08-6403-RN-ZCH66
APR 11, 2008
3.4

Release Notes for MPEG-4 AAC LC Decoder on ARM11 ELINUX

ABSTRACT:

Release Notes for MPEG-4 AAC LC Decoder on ARM11 ELINUX

KEYWORDS:

Multimedia codecs, AAC, Audio

APPROVED:

Shang Shidong

Revision History

VERSION	DATE	AUTHOR	CHANGE DESCRIPTION
1.0	20-Sep-2004	Vishalakshi	Final release 1.0
1.1	14-Mar-2005	Vishalakshi	Release 1.1 after testing on ARM11 Linux hardware
1.2	16-Jun-2005	Ashok Kumar	Release 1.2 after fixing BUS ERROR and PNS issue. This release also contains new set of test vectors for PNS.
2.0	14-Nov-2005	Ashok Kumar	Updated for final release 2.0 with all the bug fixes
2.1	18-Nov-2005	Sukruth	Updated with the build procedure for RVDS 2.2
3.0	06-Feb-2006	Lauren Post	Using new format
3.1	06-Jun-2007	Ananta Kar	Added build procedures for ARM9 (RVDS and ELINUX)
3.2	21-Nov-2007	Bing Song	Update for v1.00.01
3.3	09-Apr-2008	Wangshengjiu	ADD BSAC
3.4	15-Oct-2008	Haiting Yin	Remove BSAC

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 Linux x86. AAC Decoder provides decode ability for ADTS and ADIF input streams.

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, AAC Decoding.

1.4 References

1.4.1 Standards

- ISO/IEC 13818-7:1997 Information technology -- Generic coding of moving pictures and associated audio information -- Part 7 (popularly known as *MPEG-2 AAC*)
- ISO/IEC 13818-4:1997 Information technology -- Generic coding of moving pictures and associated audio information -- Part 4 (compliance testing)
- ISO/IEC 14496-3:1999 Information technology – Coding of audio visual objects -- Part 3 (audio)

1.4.2 General References

- Ted Painter and Andreas Spanias, "Perceptual Coding of Digital Audio", Proc. IEEE, vol-88, no.4, April 2000
- H.S.Malvar, "Lapped transforms for efficient subband/transform coding", IEEE trans. ASSP, June 1990.
- Seymour Shlien, "The Modulated Lapped Transform, Its Time-Varying Forms and Its Applications to Audio Coding Standards."
- "A Tutorial on MPEG/Audio compression" by Davis Pan

1.4.3 Freescale Multimedia References

- AAC Decoder Application Programming Interface – aac_dec_api.doc
- AAC Decoder Requirements Book - aac_dec_reqb.doc

- AAC Decoder Test Plan - aac_dec_test_plan.doc
- AAC Decoder Release notes - aac_dec_release_notes.doc
- AAC Decoder Test Results – aac_dec_test_results.doc
- AAC Decoder Performance Results – aac_dec_perf_results.doc
- AAC Decoder Interface header – aacd_dec_interface.h
- AAC Decoder Application Code – aac_main.c

1.5 Definitions, Acronyms, and Abbreviations

TERM/ACRONYM	DEFINITION
AAC	Advanced Audio Coding
ADIF	Audio_Data_Interchange_Format
ADTS	Audio_Data_Transport_Stream
API	Application Programming Interface
ARM	Advanced RISC Machine
FSL	Freescale
IEC	International Electro-technical Commission
ISO	International Standards Organization
LC	Low Complexity
MDCT	Modified Discrete Cosine Transform
MPEG	Moving Pictures Expert Group
OS	Operating System
PCM	Pulse Code Modulation
PNS	Perceptual Noise Substitution
UNIX	Linux PC x/86 C-reference binaries
RVDS	ARM RealView Development Suite
TBD	To be decided

1.6 Document Location

docs/aac_dec

2 Release History

RELEASE NUMBER	DELIVERABLES	FEATURES
1.0	• Engineering Release	• Initial Release
2.0	• Documentation • Application Interface header file for the decoder (aacd_dec_interface.h) • 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	• Supports all sampling rates from 8khz to 48 kHz) • Supports all bit rates specified by ISO • Stereo / Mono decoding • Tested on ARM11 Hardware • Added Debug log support
2.3	Same	• Shared Library Support • Upgrade to RVDS 2.2
2.4	Same	• Fixes for segmentation faults with invalid streams • Fixes for name conflicts combined with other audio codecs
2.5	Same	• Support BSAC decoder
2.6	Same	• Remove BSAC decoder

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/aac_dec	aac_dec_api.doc aac_dec_reqb.doc aac_dec_test_plan.doc aac_dec_test_results.doc aac_dec_perf_results.doc aac_dec _release_notes.doc

3.2 Public Headers

Base directory: / fsl_mad_multimedia_codec /

Subdirectory	**File**
ghdr	aacd_dec_interface.h

3.3 Test Application Source

Base directory: / fsl_mad_multimedia_codec /

Subdirectory	**Files**
test/ aac_dec	"Makefile" makefile for building RVDS, UNIX and ELINUX board executable.
test/aac_dec/c_src	*.c, application code.
utils/aac_dec	dif32.exe *(This dos executable can be used to compare the reference outputs with the decoder generated outputs.)* dif32_arm11 *(This linux executable can be used to compare the reference outputs with the decoder generated outputs on ARM11 Board platform.)* dif32_linux *(This linux executable can be used to compare the reference outputs with the decoder generated outputs on linux platform)* run_board_ftpget *(Script to run all test vectors on board)* readme.txt *(Instructions to use the script on the board)*

3.4 Library Source

Base directory: / fsl_mad_multimedia_codec /

Subdirectory	**Files**
src/aac_dec	Makefile "Makefile" for building RVDS, UNIX, and ELINUX libraries. lib_aac_dec_arm11_lervds.a – Special options for simulator testing lib_aac_dec_arm11_elinux.a - static library for board lib_aac_dec_arm11_elinux.so – shared library for board lib_aac_dec_unix.a – library for Linux x/86 – c reference code
src/aac_dec/c_src	*.c, AAC decoder source code
src/aac_dec/asm_arm	*.s assembly source
src/aac_dec/c_asm_arm	*.c with inline assembly source code
src/aac_dec/hdr	*.h, AAC decoder library header files

3.5 Common Makefiles

Base Directory: / fsl_mad_multimedia_codec/

Subdirectory	**Files**
build/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.

build/ 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 for AAC

Base Directory: /fsl_mad_multimedia_vectors/

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

Subdirectory	**Files**
aac_dec/input	Conformance test vectors for adif and adts
aac_dec/ref	Floating point reference output to bitmatch against

4 Software Setup & Tools used

- ARM RVDS 3.0 (build 441) should be installed in the PC.
- Freescale Linux OS Release L26.1.17 must be running on the evaluation board.
- Intel based Red Hat Linux Machine must have the **devtek** toolchain installed on it.
    - **devtek Toolchain** `gcc 4.1.1 glibc 2.4 nptl 6`
- '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 for AAC

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 for AAC

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 is the default option and builds both static library ‘lib_aac _dec_arm11_elinux.a’ and shared library ‘lib_aac _dec_arm11_elinux.so’ , for testing on the board.

  b. **BUILD=ARM11LERVDS**: This option builds the static library ‘lib_aac _ dec _arm11_lervds.a’, for testing on RVDS (Armulator).

  c. **BUILD=UNIX**: This option builds the static library ‘lib_aac _ dec _unix.a’, for testing on UNIX/Linux machine.

Eg:

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

b) **clean options**:

  c) **clean**: Deletes all the object files and RVDS,UNIX and ELINUX libraries.

Note: Make appropriate changes in file ‘Makefile.init’ at directory ‘/fsl_mad_multimedia_codec*/build/*’ for the location of toolchains.

The library that is built is saved as lib_aac_dec_arm11_lervds.a for RVDS build, and lib_aac_dec_arm11_elinux.a and lib_aac_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**
Board	PC (Using Cygwin)	BUILD= ARM11ELINUX	lib_aac _dec _arm11_ elinux.a lib_aac _dec _arm11_ elinux.so
RVDS	PC (Using Cygwin)	BUILD=ARM11RVDS	lib_aac _dec _arm11_lervds.a

Unix/ Linux	Linux/Unix machine	BUILDUNIX	lib_aac _dec _unix.a

5.2 Test Application for AAC

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_aac_dec_arm11_lervds for RVDS, aac_dec_arm11_ELINUX for board and test_aac_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.

Options

1) **BUILD options**:
    2) **BUILD=ARM11ELINUX**: This option builds the executable 'lib_aac_dec _arm11_ elinux', for the board.

    3) **BUILD=ARM11RVDS**: This option builds the executable 'lib_aac_dec _arm11_lervds' for the RVDS (Armulator).

    4) **BUILD=UNIX**: This option builds the executable 'lib_aac_dec _unix' for the Unix/Linux machine.

    Eg: make BUILD=ARM11ELINUX (for board)
    make BUILD= ARM11RVDS (for Armulator)
    make BUILD=UNIX (for Unix/Linux machine)

5) **LIBRARY options**:
    - **LIB= STATIC**: This option builds the ELINUX test application linked with the ELINUX static library 'lib_aac_dec_arm11_ elinux.a'.If nothing is specified ,the executable links with shared library 'lib_aac_dec_arm11_ elinux.so'
      Eg: make LIB=STATIC

6) **PROFILE options**:
TIME_PROFILE=1 is used to get cycle measurement information.
7) **clean options**:
    - **clean**: Deletes all the object files and RVDS,UNIX ELINUX executables.

Note:
In 'Makefile_test.init' at directory '/fsl_mad_multimedia_codec*/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
		LIB= STATIC
			test_aac_dec _arm11_elinux

RVDS	PC (Using Cygwin)	BUILD=ARM11LERVDS	test_aac_dec _dec_arm11_lervds
UNIX/ Linux	Unix/Linux machine	BUILD=UNIX	test_aac_dec _dec_unix

6 Test Application Execution

6.1 ELINUX Single Vector

test_aac_dec_arm11_elinux <input vector> <output vector>

The output vector will be placed into files <output_vector>_f0<X> where X starts with 1as the first channel and will increment for each additional channel in the input vector.

6.2 ELINUX Multiple Vectors

To run single vector / multiple test vectors:

Open the file "aac_main.c". To run a given test vector initialize a new char array as

```
char TestVectors[][AACD_PATH_LEN] = {"<test_vec_name>", "_end_of_vectors_"};
```

This needs to be done after defining the compile time macro "SINGLE_VECTOR". If all the test vectors need to be run, undefine the compile time macro "SINGLE_VECTOR"
Set the path of the test vector required to be tested by initializing the character array "in_path". Similarly set the output path where the output PCM files needs to be written by initializing the array "out_path"

E.g. the default path set is as follows:
char in_path[] = "..\\..\\input"; for input &
char out_path[] = "..\\..\\out"; for output.

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 -> Execution Contro*l" to run the loaded image.

The single test vector can be run by compiling the code with the compile time macro "SINGLE_VECTOR". To run multiple test vectors at the same time disable the compile time macro SINGLE_VECTOR and include all the strings of the vectors need to be tested. The output PCM samples will be generated in the directory specified in the out_path.

6.4 UNIX Reference

To execute on Linux x/86 type:

test_aac_dec_unix <input vector> <output vector>