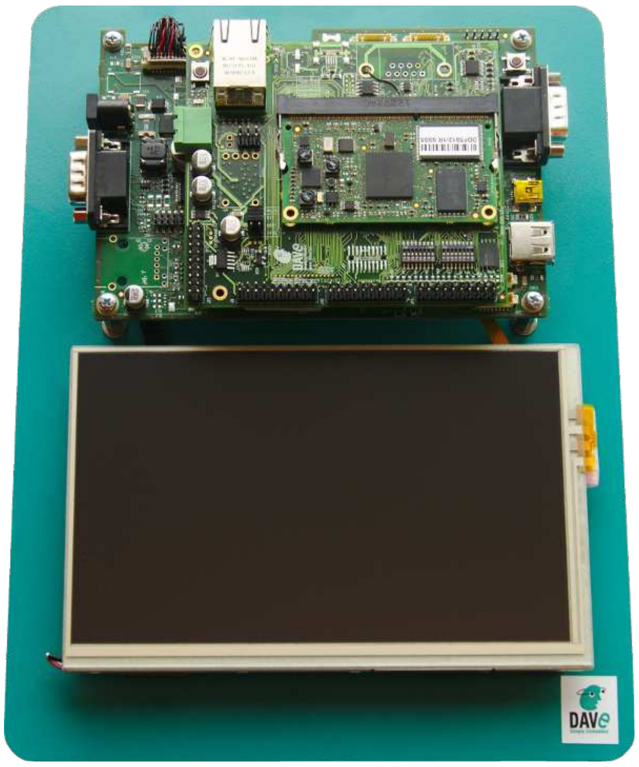
diVa

ARM Cortex-A8 CPU Module Family

Lite Line

Diva Embedded Linux Kit (*DIVELK*)

Quick Start Guide



DAVE Simply Embedded www.dave.eu info@dave.eu

<Page intentionally left blank>

Table of Contents

1 Preface

1.1 About this manual

This manual describes the Diva Embedded Linux Kit (DIVELK) 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 SrL is certified to ISO 9001 standards.

1.4 Disclaimers

DAVE 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 Diva CPU module, the DivaEVB-Lite carrier board and the Dacu carrier board.

Diva 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 Srl 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 Srl for any damage resulting from such improper use or sale.

1.5 Warranty

Diva SOM, DivaEVB-Lite and Dacu are guaranteed against defects in material and workmanship for the warranty period from the shipment date. During the warranty period, Dave SrL 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 SrL will not be responsible for any defects or damages to other products not supplied by Dave SrL that are caused by a faulty Diva module, DivaEVB-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-diva@dave.eu. Software upgrades are available for download in the restricted download area of DAVE 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-am335x-diva.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 Developers Wiki	http://wiki.dave.eu/index.php/Main_Page
AM335x Technical Reference Manual	http://www.ti.com/lit/ug/spruh73h/spruh73h.pdf
Diva main page on Dave Developers Wiki	http://wiki.dave.eu/index.php/Category:Diva
Diva Hardware Manual	http://www.dave.eu/sites/default/files/files/diva-hm.pdf
Diva Software Manual	http://wiki.dave.eu/index.php/Software_Manual_(Diva)
AM335x Portal (on TI Embedded Processors Wiki)	http://processors.wiki.ti.com/index.php/Sitara_AM335x_Portal
DivaEVB-Lite page on Dave Developers Wiki	http://wiki.dave.eu/index.php/DivaEVB-Lite
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
DIVELK	Diva Embedded Linux Kit
EMAC	Ethernet Media Access Controller
GPI	General purpose input
GPIO	General purpose input and output
GPO	General purpose output
PCB	Printed circuit board
PMIC	Power Management Integrated Circuit
PRU	Programmable Real-Time Unit
PSU	Power supply unit
RTC	Real time clock
SOC	System-on-chip
SO-DIMM	Small Outline Dual In-line Memory Module
SOM	System-on-module
WDT	Watchdog

Tab. 2: Abbreviations and acronyms used in this manual

Revision History

Version	Date	Notes
1.0.0	June 2013	First official release
1.0.1	June 2013	Added pictures Minor fixes

2 Introduction

2.1 Diva SOM

Fig. 1: Diva CPU module

Diva is a family of system-on-modules (SOM) that belongs to DAVE's ***Lite* Line** product class. Diva is based on Texas Instruments "Sitara" AM335x Cortex-A8 application processor and is built with SO-DIMM 204 pin form factor.

Diva offers lots of graphics, processing, peripherals and industrial interface options, allowing customers to implement cost-effective design. The Programmable Real-Time Unit and Industrial Communication Subsystem (PRU-ICSS) adds further flexibility and enables additional peripheral interfaces and real-time protocols such as EtherCAT, PROFINET, EtherNet/IP, PROFIBUS, Ethernet Powerlink.

Typical applications for Diva are:

- Industrial sensors and I/O units
- Industrial drives with integrated communications and multi-axis motor control
- Programmable logic/automation controllers (PLC/PAC) with integrated industrial communications such as PROFIBUS, CAN and Ethernet
- Home and Building Automation

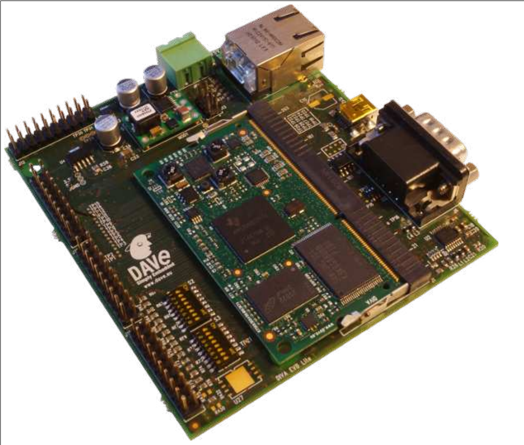


Fig. 2: Diva plugged on DivaEVB-Lite

For further information, please refer to Diva Hardware Manual.

2.2 Embedded Linux

When we talk in general about Embedded Linux[1], 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 machine.

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.

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 Diva 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 DIVELK

Diva Embedded Linux Kit (DIVELK 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 Diva-based systems
- building Linux applications that will run on the target

The heart of Diva SOM is Texas Instruments "Sitara" AM335x microprocessors. From a software point of view, Texas Instruments supports this processor family through so-called Linux EZ Software Development Kit (EZSDK for short). EZSDK releases are published on a regular basis. For more details please refer to:

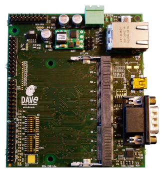
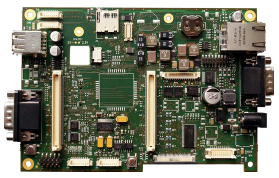
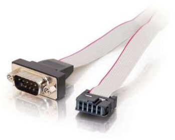
- http://www.ti.com/tool/linuxezsdk-sitara
- http://processors.wiki.ti.com/index.php/Category:EZSDK

Diva Embedded Linux Kit, in turn, is directly derived from EZSDK. Hence DIVELK documentation often refers to EZSDK resources.

Dave adds to the latest EZSDK from Texas Instruments the customization required to support the Diva platform. For this reason most of the documentation provided by TI remains valid for the DIVELK 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 DIVELK components

Component	Description
	Diva SOM CPU: TI AM3359 SDRAM: 512 MB DDR3 NOR: bootable SPI flash 32 MB NAND: 1GB
	Diva-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
	DB9 Male Serial port adapter

Component	Description
	MicroSDHC card with SD adapter and USB adapter

2.3.2 DIVELK Release Notes

2.3.2.1 Version 1.0.0

First official release

2.3.2.2 Releases history

	DIVELK Version
Release number	1.0.0
Status	Released
Release date	June 2013
Release notes	Version 1.0.0
SOM PCB version	CS133012A
Supported carrier boards	DivaEVB-Lite Dacu
U-Boot version	2012.10-divelk-1.0.0
Linux version	3.2.0-divelk-1.0.0
Drivers	SPI NOR Flash (boot) NAND 8 bit (boot) UART0 (2-wire) USB Host SD/MMC1 GPIO LCD Touch screen controller EMAC0 RMII (Fast Ethernet) PMIC RTC (battery powered) Audio (ALSA)
TI EZSDK	05.06.00.00

3 DIVELK Quick Start

This chapter describes how to quickly start working with the DIVELK 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 and fixed on a supporting plate, as shown in the picture below:

3.2 Hardware setup

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

The MicroSD provided with the DIVELK can be used to boot the system, since it contains a bootable partition (mmcblk0p1) and a root file system partition (mmcblk0p2).

1. connect the serial cable, provided with the board, to the J25 pin-strip connector on the Dacu board
2. insert the MicroSD card provided with the development kit into the MicroSD slot
3. connect the 12Vcc power supply to JP2 on the Dacu board
4. (optional) connect the serial cable to PC COM port through a NULL-modem cable (not provided)
5. (optional) start your favorite terminal software on PC; communication parameters are:

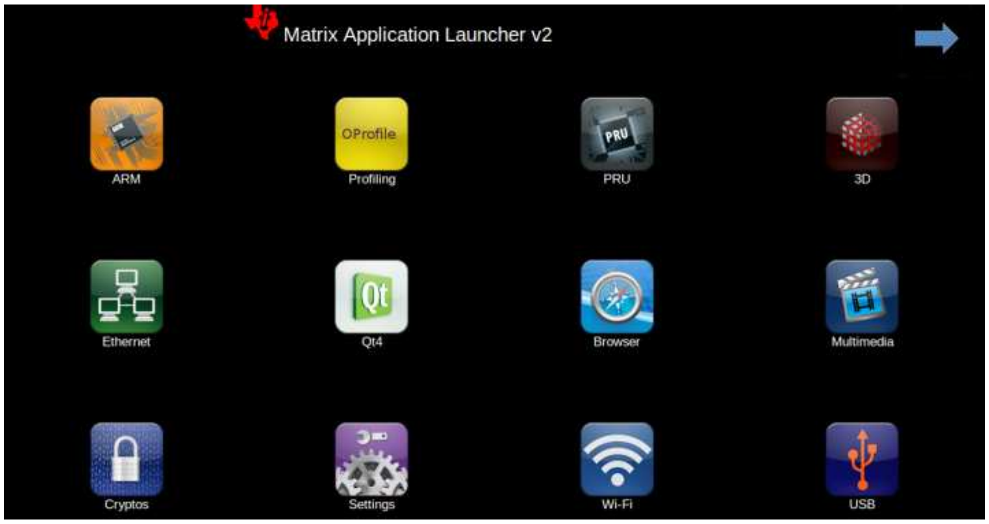
Parameter	Value
Baud rate	115200 bps
Data bits	8
Stop bits	1
Parity	None

6. (optional) to connect the system to Ethernet LAN, please plug cable on connector J7 of the DivaEVB-Lite

The system is configured to boot automatically when powered up, loading u-boot, kernel and root file system from the MicroSD card.

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 `mmcboot` 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, the **Matrix** demo application (http://processors.wiki.ti.com/index.php/Matrix_Users_Guide) 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.

3.4 Installing DVDK

Dave Virtual Development Kit is a virtual machine, based on Oracle VirtualBox that allows developers to start using Dave's 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 Developer's Wiki for further information.

3.4.1 MicroSD contents

The microSD provided with DIVELK is used to store:

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

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

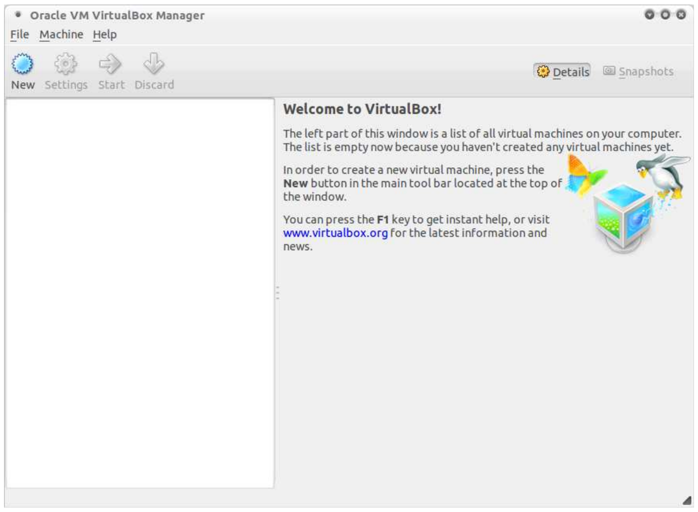
- Boot disk with pre-installed Ubuntu Linux 12.04.2 LTS and pre-configured basic Linux services (TFTP, NFS, ...)
- Secondary disk containing source code and tools:

- Bootloader (u-boot) source tree cloned from DAVE's public git repository
- Linux kernel source tree cloned from DAVE's public git repository
- Pre-installed AM335x EZSDK with setup scripts, makefiles, example applications, ...
- Toolchain

3.4.2 Importing the virtual machine

DIVELK 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



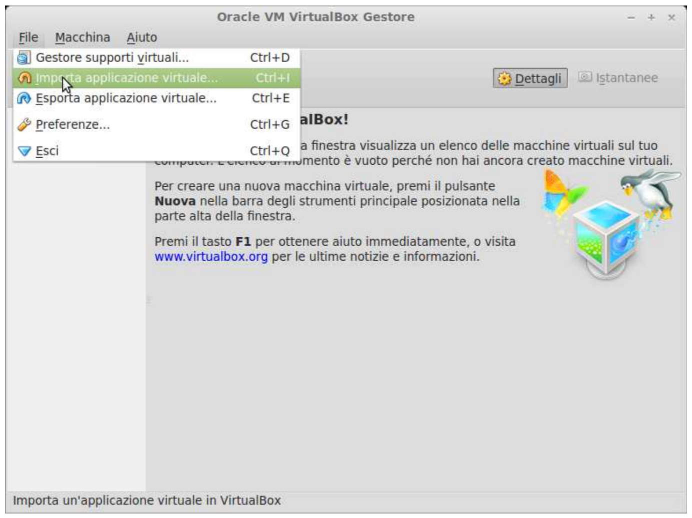
Welcome to VirtualBox!

The left part of this window is a list of all virtual machines on your computer. The list is empty now because you haven't created any virtual machines yet.

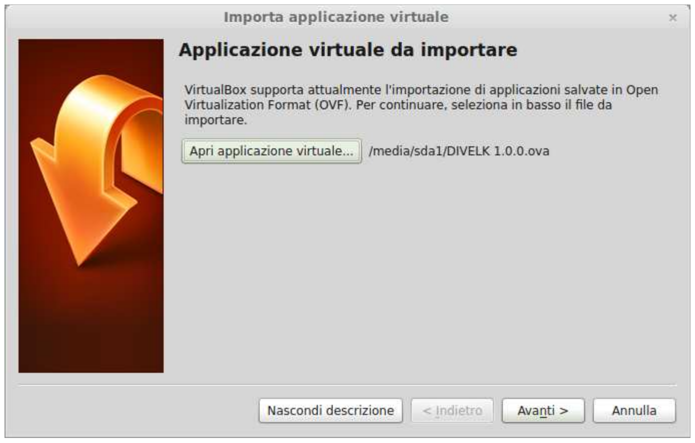
In order to create a new virtual machine, press the **New** button in the main tool bar located at the top of the window.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for the latest information and news.

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

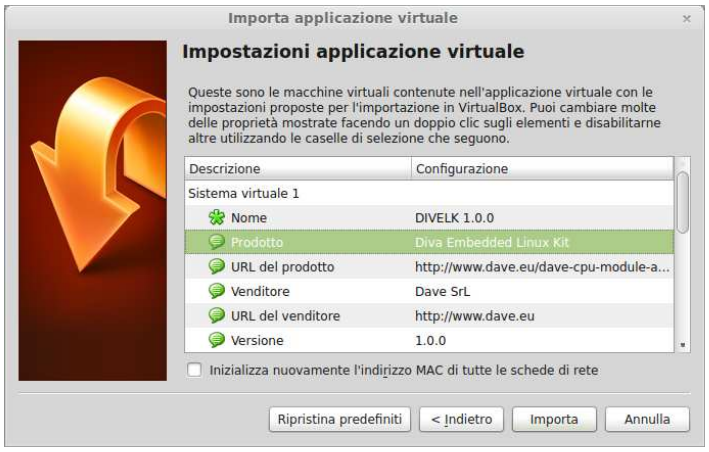

Oracle VM VirtualBox Gestore
File Macchina Aiuto
Gestore supporti virtuali... Ctrl+D
Importa applicazione virtuale... Ctrl+I
Esporta applicazione virtuale... Ctrl+E
Preferenze... Ctrl+G
Esci Ctrl+Q
Dettagli
Istantanee
Importa un'applicazione virtuale in VirtualBox

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


Importa applicazione virtuale
Applicazione virtuale da importare
Apri applicazione virtuale...
/media/sda1/DIVELK 1.0.0.ova
Nascondi descrizione
< Indietro
Avanti >
Annulla

VirtualBox supporta attualmente l'importazione di applicazioni salvate in Open Virtualization Format (OVF). Per continuare, seleziona in basso il file da importare.

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



Impostazioni applicazione virtuale

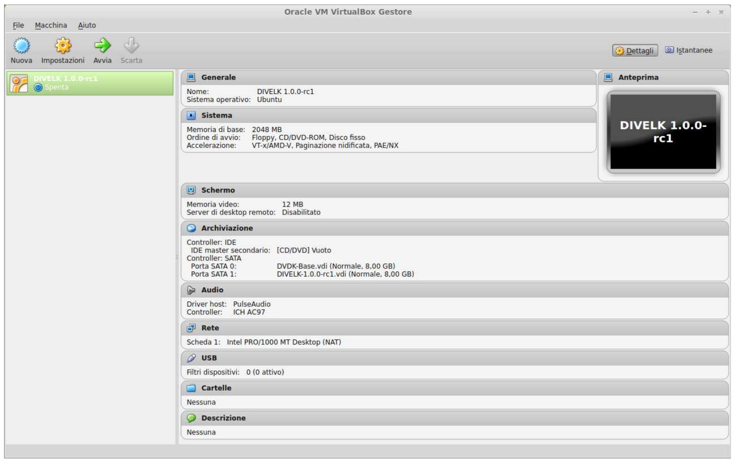
Queste sono le macchine virtuali contenute nell'applicazione virtuale con le impostazioni proposte per l'importazione in VirtualBox. Puoi cambiare molte delle proprietà mostrate facendo un doppio clic sugli elementi e disabilitarne altre utilizzando le caselle di selezione che seguono.

Descrizione	Configurazione
Sistema virtuale 1	
Nome	DIVELK 1.0.0
Prodotto	Diva Embedded Linux Kit
URL del prodotto	http://www.dave.eu/dave-cpu-module-a...
Venditore	Dave SrL
URL del venditore	http://www.dave.eu
Versione	1.0.0

☐ Inizializza nuovamente l'indirizzo MAC di tutte le schede di rete

3.4.3 Launching the virtual machine

1. Once the virtual machine is ready, launch it by clicking on the start icon



Generale
Nome: DIVELK 1.0.0-rc1
Sistema operativo: Ubuntu

Sistema
Memoria di base: 2048 MB
Ordine di avvio: Floppy, CD/DVD-ROM, Disco fisso
Accelerazione: VT-x/AMD-V, Paginazione nidificata, PAE/NX

Schermo
Memoria video: 12 MB
Server di desktop remoto: Disabilitato

Archiviazione
Controller: IDE
IDE master secondario: [CD/DVD] Vuoto
Controller: SATA
Porta SATA 0: DVDK-Base.vdi (Normale, 8,00 GB)
Porta SATA 1: DIVELK-1.0.0-rc1.vdi (Normale, 8,00 GB)

Audio
Driver host: PulseAudio
Controller: ICH AC97

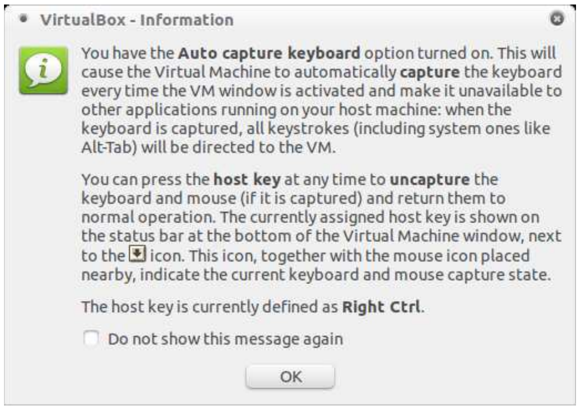
Rete
Scheda 1: Intel PRO/1000 MT Desktop (NAT)

USB
Filtri dispositivi: 0 (0 attivo)

Cartelle
Nessuna

Descrizione
Nessuna

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



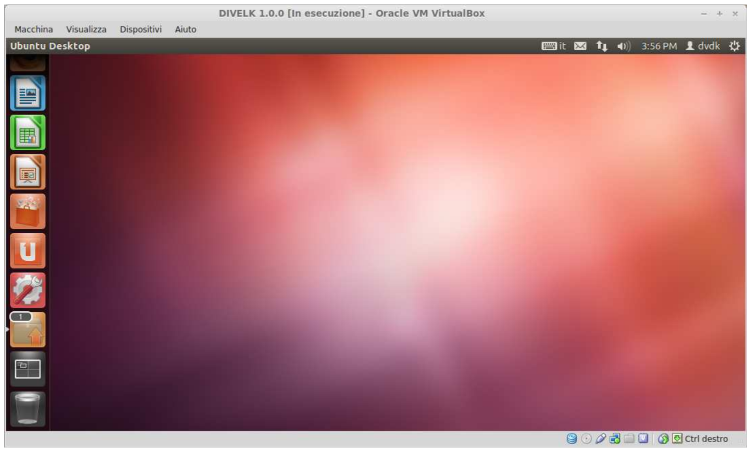
You have the **Auto capture keyboard** option turned on. This will cause the Virtual Machine to automatically **capture** the keyboard every time the VM window is activated and make it unavailable to other applications running on your host machine: when the keyboard is captured, all keystrokes (including system ones like Alt-Tab) will be directed to the VM.

You can press the **host key** at any time to **uncapture** the keyboard and mouse (if it is captured) and return them to normal operation. The currently assigned host key is shown on the status bar at the bottom of the Virtual Machine window, next to the icon. This icon, together with the mouse icon placed nearby, indicate the current keyboard and mouse capture state.

The host key is currently defined as **Right Ctrl**.

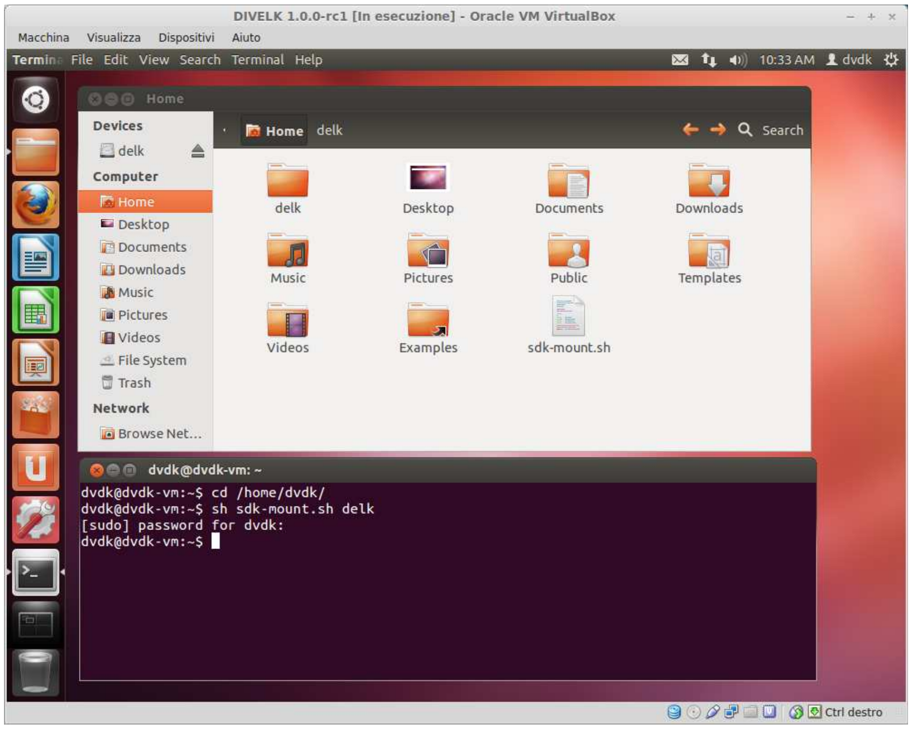
☐ Do not show this message again

3. At the end of the boot process, the Ubunu 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 '''dvdk''' 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 delk
```


DIVELK 1.0.0-rc1 [In esecuzione] - Oracle VM VirtualBox
Macchina Visualizza Dispositivi Aiuto
Terminal File Edit View Search Terminal Help
10:33 AM dvdk
Home
Devices
delk
Computer
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
File System
Trash
Network
Browse Net...
Home delk Search
delk Desktop Documents Downloads
Music Pictures Public Templates
Videos Examples sdk-mount.sh
dvdk@dvdk-vm: ~
Ctrl destro

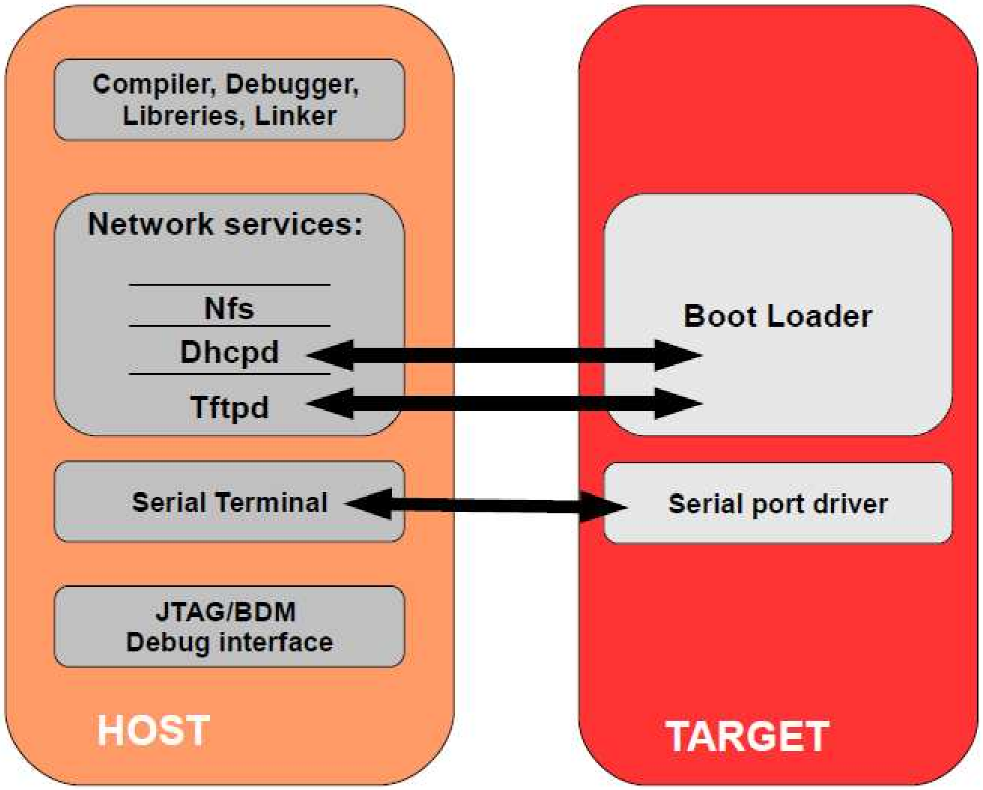
```
dvdk@dvdk-vm:~$ cd /home/dvdk/
dvdk@dvdk-vm:~$ sh sdk-mount.sh delk
[sudo] password for dvdk:
dvdk@dvdk-vm:~$
```

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 menù.
7. Configure the Virtual Machine network interface, as described in this page: http://wiki.dave.eu/index.php/VirtualBox_Network_Configuration

4 Developing Environment

4.1 Introduction

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



The typical developing environment for an Embedded Linux system 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 Diva 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 Sitara processors is maintained primarily by Texas Instruments, that constantly works in close cooperation with Linux community in order to push all the released drivers into mainstream kernel. Periodically TI releases the so-called Platform Support Product (PSP for short). PSP provides updated kernel sources.

Kernels released within DIVELK derive directly from PSP kernels.

For further information on Linux for TI processors, please refer to http://processors.wiki.ti.com/index.php/Category:Linux

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.

4.3 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:

- Arago (http://arago-project.org/wiki/index.php/Main_Page)
- OpenEmbedded (http://wiki.openembedded.net/index.php/Main_Page)
- Yocto (https://www.yoctoproject.org/)
- Buildroot (http://buildroot.uclibc.org)

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 DIVELK.

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 DIVELK.

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.

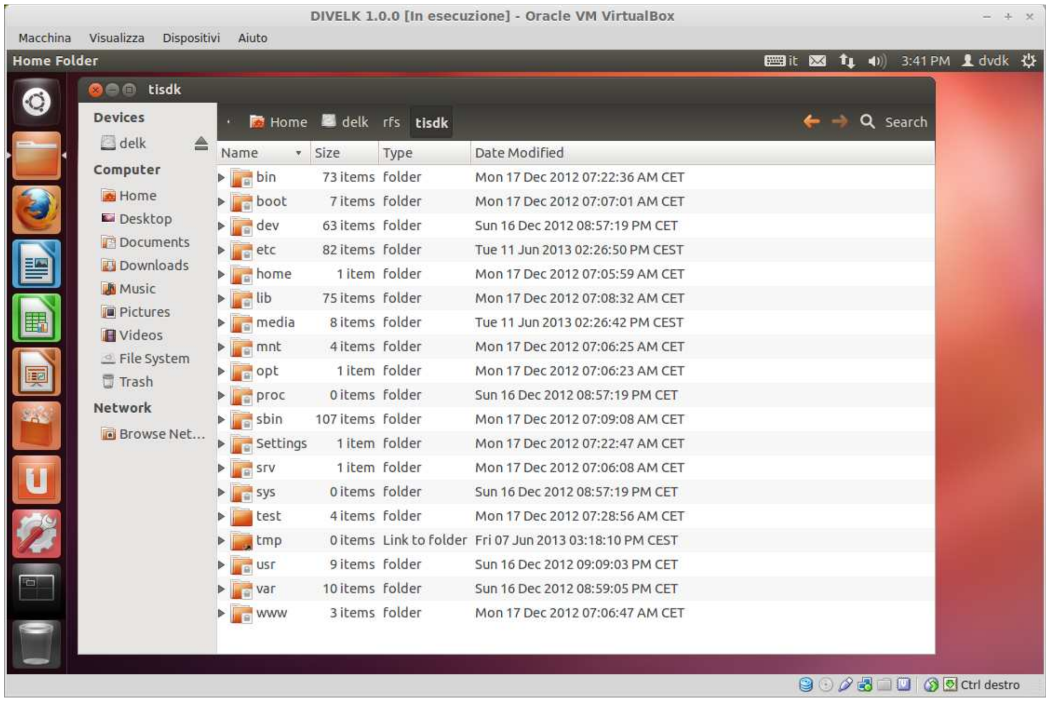
4.3.2.3 Pre-built toolchain

To start developing software for the Diva 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.

DIVELK provides the arago-2012.10 toolchain (**GCC version is 4.5.3**).

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:



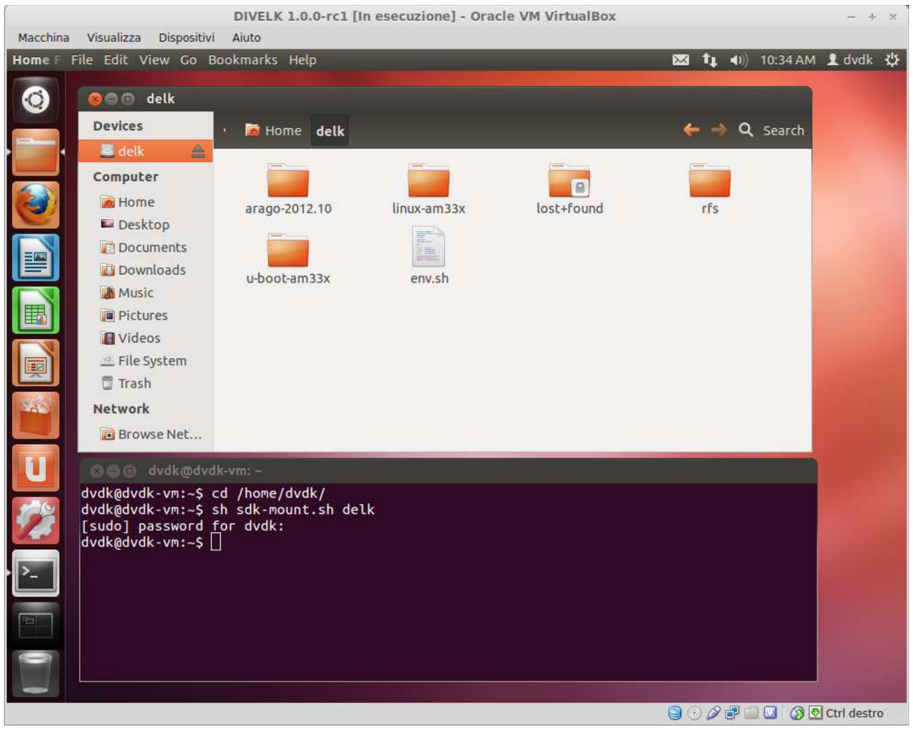
Name	Size	Type	Date Modified
bin	73 items	folder	Mon 17 Dec 2012 07:22:36 AM CET
boot	7 items	folder	Mon 17 Dec 2012 07:07:01 AM CET
dev	63 items	folder	Sun 16 Dec 2012 08:57:19 PM CET
etc	82 items	folder	Tue 11 Jun 2013 02:26:50 PM CEST
home	1 item	folder	Mon 17 Dec 2012 07:05:59 AM CET
lib	75 items	folder	Mon 17 Dec 2012 07:08:32 AM CET
media	8 items	folder	Tue 11 Jun 2013 02:26:42 PM CEST
mnt	4 items	folder	Mon 17 Dec 2012 07:06:25 AM CET
opt	1 item	folder	Mon 17 Dec 2012 07:06:23 AM CET
proc	0 items	folder	Sun 16 Dec 2012 08:57:19 PM CET
sbin	107 items	folder	Mon 17 Dec 2012 07:09:08 AM CET
Settings	1 item	folder	Mon 17 Dec 2012 07:22:47 AM CET
srv	1 item	folder	Mon 17 Dec 2012 07:06:08 AM CET
sys	0 items	folder	Sun 16 Dec 2012 08:57:19 PM CET
test	4 items	folder	Mon 17 Dec 2012 07:28:56 AM CET
tmp	0 items	Link to folder	Fri 07 Jun 2013 03:18:10 PM CEST
usr	9 items	folder	Sun 16 Dec 2012 09:09:03 PM CET
var	10 items	folder	Sun 16 Dec 2012 08:59:05 PM CET
www	3 items	folder	Mon 17 Dec 2012 07:06:47 AM CET

For more information on the Linux filesystem, please refer to The Linux filesystem explained

DIVELK provides a pre-built root file system, that can be used during the evaluation/development phase, since it provides a rich set of packages for working with the Diva platform.

4.4 Working with DIVELK

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



```
dvdk@dvdk-vm:~$ cd /home/dvdk/
dvdk@dvdk-vm:~$ sh sdk-mount.sh delk
[sudo] password for dvdk:
dvdk@dvdk-vm:~$ 
```

The delk directory contains the following subdirectories:

- `arago-2012.10`: the cross-toolchain. GNU Compiler Collection (GCC) version is 4.5.3
- `linux-am33x`: the Linux source tree
- `u-boot-am33x`: the U-Boot source tree
- `rfs`: DIVELK provides two root file systems:
  - `arago-base`: minimal root file system with basic packages (/home/dvdk/delk/rfs/arago-base)
  - `tisdk`: full root file system with lots of packages, useful during the development phase (/home/dvdk/delk/rfs/tisdk)

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

```
export PATH=~/delk/arago-2012.10/bin:$PATH
export ARCH=arm
export CROSS_COMPILE=arm-arago-linux-gnueabi-
```

4.4.1 Build/configure U-Boot

Assuming that you've configured the environment variables sourcing the `env.sh` script, enter the U-Boot sources directory (`~/diva/delk/u-boot-am33x.git`) and run the following commands:

```
dvdk@dvdk-vm:~/diva/delk/u-boot-am33x.git$ make
diva_spiboot
```

```
dvdk@dvdk-vm:~/diva/delk/u-boot-am33x.git$ make
```

The former command selects the default Diva 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.

U-boot is composed by two binary images called:

- MLO (which is called also SPL or 1st stage)
- u-boot.img (which is the main u-boot image, plus an additional header needed by SPL)

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

```
dvdk@dvdk-vm:~/diva/delk/linux-am33x.git$ sudo cp MLO
u-boot.img /srv/tftp/divelk/
```

4.4.2 Build/configure Linux kernel

Assuming that you've configured the environment variables sourcing the `env.sh` script, enter the Linux sources directory (`~/diva/delk/linux-am33x.git`) and run the following commands:

```
dvdk@dvdk-vm:~/diva/delk/linux-am33x.git$ make
diva_defconfig
```

```
dvdk@dvdk-vm:~/diva/delk/linux-am33x.git$ make uImage
```

The former command selects the default Diva 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 targets.

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

Once the build process is completed, the kernel binary image is stored into the `linux-am33x.git/arch/arm/boot/uImage` file. This file can be copied to the `/srv/tftp/divelk/` directory with the following command:

```
dvdk@dvdk-vm:~/diva/delk/linux-am33x.git$ sudo cp
arch/arm/boot/uImage /srv/tftp/divelk/
```

4.4.3 Build a custom application

This section will be completed in a future release of this manual.

5 Frequently Asked Questions

5.1 Q: Where can I found Diva SOM information?

A: please refer to the following table:

Document	Location
Diva main page on Dave Developers Wiki	http://wiki.dave.eu/index.php/Category:Diva
Diva Hardware Manual	http://www.dave.eu/sites/default/files/files/diva-hm.pdf
Diva Software Manual	http://wiki.dave.eu/index.php/Software_Manual_(Diva)
Diva product page	http://www.dave.eu/dave-cpu-module-am335x-diva.html

5.2 Q: I've received the DIVELK 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 DIVELK version?

A: please refer to the following page on the Dave Developer's Wiki: http://wiki.dave.eu/index.php/Software_Manual_%28Diva%29#DIVELK_Updates

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

A: please refer to the following page: http://processors.wiki.ti.com/index.php/AM335x_PSP_User%27s_Guide#Various_Module_User.27s_Guide

5.5 How can I configure the Diva 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

Parameter	Description	Default
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/uImage
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: Where can I found information regarding the PRUs?

A: the Programmable Real-Time Unit and Industrial Communication Subsystem (PRU-ICSS) consists of dual 32-bit RISC cores (Programmable Real-Time Units, or PRUs), memories, interrupt controller, and internal peripherals that enable additional peripheral interfaces and protocols. The programmable nature of the PRUs, along with their access to pins and events, provide flexibility in implementing custom peripheral interfaces, fast real-time responses, power saving techniques, specialized data handling and DMA operations, and in offloading tasks from the other processor cores of the system-on-chip (SoC). For detailed information, please refer to the following pages:

- http://elinux.org/Ti_AM33XX_PRUSSv2
- http://processors.wiki.ti.com/index.php/Category:PRU
- http://processors.wiki.ti.com/index.php/Programmable_Realtime_Unit
- http://processors.wiki.ti.com/index.php/Programmable_Realtim

e_Unit_Subsystem

- http://processors.wiki.ti.com/index.php/Programmable_Realtime_Unit_Software_Development

5.7 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 Diva, please refer to the Diva Integration Guide (http://wiki.dave.eu/index.php/Integration_guide_%28Diva%29)

6 Appendices

6.1 A: boot messages

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 SPL 2012.10 (May 24 2013 - 18:29:02)
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img


U-Boot 2012.10 (May 24 2013 - 18:29:02) [divelk-1.0.0]

I2C:   ready
DRAM:  512 MiB
WARNING: Caches not enabled
Now running in RAM - U-Boot at: 9ff47000
NAND:  1024 MiB
MMC:   OMAP SD/MMC: 0
SF: Detected S25FL256S_64K with page size 256 Bytes, total 32 MiB
USB Host mode controller at 47401000 using PIO, IRQ 0
USB Host mode controller at 47401800 using PIO, IRQ 0
Module id#: 0x1
Net:   cpsw connected to SMSC LAN8710/LAN8720
cpsw, usb_ether
Hit any key to stop autoboot:  0
Booting from mmc ...
reading uImage

3027568 bytes read
## Booting kernel from Legacy Image at 80007fc0 ...
   Image Name:   Linux-3.2.0-divelk-1.0.0
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3027504 Bytes = 2.9 MiB
   Load Address: 80008000
   Entry Point:  80008000
   Verifying Checksum ... OK
   XIP Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
[    0.000000] Linux version 3.2.0-divelk-1.0.0 (amon@linuxserver2) (gcc version 4.5.3
20110311 (prerelease) (GCC) ) #50 Wed May 22 15:27:18 CEST 2013
[    0.000000] CPU: ARMv7 Processor [413fc082] revision 2 (ARMv7), cr=10c53c7d
[    0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
[    0.000000] Machine: diva
[    0.000000] Memory policy: ECC disabled, Data cache writeback
[    0.000000] AM335X ES1.0 (sgx neon )
[    0.000000] Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 130048
[    0.000000] Kernel command line: console=ttyO0,115200n8 root=/dev/mmcblk0p2 ro
rootfstype=ext3 rootwait ip=none
[    0.000000] PID hash table entries: 2048 (order: 1, 8192 bytes)
[    0.000000] Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
[    0.000000] Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
[    0.000000] Memory: 512MB = 512MB total
[    0.000000] Memory: 513436k/513436k available, 10852k reserved, 0K highmem
[    0.000000] Virtual kernel memory layout:
```

```
[    0.000000]     vector  : 0xffff0000 - 0xffff1000   (   4 kB)
[    0.000000]     fixmap  : 0xfff00000 - 0xfffe0000   ( 896 kB)
[    0.000000]     vmalloc : 0xe0800000 - 0xff000000   ( 488 MB)
[    0.000000]     lowmem  : 0xc0000000 - 0xe0000000   ( 512 MB)
[    0.000000]     modules : 0xbf000000 - 0xc0000000   (  16 MB)
[    0.000000]       .text : 0xc0008000 - 0xc0569000   (5508 kB)
[    0.000000]       .init : 0xc0569000 - 0xc05a1000   ( 224 kB)
[    0.000000]       .data : 0xc05a2000 - 0xc05fbe28   ( 360 kB)
[    0.000000]        .bss : 0xc05fbe4c - 0xc062a930   ( 187 kB)
[    0.000000] NR_IRQS:396
[    0.000000] IRQ: Found an INTC at 0xfa200000 (revision 5.0) with 128 interrupts
[    0.000000] Total of 128 interrupts on 1 active controller
[    0.000000] OMAP clockevent source: GPTIMER2 at 24000000 Hz
[    0.000000] OMAP clocksource: GPTIMER1 at 32768 Hz
[    0.000000] sched_clock: 32 bits at 32kHz, resolution 30517ns, wraps every 131071999ms
[    0.000000] Console: colour dummy device 80x30
[    0.000152] Calibrating delay loop... 718.02 BogoMIPS (lpj=3590144)
[    0.058929] pid_max: default: 32768 minimum: 301
[    0.059051] Security Framework initialized
[    0.059173] Mount-cache hash table entries: 512
[    0.059539] CPU: Testing write buffer coherency: ok
[    0.079925] omap_hwmod: pruss: failed to hardreset
[    0.081085] print_constraints: dummy:
[    0.081420] NET: Registered protocol family 16
[    0.083526] OMAP GPIO hardware version 0.1
[    0.086090] omap_mux_init: Add partition: #1: core, flags: 0
[    0.087951]  omap_i2c.1: alias fck already exists
[    0.088653]  omap_hsmmc.0: alias fck already exists
[    0.089080] Try to build omap device: ti_tscadc - adc_tsc
[    0.089477] Try to build omap device: davinci-mcasp - mcasp0
[    0.091522]  da8xx_lcdc.0: alias fck already exists
[    0.092010]  omap2_mcspi.1: alias fck already exists
[    0.092254]  omap2_mcspi.2: alias fck already exists
[    0.092498]  edma.0: alias fck already exists
[    0.092529]  edma.0: alias fck already exists
[    0.092529]  edma.0: alias fck already exists
[    0.115631] bio: create slab <bio-0> at 0
[    0.117584] SCSI subsystem initialized
[    0.119689] usbcore: registered new interface driver usbfs
[    0.119995] usbcore: registered new interface driver hub
[    0.120178] usbcore: registered new device driver usb
[    0.120483] registerd cppi-dma Intr @ IRQ 17
[    0.120513] Cppi41 Init Done Qmgr-base(e087a000) dma-base(e0878000)
[    0.120513] Cppi41 Init Done
[    0.120544] musb-ti81xx musb-ti81xx: musb0, board_mode=0x11, plat_mode=0x1
[    0.120849] musb-ti81xx musb-ti81xx: musb1, board_mode=0x11, plat_mode=0x1
[    0.138977] omap_i2c omap_i2c.1: bus 1 rev2.4.0 at 100 kHz
[    0.140167] tps65910 1-002d: JTAGREVNUM 0x1
[    0.142639] print_constraints: VRTC:
[    0.144104] print_constraints: VIO: at 1500 mV
[    0.146453] print_constraints: VDD1: 600 <--> 1500 mV at 1262 mV normal
[    0.148773] print_constraints: VDD2: 600 <--> 1500 mV at 1137 mV normal
[    0.149841] print_constraints: VDD3: 5000 mV
[    0.151275] print_constraints: VDIG1: at 1800 mV
[    0.152740] print_constraints: VDIG2: at 1800 mV
[    0.154174] print_constraints: VPLL: at 1800 mV
[    0.155609] print_constraints: VDAC: at 1800 mV
[    0.157073] print_constraints: VAUX1: at 1800 mV
[    0.158538] print_constraints: VAUX2: at 3300 mV
[    0.159973] print_constraints: VAUX33: at 3300 mV
[    0.161407] print_constraints: VMMC: at 3300 mV
[    0.161926] tps65910 1-002d: No interrupt support, no core IRQ
[    0.163208] Advanced Linux Sound Architecture Driver Version 1.0.24.
[    0.164001] Switching to clocksource gp timer
[    0.179504] musb-hdrc: version 6.0, ?dma?, otg (peripheral+host)
[    0.179718] musb-hdrc musb-hdrc.0: dma type: dma-cppi41
[    0.180023] MUSB0 controller's USBSS revision = 4ea20800
[    0.180511] musb-hdrc musb-hdrc.0: MUSB HDRC host driver
[    0.180633] musb-hdrc musb-hdrc.0: new USB bus registered, assigned bus number 1
[    0.180755] usb usb1: New USB device found, idVendor=1d6b, idProduct=0002
[    0.180786] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[    0.180786] usb usb1: Product: MUSB HDRC host driver
[    0.180816] usb usb1: Manufacturer: Linux 3.2.0-divelk-1.0.0 musb-hcd
[    0.180816] usb usb1: SerialNumber: musb-hdrc.0
[    0.181610] hub 1-0:1.0: USB hub found
[    0.181640] hub 1-0:1.0: 1 port detected
[    0.182189] musb-hdrc musb-hdrc.0: USB Host mode controller at e083c000 using DMA, IRQ 18
[    0.182373] musb-hdrc musb-hdrc.1: dma type: dma-cppi41
[    0.182708] MUSB1 controller's USBSS revision = 4ea20800
[    0.183166] musb-hdrc musb-hdrc.1: MUSB HDRC host driver
[    0.183227] musb-hdrc musb-hdrc.1: new USB bus registered, assigned bus number 2
[    0.183319] usb usb2: New USB device found, idVendor=1d6b, idProduct=0002
[    0.183349] usb usb2: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[    0.183349] usb usb2: Product: MUSB HDRC host driver
[    0.183380] usb usb2: Manufacturer: Linux 3.2.0-divelk-1.0.0 musb-hcd
[    0.183380] usb usb2: SerialNumber: musb-hdrc.1
[    0.184234] hub 2-0:1.0: USB hub found
[    0.184265] hub 2-0:1.0: 1 port detected
[    0.184783] musb-hdrc musb-hdrc.1: USB Host mode controller at e083e800 using DMA, IRQ 19
[    0.185211] NET: Registered protocol family 2
[    0.185424] IP route cache hash table entries: 4096 (order: 2, 16384 bytes)
[    0.185729] TCP established hash table entries: 16384 (order: 5, 131072 bytes)
[    0.186004] TCP bind hash table entries: 16384 (order: 4, 65536 bytes)
[    0.186187] TCP: Hash tables configured (established 16384 bind 16384)
[    0.186218] TCP reno registered
[    0.186218] UDP hash table entries: 256 (order: 0, 4096 bytes)
[    0.186248] UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
[    0.186401] NET: Registered protocol family 1
[    0.186676] RPC: Registered named UNIX socket transport module.
[    0.186706] RPC: Registered udp transport module.
[    0.186706] RPC: Registered tcp transport module.
[    0.186706] RPC: Registered tcp NFSv4.1 backchannel transport module.
[    0.186920] NetWinder Floating Point Emulator V0.97 (double precision)
[    0.187164] omap-gpmc omap-gpmc: GPMC revision 6.0
[    0.187164] Registering NAND on CS0
[    0.187194] GPMC CS0: cs_on     :   0 ticks,   0 ns (was   0 ticks)   0 ns
[    0.187225] GPMC CS0: cs_rd_off :   5 ticks,  50 ns (was  30 ticks)  50 ns
[    0.187225] GPMC CS0: cs_wr_off :   5 ticks,  50 ns (was  30 ticks)  50 ns
[    0.187255] GPMC CS0: adv_on    :   1 ticks,  10 ns (was   0 ticks)  10 ns
[    0.187255] GPMC CS0: adv_rd_off:   4 ticks,  40 ns (was  30 ticks)  40 ns
[    0.187286] GPMC CS0: adv_wr_off:   5 ticks,  50 ns (was  30 ticks)  50 ns
[    0.187286] GPMC CS0: oe_on     :   1 ticks,  10 ns (was   7 ticks)  10 ns
[    0.187316] GPMC CS0: oe_off    :   6 ticks,  60 ns (was  24 ticks)  60 ns
[    0.187316] GPMC CS0: we_on     :   1 ticks,  10 ns (was   5 ticks)  10 ns
[    0.187347] GPMC CS0: we_off    :   4 ticks,  40 ns (was  22 ticks)  40 ns
[    0.187347] GPMC CS0: rd_cycle  :   9 ticks,  90 ns (was  30 ticks)  90 ns
[    0.187377] GPMC CS0: wr_cycle  :   9 ticks,  90 ns (was  30 ticks)  90 ns
[    0.187377] GPMC CS0: access    :   7 ticks,  70 ns (was  21 ticks)  70 ns
[    0.187408] GPMC CS0: page_burst_access:   0 ticks,   0 ns (was   0 ticks)   0 ns
[    0.187408] GPMC CS0: wr_data_mux_bus:   0 ticks,   0 ns (was   0 ticks)   0 ns
[    0.187438] GPMC CS0: wr_access :   4 ticks,  40 ns (was  22 ticks)  40 ns
[    0.187438] GPMC CS0 CLK period is 10 ns (div 1)
[    0.207489] VFS: Disk quotas dquot_6.5.2
[    0.207550] Dquot-cache hash table entries: 1024 (order 0, 4096 bytes)
[    0.208099] msgmni has been set to 1002
[    0.208862] io scheduler noop registered
[    0.208892] io scheduler deadline registered
[    0.208953] io scheduler cfq registered (default)
[    0.210296] da8xx_lcdc da8xx_lcdc.0: GLCD: Found AM800480STMQW_TA1 panel
[    0.227233] Console: switching to colour frame buffer device 100x30
[    0.236999] omap_uart.0: ttyO0 at MMIO 0x44e09000 (irq = 72) is a OMAP UART0
[    1.082336] console [ttyO0] enabled
[    1.086791] omap_uart.1: ttyO1 at MMIO 0x48022000 (irq = 73) is a OMAP UART1
[    1.094604] omap_uart.2: ttyO2 at MMIO 0x48024000 (irq = 74) is a OMAP UART2
[    1.102416] omap_uart.3: ttyO3 at MMIO 0x481a6000 (irq = 44) is a OMAP UART3
[    1.110198] omap_uart.4: ttyO4 at MMIO 0x481a8000 (irq = 45) is a OMAP UART4
[    1.117980] omap_uart.5: ttyO5 at MMIO 0x481aa000 (irq = 46) is a OMAP UART5
[    1.135192] brd: module loaded
[    1.143188] loop: module loaded
[    1.146606] at24 1-0050: 32768 byte 24c32 EEPROM, writable, 64 bytes/write
[    1.156250] mtdoops: mtd device (mtddev=name/number) must be supplied
[    1.163177] m25p80 spi1.0: found s25fl256s1, expected m25p80
[    1.169158] m25p80 spi1.0: s25fl256s1 (32768 Kbytes)
[    1.174591] Creating 6 MTD partitions on "spi_flash":
[    1.179901] 0x000000000000-0x000000040000 : "SPI U-Boot 1st (SPL)"
[    1.187744] 0x000000040000-0x0000000c0000 : "SPI U-Boot 2nd"
[    1.194946] 0x0000000c0000-0x000000100000 : "SPI U-Boot env1"
[    1.202209] 0x000000100000-0x000000140000 : "SPI U-Boot env2"
[    1.209411] 0x000000140000-0x000000540000 : "SPI Linux Kernel"
[    1.216705] 0x000000540000-0x000002000000 : "SPI Free Space"
[    1.224334] omap2-nand driver initializing
[    1.229064] NAND device: Manufacturer ID: 0xec, Chip ID: 0xd3 (Samsung NAND 1GiB 3,3V
8-bit)
[    1.238098] Creating 9 MTD partitions on "omap2-nand.0":
[    1.243652] 0x000000000000-0x000000020000 : "NAND U-Boot 1st (SPL)"
[    1.251525] 0x000000020000-0x000000040000 : "NAND U-Boot 1st (SPL) - backup1"
[    1.260314] 0x000000040000-0x000000060000 : "NAND U-Boot 1st (SPL) - backup2"
[    1.269073] 0x000000060000-0x000000080000 : "NAND U-Boot 1st (SPL) - backup3"
[    1.277954] 0x000000080000-0x000000260000 : "NAND U-Boot"
[    1.285675] 0x000000260000-0x000000280000 : "NAND U-Boot env #1"
[    1.293151] 0x000000280000-0x0000002a0000 : "NAND U-Boot env #2"
[    1.300689] 0x0000002a0000-0x0000007a0000 : "NAND Kernel"
[    1.309631] 0x0000007a0000-0x000040000000 : "NAND File System"
[    1.738342] OneNAND driver initializing
[    1.744110] tun: Universal TUN/TAP device driver, 1.6
[    1.749420] tun: (C) 1999-2004 Max Krasnyansky <maxk@qualcomm.com>
[    1.756103] CAN device driver interface
[    1.760101] CAN bus driver for Bosch D_CAN controller 1.0
[    1.814697] davinci_mdio davinci_mdio.0: davinci mdio revision 1.6
[    1.821136] davinci_mdio davinci_mdio.0: detected phy mask ffffffbf
[    1.828369] davinci_mdio.0: probed
[    1.831939] davinci_mdio davinci_mdio.0: phy[6]: device 0:06, driver SMSC LAN8710/LAN8720
[    1.840850] usbcore: registered new interface driver cdc_ether
[    1.847106] usbcore: registered new interface driver cdc_eem
[    1.853149] usbcore: registered new interface driver dm9601
[    1.859039] cdc_ncm: 04-Aug-2011
[    1.862548] usbcore: registered new interface driver cdc_ncm
[    1.868469] Initializing USB Mass Storage driver...
[    1.873779] usbcore: registered new interface driver usb-storage
[    1.880065] USB Mass Storage support registered.
[    1.885345] mousedev: PS/2 mouse device common for all mice
[    1.892120] input: ti-tsc as /devices/platform/omap/ti_tscadc/tsc/input/input0
[    1.909027] tps65910-rtc tps65910-rtc: rtc core: registered tps65910-rtc as rtc0
[    1.916961] i2c /dev entries driver
[    1.920959] Linux video capture interface: v2.00
[    1.926147] usbcore: registered new interface driver uvcvideo
[    1.932128] USB Video Class driver (1.1.1)
[    1.936431] Driver for 1-wire Dallas network protocol.
[    1.942810] INA Probe
[    1.946289] ina2xx 1-0041: power monitor INA226 (Rshunt = 10000 uOhm)
[    1.955566] cpuidle: using governor ladder
[    1.960357] cpuidle: using governor menu
[    1.983306] usbcore: registered new interface driver usbhid
[    1.989135] usbhid: USB HID core driver
[    1.995117] Audio: davinci-evm.c - evm_init
[    1.999847] UDA134X SoC Audio Codec
[    2.015502] asoc: uda134x-hifi <-> davinci-mcasp.0 mapping ok
[    2.025756] ALSA device list:
[    2.028900]   #0: Diva Dacu
[    2.031799] oprofile: hardware counters not available
[    2.037078] oprofile: using timer interrupt.
[    2.041534] nf_conntrack version 0.5.0 (8022 buckets, 32088 max)
[    2.048370] ip_tables: (C) 2000-2006 Netfilter Core Team
[    2.054016] TCP cubic registered
[    2.057434] NET: Registered protocol family 17
[    2.062072] can: controller area network core (rev 20090105 abi 8)
[    2.068634] NET: Registered protocol family 29
[    2.073272] can: raw protocol (rev 20090105)
[    2.077728] can: broadcast manager protocol (rev 20090105 t)
[    2.083679] Registering the dns_resolver key type
[    2.088684] VFP support v0.3: implementor 41 architecture 3 part 30 variant c rev 3
[    2.096679] ThumbEE CPU extension supported.
[    2.101196] mux: Failed to setup hwmod io irq -22
[    2.106811] Power Management for AM33XX family
[    2.111663] Trying to load am335x-pm-firmware.bin (60 secs timeout)
[    2.118347] Copied the M3 firmware to UMEM
[    2.122680] Cortex M3 Firmware Version = 0x18
[    2.130737] clock: disabling unused clocks to save power
[    2.138702] Detected MACID=00:18:30:F0:0D:6B
[    2.144134] cpsw: Detected MACID = 00:18:30:f0:0d:6c
[    2.154083] tps65910-rtc tps65910-rtc: setting system clock to 2012-12-17 10:16:31 UTC
(1355739391)
[    2.164123] Waiting for root device /dev/mmcblk0p2...
[    2.195617] mmc0: host does not support reading read-only switch. assuming write-enable.
[    2.206085] mmc0: new high speed SDHC card at address 1234
[    2.212463] mmcblk0: mmc0:1234 SA16G 14.6 GiB
[    2.219696]  mmcblk0: p1 p2
[    2.284118] kjournald starting.  Commit interval 5 seconds
[    2.289947] EXT3-fs (mmcblk0p2): mounted filesystem with ordered data mode
[    2.297149] VFS: Mounted root (ext3 filesystem) readonly on device 179:2.
[    2.304565] Freeing init memory: 224K
INIT: version 2.88 booting
Starting udev
modprobe: FATAL: Could not load /lib/modules/3.2.0-divelk-1.0.0/modules.dep: No such file or
directory

WARNING: -e needs -E or -F
WARNING: Couldn't open directory /lib/modules/3.2.0-divelk-1.0.0: No such file or directory
FATAL: Could not open /lib/modules/3.2.0-divelk-1.0.0/modules.dep.temp for writing: No such
file or directory
Starting Bootlog daemon: bootlogd: cannot allocate pseudo tty: No such file or directory
bootlogd.
[    6.686279] EXT3-fs (mmcblk0p2): using internal journal
ALSA: Restoring mixer settings...
Configuring network interfaces... [    7.202239] net eth0: CPSW phy found : id is : 0x7c0f1
udhcpc (v1.19.4) started
Sending discover...
Sending discover...
Sending discover...
No lease, failing
done.
INIT: Entering runlevel: 5
Starting system message bus: dbus.
Starting Dropbear SSH server: modprobe: FATAL: Could not load
/lib/modules/3.2.0-divelk-1.0.0/modules.dep: No such file or directory

modprobe: FATAL: Could not load /lib/modules/3.2.0-divelk-1.0.0/modules.dep: No such file or
directory

dropbear.
Starting telnet daemon.
Performing wifi calibration...
modprobe: FATAL: Could not load /lib/modules/3.2.0-divelk-1.0.0/modules.dep: No such file or
directory

rm: can't remove '/lib/firmware/ti-connectivity/wl1271-nvs.bin': No such file or directory
ERROR: Module wl12xx_sdio does not exist in /proc/modules
nl80211 not found.
FATAL: Could not load /lib/modules/3.2.0-divelk-1.0.0/modules.dep: No such file or directory
Starting network benchmark server: netserver.
Starting syslogd/klogd: done
Starting thttpd.
[   18.064361] Unhandled fault: external abort on non-linefetch (0x1018) at 0x4013d44c
No SGX hardware, not starting PVR
Starting Lighttpd Web Server: lighttpd.
2012-12-17 10:16:46: (log.c.166) server started
/
Starting Matrix GUI application.
FATAL: Could not load /lib/modules/3.2.0-divelk-1.0.0/modules.dep: No such file or directory
***************************************************************
***************************************************************
NOTICE: This file system contains the followin GPLv3 packages:
        binutils-symlinks
        binutils
        gdbserver

If you do not wish to distribute GPLv3 components please remove
the above packages prior to distribution.  This can be done using
the opkg remove command.  i.e.:
    opkg remove <package>
Where <package> is the name printed in the list above

NOTE: If the package is a dependency of another package you
      will be notified of the dependent packages.  You should
      use the --force-removal-of-dependent-packages option to
      also remove the dependent packages as well
***************************************************************
***************************************************************
Stopping Bootlog daemon: bootlogd.

 _____                    _____           _         _
|  _  |___ ___ ___ ___   |  _  |___ ___  |_|___ ___| |_
|     |  _| .'| . | . |  |   __|  _| . | | | -_|  _|  _|
|__|__|_| |__,|_  |___|  |__|  |_| |___|_| |___|___|_|
              |___|                    |___|

Arago Project http://arago-project.org am335x-evm ttyO0

Arago 2012.10 am335x-evm ttyO0

am335x-evm login:
```