freescale semiconductor

08-6470-RN-ZCH66

MARCH 9, 2006

4.0

Release Notes for GIF Decoder on ARM11 ELINUX

ABSTRACT:

Release Notes for GIF Decoder on ARM11 ELINUX

KEYWORDS:

Multimedia codecs,Image, GIF

APPROVED:

Wang Zening

Revision History

Version	Date	Author	Change Description
1.0	27-Dec-2004	Shailesh R, Sameer PR	Release 1.0 of GIF Decoder on ARM11 RealView Simulator platform
2.0	01-Apr-2005	Shailesh R, Sameer PR	Release 2.0 of GIF Decoder on board with ELINUX
3.0	01-Sep-2005	Puneet Gulati	Build Procedure changes for RVDS2.2
4.0	06-Feb-2006	Lauren Post	Using new format

Table of Content

Introduction **4**
1.1 Purpose 4
1.2 Scope 4
1.3 Audience Description 4
1.4 References 4
1.4.1 References 4
1.4.2 Freescale Multimedia References 4
1.5 Definitions, Acronyms, and Abbreviations 4
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 **14**
6.1 Scripts 14
6.2 ELINUX 14
6.3 RVDS 14
6.4 UNIX Reference 14
7 Pre compilation Options **15**

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 GIF decoding.

1.4 References

1.4.1 References

- Compressed Image File formats by John Miano, ACM Press/Addison Wesley Longman.

1.4.2 Freescale Multimedia References

- GIF Decoder Application Programming Interface - gif_dec_api.doc
- GIF Decoder Requirements Book - gif_dec_reqb.doc
- GIF Decoder Test Plan - gif_dec_test_plan.doc
- GIF Decoder Release notes - gif_dec_release_notes.doc
- GIF Decoder Test Results – gif_dec_test_results.doc
- GIF Decoder Performance Results – gif_dec_perf_results.doc
- GIF Decoder Interface Header – gif_def_interface.h
- GIF Decoder Application Code – gif_test.c

1.5 Definitions, Acronyms, and Abbreviations

TERM/ACRONYM	DEFINITION
API	Application Programming Interface

ARM	Advanced RISC Machine
GIF	Bitmap
FSL	Freescale
IEC	International Electro-technical Commission
ISO	International Organization for Standardization
OS	Operating System
RGB	Raw pixel data organized in the order of Red, green and blue components. RGB888 denotes 8 bits per pixel each for R, G and B components
RVDS	ARM RealView Development Suite
TBD	To Be Determined
UNIX	Linux PC x/86 C-reference binaries

1.6 Document Location

docs/gif_dec

2 Release History

RELEASE NUMBER	DELIVERABLES	FEATURES
1.0	• Documentation • Application Interface header file for the decoder (gif_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	• Initial Release
2.1	• Same	• Shared Library Support • Upgrade to RVDS 2.2

Table 1. Details of the release

2.1 Assumptions and Known Problems

- Suspension: It is assumed that the global header data is made available in its entirety to the APIs GIF_query_dec_mem() and GIF_decoder_init(). Suspension can take place when new data is requested from GIF_query_dec_mem_frame(), GIF_decoder_init_frame or GIF_decode()

2.2 Contacts

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

3 List of Deliverables

3.1 Documentation

Base directory: /ARM11/

Subdirectory	Files
docs/gif_dec	gif_dec_api.doc gif_dec_reqb.doc gif_dec_test_plan.doc gif_dec_test_results.doc gif_dec_perf_results.doc gif_dec _release_notes.doc

3.2 Public Headers

Base directory: /ARM11/

Subdirectory	File
API_include	gif_interface.h

3.3 Test Application Source

Base directory: /ARM11/src/image/gif_dec

Subdirectory	Files
test/	"Makefile" makefile for building RVDS, UNIX and ELINUX board executables.
test/c_source	*.c application code.
test/test_util/scripts	Batch files to be run on the board

3.4 Library Source

Base directory: /ARM11/src/image/gif_dec

Subdirectory	Files
library	Makefile "Makefile" for building RVDS, UNIX, and ELINUX libraries. libgif_dec_arm11_RVDS.a – Special options for simulator testing libgif_dec_arm11_ELINUX.a - static library for board libgif_dec_arm11_ELINUX.so – shared library for board libgif_dec_UNIX.a – library for Linux x/86 – c reference code
library/c_source	*.c, GIF decoder source code
library/include	*.h, GIF decoder library header files

3.5 Common Makefiles

Base Directory: /ARM11/common

Makefile	Description
common.mk	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.
common_testapp.mk	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
gif_dec/input /sample_test_cases	*. gif – input files to decoder, in the form of gif code-stream
gif_dec/ref/sample_test_cases	Reference outputs for the *.gif files in the sample test cases – the reference vectors are the decoded output in the form of ppm files[1]
gif_dec/ref/sample_test_cases /big_end_ref_565_555	Reference vectors for 16-bit output format in the specific case of big-endian organization of data

[1] The raw decoded output is formatted in the ppm format and can be viewed using a tool like IrfanView (www.irfanview.com)

4 Software Setup & Tools used

- ARM RVDS 2.2 should be installed in the PC.
- Freescale Linux OS Release L26.1.15 or higher 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**:
   - **BUILD= ELINUX** : This is the default option and builds both static library 'libgif_dec_arm11_ELINUX.a' and shared library 'libgif_dec_arm11_ELINUX.so' , for testing on the board.
   - **BUILD=RVDS**: This option builds the static library 'libgif_dec_arm11_RVDS.a', for testing on RVDS (Armulator).
   - **BUILD=UNIX**: This option builds the static library 'libgif_dec_UNIX.a', for testing on UNIX/Linux machine.

     Eg: make BUILD=ELINUX
     make BUILD=RVDS
     make BUILD=UNIX

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 –f Makefile BUILD=RVDS TARGET_ENDIAN=BIG

c) **clean options**:
   - **clean_RVDS**: Deletes all the object files and the RVDS library 'libgif_dec_arm11_RVDS.a'.
   - **clean_ELINUX**: Deletes all the object file and the ELINUX libraries libgif_dec_arm11_ELINUX.a and libgif_dec_arm11_ELINUX.so.
   - **clean_UNIX**: Deletes all the object files and the UNIX library 'libgif_dec_UNIX.a'.
   - **clean**: Deletes all the object files and RVDS,UNIX and ELINUX libraries.

Note: Make appropriate changes in file 'common.mk' at directory 'ARM11/common' for the location of toolchains.

The library that is built is saved as libgif_dec_arm11_RVDS.a for RVDS build, and libgif_dec_arm11_ELINUX.a and libgif_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).

Compilation Environment	Target	build options	library name
PC (Using Cygwin)	RVDS	BUILD=RVDS TARGET_ENDIAN=BIG/ LITTLE	libgif_dec_arm11_RVDS.a
PC (Using Cygwin)	Board	BUILD= ELINUX	libgif_dec_arm11_ELINUX.a, libgif_dec_arm11_ELINUX.so
Using Linux/Unix machine	Unix/ Linux	BUILD=UNIX TARGET_ENDIAN=BIG/ LITTLE	libgif_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 Linux x86, ARM11 board and RVDS for ARM11. The executables gif_dec_arm11_RVDS for RVDS, gif_dec_arm11_ELINUX for board and gif_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**:
   - **BUILD=ELINUX**: This is the default option and builds the executable 'gif_dec_arm11_ELINUX', for the board.
   - **BUILD=RVDS**: This option builds the executable 'gif_dec_arm11_RVDS' for the RVDS (Armulator).
   - **BUILD=UNIX**: This option builds the executable 'gif_dec_UNIX' for the Unix/Linux machine.

   Eg:
   ```
   make BUILD=ELINUX   (for board)
   make BUILD=RVDS     (for Armulator)
   make BUILD=UNIX     (for Unix/Linux machine)
   ```

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 endian-ness to big
  Eg: make BUILD=RVDS TARGET_ENDIAN=BIG

3) **LIBRARY options**:
   - **LIB= STATIC**: This option builds the ELINUX test application linked with the ELINUX static library 'libgif_dec_arm11_ELINUX.a'.If nothing is specified ,the executable links with shared library 'libgif_dec_arm11_ELINUX.so'
     Eg: make LIB=STATIC

4) **clean options**:
   - **clean_RVDS**: Deletes all the object files and the RVDS executable 'gif_dec_arm11_RVDS' .
   - **clean_ELINUX**: Deletes all the object file and the ELINUX 'gif_dec_arm11_ELINUX'.
   - **clean_UNIX**: Deletes all the object files and the Unix/Linux executable 'gif_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,

Compilation Environment	Target	Build options	executable name
PC (Using Cygwin)	RVDS	BUILD=RVDS TARGET_ENDIAN=LITTLE/BIG	gif_dec_arm11_RVDS
Redhat Linux Machine	ARM11 board	BUILD=ELINUX LIB= STATIC	gif_dec_arm11_ELINUX
PC (Using Unix/Linux machine)	UNIX/ Linux	BUILD=UNIX TARGET_ENDIAN=LITTLE/BIG	gif_dec_UNIX

6 Test Application Execution

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

gif_dec_arm11_ELINUX <input vector> <output vector> <options>

To know the options available, excecute the application without any arguments.

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.

6.4 UNIX Reference

To execute on Linux x/86 type:

gif_dec_UNIX <input vector> <output vector> <options>

7 Pre compilation Options

The following C options need to be set

C Defines		Remarks
LOG_TIMING	To log performance timing results	