Bluetooth Data Demonstration 4 with SPP
Demonstration ReadMe

1.1 DESCRIPTION

Note: This ReadMe provides detailed demonstration information. For additional information, please consult the companion document, “PIC32 Bluetooth Audio Development Kit Reference Guide” (DS70005140). This reference guide is available for download from www.microchip.com.

Bluetooth data support only for Windows and Android mobile devices. Bluetooth Stack with SPP and Bootloader.

Note: Apple iOS support is also available. This support is restricted to Apple MFi licensees only. Please contact a Microchip sales office for information.

Refer to Table 3-1 in Chapter 3. “Interoperability Testing Results” of the “PIC32 Bluetooth Audio Development Kit Reference Guide” (DS70005140) for the list of tested Bluetooth-enabled devices.

In this demonstration, the basic Bluetooth audio profiles and decoders have been removed, (i.e., A2DP, AVRCP, SBC and/or AAC). This demonstration provides basic data transport of non-audio full-duplex data transfers over the Bluetooth link. This demonstration does not provide any USB audio support. The demonstration allows a user to perform terminal emulation and echo characters from an Android smartphone or PC over a Bluetooth connection to the graphic display of the development board, and then back to the PC or smartphone emulation application menu. This demonstration also supports the capability to write binary images into external memory over SPP for use by the PIC32 bootloader.

1.2 BASIC FUNCTIONALITY

1.2.1 Bluetooth Module

The PIC32 Bluetooth Audio Development Kit provides hardware support for the BlueCore® CSR8811™ and the RDA Microelectronics RDA5876 through compile-time switches.

1.2.1.1 CSR8811

The CSR8811 is a single-chip radio and baseband IC for Bluetooth 2.4 GHz systems including Enhanced Data Rate (EDR) to 3 Mbps and Bluetooth low energy. The CSR8811 supports Bluetooth Class 1 transmission, and supports multiple device connection. The PIC32 Bluetooth Audio Development Kit uses a module based on the CSR8811 radio in its default configuration (see Note). Software projects using the default board configuration should select the CSR8811 configuration in MPLAB X IDE.

Note: The CSR8811 daughter board is included in the PIC32 Bluetooth Audio Development Kit.
1.2.1.2 RDA5876

The RDA5876, integrates Bluetooth and a FM radio tuner into one device and is optimized for mobile applications. Bluetooth and FM can work simultaneously and independently, with low-power consumption levels targeted to battery powered devices. For the highest integration level, the required board space has been minimized and customer cost has been reduced. The RDA5876 meets Class 2 and Class 3 transmitting power requirements. The RDA5876 radio solution is a low-cost alternative for single point Bluetooth audio and data applications.

**Note:** To request an RDA5876 daughter board, please contact your local Microchip sales office.

1.2.1.3 SELECTING THE CONFIGURATION

A compile-time switch in the application code provides software support for the either the CSR8811 or the RDA5876. This can be done by selecting the desired configuration during compilation, as shown in the following figure.

**FIGURE 1-1: CONFIGURATION SELECTION**

1.2.2 Connections

If the Bluetooth device is connected and paired but the user walks out of Bluetooth range, by default, the system will continuously search for the last Bluetooth connection source (see Section 1.2.6 “Connection Retry Time Limit”). Then, when and if, the Bluetooth-enabled device comes back into range, it will automatically reconnect. If the smartphone or Bluetooth-enabled device, while still in range, disables it’s Bluetooth connectivity, the Bluetooth Audio Development Board Bluetooth software will not attempt to automatically reconnect with the device.

1.2.3 Bluetooth Device IDs

The Bluetooth software remembers and stores in Flash memory the last 10 unique Bluetooth device IDs to which it successfully paired to facilitate faster automatic reconnection when there is no currently active Bluetooth connection. If Bluetooth is turned OFF on a user smartphone that is currently connected and re-enabled later, it will automatically reconnect if in range or when it comes back into range. In addition, when the Bluetooth Audio Development Board is powered on, the Bluetooth software will automatically pair and connect to the last Bluetooth-enabled device, assuming it is still active; otherwise, it will search for the next most recently connected device in the list and repeat.

1.2.4 Bluetooth Pair/Unpair

If the user presses and holds SW1, which forces Bluetooth to unpair, the user must manually force their smartphone to “forget” the Bluetooth demonstration name of the development kit to enable their smartphone to rediscover and subsequently re-pair with the development kit. If the user selects SW2 (Bluetooth disconnect), the user does not need to force their smartphone to forget the demonstration name of the development kit and can reconnect at will.

---

**Note:** To request an RDA5876 daughter board, please contact your local Microchip sales office.
1.2.5 Bluetooth Device Address

By default, when the development kit is powered on for the first time, it generates a one-time random unique Bluetooth Device Address for any given development kit hardware. Optionally, at design time, the user can specify a Bluetooth Device Address in the application code of the development kit.

The device address is a six byte hexadecimal value. The macro, `BT_DEVICE_ID`, defines the first 4 bytes of the hexadecimal value and `BTDEVICE_ID_2LSB` defines the last 2 bytes of the hexadecimal value. The last two bytes of the device address can be randomized by enabling `BTDEVICE_ID_2LSBRANDOMIZE`. These macros are defined in `HardwareProfile_PIC32_Bluetooth_Audio_Development_Board.h`.

Setting a specific hard-coded device address is not recommended during the design and development state, as Bluetooth connection problems may be experienced if another development board with the same Bluetooth Device Address is within range.

1.2.6 Connection Retry Time Limit

Starting with demonstration v2.0, a new feature enables a connection retry time limit. The limit will define a set period (in approximate seconds) that the unit will continue to retry to connect to the Bluetooth device from which it has lost a connection. After this period, the device will discontinue trying to automatically connect. However, the device can still manually establish a previously paired connection, or form a new pair as previously stated. The feature can be enabled in the `user_config.h` file.

1.2.7 Bootloader

The BT_Data_Demo_4 supports the PIC32 bootloader data interface. The PIC32 bootloader is a specialized program that is capable of writing a new program to the device without the use of a physical memory programmer. This functionality is supported through the use of the Bluetooth SPP protocol to receive program data.

| Note: | This configuration is only supported by the Revision 3 PIC32 Bluetooth Audio Development Kit. |
1.3 BLUETOOTH DATA DEMONSTRATION SETUP

Download a terminal emulator echo application. An Android application can be obtained by visiting:

O&hl=en

1.4 RUNNING BLUETOOTH DATA ANDROID DEMONSTRATION 4

1. Program the device with the hex file, BT Data Demo 4.hex for the CSR8811_BT_RADIO or RDA5876_BT_RADIO Bluetooth controller. This can be done by selecting the hardware support from the drop down box during compilation, as shown in the following figure.

2. Enable Bluetooth on the Android device.
3. Open the Android terminal emulator application.
4. Connect and pair the Android device with the Bluetooth Audio Development Board to the device named BT Data Demo 4.
5. If prompted by your Bluetooth device for a PIN, enter 0000. The device should connect and pair, and indicate as such on the display, as shown in the following figure.

6. After pairing, the Android device must be connected to the development kit. Please note that some terminal emulator applications do this automatically upon pairing. Once the device is connected, the display updates from “Not Connected” to “BT Connected” (see the following figure), and LED D5 illuminates.

7. Enter characters in the Android device application under “CMD Line Mode”, “Keyboard Mode”, or “Byte Stream Mode”. The characters will be displayed on the LCD of the development board, and are then echoed back by the PIC32 device and displayed on the Android device menu.
When SW1 through SW5 are pressed on the development kit a message is sent to the smartphone, “Button 1” will appear on the handset when SW1 is pressed. The same applies for when “SWx” (x = 1-5) is pressed, the corresponding “Button x” (x = 1-5) will appear on the smartphone (see the following figure).

In Keyboard Mode, phases can be saved to buttons (see the previous figure). Once the button is pressed or a string is entered, it will appear on the display of the development kit, as shown in the following figure.

The received text will “round robin” on the bottom three lines. In this example, the next string sent takes the place of Button 1, which is moved down one line until it repeats and starts from the top again.
1. Program the device with the hex file, BT Data Demo 4.hex for the CSR8811_BT_RADIO or RDA5876_BT_RADIO Bluetooth controller. This can be done by selecting the hardware support from the drop down box during compilation, as shown in the following figure.

2. On a Windows PC, select Start > Control Panel > Hardware and Sounds > Add a device. A dialog appears with a list of available bluetooth devices. From the list, select BT Data Demo 4.

3. Once BT Data Demo 4 is selected, the Windows PC will install the drivers and will allocate COM ports. When driver installations are complete, a pop-up message appears indicating the device is ready for use. The allocated COM ports vary, depending on the Windows configuration.
4. The COM ports used can also be identified from the Device Manager.

![Device Manager](image1)

5. Open a terminal emulator. For this example, PuTTY was used. This PC application can be obtained by visiting: [http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html).

![PuTTY Configuration](image2)
6. Once the terminal connection is established, a new terminal window opens and the transmitted and received data will be displayed, as shown in the following figure.

![Terminal Window]

7. The transmitted data is displayed, as shown in the following figure.

![Transmitted Data Display]
1.6 BLUETOOTH DATA WINDOWS PHONE DEMONSTRATION SETUP

Download a terminal emulator application for windows mobile. An application can be obtained from Microsoft by visiting:

http://apps.microsoft.com/windows/en-us/app/b9a0abda-0e28-4b3c-936b-3b9f415fdcdf

1.7 RUNNING BLUETOOTH DATA DEMONSTRATION 4

1. Program the device with the hex file, BT Data Demo 4.hex for the CSR8811_BT_RADIO or RDA5876_BT_RADIO Bluetooth controller. This can be done by selecting the hardware support from the drop down box during compilation, as shown in the following figure.

2. Enable Bluetooth on the smartphone.

3. Connect and pair the Windows phone with the PIC32 Bluetooth Audio Development Kit to the device named BT Data Demo 4.

4. If prompted by your Bluetooth device for a PIN, enter 0000. The device should connect and pair, and indicate as such on the display, as shown in the following figure.

5. After pairing, the Windows phone must be connected to the development kit using a terminal emulator. Connect to the device in Settings > Bluetooth.

6. After connecting to the device, open BT TERMINAL (the Windows application for SPP communication), as shown in the following figure.
7. The connected device name will be listed on the terminal emulator window.
8. Select the required device name and the BT terminal will establish connection with the device.

9. Once the Windows phone is connected, the display updates from “Not Connected” to “BT Connected” (see the following figure), and LED D5 illuminates.
10. Data can now be received and sent from the terminal emulator. The data will be displayed on the LCD of the development board, and is then echoed back by the PIC32 device and displayed on the Windows phone. When SW1 through SW5 are pressed on the development kit, a message is sent to the Windows phone, “Button 1” will appear on the handset when SW1 is pressed. The same applies for when “SWx” (x = 1-5) is pressed, where the corresponding “Button x” (x = 1-5) will appear on the Windows phone, as shown in the following figure.

1.8 BLUETOOTH DEMONSTRATION 4 SWITCH DESCRIPTIONS

1.8.1 Buttons
When SW1 through SW5 are pressed, a message is sent to the smartphone. The string “Button 1” will appear on the handset when SW1 is pressed. The same applies when SW1 through SW5 are pressed, the corresponding Button x, where x = 1 through 5, respectively, will appear on the device or Windows terminal.

1.8.2 LEDs
When D5 illuminates, this indicates a Bluetooth device is connected and paired. If D5 is off, no device is connected to the development kit. When D8 and D9 are illuminated, this indicates a SPP message was sent or received.

<table>
<thead>
<tr>
<th>TABLE 1-1: BLUETOOTH SPP DEMONSTRATION 4 CONTROLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>Switch</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>LED</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
1.9  BLUETOOTH BOOTLOADER DESCRIPTION

The BT_Data_Demo_4 supports the PIC32 bootloader data interface. The PIC32 bootloader is a specialized program that is capable of writing a new program to the device without the use of a physical memory programmer. This functionality is supported through the use of the Bluetooth SPP protocol to receive program data.

Note: This configuration is only supported by the Revision 3 PIC32 Bluetooth Audio Development Kit.

PIC32 bootloader memory placement:

![Diagram of PIC32 bootloader flow diagram]

The PIC32 bootloader occupies the beginning of program Flash memory and the application is placed after it. The actual memory addresses used depends on the setting specified in the specialized linker scripts that were used to compile the two programs.
1.10  BLUETOOTH BOOTLOADER DEMONSTRATION SETUP

1.10.1  PIC32 Bluetooth Audio Development Kit Setup

To be used as a bootloader target, BT_Data_Demo_4 must be configured for:

• Bootloader awareness
• Residence at the proper memory address

1.10.1.1  BOOTLOADER AWARENESS

Bootloader awareness is defined as being able to recognize incoming program binary data in the SPP data stream and storing that data in the proper external memory location.

The BT_Data_Demo_4 project contains configurations to easily enable bootloader awareness. These configurations are appended with the word “Bootloader”.

1.10.1.2  PROGRAM MEMORY PLACEMENT

Proper memory addressing is configured through the use of specialized linker scripts. The PIC32 bootloader program is compiled with a specialized linker script to place it properly into both boot and program memory. Any program that needs to be loaded by the bootloader must also be compiled with a specialized application linker script.

**Note:** As memory requirements differ for each application, the provided application linker script may not be correct would require alterations.

1.10.1.3  UNIFIED PROGRAM BINARY

The PIC32 bootloader and target application must be compiled into a single binary to be flashed onto the target device. This is done through the use of the “Loadable Projects and Files” capability of MPLAB X IDE.
1.10.2 Target Application Setup

1. Open the BT_Data_Demo_4 project in MPLAB X IDE.
2. Right click the Linker Files folder and select Add Existing File. Add the provided linker script pic32_application.ld.

The file name, pic32_application.ld, will appear as an entry within the Linker Files folder, as shown in the following figure.

3. Open the PIC32 bootloader workspace in MPLAB X IDE and make it the main project.
4. Right click the Loadables folder and select Add Loadable Project.
5. When the open file dialog appears, navigate to and select the Bluetooth Demonstration 4 MPLAB X IDE workspace folder. This will add an entry within the Loadables folder named bt_data_demo_4.

6. Make and program the PIC32 bootloader project.

Once the loading process is complete, the PIC32 bootloader and BT_Data_Demo_4 will both be installed onto the PIC32 Bluetooth Audio Development Kit.

1.10.3 Apple MFi Setup

Users desiring to enable Apple MFi capabilities must be licensed by Apple and obtain a copy of the Microchip MFi framework library and related files. To build properly, the files that exist in the BT_Data_Demo_4 project directory within h/MFi and src/MFi must be replaced with the versions from the obtained Microchip MFi package. The MFi binary library must also be added to the MPLAB X IDE project within the Libraries folder.
1.10.4 New Application Setup

1. Select an application that will be uploaded and bootloaded to replace the previously loaded BT_Data_Demo_4 program.

Note: If a program is uploaded that is not "Bootloader Aware" then bootloader capability will be left inert as there will be no program available to handle new binary program data.

The demonstration application, bluetooth_bootloader_test_app, will be used for this example.

2. Open the bluetooth_bootloader_test_app MPLAB X IDE workspace located within bootloader/pic32_apps/bluetooth_bootloader_test_app and build the application.

Note: This program is already configured to use the "pic32_application.ld" linker script. Other applications will need to be configured to use it using the previously stated method.

3. Once the application has been built, locate the .hex file produced by the build process, which is located at:

```
bootloader\pic32_apps\bluetooth_bootloader_test_app\bluetooth_bootloader_test_app.X\dist\default\production\bluetooth_bootloader_test_app.X.production.hex
```

4. Use the utility application, hex_to_bin, to convert the compiled .hex application into a bootloader binary file of the same name. This can be found within the folder, bootloader\hex_to_bin\dist.

   The output .bin file will be placed in the same folder as the .hex file regardless of where hex_to_bin.jar is located.

   An example command-line execution of this program is:

   ```
   bootloader\hex_to_bin\dist\>javaw –jar hex_to_bin.jar <path to .hex file>
   ```

Note: This program is a Java-based .jar file and requires Java JRE v1.7. For information on how to properly install and configure Java, please visit: http://java.com/en/download/help/index_installing.xml

5. Rename this file to bootloader_image.bin. This is the file name that the Microchip Bluetooth bootloader demonstration application is expecting.

6. Upload this file to the handset’s Download directory. Please note that the method for copying files to handsets varies by model.
1.10.5 Android Handset Setup
An Android application is available to provide the ability to upload an application binary file from the handset to the PIC32 Bluetooth Audio Development Kit.

1.10.6 Microchip Bluetooth Bootloader Demonstration Application
This application can either be compiled from source or the provided .APK file can be used. The Eclipse workspace for this project exists at:

```
bootloader\mobile_apps\android\Bluetooth_Bootloader_App
```

The precompiled .APK is located at:

```
bootloader\mobile_apps\android\Bluetooth_Bootloader_App.apk
```

1.10.7 Bootloading a New Application

1.10.7.1 BLUETOOTH BOOTLOADER APPLICATION
1. Connect to the PIC32 Bluetooth Audio Development Kit.
2. Launch the Microchip Bluetooth Bootloader Application on the Android handset.

If Bluetooth is not enabled on the handset, the user will be prompted to enable it.
3. Once Bluetooth is enabled, the user will be prompted to scan for active Bluetooth devices.
4. Once scanning is complete, the user should select the appropriate device from the list of available devices.

![Bluetooth Bootloader Demo](image)

**Note:** If the handset is not currently paired with the development kit, the Bluetooth MAC address of the development kit may be displayed instead of the formal name. This inconvenience may be avoided by pairing with the device before attempting to connect through the Microchip Bluetooth Bootloader Application.

5. Once the device has been selected, the handset will attempt to connect to it. This process may take a few seconds. If no pairing exists, the user will be prompted to accept the pair.

**Note:** It may take more than one try to establish a connection. If repeated attempts fail then power cycle the Bluetooth radio in the handset's settings. If this does not work, reset the PIC32 Bluetooth Audio Development Kit as well.
1.10.7.2 UPLOADING THE FILE

1. Once the handset has connected, the application screen will display the name of the file to be copied to the PIC32 Bluetooth Audio Development Kit and a button to initiate the transfer.

2. Touch Upload to transfer the file. During the transfer, the LEDs on the PIC32 Bluetooth Audio Development Kit will cycle for every four kilobytes the device receives. LED 5 will cycle to indicate when a four kilobyte page has been written to the external Flash memory. A prompt on the handset will indicate when the transfer is complete. The LEDs on the PIC32 Bluetooth Audio Development Kit will turn off as well.

Note: Current connection speed is approximately 4 Kilobytes per second.

1.10.7.3 RESTARTING THE DEVICE

With the new binary uploaded into external Flash memory, the PIC32 Bluetooth Audio Development Kit is ready to be restarted (i.e., turn the device off and then on). The bootloader will execute upon power-on and light LEDs 1 and 2. The bootloader will recognize the new program in memory and overwrite the previous program. During this process, LEDs 1-4 will be lit. Once the flashing process is complete, the bootloader will call the new application. In the case of the sample test application, the LEDs on the PIC32 Bluetooth Audio Development Kit will light up in sequence back and forth.
1.11 BOOTLOADER COMMANDS AND PROTOCOL

This section discusses the communication protocol used by BT_Data_Demo_4 to identify, receive, and verify incoming program binary data, and lists the commands that can be issued to the demonstration.

Binary data is sent in four kilobyte logical segments. Each segment is hashed previous to sending and the hash is sent before the segment.

Note: No hashing algorithm has been implemented at this time. Dummy hashes are sent and are not checked for consistency.

<table>
<thead>
<tr>
<th>Command</th>
<th>Size (bytes)</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>4</td>
<td>0x00B00701</td>
<td>Attempts to soft-reset the device. (Not verified)</td>
</tr>
<tr>
<td>Identify</td>
<td>4</td>
<td>0x00B00702</td>
<td>Returns an identification string.</td>
</tr>
<tr>
<td>Receive*</td>
<td>8</td>
<td>0x00B00703 00000000</td>
<td>Signals the impending arrival of program binary data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Last 4 bytes is the size of the incoming data. Normal demo SPP functions will be suspended until this many bytes are received.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response</th>
<th>Size (bytes)</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>1</td>
<td>0x0</td>
<td>Positive Acknowledgment</td>
</tr>
<tr>
<td>NAK</td>
<td>1</td>
<td>0x1</td>
<td>Negative Acknowledgment</td>
</tr>
</tbody>
</table>

The protocol for executing a binary transfer to the BT_Data_Demo_4 is shown in Figure 1-3.

FIGURE 1-3: BINARY TRANSFER EXECUTION PROTOCOL
1.12 KNOWN ISSUES

The following issues are known to exist in this demonstration:
• The boot image transfer protocol cannot recover from an interrupted transfer. A reset is required.

1.13 REVISION HISTORY

Version 1.0.1
This revision includes the following updates:
• Updates to RTOS files
• XC32 v1.3 compiler compatibility improvements

Version 1.0.2
This revision includes the following updates:
• Missing library components were added to enable compilation

Version 2.0
This revision includes the following updates:
• RTOS was removed
• The Graphics Driver was updated to support future displays
• A top-level user configuration header file (user_config.h), which includes global constraints was added

Version 3.0
This revision includes the following updates:
• RDA Bluetooth module support was added
• A global definition for Bluetooth Device ID value (CSR only) was added
• The common display design defined by display_config.h was added
• The application architecture was updated to enable compatibility with other demonstrations

Version 3.0.1
This revision includes the following updates:
• The assert diagnostics were revised, which are now user configurable (default is OFF)
• Stack usage diagnostics were added (default is OFF)
• Options in the user_config.h file were revised
• The Bluetooth SPP Library (1.3.1 - 1.5.1) was revised

Version 3.5
This revision includes the following updates:
• Improved random addressing feature, automatic reconnect
• Added new features to user configuration including number of pairing systems
• Added ability to eliminate the display from the project source, saving Flash memory
• Added iAP over Bluetooth support
• Added Bootloader over Bluetooth support