06-3737-RN-TX30
MARCH 9, 2006
1.0

Release Notes for Ogg Vorbis Decoder on ARM11 ELINUX

ABSTRACT:

Release Notes for Ogg Vorbis Decoder on ARM11 ELINUX

KEYWORDS:

Multimedia codecs, Ogg Vorbis, Audio

Revision History

VERSION	DATE	AUTHOR	CHANGE DESCRIPTION
0.1	02-Jan-2006	Anand Narayanan	Initial version
0.2	18-Jan-2006	Anand Narayanan	Updated with relocation and reentrancy changes
1.0	06-Feb-2006	Lauren Post	Using new format
2.0	29-Dec-2008	Lyon Wang	Support Vorbis raw data decode
2.1	31-Mar-2009	Guo Yue	Add new build option BUILD= ARM11ELINUXDLIB

Table of Contents

Introduction 4
1.1 Purpose 4
1.2 Scope 4
1.3 Audience Description 4
1.4 References 4
1.4.1 Standards 4
1.4.2 Freescale Multimedia References 4
1.5 Definitions, Acronyms, and Abbreviations 5
1.6 Document Location 5
2 Release History 6
2.1 Assumptions and Known Problems 6
2.2 Contacts 6
3 List of Deliverables 7
3.1 Documentation 7
3.2 Public Headers 7
3.3 Test Application Source 7
3.4 Library Source 7
3.5 Common Makefiles 8
3.6 Test Vectors 8
4 Software Setup & Tools used 10
5 Build Procedure 11
5.1 Library 11
5.2 Test Application 12
6 Test Application Execution 15
6.1 Scripts 15
6.2 ELINUX 15
6.2.1 Re-entrancy 15
6.3 RVDS 16
6.4 UNIX Reference 16
7 Pre compilation Options 17
Compiler switches for Library 17
Compiler switches for Test Application 17

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 Linux x86.

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 Compression and Vorbis decoding.

1.4 References

1.4.1 Standards

- Vorbis I Specification

1.4.2 Freescale Multimedia References

- Ogg Vorbis Decoder Application Programming Interface – oggvorbis_dec_api.doc
- Ogg Vorbis Decoder Requirements Book - ogg vorbis_dec_reqb.doc
- Ogg Vorbis Decoder Test Plan – oggvorbis_dec_test_plan.doc
- Ogg Vorbis Decoder Release notes – oggvorbis_dec_release_notes.doc
- Ogg Vorbis Decoder Test Results – oggvorbis_dec_test_results.doc
- Ogg Vorbis Decoder Performance Results – oggvorbis_dec_perf_results.doc
- Ogg Vorbis Decoder Interface Header – oggvorbis_dec_api.h, oggvorbis_dec_os_types.h Ogg Vorbis Decoder Application Code – ivorbisfile_example.c

1.5 Definitions, Acronyms, and Abbreviations

TERM/ACRONYM	DEFINITION
API	Application Programming Interface
ARM	Advanced RISC Machine
FSL	Freescale
MDCT	Modified Discrete Cosine Transform
OS	Operating System
PCM	Pulse Code Modulation
RVDS	ARM RealView Development Suite
TBD	To Be Determined
UNIX	Linux PC x/86 C-reference binaries

1.6 Document Location

docs/oggvorbis_dec

2 Release History

RELEASE NUMBER	DELIVERABLES	FEATURES
1.0	• Documentation • Application Interface header file • 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	• Engineering Release • Assembly optimized library for Arm11
1.02	*BLN_MAD-MMCODECS_OGGVORBISD_ARM11_01.02.00* • Documentation • Application Interface header file • ELINUX and RVDS libraries and test applications • Makefiles and Source code for library and test application including optimized assembler for the ELINUX and RVDS libraries.	• Engiering Release • For Vorbis raw data input decoding • Support push mode

Table 1. Details of the release

2.1 Assumptions and Known Problems

None

2.2 Contacts

Please report any problems to Freescale customer representative

3 List of Deliverables

3.1 Documentation

Base directory: /fsl_mad_multimedia_codec/

Subdirectory	**Files**
docs/oggvorbis_dec	oggvorbis_dec_api.doc oggvorbis_dec_reqb.doc oggvorbis_dec_test_plan.doc oggvorbis_dec_test_results.doc oggvorbis_dec_perf_results.doc oggvorbis_dec _release_notes.doc

3.2 Public Headers

Base directory: /fsl_mad_multimedia_codec/

Subdirectory	**File**
ghdr	oggvorbis_dec_api.h oggvorbis_dec_os_types.h

3.3 Test Application Source

Base directory: /fsl_mad_multimedia_codec/

Subdirectory	**Files**
test/oggvorbis	"Makefile" makefile for building RVDS, UNIX and ELINUX board executables.
test/oggvorbis/c_src	*.c, application code.

3.4 Library Source

Base directory: /fsl_mad_multimedia_codec/

Subdirectory	**Files**
src/oggvorbis_dec	Makefile "Makefile" for building RVDS, UNIX, and ELINUX libraries. liboggvorbis_dec_arm11_lerves.a – library for testing on simulator liboggvorbis_dec_arm11_elinux.a - static library for board liboggvorbis_dec_arm11_elinux.so – shared library for board liboggvorbis_dec_UNIX.a – library for Linux x/86 – c reference code
src/oggvorbis_dec/c_src	*.c, Ogg Vorbis decoder source code
src/oggvorbis_dec/hdr	*.h, Ogg Vorbis decoder library header files

3.5 Common Makefiles

Base Directory: /fsl_mad_multimedia_codec/build

Subdirectory	Files
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 (TC_ROOT) 2. GNU header file path (HEADER_PATHS) 3. GNU library path (LIB_PATHS) 4. GNU Compiler/Assembler Options (GNU_CFLAGS, GNU_AFLAGS) 5. Endian Flags 6. Optimization Flags(OPTIM_LEVEL, OPTIM_TYPE) 7. Common options for RVDS,UNIX and ELINUX (CFLAGS,AFLAGS) 8. Build specific flags 9. Source directory of 'C' code 10. Source directory of 'assembly(.s)' code 11. Object directory for .o files 12. RVDS Compilation Tools 13. Codec header path 14. Arguments for librarian for UNIX builds 15. 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 7. CODEC_LIB generation

3.6 Test Vectors

Base Directory: multimedia_vectors/test_vectors

The test vectors are provided in another location from the library and test source.

Subdirectory	Files
oggvorbis_dec/input	All .ogg streams Input vectors
oggvorbis_dec/ref	Reference pcm files to compare against

4 Software Setup & Tools used

- ARM RVDS 2.2 (build 503) should be installed in the PC.
- Freescale Linux OS Release L26.1.15 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
- ‘Cygwin’ **Version** CYGWIN_NT-5.1, a freely downloadable linux emulator is installed in PC - **http://www.cygwin.com/**.
- ‘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 liboggvorbis_dec_arm11_elinux.a' for testing on the board.

b. **BUILD=ARM11ELINUXDLIB** : This option builds shared library liboggvorbis_dec_arm11_elinux.so' for testing on the board.

c. **BUILD=ARM11LERVDS**: This option builds the static library 'liboggvorbis_dec_arm11_lervds.a', for testing on RVDS (Armulator).

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

Eg:

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

e. Note: When support OGG container decoder, please add RAW_DATA=0 in command line. or by default decoder is RAW DATA INPUT mode without set RAW_DATA=0

Eg:

```
make BUILD= ARM11ELINUX RAW_DATA=0
make BUILD= ARM11ELINUXDLIB RAW_DATA=0
make BUILD=ARM11LERVDS RAW_DATA=0
make BUILD=UNIX RAW_DATA=0
```

b) **ENDIAN options for RVDS**:

- **TARGET_ENDIAN=LITTLE**: This is the default option and sets the endian-ness to 'little'
- **TARGET_ENDIAN=BIG**: This option sets the endian-ness to big
  - **Eg**: make BUILD=ARM11LERVDS TARGET_ENDIAN=BIG

c) **clean options**:

- **clean_ARM11RVDS**: Deletes all the object files and the RVDS library 'liboggvorbis_dec_arm11_lervds.a'.
- **clean_ARM11ELINUX**: Deletes all the object files and the ELINUX libraries liboggvorbis_dec_arm11_elinux.a and liboggvorbis_dec_arm11_elinux.so.
- **clean_UNIX**: Deletes all the object files and the UNIX library 'liboggvorbis_dec_UNIX.a'.
- **clean**: Deletes all the object files and RVDS,UNIX and ELINUX libraries.

Note: Make appropriate changes in file 'Makefile.init at directory '/build' for the location of toolchains.

The library that is built is saved as liboggvorbis_dec_arm11_lervds.a for RVDS build, and liboggvorbis_dec_arm11_elinux.a and liboggvorbis_dec_arm11_elinux.so for board build. 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**

RVDS	PC (Using Cygwin)	BUILD=ARM11RVDS
		TARGET_ENDIAN= BIG/LITTLE
			Lib_oggvorbis_dec_arm11_lervds.a

Board	PC (Using Cygwin)	BUILD= ARM11ELINUX
		BUILD= ARM11ELINUXDLIB
			Lib_oggvorbis_dec_arm11_elinux.a
			lib_oggvorbis_dec_arm11_elinux.so

Unix/ Linux	Linux/Unix machine	BUILD=UNIX
		TARGET_ENDIAN= BIG/LITTLE
			Lib_oggvorbis_dec_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 oggvorbis_dec_arm11_lervds for RVDS, oggvorbis_dec_arm11_elinux for board and oggvorbis_dec_UNIX for UNIX 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.

Note: add –DRAW_DATA_INPUT in the test/oggvorbis_dec/Makefile when vorbis raw data input.

Options

1) **BUILD options**:
   - **BUILD=ARM11ELINUX**: This is the default option and builds the executable 'oggvorbis_dec_arm11_elinux, for the board.

- **BUILD=ARM11LERVDS**: This option builds the executable 'oggvorbis_dec_arm11_lervds for the RVDS (Armulator).

- **BUILD=UNIX**: This option builds the executable 'oggvorbis_dec_UNIX' for the Unix/Linux machine.

  Eg:
  ```
  make BUILD=ARM11ELINUX          (for board)
  make BUILD=ARM11LERVDS          (for Armulator)
  make BUILD=UNIX                 (for Unix/Linux machine)
  ```
- **Note: When support OGG container decoder, please add RAW_DATA=0 in command line. or by default decoder is RAW DATA INPUT mode without set RAW_DATA=0**

  Eg:
  ```
  make BUILD= ARM11ELINUX RAW_DATA=0
  make BUILD=ARM11LERVDS RAW_DATA=0
  make BUILD=UNIX RAW_DATA=0
  ```

2) **ENDIAN options for RVDS**:
   - **TARGET_ENDIAN=LITTLE**: This is the default option and sets the endian-ness to 'little'

   - **TARGET_ENDIAN=BIG**: This option sets the endianess to big

     Eg: make BUILD=ARM11BERVDS

3) **LIBRARY options**:
   - **LIB= STATIC**: This option builds the ELINUX test application linked with the ELINUX static library 'liboggvorbis_dec_arm11_ELINUX.a'.If nothing is specified ,the executable links with shared library 'liboggvorbis_dec_arm11_ELINUX.so'

     Eg: make LIB=STATIC

4) **PROFILE options**:
   TIME_PROFILE=1 is used to get cycle measurement information.
5) **clean options**:
   - **clean_RVDS**: Deletes all the object files and the RVDS executable 'oggvorbis_dec_arm11_RVDS' .
   - **clean_ELINUX**: Deletes all the object file and the ELINUX 'oggvorbis_dec_arm11_ELINUX'.
   - **clean_UNIX**: Deletes all the object files and the Unix/Linux executable 'oggvorbis_dec_UNIX'.
   - **clean**: Deletes all the object files and RVDS,UNIX ELINUX executables.

Note:

In 'common_testapp.mk' at directory 'ARM11/common', 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 'common_testapp.mk' 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=ELINUX
		LIB= STATIC
			test_oggvorbis_dec_arm11_elinux

RVDS	PC (Using Cygwin)	BUILD=RVDS
		TARGET_ENDIAN= LITTLE/BIG
			test_oggvorbis_dec_arm11_lervds

UNIX/ Linux	Unix/Linux machine	BUILD=UNIX
		TARGET_ENDIAN= LITTLE/BIG
			test_oggvorbis_dec_x86_unix

6 Test Application Execution

To know the options provided by the test application, run the executable without any argument. It shall print a brief summary of all the options available.

6.1 Scripts

In the test/test_util/scripts directory, a script file exists for doing batch processing on several vectors. The script can be modified or parameters set to specify the binaries to use.

6.2 ELINUX

To run the ELINUX executable (SCMA11):

```
oggvorbis_dec_arm11_elinux  <infile> <outfile> <logfile>
```

Where:

<infile> is the input test vector (ogg bitstream).
<outfile> is the output file name (pcm format).
<logfile> is the file in which the decoder dumps the comments about in the stream header, or, dumps the error if any.

6.2.1 Re-entrancy

To run the ELINUX executable for testing reentrancy on board:

```
oggvorbis_dec_arm11_ELINUX <infile1> <outfile1> <logfile1> <infile2> <outfile2>
<logfile2>
```

Note: Here the tester is modified to accommodate two threads.

Where:

<infile1> is the first input test vector (ogg bit stream).
<outfile1> is the first output file name (pcm format).
<logfile1> is the file in which the decoder dumps the comments about the first inputs stream header, or, dumps the error if any.
<infile2> is the second input test vector (ogg bit stream).
<outfile2> is the second output file name (pcm format).
<logfile2> is the file in which the decoder dumps the comments about the second inputs stream header, or, dumps the error if any.

Note: The exe name infile and outfile in the above syntax should be specified along with their absolute or relative path from the current directory.

6.3 RVDS

To run the RVDS executable (armsd):

```
armsd –cpu arm1136j-s oggvorbis_dec_arm11_RVDS <infile> <outfile> <logfile>
```

6.4 UNIX Reference

To execute on Linux x/86 type:

```
oggvorbis_dec_UNIX <infile> <outfile> <logfile>
```

7 Pre compilation Options

The following C options need to be set

Compiler switches for Library

C Defines	Description	Comments
ARM_ADS	User defined data types	Should be defined for compiling for RVDS.
ARM_ASM	To enable or disable optimizations.	Optimization will be enabled if it is defined.

Compiler switches for Test Application

Switch	Description	Comments
ARM_ADS		RVDS builds only
CYCLE_MEASUREMENT	RVDS cycle measurement.	Gets the core cycles on Armulator dumped in a file if defined.
LOG_TIMING	Board microsecond measurement.	Gets the micro seconds on Board dumped in a file if defined.
STACK_USAGE	Measurements of peak stack usage.	Prints peak stack usage value if defined.
REENTRANCY	Tests the code for reentrancy	For testing the reentrancy of the decoder.
RELOCATION	Tests the code for relocation	For performing the relocation test on the decoder.