User Manual

DA1468x Serial Port Service Reference Application

UM-B-084

Abstract

This document describes the architecture and the implementation details of the Dialog Bluetooth® Smart Serial Port Service application running on the DA1468x development kit and on Android or iOS.

DA1468x Serial Port Service Reference Application

Contents

Figures

Tables

1 Terms and Definitions

BD	Bluetooth® Device
BLE	Bluetooth® Low Energy (Bluetooth Smart)
CCCD	Client Characteristic Configuration Descriptor
DK	Development Kit
DSPS	Dialog Serial Port Service
FSM	Finite State Machine
GAP	Generic Access Profile
GATT	Generic Attribute profile
HWM	High Watermark
LWM	Low Watermark
MTU	Maximum Transmission Unit
SDK	Software Development Kit
SPS	Serial Port Service
UART	Universal Asynchronous Receiver/Transmitter
UUID	Universally Unique Identifier

2 References

[1] UM-B-044, DA1468x Software Platform Reference User manual, Dialog Semiconductor.

[2] UM-B-056, DA1468x Software Developer's Guide, User manual, Dialog Semiconductor.

[3] UM-B-066, DA1468x Development Kit - Basic, User manual, Dialog Semiconductor.

[4] UM-B-060, DA1468x/DA1510x Development Kit - Pro, User manual, Dialog Semiconductor.

[5] Bluetooth, S. I. G. (2010). Bluetooth core specification version 4.2. Specification of the Bluetooth System.

[6] UM-B-047, DA1468x Getting Started with the Development Kit, User manual, Dialog Semiconductor.

[7] AN-B-041, DA14681 Dev. kit - Basic: UART with HW handshake, Application note, Dialog Semiconductor.

3 Introduction

The Dialog Serial Port Service (DSPS) emulates a serial cable communication. It provides a simple substitute for RS-232 connections, including the familiar software flow control logic via Bluetooth® Smart. The DSPS software distribution includes the application and profile source codes. It is based on a patched version of the 68x SDK1.0.12.

Software has been developed for the DA1468x Development Kit – Pro, DA1468x Development Kit - Basic and for Android / iOS tablets or phones, allowing a serial port to be emulated using two DA1468x DKs or using a DA1468x DK and an Android / iOS device. The DA1468x DK can either function in the GAP central role or the peripheral role. The Android / iOS device only functions in the GAP central role.

The application on the central device automatically connects to a peripheral device with the fixed BD address. It also handles connection loss by stopping the flow of data and automatically trying to re-establish a connection to the same device. After connection is established, both ends start reading from UART and ready to send data over the BLE through the message queue. Meanwhile, they are ready to receive data from BLE and write data to the UART through the message queue. Both central and peripheral devices can operate either in Active mode or Extended Sleep mode.

To get familiar with the DA1468x's software and hardware, the reader could consult the DA1468x Software Developer's Guide [2], the Software Platform Reference [1], the Development Kit – Basic user manual [3], and the Development Kit – Pro user manual [4] documents.

The Android / iOS application, which is described later in Section 6, scans for BLE peripheral devices that are advertising with DSPS UUID and displays them in a scan list. The user can select the peripheral device to connect to the Android / iOS device. After successful SPS service discovery, they are ready to exchange data over the air.

With the power of data length extension feature from BLE specification v4.2 [5], 80 kByte/s throughput can be achieved between two Pro DA1468x DKs under optimal conditions and settings.

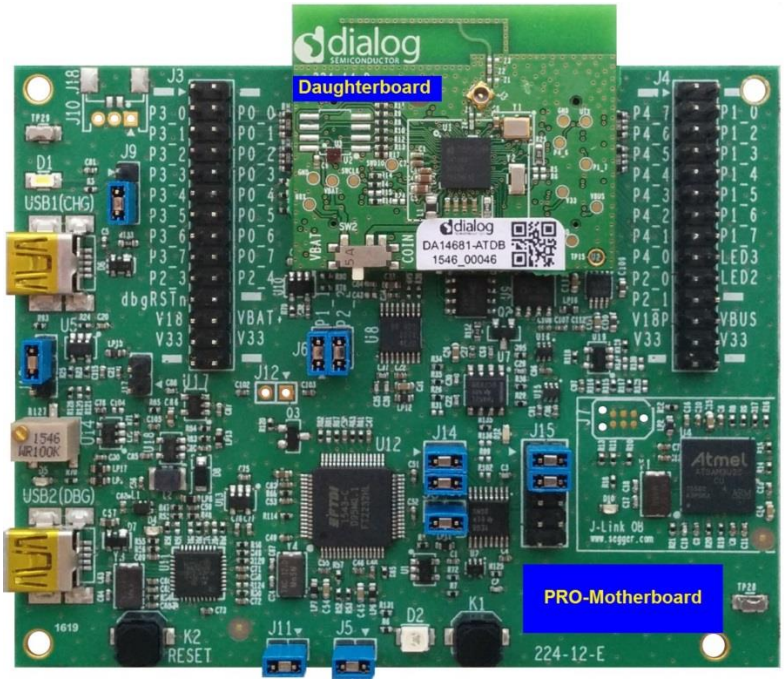

Figure 1 DA1468x Development Kit - Pro

4 Application Feature List

This section lists the advanced software features of the Dialog SPS reference application.

4.1 Bluetooth

- Two projects for the GAP Central/Peripheral role
- GATT-based bidirectional serial link
- Write without Response/Notification methods for data streaming.
- Bluetooth flow control supported.
- Data length extension feature supported
- Single point-to-point connection
- Automatic reconnection in case of link loss

4.2 UART

- Hardware and Software flow control are supported.
- Binary data transfer supported in hardware flow control mode
- UART baud rates: 921600, 460800, 230400, 115200, 57600, 38400, 19200, 9600, 4800, 2400[1]

4.3 System

- Compatible with the existing DSPS iOS/ Android application.
- Extended sleep mode and external GPIO or CTS wakeup
- Maximum 80 kByte/s[2] data throughput between two Pro DA1468x DKs
- SEGGER RTT Viewer is used to print logs through JTAG
- Peripheral application supports software update over the air (SUOTA)

[1] For Basic DK, maximum supported UART (through JLink CDC) baud rate is 115200.

[2] This throughput result is achieved under optimal settings and conditions which will be explained in a later section.

5 Application Architecture

The DA1468x SPS application contains two projects for the central and peripheral roles respectively. The project `ble_sps_central` implements the central role, operating as SPS client. The project `ble_sps_peripheral` implements the peripheral role, operating as SPS server. One BLE task and two UART tasks are defined and scheduled seamlessly by the FreeRTOS. The BLE task implements all BLE-related activities, such as connection establishment, service discovery, service data exchange, etc. The UART read task and UART write task are responsible for all UART-related activities. The data flow control scheme is applied across tasks to maintain the throughput and system stability. The data flow sequence diagram is presented to give a better overview.

5.1 Serial Port Service

The proprietary Dialog Serial Port Service (DSPS) is used to exchange data and flow control signals through a BLE connection. The BLE database is initialized in the server's profile. The details of the service, characteristics and descriptors are outlined in Table 1.

The service has two 250-byte characteristics for data transmission and reception, and 1-byte characteristic for the flow control. 128-bit UUIDs are used for the service and the characteristics. For each characteristic, two descriptors are followed. One is Client Characteristic Configuration to enable or disable notifications from server to client. Another descriptor is Characteristic User Description with the description value (e.g., "Server RX data").

The Serial Port Service uses a 'Write with no response' method for transmission of data from the GATT client to the GATT server and 'Notify' for the reverse path. Similar approach is used for exchange SPS flow control signals.

Table 1 Serial Port Service Details

Type	UUID	GATT Property	ATT Permission	Size (B)
Service	UUID_SPS	-	Read only	-
Characteristic	UUID_SPS_SERVER_TX	Notify	None	250
Descriptor	UUID_GATT_CLIENT_CHAR_CONFIGURATION	-	Read/Write	2
Descriptor	UUID_GATT_CHAR_USER_DESCRIPTION	-	Read	Actual
Characteristic	UUID_SPS_SERVER_RX	Write with no response	Write	250
Descriptor	UUID_GATT_CHAR_USER_DESCRIPTION	-	Read	Actual
Characteristic	UUID_SPS_FLOW_CTRL	Write with no response/Notify	Write	1
Descriptor	UUID_GATT_CLIENT_CHAR_CONFIGURATION	-	Read/Write	2
Descriptor	UUID_GATT_CHAR_USER_DESCRIPTION	-	Read	Actual

5.2 BLE Task

The Finite State Machines used for the BLE task of the central role and peripheral role is different. We will introduce them separately.

5.2.1 Central Role

The FSM used for the BLE task of the central role is shown in Figure 2. Upon the task entry, it first starts as the central role and sets the MTU size. Then it directly connects to a peripheral device with the pre-defined static BD address. After connection with the peer device, it starts Serial Port Service discovery. In the corresponding event handle, it identifies and stores handles for the service,

characteristic and descriptor. At the same time, it enables server notification by writing Client Characteristic Configuration Descriptor (CCCD). When the service browse is completed, it creates SPS message queues, initializes UART, opens SPS and UART flow and sends OS notification to UART read task. After connection establishment, MTU exchange and connection parameter update request are properly handled. When disconnection happens, it closes the UART, deletes the SPS queues and tries to reconnect to the device.

After DSPS application is ready and UART data are available, the task will receive notification for BLE transmission. If TX is not in progress and the current SPS flow is on, it reads the first message in the queue and sends transmission command to the stack. After this TX is done, it releases the message and notifies itself if there are still data in the queue.

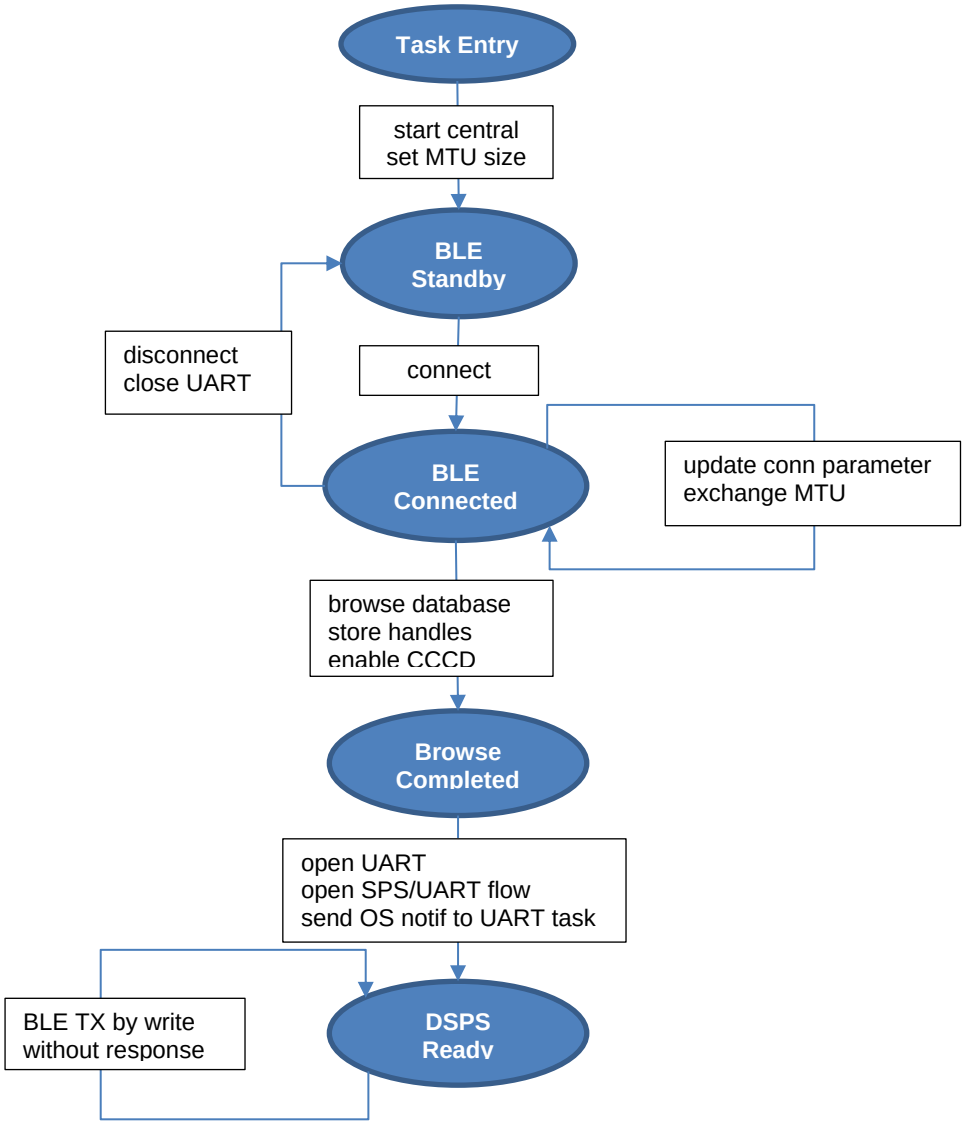

Figure 2 Central BLE Task FSM

5.2.2 Peripheral Role

The FSM used for the BLE task of the peripheral role is shown in Figure 3. Upon the task entry, it first starts as the peripheral role and sets the MTU size. Then it starts advertising with SPS UUID. After connection with the peer device, it starts the timer for connection parameter update request. Then it creates SPS message queues, initializes UART, opens SPS and UART flow and sends OS notification to UART read task. When connection parameter update is done, there follows the MTU exchange procedure. At the same time, the Serial Port Service discovery is done by the central

device. When disconnection happens, it closes the UART, deletes the SPS queues and starts advertising again. Similar operations are done to pull and transmit available UART data.

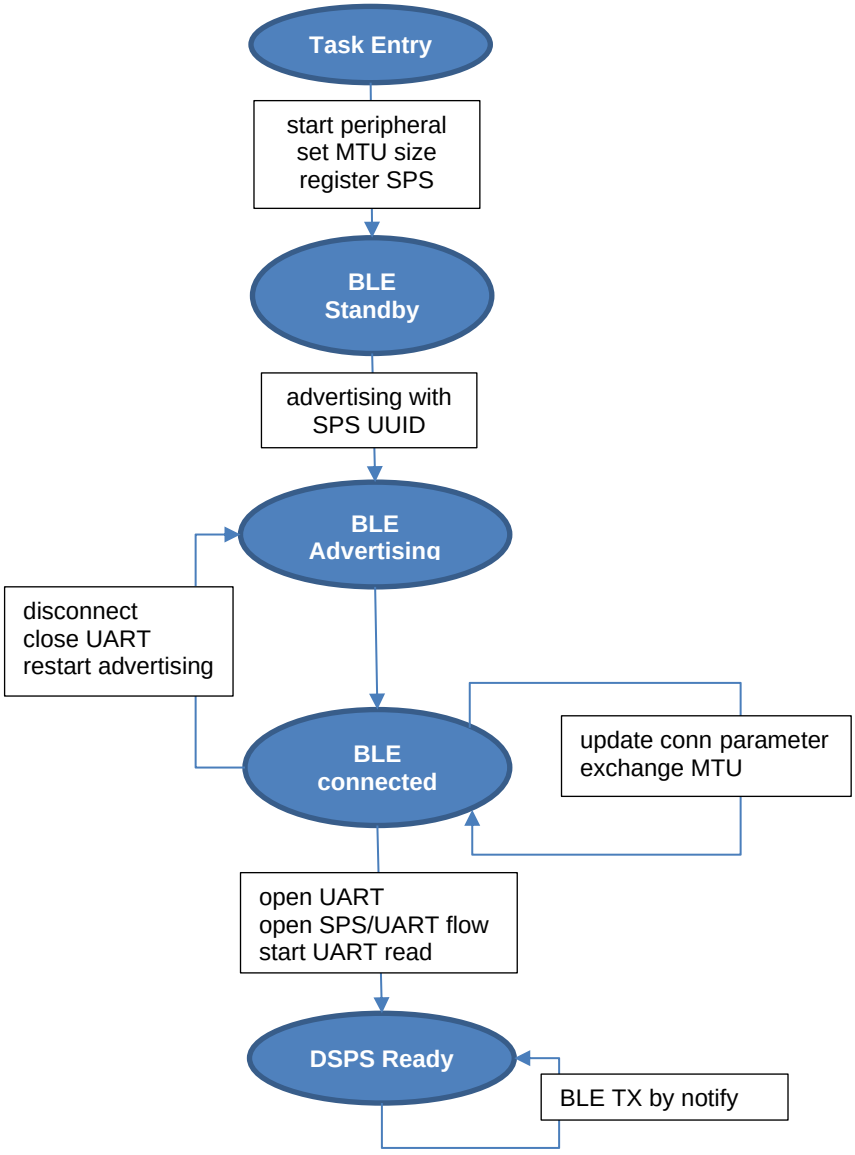

Figure 3 Peripheral BLE Task FSM

5.3 UART Task

Two UART tasks controls the UART peripheral module of DA1468x. One manages UART read activity and another one manages UART write activity. This makes sure that the write and read operation do not block each other on the application level. The application uses blocking UART transmission and reception. Because the FreeRTOS provides time-slice based scheduling, other tasks can be executed while one UART task is waiting for data transaction. In this way, the CPU resources are used efficiently. Both software (XON/XOFF) and hardware (RTS/CTS) flow control schemes are provided to control the UART data flow.

After DSPS application is ready (the master is connected and the UART flow is on), the UART read task continuously polls the UART instance for data reception with certain timeout. The expected data size is set to match the MTU size to increase the throughput. When UART data are received, they are pushed into the message queue as a message and the OS notifies the BLE task for transmission. When the UART write task gets OS notification from BLE task for received data, it pushes data into message queue and writes them to UART.

5.4 Data Flow Control

The data flow control is an important part of this application. The mechanism is implemented across tasks with SPS and UART flow control. This is illustrated in Figure 4.

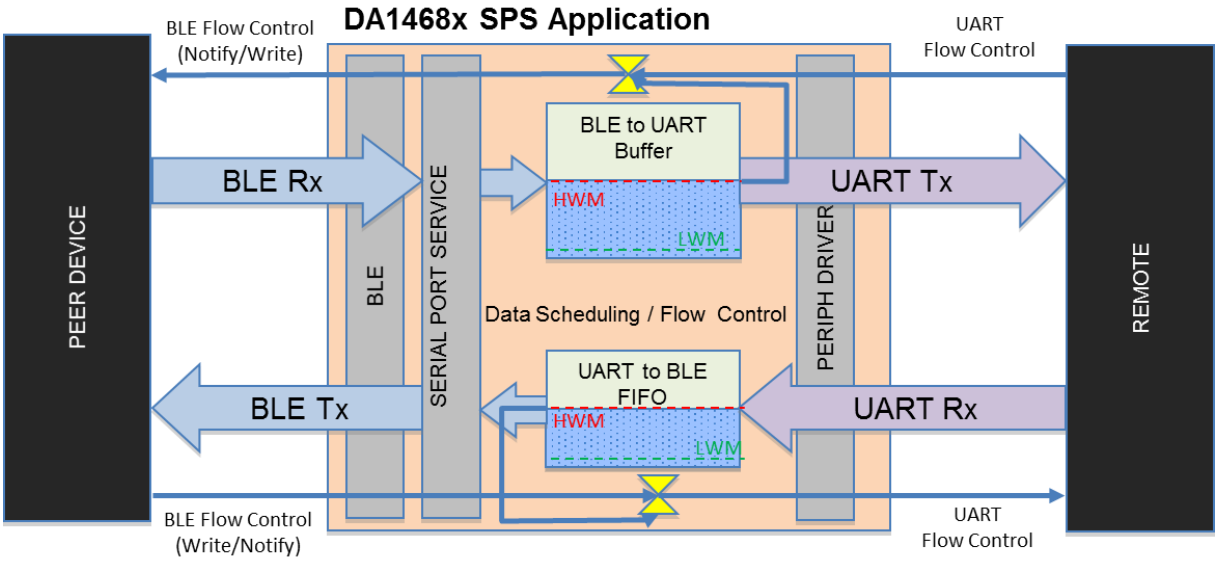

Figure 4 Data and Flow Control

Two message queues are allocated when the connection establishment and service discovery is done. The RX queue is used for BLE to UART data path and the TX queue is used for UART to BLE data path. High watermark (HWM) and low watermark (LWM) values are defined for each queue to set the buffer utilization level. Each time one message is pushed into the queue, HWM is checked for stopping the data flow. Similarly, each time one message is popped from the queue, LWM is checked for opening the data flow.

5.5 Project Files Overview

The SPS reference application is developed based on DA1468x SDK 1.0.12 with minor modifications. The detailed SDK structure explanation can be found in the Software Platform Reference manual [1]. Several files are application related. Their description is shown in Table 2.

The patched SDK files are: `hw_uart.h`, `ad_uart.h`, `ad_uart.c`, `sps.h` and `sps.c`.

Table 2 SPS Application Files

File name	Description
`.\config\ custom_config_qspi.h` `.\config\ custom_config_qspi_suota.h`	All project configuration options, regarding UART settings, BLE settings, system clocks, the execution mode, the OS heap size, retention RAM size etc. Please use another configuration file for SUOTA version.
`.\ble_services\src\sps.c` `.\ble_services\include\sps.h`	Application service and profile specific functions (Peripheral role).
`.\ble_sps_peripheral_task.c` `.\ble_sps_central_task.c`	Application task and relevant functions (Peripheral/Central role).
`.\main.c`	Application system and task initialization
`.\sps_queue.c` `.\sps_queue.h`	Application queue specific APIs

.\sps_uart.c .\sps_uart.h	Application UART specific APIs
.\sw_version.h .\mkimage.bat	SUOTA related files
.\readme.md	Simple readme file

5.6 Data Path Sequence Diagrams

In the sequence diagrams of this section, the API call sequence for data path after service discovery between UART interface and BLE is illustrated.

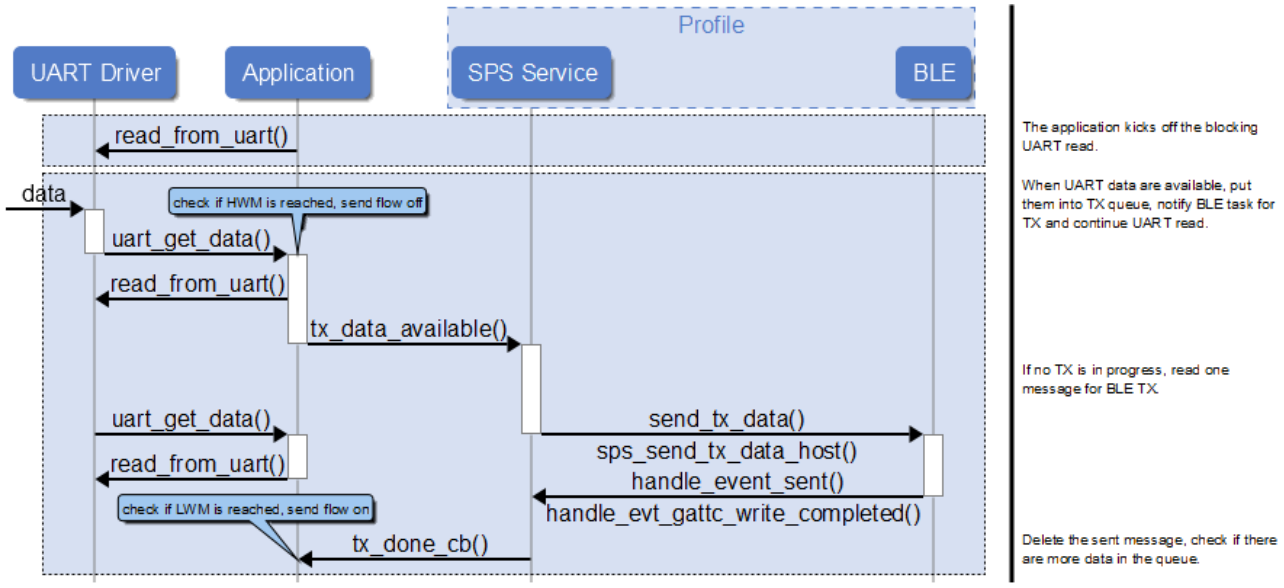

Figure 5 UART to BLE Data Transfer Sequence Diagram

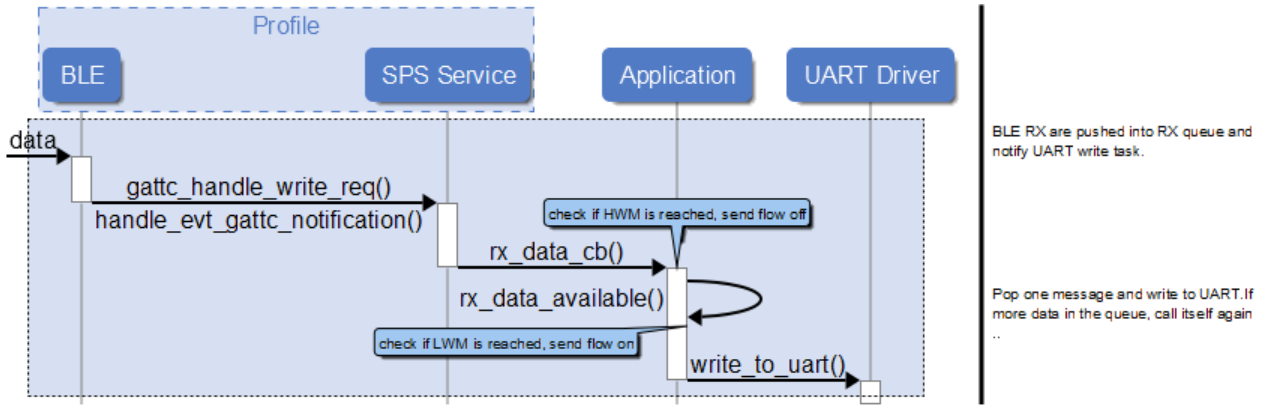

Figure 6 BLE to UART Data Transfer Sequence Diagram

5.7 Sleep and Wake-up

The DSPS application supports extended sleep mode. The device can wake up from user-defined GPIO or CTS pin. Simply define `ENABLE_SLEEP` to enable sleep mode and define customized wakeup GPIO. CTS wakeup is by default enabled in sleep mode. If the user uses proper terminal application

that assets RTS line (CTS of device) before sending data and de-asserts RTS line after data sent, the DSPS application is more power-efficient.

When the FreeRTOS is in idle mode, the Clock Power Manager (CPM) is invoked to check the sleep condition. If both UART interfaces are not busy, the API stops the UART flow and set CTS as wake-up pin. On the application layer, if there is no data in the RX/TX message queue, it sets wake-up settings for user-defined GPIO. If sleep is cancelled by one of the adapters, the wake-up settings are disabled. When the system wakes up and the XTAL16M is ready, the API restores the UART setting and opens the UART flow control.

Table 3 explains the API for the communication with the CPM. User can define their own functionality for sleep and wake-up.

Table 3 API for the Communication with the CPM

API for the communication with the CPM	Description
`bool ad_prepare_for_sleep(void)`	The CPM inquires the Adapter about whether the system can go to sleep
`void ad_sleep_canceled(void)`	If an Adapter rejects sleep, the CPM calls this function to resume any Adapters that have previously accepted it
`void ad_wake_up_ind(bool)`	The CPM informs the Adapter that the pad latches are to be removed so that it re-initializes the GPIOs that are used by the hardware resource it controls and, depending on the resource type, the resource itself
`void ad_xtal16m_ready_ind(void)`	The CPM informs the Adapter that the XTAL16M is ready and is or may be set as the system clock.

6 DSPS Android and iOS Application

6.1 Overview

The DSPS reference application comes with an SPS demo application for Android and iOS systems. The DSPS application can discover, connect and exchange data with SPS enabled devices within the Bluetooth RF range of the mobile device. The application has been tested with Android version 5 to 7 and iOS 7 to 11.

The main features are

- Device discovery
- Connection to a discovered device that supports DSPS
- Reception of data (ASCII or HEX) from a peer device with flow control
- Transmission of data (ASCII or HEX) to a peer device, either once or repeatedly
- Transmission of file to a peer device with flow control

Figure 7 DSPS Icon

The Android application can be found in Google Play Store and easily installed from there as any other android application. To find the application, user can search for “DSPS Dialog’”. The user can also install the application using the *.apk installer package file. The iOS application can be found in Apple App Store and easily installed from there like any other iOS application. To find the application, the user can search for “DSPS”. The user can also install the application using the *.ipa installer package file. The source code of iOS and Android application can be found on our support website (Products->DA14580->Software & Tools->Mobile->DSPS).

6.2 Device List

Click on the DSPS icon to start the application and it starts searching for peripheral devices which expose SPS UUID. While searching, all devices found are displayed (device name and Bluetooth address). The complete list of all discovered devices is listed on the screen when the search has finished. To refresh the list of devices found, the user can press the button “SCAN”. Click on the name of the desired device to connect to it.

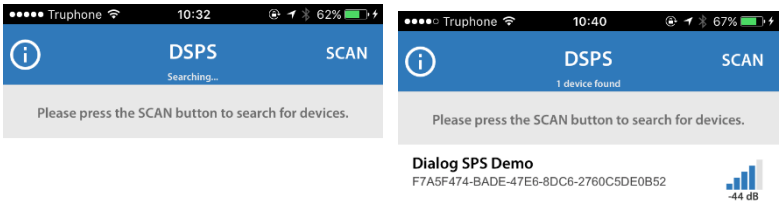

Figure 8 Search Screen and Device List

6.3 Functionality Tabs

When connected to a device, the console tab is shown to the user. The top of the screen shows information about the currently connected device. Click on the menu tab to get the sliding function menu from the left side. There you can switch functionality tabs, get information or disconnect device.

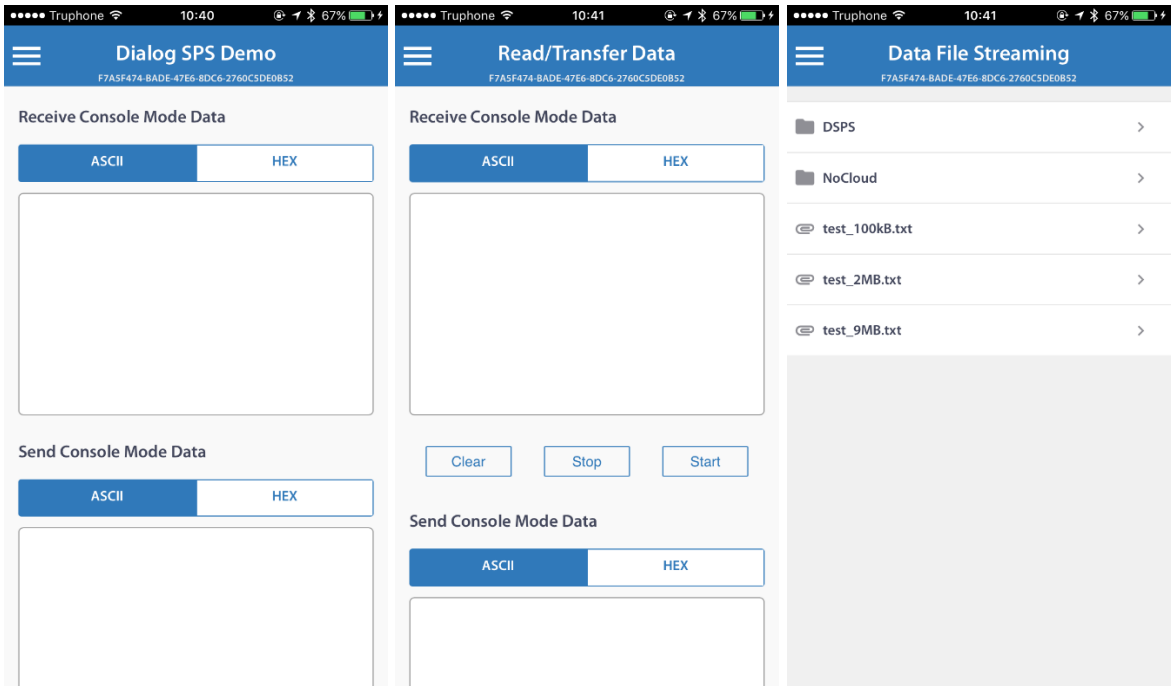

Figure 9 Application Main Screens

6.3.1 Console Tab

In console mode, any character entered in the connected UART terminal is immediately sent to the peer device and any character received is displayed in the reception display box. SPS flow needs to be open in the beginning.

6.3.2 Read/Transfer Data Tab

This tab allows the user to send data to and receive data from the peer device. The upper part shows the reception display box, where received data can be displayed in ASCII or HEX format. Also, the user can clear the received data, enable or disable the incoming flow by pressing the ‘start’ or ‘stop’ button respectively.

For the TX part, a text box is shown where data can be entered via the keyboard or pasted. Pressing the ‘send’ button will send the displayed data once or repeatedly (depending on the state of the ‘cyclic sending’ tick box) with the chosen interval.

6.3.3 Data File Streaming Tab

The user has also the ability to send a file instead of individual characters. In this tab, the user can browse the device’s file system to find the file to be transmitted. Note that the received data displayed in the box will generate a text file stored under DSPS folder automatically. This file can be sent as an echo-back test.

7 DSPS Demonstration Setups

7.1 Hardware setup for Basic DK

The Basic DK uses SEGGER JLink simulated UART interface. The maximum supported UART baud rate is 115200. To use the full UART functionality, the virtual UART port needs to be enabled for JLink interface. Please refer to application note [7] for detailed setup steps. The signals used for RTS/CTS are connected by default to the on-board LED (D2) and button (K1). So jumper J14 and J15 need to be removed and put on header J13, pins 5-6 and 7-8. The jumper placement is shown in Figure 10. But user can always use extra FTDI cable to achieve better UART performance.

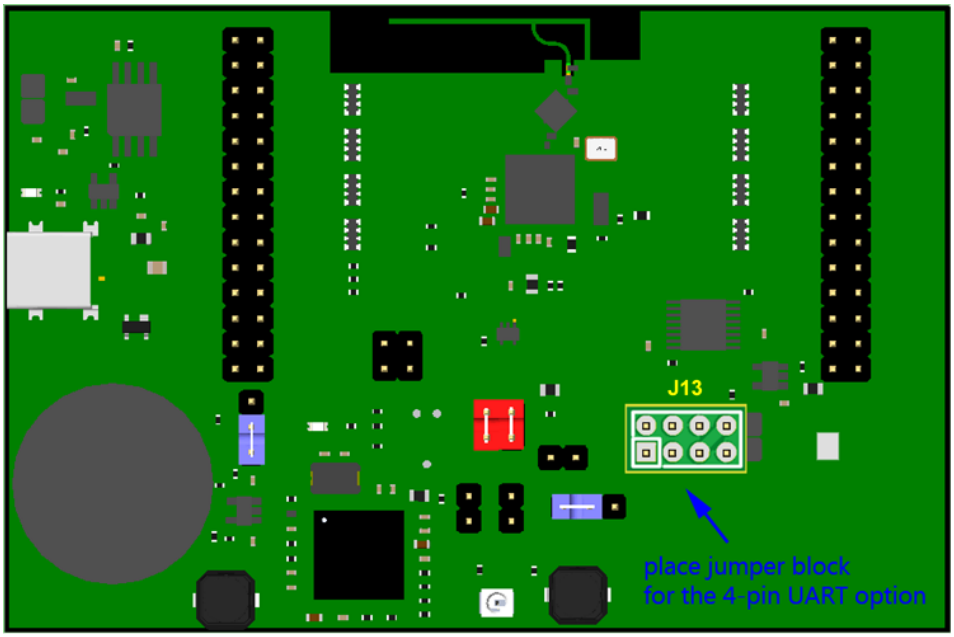

Figure 10 Basic DK Jumper Placement for Full UART Functionality

7.2 Hardware Setup for Pro DK

By default jumper settings on mother board shown in Figure 1, the Pro DK board can run the DSPS application with UART Software flow control. To use UART Hardware flow control, jumper J5 and J8 need to be removed, so GPIO P1_5 and P1_6 are not connected anymore to LED D2 and button K1. Jumper J15 needs to be mounted on 5-6 and 7-8 to connect RTS and CTS. This is shown in Figure 11. For more information about the Pro DK, please refer to [4].

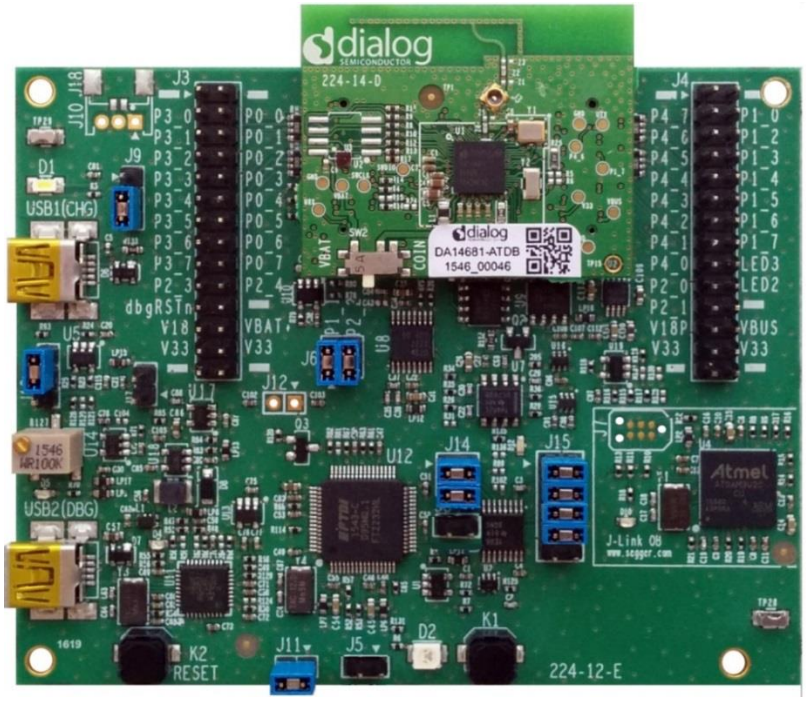

Figure 11 Pro DK Jumper Placement for Full UART Functionality

7.3 DK User Configuration

For users new to the DA1468x development environment please see the getting started guide [6]. Before getting started with the software projects, it is good to get an overview of the user configurations. Almost all important user configuration parameters are listed and explained in `[ProjectRoot]\config\custom_config_qspi.h`. The default settings should work out-of-box. User can also change them to tailor their own needs. The settings are listed in Table 4.

Table 4 User Configuration List

User Configuration	Description
`LOG_ON`	Define this to enable the log on SEGGER RTT Viewer, or printf will be empty function

`UART_TX_PORT, UART_TX_PIN`
`UART_RX_PORT, UART_RX_PIN`
`UART_RTS_PORT, UART_RTS_PIN`
`UART_CTS_PORT, UART_CTS_PIN`
	GPIO configurations for UART2

`CFG_UART_HW_FLOW_CTRL` `CFG_UART_SW_FLOW_CTRL`	Define one of them for the UART flow control mode
`CFG_UART_SPS_BAUDRATE`	Set UART2 baud rate with proper enumerations.
`ENABLE_SLEEP` `WAKEUP_PORT, WAKEUP_PIN`	Define ENABLE_SLEEP to enable sleep mode and define customized wakeup GPIO. CTS wakeup is by default enabled in sleep mode.
`DK_HIGH_THROUGHPUT` (central only)	Define this to have the optimal BLE and clock setting for high data throughput between DKs
`CUSTOM_SYS_CLK`	CPU clock option: 16, 32, 48, 96MHz
`dg_configBLE_DATA_LENGTH_RX_MAX` `dg_configBLE_DATA_LENGTH_TX_MAX`	Define BLE packet TX/RX data length: max 251 by v4.2 and max 27 by v4.0.

MTU_SIZE	MTU size.
defaultBLE_PPCP_INTERVAL_MIN defaultBLE_PPCP_INTERVAL_MAX defaultBLE_PPCP_SLAVE_LATENCY defaultBLE_PPCP_SUP_TIMEOUT	Define connection interval (min and max), slave latency and supervision timeout.
dg_configBLE_CONN_EVENT_LENGTH_MIN (central only)	Define connection event length. This is important for BLE throughput.
defaultBLE_STATIC_ADDRESS targetBLE_STATIC_ADDRESS (central only)	Define BD address of own device. For central project, also need to define the target device to connect.
TX_SPS_QUEUE_SIZE RX_SPS_QUEUE_SIZE TX_QUEUE_HWM, TX_QUEUE_LWM RX_QUEUE_HWM, RX_QUEUE_LWM (in ble_sps_central_task.c or ble_sps_peripheral_task.c file)	Set TX/RX message queue size, HWM and LWM.

7.4 DK to DK Connection Setup

1. Extract the DSPS reference application zip file and use the patched SDK1.0.12 folder (black_orca_sdk) as SmartSnippets Studio workspace.
2. Import ble_sps_peripheral, ble_sps_central and scripts projects.
3. Set the desired user configuration settings in [ProjectRoot]\config\custom_config_qspi.h.
4. Build and program both ble_sps_peripheral and ble_sps_central project on two DKs respectively.
5. Open two PC serial consoles (e.g. TeraTerm or Realterm) and select appropriate COM port for central and peripheral device. Set terminal baud rate and flow control scheme to the defined settings in [ProjectRoot]\config\custom_config_qspi.h.
6. Press RESET button on both DKs. The connection should be quickly established. Upon completion, data can be transferred bi-directionally by either sending single characters or a whole file. When sending large chunk of data, the DLE packets can be captured with proper BLE packet sniffer (i.e. Frontline Sodera LE). The screenshot of a sniffer trace is shown in Figure 12. Data throughput can be monitored with proper BLE packet sniffer, PC terminal timestamp or build-in throughput calculation function. The throughput performance is discussed in Chapter 8.

Frame#	Ch...	Access Ad...	Event Co...	Si...	LLID	PHY	NE...	SN	MD	SP	Len	Fra...	Delta
41,380	25	0x25c8cfc8	0x0840	1	Start	1M	1	1	1	No Suppln...	251	277	00:00:00.005295
41,381	25	0x25c8cfc8	0x0840	2	Empty	1M	0	1	0	No Suppln...	0	26	00:00:00.002238
41,382	25	0x25c8cfc8	0x0840	1	Start	1M	0	0	1	No Suppln...	251	277	00:00:00.000229
41,383	25	0x25c8cfc8	0x0840	2	Empty	1M	1	0	0	No Suppln...	0	26	00:00:00.002238
41,384	30	0x25c8cfc8	0x0841	1	Start	1M	1	1	1	No Suppln...	251	277	00:00:00.005295
41,385	30	0x25c8cfc8	0x0841	2	Empty	1M	0	1	0	No Suppln...	0	26	00:00:00.002238
41,386	30	0x25c8cfc8	0x0841	1	Start	1M	0	0	1	No Suppln...	251	277	00:00:00.000230
41,387	30	0x25c8cfc8	0x0841	2	Empty	1M	1	0	0	No Suppln...	0	26	00:00:00.002238

Figure 12 BLE Sniffer Trace for DLE Packet

SEGGER JLink RTT Viewer[3] is used to print information through JTAG instead of UART. To use it, find the allocated address of _SEGGER_RTT symbol in [ProjectRoot]\DA14681-01-Debug_QSPI\ble_sps_peripheral.map (or Release_QSPI folder). Fill this address in JLink RTT Viewer address field and connect to the DK. The same steps apply for ble_sps_central. By default, connection status and UART RX data throughput are displayed.

[3] This can be found in SEGGER Jlink installation path.

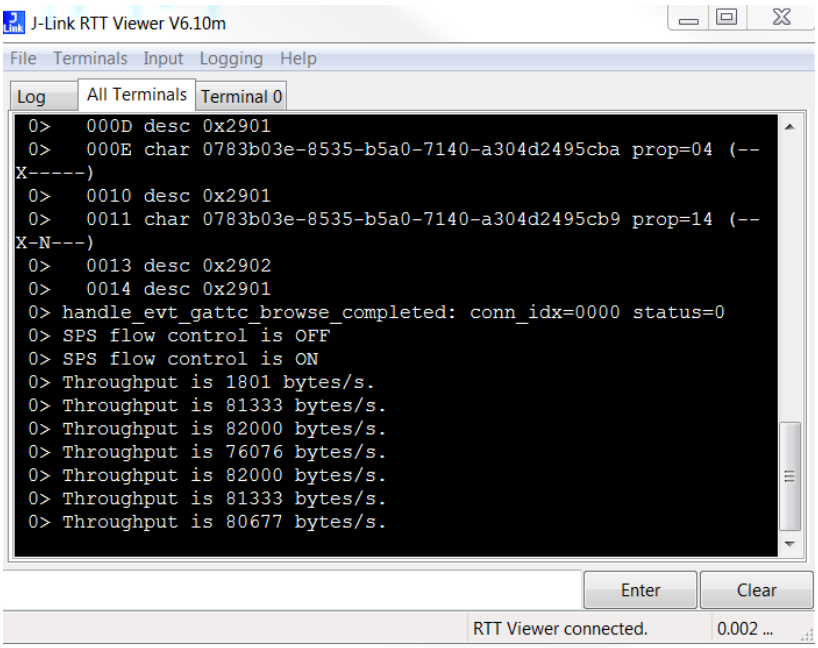

Figure 13 RTT Viewer Log (central)

7.5 Android / iOS Application to DK Connection Setup

7. Only the `ble_sps_peripheral` project is needed. Setup steps are given in previous section.
8. Install and open the DSPS Android / iOS application. Scan and connect to the device named “Dialog SPS Demo”.
9. Upon successful connection and SPS discovery, press “start” button under Read/Transfer Data tab to enable the SPS flow. Then the transmission and reception of characters can be performed between the PC terminal and the application. Data throughput can be monitored with BLE packet sniffer, PC terminal timestamp or build-in throughput calculation function.

8 DSPS Performance

8.1 Data Throughput in Theory

Because the UART baud rate can reach more than 1 Mbit/s and it is over-wire communication, the data throughput is limited by BLE communication. With the DLE feature, the maximum packet data length is 251 bytes, in which 244 bytes are application data payload. For half-duplex transmission, the time needed to transmit one max-size packet is composed of:

a. TX data packet length = 2120 us
b. T_IFS between packets = 150 us
c. RX empty packet length = 80 us
d. T_IFS between packets = 150 us

Totally 2467 us is needed to transmit the 244 bytes application data payload. This results in a theoretical data throughput of 781 kbit/s. The illustration for packet timing is shown in Figure 14.

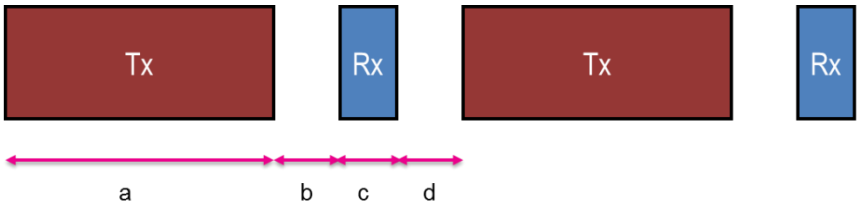

Figure 14 DLE Packet Timing

8.2 Data Throughput in Practice

To achieve such high theoretical data throughput, we need to set UART baud rate to 921600 using HW flow control with Pro DK. Using lower baud rate results in lower throughput limited by UART speed. We also set CPU speed to 96 MHz to boost the processing speed. For the optimal BLE performance between two 68x devices, connection interval is set to 30 ms and minimum connection event length is set to 25 ms. So during one connection interval, the stack tries to transmit as many packets as possible. The BLE sniffer trace timeline is shown in Figure 15 which is a good illustration for the concept. It is noticed that increasing connection event length further results in throughput fluctuation or degradation. For full-duplex communication, the throughput is reasonably reduced.

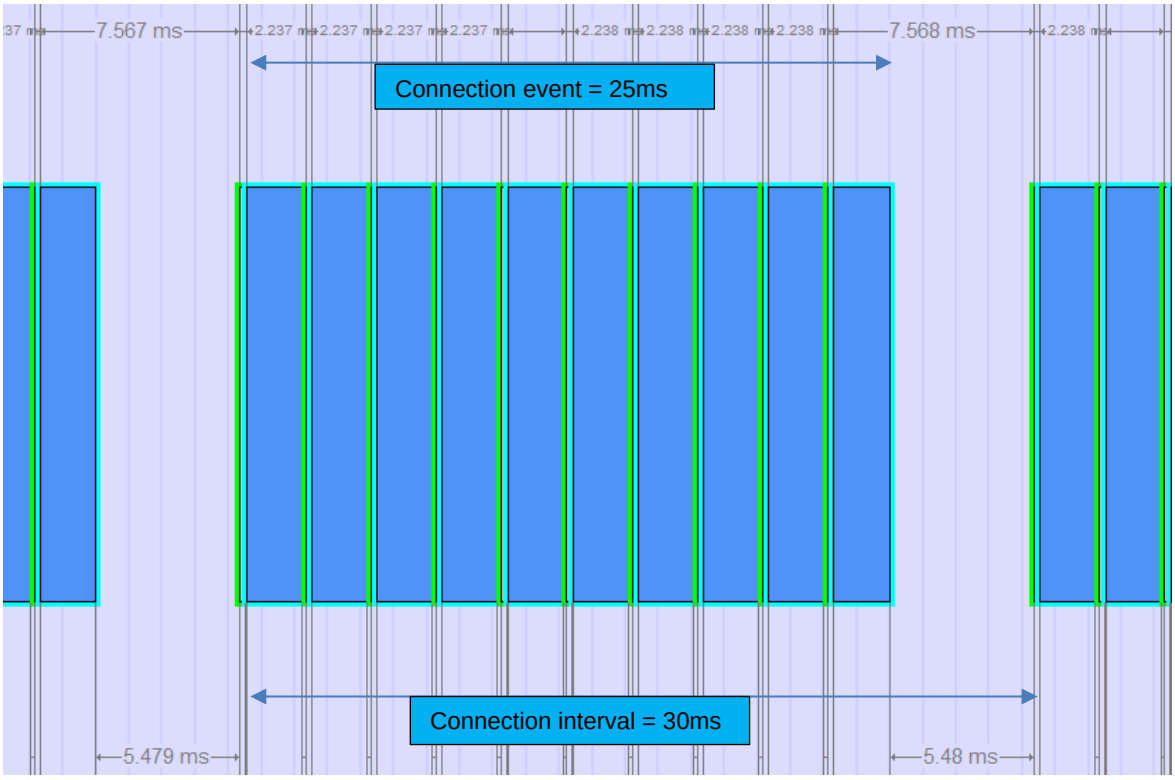

Figure 15 BLE Sniffer Trace Timeline

For iOS and Android smartphone, lower throughput is achieved due to smartphone OS limitation or application implementation. The performance results are shown in Table 5.

Table 5 Performance Results (optimal settings)

Host			Google Pixel	iPhone 7	DA1468x (with DLE)	DA1468x (without DLE)
Connection parameters	**MTU**	octets	247	247	247	247
	Connection Interval	ms	30	30	30	30
	Connection event length	ms	-	-	25	25
Throughput measurements	**Half Duplex**					
	Central Tx	kB/s	9.8	3.5	79.8	19.2
		kbit/s	78.4	28	638.4	153.6
	Peripheral Tx	kB/s	56.3	20.1	79.2	18.7
		kbit/s	450.4	160.8	633.6	149.6
	Full Duplex					
	Central Tx	kB/s	-	-	45.9	19.1
		kbit/s	-	-	367.2	152.8
	Peripheral Tx	kB/s	-	-	42.6	18.6
		kbit/s	-	-	340.8	148.8

9 Known Limitations

- UART read under SW flow control sees data loss under high baud rate

Without the "auto flow control" mechanism used by HW flow control, data loss is seen for UART baud rate larger than 115200 when sending a file from terminal to 68x device. The larger the baud rate is, the more the data loss is. This is only seen when HW flow control is not used.

- RX queue full assertion

Each time after SPS flow off, certain number of on-the-fly packets will still come. This number can be 5-30 depends on the UART speed. This causes RX queue full assertion in several corner cases. It is suggested to increase RX queue size or reduce high-water-mark.

- Full-duplex transmission (CPU = 16MHz) causes memory shortage

Because of insufficient CPU processing speed for packets and messages, assertion of OS or BLE stack memory allocation failed could happen. Reducing BLE and UART speed or increasing CPU speed will help.

- PC UART terminal prevents sleep mode

Many PC UART terminal always asserts RTS when running. In this case, the 68x device cannot enter sleep mode because of the sleep check implementation. User needs to find proper terminal software or write own terminal RTS/CTS control software.

- DSPS mobile APP to 68x data transfer has low throughput

Delay is added when sending data block from APP to 68x device for safety reason, which lowers the throughput. Decreasing or removing the delay is possible, but there is risk to crash the smartphone BLE stack.

- MTU size should be larger than 131 bytes

Due to the data block fragmentation mechanism in the current DSPS APP (version 3.220), transferring file from mobile to 68x may not work properly when MTU size is smaller than 131 bytes.

Revision History

Revision	Date	Description
1.0	27-Feb-2018	Initial version.
1.1	17-Sep-2018	Correct TX data packet time
1.2	24-Jan-2022	Updated logo, disclaimer, copyright.

Status Definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.