QCIOT-DA7212EVZ

Click Board™

Introduction

This document demonstrates how to use the EK-RA2L1 MCU as the core of a basic media player platform with voice-activated commands using the Cyberon Dspotter and QCIOT-DA7212EVZ Click board.

Contents

1. Hardware

1.1 RA2L1 MCU

The project is centered around the RA2L1 MCU. This device demonstrates the capabilities of lower-end RA devices, including voice command recognition. With its 32k of RAM, the RA2L1 has the minimum amount of memory available to run the DSpotter engine and other components. The EK-RA2L1 serves as the base platform for this project.

1.2 Audio Codec

For audio input/output handling, the QCIOT-DA7212EVZ Click board™ is connected to the mikroBUS socket on the EK-RA2L1. The on-board microphone provides the voice input to the DSpotter engine, and audio is played back through the on-board speaker and/or audio output jack.

For audio data exchange, the audio codec is configured as the master (controller) device while the RA2's SPI0 channel is configured as the slave (target) device. The DA7212 is configured via I^{2}C and channel IIC0 of the RA2 is used for this purpose.

The mikroBUS specification allows for SPI and I^{2}C connections which can be used to transmit PCM data and configure the codec, respectively. The SPI connection is intended to be used as an I^{2}S interface, with the Chip Select line functioning as the L/R clock. The I^{2}S interface makes it possible for the host board to exchange PCM data with the audio codec.

The input and output jacks, along with the speaker and microphone, are configured by the audio codec to enable and route the necessary channels. The board is powered by the +5V and +3.3V inputs of the mikroBUS connection. The 5V input is used by an LDO regulator to provide regulated 3.3V and 1.8V outputs used by the codec. The 3.3V input is used to drive indicator LEDs.

The codec requires a reference clock signal to function and this is provided by an on-board crystal oscillator. Alternatively, a PWM input on the mikroBUS interface can provide the necessary clock.

The complete list of connectors and headers and their functions are listed in Table 1.

Table 1. Connector and Header Functions

Connection Designator	Function
J1	Header to open or close + output to SP1 (speaker).
J2	Header to open or close – output to SP1 (speaker).
J3	AUX input.
J4	Headphone/Audio output.
J5	mikroBUS connection.
J6	
J7	Header to select the on-board 12.288MHz crystal oscillator or external PWM input for the codec MCLK signal.

1.3 Media Storage

The audio files are stored in a Micro SD card inserted in an external adapter Pmod board; connected to PMOD2 on EK-RA2L1. The RA2 uses the SCI9 channel to communicate with the Micro SD card.

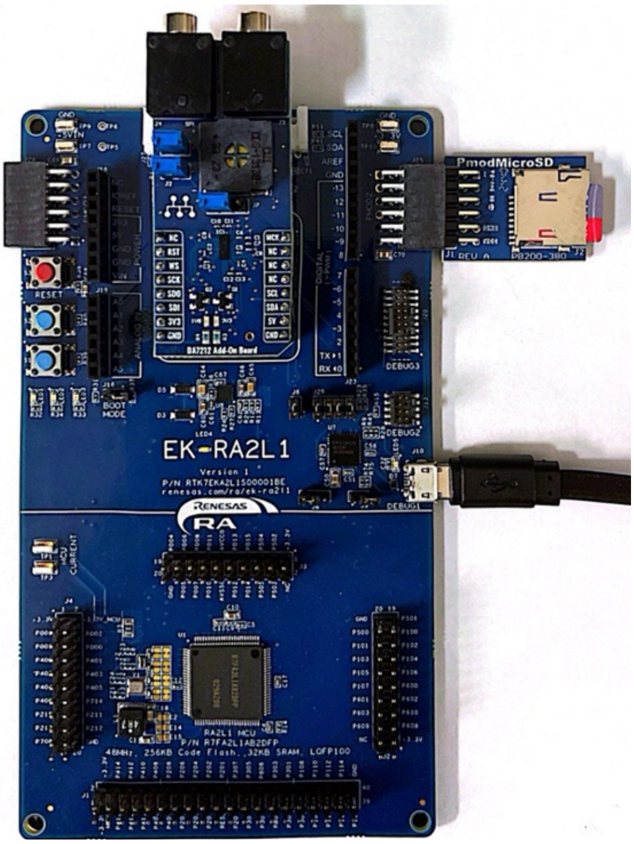


Figure 1. QCIOT-DA7212EVZ Click Board and Micro SD with EK-RA2L1 MCU

2. Software

2.1 DSpotter Voice User Interface (VUI)

The DSpotter voice command recognition engine is used to process and execute the audio player’s control commands. The VUI uses a single-stage command recognition format and listening is activated by a user button press. When DSpotter is running, the application periodically feeds audio samples to the engine, and when a command is recognized, it executes the appropriate functions.

2.2 Codec Configuration

At startup, the DA7212 is configured for use by enabling the appropriate inputs/outputs and configuring settings for channel gain, mixing, etc.

2.3 FatFs

The project uses FatFs to access the Micro SD card contents and read file data for audio playback. When the playback is activated, the appropriate file is opened and file data is read periodically until playback is halted.

2.4 Audio Playback

Audio playback uses **.WAV** file format, reading PCM data from the file system and sending it to the audio codec while playback is enabled.

Playback is controlled via voice command, which includes commands to start, stop, pause, and skip audio tracks. To format the Audio Files as **.WAV** files, refer to the QCIOT-DA7212EVZ manual.

See Figure 2 for a high-level algorithm flowchart.

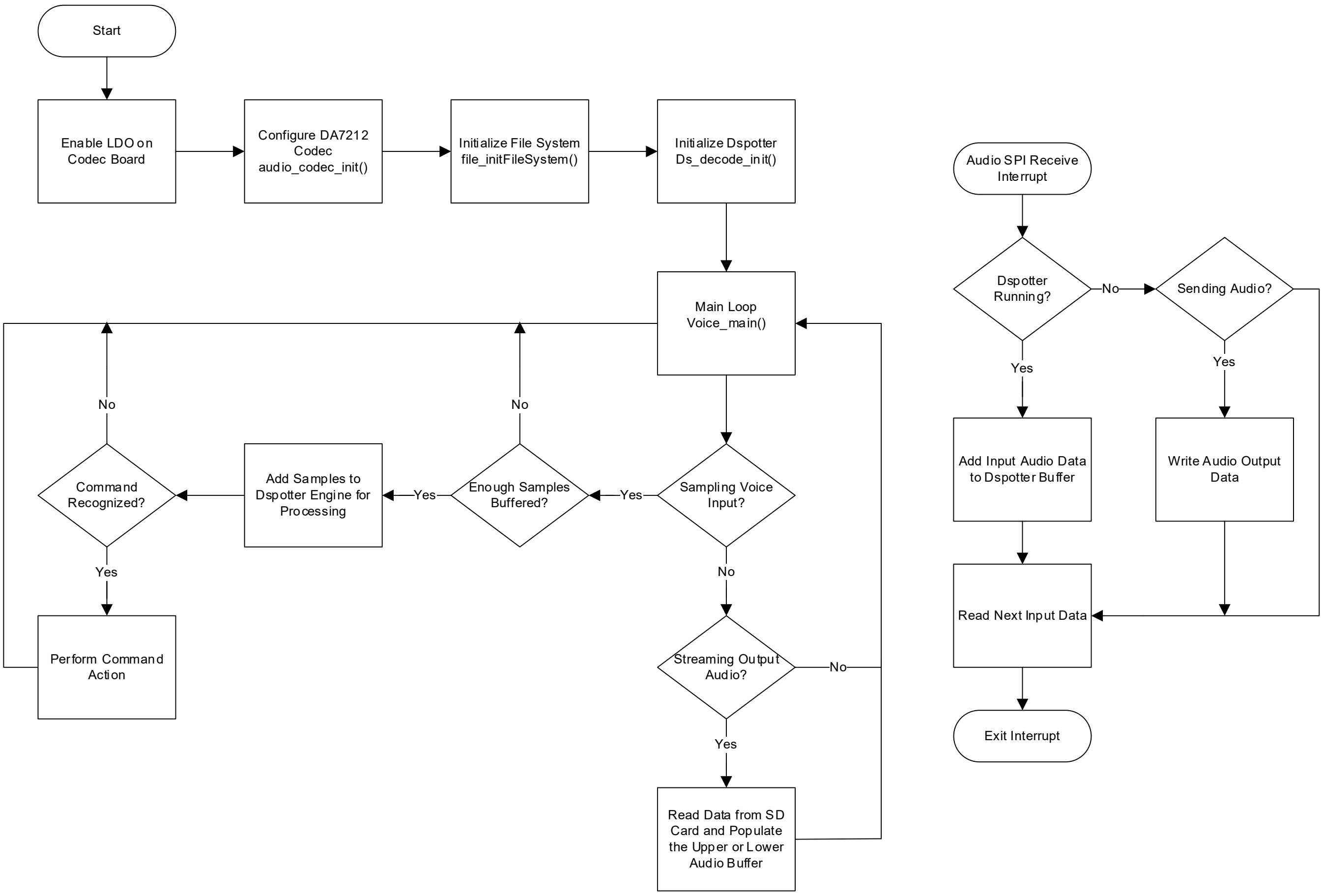

Start
Enable LDO on Codec Board
Configure DA7212 Codec audio_codec_init()
Initialize File System file_initFileSystem()
Initialize Dspotter Ds_decode_init()
Main Loop Voice_main()
Sampling Voice Input?
Yes
Enough Samples Buffered?
Yes
Add Samples to Dspotter Engine for Processing
Command Recognized?
No
No
Yes
Perform Command Action
No
Streaming Output Audio?
No
Yes
Read Data from SD Card and Populate the Upper or Lower Audio Buffer
Audio SPI Receive Interrupt
Dspotter Running?
No
Sending Audio?
Yes
Yes
Add Input Audio Data to Dspotter Buffer
Write Audio Output Data
Read Next Input Data
Exit Interrupt

Figure 2. High-Level Algorithm Flowchart

3. Functional Description

During firmware startup, the application initializes the DSpotter instance, audio codec, file system, and the necessary communication interfaces (SPI, I^{2}C, etc.).

3.1 Voice Commands

This project functions as a music player and the supported voice commands are focused on controlling audio playback. The following commands are supported:

- **Play Music**
  Starts playback of the current file.
- **Stop Music**
  Stops playback and closes the current file.
- **Pause Music**
  Stops playback but does not close the file.
- **Next Track**
  Stops playback, increments file index, and opens the new file.
- **Previous Track**
  Stops playback, decrements file index, and opens the new file.

When the Stop or Pause command is processed, playback can be restarted with the Play command. If playback was stopped, then the track that was playing will restart from the beginning. If playback was paused, then the track will resume from the current point.

If the **Next Track** or **Previous Track** command is issued, the next file is opened, but the new track will only start playing if playback is not stopped or paused.

While playing audio, the green LED (LED2) blinks slowly. If the playback is paused, the LED turns solid green, and if stopped it is turned off completely. When skipping tracks, the LED flashes quickly several times.

3.2 DA7212 Digital Audio Interface (DAI)

The DA7212's Digital Audio Interface (DAI) is used in conjunction with the RA2's SPI peripheral to exchange audio input and output data. The DAI is a four-wire serial interface with bit clock (BCLK), word clock (WCLK), data-in, and data-out lines. Most commonly, this would be used in I^{2}S format; however, because a standard SPI channel is used on the MCU, the data clocking will not line up properly. Thus, the DAI is configured to use the left-justified format. For more details on the format differences, see the DA7212 Datasheet.

The sample rate is configured to be 16kHz with 32 bits per WCLK period and a word length of 16 bits.

The DA7212 serves as the master (controller) in this configuration and the RA2 serves as the slave (target) device.

3.3 Audio Format

To simplify this example project, the audio formats for both input and output data are the same. Since the DSpotter requires 16kHz PCM data, the audio files stored in the file system are also formatted as 16kHz PCM mono-channel.

The audio files being mono-channel further simplifies the data transfer because the SPI peripheral's configuration only clocks data out when the WCLK (used as Chip Select) is low. Thus, only right-channel data is sent to the codec.

3.4 File System

As previously stated, FatFs is used to access the files on the Micro SD card. At startup, the file system is initialized and the application reads the contents of the root directory and any subdirectories. For project simplicity, the names of the found files are stored in a char-array with a configurable maximum length. FatFs is configured to use shortened file names, so only 13 characters maximum is allowed for each file name entry.

During development, note that not all SD cards work with FatFs. It is recommended to use one produced by one of the major manufacturers. For this project, a Sandisk Micro SD card is used.

Only **.WAV** format or other files containing raw PCM data are intended to be stored on the Micro SD card. When opening and sending file data, the file system implementation does not distinguish the file type, so it is important that any files on the SD card are formatted properly for audio playback.

3.5 Playing Audio

When audio tracks are being played, a constant stream of data from the Micro SD card to the audio codec is required. To accomplish this, a 1kB buffer is used to hold audio data. Data reads from the SD card are performed in 512-byte blocks, using one read to fill the lower buffer and another for the upper buffer.

Data is sent to the codec from this same 1kB buffer in 128-byte increments. When new data needs to be read from the Micro SD card, the application ensures that it is not overwriting the buffer that contains data currently being transmitted.

To simplify the project, the entire contents of the **.WAV** file (including the header) are sent to the codec. Though this technically results in audio noise, any perceived noise is brief enough to ignore for the purposes of this example.

3.6 External Hardware Connections

A Micro SD card reader is required for this example project. The application uses SCI9 to communicate with the external hardware. A Diligent Micro SD card reader is connected to PMOD2 on the EK-RA2L1 MCU.

A valid UART connection to the MCU should be made for loading the DSpotter license. Details on how to load the license is provided in the QCIOT-DA7212EVZ manual.

4. Project Code Contents

4.1 Voice Processing

The primary execution loop is in **voice_main.c** and this file also contains the application's calls to the DSpotter engine for voice processing. When voice command recognition is active, each main loop iteration checks for a full voice data buffer, and if it is full, calls the DSpotter engine to process the samples. If a voice command is recognized, the application acts based on which command was detected.

4.2 Audio Record

AudioRecord.c controls the acquisition of voice samples and adds them to the voice data buffer. It also contains the SPI callback function for the digital interface with the DA7212 codec. In this callback function, new voice data is read in and added to the DSpotter buffer, while any audio output data is written to the codec.

4.3 DA7212

The `DA7212.c/.h` files are the hardware-agnostic driver files to handle configuration of and communication with the DA7212 audio codec. It defines the data type `da7212_codec_t`, which contains a list of the configuration registers of the DA7212. These virtual registers can be manipulated directly, or the various dedicated function calls can be used to modify their contents based on certain input parameters. These values can then be written to the codec to update its configuration.

The file also includes `#define` statements for INIT, READ, and WRITE that can be redefined to call the proper functions for the associated I^{2}C peripheral.

4.4 Audio Codec

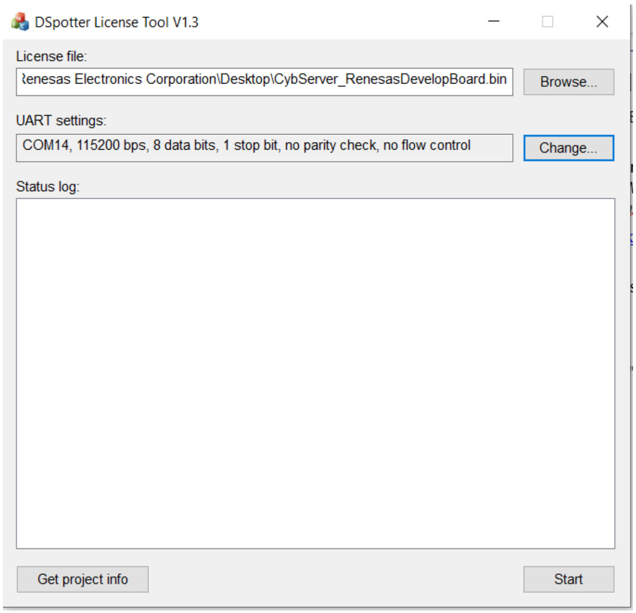
The `audio_codec.c/h` files act as the interface between the main application and the DA7212 driver code. It is intended that any calls directly to the DA7212 driver functions go in functions here, and the application calls the functions from `audio_codec.c`.

Note: To program the example code on RA MCU (EK-RA2L1), use the procedure found in section 4 of the QCIOT-DA7212EVZ manual.

5. Adding the Cyberon DSpotter License on EK-RA2L1

1. Run the sample code in Debug Mode with the USB to TTL connection on the MCU. The USB to TTL connection is provided in section 4.
2. Ensure the QCIOT-DA7212EVZ is connected to the EK-RA2L1 via the mikroBUS connector, and the Micro SD card reader is connected to PMOD2 on the EK-RA2L1.

   Note: Ther Micro SD card reader connection can be ignored if the **file_initFileSystem()** routine is commented out in **hal_entry.c**
3. Download the files from Voice User Interface - Renesas Cyberon VUI Installation Package - All Documents.
4. Run **DSpotterLicenseTool.exe** file. Load the **CybServer_RenesasDevelopBoard.bin** file and ensure you have the correct UART COM port setting selected with 115200 bps Speed, 8 bits Data and 1 stop bit. Click on *Start*.



DSpotter License Tool V1.3

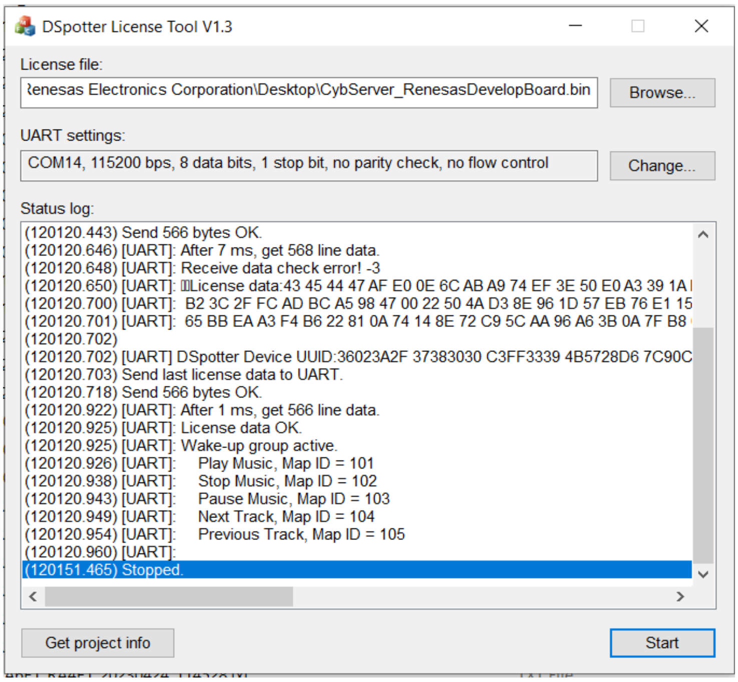
License file: enesas Electronics Corporation\Desktop\CybServer_RenesasDevelopBoard.bin

UART settings: COM14, 115200 bps, 8 data bits, 1 stop bit, no parity check, no flow control

Status log:

Figure 3. DSpotter License Tool

5. The DSpotter license should now be downloaded on the MCU.



License file:

ʀenesas Electronics Corporation\Desktop\CybServer_RenesasDevelopBoard.bin

UART settings:

COM14, 115200 bps, 8 data bits, 1 stop bit, no parity check, no flow control

Status log:

```
(120120.443) Send 566 bytes OK.
(120120.646) [UART]: After 7 ms, get 568 line data.
(120120.648) [UART]: Receive data check error! -3
(120120.650) [UART]: ▯License data:43 45 44 47 AF E0 0E 6C AB A9 74 EF 3E 50 E0 A3 39 1A
(120120.700) [UART]:  B2 3C 2F FC AD BC A5 98 47 00 22 50 4A D3 8E 96 1D 57 EB 76 E1 15
(120120.701) [UART]:  65 BB EA A3 F4 B6 22 81 0A 74 14 8E 72 C9 5C AA 96 A6 3B 0A 7F B8
(120120.702)
(120120.702) [UART] DSpotter Device UUID:36023A2F 37383030 C3FF3339 4B5728D6 7C90C
(120120.703) Send last license data to UART.
(120120.718) Send 566 bytes OK.
(120120.922) [UART]: After 1 ms, get 566 line data.
(120120.925) [UART]: License data OK.
(120120.925) [UART]: Wake-up group active.
(120120.926) [UART]:     Play Music, Map ID = 101
(120120.938) [UART]:     Stop Music, Map ID = 102
(120120.943) [UART]:     Pause Music, Map ID = 103
(120120.949) [UART]:     Next Track, Map ID = 104
(120120.954) [UART]:     Previous Track, Map ID = 105
(120120.960) [UART]:
(120151.465) Stopped.
```

Figure 4. Successful License Load

6. Revision History

Revision	Date	Description
1.00	Jun 9, 2025	Initial release.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.