USER'S GUIDE

AMDTP Example

Ultra-Low Power Apollo SoC Family

A-SOCAP3-UGGA03EN v1.1

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.

Revision History

Revision	Date	Description
0.1	September 14, 2017	Initial release
1.0	March 8, 2022	Updated template
1.1	January 3, 2023	Updated document part number

Reference Documents

Document ID	Description

Table of Contents

List of Tables

Service Declaration

The service UUID of Ambiq DTP (Data Transfer Protocol) service is defined as below 00002760-08C2-11E1-9073-0E8AC72E1011.

> **NOTE:** Base UUID of Bluetooth SIG is 00000000-0000-1000-8000-00805F9B34FB.
>
> https://www.bluetooth.com/specifications/assigned-numbers/service-discovery

1.1　Service Characteristics Definitions

Rx: 00002760-08C2-11E1-9073-0E8AC72E0011
Tx: 00002760-08C2-11E1-9073-0E8AC72E0012
ACK/Control: 00002760-08C2-11E1-9073-0E8AC72E0013

Table 1-1: Service Characteristics Definitions

Characteristic	Requirements	Mandatory Properties	Security Permissions	Description
Characteristic Rx	M	Write	None	Data from client
Characteristic Rx User Description	O	Read	None	Value read by client
Characteristic Tx	M	Notify	None	Value notification to client
Characteristic Tx Client Characteristic Configuration descriptor	M	Read/Write	None	Value notification configuration

Table 1-1: Service Characteristics Definitions *(Continued)*

Characteristic	Requirements	Mandatory Properties	Security Permissions	Description
Characteristic ACK	M	Write/Notify	None	ACK/Control to client
Characteristic ACK Client Characteristic Configuration descriptor	M	Read/Write	None	ACK notification configuration

1.2   Characteristics

The following characteristics are defined in the AM DTP Service. Only one instance of each characteristic is permitted within this service.

Table 1-2: AM DTP Service Characteristics

Characteristic Name	Mandatory Properties	Security Permission
Characteristic Rx	Write Command	None
Characteristic Tx	Notify	None
Characteristic ACK	Write Command/Notify	None

1.3   Characteristic Descriptors

1.3.1   Characteristic User Description

This characteristic descriptor defines the AM DTP version with read permission property.

1.3.2   Client Characteristic Configuration Descriptor

The notification characteristic will start to notify if the value of the CCCD (Client Characteristic Configuration Descriptor) is set to 0x0001 by client. The send data characteristic will stop notifying if the value of the CCCD is set to 0x0000 by client.

Service Behaviors

This section describes the service behaviors:

1. Either server or client may initiate data transfer.

2. Client enables notification of Tx over the CCCD upon connection establishment.

3. Client enables notification of ACK over its CCCD upon connection establishment.

4. Server to Client transmission (ACK mechanism enabled):

   a. Server starts data transmission by sending data packet to the client via notification (Characteristic Tx).

   b. Client response with acknowledgment by writing to ACK characteristic.

   c. Upon the ACK, Server behaviors are as following:

      i. **AMDTP_STATUS_SUCCESS** - Checksum is good. Server sends next packet.

      ii. **AMDTP_STATUS_CRC_ERROR** - Checksum is bad. Server resend current packet.

      iii. **AMDTP_STATUS_TIMEOUT** - Packet sent timeout. Server resend current packet.

5. Client to Server transmission (ACK mechanism enabled):

   a. Client starts data transmission by sending data data packet to the server via writing to the RX characteristic (Characteristic Rx).

   b. Server response with acknowledgment by ACK notification (Characteristic ACK).

   c. Upon the ACK, Client behaviors are as following:

      i. **AMDTP_STATUS_SUCCESS** - Checksum is good. Client sends next packet.

      ii. **AMDTP_STATUS_CRC_ERROR** - Checksum is bad. Client resend current packet.

      iii. **AMDTP_STATUS_TIMEOUT** - Packet sent timeout. Client resend current packet.

2.1 AMDTP Packet Definition

2.1.1 AMDTP Packet Format

Length: 2 bytes (data + checksum)
Header: 2 bytes
Data: 0 ~ 512 bytes
Checksum (CRC32): 4 bytes (Header and length are excluded, only data part is calculated)

Table 2-1: AMDTP Packet Format

Length	Header	Data	Checksum (CRC32)
2 bytes	2 bytes	0 ~ 512 bytes	4 bytes

Table 2-2: AMDTP Header Format

Bit15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pkt Type				Serial Number				Enc	Ack			RFU			

Serial Number: Packet serial number
Enc: Encryption enabled
Ack: Ack mechanism enabled

2.1.2 Packet Types

```
typedef enum eAmdtpPktType
{
    AMDTP_PKT_TYPE_UNKNOWN,
    AMDTP_ PKT_TYPE _DATA,
    AMDTP_ PKT_TYPE _ACK,
    AMDTP_PKT_TYPE_CONTROL,
    AMDTP_ PKT_TYPE_MAX
}eAmdtpPktType_t;
```

Data Packet
Prefix : 2 bytes length + 2 bytes header
Data : 0 ~ 512 bytes
Checksum : 4 bytes

ACK Packet
Prefix : 2 bytes length + 2 bytes header
Status : 1 byte
Checksum : 4 bytes

CONTROL Packet
Prefix: 2 bytes length + 2 bytes header
Status: 1 byte
Serial Number: 1 byte
Checksum: 4 bytes

2.1.3   AMDTP Status Code

```
typedef enum
{
    AMDTP_STATUS_SUCCESS,
    AMDTP_STATUS_CRC_ERROR,
    AMDTP_STATUS_INVALID_METADATA_INFO,
    AMDTP_STATUS_INVALID_PKT_LENGTH,
    AMDTP_STATUS_INSUFFICIENT_BUFFER,
    AMDTP_STATUS_UNKNOWN_ERROR,
    AMDTP_STATUS_BUSY,
    AMDTP_STATUS_NOTIFY_DISABLED,
    AMDTP_STATUS_TX_NOT_READY,
    AMDTP_STATUS_RESEND_REPLY,
    AMDTP_STATUS_RECEIVE_CONTINUE,
    AMDTP_STATUS_RECEIVE_DONE,
    AMDTP_STATUS_MAX
}eAmdtpStatus;
```

2.1.4   AMDTP Fragmentation and Reassemble

The maximum transmit unit size in ATT layer can be different from various products which introduced a limitation to the maximum payload size of a notification packet in Bluetooth Low Energy. In order to overcome above limitation, we have implemented an AMDTP packet fragmentation and reassemble mechanism in AMDTP service. The user can configure the maximum AMDTP packet size for fitting different applications. When transmitting, an AMDTP packet will be fragmented into maximum link layer MTU size and will be sent from the length field to the CRC filed in an AMDTP packet. The receiver side will check the whole AMDTP packet is received based on the AMDTP packet length information and does a CRC check for AMDTP packet correctness.

2.1.5   AMDTP Data Deliver Reliability

An ACK mechanism is added into AMDTP profile level for the data deliver reliability. Figure 2-1 AMDTP Packet Transfer Flowchart shows the communications between sender and receiver.

Figure 2-1: AMDTP Packet Transfer Flowchart
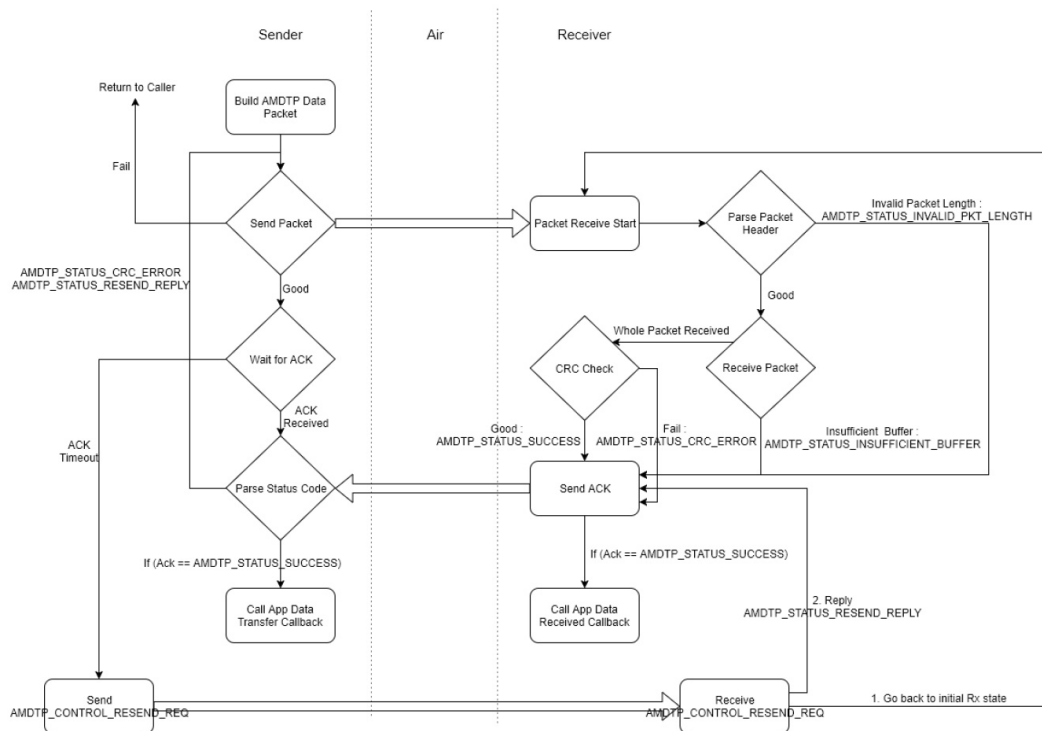
2.2   AMDTP Integration with Applications

Below are the procedures to add AMDTP profile into an example that uses Cordio Bluetooth Low Energy stack.

1. Add below files into project

   amdtp_main.c
   amdtp_common.c
   amdtps_main.c
   svc_amdtp.c

2. Add below paths into "include" folder

   sdk_root/ambiq_ble/apps/amdtps
   sdk_root/ambiq_ble/profiles/amdtpcommon
   sdk_root/ambiq_ble/profiles/amdtps
   sdk_root/ambiq_ble/services

3. In application handler initialization function (e.g., **AmdtpHandlerInit()**", call below function to initialize AMDTP server. Two callback functions **amdtpDtpRe-cvCback()** and **amdtpDtpTransCback()** need to be created in the project
   amdtps_init(handlerId, (AmdtpsCfg_t *) &amdtpAmdtpsCfg, amdtpDt-pRecvCback, amdtpDtpTransCback);

4. Add **AMDTPS_TX_CH_CCC_HDL** and **AMDTPS_ACK_CH_CCC_HDL** to CCC set

5. Call function `amdtps_start()` or `amdtps_stop()` when `AMDTP_AMDTPS_TX_C-CC_IDX` value changed

6. Call function `amdtps_proc_msg()` in the message process function for below messages:
    AMDTP_TIMER_IND
    ATTS_HANDLE_VALUE_CNF
    DM_CONN_OPEN_IND
    DM_CONN_CLOSE_IND
    DM_CONN_UPDATE_IND

7. Call below functions when add the characteristics
    `SvcAmdtpsCbackRegister(NULL, amdtps_write_cback);`
    `SvcAmdtpsAddGroup();`

3

Hands on AMDTP Example

3.1   AMDTP Server

1. Program AMDTPS project into the development board.

2. Reset the board and it will start to advertise automatically.

3. Debug output is going through SWO.

3.2   AMDTP Client

1. Program AMDTPC project into the development board.

2. Debug output is going through SWO.

3. Start a COM terminal tool (e.g., Tera Term) and connects to the board.
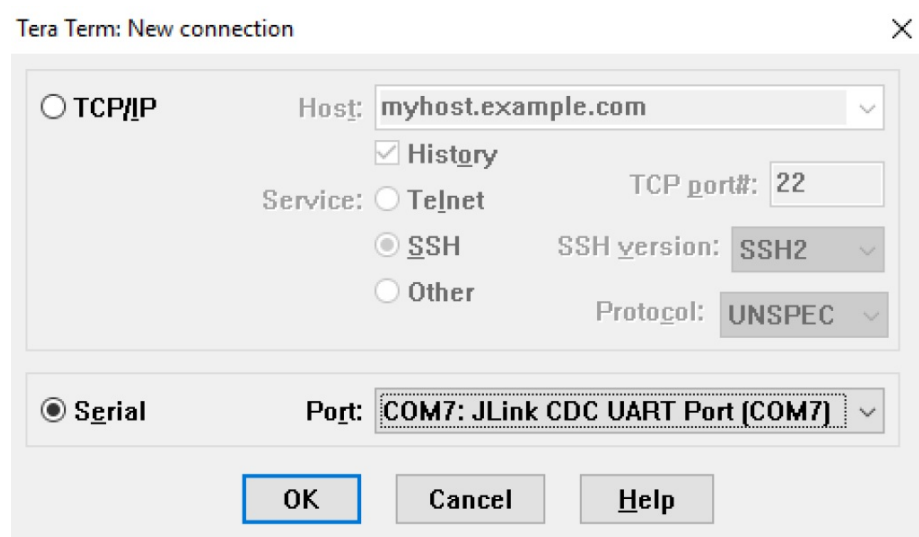
4. Click **Setup**, then select **Serial** port.

5. Change **Baud rate** to **115200**, then click **OK**.

6. Click **Setup**, then select **Terminal**.

7. Select **CR+LF** in the **Transmit** drop-down option.

8. Select the **Local echo** check box, then click **OK**.
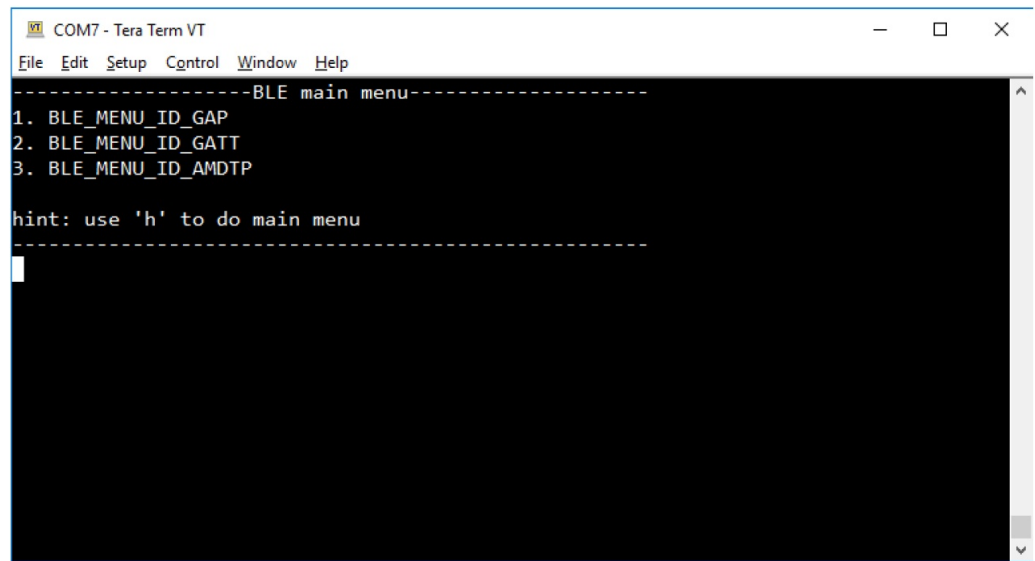
After resetting the board, you should see the output in terminal.
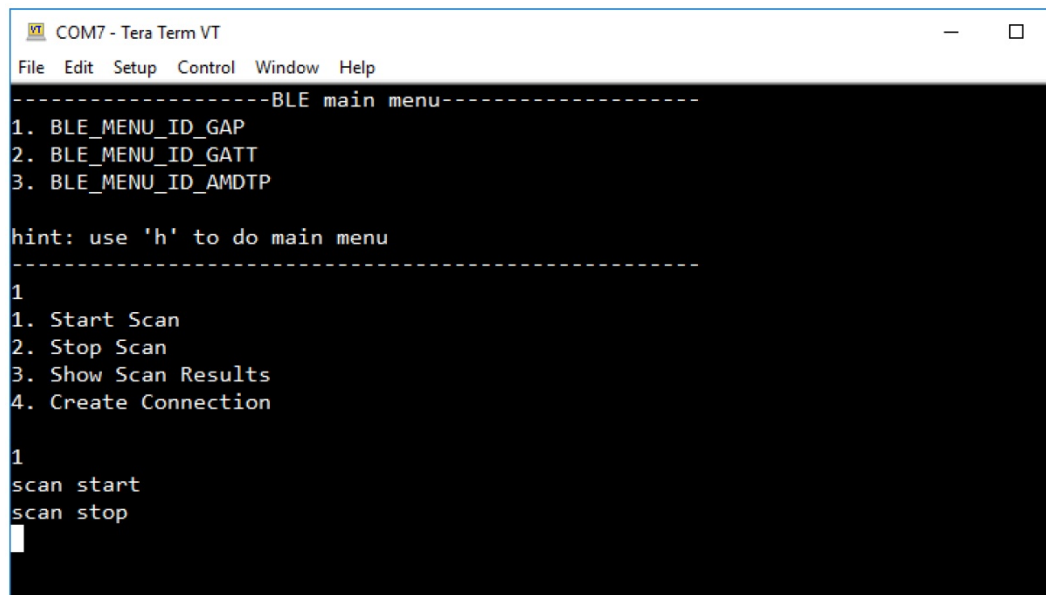
9. Complete the following to create connection with AMDTPS:

    a. Input **1** and press **ENTER** to go into `BLE_MENU_ID_GAP`.

    b. Input **1** again to **Start scan** and wait for 5 seconds for the scan to complete.

        Hint: Observe AMDTP client activities from the J-link SWO output while operating.


10. After the scan completes, input **4** to **Create Connection**.

    Note: Scan results will popped as shown below. The first number in the list is the index and followed by BD address type and BD address.

11. Input the target index to be connected to. A **Connection opened** message will show up after connecting to target device.

12. Input **h** to go back to root menu.

13. Input **3** to go into `BLE_MENU_ID_AMDTP` and it provides 4 commands to interact with AMDTP Server.