i.MX6 series Firmware

1 Introduction

The purposes of the firmware library for the i.MX6 series are to:

- Provide example driver code that enables main features of a peripheral without the need for an operating system.
- Provide example usage for each driver to demonstrate their main features.
- Provide a solid and simple environment for quick board bring-up.

The outcome is finally a test suite where each module can be individually tested. Based on a common initialization of the i.MX6 and the development board, a test for a specific peripheral is launched with the help of a serial console for human interaction.
Included in the release package are Firmware Guide (iMX6FG_RevB.pdf) and IOMUX Tool application (IOMux.exe).

2 Versions Available

	Version	Released Date	Description
Software (iMX6DQ_platlib_release_2011w50.tar.gz)	V2.0.0	12/15/2011	Initial Release with following drivers IPU, ESAI, SSI, AUDMUX, I2C, HDMI, GPT, EPIT, GIC, SDMA, UART and USDHC
Firmware Guide (IMX6FG_RevB.pdf)	Rev B	12/15/2011	Detailed driver documentation available with the release
IOMUX Tool	V3.0.4	10/05/2011	IOMUX configuration tool windows application

3 Source code structure of the library:

Description	Location
Configuration file used to enable what should be supported by a board. e.g mx61_evb.conf	`./configs`
Source code related to the ARM Cortex-A9 core.	`./src/cortex_a9`
Common headers which are platform and i.MX independent.	`./src/include`
Headers specific to the i.MX6 series.	`./src/include/mx61`
Global library source file such main, system initialization,...	`./src/init`
Source code specific to the i.MX6 series.	`./src/mx61/`
Parent directory for all drivers, includes, and their unit tests and docs.	`./sdk/`

Example for a driver:

Description	Location
SDMA driver.	`./sdk/sdma/drv`
SDMA driver related includes.	`./sdk/sdma/inc`
SDMA driver unit tests.	`./sdk/sdma/test`

4 Build procedure

4.1 System requirements

To build the firmware library a 'Linux host' or 'Windows host + Cygwin' is necessary.

(a) Working with Cygwin:

Cygwin can be installed and obtained from here:
http://www.cygwin.com/
The installation procedure is documented on this site.
Some packages are necessary such '`make`', '`bash`', '`perl`', but that list is not thorough.

The cygpath program is a utility that converts Windows native filenames to Cygwin POSIX-style pathnames and vice versa. It can be used when a Cygwin program needs to pass a file name to a native Windows program, or expects to get a file name from a native Windows program.

When using the codesourcery toolchain, it is necessary to set this environment variable:
`export CYGPATH=cygpath`

Note: The make package of Cygwin is the version v3.81. The build might fail due to known problem with that version to handle correctly Windows pathname. To avoid this issue, it is recommended to upgrade the version manually to v3.82. Please replace make.exe with the one provided in package under `./tools/Cygwin` folder.

(b) Installing the toolchain:

The code is compiled with the ARM-NONE-EABI GCC toolchain obtained here:
https://sourcery.mentor.com/sgpp/lite/arm/portal/subscription3053
Click on "Download Sourcery G++ Lite 2011.03-42" and download the Windows or Linux tar ball or installer.

Install the package in "`/opt`" or any other local folder, and make sure that the PATH environment variable allows accessing these executables used for the build process:

```
CC= arm-none-eabi-gcc
AS = arm-none-eabi-as
AR = arm-none-eabi-ar
LN = arm-none-eabi-ld
```

Example:
`export PATH=$PATH:/toolchain_install_path`

Otherwise, it is possible to re-define the above list in this file
"`./install_path/make.def`" with the complete path to the executable.

4.2 Source code installation

The source code package can be installed anywhere.

It is not recommended to use a Windows unzip program to extract the sources. Instead, the equivalent command of Cygwin or Linux should be used.
Example:
`tar zxvf mx61_evb_sdk_release.tar.gz`

4.3 Build command

Each module can be tested by using a command like:

```
./tools/build_sdk –target mx61 –board evb –board_rev 1 –test uart
```

Note: even though the above command generates a makefile in the installation path, this one can't be used to build the SDK.

Only the i.MX6 EVB rev1 is available in this release.

Tests list is:

Module	parameter
Audio : SSI and audiomux	- audio
CAN	-flexcan
EPIT	- epit
Generic Interrupt Controller - GIC	- gic
GPMI	-gpmi
GPT	- gpt
HDMI	- hdmi
I2C	-i2c
IPU	- ipu
SATA	-sata
SDMA	- sdma
SNVS RTC	-snvs_rtc
SNVS SRTC	-snvs_srtc
SPI	-spi
UART	- uart
USDHC	- usdhc
Video decoder	-vdec
ALL	- ALL

The option ALL includes all tests with the possibility to run, re-run, or skip some of tests.

Note: the warnings detected during the build process are considered as errors.

Once the build is finished a message should appear such:

```
************************************************
  Build has completed. ELF file available in:
  => /install_path/output/mx61/bin

Done ... Build script completed
```

The output binary and ELF files can be found in:

```
./output/mx61/bin/mx61ard-epit-sdk.elf
./output/mx61/bin/mx61ard-epit-sdk.bin
```

The file name contains the targeted CPU and board, as well as the tested module.

It is sometimes required to clean the build to take into account some changes. The following command will remove the directory `./install_path/output/mx61` before building:

```
./tools/build_sdk –target mx61 –board evb –board_rev 1 –test uart -
clean
```

5 How to run a test

5.1 Setup required

An i.MX61 EVB board is necessary to run a test. Please refer to the appropriate board user guide to prepare this setup.

The user interaction and output information are available through the serial port UART4 connected to a host running a terminal such TeraTerm, minicom, HyperTerminal,…
The configuration for the terminal is common: 115200bps, 8-bit, no parity, 1 stop bit.

5.2 Download with a JTAG probe

The code can be downloaded and executed through any JTAG tool which supports Cortex-A9.
It is necessary to have a board initialization file that will initialize some of the clocks, the MMDC controller and its interface, as well as the DDR3.
The source package contains the include file used by the ARM Realview Debugger to initialize the board (e.g. mx61_ard_init.inc).

5.3 Using SD boot

The SDK binary image can be programmed on an SD card using 2 different methods.

Using a Windows host:
By using the cfimager-imx.exe provided with the release package under folder ./tools/windows.
Command:

```
cfimager-imx.exe –o 0x0 –f ./output/mx61/evb/bin/mx61evb-ALL-sdk.bin
–d <your drive letter for the SD card>
```

For example, if the drive letter of the SD card reader is "f", type the following:

```
cfimager-imx.exe –o 0x0 –f ./output/mx61/evb/bin/mx61evb-ALL-sdk.bin
–d f
```

There is also the option of formatting the card first before programming it, though be warned it will take more time for this operation to complete due to the card formatting.
To invoke the card formatting, simply append the above command with "-a", as provided in the following example (again, assuming the drive letter is "f"):

```
cfimager-imx.exe –o 0x0 –f ./output/mx61/evb/bin/mx61evb-ALL-sdk.bin
–d f -a
```

Using a Linux host:

By using the command "dd" under Linux.

```
dd if=output/mx61/evb/bin/mx61evb-ALL-sdk.bin of=/dev/sdx seek=2
skip=2 && sync
```

/dev/sdx is the Linux device for your SD card.

Once the SD card is programmed successfully with a bootable SDK image, follow below instructions on how to run it on the i.MX61 EVB board:

- Insert SD card with SDK binary into SD slot.
- Configure boot dip switches for boot from SD – SW1: off-off-off-off-off-off-on-off; SW2: off-off-off-on-on-off-off-off; SW5: all off; SW8: all off.
- Power on board, and follow on screen prompt commands on host' terminal window.

6 How to add a new driver and test

A new driver should typically be added in the ./sdk/new_driver/drv and its test into ./sdk/new_driver/test.

In the board config file such mx61_evb.conf, the following lines must be added:

```
./src/sdk/new_driver/drv
./src/sdk/new_driver/test
```

Everything contained in these repositories will be compiled.

The main test file should contain a test function named: `int32_t new_driver_test(void)`. Hence, when building, the parameter "`– new_driver`" can simply be used.

When using the build ALL option, this test can also be added into the function `void ALL_test(void)` available in `./src/mx61/mx61.c`.

7 Firmware Guide

Firmware guide is a detailed documentation of the work done in platform library with pseudo code examples of driver code. The document (IMX6FG_RevA.pdf) is placed in the root folder of release package.

8 IOMUX tool

IOMUX tool helps generate IOMUX configuration source code in C programming language. The source code generated is used as is in platform library. IOMUX tool is to be used by board designers to generate the design XML file for PINMUX and pad configuration for a board. It is also used to generate source code (.h and .c files) with functions to setup IO configuration to free device driver developers from board design details of each pin and pad.

The IOMUX tool is available with release package in folder "IOMUX Tool", also included is readme file and sample XML IOMUX design file for EVB board type for i.MX6 Series.

9 Install the video decoding demo:

This guide is elaborated to show how to setup and run the dual video + dual HD1080P60 display demo on i.MX6x EVB board.
Text for user input.
Text for important hints.

1. How to build the program?
   ./tools/build-sdk –target mx61 –board evb –board_rev 1 –test vdec –clean

2. How to setup the demo?

a) Create the image over SD card. Under linux, using fdisk/mkfs.vfat/dd to create a bootable image together with the FAT32 file system on the same SD card.

- sudo fidsk /dev/sdx , sdx is the device name of your SD card.

sudo fdisk /dev/sdb
Command (m for help): **m**

Delete existing partition if there is.
Command (m for help): **d**
Selected partition 1

Command (m for help): **n**
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): **1**
First cylinder (1-1023, default 1): **256**
Here the start address should be larger than 2M (space reserved to program the test binary). If one cylinder is 4k, then here 1G is reserved.
Last cylinder or +size or +sizeM or +sizeK (256-1023, default 1023): **1023**

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

Now there's one partition on the SD card.
cat /proc/partitions
major minor #blocks name

8 0 78125000 sda
8 1 104391 sda1
8 2 78019672 sda2
253 0 75956224 dm-0
253 1 2031616 dm-1
8 16 3872256 sdb
8 17 2904576 sdb1

- Using mkfs.vfat to format the partition:
  sudo mkfs.vfat /dev/sdb1
- Copy two video clips to the SD card. ***Note that the filename should be less than 8 characters with ".264" extension. The video should be RAW h264 encoded files with no any container. The program will find the first two valid 264 files for playing.***
- Burn the image to the SD card.
  sudo dd if=output/mx61/evb/bin/mx61evb-vdec-sdk.bin of=/dev/sdb seek=2 skip=2 bs=512 && sync
  NOTE that seek=2 skip=2 is mandatory, without them the MBR of the file system would be overwritten!!

b) Put the SD card into SLOT4.
c) Set the boot switch to boot from SD4.
d) Plug the HDMI cable of the first display to J5 of the CPU board.
e) Optional second display: plug the HDMI expansion board (P/N : MCIMXHDMICARD) based on the SiI9024 chip onto the parallel display connector J19 of the CPU board, and plug the HDMI cable of the second display.
f) Connect the serial cable for console output and 5V power supply, power on the board.

```
**********************************************************************
   Diagnostics Suite (1.0) for i.MX61 TO1.0 Armadillo EVB Rev A
   Build: Nov 25 2011, 17:27:40
   Freescale Semiconductor, Inc.
**********************************************************************

======== Clock frequencies(HZ) =========
Cortex A9 core  : 792,000,000
DDR memory      : 528,000,000
UART4 for debug : 80,000,000
EPIT1 for system timer  : 66,000,000
=======================================

========== DDR configuration ===========
data bits: 64, num_banks: 8
row: 14, col: 10
DDR type is DDR3
Both chip select CSD0 and CSD1 are used
Density per chip select: 1024MB
```

```
=======================================

File 0 is    vid1.264
File 1 is     vid2.264
The first HDMI display is configured!!
The second HDMI display is configured!!
```

Now the endless loopback decoding + display are working.

3. Limitations
    a) There is no resizing on the video output. For example if you are decoding some video clips with resolution other than 1080p, the video will show on the top-left of the screen with its original size.
    b) Video clips must be raw without container.
    c) Currently only H264 video decoding is supported. Other formats such as VC1, H263, MPEG3, MPEG4 will be added in a next release.