08-6294-RN-ZCH70
MARCH 31, 2009
0.2

Release Notes for FLAC Decoder on ARM11 ELINUX

ABSTRACT:

Release Notes for FLAC Decoder on ARM11 ELINUX

KEYWORDS:

Multimedia codecs, FLAC, Decoder, Audio, Elinux

APPROVED:

Shang Shidong

Revision History

Version	Date	Author	Change Description
0.1	25-July-2008	Guo Yue	Initial Draft
0.2	31-March-2009	Guo Yue	Add the new build option BUILD=ARM11ELINUXDLIB

Table of Contents

Introduction

1.1 Purpose

The purpose of this document is to provide information on the package contents, instructions on building library and test applications and test execution on ARM11 ELINUX, RVDS and Unix.

1.2 Scope

The scope is restricted to information on the package contents and instructions for building and testing. This document does not provide architecture or details about the APIs provided in the package. Performance data will be provided in another document as detailed in the Requirements Book.

1.3 Audience Description

The reader is expected to have basic understanding of Audio Signal processing and FLAC decoding. The intended audience for this document is the development community who wish to use the FLAC decoder in their systems.

1.4 References

1.4.1 Standards

- FLAC official website: http://flac.sourceforge.net/index.html

1.4.2 General References

- A. J. Robinson for his work on Shorten; FLAC trivially extends and improves the fixed predictors, LPC coefficient quantization, and Rice coding used in Shorten.
- S. W. Golomb and Robert F. Rice; their universal codes are used by FLAC's entropy coder.
- N. Levinson and J. Durbin; the reference codec uses an algorithm developed and refined by them for determining the LPC coefficients from the autocorrelation coefficients

1.4.3 Freescale Multimedia References

- FLAC Decoder Application Programming Interface – flac_dec_api.doc
- FLAC Decoder Requirements Book - flac _dec_reqb.doc
- FLAC Decoder Test Plan - flac _dec_test_plan.doc
- FLAC Decoder Release notes - flac _dec_release_notes.doc
- FLAC Decoder Test Results – flac _dec_test_results.doc
- FLAC Decoder Performance Results – flac _dec_perf_results.doc
- FLAC Decoder Interface Header – flac_dec_interface.h

- FLAC Decoder Application Code – flac _dec_api.c

1.5 Definitions, Acronyms, and Abbreviations

TERM/ACRONYM	DEFINITION
API	Application Programming Interface
ARM	Advanced RISC Machine
DAC	Digital to Analog Converter
FSL	Freescale
IEC	International Electro-technical Commission
ISO	International Standards Organization
LC	Low Complexity
OS	Operating System
PCM	Pulse Code Modulation
RVDS	ARM RealView Development Suite
TBD	To Be Determined
UNIX	Linux PC x/86 C-reference binaries
FLAC	Free Lossless Audio Codec
LPC	Linear Predicting Coding

1.6 Document Location

docs/flac_dec

2 Release History

RELEASE NUMBER	DELIVERABLES	FEATURES
1.0	• Header file for the encoder (flac_dec_interface.h) • Example Application (Name: flac_dec_test.c) • ELINUX and RVDS libraries and test applications • UNIX/Linux x/86 Reference library and test application • Makefiles and Source code for library and test application including optimized assembler for the ELINUX and RVDS libraries. • Test vectors • Project files for RVDS and ELINUX	• Supports 8kHz, 16kHz, 22.05kHz, 24kHz, 32kHz, 44.1kHz, 48kHz, 88.2kHz, 96kHz, 176.4kHz, 192kHz sampling rates • Supports maximum 8 channels

Table 1. Details of the release

2.1 Assumptions and Known Problems

None

2.2 Contacts

Please report any problems to the following email address: mmsw@freescale.com

3 List of Deliverables

3.1 Documentation

Base directory: /vobs/multimedia_codecs/

Subdirectory	**Files**
docs/flac_dec	flac_dec_api.doc flac_dec_reqb.doc flac_dec_test_plan.doc flac_dec_test_results.doc flac_dec_perf_results.doc flac_dec_release_notes.doc

3.2 Public Headers

Base directory: /vobs/multimedia_codecs/

Subdirectory	**File**
ghdr/flac_dec	flac_dec_interface.h

3.3 Test Application Source

Base directory: /vobs/multimedia_codecs/test/

Subdirectory	**Files**
flac_dec/	"Makefile" makefile for building RVDS, UNIX and ELINUX board executable.
flac_dec/c_src	*.c, application code.
flac_dec/hdr	*.h, application header files

3.4 Library Source

Base directory: /src/

Subdirectory	**Files**
library	Makefile "Makefile" for building RVDS, UNIX, and ELINUX libraries. lib_flac_dec_arm11_lervds.a - Special options for ARM11 simulator testing lib_flac_dec_arm11_elinux.a - static library for ARM11 board lib_flac_dec_arm11_elinux.so - shared library for ARM11 board lib_flac_dec_x86_unix.a - library for Linux x/86 c reference code
flac_dec/c_src	*.c, FLAC decoder source code
flac_dec/asm_arm	*.s assembly source

flac_dec/hdr	*.h, FLAC decoder library header files

3.5 Common Makefiles

Base Directory: /vobs/multimedia_codecs/build

Makefile	Description
Makefile.init	This is a common makefile included in the codec library makefile for building the libraries. This file includes common options used by all codecs. Following flags can be overwritten or added to in the codec library makefile 1. Path to toolchain tools (TOOLS_DIR) 2. Header file path 3. Library path (SYS_INCLUDE) 4. Endian Flags 5. Optimization Flags(OPTIM_LEVEL, OPTIM_TYPE) 6. Common options for RVDS,UNIX and ELINUX (CFLAGS,AFLAGS) 7. Build specific flags 8. Source directory of 'C' code 9. Source directory of 'assembly(.s)' code 10. Object directory for .o files 11. RVDS Compilation Tools 12. Codec header path 13. Arguments for librarian for UNIX builds 14. SHARED_ELINUX builds for libraries that must be linked using the toolchain because of external library includes.
Makefile_test.init	This is the common makefile included in the codec test makefile for building the test application. This file includes the common options used by the all the codecs. Following flags can be overwritten or added to in the codec test makefile 1. Toolchain path depending on the build option 2. Compiler Flags 3. Linker flags 4. Paths for c_source, exe and object directories 5. Codec header files' INCLUDES path 6. Endian Flags

3.6 Test Vectors

Base Directory: /vobs/multimedia_vectors/flac_dec

Subdirectory	Files

flac_dec/input	Raw test vectors

4 Software Setup & Tools used

- ARM RVDS 3.0 (build 586) should be installed in the PC.
- Freescale Linux OS Release L26.1.16 must be running on the evaluation board.
- Intel based Red Hat Linux Machine must have the Montavista toolchain installed on it.
    - MontaVista 3.4.3-25.0.36.0501313 2005-08-21
- ‘make’ utility available for targeted platforms

5 Build Procedure

All the required makefiles are provided under individual directories. The library can be built for windows / target processor (ARM1136J-S). The details for the build procedure are described below.

5.1 Library

To build the library, run ‘make’ on ‘Makefile’ from library directory. The makefile shall create the required directory to hold the object files. The makefile can be used if you want to build the library only. The same makefile can used to build libraries for both board, Unix/Linux and RVDS with different build options. The following options are available to build the library.

Options

a) **BUILD options**:

a. **BUILD= ARM11ELINUX** : This option builds static library ‘lib_flac_dec_arm11_elinux.a’ for testing on the board.

b. **BUILD= ARM11ELINUXDLIB** : This builds shared library ‘lib_flac_dec_arm11_elinux.so’ for testing on the board.

c. **BUILD=ARM11LERVDS**: This option builds the static library ‘lib_flac_dec_arm11_lervds.a’, for testing on RVDS (Armulator).

d. **BUILD=UNIX**: This option builds the static library ‘lib_flac_dec_x86_unix.a’, for testing on UNIX/Linux machine.

Eg:

```
make BUILD=ARM11ELINUX
make BUILD=ARM11ELINUXDLIB
make BUILD=ARM11LERVDS
make BUILD=UNIX
```

b) **clean options**:

- **clean :** Deletes all the object files and libraries. To be used with the BUILD option.

Note: Make appropriate changes in file ‘makefile.init’ at directory ‘/vobs/multimedia_codecs/build’ for the location of toolchains.

The library that is built is saved as lib_flac_dec_arm11_LERVDS.a for LERVDS build, and lib_flac_dec_arm11_ELINUX.a and lib_flac_dec_arm11_ELINUX.so for board build, and lib_flac_dec_x86_unix.a is for UNIX/Linux machine. These libraries are saved in the current directory (the same directory in which the source and assembly directories are listed).

Target	Compilation Environment	Build Options	Library Name

Board	Redhat Linux Machine	BUILD=ARM11ELINUX
		BUILD=ARM11ELINUXDLIB
			lib_flac_dec_arm11_ELINUX.a
			lib_flac_dec_arm11_ELINUX.so

RVDS	Redhat Linux Machine	BUILD=LERVDS	lib_flac_dec_arm11_LERVDS.a

Unix/ Linux	Unix/Linux machine	BUILD=UNIX	lib_flac_dec_x86_unix.a

5.2 Test Application

To build the test application, run ‘make’ on ‘Makefile’ from the test directory. This makefile can create executables for testing on both board and RVDS for ARM11. The executables test_flac_dec_arm11_RVDS for ARM11 RVDS, test_flac_dec_arm11_ELINUX for ARM11 board, test_flac_dec_x86_unix for UNIX/Linux are stored under test/exe directory. The makefile shall create the required directory structure to hold the object files and executables. The following commands should be invoked so as to build the executables.

Options

1) **BUILD options**:
   - **BUILD=ARM11ELINUX**: This is the default option and builds the executable ‘test_ flac_dec_arm11_elinux’, for the board.
   - **BUILD=ARM11LERVDS**: This option builds the executable ‘test_flac_dec_arm11_lervds’ for the RVDS (Armulator).

   - **BUILD=UNIX**: This option builds the executable ‘test_flac_dec_x86_unix’ for the Unix/Linux machine.

     Eg:

     ```
     make BUILD=ARM11ELINUX (for ARM11 board)
     make BUILD=ARM11LERVDS       (for Armulator)
     make BUILD=UNIX      (for Unix/Linux machine)
     ```

2) **LIBRARY options**:
   - **LIB= STATIC**: This option builds the ELINUX test application linked with the ELINUX static library ‘lib_flac_dec_arm11_ELINUX.a’.If nothing is specified ,the executable links with shared library ‘lib_flac_dec_arm11_ELINUX.so’

     Eg: make LIB=STATIC

3) **clean options**:
   - **clean** : Deletes all the object files and executables. To be used along with build options.

Note:

In ‘makefile_test.init’ at directory ‘/vobs/multimedia_codecs/build’, the paths for the compiling and linking tools are hard coded for the current set-up. These paths may not be the same in the user’s directory set up. Hence, the ‘makefile_test.init’ should be modified to point to the directories where the linking and compilation tools are present before building the application for board.

The following table summarises the build options,

Target	**Compilation Environment**	**Build Options**	**Executable Name**
Board	Redhat Linux Machine	BUILD=ARM11ELINUX	test_flac_dec_arm11_elinux

		LIB= STATIC	
ARM11RVDS	PC (Using Cygwin)	BUILD=ARM11LERVDS	test_flac_dec_arm11_lervds
UNIX/Linux	Unix/Linux machine	BUILD=UNIX	test_ flac_dec_x86_unix

6 Test Application Execution

6.1 Scripts

TBD

6.2 ELINUX

```
./test_flac_dec_arm11_elinux <input vector>  <output vector>
```

The output vector will be placed into file <output vector>.

6.3 RVDS

Please refer ARM documentation regarding loading the image and configuring the RVDS debugger for ARM1136J-S

- RVDS :
  Once the image is loaded press "*F5*" or select the pull down menu option "*Debug -> run*" to run the loaded image.

6.4 UNIX Reference

To execute on Linux x/86 type:

```
./test_flac_dec_x86_unix <input vector>  <output vector>
```

7 Pre compilation Options

The following C options need to be set

7.1 Test Application compilation Options

The following C options need to be set

Application	Description	Remarks
• DISCARD_OUTPUT	To disable dumping of output.	All builds except UNIX
• TEST_PERFORMANCE	To disable dumping of output while taking performance	All builds except UNIX
• TIME_PROFILE	To enable performance test	Only for ELINUX
• CCM_MHZ_MEASURE	To enable performance test	Only for RVDS

7.2 Library compilation options

Library	Desciption	Remarks
• BUILD_INT_FLOAT • __GNUC__	Always Enabled	All Builds
•	Enabled to use optimized C and ASM code	ARM Platform
• __WINCE	Disabled for ARM Platform	All Builds