
AXEL

**Solo / Dual / Quad ARM Cortex-A9
CPU Module**

Axel Embedded Linux Kit (XELK)

Quick Start Guide

<Page intentionally left blank>

Table of Contents

1 Preface..... 5

 1.1 About this manual..... 5

 1.2 Copyrights/Trademarks..... 5

 1.3 Standards..... 5

 1.4 Disclaimers..... 5

 1.5 Warranty..... 6

 1.6 Technical Support..... 6

 1.7 Related documents..... 7

 1.8 Conventions, Abbreviations, Acronyms..... 8

2 Introduction..... 11

 2.1 Axel SOM..... 11

 2.2 Embedded Linux..... 13

 2.3 XELK..... 14

 2.3.1 Kit Contents..... 16

 2.3.2 XELK Release Notes..... 16

 2.3.2.1 Version 1.0.0..... 16

 2.3.2.2 Version 1.1.0..... 17

 2.3.2.3 Version 1.2.0..... 17

 2.3.2.4 Releases history..... 17

 2.3.2.5 Known limitations..... 18

3 XELK Quick Start..... 19

 3.1 Unboxing..... 19

 3.2 Hardware setup..... 20

 3.3 First boot..... 20

 3.4 Installing DVDK..... 22

 3.4.1 MicroSD contents..... 22

 3.4.2 Importing the virtual machine..... 23

 3.4.3 Launching the virtual machine..... 25

4 Developing Environment..... 28

 4.1 Introduction..... 28

 4.2 Software components..... 29

 4.2.1 Toolchain..... 29

 4.2.2 Bootloader..... 29

 4.2.3 Kernel..... 30

 4.2.4 Target root file system..... 30

 4.2.5 U-Boot and Linux git repositories..... 31

 4.2.5.1 RSA key generation..... 31

 4.3 Build system..... 32

 4.3.1 Introduction..... 32

 4.3.2 Setting up the server environment..... 33

 4.3.2.1 TFTP Server..... 33

4.3.2.2 NFS Server.....	33
4.3.2.3 Pre-built toolchain.....	34
4.3.2.4 Pre-built root file system.....	34
4.3.2.5 Using a build tool.....	35
4.4 Working with XELK.....	37
4.4.1 Building u-boot.....	38
4.4.2 Building Linux kernel.....	38
4.4.3 Recovery procedure.....	39
5 Frequently Asked Questions.....	41
5.1 Q: Where can I found Axel SOM information?.....	41
5.2 Q: I've received the XELK package. How am I supposed to start working with it?.....	41
5.3 Q: How can I update the XELK version?.....	42
5.4 Q: How can I work with the XYZ peripheral/interface?.....	42
5.5 How can I configure the Axel system to boot from network?.....	42
5.6 Q: Can you suggest some guidelines for the carrier board design?.....	43
6 Appendices.....	44
6.1 U-Boot startup and environment.....	44
6.2 Boot messages on the serial console.....	44

1 Preface

1.1 About this manual

This manual describes the Axel Embedded Linux Kit (XELK) and serves as a quick guide for start working with the development kit.

1.2 Copyrights/Trademarks

Ethernet® is a registered trademark of XEROX Corporation.

All other products and trademarks mentioned in this manual are property of their respective owners.

All rights reserved. Specifications may change any time without notification.

1.3 Standards

DAVE Embedded Systems is certified to ISO 9001 standards.

1.4 Disclaimers

DAVE Embedded Systems does not assume any responsibility for availability, supply and support related to all products mentioned in this manual that are not strictly part of the Axel CPU module, the AxelEVB-Lite carrier board and the Dacu carrier board.

Axel CPU Modules are not designed for use in life support appliances, devices, or systems where malfunctioning of these products can reasonably be expected to result in personal injury. **DAVE Embedded Systems** customers who are using or selling these products for use in such applications do so at their own risk and agree to fully indemnify **DAVE Embedded Systems** for any damage resulting from such improper use or sale.

1.5 Warranty

Axel SOM, AxelEVB-Lite and Dacu are guaranteed against defects in material and workmanship for the warranty period from the shipment date. During the warranty period, **DAVE Embedded Systems** will at its discretion decide to repair or replace defective products. Within the warranty period, the repair of products is free of charge provided that warranty conditions are observed.

The warranty does not apply to defects resulting from improper or inadequate maintenance or handling by the customer, unauthorized modification or misuse, operation outside of the product's specifications or improper installation or maintenance.

DAVE Embedded Systems will not be responsible for any defects or damages to other products not supplied by **DAVE Embedded Systems** that are caused by a faulty Axel module, AxelEVB-Lite or Dacu.

1.6 Technical Support

We are committed to making our products easy to use and will help customers use our CPU modules in their systems.

Technical support is delivered through email for registered kits owners. Support requests can be sent to support-axel@dave.eu. Software upgrades are available for download in the restricted download area of **DAVE Embedded Systems** web site: <http://www.dave.eu/reserved-area>. An account is required to access this area.

Please refer to our Web site at <http://www.dave.eu/dave-cpu-module-imx6-axel.html> for the latest product documents, utilities, drivers, Product Change Notices, Board Support Packages, Application Notes, mechanical drawings and additional tools and software.

1.7 Related documents

Document	Location
DAVE Embedded Systems Developers Wiki	http://wiki.dave.eu/index.php/Main_Page
i.MX6 Application Processor Reference Manual	http://cache.freescale.com/files/32bit/doc/ref_manual/IMX6DQRM.pdf?fpsp=1&WT_TYPE=Reference%20Manuals&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=pdf&WT_ASSET=Documentation
Freescale I.MX community website	https://community.freescale.com/community/imx
Freescale L3.0.35-4.1.0 documentation package	https://www.freescale.com/webapp/Download?colCode=L3.0.35_4.1.0_LINUX_DOCS&location=null&fpsp=1&WT_TYPE=Supporting%20Information&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=gz&WT_ASSET=Documentation&Parent_nodeId=1337637154535695831062&Parent_pageType=product
Axel main page on DAVE Embedded Systems Developers Wiki	http://wiki.dave.eu/index.php/Category:Axel
Axel Hardware Manual	http://www.dave.eu/sites/default/files/files/axel-hm.pdf
Axel Software Manual	http://wiki.dave.eu/index.php/Software_Manual_(Axel)
AxelEVB-Lite page on DAVE Embedded Systems	http://wiki.dave.eu/index.php/AxelEVB-Lite

Document	Location
Developers Wiki	
Dacu User's Guide	Provided with kit documentation
Building Embedded Linux Systems By Karim Yaghmour.	This book covers all matters involved in developing software for embedded systems. It is not a reference guide, but it provides a complete and exhaustive overview that helps the developer save a lot of time in searching for such information on the Internet
Training and Docs sections of Free Electrons website.	Brief but still exhaustive overview of the Linux and Embedded Linux world.

Tab. 1: Related documents

1.8 Conventions, Abbreviations, Acronyms

Abbreviation	Definition
BTN	Button
DVDK	Dave Virtual Development Kit
EMAC	Ethernet Media Access Controller
GPI	General purpose input
GPIO	General purpose input and output
GPO	General purpose output
LTIB	Linux Target Image Builder
OVA	Open Virtualization Archive
PCB	Printed circuit board
PMIC	Power Management Integrated Circuit
PSU	Power supply unit
RTC	Real time clock
SOC	System-on-chip
SOM	System-on-module

Abbreviation	Definition
WDT	Watchdog
XELK	Axel Embedded Linux Kit

Tab. 2: Abbreviations and acronyms used in this manual

Revision History

<i>Version</i>	<i>Date</i>	<i>Notes</i>
1.0.0	November 2013	First official release
1.0.1	January 2014	Released with XELK 1.1.0 Minor fixes
1.0.2	May 2014	Added support for AxelLite SOM Minor fixes Released with XELK 1.2.0

2 Introduction

2.1 Axel SOM

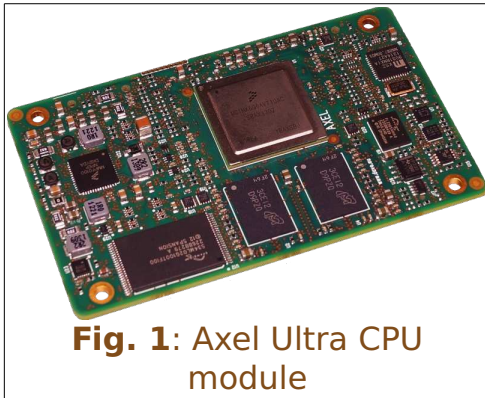


Fig. 1: Axel Ultra CPU module

Axel is the new top-class Solo/Dual/Quad core ARM Cortex-A9 CPU module by **DAVE Embedded Systems**, based on the recent Freescale i.MX6 application processor.

Thanks to Axel, customers have the chance to save time and resources by using a compact solution that permits to reach scalable

performances that perfectly fits the application requirements avoiding complexities on the carrier board.

The use of this processor enables extensive system-level differentiation of new applications in many industry fields, where high-performance and extremely compact form factor (85mm x 50mm) are key factors. Smarter system designs are made possible, following the trends in functionalities and interfaces of the new, state-of-the-art embedded products. Axel offers great computational power, thanks to the rich set of peripherals, the Scalable ARM Cortex-A9 together with a large set of high-speed I/Os (up to 5GHz).

Axel enables designers to create smart products suitable for harsh mechanical and thermal environments, allowing the development of high computing and reliable solutions. Thanks to the tight

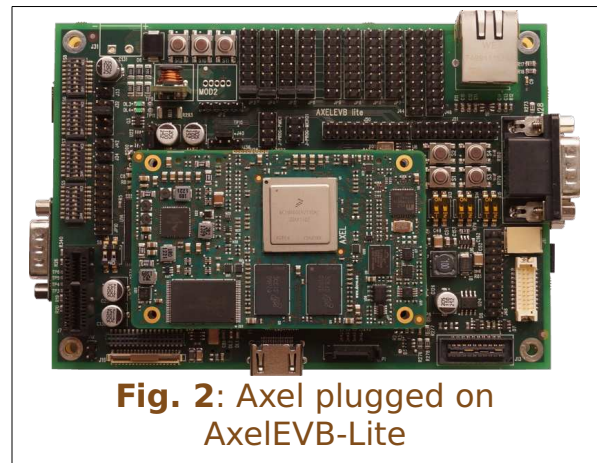


Fig. 2: Axel plugged on AxeLEVB-Lite

integration between the ARM Core-based processing system, designers are able to share the application through the multi-core platform and/or to divide the task on different cores in order to match with specific application requirements (AMP makes possible the creation of applications where RTOS and Linux work together on different cores). Thanks to Axel, customers are going to save time and resources by using a powerful and scalable compact solution, avoiding complexities on the carrier PCB.

Axel is designed and manufactured according to **DAVE Embedded Systems** Ultra Line specifications, in order to guarantee premium quality and technical value for customers who require top performances and flexibility. Axel is suitable for high-end applications such as medical instrumentation, advanced communication systems, critical real-time operations and safety applications.

For further information, please refer to Axel Hardware Manual.

2.2 Embedded Linux

When we talk in general about Embedded Linux¹, we refer to an embedded system running Linux operating system. As the reader probably knows, Linux was first developed on the PC platform, based on the famous x86 architecture. Typical embedded systems using an operating system (O.S. for short), are equipped with much lighter software. Recent hardware advances made these systems so powerful that now they can run a complex O.S. such as Linux. This choice has several benefits:

- The developer can count on a reliable and efficient software, developed and maintained by a large community all over the world
- The software is open-source, so developers have access to the whole source code
- Since Linux runs on many different platforms (x86, PowerPC, ARM, SuperH, MIPS etc.), applications are portable by definition
- There are a lot of open-source applications running on top of Linux that can easily be integrated in the embedded system
- Last but not least, there are no license fees.
- The typical Embedded Linux system is composed of:
 - the bootloader – this software is run by the processor after exiting the reset state. It performs basic hardware initialization, retrieves the Linux kernel image (for example from a remote server via the TFTP protocol) and launches it by passing the proper arguments (command line and tags)
 - the Linux kernel
 - the root file system – this file system is mounted (which means "made available", "attached") by the kernel during the boot process on the root directory ("/").

The typical developing environment for an Embedded Linux system is composed of a host machine and a target

¹ An exhaustive description of this topic is beyond the scope of this document. We recommend reading specific documents, eg Building Embedded Linux Systems By Karim Yaghmour.

machine. The host is used by the developer to compile the code that will run on the target. In our case the target is obviously the Axel module, while the host is assumed to be a PC running the Linux operating system. The Linux kernel running on the target can mount the root file system from different physical media. For example, during the software development, we strongly recommend using a directory exported via NFS by the host for this purpose (see the example configuration called `net_nfs`); however, for system deployed to the field, the root file system is usually stored into a flash device.

2.3 XELK

Axel Embedded Linux Kit (XELK for short) provides all the necessary components required to set up the developing environment for:

- building the bootloader (U-Boot)
- building and running Linux operating system on Axel-based systems
- building Linux applications that will run on the target

The heart of Axel SOM is Freescale i.MX6 Solo/Dual/Quad core application processor. From a software point of view, Freescale supports this processor family through so-called LTIB (Linux Target Image Builder) BSPs. The Linux BSP releases are published on a regular basis and the release packages have a reference code as `L<Kernel_version>_<x.y.z>` (eg: `L3.0.35_4.1.0`). For more details please refer to:

- http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=i.MX6Q&fsp=1&tab=Design_Tools_Tab
- [https://community.freescale.com/community/imx/content?filterID=contentstatus\[published\]~category\[imx6all\]&filterID=contentstatus\[published\]~objecttype~objecttype\[document\]](https://community.freescale.com/community/imx/content?filterID=contentstatus[published]~category[imx6all]&filterID=contentstatus[published]~objecttype~objecttype[document])

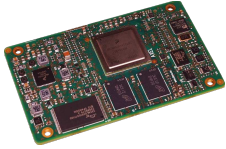
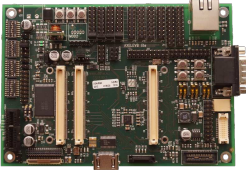
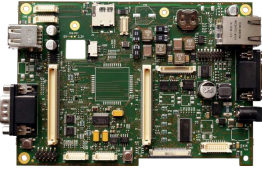



Axel Embedded Linux Kit, in turn, is directly derived from

L<Kernel_version>_<x.y.z> BSP versions. Hence XELK documentation often refers to L<Kernel_version>_<x.y.z> resources.

DAVE Embedded Systems adds to the latest L<Kernel_version>_<x.y.z> BSP from Freescale the customization required to support the Axel platform. For this reason most of the documentation provided by Freescale remains valid for the XELK development kit. However, some customization is required, in particular at bootloader and Linux kernel levels.

2.3.1 Kit Contents

The following table lists the XELK components

Component	Description
	Axel Ultra or AxelLite SOM CPU: Freescale i.MX6
	Axel-EVB-Lite Carrier board
	Dacu Carrier board
	Ampire AM-800480STMQW 7" 800x480 LCD display LVDS interface
	AC/DC Single Output Wall Mount adapter Output: +12V – 2.0 A
	MicroSDHC card with SD adapter and USB adapter

2.3.2 XELK Release Notes

2.3.2.1 Version 1.0.0

First official release

2.3.2.2 Version 1.1.0

- Minor update that adds support for more peripherals: NAND, RTC, I²C, SPI
- Touch screen works properly
- CAN works @ 1Mbps
- The system can boot from SD

2.3.2.3 Version 1.2.0

- Added support for AxelLite SOMs
- Bug fixes and minor changes

2.3.2.4 Releases history

	XELK Version		
Release number	1.0.0	1.1.0	1.2.0
Status	Released	Released	Released
Release date	November 2013	January 2014	May 2014
Release notes	Version 1.0.0	Version 1.1.0	Version 1.2.0
SOM PCB version	Axel Ultra: CS030713	Axel Ultra: CS030713A	Axel Ultra: CS030713A Axel Lite: CS335013A
Supported carrier boards	AxelEVB-Lite Dacu	AxelEVB-Lite Dacu	AxelEVB-Lite Dacu
U-Boot version	2013.10-xelk-1.0.0	2013.10-xelk-1.1.0	2013.10-xelk-1.2.0
Linux version	3.0.35-xelk-1.0.0	3.0.35-xelk-1.1.0	3.0.35-xelk-1.2.0
Drivers	SPI NOR Flash (boot) UART debug (2-wire) USB Host USB OTG	SPI NOR Flash (boot) UART debug (2-wire) USB Host USB OTG	SPI NOR Flash (boot) UART debug (2-wire) USB Host USB OTG

	XELK Version		
	SD/MMC1 CAN Touch screen controller EMAC SATA HMDI LVDS1	SD/MMC1 CAN Touch screen controller EMAC SATA HMDI LVDS1 NAND RTC I ² C SPI	SD/MMC1 CAN Touch screen controller EMAC SATA HMDI LVDS1 NAND RTC I ² C SPI
Freescale BSP version	L3.0.35-4.1.0	L3.0.35-4.1.0	L3.0.35-4.1.0

2.3.2.5 Known limitations

The following table reports the known limitations of the latest XELK version, which will be solved for the next releases of the development kit:

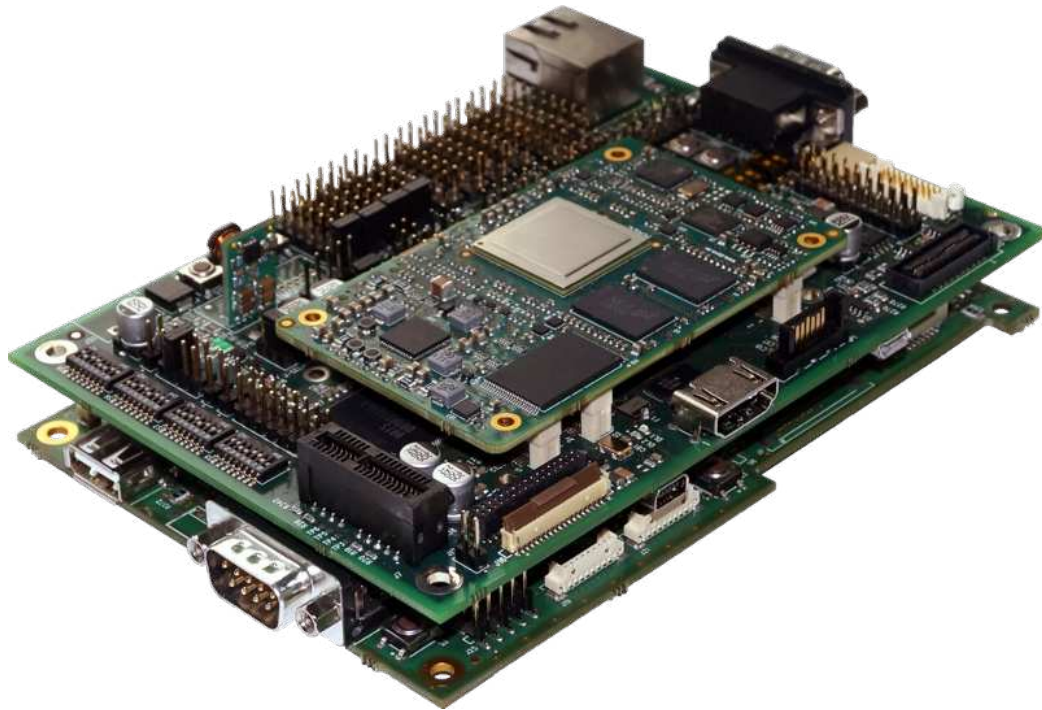
Issue	Description
USB OTG	Verified in Host and Device modes
Reboot from software	Rebooting the system from software (eg: launching the reboot command from Linux user space) can lead to a system lock. To solve it, reset the board with the dedicated button (S10)
SD1	SD1 interface (available on the Axel Lite adapter board as microSD connector) detects the card only if it is inserted before booting Linux
Ethernet	10 Mbps connections have not been tested

3 XELK Quick Start

This chapter describes how to quickly start working with the XELK kit. The following paragraphs will guide you through the setup and installation procedures.

3.1 Unboxing

Once you've received the kit, please open the box and check the kit contents with the packing list included in the box and using the table on chapter 2.3.1 as a reference. The hardware components (SOM, carrier boards and display) are pre-assembled, as shown in the picture below:



3.2 Hardware setup

This section describes how to quick start an Axel system composed of a Axel SOM plugged into the AxelEVB-Lite and then mounted on the Dacu carrier board, provided that it is programmed according to XELK configuration.

The MicroSD provided with the XELK can be used to boot the system, since it contains a bootable partition (mmcbk0p1) and a root file system partition (mmcbk0p2).

1. insert the MicroSD card provided with the development kit into the MicroSD slot
2. connect the 12Vcc power supply to JP2 on the Dacu board
3. (optional) connect a serial cable between the J28 connector on the AxelEVB-Lite board and PC COM port through a NULL-modem cable (not provided)
4. (optional) start your favorite terminal software on PC; communication parameters are:

Parameter	Value
Baud rate	115200 bps
Data bits	8
Stop bits	1
Parity	None

5. (optional) to connect the system to Ethernet LAN, please plug cable on connector J6 connector of the AxelEVB-Lite

The system is configured to boot automatically from the SD card when powered up.

3.3 First boot

Once power has been applied, U-Boot bootloader will be executed and the debug messages will be printed on the serial console. U-Boot automatically runs the autoboot macro, that loads the kernel and launches it with the options for mounting the root file system from the

mmcblk0p2 partition. At the end of the boot process, a demo application is launched and you can interact with the system using the touchscreen.

Moreover, the Linux shell is available on the serial console. Lastly, both telnet and ssh services are available to connect to the system through the network.

Please refer to Appendix 6.2 for an example of the boot messages.

3.4 Installing DVDK

DAVE Embedded Systems Virtual Development Kit is a virtual machine, based on Oracle VirtualBox that allows developers to start using **DAVE Embedded Systems** platform without wasting time in installing the development environment. The Virtual Machine comes with all the development tools and source code (pre-configured), and requires only a minimal setup by the end user (usually only to adapt network interface to the user environment).

DVDK can also be converted, easily, into a physical environment, for example to increase speed on slower machines. Please note that DVDK can be used also with VMWare.

Please refer to DVDK page (<http://wiki.dave.eu/index.php/Category:DVDK>) on **DAVE Embedded Systems** Developer's Wiki for further information.

3.4.1 MicroSD contents

The microSD provided with XELK is used to store:

- A bootable partition (**mmcblk0p1, vfat**) containing:
 - binary images (u-boot and kernel images)
 - XELK documentation
 - DVDK virtual machine image (in .OVF format)
- XELK root file system partition (**mmcblk0p2, ext3**)

XELK contains all the required software and documentation to start developing Linux application on the Axel platform. In particular, XELK provides a virtual machine saved in Open Virtualization Format with two emulated disks:

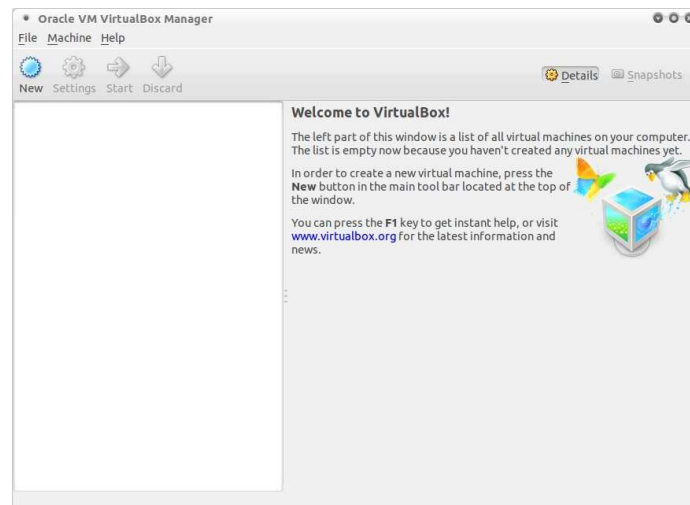
- Boot disk with pre-installed Ubuntu Linux 12.04.2 LTS and pre-configured basic Linux services (TFTP, NFS, ...)

- Secondary disk containing source code and tools:
 - Bootloader (u-boot) source tree cloned from **DAVE Embedded Systems** public git repository
 - Linux kernel source tree cloned from **DAVE Embedded Systems** public git repository
 - Pre-installed development root file systems
 - Toolchain
 - Miscellaneous tools

3.4.2 Importing the virtual machine

XELK provides a virtual machine image as a .OVA file, which is a virtual application exported in Open Virtualization Format (OVF). Please find below the instructions for importing the virtual machine into Virtualbox:

1. Start the Oracle VM VirtualBox Manager



2. Click on File and select "Import Virtual Application", then click on "Open Virtual Application"



3. Navigate your file system and select the .ova file provided with the XELK

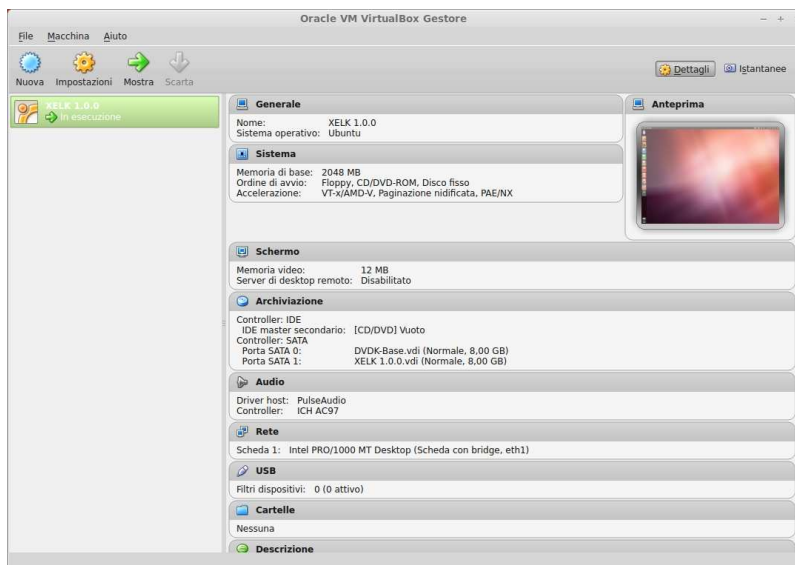


4. Click "Next" and on the next window click on "Import"

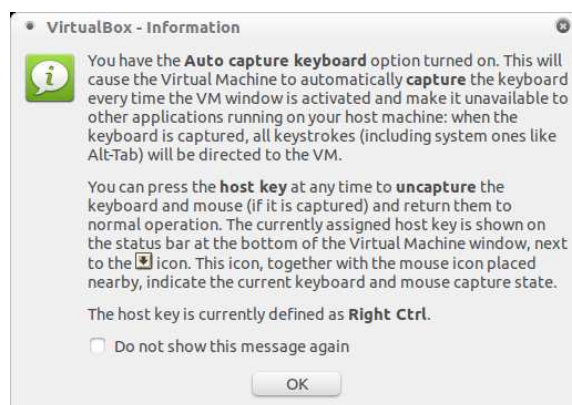


3.4.3 Launching the virtual machine

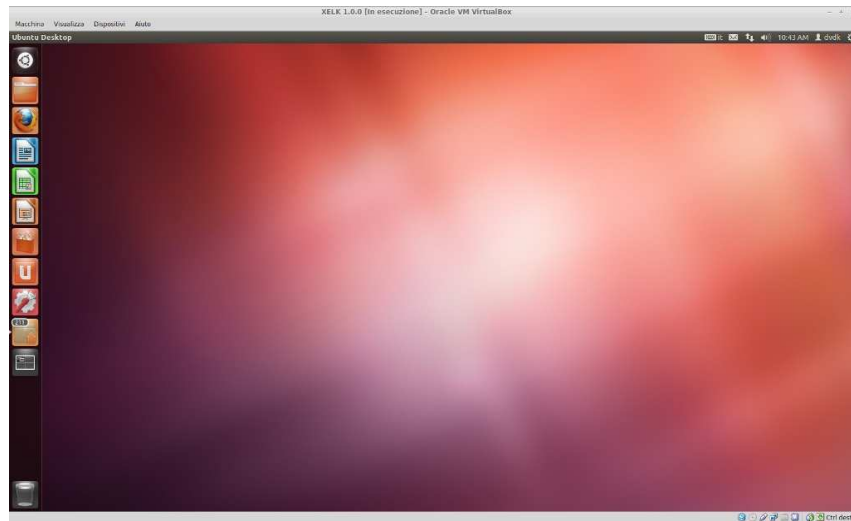
1. VirtualBox will open some message windows like the following, you can click "Ok" to close them



2. VirtualBox will open some message windows like the following, you can click "Ok" to close them



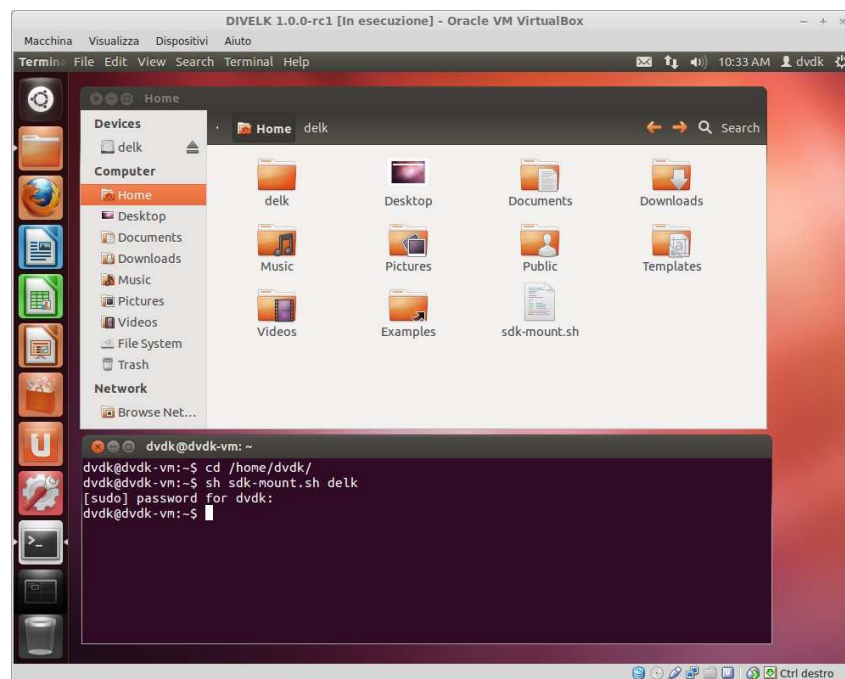
3. At the end of the boot process, the Ubuntu desktop will be available. Please note that the user account credentials are provided with the development kit (you can find them into the "README" file contained in the "dvd" folder of the kit distribution)



4. Mount the sdk disk launching the following commands from a shell terminal:

```
cd /home/dvdk
```

```
sh sdk-mount.sh xelk
```



5. Once logged in, the system could suggest to update the Virtualbox Guest Additions package.

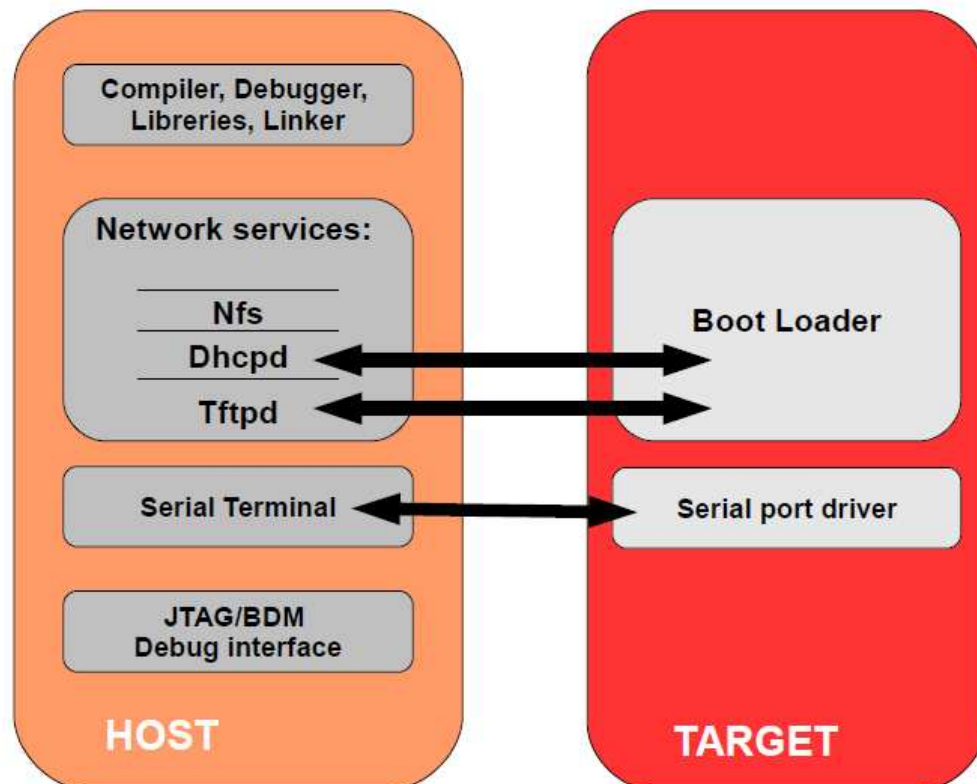
You can follow the on-screen instructions to easily install the updated package.

6. Check if your keyboard layout matches the Ubuntu keyboard settings. You can change the keyboard layout selecting System->Preferences->Keyboard from the top panel menu.
7. Configure the Virtual Machine network interface, as described in this page:
http://wiki.dave.eu/index.php/VirtualBox_Network_Configuration

4 Development Environment

4.1 Introduction

The following figure show the typical development environment for an Embedded Linux system: it is composed of a host machine and a target machine.



The host is used by the developer to (cross-)compile the code that is to run on the target. In our case the target is the Axel CPU module, while the host is assumed to be a PC running the Linux operating system, either in a physical installation or as a virtual machine. The bootloader running on the target can download the Linux kernel image through the network (TFTP), as well as the u-boot binary images (useful when an update of the bootloader is required). Moreover, the Linux kernel running on the target is able to

mount the root file system from different physical media, for example from a directory exported via Network File System (NFS) by the host. This strategy (kernel image and RFS retrieved from the network) saves time during the development phase, since no flash reprogramming or removable storage (SD, usb pen drives, external disks) is required to test new versions or updates of the software components.

4.2 Software components

4.2.1 Toolchain

With the term "toolchain" we refer to the set of programs that allow the building of a generic application. For applications built to run on the same platform as the tool chain, we use a native toolchain. On the contrary, for applications built to run on a target architecture different from the host architecture, we use a cross-toolchain. In this case all the tools involved in this process are lead by the "cross-" prefix. So we talk about cross-compiler, cross-toolchain and so on. The cross-toolchain used to build U-Boot and the Linux kernel is the GNU toolchain for the ARM architecture built for x86 hosts. In other words, the toolchain runs on x86 machines but generates binaries for ARM processors. As for all the software compliant to the GPL license, it is released in source code. Thus the first thing to do to set up the developing environment should be building the cross-toolchain. This is not a trivial task, it takes a lot of time and hard disk space. To avoid this tedious task, we suggest use of a pre-built toolchain as explained in the following sections.

4.2.2 Bootloader

U-Boot is a very powerful boot loader and it became the "de facto" standard on non-x86 embedded platforms. The main tasks performed by U-Boot are:

- hardware initialization (external bus, internal PLL, SDRAM controller etc.)

- starting a shell on the serial port allowing the user to interact with the system through the provided commands
- automatic execution of the boot script (if any)

After system power-up, U-Boot prints some information about itself and about the system it is running on. Once the bootstrap sequence is completed, the prompt is printed and U-Boot is ready to accept user's commands. U-Boot manages an environment space where several variables can be stored. These variables are extremely useful to permanently save system settings (such as ethernet MAC address) and to automate boot procedures. This environment is redundantly stored in two physical sectors of boot flash memory; the default variables set is hard-coded in the source code itself. User can modify these variables and add new ones in order to create his/her own custom set of configurations. The commands used to do that are `setenv` and `saveenv`. This process allows the user to easily set up the required configuration. Once U-Boot prompt is available, it is possible to print the whole environment by issuing the command `printenv`.

For further information on use of U-Boot, please refer to <http://www.denx.de/wiki/view/DULG/UBoot>

4.2.3 Kernel

Linux kernel for i.MX processors is maintained primarily by Freescale. Periodically Freescale releases the so-called Linux BSP, which provides updated kernel sources.

Kernels released within XELK derive directly from Freescale Linux BSP kernels.

4.2.4 Target root file system

The Linux kernel running on the target needs to mount a root file system. Building a root file system from scratch is definitively a complex task because several well known directories must be created and populated with a lot of files that must follow some standard rules. Again we will use

pre-packaged root file systems that make this task much easier.

4.2.5 U-Boot and Linux git repositories

XELK source trees for U-Boot and Linux kernel are provided as git repositories, so the user can immediately get access to the development trees and keep these components in sync and up to date with **DAVE Embedded Systems** repositories.

Component	Git Remote
U-Boot	git@git.dave.eu:dave/axel/u-boot-imx.git
Linux	git@git.dave.eu:dave/axel/linux-2.6-imx.git

When the git account is enabled (please refer to section 4.2.5.1), the developer can synchronize the source tree entering the repository directory and launching the `git fetch` command.

Please note that `git fetch` doesn't merge the commits on the current branch. To do that, the developer should run

```
git merge origin/axel
```

or replace the "fetch-merge" process with a single `git pull` command. Please note that the recommended method is the "fetch-merge" process. For further information on Git, please refer to the official Git Documentation (<http://git-scm.com/documentation>).

4.2.5.1 RSA key generation

For getting access to the Git repositories, a ssh key is required. Please follow the procedure reported below to generate the RSA ssh key (we assume that the ssh package and the required tools are installed on the Linux development server):

- select your username (ad es. `username@myhost.com`)
- start a shell session on the Linux host
- enter the `.ssh` subdirectory into your home directory:

```
cd ~/.ssh/
```

- launch the following command:
- this command creates the files
- edit your `~/.ssh/config` adding the following lines:

```
ssh-keygen -t rsa -C "username@myhost.com"

~/.ssh/username@myhost.com ("private key") and
~/.ssh/username@myhost.com.pub ("public key")

Host git.dave.eu
    User git
    Hostname git.dave.eu
    PreferredAuthentications publickey
    IdentityFile
~/.ssh/username@myhost.com.pub
```

Please send the public key file to the following email support addresses

support-axel@dave.eu

with the request for the creation of a new public git account associated to your username. The support team will enable the account and send you a confirmation as soon as possible.

4.3 Build system

4.3.1 Introduction

A build system is a set of source trees, Makefiles, patches, configuration files, tools and scripts that makes it easy to generate all the components of a complete embedded Linux system. A build system, once properly set up, automates the configuration and cross-compilation processes, generating all the required targets (userspace packages (libraries, programs), the kernel, the bootloader and root filesystem images) depending on the configuration. Some well known build systems are the following:

- Linux Target Image Builder (LTIB, <http://ltib.org/>)
- OpenEmbedded

(http://wiki.openembedded.net/index.php/Main_Page)

- Yocto (<https://www.yoctoproject.org/>)
- Buildroot (<http://buildroot.uclibc.org>)

For the Linux BSP release L3.0.35-4.1.0, Freescale officially supports LTIB as build system. However, a switch to the more modern Yocto is planned for the next releases.

4.3.2 Setting up the server environment

During development, user needs to interact with the target system. This section describes the tools that must be installed and configured on the host system for this purpose. Please note that all these tools are already installed and properly configured on the virtual machine image provided with the XELK.

4.3.2.1 TFTP Server

One of the most useful features of a bootloader during development is the capability to download the Linux kernel from the network. This saves a lot of time because developer doesn't have to program the image in flash every time he/she modifies it. U-Boot implements the TFTP protocol (see the tftp command), so the host system must be configured to enable the TFTP service. Installation and configuration of a TFTP server depends on the host Linux distribution.

The default DVDK tftp installation has `/srv/tftp` as work directory. It is recommended to create a subdirectory dedicated to the image files created with the XELK.

4.3.2.2 NFS Server

One of the most important components of a Linux system is the root file system. A good development root file system provides the developer with all the useful tools that can help him/her on his/her work. Such a root file system can become very big in size, so it's hard to store it in flash memory. User could split the file system in different parts, mounting them from different media (flash, network, usb...).

But the most convenient thing is to mount the whole root file system from the network, allowing the host system and the target to share the same files. In this way, the developer can quickly modify the root file system, even “on the fly” (meaning that the file system can be modified while the system is running). The most common way to setup a system like the one described is through NFS. As for TFTP, installation and configuration depends on the host Linux distribution.

The default DVDK NFS installation is configured for sharing /home directory and all the subdirectories.

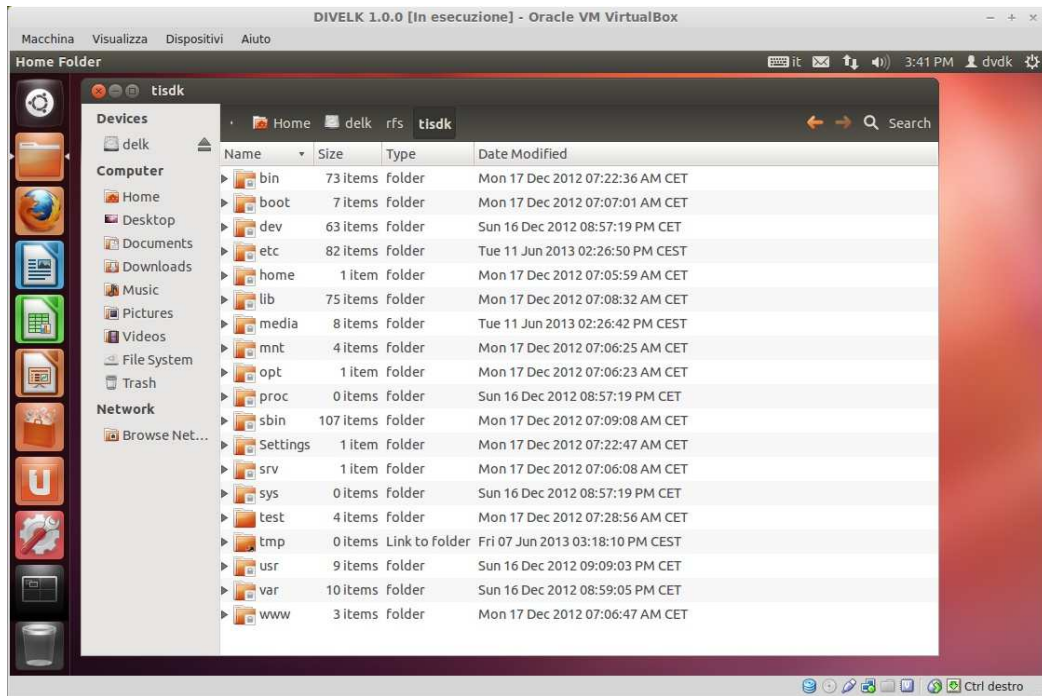
4.3.2.3 Pre-built toolchain

To start developing software for the Axel platform, users need a proper toolchain, which can be pre-built or built-from-scratch. Building a toolchain from scratch is not a trivial task (though using a recent build system is easier than in the past), so the recommended approach consists in using a pre-built toolchain.

XELK provides the cross-target development **arm-eabi** toolchain (**GCC version is 4.4.3**). As an alternative, the toolchains from the LTIB or Yocto build system can be used.

4.3.2.4 Pre-built root file system

Linux needs a root file system: a root file system must contain everything needed to support the Linux system (applications, settings, data, ..). The root file system is the file system that is contained on the same partition on which the root directory is located. The Linux kernel, at the end of its startup stage, mounts the root file system on the configured root device and finally launches the /sbin/init, the first user space process and "father" of all the other processes. An example of root file system is shown below:



For more information on the Linux filesystem, please refer to [The Linux filesystem explained](#)

XELK provides two pre-built root file systems, that can be used during the evaluation/development phase, since they provides a rich set of packages for working with the Axel platform:

- LTIB root file system (directory rfs/ltib). This is the default root file system
- Yocto root file system (directory rfs/yocto). This is an optional root file system

LTIB root file system is provided by Freescale as part of the Linux BSP package. Moreover, it can be re-built from scratch or updated using the officially supported LTIB tool.

4.3.2.5 Using a build tool

When rebuilding of some of the software packages is required, an automated build tool can be very helpful. Both LTIB and Yocto can be used, the former being the officially supported tool by Freescale, the latter being the emerging

solution in the embedded world.

For more information on LTIB, please refer to the following links:

- <http://www.bitshrine.org/ltib/>
- <http://www.bitshrine.org/autodocs/LtibFaq.html>
- https://www.freescale.com/webapp/Download?colCode=L3.0.35_4.1.0_DEMO_IMAGE_BSP&appType=license&location=null&fsp=1&WT_TYPE=Board%20Support%20Packages&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=gz&WT_ASSET=Downloads&sr=108&Parent_nodeId=1337637154535695831062&Parent_pageType=product

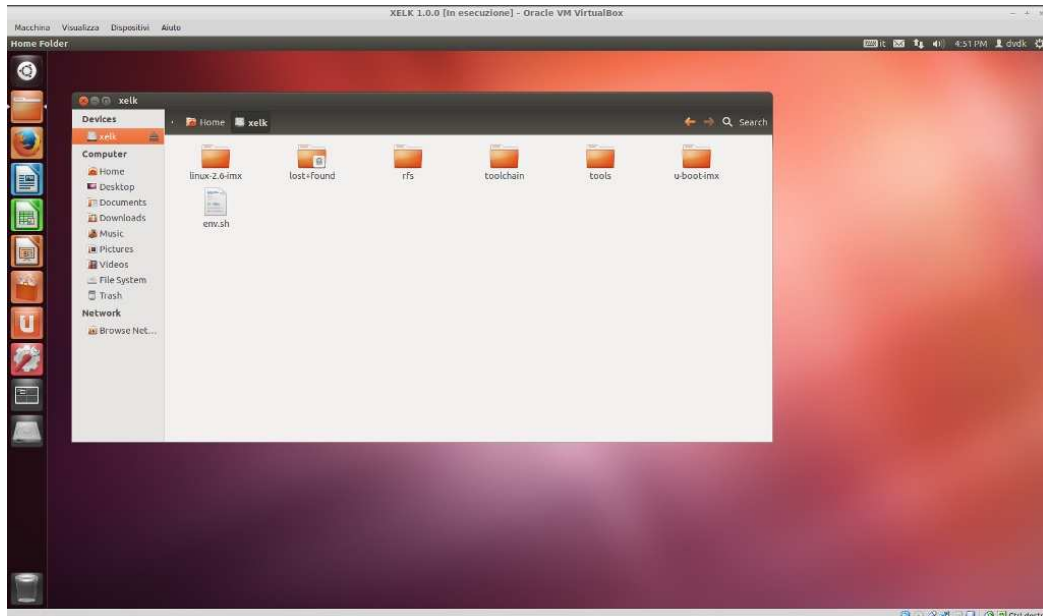
Please note that LTIB can work on Ubuntu 12.04 LTS with some adjustments (Freescale documentation reports just Ubuntu 9.04 as supported distribution). Please contact the support team for more information on how to adjust Freescale LTIB installation for running on Ubuntu 12.04.

For more information on Yocto, please refer to the following links

- <https://www.yoctoproject.org/>
- <https://community.freescale.com/docs/DOC-1616>
- <https://github.com/Freescale/meta-fsl-arm>

4.4 Working with XELK

Once the virtual machine is ready, the actual development kit can be found into the directory `/home/dvdk/xelk`:



The `xelk` directory contains the following subdirectories:

- `toolchain`: the cross-toolchain. GNU Compiler Collection (GCC) version is 4.4.3
- `linux-2.6-imx`: the Linux source tree
- `u-boot-imx` the U-Boot source tree
- `rfs`: XELK provides two root file systems:
 - `ltib`: root file system provided by Freescale with the BSP and built with LTIB (`/home/dvdk/xelk/rfs/ltib`). This is the default root file system
 - `yocto`: root file system built with the Yocto build system (`/home/dvdk/xelk/rfs/yocto`). Since Yocto is not officially supported by Freescale, this is a preliminary version, included in the XELK 1.0.0 release as a preview of the future advancements.

- `env.sh`: a bash script containing the following lines:

```
export PATH=<path_to_toolchain>:$PATH
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabi-
```

Using the following command, the environment shell variables required during the building procedures are set up:

```
source env.sh
```

4.4.1 Building u-boot

Assuming that the environment variables are set using the `env.sh` script as described above, enter the U-Boot sources directory (`xelk/u-boot-imx`) and run the following commands:

```
dvdk@dvdk-vm:~/xelk/u-boot-imx$ make
mx6qaxelbase_config
dvdk@dvdk-vm:~/xelk/u-boot-imx$ make
```

The former command selects the default Axel configuration which is used to boot from SPI NOR Flash, while the latter builds the u-boot binary images.

Subsequent builds just require `make` command, without targets, to update the binary images.

Once the build process is complete, the binary images can be copied to the `/srv/tftp/xelk/` directory with the following command:

```
dvdk@dvdk-vm:~/xelk/u-boot-imx$ sudo cp
u-boot.imx /srv/tftp/xelk/
```

4.4.2 Building Linux kernel

Assuming that the environment variables are set using the `env.sh` script as described above, enter the Linux sources directory (`xelk/linux-2.6-imx`) and run the following commands:

```
dvdk@dvdk-vm:~/xclk/linux-2.6-imx$ make
imx6_axel_defconfig
dvdk@dvdk-vm:~/xclk/linux-2.6-imx$ make uImage
```

The former command selects the default Axel configuration, while the latter builds the kernel binary image with the required u-boot header.

Default linux kernel configuration can be changed by using the standard `menuconfig`, `xconfig`, `gconfig` `make target`².

Subsequent builds just require `uImage` `make target` to update the binary image.

Once the build process is complete, the kernel binary image is stored into the `arch/arm/boot/uImage` file. This file can be copied to the `/srv/tftp/xclk/` directory with the following command:

```
dvdk@dvdk-vm:~/xclk/linux-2.6-imx$ sudo cp
arch/arm/boot/uImage /srv/tftp/xclk/
```

4.4.3 Recovery procedure

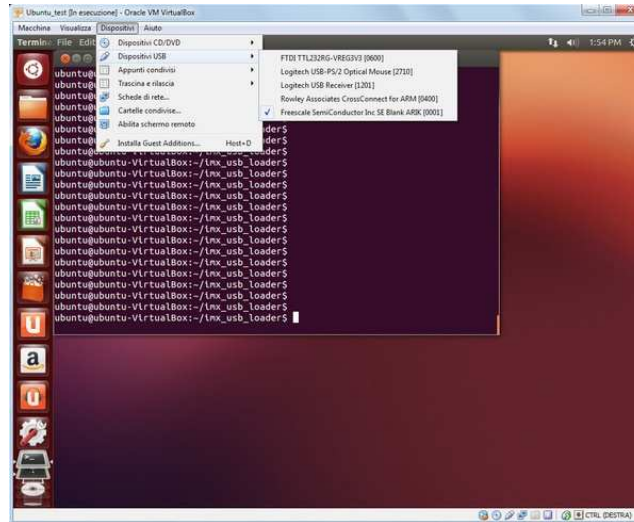
Recovery procedure using USB OTG is handled by the Linux userspace tool `imx_usb_loader` (for further details, please refer to <https://community.freescale.com/docs/DOC-94117>). The `imx_usb_loader` program is included into the XELK distribution (`~/xclk/tools/imx_usb_loader`).

Please follow the steps listed below to load an U-Boot image via USB OTG, using the XELK DVDK:

- turn off AxleEVb-Lite board
- set the dip switch S9.2 to ON position
- connect the AxleEVb-Lite USB OTG port to a PC USB port
- if not already running, start the XELK DVDK
- copy the u-boot image (`u-boot.imx`) to the `imx_usb_loader` directory
- turn on AxleEVb-Lite board. The DVDK should recognize

² The `gconfig` target requires the installation of the GTK+ libraries; the `xconfig` target requires the installation of the Qt libraries. These libraries must be installed manually by the developer using the Ubuntu package management system.

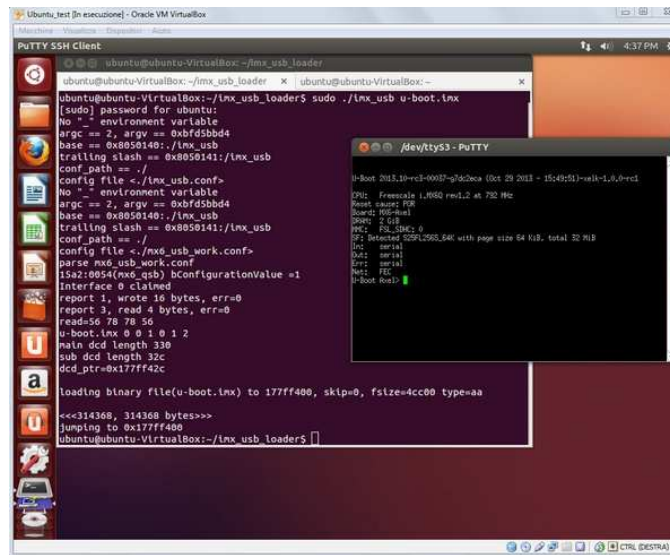
a new USB device named "Freescale Semiconductor INC Se Blank Arik". Add this device on the VM:



- from the VM console, launch the following command:

```
sudo ./lnx_usb u-boot.imx
```

lnx_usb loads the u-boot image on the volatile memory of the Axcel module, and the u-boot prompt appears on the serial console.



5 Frequently Asked Questions

5.1 Q: Where can I found Axel SOM information?

A: please refer to the following table:

Document	Location
Axel main page on DAVE Embedded Systems Developers Wiki	http://wiki.dave.eu/index.php/Category:Axel
Axel Hardware Manual	http://www.dave.eu/sites/default/files/files/axel-hm.pdf
Axel Software Manual	http://wiki.dave.eu/index.php/Software_Manual_(Axel)
Axel product page	http://www.dave.eu/dave-cpu-module-imx6-axel.html

5.2 Q: I've received the XELK package. How am I supposed to start working with it?

A: You can follow the steps listed below:

1. Check the kit contents with the packing list included in the box
2. Insert the SD into the card slot on the carrier board
3. Connect the power supply adapter and the serial cable as described in Section 3.2
4. Start your terminal emulator program
5. Switch on the power supply
6. Monitor the boot process on the serial console
7. Install the DVDK virtual machine image (Section 3.4)
8. Check the virtual machine components (please refer to Section 4.4)

5.3 Q: How can I update the XELK version?

A: please refer to the following page on the **DAVE Embedded Systems** Developer's Wiki:
http://wiki.dave.eu/index.php/Software_Manual_%28Axel%29#XELK_Updates

5.4 Q: How can I work with the XYZ peripheral/interface?

A: please refer to the “i.MX 6Dual/6Quad Linux Reference Manual” provided with the L3.0.35_4.1.0 documentation package (https://www.freescale.com/webapp/Download?colCode=L3.0.35_4.1.0_LINUX_DOCS&location=null&fpSP=1&WT_TYPE=Supporting%20Information&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=gz&WT_ASSET=Documentation&Parent_nodeId=1337637154535695831062&Parent_pageType=product).

5.5 How can I configure the Axel system to boot from network?

A: booting from network is very helpful during the software development (both for kernel and applications). The kernel image is downloaded via TFTP while the root file system is remotely mounted via NFS from the host. It is assumed that the development host:

- is connected with the target host board through an Ethernet LAN
- exports the directory containing the root file system for the target through the NFS server
- runs a TFTP server.
- has a proper subnet IP address

If your system does not match this configuration, just change the necessary variables and store them permanently with the u-boot `setenv/saveenv` commands. To do that, from the U-boot shell, please check the following parameters and set them accordingly with your host and

target configuration:

Parameter	Description	Default
serverip	IP address of the host machine running the tftp/nfs server	192.168.0.23
ipaddress	IP address of the target	192.168.0.60
ethaddr	MAC address of the target	00:50:c2:1e:af:af
netmask	Netmask of the target	255.255.255.0
gatewayip	IP address of the gateway	192.168.0.254
netdev	Ethernet device name	eth0
rootpath	Path to the NFS-exported directory	/home/dvdk/delk/rfs/tisdk
bootfile	Path to the kernel binary image on the tftp server	delk/ulmage
nfsargs	Kernel command line with parameters for loading the root file system through NFS	setenv bootargs root=/dev/nfs rw nfsroot=\${serverip}:\${rootpath} rootdelay=2

To run this configuration, just enter the command

```
run net_nfs
```

5.6 Q: Can you suggest some guidelines for the carrier board design?

A: As a starting point, you can refer to the Wiki page dedicated to the carrier board design guidelines (http://wiki.dave.eu/index.php/Carrier_board_design_guidelines_%28SOM%29), that will highlight some best practices that applies to all SOMs. For specific information on Axel, please refer to the Axel Integration Guide (http://wiki.dave.eu/index.php/Integration_guide_%28Axel%29)

6 Appendices

6.1 U-Boot startup and environment

```

U-Boot 2013.10 (Oct 31 2013 - 11:12:38)-xclk-1.0.0

CPU:   Freescale i.MX6Q rev1.2 at 792 MHz
Reset cause: WDOG
Board: MX6-Axel
DRAM:  2 GiB
MMC:   FSL_SDHC: 0
SF: Detected S25FL256S_64K with page size 64 KiB, total 32 MiB
In:    serial
Out:   serial
Err:   serial
Net:   FEC

U-Boot Axcl> print
addcons=setenv bootargs ${bootargs} console=ttyMxc2,115200n8 earlyprintk
addip=setenv bootargs ip=${ipaddr}:${serverip}:${gatewayip}:${netmask}:${hostname}:${netdev}:off panic=1 fec_mac=${ethaddr}
addlvs1=setenv bootargs ${bootargs} video=mxcfb0:dev=ldb,LDB-AM-800480STMQW-TA1,if=RGB666
ldb=spl0
addmisc=setenv bootargs ${bootargs} maxcpus=2 vmalloc=400M
baudrate=115200
bootdelay=3
ethact=FEC
ethaddr=00:50:c2:1e:af:be
ethprime=FEC0
fileaddr=12000000
filesize=4cc00
ipaddr=192.168.0.91
kernel=xclk/uImage
load=tftp ${loadaddr} ${uboot}
loadaddr=0x12000000
mmcargs=setenv bootargs ${bootargs} root=${mmcroot}
mmcboot=run mmcargs addcons addmisc mmcloadk; bootm ${loadaddr}hostname=axel
mmcloadk=fatload mmc 0:1 ${loadaddr} uImage
mmcroot=/dev/mmcblk0p2 rootwait rw'
net_nfs=tftpboot ${loadaddr} ${serverip}:${kernel}; run addip; run nfsargs addcons addmisc
addlvs1; bootm ${loadaddr}
netdev=eth0
netmask=255.255.255.0
nfsargs=setenv bootargs ${bootargs} root=/dev/nfs rw nfsroot=${serverip}:${nfsroot},v3,tcp
nfsroot=/home/dvdk/xclk/rfs/ltib
serverip=192.168.0.92
spi_update=sf probe; sf erase 0 60000;sf write ${loadaddr} 400 60000
uboot=xclk/u-boot.imx
ver=U-Boot 2013.10 (Oct 31 2013 - 11:12:38)-xclk-1.0.0

Environment size: 1282/65531 bytes
U-Boot Axcl>

```

6.2 Boot messages on the serial console

The following messages will be printed on serial console during the boot process (please note that messages may vary for different U-Boot/Linux releases):

```

U-Boot 2013.10-rc3-g7dc2eca (Oct 31 2013 - 11:12:38)-xclk-1.0.0-rc1

CPU:   Freescale i.MX6Q rev1.2 at 792 MHz
Reset cause: WDOG

```



```

[ 0.000000] Linux version 3.0.35 (dvdk@dvdk-vm) (gcc version 4.4.3 (GCC) ) #1 SMP PREEMPT
Thu Oct 31 12:32:00 CET 2013
[ 0.000000] CPU: ARMv7 Processor [412fc09a] revision 10 (ARMv7), cr=10c53c7d
[ 0.000000] CPU: VIPT nonaliasing data cache, VIPT aliasing instruction cache
[ 0.000000] Machine: DAVE i.MX 6Quad/DualLite/Solo Axel Board
[ 0.000000] Memory policy: ECC disabled, Data cache writealloc
[ 0.000000] CPU identified as i.MX6Q, silicon rev 1.2
[ 0.000000] PERCPU: Embedded 7 pages/cpu @8c810000 s5312 r8192 d15168 u32768
[ 0.000000] Built 1 zonelists in Zone order, mobility grouping on. Total pages: 454656
[ 0.000000] Kernel command line: ip=192.168.0.91:192.168.0.92::255.255.255.0::eth0:off
panic=1 eth=00:50:c2:1e:af:be root=/dev/nfs rw
nfsroot=192.168.0.92:/home/dvdk/xelk/rfs/ltib,v3,tcp console=ttymx2,115200n8 earlyprintk
maxcpus=2 vmalloc=400M video=mxcfb0:dev=ldb,LDB-AM-800480STMQW-TA1,if=RGB666 ldb=spl0
[ 0.000000] PID hash table entries: 4096 (order: 2, 16384 bytes)
[ 0.000000] Dentry cache hash table entries: 262144 (order: 8, 1048576 bytes)
[ 0.000000] Inode-cache hash table entries: 131072 (order: 7, 524288 bytes)
[ 0.000000] Memory: 512MB 1280MB = 1792MB total
[ 0.000000] Memory: 1805432k/1805432k available, 291720k reserved, 638976K highmem
[ 0.000000] Virtual kernel memory layout:
[ 0.000000]   vector   : 0xffff0000 - 0xffff1000   ( 4 kB)
[ 0.000000]   fixmap   : 0xffff0000 - 0xffffe000   ( 896 kB)
[ 0.000000]   DMA     : 0xf4600000 - 0xffe00000   ( 184 MB)
[ 0.000000]   vmalloc  : 0xd9800000 - 0xf2000000   ( 392 MB)
[ 0.000000]   lowmem   : 0x80000000 - 0xd9000000   (1424 MB)
[ 0.000000]   pkmap   : 0x7fe00000 - 0x80000000   ( 2 MB)
[ 0.000000]   modules  : 0x7f000000 - 0x7fe00000   ( 14 MB)
[ 0.000000]     .init  : 0x80008000 - 0x80038000   ( 192 kB)
[ 0.000000]     .text  : 0x80038000 - 0x80a87ba0   (10559 kB)
[ 0.000000]     .data  : 0x80a88000 - 0x80ade4a0   ( 346 kB)
[ 0.000000]     .bss  : 0x80ade4c4 - 0x80b2c36c   ( 312 kB)
[ 0.000000] SLUB: Genslabs=13, HWalign=32, Order=0-3, MinObjects=0, CPUs=4, Nodes=1
[ 0.000000] Preemptible hierarchical RCU implementation.
[ 0.000000] NR_IRQS:624
[ 0.000000] MXC GPIO hardware
[ 0.000000] sched_clock: 32 bits at 3000kHz, resolution 333ns, wraps every 1431655ms
[ 0.000000] arm_max_freq=1GHz
[ 0.000000] MXC_Early serial console at MMIO 0x21ec000 (options '115200n8')
[ 0.000000] bootconsole [ttymx2] enabled
[ 0.000000] Console: colour dummy device 80x30
[ 0.229398] Calibrating delay loop... 1581.05 BogoMIPS (lpj=7905280)
[ 0.317704] pid_max: default: 32768 minimum: 301
[ 0.322693] Mount-cache hash table entries: 512
[ 0.328160] CPU: Testing write buffer coherency: ok
[ 0.333354] hw perfevents: enabled with ARMv7 Cortex-A9 PMU driver, 7 counters available
[ 0.433914] CPU1: Booted secondary processor
[ 0.473403] Brought up 2 CPUs
[ 0.480667] SMP: Total of 2 processors activated (3162.11 BogoMIPS).
[ 0.487661] devtmpfs: initialized
[ 0.511078] print_constraints: dummy:
[ 0.515128] NET: Registered protocol family 16
[ 0.524546] print_constraints: vddpu: 725 <--> 1300 mV at 1150 mV fast normal
[ 0.532146] print_constraints: vddcore: 725 <--> 1300 mV at 1150 mV fast normal
[ 0.539954] print_constraints: vddsoc: 725 <--> 1300 mV at 1200 mV fast normal
[ 0.547710] print_constraints: vdd2p5: 2000 <--> 2775 mV at 2400 mV fast normal
[ 0.555548] print_constraints: vdd1p1: 800 <--> 1400 mV at 1100 mV fast normal
[ 0.563233] print_constraints: vdd3p0: 2625 <--> 3400 mV at 3000 mV fast normal
[ 0.706626] failed to get GPIO AXEL_USB_H1_PWR: -16
[ 0.714280] L310 cache controller enabled
[ 0.718317] l2x0: 16 ways, CACHE_ID 0x410000c7, AUX_CTRL 0x02070000, Cache size: 1048576
B
[ 0.751836] bio: create slab <bio-0> at 0
[ 0.759721] mxs-dma mxs-dma-apbh: initialized
[ 0.764502] print_constraints: vmmc: 3300 mV
[ 0.770257] SCSI subsystem initialized
[ 0.775330] usbcore: registered new interface driver usbfs
[ 0.781111] usbcore: registered new interface driver hub
[ 0.786642] usbcore: registered new device driver usb
[ 0.791715] Freescale USB OTG Driver loaded, $Revision: 1.55 $
[ 0.993320] mc_pfuze 1-0008: recv failed!:-5,0
[ 0.997793] mc_pfuze: probe of 1-0008 failed with error -1
[ 1.023396] imx-ipuv3 imx-ipuv3.0: IPU DMFC NORMAL mode: 1(0~1), 5B(4,5), 5F(6,7)
[ 1.043393] imx-ipuv3 imx-ipuv3.1: IPU DMFC NORMAL mode: 1(0~1), 5B(4,5), 5F(6,7)
[ 1.051331] MIPI CSI2 driver module loaded

```

```
[ 1.055710] Advanced Linux Sound Architecture Driver Version 1.0.24.
[ 1.063056] Bluetooth: Core ver 2.16
[ 1.066801] NET: Registered protocol family 31
[ 1.071261] Bluetooth: HCI device and connection manager initialized
[ 1.077653] Bluetooth: HCI socket layer initialized
[ 1.082550] Bluetooth: L2CAP socket layer initialized
[ 1.087736] Bluetooth: SCO socket layer initialized
[ 1.093131] cfg80211: Calling CRDA to update world regulatory domain
[ 1.100000] Switching to clocksource mxc_timer1
[ 1.121395] NET: Registered protocol family 2
[ 1.126042] IP route cache hash table entries: 65536 (order: 6, 262144 bytes)
[ 1.134269] TCP established hash table entries: 262144 (order: 9, 2097152 bytes)
[ 1.145509] TCP bind hash table entries: 65536 (order: 7, 786432 bytes)
[ 1.153244] TCP: Hash tables configured (established 262144 bind 65536)
[ 1.159905] TCP reno registered
[ 1.163066] UDP hash table entries: 1024 (order: 3, 32768 bytes)
[ 1.169166] UDP-Lite hash table entries: 1024 (order: 3, 32768 bytes)
[ 1.176196] NET: Registered protocol family 1
[ 1.180938] RPC: Registered named UNIX socket transport module.
[ 1.186906] RPC: Registered udp transport module.
[ 1.191626] RPC: Registered tcp transport module.
[ 1.196360] RPC: Registered tcp NFSv4.1 backchannel transport module.
[ 1.203856] PMU: registered new PMU device of type 0
[ 1.208995] Static Power Management for Freescale i.MX6
[ 1.214240] wait mode is enabled for i.MX6
[ 1.218567] cpaddr = d9880000 suspend_iram_base=d9918000
[ 1.223978] PM driver module loaded
[ 1.227783] IMX usb wakeup probe
[ 1.231590] add wake up source irq 75
[ 1.241963] IMX usb wakeup probe
[ 1.245530] cpu regulator mode:ldo_enable
[ 1.249749] i.MXC CPU frequency driver
[ 1.279495] highmem bounce pool size: 64 pages
[ 1.300323] JFFS2 version 2.2. (NAND) © 2001-2006 Red Hat, Inc.
[ 1.307076] msgmni has been set to 2278
[ 1.312702] alg: No test for stdrng (krng)
[ 1.316987] io scheduler noop registered
[ 1.320924] io scheduler deadline registered
[ 1.325310] io scheduler cfq registered (default)
[ 1.331659] MIPI DSI driver module loaded
[ 1.335891] mxc_sdc_fb mxc_sdc_fb.0: register mxc display driver ldb
[ 1.342284] _regulator_get: get() with no identifier
[ 1.416964] Console: switching to colour frame buffer device 100x30
[ 1.453034] mxc_sdc_fb mxc_sdc_fb.1: register mxc display driver ldb
[ 1.459444] mxc_ldb mxc_ldb: Input pixel format not valid use default RGB666
[ 1.466529] mxc_ldb mxc_ldb: for second ldb displdb mode should in separate mode
[ 1.473951] mxc_sdc_fb mxc_sdc_fb.1: NO mxc display driver found!
[ 1.480098] mxc_sdc_fb mxc_sdc_fb.2: register mxc display driver lcd
[ 1.491229] mxc_sdc_fb mxc_sdc_fb.3: register mxc display driver ldb
[ 1.497626] mxc_ldb mxc_ldb: for second ldb displdb mode should in separate mode
[ 1.505057] mxc_sdc_fb mxc_sdc_fb.3: NO mxc display driver found!
[ 1.512017] imx-sdma imx-sdma: loaded firmware 1.1
[ 1.521858] imx-sdma imx-sdma: initialized
[ 1.670785] Serial: IMX driver
[ 1.673984] imx-uart.2: ttyxc2 at MMIO 0x21ec000 (irq = 60) is a IMX
[ 1.680490] console [ttyxc2] enabled, bootconsole disabled
[ 1.680490] console [ttyxc2] enabled, bootconsole disabled
[ 1.692103] imx-uart.0: ttyxc0 at MMIO 0x2020000 (irq = 58) is a IMX
[ 1.698973] imx-uart.3: ttyxc3 at MMIO 0x21f0000 (irq = 61) is a IMX
[ 1.705813] imx-uart.1: ttyxc1 at MMIO 0x21e8000 (irq = 59) is a IMX
[ 1.718643] loop: module loaded
[ 1.765257] No sata disk.
[ 1.769286] GPMI NAND driver registered. (IMX)
[ 1.774867] vcan: Virtual CAN interface driver
[ 1.779317] CAN device driver interface
[ 1.783160] flexcan netdevice driver
[ 1.787778] flexcan imx6q-flexcan.0: device registered (reg_base=d9a50000, irq=142)
[ 1.795583] FEC Ethernet Driver
[ 1.801278] fec_enet_mii_bus: probed
[ 1.806326] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
[ 1.813056] fsl-ehci fsl-ehci.0: Freescale On-Chip EHCI Host Controller
[ 1.820099] fsl-ehci fsl-ehci.0: new USB bus registered, assigned bus number 1
[ 1.854643] fsl-ehci fsl-ehci.0: irq 75, io base 0x02184000
```

```
[ 1.874624] fsl-ehci fsl-ehci.0: USB 2.0 started, EHCI 1.00
[ 1.880300] usb usb1: New USB device found, idVendor=1d6b, idProduct=0002
[ 1.887109] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 1.894339] usb usb1: Product: Freescale On-Chip EHCI Host Controller
[ 1.900795] usb usb1: Manufacturer: Linux 3.0.35-gf76c146 ehci_hcd
[ 1.906992] usb usb1: SerialNumber: fsl-ehci.0
[ 1.912126] hub 1-0:1.0: USB hub found
[ 1.915906] hub 1-0:1.0: 1 port detected
[ 1.920303] add wake up source irq 72
[ 1.924008] fsl-ehci fsl-ehci.1: Freescale On-Chip EHCI Host Controller
[ 1.930950] fsl-ehci fsl-ehci.1: new USB bus registered, assigned bus number 2
[ 1.964642] fsl-ehci fsl-ehci.1: irq 72, io base 0x02184200
[ 1.984622] fsl-ehci fsl-ehci.1: USB 2.0 started, EHCI 1.00
[ 1.990287] usb usb2: New USB device found, idVendor=1d6b, idProduct=0002
[ 1.997096] usb usb2: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 2.004326] usb usb2: Product: Freescale On-Chip EHCI Host Controller
[ 2.010782] usb usb2: Manufacturer: Linux 3.0.35-gf76c146 ehci_hcd
[ 2.017006] usb usb2: SerialNumber: fsl-ehci.1
[ 2.022128] hub 2-0:1.0: USB hub found
[ 2.025907] hub 2-0:1.0: 1 port detected
[ 2.030315] Initializing USB Mass Storage driver...
[ 2.035393] usbcore: registered new interface driver usb-storage
[ 2.041404] USB Mass Storage support registered.
[ 2.046045] ARC USBOTG Device Controller driver (1 August 2005)
[ 2.052801] gs_bind: controller 'fsl-usb2-udc' not recognized
[ 2.058654] g_serial gadget: Gadget Serial v2.4
[ 2.063205] g_serial gadget: g_serial ready
[ 2.067407] Suspend udc for OTG auto detect
[ 2.075032] fsl-usb2-udc: bind to driver g_serial
[ 2.080398] mousedev: PS/2 mouse device common for all mice
[ 2.594599] tsc2007 2-0048: i2c io error: -110
[ 2.599862] i2c-core: driver [isl29023] using legacy suspend method
[ 2.606153] i2c-core: driver [isl29023] using legacy resume method
[ 2.612493] i2c /dev entries driver
[ 2.616915] Linux video capture interface: v2.00
[ 2.621956] mxc_v4l2_output mxc_v4l2_output.0: V4L2 device registered as video16
[ 2.629624] mxc_v4l2_output mxc_v4l2_output.0: V4L2 device registered as video17
[ 2.637300] mxc_v4l2_output mxc_v4l2_output.0: V4L2 device registered as video18
[ 2.644969] mxc_v4l2_output mxc_v4l2_output.0: V4L2 device registered as video19
[ 2.652984] imx2-wdt imx2-wdt.0: IMX2+ Watchdog Timer enabled. timeout=60s (nowayout=1)
[ 2.661117] Bluetooth: Virtual HCI driver ver 1.3
[ 2.666079] Bluetooth: HCI UART driver ver 2.2
[ 2.670527] Bluetooth: HCIATH3K protocol initialized
[ 2.675672] sdhci: Secure Digital Host Controller Interface driver
[ 2.681856] sdhci: Copyright(c) Pierre Ossman
[ 2.686319] sdhci sdhci-esdhc-imx.1: no card-detect pin available!
[ 2.695185] mmc0: SDHCI controller on platform [sdhci-esdhc-imx.1] using DMA
[ 2.702535] mxc_vdoa mxc_vdoa: i.MX Video Data Order Adapter(VDOA) driver probed
[ 2.710649] VPU initialized
[ 2.714504] mxc_asrc registered
[ 2.717833] Galcore version 4.6.9.6622
[ 2.742654] Thermal calibration data is 0x5734b57d
[ 2.749090] Thermal sensor with ratio = 180
[ 2.774900] Anatop Thermal registered as thermal_zone0
[ 2.782656] anatop_thermal_probe: default cooling device is cpufreq!
[ 2.789913] usbcore: registered new interface driver usbhid
[ 2.798716] usbhid: USB HID core driver
[ 2.803446] mxc_hdmi_soc mxc_hdmi_soc.0: MXC HDMI Audio
[ 2.809628] imx-hdmi-soc-dai imx-hdmi-soc-dai.0: Failed: Load HDMI-video first.
[ 2.828644] Initialize HDMI-audio failed. Load HDMI-video first!
[ 2.836295] ALSA device list:
[ 2.840884]   No soundcards found.
[ 2.846184] NET: Registered protocol family 26
[ 2.855463] TCP cubic registered
[ 2.858698] NET: Registered protocol family 17
[ 2.863184] can: controller area network core (rev 20090105 abi 8)
[ 2.871080] NET: Registered protocol family 29
[ 2.877182] can: raw protocol (rev 20090105)
[ 2.883084] can: broadcast manager protocol (rev 20090105 t)
[ 2.890462] Bluetooth: RFCOMM TTY layer initialized
[ 2.897813] Bluetooth: RFCOMM socket layer initialized
[ 2.902958] Bluetooth: RFCOMM ver 1.11
[ 2.906723] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
```



```
[ 2.912038] Bluetooth: BNEP filters: protocol multicast
[ 2.920487] Bluetooth: HIDP (Human Interface Emulation) ver 1.2
[ 2.926552] lib80211: common routines for IEEE802.11 drivers
[ 2.932258] VFP support v0.3: implementor 41 architecture 3 part 30 variant 9 rev 4
[ 2.941768] Bus freq driver module loaded
[ 2.950615] Bus freq driver Enabled
[ 2.967611] mxc_dvfs_core_probe
[ 2.973348] DVFS driver module loaded
[ 2.978674] drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
[ 2.998167] eth0: Freescale FEC PHY driver [Micrel KSZ9031 Gigabit PHY]
(miibus:phy_addr=1:07, irq=-1)
[ 7.995050] PHY: 1:07 - Link is Up - 1000/Full
[ 8.024954] IP-Config: Complete:
[ 8.028205]     device=eth0, addr=192.168.0.91, mask=255.255.255.0, gw=255.255.255.255,
[ 8.036169]     host=192.168.0.91, domain=, nis-domain=(none),
[ 8.042125]     bootserver=192.168.0.92, rootserver=192.168.0.92, rootpath=
[ 8.073396] VFS: Mounted root (nfs filesystem) on device 0:14.
[ 8.080298] devtmpfs: mounted
[ 8.083482] Freeing init memory: 192K
starting pid 1298, tty '': '/etc/rc.d/rcS'
Mounting /proc and /sys
Starting the hotplug events dispatcher udevd
[ 8.452902] udevd (1307): /proc/1307/oom_adj is deprecated, please use
/proc/1307/oom_score_adj instead.
Synthesizing initial hotplug events
Setting the hostname to freescale
Mounting filesystems
Starting the dropbear ssh server:
D-Bus per-session daemon address is:
unix:abstract=/tmp/dbus-xDt3toyBZF,guid=7103e28b9a94a9c9d389866000000000
gtk: creating gdk-pixbuf.loaders
pango: creating module list
starting pid 2329, tty '': '/etc/rc.d/rc_gpu.S'
_XSERVTransSocketOpenCOTSServer: Unable to open socket for inet6
_XSERVTransOpen: transport open failed for inet6/freescale:0
_XSERVTransMakeAllCOTSServerListeners: failed to open listener for inet6
starting pid 2341, tty '': '/etc/rc.d/rc_mxc.S'
(EE) XKB: Couldn't open rules file /usr/share/X11/xkb/rules/base
(EE) XKB: No components provided for device Virtual core keyboard
arm-none-linux-gnueabi-gcc (Freescale MAD -- Linaro 2011.07 -- Built at 2011/08/10 09:20)
4.6.2 20110630 (prerelease)
root filesystem built on Fri, 16 Aug 2013 20:23:44 +0800
Freescale Semiconductor, Inc.
freescale login: matchbox: keyboard does not appear to have a <alt> key.
matchbox: ignoring key shortcut <Alt>n=next
matchbox: keyboard does not appear to have a <alt> key.
matchbox: ignoring key shortcut <Alt>p=prev
matchbox: keyboard does not appear to have a <alt> key.
matchbox: ignoring key shortcut <Alt>c=close
matchbox: keyboard does not appear to have a <alt> key.
matchbox: ignoring key shortcut <Alt>d=desktop
matchbox: keyboard does not appear to have a <alt> key.
matchbox: ignoring key shortcut <Alt>m=!matchbox-remote -mbmenu
matchbox: keyboard does not appear to have a <alt> key.
matchbox: ignoring key shortcut <ctrl><alt>x=!xterm
matchbox: keyboard does not appear to have a <alt> key.
matchbox: ignoring key shortcut <ctrl><alt>r=!rxvt
matchbox: keyboard does not appear to have a <alt> key.
matchbox: ignoring key shortcut <ctrl><alt>e=!gpe-calender
matchbox: keyboard does not appear to have a <alt> key.
matchbox: ignoring key shortcut <alt>Tab=next
matchbox: keyboard does not appear to have a <alt> key.
matchbox: ignoring key shortcut <alt><shift>Tab=prev
matchbox: keyboard does not appear to have a <alt> key.
matchbox: ignoring key shortcut <alt>space=taskmenu
matchbox: keyboard does not appear to have a <alt> key.
matchbox: ignoring key shortcut <alt>escape=!matchbox-remote -mbmenu
matchbox: keyboard does not appear to have a <alt> key.
matchbox: ignoring key shortcut <alt>f4=close
matchbox: Cant find a keycode for keysym 65480
matchbox: ignoring key shortcut f11=fullscreen
matchbox-desktop: loading /usr/lib/matchbox/desktop/tasks.so with args ( None )
matchbox-desktop: loading /usr/lib/matchbox/desktop/dotdesktop.so with args ( None )
```

```
user_overrides is (nil)
mb-desktop-dotdesktop: failed to open /usr/local/share/applications
mb-desktop-dotdesktop: failed to open //.applications
mb-applet-menu-launcher: bonobo-browser.desktop has no icon, png or name
mb-applet-menu-launcher: failed to open /usr/local/share/applications
mb-applet-menu-launcher: failed to open //.applications
mb-applet-menu-launcher: bonobo-browser.desktop has no icon, png or name
mb-applet-menu-launcher: failed to open /usr/local/share/applications
mb-applet-menu-launcher: failed to open //.applications
arm-none-linux-gnueabi-gcc (Freescale MAD -- Linaro 2011.07 -- Built at 2011/08/10 09:20)
4.6.2 20110630 (prerelease)
root filesystem built on Fri, 16 Aug 2013 20:23:44 +0800
Freescale Semiconductor, Inc.

freescale login: root
login[2345]: root login on 'ttymxc2'

BusyBox v1.20.2 () built-in shell (ash)
Enter 'help' for a list of built-in commands.

root@freescale ~$
```