO VARY IN DIFFERENT APPLICATIONS AND ACTUAL PERFORMANCE MAY VARY OVER TIME. ALL OPERATING PARAMETERS, INCLUDING “TYPICALS” MUST BE VALIDATED FOR EACH CUSTOMER APPLICATION BY CUSTOMER’S TECHNICAL EXPERTS. AMBIQ MICRO DOES NOT CONVEY ANY LICENSE UNDER NEITHER ITS PATENT RIGHTS NOR THE RIGHTS OF OTHERS. AMBIQ MICRO PRODUCTS ARE NOT DESIGNED, INTENDED, OR AUTHORIZED FOR USE AS COMPONENTS IN SYSTEMS INTENDED FOR SURGICAL IMPLANT INTO THE BODY, OR OTHER APPLICATIONS INTENDED TO SUPPORT OR SUSTAIN LIFE, OR FOR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE AMBIQ MICRO PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR. SHOULD BUYER PURCHASE OR USE AMBIQ MICRO PRODUCTS FOR ANY SUCH UNINTENDED OR UNAUTHORIZED APPLICATION, BUYER SHALL INDEMNIFY AND HOLD AMBIQ MICRO AND ITS OFFICERS, EMPLOYEES, SUBSIDIARIES, AFFILIATES, AND DISTRIBUTORS HARMLESS AGAINST ALL CLAIMS, COSTS, DAMAGES, AND EXPENSES, AND REASONABLE ATTORNEY FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PERSONAL INJURY OR DEATH ASSOCIATED WITH SUCH UNINTENDED OR UNAUTHORIZED USE, EVEN IF SUCH CLAIM ALLEGES THAT AMBIQ MICRO WAS NEGLIGENT REGARDING THE DESIGN OR MANUFACTURE OF THE PART.