

---

## Bluetooth Audio Demonstration 11 with SBC and 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](http://www.microchip.com).

Bluetooth audio only support (A2DP + AVRCP + SPP + SBC decoder).

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.

This non-USB audio demonstration supports streaming wireless Bluetooth audio from any smartphone (i.e., Apple, Samsung, Google, etc.), PC, or Bluetooth-enabled device.

SPP communication is enabled between Bluetooth device and mobile handset for data communication during audio playback. A means of bidirectional data transfer is provided in this demonstration as an example of communication. This communication can be modified to provide any data transfer means needed by an application. In the case of the demonstration application, a means of observation is provided by exchanging some text strings both to and from the handset.

SPP and audio communications are simultaneous but on different channels. A single handset (smart phone) can establish both channels to enable control while streaming music.

### 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.

# Bluetooth Audio Demonstration 11 with SBC and SPP

## 1.2.1.2 RDA5876

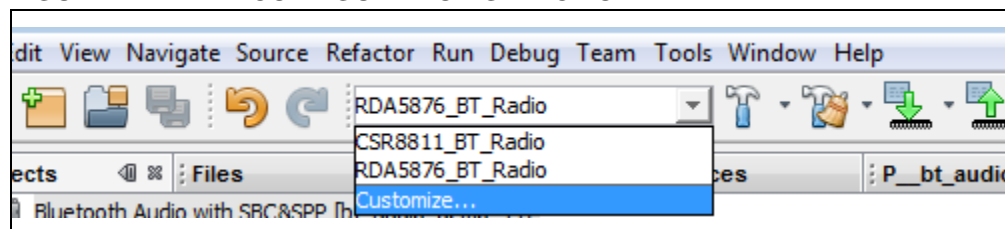
The RDA5876, integrates Bluetooth and a FM radio tuner into one device and is optimized for mobile applications. Bluetooth and FM can work simultaneously and independently, with low-power consumption levels targeted to battery powered devices. For the highest integration level, the required board space 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 its 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 smartphone that is currently connected and is re-enabled later, it will automatically reconnect when in range or when it comes back into range. In addition, when the development kit 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.

## 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 `BT_DEVICE_ID_2LSB` defines the last 2 bytes of the hexadecimal value. The last two bytes of the device address can be randomized by enabling `BT_DEVICE_ID_2LSB_RANDOMIZE`. 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 Voice Prompt

Starting with v2.0 of the demonstrations, the system is now equipped with a voice prompt option for some actions. To use the voice prompts two elements are required.

First, the voice prompt files must be loaded into the external Flash memory. To do this, Microchip has provided a separate external Flash memory writer project that will enable users to load voice prompts to their development kit. The documentation and project also allow the user to customize the voice prompt starting with their own audio voice or sound (in \*.wav format) and process walk through the steps to using this data in the correct format.

Second, the user-defined switch in the `user_config.h` file must be enabled to turn on voice prompt audio. Even while the prompts are loaded the function may be disabled at compile time with this switch. Note that if the voice prompts are loaded, or are loaded improperly, the function will be automatically bypassed even if this switch is enabled.

In general, voice prompts have a limited memory footprint and are designed to be approximately 1.5 seconds in length. This can be extended with changes in both the external Flash memory writer example, as well as the function offsets that are part of the `voice_prompt.c` file in the project.

Voice prompt events are run during the following events:

- Bluetooth Connected
- Bluetooth Disconnected

In each case, assuming the conditions previously described are met, a voice message will occur with this event. In the code, additional space has been allocated for future events. These can be tied to events that are hardware or Bluetooth-specific, such as 'Bluetooth paired', 'Power ON' or 'Power OFF'.

<p><b>Note:</b> Refer to the “Voice Prompt Programmer User's Guide” for information on the process to capture audio, and to format and download voice prompts. This document is available for download from <a href="http://www.microchip.com">www.microchip.com</a>.</p>
---

# Bluetooth Audio Demonstration 11 with SBC and SPP

---

## 1.2.8 Automatic Soft Mute

It is possible to enter a Soft Mute mode manually within the demonstration to stop audio output. See [Section 1.6 “Bluetooth Demonstration 11 Switch Descriptions”](#) for manual entry.

In addition to the manual *Soft Mute*, a user switch is provided to enter Soft Mute mode when fast forwarding and rewinding. When enabled, this action will turn off the audio output during the fast forward period, and then return the system to its previous state when the fast forward sequence ends. This may be particularly effective when using one of the AVRCP profile options like option 3, where the audio track will “skip” as it accelerates through the track, but stop temporarily at a new track and play at normal speed for a short time prior to resuming the *Fast Forward* function. This feature has the same effect in the reverse direction.

To enable this feature, a compile time switch is provided in the `user_config.h` file. To remove this feature, simply undefine this setting.

## 1.2.9 SPP Connectivity

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 allows the user to send a single command to change both the display name and the device name. This demonstration also allows user to send track name from bluetooth device to the terminal emulator of the mobile device.

## 1.2.10 Volume Sync

A new feature in demonstration v3.0 enables volume sync mode. The potentiometer of the DAC in the development kit is adjusted to increase or decrease the volume. When the potentiometer is adjusted the volume of the handset is adjusted accordingly.

<b>Note:</b> This feature is supported by Apple and Android (v4.4 and higher) devices.
--

To enable this feature, a compile time switch is provided in the `user_config.h` file. To remove this feature, undefine this setting.

## 1.2.11 Manual Pairing

A new feature in demonstration v3.0 enables manual pairing mode. See [Section 1.6 “Bluetooth Demonstration 11 Switch Descriptions”](#) for manual pairing.

A user compile-time switch is provided to enter Manual Pairing mode. When enabled, this feature manually allows the device to be discoverable for pairing for a time period of three minutes. The user can modify the time duration by varying the `PAIRING_TIMEOUT` macro defined in `user_config.h`. The device becomes unavailable for pairing after the time period. The user can pair the device without the requirement of a password or pin number.

This feature also allows the user to make the device unavailable for pairing manually within the stipulated time.

To enable this feature, a compile time switch is provided in the `user_config.h` file. To remove this feature, undefine this setting.

By default, the system is in automatic pairing mode. In this mode, the device will be discoverable at all times. When a handset attempts to pair, the pairing will be accepted by the device and it will not require a password or user intervention to accept the pairing.

Automatic pairing mode is very useful for development purposes, as random Bluetooth addressing is common and forces pairing on each power up. Manual pairing is common for commercial product devices.

## 1.3 DIAGNOSTICS

### 1.3.1 Asserts

A new diagnostic feature in demonstration v3.0 enables asserts to identify possible errors. To enable this feature, a compile time switch, `USE_ASSERT`, is provided in the `user_config.h` file. In addition, `USE_ASSERT_TO_DISPLAY` when enabled, displays the error message on the display screen of the development kit.

If `USE_ASSERT` is not defined `USE_ASSERT_TO_DISPLAY` will be disabled. This feature is primarily used for debugging purpose. `USE_ASSERT` is not defined by default, and therefore, asserts are disabled. To enable this feature, a compile-time switch is provided in the `user_config.h` file. To remove this feature, undefine this setting.

### 1.3.2 Stack Usage

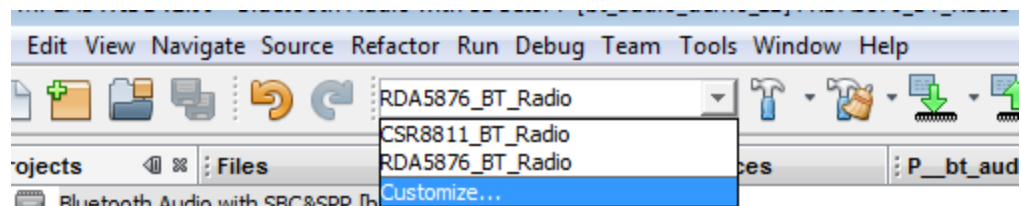
A new diagnostic feature in demonstration v3.0 enables the display of the maximum stack usage on the display screen of the development kit. To enable this feature, a compile time switch `DISPLAY_MAX_STACK_USAGE` is provided in the `user_config.h` file. This feature is used to measure the maximum stack usage.

**Note:** `BTAPP_USE_MANUAL_PAIRING` and `DISPLAY_MAX_STACK_USAGE` cannot be enabled simultaneously because usage of the SW6 for Manual Pairing has higher priority.

## 1.4 SETTING UP AND RUNNING BLUETOOTH AUDIO DEMONSTRATION 11

To set up and run this demonstration, follow these steps:

1. 1. Connect to the device using one of the two following options:
  - a) Use the 6-pin male interface for direct connection to the PICkit 3 In-Circuit Debugger/Programmer connector.
  - b) For MPLAB REAL ICE or MPLAB ICD 3, use the RJ-11 to ICSP adapter (P/N AC164110 purchased separately from [www.microchipDIRECT.com](http://www.microchipDIRECT.com)).
2. Program the device with the hex file, `BT Audio Demo 11.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.



3. Run the setup.
4. Enable Bluetooth for the Bluetooth audio device.
5. Select **BT Audio Demo 11** from the discovered Bluetooth devices on your smartphone or Bluetooth device.
6. If prompted by your Bluetooth device for a PIN, enter 0000.

The device should connect and pair, which is indicated by the message "Bluetooth Audio Dev Board" on the display of the development kit.

7. Connect a speaker or headphones to the line-out/headphone jack of the development board.
8. If using an external speaker, make sure that the volume control on the development kit is set to maximum (turned fully counter-clockwise).
9. Select the music track and press **Play**.

# Bluetooth Audio Demonstration 11 with SBC and SPP

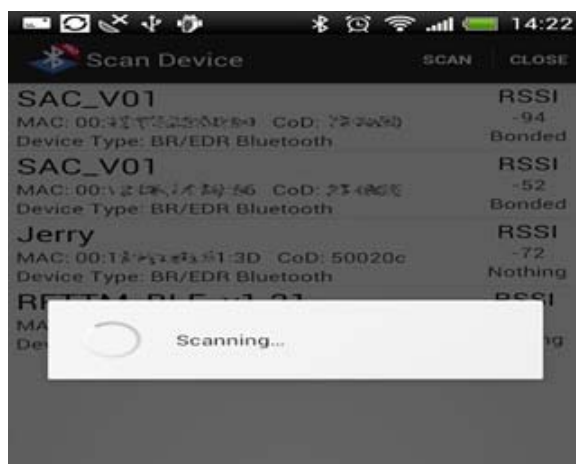
## 1.5 BLUETOOTH DATA SMARTPHONE DEMONSTRATION SETUP

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

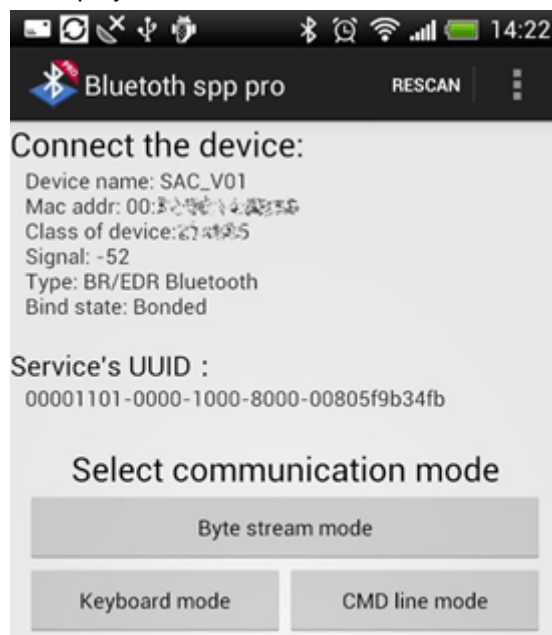
[https://play.google.com/store/apps/details?id=mobi.dzs.android.BLE\\_SPP\\_PRO&hl=en](https://play.google.com/store/apps/details?id=mobi.dzs.android.BLE_SPP_PRO&hl=en)

After installation:

1. Open the Android terminal emulator application and scan for the list of nearby devices.
2. While scanning, the device name will be identified faster in Pause mode.
3. Change the audio from Pause mode to Play mode once the device name is listed before connecting to the device.
4. Connect the smartphone with the Bluetooth Audio Development Board to the device named **BT Audio Demo 11**.
5. If prompted by your Bluetooth device for a PIN, enter 0000. The device should connect and indicate as such on the display, as shown in the following figure.



Once the Bluetooth device is connected to the mobile handset, data can be transferred and received by selecting the required communication mode during audio playback.



As shown in the following figure, CMD line mode was chosen to transmit and receive data. Commands can be typed in using the keyboard emulator.



The demonstration allows user to change the name of the Bluetooth device and the display name by sending the name change command via SPP Pro application. The Bluetooth device name and the display name are defined in `user_config.h` as `BT_AUDIO_CONNECT_NAME` and `BT_AUDIO_DISPLAY_NAME`.

Name change command is denoted by the letter "N" with the user-defined name.

For instance, to change the Bluetooth device name to "Microchip" the command "NMicrochip" is entered using the keyboard and Enter key is pressed.

The name change can be visualized by disconnecting from the Bluetooth device and rescanning for available device. The Bluetooth device with the new name "Microchip" is listed in the list of scanned devices. The command that is sent is displayed on the Bluetooth device display screen as "NMicrochip".



# Bluetooth Audio Demonstration 11 with SBC and SPP

Initially, when connected, the display screen appears as follows:



This command also modifies the display name that is present below the Microchip logo on the display screen, as shown in the following figure.



**TABLE 1-1: BLUETOOTH AUDIO DEMONSTRATION 11 CONTROLS**

Component	Label	Bluetooth Mode (no USB)
Switch	SW1	Shuffle (toggle)/Force Bluetooth Device to Unpair
	SW2	Repeat Track (toggle)/Bluetooth Device Disconnect
	SW3	Next Track/Fast Forward
	SW4	Play/Pause (toggle)/Soft Mute (toggle)
	SW5	Previous Track/Rewind
	SW6	SPP Data Transmission/Manual Pairing (toggle)/Display Stack Usage (toggle)
LED	D5	Bluetooth Device Connected/Blink of RTOS Bluetooth Error
	D6	RTOS Bluetooth Error
	D7	Audio Stream Indication
	D8	N/A
	D9	N/A
	D5-D9	CPU Exception Error



## 1.6 BLUETOOTH DEMONSTRATION 11 SWITCH DESCRIPTIONS

### 1.6.1 SW1: Shuffle (toggle)/Force Bluetooth Unpair

When SW1 is pressed and released quickly (less than three seconds), the system will send an AVRCP signal to the Bluetooth device to toggle the shuffle on and off. The action causes the Bluetooth device to randomly select the next track when enabled. This is often displayed on the player of the Bluetooth device.

If the user presses and holds SW1 for more than three seconds, this action will unpair with the currently connected Bluetooth device and causes the PIC32 device to erase the link key that was exchanged between the smartphone and the development board during the initial Bluetooth connect and pairing. When this occurs, the name of the Bluetooth device associated on the smartphone no longer matches the erased keys on the Bluetooth Audio Development Board. To reconnect, the user must manually force the smartphone to “forget” the demonstration name of the Bluetooth Audio Development Board Bluetooth to reset the link keys so that the user's smartphone can rediscover and subsequently re-pair.

### 1.6.2 SW2: Repeat Track (toggle)/Bluetooth Disconnect

When SW2 is pressed and released quickly (less than three seconds), the system will send an AVRCP signal to the Bluetooth device to toggle the repeat mode. Generally, Bluetooth audio devices will have three repeat modes, which are 'repeat off', 'repeat all', and 'repeat single/current track'. The system will rotate through these modes upon each toggle (i.e., quick press-and-release) of SW2.

If the user presses and holds SW2 for more than three seconds, this action will cause both the smartphone and the Bluetooth Audio Development Board to disconnect. Unlike the Force Bluetooth Unpair button (SW1), this action does not erase the link keys and the user does not need to force their smartphone to “forget” the demonstration name of the Bluetooth Audio Development Board and the user can simply reconnect at will.

### 1.6.3 SW3: Next Track/Fast Forward

This is a multi-purpose button, which has a primary and an alternate function. The primary function, *Next Track*, is listed first and is activated by a quick press (less than three seconds) and release. This action causes the audio device to advance to the beginning of the next sequential song.

The alternate *Fast Forward* function is activated if the user presses and holds SW3 for more than three seconds. This action will cause the audio device to fast forward to the next song on the audio device.

**Note:** Starting with demonstration v2.0, the *Fast Forward* function on the device is equipped with multiple demonstration profiles. The options are selected using a compile-time switch within the `user_config.h` file. In addition, the selected profile applies to the *Rewind* function. The different profile functions are as follows:

- Profile Option 1: (Default) Current software will *Fast Forward* through the current song to the beginning of the current song, even if the *Fast Forward* button is still being pressed and held (same as Apple iPhone model).
- Profile Option 2: *Fast Forward* continuously through all songs until the *Rewind* button is released (same functionality as Samsung Android).
- Profile Option 3: *Fast Forward* through the current song to the beginning of the next song, play for three seconds, and then resume *Rewind*. This process is repeated until the *Rewind* button is released.
- Profile Option 4: *Fast Forward* through the current song and continue rewinding until the *Rewind* button is released.

If no profile option is selected, the *Fast Forward* function will revert back to the handset-specific function. Different operating systems and different handset manufacturers have different use models for this function.

# Bluetooth Audio Demonstration 11 with SBC and SPP

---

## 1.6.4 SW4: Play/Pause (toggle)/Soft Mute (toggle)

This is a multi-purpose button that alternates between *Pause* and *Play* with each quick press and release. *Play* causes the audio device to start or resume music play and *Pause* will cause it to temporarily stop until the button is pressed.

When SW4 is pressed and held for more than three seconds, it will toggle the *Soft Mute* function. This function, when active, will disable the audio output at the DAC driver level, although it will not interfere with the Bluetooth device streaming. A display message will appear indicating that the system is now in Soft Mute mode. To stop the *Soft Mute* function, press and hold the SW4 button again.

## 1.6.5 SW5: Previous Track/Rewind

This is a multi-purpose button, which has a primary and an alternate function. The primary function, *Previous Track*, is listed first and is activated by a quick press (less than three seconds) and release. This action causes the audio device to return to the beginning of the previous song.

The alternate *Rewind* function is activated when the user presses and holds SW5 for more than three seconds. This action causes the audio device to begin to rewind towards the beginning of the current song on the audio device.

**Note:** Starting with demonstration v2.0, the *Rewind* function on the device is equipped with multiple demonstration profiles. The options are selected using a compile-time switch within the `user_config.h` file. In addition, the selected profile applies to the *Fast Forward* function. The different profile functions are as follows:

- Profile Option 1: (Default) Current software will *Rewind* through the current song to the beginning of the current song, even if the *Rewind* button is still being pressed and held (same as Apple iPhone model).
- Profile Option 2: *Rewind* continuously through all songs until the *Rewind* button is released (same functionality as Samsung Android).
- Profile Option 3: *Rewind* through the current song to the beginning of the next song, play for three seconds, and then resume *Rewind*. This process is repeated until the *Rewind* button is released.
- Profile Option 4: *Rewind* through the current song and continue rewinding until the *Rewind* button is released.

If no profile option is selected, the *Rewind* function will revert back to the handset-specific function. Different operating systems and different handset manufacturers have different use models for this function.

## 1.6.6 SW6: SPP Data Transmission/Manual Pairing

When SW6 is pushed and released quickly (less than three seconds), the track name of the songs that is playing currently will be sent via SPP and will be displayed in the terminal emulator of the SPP Pro application. This will be visible in KEYBOARD MODE/BYTESTREAM MODE.

When SW6 is pressed and held for more than three seconds, it will toggle the Manual Pairing function. When active, this function will make the device available for pairing for a time period of three minutes. A display message will appear indicating that the system is now “discoverable” for pairing. To toggle the *Manual Pairing* function, press and hold the SW6 button again. This will make the device unavailable for pairing. A display message will appear indicating that the system is “not discoverable” for pairing.

When SW6 is pressed and held for more than three seconds, a display message will appear indicating the maximum number of stack bytes used.

**Note:** The `BTAPP_USE_MANUAL_PAIRING` and `DISPLAY_MAX_STACK_USAGE` compile time switches cannot be enabled simultaneously because usage of the SW6 for Manual Pairing has higher priority

## 1.7 KNOWN ISSUES

The following issues are known to exist in this demonstration:

- Manual (click and drag) advance through the track can cause the system to lock up on older Bluetooth handsets (Android v2.2 and older)
- Some Android handsets do not support AVRCP commands for Fast Forward, Rewind, Repeat, and Shuffle; these commands are ignored
- Some Android players do not support track position or time information although AVRCP is functioning
- With voice prompts turned ON and programmed, a Bluetooth reconnect event with music already in the queue can cause an audio glitch
- Fast Forward and Rewind button profiles may not react as expected after multiple button functions; default operation is profiles OFF
- Manual pairing mode is not available in this demonstration
- To establish SPP pairing, the A2DP must not be streaming (in Pause)
- When sending SPP data in either direction, an audio glitch may be heard depending on the length and command type of the data

## 1.8 REVISION HISTORY

### Version 3.0

This is the initial version of this demonstration, which includes:

- RDA Bluetooth module support
- A global definition for Bluetooth Device ID value (CSR only)
- Manual Pairing feature
- Volume Sync for Apple devices and Android (v4.4 and higher)
- A common display design defined by `display_config.h`
- Bluetooth Stack for improved connectivity and compatibility with the Break-in/Party mode feature
- The application architecture is compatible with the USB MFi Stack

### 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
- Bluetooth name function control via SPP was revised to use a default name on power-up until changed via SPP
- The Manual Pairing function was added

# Bluetooth Audio Demonstration 11 with SBC and SPP

---

NOTES: