Contents

Welcome to the OMAP5 GLSDK Software Developer's Guide

Thank you for choosing the OMAP5 uEVM for your application. The purpose of this guide is to get you going with developing software for the OMAP5 on a Linux development host only.

Note! This Software Developer's Guide (SDG) supports version 6.00 of the OMAP5 GLSDK which is only for Linux host development.

Note! All instructions in this guide are for Ubuntu 12.04. At this time, this is the only supported Linux host distribution for development.

Throughout this document there will be commands spelled out to execute. Some are to be executed on the Linux development host, some on the Linux target and some on the u-boot (bootloader) prompt. They are distinguished by different command prompts as follows:

```
host $ <this command is to be executed on the host>
target # <this command is to be executed on the target>
u-boot :> <this command is to be executed on the u-boot prompt>
```

Starting your software development

The GLSDK should be installed before you continue. Throughout this document it will be assumed you have an environment variable *GLSDK* which points to where your GLSDK is installed. You can set it as follows (the following assumes that GLSDK was installed at default location):

```
host $ export GLSDK="${HOME}/ti-glsdk_omap5-uevm_xx_xx_xx_xx"
```

Setting up the GLSDK

You will need an ARM Linux development environment, in case you do not have one please refer to this link to see how to set one up.

Configuration of ARM Linux development Environment

The GLSDK comes with a script for setting up your Ubuntu 12.04 LTS development host as well as your target boot environment. It is an interactive script, but if you accept the defaults by pressing return you will use the recommended settings. This is recommended for first time users. Note that this script requires ethernet access as it will update your Ubuntu Linux development host with the packages required to develop using the GLSDK. Execute the script using:

```
host $  ${GLSDK}/setup.sh
```

The setup script would perform the following operations:

1. Installs all the package on the host for the SDK.
2. Setup the target filesystem based on Ubuntu 12.04 core filesystem.
3. Configures the target filesystem for network settings and console settings. For network settings, it would ask for DNS settings, environment variables and apt-get proxy
4. Copies the first-boot.sh script and sets up local package repository on the board
5. Minicom is set up to communicate with the target over Debug USB. If you want to use a windows host for connecting to the target instead, see the #Setting_up_Tera_Term section

If you start minicom on your Linux development host using *minicom -w* (or Tera Term on Windows) and power cycle the uEVM, Linux will boot.

Prepare SD Card

To install the release image, you need a µSD Card (at least 4GB) with 2 partitions:

- boot (vfat) partition.
- rootfs (ext4 or ext3) partition.

You can do it manually or you can use following procedure:

- Plug an SD card reader to your PC and insert a µSD card. It must be at least 4GB size.

- Identify which device corresponds to the SD card reader. `sudo fdisk -l` command can help you find out where the µSD Card is mapped. We will call it `/dev/sdY` here.
- Re-format your µSD card with this script mksdboot.sh from the bin directory in the GLSDK

```
$ sudo ${GLSDK}/bin/mksdboot.sh --device /dev/sdY --sdk ${GLSDK}
```

This script would prepare the SD card with the prebuilt images and filesystem for SD boot.

To boot the board with this SD card, refer to the Quick Start Guide in *docs/* folder in the GLSDK release.
Once the board boots with this SD card, login with username as **root**.

Configuring the Filesystem

Once the board boots with the ubuntu core filesystem, run the following script on the target:
Note! The board should have a working internet connection to install the packages.

```
target # ./first-boot.sh
```

This script would perform following operations:

1. Creates a new user with username and password as "omapuser". It also adds a password "root" for the root user
2. Install basic packages like ssh and vim, etc.
3. Install packages required to build kernel and userspace.
4. Installs libraries like drm, dce, gstreamer, X11, example applications.
5. Installs window manager i.e Openbox and browser.

Note 1: This is an almost fully automated step, requiring one user input at the end of the execution.
Note 2: The average time taken for the script to complete is about 90 minutes.

Once the installation is finished, reboot the board. The console prompt asking for login would appear on the minicom as well as on the HDMI display. Login either with root as root user OR omapuser as a user.
One could also start the window manager by running the following command on the board:

```
target # startx
```

This would display a blank screen on the HDMI Display with a mouse pointer. Right click and you would see a list of options to start.

A Gstreamer example could then be run using the playbin2 element. The following command would play a H264 file on the the dri2videosink:

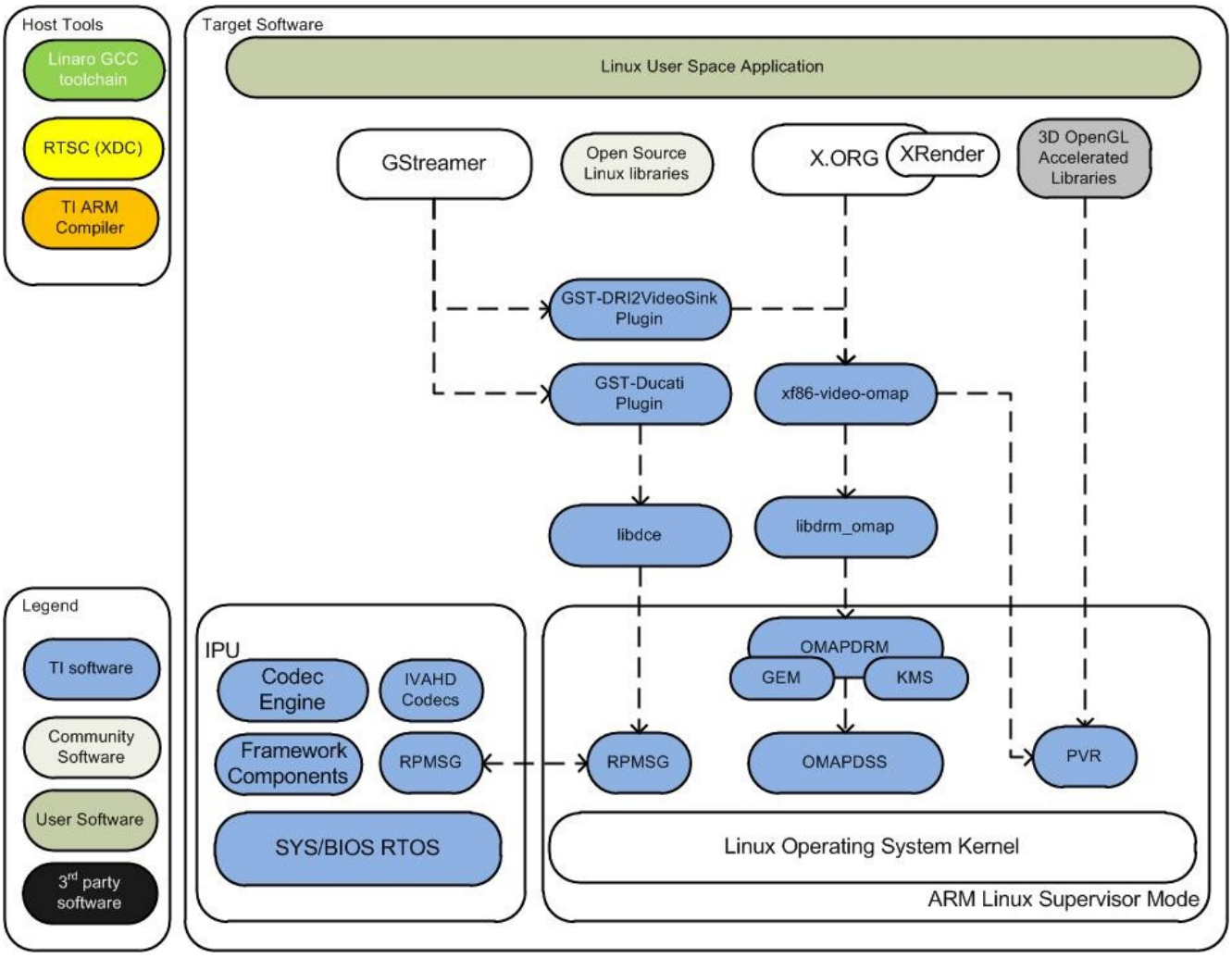
```
target #  gst-launch-0.10 playbin2 uri=file://<path-to-H264-file> video-sink=dri2videosink
```

For audio playback, following settings are required to be executed on the board:

```
target #  amixer cset name='Headset Left Playback' 1
   target #  amixer cset name='Headset Right Playback' 1
   target #  amixer cset name='Headset Playback Volume' 12
```

```
target #  amixer cset name='DL1 PDM Switch' 1
target #  amixer cset name='Sidetone Mixer Playback' 1
target #  amixer cset name='SDT DL Volume' 120
target #  amixer cset name='DL1 Mixer Multimedia' 1
target #  amixer cset name='DL1 Media Playback Volume' 110
target #  amixer sset 'Analog Left',0 'Aux/FM Left'
target #  amixer sset 'Analog Right',0 'Aux/FM Right'
target #  amixer sset 'Aux FM',0 7
target #  amixer sset 'AUDUL Media',0 149
target #  amixer sset 'Capture',0 4
target #  amixer sset MUX_UL00,0 AMic0
target #  amixer sset MUX_UL01,0 AMic1
target #  amixer sset 'AMIC UL',0 120
```

GLSDK software overview



Overview of the GLSDK Software stack

The GLSDK contains many software components. Some are developed by Texas Instruments and some are developed in and by the open source community(White). TI contributes, and sometimes even maintains, some of

these open source community projects, but the support model is different from a project developed solely by TI.

Additional Procedures

Build Environment Setup

Kernel packages can be build natively on OMAP5 boards with Ubuntu Filesystem or Cross Compiled on Ubuntu Machine.

We encourage to use Cross Compilation environment.

Cross Compiler setup

Since Ubuntu 12.04, hardware floating-point support was introduced (armhf). So armhf cross-compiler is required for Ubuntu 12.04 environment.

armhf cross-compiler setup

On Ubuntu 12.04 machine, you would need to install the following meta-packages:

```
$ sudo apt-get install gcc-arm-linux-gnueabihf cpp-arm-linux-gnueabihf
```

If your are on Ubuntu 11.10 machine or lower distro, you can either upgrade to more recent distro or install armhf standalone linaro cross compiler, please refer to the following page: https://launchpad.net/linaro-toolchain-binaries and download the most recent linux armhf cross compiler. Please refer to the README.txt to complete the setup and to export Compiler directory to the PATH.

Additional Build Environment Prerequisites

- Install packages required for kernel packages build - this is valid on Cross Compilation and Native environment:

```
sudo apt-get build-dep linux-image-$(uname -r)
sudo apt-get install dpkg-dev kernel-wedge
```

Rebuilding the GLSDK components

The GLSDK has provided a top level Makefile to allow the re-building of the various components within the GLSDK.

Rebuild the GLSDK components by first entering the GLSDK directory using:

```
host $ cd ${GLSDK}
```

The GLSDK makefile has a number of build targets which allows you to rebuild the GLSDK components. For a complete list execute:

```
host $ make help
```

After that, each of the build targets listed by 'make help' can then be executed using:

```
host $ make <target>_clean
host $ make <target>
host $ make <target>_install
```

For example, to compile the Linux Kernel, you can use the following commands

```
host $ make linux_clean
host $ make linux
host $ make linux_install
```

In order to install the resulting binaries on your target, execute one of the "install" targets. Where the binaries are copied is controlled by the `EXEC_DIR` variable in `${GLSDK}/Rules.make`. By default, this variable is set up to point to your NFS mounted target file system when you execute the GLSDK setup (`setup.sh`) script, but can be manually changed to fit your needs.

You can remove all components generated files at any time using:

```
host $ make clean
```

And you can rebuild all components using:

```
host $ make all
```

You can then install all the resulting target files using:

```
host $ make install
```

Creating your own Linux kernel image

The pre-built Linux kernel image (uImage) provided with the GLSDK is compiled with a default configuration. You may want to change this configuration for your application, or even alter the kernel source itself. This section shows you how to recompile the Linux kernel provided with the GLSDK, and shows you how to boot it instead of the default Linux kernel image.

1. If you haven't already done so, follow the instructions in #Setting_up_the_GLSDK to setup your build environment.

2. Recompile the kernel provided with the GLSDK by executing the following:

```
host $ cd ${GLSDK}
host $ make linux_clean
host $ make linux
host $ make linux_install
```

3. You will need a way for the boot loader (u-boot) to be able to reach your new uImage. Copy the new uImage that is generated in arch/arm/boot/ directory to the boot filesystem

4. Copy the exported Linux kernel modules from the EXEC_DIR to the /lib/modules directory to the root file system

Setting up Tera Term

Tera Term is a commonly used terminal program on Windows. If you prefer to use it instead of Minicom, you can follow these steps to set it up.

1. Download Tera Term from this location, and start the application.

2. In the menu select *Setup->General...* and set:

```
Default port: COM1
```

3. In the menu select *Setup->Serial Port...* and set the following:

```
Port:         COM1
Baud rate:    115200
Data:         8 bits
Parity:       none
Stop:         1 bit
Flow control: none
```

NOTE: ***Kernel Bootargs can be generated by running the setup script. See the section #Setting_up_the_GLSDK for details on running the setup script.***