

DevCon 2008 Training

Lab B01 - Low Cost Audio

Description: Subatomic Particle Board ADPCM Audio Playback lab

Objectives

After completing this lab you will:

1. Have understood the features of the Subatomic Particle Board.
2. Understand what the ADPCM sample code does and how to use it to playback audio clips.
3. Understand how HEW Target Server can be used to program the SPB board's serial flash.
4. Explain the usage of timers/PWM peripheral settings when decoding sound of a particular sampling rate.

Lab Materials

Please verify you have the following materials at your lab station.

- SPB 'Subatomic Particle Board' type 'YM16CSPB' which uses the MCU M16C/26A.
- HEW, v 4.03.
- M16C E8 Emulator V.2.02.00 or above
- NC30 compiler 5.43 or higher
- Soft content in C:\WorkSpace\ADPCM:
 - SPB_PGs.zip. This contains directories for wav2adpcm_for_lab and Flash_memory_loader with win_sys32.

Skill Level: Familiar with Renesas tools and MCUs

Total Time to Complete Lab: 90 minutes

Lab Sections

1

Preparations

Time to complete task: 10 minutes (This should already have been done by lab assistant).

2

Create the SPB Project & Download the code

Time to complete task: 10 minutes

3

Run the SPB ADPCM demo with the debugger

Time to complete task: 15 minutes

4

Create your own audio MOT-file

Time to complete task: 15 minutes

5

Load your audio file to the SPB's serial flash chip using HTS

Time to complete task: 15 minutes

Lab Procedure


PART I

Preparations

- 1** 10 minutes (0 minutes at DevCon since this will be done by a lab assistant).

Overview: Prior to the lab, this should exist/have been done on your PC

Procedural Steps:

 Skip to next section at DevCon lab as it should already have been run.

Check that the following is installed on your PC:

1. Strat HEW from the start menu and check that version 5.43 or more of the Renesas M16C Toolchain is selected. This can be seen under **Tools -> Administration -> Toolchain**.
2. E8 V2.10 or more installed. That is equivalent to version 2.07 or more of the R8C_E8_SYSTEM debugger. Check under **Tools -> Administration -> Debugger Components**.

HEW Target Server

3. Configure **HEW Target Server** by running
`C:\Program Files\Renesas\Hew\registerserver.bat`
4. **Register HEW Target Server** from within HEW. Start HEW and go to menu
Tools -> Administration -> Search Disk -> Start, select HEW target server -> Register.
5. Press 'Close', 'OK'.
6. Close HEW.

Add SPB Project Generator files

With HEW NOT running:

7. **Install** SPB Project Generator files by extracting **SPB_PGs.zip** with full paths to C:\. This assumes HEW is in default location. Extract files by right-clicking and selecting "Extract to" and "**C:**", not the default path. The HEW Project Generator files will be updated.
8. **Delete** the file `C:\Program Files\Renesas\Hew\pretools.hdb` to force HEW to register the new Project Generator files.

Install Serial Flash Memory Loader project

9. Install Serial Flash Memory Loader project by extracting **Flash_memory_loader.zip** to e.g.
`C:\Workspace\ADPCM_B01`
10. Copy the files in `..\Flash_memory_loader\win_sys32\` to `C:\WINDOWS\system32`


2 Create SPB Project & Download the code

Time to complete task: 10 minutes

Overview: Start HEW and the lab's firmware project. Download the linked object code firmware to target for debugging.

Procedural Steps:

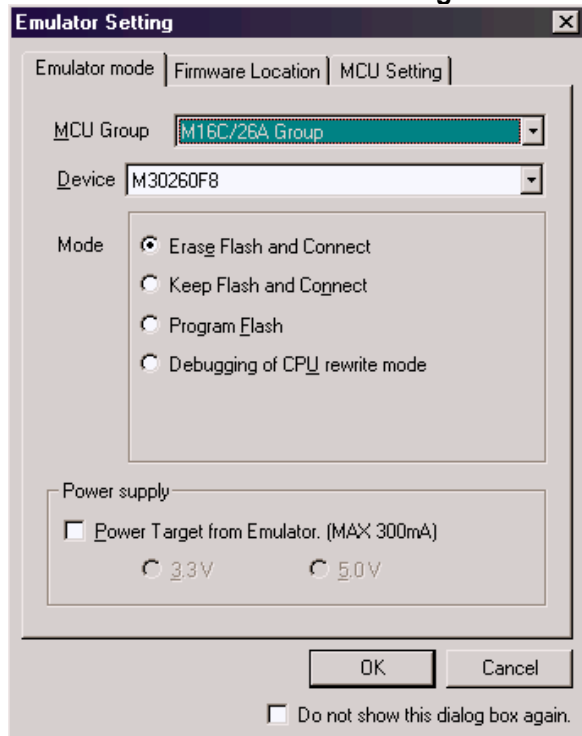
HEW

1. Launch HEW from the shortcut on the desktop or from the **Start Menu -> Programs -> Renesas -> High Performance Embedded Workshop -> High Performance Embedded Workshop**.
2. Select **File -> New Project Workspace** and use the following settings:
 - Name the project "**M16C_SPB**" or similar.
 - CPU Family: **M16C**
 - Project Type **Neutrino**
3. Click **OK**.
4. Select **Tutorial ->Next**.
5. Click **Finish -> OK**.
6. **Build**. Compile and link the source code by pressing F7, or click the Build icon .

Create new E8 Session and connect to target

1. **Connect** your SPB board with a USB mini cable to your PC. it should **enumerate** as standard Renesas E8 device.
2. Click **File -> New Session**. You don't have to save the default session.
 - Enter session Name e.g. "**E8**".
 - Select "Create new session" to **M16C E8 System**. Click **OK**.
3. CPU Series: **M16C/Tiny**
CPU Group: **26A ->Finish**.

4. You are now in the **Emulator Setting** window.



5. In the **Emulator Mode tab** select
- MCU Group: **M16C/26A Group**.
 - Device: **M30260F8**.
 - Mode: **Erase Flash and Connect**
- Select "Power Target from Emulator", 3.3 V.
6. The **Firmware Location tab** determines where the target debug kernel is to be located
- Set **Program location** to e.g. **FF700**.
 - Set **RAM location** to e.g. **B80**.



The map file (select Tools -> MapView) tells us that the application resides at 0xF0000 to around 0xF0C00, and RAM is occupied to about 0x700.

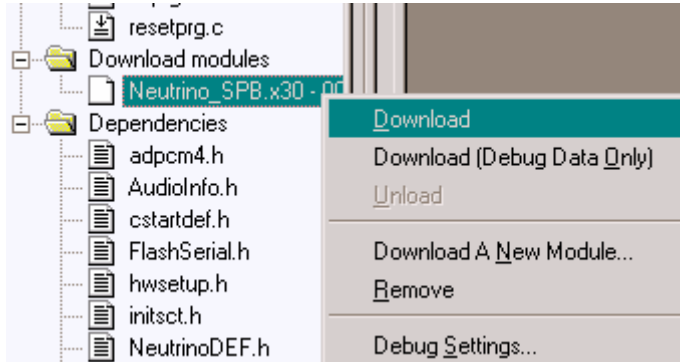
7. Click **OK**

Download firmware to target



You can find the X30 debug version binary in the file window pane to the left, in the 'Download modules' folder. The debug kernel will be downloaded at the same time with this file.

8. Compile and link the source code (F7).
9. After successful compilation, download the program to the SPB by **right clicking** and selecting **Download** (or just double-click on the file).



10. Accept any dialogue that a different version of the E8 firmware is necessary and let it reprogram the E8.

3 Run the SPB ADPCM demo with the debugger

10 minutes

Overview: We will run the code with the debugger, take a look at how audio clips are selected, and watch the data come in from the SPI interface.

Procedural Steps:

1. **Run.** You can now run the code by selecting Debug->Reset->GO, (or Shift+F5). Make sure it is running by checking that the stop sign is red in HEW, and that the RED and AMBER LEDs are blinking.
2. **Connect headphones** to the SPB.
3. Press the board's **pushbutton**. Pressing PB1 again while audio playing on one channel will cause another audio stream on the other channel.

Modify play order

4. Open the file `Streamingaudio.c` and find the string array `audio_file_array[]`.

The individual sound clips will be played not in the order they are stored in the serial flash, but in the order their names are placed in this array.

5. Go ahead and **rearrange** the order in which the audio files are played.
6. **Recompile** (F7), press **Reset->Go** (Shift+F5), and listen to the playbacks by pressing the pushbutton.

Question:

3.1) Did the playing order change? _____

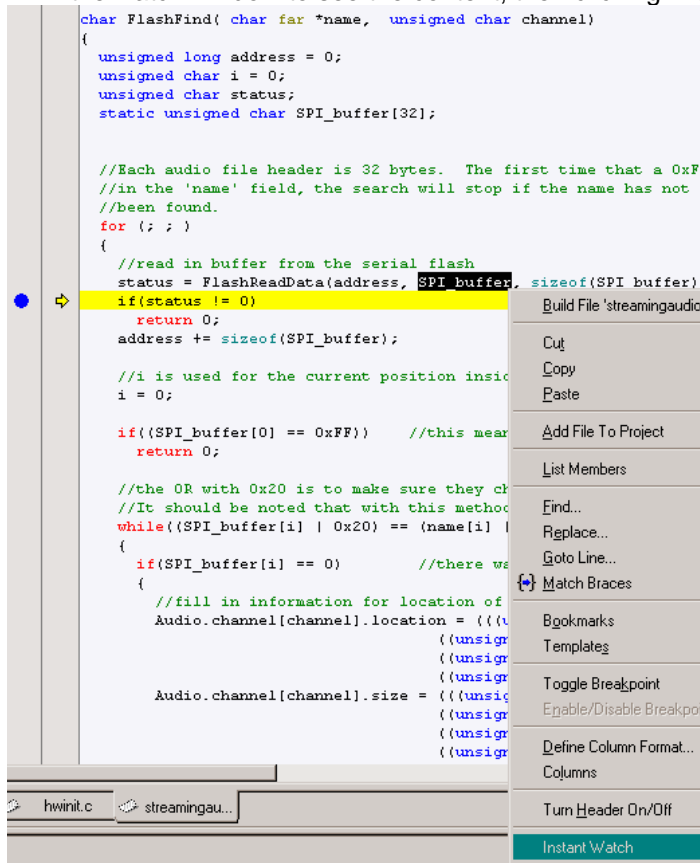
Observe search for audio clip

7. Go to the function definition of `FlashFind` in `StreamingAudio.c`, ~line 300.
8. **Set a breakpoint** in function `FlashFind()` in file `streamingaudio.c`, right after the call to `FlashReadData()`.

Question:

3.2) Which variable in `flashfind()` contains the string name of the audio file to find and play?

9. Press the pushbutton (with the code running). The debugger should now stop at line 315.
10. Highlight the **SPI_buffer** variable and add it to the **watch** window by first right-clicking on it, then pressing the little '+' in the watch window to see the content, then clicking "Add".



Question:

3.3) What happens to the SPI_buffer variable each time you press Go (F5)?



The code searches through the audio clips as they are retrieved one by one from the serial flash chip (via the SPI interface). The actual match of SPI_buffer filename and in data name occurs in the while loop a little further down in the function.

11. Close HEW.



Create your own audio MOT-file

10 minutes

Overview: Creating a custom MOT-file with sound clips for the serial flash memory chip.



You can create a custom MOT-file for the serial flash memory chip containing a series of your own sound clips.

We will create a MOT-file from a series of sound clips that are in the PC native WAV-format, then load it to the SPB serial flash with a HEW Target Server based loader tool.

The process has 2 steps:

- Convert from WAV-format to ADPCM-format (.aud)
- Convert all ADPCM-files to one MOT-file (.mot)

Procedural Steps:

Convert a WAV-file to an AUD file ADPCM format file

1. Unzip the file wav2adpcm_for_lab.zip. It should be under the C:\WorkSpace\ADPCM_B01 directory.

To convert from .wav (WAVE file) to .aud (ADPCM) there is a command line executable called wav2adpcm.exe.

Here is an example of running it from a windows command:
`>wav2adpcm.exe "my_wav_file.wav" "my_adpcm_file.aud"`
 where the first argument is the wave file you want to convert, and the second argument is the adpcm output file.

The best way to convert multiple files is obviously to create a batch file that will convert multiple files at once, instead of just one by one as shown above.

2. Open `convert_spb_jetsons.bat` for editing. You will see all the commands used to make a loadable audio file.

3. Before we run it, let's look at the line at the end:

```
bin_to_mot.exe -m "greatest.aud" "importantassignment.aud" ...
```

Once we have converted our AUD (adpcm) files, we need to combine them all into one large MOT file. This is done with the command line executable program `bin_to_mot.exe`.

The utility `bin_to_mot` combines multiple files into one single file. Here is an example:

```
>bin_to_mot.exe -m "my_adpcm_file_1.aud" "my_adpcm_file_2.aud" ... "my_adpcm_file_n.aud"
my_mot_file.mot
```

Here, the "-m" means output to a file of type MOT. Put each ADPCM file one after the other as arguments. The last argument will be the name of the output MOT-file.

4. Run `convert_spb_jetsons.bat` to generate a multi sound-clip MOT file.

5 Load your audio file to the SPB's serial flash chip using HTS

10 minutes

Overview: We will run the HEW "Memory Loader" project. This code runs on the M16C and receives the multi sound-clip MOT-file data from HEW Target Server, and via SPI writes the file content to the serial flash chip.

Procedural Steps:

Program the serial flash chip

1. Close HEW before proceeding.
2. Run the program "RTA Memory Loader.exe" in
`C:\Workspace\ADPCM_B01\Flash_memory_loader\Visual Basic`
3. Press **Select Workspace** and browse to:
`C:\Workspace\ADPCM_B01\Flash_memory_loader\Target\NeutrinoMemLoader\NeutrinoLoader.hws`

4. Press the **Select File** button and browse to the "spb_audio_jetsons.mot" file that you created by running convert_spb_jetsons.bat in section 4.
5. Press the **Open Workspace** button. Connect as before when the Emulator Settings window pops up.
6. The Open Workspace button should change its name to **Program** after connection. Press it. Note that once programming is started there is about a 10 second delay before programming starts to while erasure is executing.
7. If the debugger pops open a window asking you for the location of a source file, locate the file and press OK.
8. If you get "Memory Loading Complete", you succeeded. Your new sound file should now be in place in **the serial flash** chip.
9. Do not close HEW. Close the RTA Serial Memory Loader window which will close HEW.

Using the new audio information

10. Re-open the tutorial workspace. Start Hew, **File -> Recent Workspaces -> C:\Workspace\M16C_SPB** or the name you chose.
11. Download the X30-file again.

Question:

5.1. WHY don't you hear the new audio clips when you press the board's pushbutton?

12. Adapt the data variables `audio_file_array` and `NumAudioFiles` in `streamingaudio.c` to the new audio file data. Examine the batchfile's output names to get the correct name strings. Only include the name of the file, that is, without the ".aud". You can copy from this lab manual if you have it on your PC (See the C:\WorkSpace\ADPCM_B01 directory).

```
#define NumAudioFiles 11
/* This array holds the possible audio files in the current mot file */
char far * const audio_file_array[] =
{
/* 11kHz */
"jetsnbel",
"greatest",
"jetphone",
"lateschool",
"workwork",
/* 16kHz */
"report2myoffice",
"importantassignment",
"wrongbutton",
"reallydaddy",
"jetcar",
"fired"
};
```

Change playback sample frequency



Several of the new audio clips were sampled at a 16 kHz sampling rate. We need to change the playback frequency so these clips are replayed correctly.

13. Open `hwinit.c` and go to the `InitTimer` function.



The timer is set to count the system clock, or f_1 . In our case:
 $f_1 = 20$ MHz.
 f_s = sampling frequency. Currently set to 11025 kHz.
 trv = Timer reload value, when the countdown timer wraps around zero it resets to this value

To get the correct sampling interrupt rate (for 11 kHz), we needed to solve the equation

$$f_s = f_1 / trv$$

which gives $trv = 1815$. Since the timer doesn't wrap until after a whole cycle during which the counter is 0, we should also decrement the value by one.

14. We need to change to a **16 kHz** sample rate to play the other clips correctly.

Question:

5.2. To change the playback frequency for 16 KHz, what must the timer reload value 'tb1' be changed to?

(Do your calculation here)

_____ → tb1 = _____

15. **Recompile and play the audio clips again!**

/End of lab

Restoring the boards serial flash after the lab

To restore the ADPCM board as it was before the lab, reprogram the serial flash chip as in section 5 but using the file `C:\Workspace\ADPCM_B01\wav2adpcm_for_lab\mot_files\spb_audio.mot` instead in step 4.