
diVa

ARM Cortex-A8 CPU Module Family

Lite Line

Diva Embedded Linux Kit (*DIVELK*)

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.....	10
2.1	DIVA SOM.....	10
2.2	Embedded Linux.....	11
2.3	DIVELK.....	12
2.3.1	Kit Contents.....	14
2.3.2	DIVELK Release Notes.....	15
2.3.2.1	Version 2.1.0.....	15
2.3.2.2	Version 2.0.0.....	15
2.3.2.3	Version 1.0.0.....	16
2.3.2.4	Releases history.....	16
3	DIVELK Quick Start.....	17
3.1	Unboxing.....	17
3.2	Hardware setup.....	18
3.3	First boot.....	18
3.4	DVDK installation.....	20
3.4.1	DVDK features.....	20
3.4.2	MicroSD contents.....	21
3.4.3	Importing the virtual machine.....	21
3.4.4	Launching the virtual machine.....	23
4	Developing Environment.....	26
4.1	Introduction.....	26
4.2	Software components.....	27
4.2.1	Toolchain.....	27
4.2.2	Bootloader.....	27
4.2.3	Kernel.....	28
4.2.3.1	Linux Device Tree.....	28
4.2.4	Target root file system.....	29
4.2.5	Yocto.....	29
4.3	Working with DIVELK.....	30
4.3.1	The build system.....	30
4.3.2	Overview of the installed components.....	30
4.3.3	Setting up the server environment.....	32

4.3.3.1	TFTP Server.....	32
4.3.3.2	NFS Server.....	33
4.3.3.3	Pre-built toolchain.....	33
4.3.3.4	Pre-built root file system.....	34
4.4	Building the software components with Yocto.....	35
4.4.1	Prerequisites.....	35
4.4.2	Initializing the build environment.....	36
4.4.3	Build the Yocto image.....	36
4.5	Building the software components outside Yocto.....	37
4.5.1	Build/configure U-Boot.....	37
4.5.2	Build/configure Linux kernel.....	38
4.5.3	Build a custom application.....	39
5	DIVELK use cases.....	40
5.1	DIVA Frequency Scaling.....	40
5.2	Forcing the kernel to use a limited amount of RAM.....	41
6	Frequently Asked Questions.....	42
6.1	Q: Where can I found DIVA SOM information?.....	42
6.2	Q: I've received the DIVELK package. How am I supposed to start working with it?.....	42
6.3	Q: How can I update the DIVELK version?.....	43
6.4	Q: How can I work with the XYZ peripheral/interface?.....	43
6.5	How can I configure the DIVA system to boot from network?.....	43
6.6	Q: Where can I found information regarding the PRUs?.....	44
6.7	Q: Can you suggest some guidelines for the carrier board design?.....	45
6.8	Q: The system doesn't boot anymore. On the serial console I see just "CCCC" characters... Is there a recovery procedure?.....	45
7	Appendices.....	46
7.1	A: boot messages.....	46

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 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 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 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

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 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 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 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-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 Embedded Systems Developers Wiki	http://wiki.dave.eu/index.php/Main_Page
AM335x Technical Reference Manual	http://www.ti.com/lit/ug/spruh73k/spruh73k.pdf
DIVA main page on DAVE Embedded Systems 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 Embedded Systems 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

Document	Location
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
1.0.2	January 2014	Minor fixes
1.0.3	October 2014	Updated for DIVELK 2.0.0 release
1.0.4	November 2014	Minor fixes
1.0.5	June 2015	Updated for DIVELK 2.1.0 release
1.0.6	July 2015	Minor fixes

2 Introduction

2.1 DIVA SOM

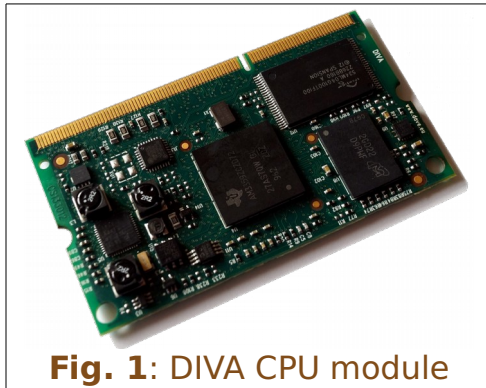


Fig. 1: DIVA CPU module

DIVA is a family of system-on-modules (SOM) that belongs to **DAVE Embedded Systems 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

For further information, please refer to DIVA Hardware Manual.

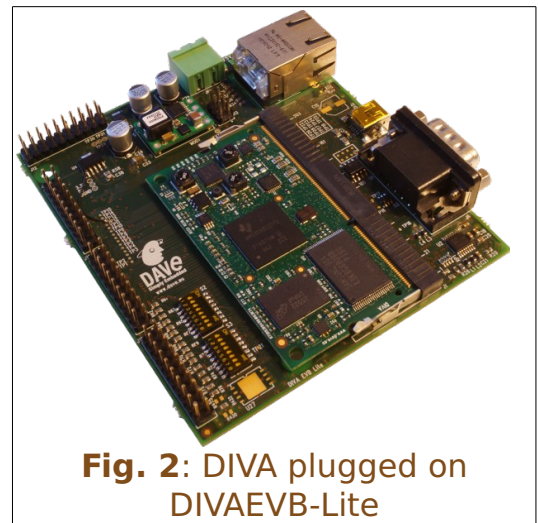


Fig. 2: DIVA plugged on DIVAEVB-Lite

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 machine. The host is used by the developer to compile the code that will

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.

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.




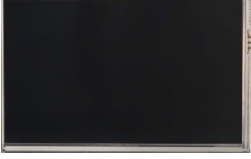



DAVE Embedded Systems adds to the latest EZSDK from Texas Instruments the customization required to support the DIVA platform. For this reason most of the documentation provided by TI remains valid for the DIVELK development kit. However, some customization is required, in particular at bootloader and linux kernel levels.

DIVELK 2.0.0 introduces support for the **Yocto** build system,

an open source collaboration project that provides templates, tools and methods to help creating custom Linux-based systems for embedded products. It is derived from OpenEmbedded, but it provides a less steep learning curve, a graphical interface for Bitbake and very good documentation. **DAVE Embedded Systems** provides the so-called recipes/meta-repositories required to build all the DIVELK software components (bootloader, kernel and root file system) with the **Yocto** build system. For further details, please refer to <https://wiki.yoctoproject.org/wiki/FAQ>.

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
	DIVAEVB-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
	MicroSDHC card with SD adapter and USB adapter

2.3.2 DIVELK Release Notes

2.3.2.1 Version 2.1.0

Notes

- Maintenance release
- Updated kernel and u-boot versions
- Bug fixes

Known limitations:

Issue #	Component	Subsystem	Description
0001	Linux	USB	USB0, when configured as OTG, requires loading of the gadget module for detecting devices (and entering host mode)
0002	Yocto	ADT	The Yocto ADT plugin for Eclipse doesn't work
0003	Linux	Audio	Codec on the DACU board doesn't work out-of-the-box. A software patch is required, please contact support-diva@dave.eu for information on how to enable the audio codec.

2.3.2.2 Version 2.0.0

Notes

- First major update
- Added Yocto support
- Updated kernel and u-boot versions

Known limitations:

Issue #	Component	Subsystem	Description
0001	Linux	USB	USB0, when configured as OTG, requires loading of the gadget module for detecting devices (and entering host mode)

Issue #	Component	Subsystem	Description
0002	Yocto	ADT	The Yocto ADT plugin for Eclipse doesn't work

2.3.2.3 Version 1.0.0

First official release

2.3.2.4 Releases history

	DIVELK Version		
Release number	1.0.0	2.0.0	2.1.0
Status	Released	Released	Released
Release date	June 2013	October 2014	June 2015
Release notes	Version 1.0.0	Version 2.0.0	Version 2.1.0
SOM PCB version	CS133012A	CS133012A	CS133012A
Supported carrier boards	DIVAEVB-Lite DACU	DIVAEVB-Lite DACU	DIVAEVB-Lite DACU
U-Boot version	2012.10-divelk-1.0.0	2013.10-divelk-2.0.0	2013.10-divelk-2.0.0
Linux version	3.2.0-divelk-1.0.0	3.12.10-divelk-2.0.0	3.12.10-divelk-2.0.0
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)	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) USB OTG CAN ADC	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) USB OTG CAN ADC Splash screen in u-boot ConfigID
TI EZSDK	05.06.00.00	07.00.00.00	07.00.00.00
Build Sytem	-	Yocto Dylan (1.4)	Yocto Dylan (1.4)
Graphic libraries	Qt 4.8.4	Qt 4.8.4 Qt 5.3.2	Qt 4.8.4 Qt 5.3.2

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 DVDK installation

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 and configuring a full development environment from scratch. 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 just adapting the 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 also be used 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 DVDK features

- VirtualBox virtual machine (.OVA archive)
- Based on Lubuntu 12.04 LTS (64-bit version)
- Pre-installed VirtualBox Guest Additions
- LXDE desktop environment available
- Boot disk with pre-installed Lubuntu 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
 - External pre-built toolchain
 - Yocto bsp for DIVA
 - Pre-installed Yocto-based root file systems with setup scripts, makefiles, example applications, ...

- Administrator account (dvdk) with autologin. Please note that the user account credentials are provided with the development kit (you can find them into the README file contained in the sw/dvdk folder of the kit distribution)

3.4.2 MicroSD contents

The microSD provided with DIVELK is used to store:

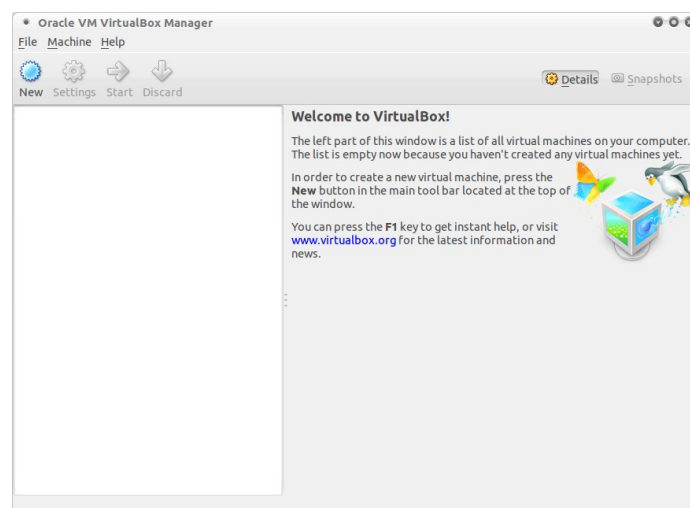
- A bootable partition (**mmcblk0p1, vfat**) containing:
 - binary images (MLO, u-boot and kernel images)
 - DIVELK documentation
 - DVDK virtual machine image (in an .OVA archive)
- DIVELK root file system partition (**mmcblk0p2, ext3**)

DIVELK contains all the required software and documentation to start developing Linux application on the DIVA platform

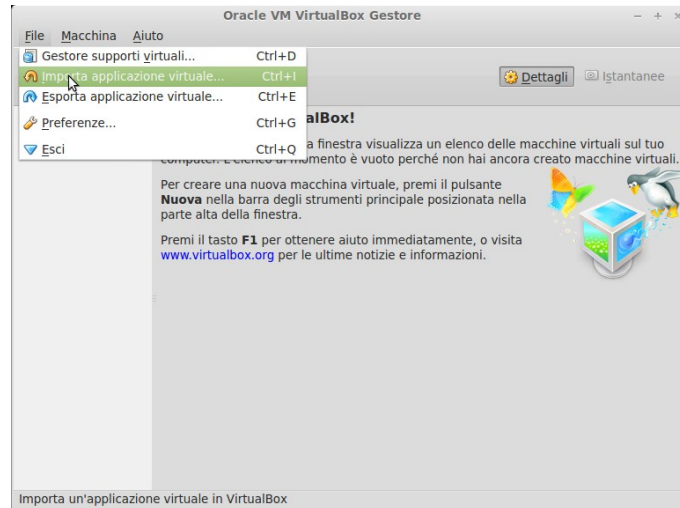
3.4.3 Importing the virtual machine

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



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



3. Navigate your file system to the MicroSD card provided with the DIVEL, move to the `sw/dvdk` directory and select the `.ova` file:

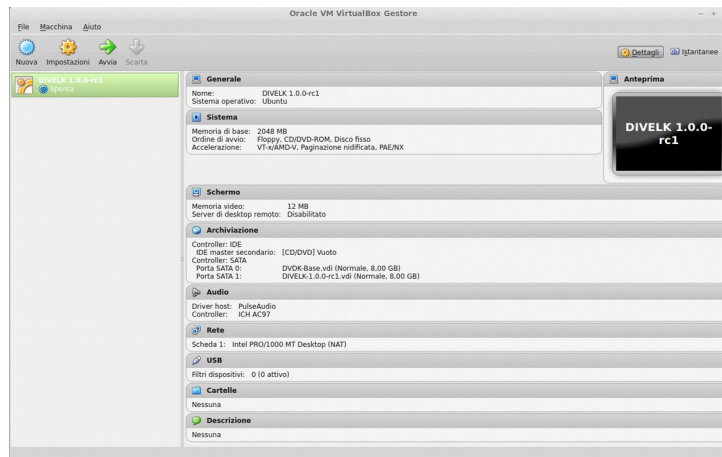


4. Click "Next" and on the next window click on "Import"

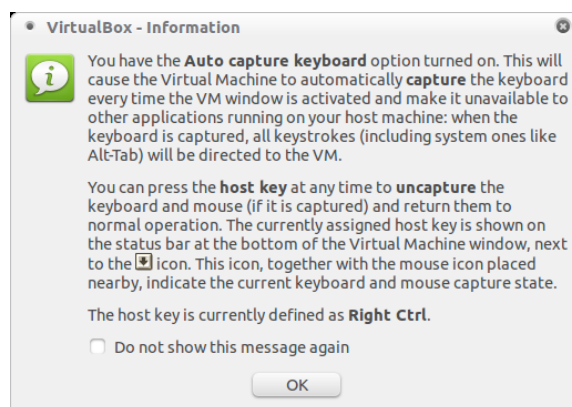


3.4.4 Launching the virtual machine

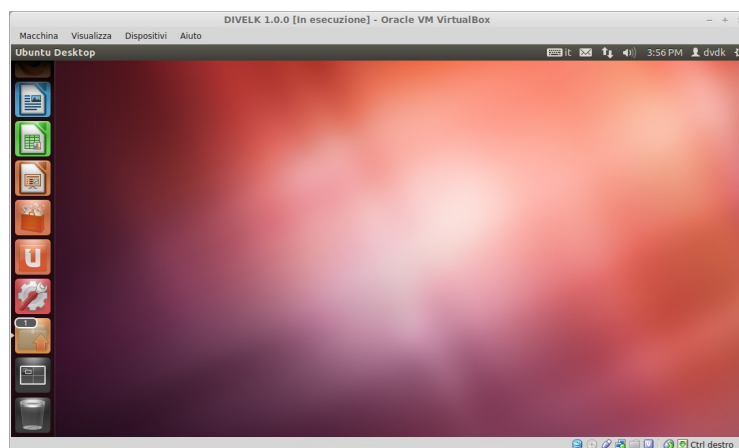
1. Once the virtual machine is ready, launch it by clicking on the start icon



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

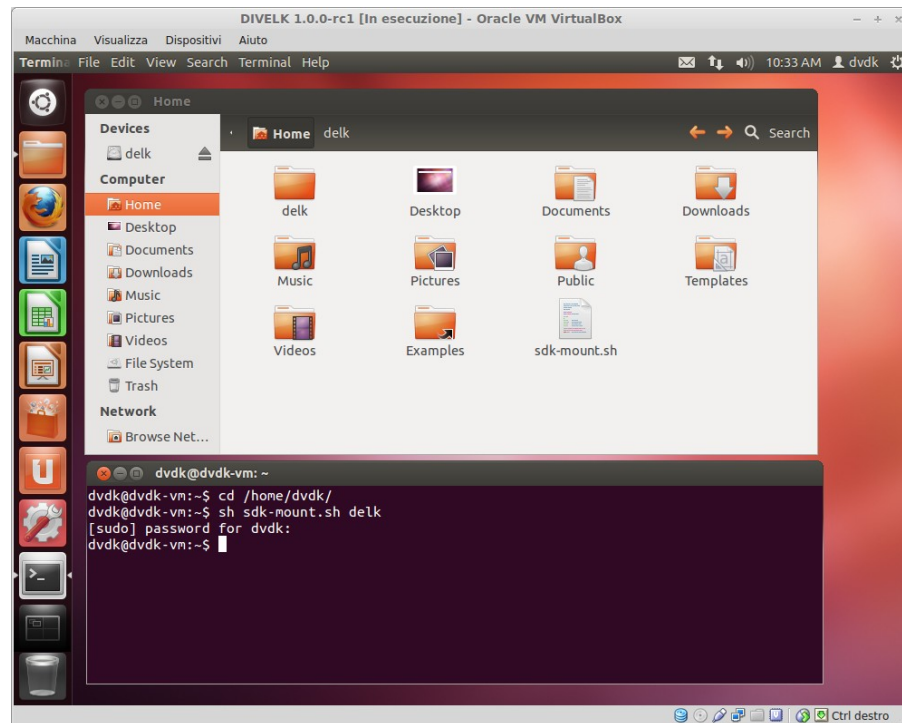


3. At the end of the boot process, the Ubuntu/Lubuntu desktop will be available.



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

```
cd /home/dvdk  
sh sdk-mount.sh divelk
```

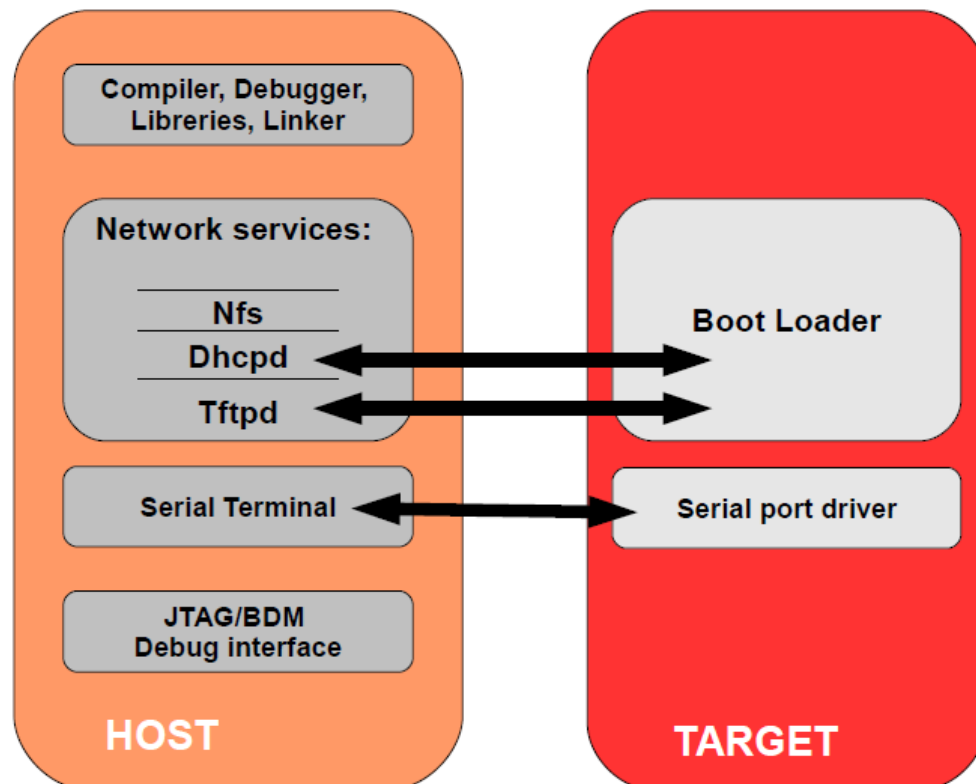



5. Once logged in, the system could suggest to update the Virtualbox Guest Additions package. You can follow the on-screen instructions to easily install the updated package.
6. Check if your keyboard layout matches the Ubuntu keyboard settings. You can change the keyboard layout selecting System->Preferences->Keyboard 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 for building 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 use 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>.

4.2.3.1 Linux Device Tree

The Flattened Device Tree (FDT) is a data structure for describing the hardware in a system (for further information,

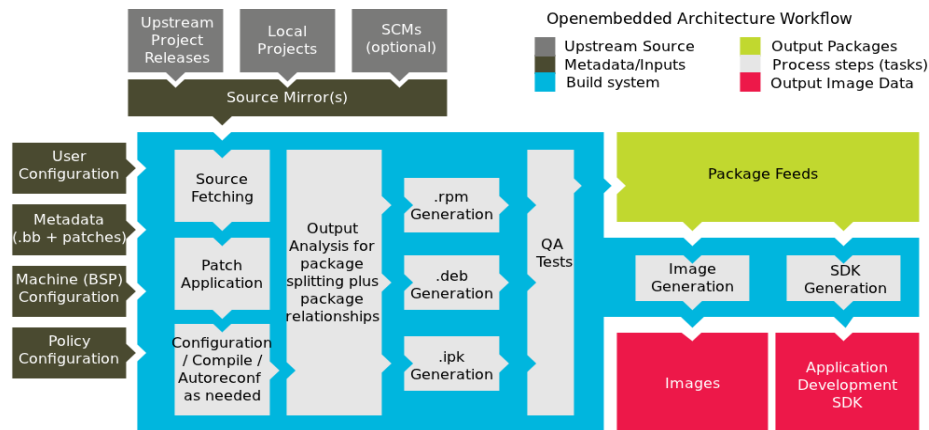
please refer to http://elinux.org/Device_Tree). Device tree source code is stored into the `arch/arm/boot/dts/` directory.

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 definitely 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 provide a pre-packaged root file systems that make this task much easier. However, usign a build tool as Yocto, developers can build their own version of the root file system.

4.2.5 Yocto

The Yocto Project, hosted by the Linux Foundation, provides open source, high-quality infrastructure and tools to help developers create their own custom Linux distributions for any hardware architecture and across multiple market segments. The Yocto Project is intended to simplify the work of the developers, providing a set of tools and components, including a highly configurable build system, that enables users to construct their own custom distributions, targeted for specific embedded devices. It is not, itself, a Linux distribution. Rather, it is capable of producing an image for a particular embedded device without dictating the composition of the Linux distribution actually built or the hardware architecture used.



The software components for the DIVA platform can be built

using Yocto and the source trees released with the DIVELK 2.0.0.

4.3 Working with DIVELK

4.3.1 The build system

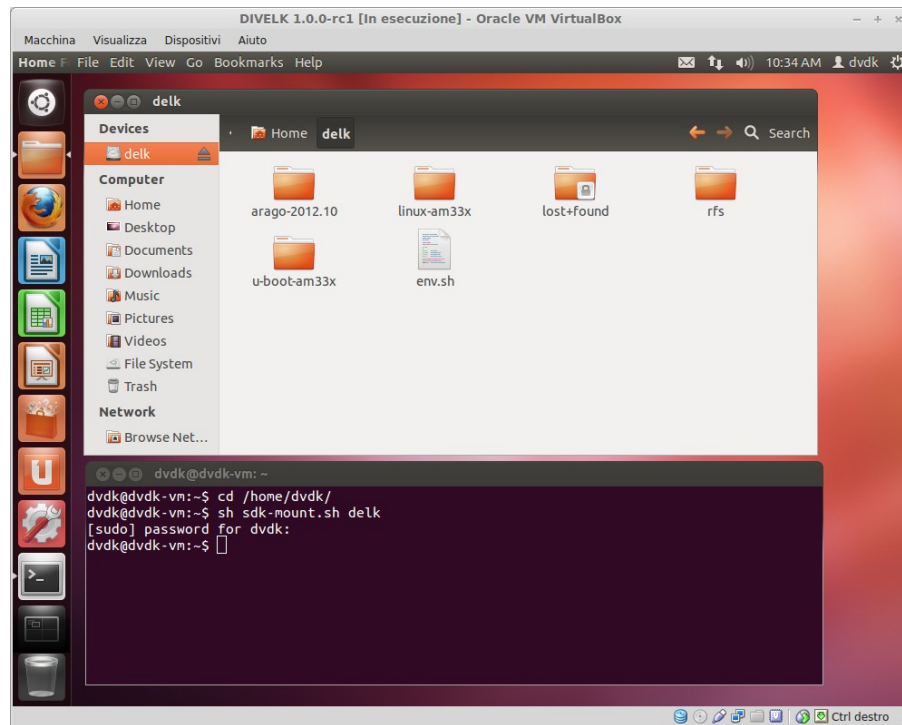
A build system is a set of source trees, Makefiles, patches, configuration files, tools and scripts that makes it easy to generate all the components of a complete embedded Linux system. A build system, once properly set up, automates the configuration and cross-compilation processes, generating all 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:

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

The Texas Instruments Arago distribution (http://arago-project.org/wiki/index.php/Main_Page) and therefore DIVELK 2.0.0 are based on Yocto.

4.3.2 Overview of the installed components

Once the virtual machine is running and the secondary disk is mounted, the actual development kit can be found into the directory `/home/dvdk/divelk`:



The divelk directory contains the following subdirectories:

- **yocto**: the Yocto SDK installation directory (including the cross-toolchain - GNU Compiler Collection (GCC) version is 4.7.3)
- **linux-am33x**: the Linux source tree
- **u-boot-am33x**: the U-Boot source tree
- **diva-bsp**: the Yocto BSP for DIVA
- **qtcreator-x.y.z**: pre-installed and pre-configured QtCreator IDE for Qt application development
- **rfs**: DIVELK provides two root file systems:
 - **diva-base**: minimal root file system with basic packages (/home/dvdk/divelk/rfs/diva-base)
 - **diva-matrix**: full root file system with lots of packages (including the matrix gui application from Texas Instruments) useful during the development phase (/home/dvdk/divelk/rfs/diva-matrix)

- `diva-qt5`: root file system with Qt 5.3.2 libraries and examples (`/home/dvdk/divelk/rfs/diva-qt5`)
- `env.sh`: a bash script for setting the environment variables, containing the following lines:

```
source
~/divelk/yocto/sdk/diva-qt5/environment-setup
export LDFLAGS=-L$
{PKG_CONFIG_SYSROOT_DIR}/usr/lib/4.7.3
export CFLAGS=""
export CPPFLAGS=""
export CXXFLAGS=""
export CPATH=""
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-
```

4.3.3 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.

4.3.3.1 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. A subdirectory dedicated to the image files associated to the DIVELK (`/srv/tftp/divelk`) is available,

but developers can add their custom subdirectories when required.

We recommend reading the http://wiki.dave.eu/index.php/VirtualBox_Network_Configuration page for additional details on the DVDK networking configuration.

4.3.3.2 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 developers on their 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, developers 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 (Network File System). 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.

We recommend reading the http://wiki.dave.eu/index.php/VirtualBox_Network_Configuration page for additional details on the DVDK networking configuration and on how to enable NFS.

4.3.3.3 Pre-built toolchain

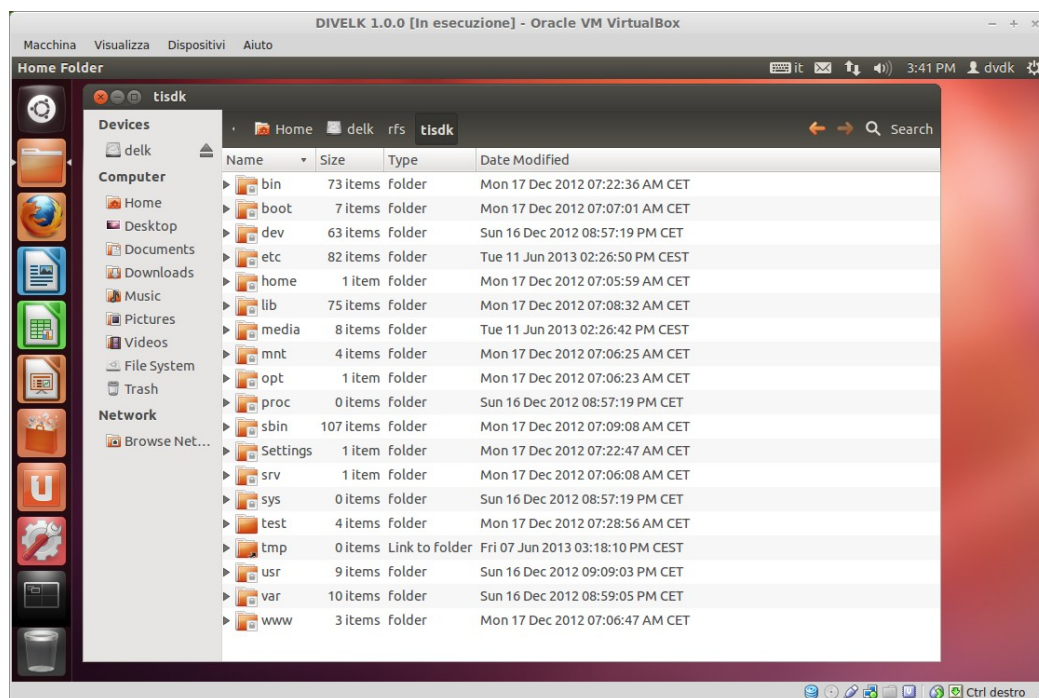
To start developing software for the DIVA platform, users need a proper toolchain, which can be pre-built or built-from-scratch. Building a toolchain from scratch is not a trivial task (though using a recent build system is easier than in the past), so the recommended approach consists in using a pre-built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

DIVELK provides the Linaro gnueabihf-4.7-2013.03 toolchain

(GCC version is 4.7.3).

4.3.3.4 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 <http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/>.

DIVELK provides pre-built root file systems, that can be used during the evaluation/development phase, since they contains the software packages for working with the DIVA platform.

DIVELK root file systems are built with the Yocto build system and are stored into the following directories:

- `/home/dvdk/divelk/rfs/diva-base`

- `/home/dvdk/divelk/rfs/diva-matrix`
- `/home/dvdk/divelk/rfs/diva-qt5`

4.4 Building the software components with Yocto

The build process creates an entire Linux distribution from source. The build process can be summarized as follows:

- Make sure that all the prerequisites are met
- Initialize the build environment, as described in 4.4.2.
- Optionally ensure the `conf/local.conf` configuration file, which is found in the Build Directory, is set up how you want it. This file defines many aspects of the build environment including the target machine architecture through the `MACHINE` variable, the development machine's processor use through the `BB_NUMBER_THREADS` and `PARALLEL_MAKE` variables, and a centralized tarball download directory through the `DL_DIR` variable.
- Build the image using the `bitbake` command. If you want information on BitBake, see the BitBake User Manual.

N.B. Since the DIVELK virtual machine is already configured to match all the requirements for using the Yocto build system, developers who wants to quickly build a Yocto image can directly go to section 4.4.3.

4.4.1 Prerequisites

The following prerequisites are required and only need to be done once. Please note that the DIVELK virtual machine is already configured to match all the requirements for using the Yocto build system.

Some generic development tools are required on the host Linux machine:

- `git`
- `curl`
- `build-essential`
- `diffstat`

- texinfo
- gawk
- chrpath
- ia32-libs (if the host machine is running a 64-bit OS)
- python-m2crypto

These packages can be installed with the following command:

```
sudo apt-get install git build-essential diffstat  
texinfo gawk chrpath ia32-libs python-m2crypto
```

It is also recommended to switch the system shell from Ubuntu's standard dash to more universal bash:

```
$ sudo dpkg-reconfigure dash
```

4.4.2 Initializing the build environment

In the DIVELK, we have simplified the Yocto initialization phase, relying on the repo tool and on a Diva bsp git repository, so that the initialization can be completed with a few commands as reported below:

```
$ curl  
http://commondatastorage.googleapis.com/git-repo-downloads/repo > repo  
$ chmod +x repo  
$ ./repo init -u  
/home/shared/devel/dave/diva-DDx/sw/linux/yocto/diva-bsp.git -b diva-dev-dylan  
$ ./repo sync
```

4.4.3 Build the Yocto image

Please note that even building the basic root file system requires a few hours to complete the process on a mid-hi range desktop PC (4-6 cores, 8-12 GiB RAM), also depending on the Internet connection speed (all source are fetched from the network). Nearly 20GiB of disk space is required for the build. Moreover, building inside the DVDK adds some overhead, since the performances of a virtual machine are reduced if compared to the physical hardware. Thus, it's recommended to check the

hardware capabilities of the host system and, when building with Yocto is required, developers should consider the following options:

- migrating the build system to a physical machine
- assuming that the host system has the required resources, extending the hardware capabilities of the default DVDK (eg: adding more cores and disk space)

Once completed the initialization phase, developers can launch the Yocto image build process with the following commands:

```
$ cd ~/divelk/diva-bsp
$ source ~/divelk/env.sh
$ source diva-bsp-init-env.sh
$ bitbake matrix-rootfs-image
```

Please note that three different images are available:

- matrix-rootfs-image (includes the matrix gui application from Texas Instruments)
- qt5-rootfs-image (root file system with Qt 5.1.1 libraries and examples)
- base-rootfs-image (minimal root file system)

The resulting files (kernel, device tree and u-boot binaries, plus root file system in a .tar.gz archive) will then be available inside the `build/tmp/deploy/images/diva` directory.

4.5 Building the software components outside Yocto

4.5.1 Build/configure U-Boot

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

```
dvdk@dvdk-vm:~/divelk/u-boot-am33x$ make diva_spiboot
dvdk@dvdk-vm:~/divelk/u-boot-am33x$ 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:

```
dvd@dvdk-vm:~/dvd/divelk/u-boot-am33x$ sudo cp MLO
MLO.byteswap u-boot.img /srv/tftp/divelk/
```

4.5.2 Build/configure Linux kernel

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

```
dvd@dvdk-vm:~/divelk/linux-am33x$ make diva_defconfig
dvd@dvdk-vm:~/divelk/linux-am33x$ make uImage
am335x-divelk.dtb
```

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

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/arch/arm/boot/uImage` file, while the dtb is stored into the

`linux-am33x/arch/arm/boot/dts/am335x-divelk.dtb` file.

These files can be copied to the `/srv/tftp/divelk/` directory with the following commands:

```
dvd@dvdk-vm:~/divelk/linux-am33x$ sudo cp
```

```
arch/arm/boot/uImage /srv/tftp/divelk/  
dvdk@dvdk-vm:~/dvdk/divelk/linux-am33x$ sudo cp  
arch/arm/boot/dts/am335x-divelk.dtb /srv/tftp/divelk/
```

4.5.3 Build a custom application

Some users may prefer to cross-compile their applications outside of the Yocto flow. It may be specifically useful and easier for new projects in their prototyping and proof-of-concept stages or for any smaller applications in general. This way users don't have to worry about creating Yocto "recipes" for their applications and becoming familiar with the entire Yocto build system.

In order to cross-compile an application, written in C/C++, the cross-toolchain provided with the DIVELK is required.

Assuming that you've configured the environment variables sourcing the `env.sh` script, developers can write a simple "Hello world" application, called for example `hello.c`:

```
#include <stdio.h>  
  
int main()  
{  
    printf("Hello world\n");  
    return 0;  
}
```

To cross-compile it:

```
$ arm-none-linux-gnueabi-gcc -o hello hello.c
```

Copy the executable file to the root file system, and execute it from the Diva system:

```
root@diva:~# ./hello  
Hello world
```


5 DIVEK use cases

5.1 DIVA Frequency Scaling

The frequency of the CPU can be changed on the run using the Cpubfreq framework (please refer to the documentation included into the `linux-am33x/Documentation/cpu-freq` directory of the kernel source tree). The `cpufreq` framework works in conjunction with the related driver & governor.

Cpubfreq implementation controls the Linux OPP (Operating Performance Points) adjusting the Cortex-A8 core voltages and frequencies. CPUFreq is enabled by default in the DIVA kernel configuration.

To view the available governors:

```
root@am335x-evm:~# cat
/sys/devices/system/cpu/cpu0/cpufreq/scaling_available
_governors
conservative ondemand powersave userspace performance
root@am335x-evm:~#
```

To view the supported OPP's (frequency in KHz):

```
root@am335x-evm:~# cat
/sys/devices/system/cpu/cpu0/cpufreq/scaling_available
_frequencies
275000 500000 600000 720000
root@am335x-evm:~#
```

To change the OPP:

```
root@am335x-evm:~# echo 275000 >
/sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
root@am335x-evm:~#
```

Please note that OPP can be changed only using the userspace governor. If governors like `ondemand` is used, OPP change happens automatically based on the system load.

For further details, please refer to

http://processors.wiki.ti.com/index.php/AM335x_Linux_Power

[Management User Guide](#).

5.2 Forcing the kernel to use a limited amount of RAM

To evaluate the performances of the system with a specific amount of available RAM, the user can pass the mem parameter to the kernel, by setting the command line arguments in u-boot (for detailed instruction, please refer to the [Change Linux Command Line Parameter from U-boot](#) page). For example, to limit the amount of RAM to 128 MB, create the following variables in u-boot:

```
DIVA# setenv mem 128MB
```

```
DIVA# setenv addmem 'setenv bootargs ${bootargs} mem=${mem}'
```

And add the addmem variable to the boot macro:

```
DIVA# setenv net_nfs 'run loadk nfsargs addip addcons  
addmem; bootm ${buf}'
```

For further details, please refer to the [linux-am33x/Documentation/kernel-parameters.txt](#) file of the kernel source tree provided with the DIVELK.

6 Frequently Asked Questions

6.1 Q: Where can I found DIVA SOM information?

A: please refer to the following table:

Document	Location
DIVA main page on DAVE Embedded Systems 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

6.2 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 MicroSD 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)

6.3 Q: How can I update the DIVELK version?

A: please refer to the following page on the **DAVE Embedded Systems** Developer's Wiki:

http://wiki.dave.eu/index.php/Software_Manual_%28Diva%29#DIVELK_Updates

6.4 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

6.5 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/divelk/rfs/ diva-matrix
bootfile	Path to the kernel binary image on the tftp server	divelk/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
```

6.6 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_Realtime_Unit_Subsystem

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

6.7 **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)

6.8 **Q: The system doesn't boot anymore. On the serial console I see just "CCCC" characters... Is there a recovery procedure?**

A: "CCCC" characters are the output of the internal bootrom that tries to boot from UART. It's likely that the first stage bootloader is missing or corrupted. System recovery is possible using one of the following strategies:

- bootable SD card (if available, for example the one provided with the DIVELK)
- serial file transfer (via XMODEM/YMODEM) - please refer to http://processors.wiki.ti.com/index.php/AM335x_U-Boot_User%27s_Guide#Boot_Over_UART
- JTAG (requires dedicated hardware - JTAG port, debugger/programmer - and software tools)

7 Appendices

7.1 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 2013.10 (Sep 13 2014 - 10:43:18) [divelk-2.0.0]
reading u-boot.img
reading u-boot.img

U-Boot 2013.10 (Sep 13 2014 - 10:43:18) [divelk-2.0.0]

AM335X-GP rev 1 720 MHz
I2C: ready
DRAM: 512 MiB
WARNING: Caches not enabled
Now running in RAM - U-Boot at: 9ff50000
NAND: 1024 MiB
MMC: OMAP SD/MMC: 0, OMAP SD/MMC: 1
SF: Detected S25FL256S_64K with page size 256 Bytes, erase size 64 KiB, total 32 MiB
*** Warning - bad CRC, using default environment

Net: cpsw, usb_ether
Hit any key to stop autoboot: 0
trying to boot from DIVEK MMC

reading boot.scr
419 bytes read in 5 ms (81.1 KiB/s)
## Executing script at 80900000
reading uImage
4714280 bytes read in 490 ms (9.2 MiB/s)
reading am335x-divelk.dtb
37129 bytes read in 11 ms (3.2 MiB/s)
## Booting kernel from Legacy Image at 80200000 ...
Image Name: Linux-3.12.10-divelk-2.0.0
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 4714216 Bytes = 4.5 MiB
Load Address: 80008000
Entry Point: 80008000
Verifying Checksum ... OK
## Flattened Device Tree blob at 80f80000
Booting using the fdt blob at 0x80f80000
Loading Kernel Image ... OK
Loading Device Tree to 9fe02000, end 9fe0e108 ... OK

Starting kernel ...

[ 0.000000] Booting Linux on physical CPU 0x0
[ 0.000000] Linux version 3.12.10-divelk-2.0.0 (amon@linuxserver2) (gcc version 4.7.3
20130226 (prerelease) (crosstool-NG linaro-1.13.1-4.7-2013.03-20130313 - Linaro GCC 2013.03)
) #26 SMP Sat Sep 13
10:45:59 CEST 2014
[ 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: Generic AM33XX (Flattened Device Tree), model: DAVE Embedded System
DIVEK
[ 0.000000] cma: CMA: reserved 24 MiB at 9e000000
[ 0.000000] Memory policy: ECC disabled, Data cache writeback
[ 0.000000] CPU: All CPU(s) started in SVC mode.
[ 0.000000] AM335X ES1.0 (sgx neon )
[ 0.000000] PERCPU: Embedded 8 pages/cpu @c0d9a000 s9024 r8192 d15552 u32768

```

```

[ 0.000000] Built 1 zonelists in Zone order, mobility grouping on. Total pages: 129280
[ 0.000000] Kernel command line: root=/dev/mmcblk0p2 rw console=tty00,115200n8
consoleblank=0
mtdparts=omap2-nand.0:128k(SPL),128k(SPL.backup1),128k(SPL.backup2),128k(SPL.backup3),1920k(
u-boot),128k(
u-boot-env),128k(fdt),6m(kernel),-(rootfs);spi.0:256k(MLO),512k(u-boot),256k(env1),256k(env
2),256k(fdt),6m(kernel),-(free)
[ 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: 482188K/521216K available (6358K kernel code, 562K rwdata, 2168K
rdata, 368K init, 254K bss, 39028K reserved, 0K highmem)
[ 0.000000] Virtual kernel memory layout:
[ 0.000000]   vector : 0xffff0000 - 0xffff1000   ( 4 kB)
[ 0.000000]   fixmap : 0xffff0000 - 0xffffe000   ( 896 kB)
[ 0.000000]   vmalloc : 0xe0800000 - 0xff000000   ( 488 MB)
[ 0.000000]   lowmem  : 0xc0000000 - 0xe0000000   ( 512 MB)
[ 0.000000]   pkmap   : 0xbf000000 - 0xc0000000   ( 2 MB)
[ 0.000000]   modules : 0xbf000000 - 0xbf000000   ( 14 MB)
[ 0.000000]   .text   : 0xc0008000 - 0xc085bf9c   (8528 kB)
[ 0.000000]   .init   : 0xc085c000 - 0xc08b8340   ( 369 kB)
[ 0.000000]   .data   : 0xc08ba000 - 0xc09468b0   ( 563 kB)
[ 0.000000]   .bss   : 0xc09468b8 - 0xc09863f4   ( 255 kB)
[ 0.000000] Hierarchical RCU implementation.
[ 0.000000] RCU restricting CPUs from NR_CPUS=2 to nr_cpu_ids=1.
[ 0.000000] NR_IRQS:16 nr_irqs:16 16
[ 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: timer2 at 24000000 Hz
[ 0.000000] sched_clock: 32 bits at 24MHz, resolution 41ns, wraps every 178956ms
[ 0.000000] OMAP clocksource: timer1 at 24000000 Hz
[ 0.000000] Console: colour dummy device 80x30
[ 0.000409] Calibrating delay loop... 479.23 BogoMIPS (lpj=2396160)
[ 0.119684] pid_max: default: 32768 minimum: 301
[ 0.119839] Security Framework initialized
[ 0.119964] Mount-cache hash table entries: 512
[ 0.131811] CPU: Testing write buffer coherency: ok
[ 0.132332] CPU0: thread -1, cpu 0, socket -1, mpidr 0
[ 0.132408] Setting up static identity map for 0xc063b648 - 0xc063b6b8
[ 0.137504] Brought up 1 CPUs
[ 0.137533] SMP: Total of 1 processors activated.
[ 0.137544] CPU: All CPU(s) started in SVC mode.
[ 0.138587] devtmpfs: initialized
[ 0.143587] VFP support v0.3: implementor 41 architecture 3 part 30 variant c rev 3
[ 0.209601] omap_hwmod: debugss: _wait_target_disable failed
[ 0.210581] pinctrl core: initialized pinctrl subsystem
[ 0.211857] regulator-dummy: no parameters
[ 0.214690] NET: Registered protocol family 16
[ 0.217798] DMA: preallocated 256 KiB pool for atomic coherent allocations
[ 0.221492] cpuidle: using governor ladder
[ 0.221516] cpuidle: using governor menu
[ 0.230756] platform mpu.1: FIXME: clock-name 'fck' DOES NOT exist in dt!
[ 0.232413] platform 49000000.edma: FIXME: clock-name 'fck' DOES NOT exist in dt!
[ 0.234218] OMAP GPIO hardware version 0.1
[ 0.247827] omap-gpmc 50000000.gpmc: could not find pctldev for node
/pinmux@44e10800/nandflash_pins_default, deferring probe
[ 0.247871] platform 50000000.gpmc: Driver omap-gpmc requests probe deferral
[ 0.248489] platform 56000000.sgx: FIXME: clock-name 'fck' DOES NOT exist in dt!
[ 0.253038] DSS not supported on this SoC
[ 0.253065] No ATAGs?
[ 0.253079] hw-breakpoint: debug architecture 0x4 unsupported.
[ 0.295970] bio: create slab <bio-0> at 0
[ 0.319222] edma-dma-engine edma-dma-engine.0: TI EDMA DMA engine driver
[ 0.321038] vin: 5000 mV
[ 0.323718] SCSI subsystem initialized
[ 0.325674] usbcore: registered new interface driver usbfs
[ 0.325939] usbcore: registered new interface driver hub
[ 0.326254] usbcore: registered new device driver usb
[ 0.327385] omap_i2c 44e0b000.i2c: could not find pctldev for node
/pinmux@44e10800/pinmux_i2c0_pins, deferring probe
[ 0.327421] platform 44e0b000.i2c: Driver omap_i2c requests probe deferral
[ 0.327950] pps_core: LinuxPPS API ver. 1 registered
[ 0.327964] pps_core: Software ver. 5.3.6 - Copyright 2005-2007 Rodolfo Giometti

```



```
<giometti@linux.it>
[ 0.328161] PTP clock support registered
[ 0.330978] Advanced Linux Sound Architecture Driver Initialized.
[ 0.332716] Switched to clocksource timer1
[ 0.363794] NET: Registered protocol family 2
[ 0.364657] TCP established hash table entries: 4096 (order: 3, 32768 bytes)
[ 0.364762] TCP bind hash table entries: 4096 (order: 3, 32768 bytes)
[ 0.364847] TCP: Hash tables configured (established 4096 bind 4096)
[ 0.364928] TCP: reno registered
[ 0.364948] UDP hash table entries: 256 (order: 1, 8192 bytes)
[ 0.364976] UDP-Lite hash table entries: 256 (order: 1, 8192 bytes)
[ 0.365256] NET: Registered protocol family 1
[ 0.365759] RPC: Registered named UNIX socket transport module.
[ 0.365774] RPC: Registered udp transport module.
[ 0.365783] RPC: Registered tcp transport module.
[ 0.365793] RPC: Registered tcp NFSv4.1 backchannel transport module.
[ 0.367016] NetWinder Floating Point Emulator V0.97 (double precision)
[ 0.367827] PM: Loading am335x-pm-firmware.bin
[ 0.530548] VFS: Disk quotas dquot_6.5.2
[ 0.530738] Dquot-cache hash table entries: 1024 (order 0, 4096 bytes)
[ 0.531740] NFS: Registering the id_resolver key type
[ 0.531867] Key type id_resolver registered
[ 0.531880] Key type id_legacy registered
[ 0.532254] NTFS driver 2.1.30 [Flags: R/W].
[ 0.532307] jffs2: version 2.2. (NAND) (SUMMARY) © 2001-2006 Red Hat, Inc.
[ 0.532535] msgmni has been set to 989
[ 0.535449] NET: Registered protocol family 38
[ 0.535511] io scheduler noop registered
[ 0.535523] io scheduler deadline registered
[ 0.535560] io scheduler cfq registered (default)
[ 0.538175] pinctrl-single 44e10800.pinmux: 142 pins at pa f9e10800 size 568
[ 0.590593] Console: switching to colour frame buffer device 100x30
[ 0.598277] Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
[ 0.601816] 44e09000.serial: ttyO0 at MMIO 0x44e09000 (irq = 88, base_baud = 3000000) is
a OMAP UART0
[ 1.281209] console [ttyO0] enabled
[ 1.286942] omap_rng 48310000.rng: OMAP Random Number Generator ver. 20
[ 1.310673] brd: module loaded
[ 1.323060] loop: module loaded
[ 1.333908] mtdoops: mtd device (mtddev=name/number) must be supplied
[ 1.343735] edma-dma-engine edma-dma-engine.0: allocated channel for 0:17
[ 1.350969] edma-dma-engine edma-dma-engine.0: allocated channel for 0:16
[ 1.358842] m25p80 spi1.0: s25fl1256s1 (32768 Kbytes)
[ 1.364293] 7 cmdlinepart partitions found on MTD device spi1.0
[ 1.370563] Creating 7 MTD partitions on "spi1.0":
[ 1.375888] 0x0000000000000-0x000000040000 : "MLO"
[ 1.383286] 0x000000040000-0x0000000c0000 : "u-boot"
[ 1.390594] 0x0000000c0000-0x000000100000 : "env1"
[ 1.397868] 0x000000100000-0x000000140000 : "env2"
[ 1.405253] 0x000000140000-0x000000180000 : "fdt"
[ 1.412191] 0x000000180000-0x000000780000 : "kernel"
[ 1.419714] 0x000000780000-0x0000002000000 : "free"
[ 1.428475] CAN device driver interface
[ 1.434708] c_can_platform 481cc000.d_can: c_can_platform device registered
(regs=fa1cc000, irq=68)
[ 1.446015] usbcore: registered new interface driver asix
[ 1.452044] usbcore: registered new interface driver ax88179_178a
[ 1.458930] usbcore: registered new interface driver cdc_ether
[ 1.465550] usbcore: registered new interface driver r815x
[ 1.471615] usbcore: registered new interface driver smc95xx
[ 1.477990] usbcore: registered new interface driver net1080
[ 1.484283] usbcore: registered new interface driver cdc_subset
[ 1.490779] usbcore: registered new interface driver zaurus
[ 1.497093] usbcore: registered new interface driver cdc_ncm
[ 1.503937] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
[ 1.510886] ehci-omap: OMAP-EHCI Host Controller driver
[ 1.517235] usbcore: registered new interface driver cdc_wdm
[ 1.523592] usbcore: registered new interface driver usb-storage
[ 1.530485] usbcore: registered new interface driver usbserial
[ 1.537087] usbcore: registered new interface driver usbserial_generic
[ 1.544375] usbserial: USB Serial support registered for generic
[ 1.550984] usbcore: registered new interface driver ftdi_sio
[ 1.557337] usbserial: USB Serial support registered for FTDI USB Serial Device
```

```
[ 1.569745] musb-hdrc musb-hdrc.0.auto: Enabled SW babble control
[ 1.576609] musb-hdrc musb-hdrc.0.auto: Failed to request rx1.
[ 1.583022] musb-hdrc musb-hdrc.0.auto: musb_init_controller failed with status -517
[ 1.591267] platform musb-hdrc.0.auto: Driver musb-hdrc requests probe deferral
[ 1.601463] musb-hdrc musb-hdrc.1.auto: Enabled SW babble control
[ 1.608210] musb-hdrc musb-hdrc.1.auto: Failed to request rx1.
[ 1.614488] musb-hdrc musb-hdrc.1.auto: musb_init_controller failed with status -517
[ 1.622856] platform musb-hdrc.1.auto: Driver musb-hdrc requests probe deferral
[ 1.653327] mousedev: PS/2 mouse device common for all mice
[ 1.660663] input: ti-tsc as /devices/ocp.2/44e0d000.tscadc/TI-am335x-tsc/input/input0
[ 1.671593] i2c /dev entries driver
[ 1.678569] omap_wdt: OMAP Watchdog Timer Rev 0x01: initial timeout 60 sec
[ 1.688151] edma-dma-engine edma-dma-engine.0: allocated channel for 0:25
[ 1.695572] edma-dma-engine edma-dma-engine.0: allocated channel for 0:24
[ 1.702910] omap_hsmmc 48060000.mmc: unable to get vmmc regulator -517
[ 1.709845] edma-dma-engine edma-dma-engine.0: freeing channel for 24
[ 1.716920] edma-dma-engine edma-dma-engine.0: freeing channel for 25
[ 1.723910] platform 48060000.mmc: Driver omap_hsmmc requests probe deferral
[ 1.732299] ledtrig-cpu: registered to indicate activity on CPUs
[ 1.739174] edma-dma-engine edma-dma-engine.0: allocated channel for 0:36
[ 1.746525] omap-sham 53100000.sham: hw accel on OMAP rev 4.3
[ 1.754574] omap-aes 53500000.aes: OMAP AES hw accel rev: 3.2
[ 1.760718] edma-dma-engine edma-dma-engine.0: allocated channel for 0:5
[ 1.768028] edma-dma-engine edma-dma-engine.0: allocated channel for 0:6
[ 1.777248] usbcore: registered new interface driver usbhid
[ 1.783237] usbhid: USB HID core driver
[ 1.791362] usbcore: registered new interface driver snd-usb-audio
[ 1.801674] UDA134X SoC Audio Codec
[ 1.824052] davinci_evm sound.4: uda134x-hifi <-> 48038000.mcaspc mapping ok
[ 1.837299] oprofile: no performance counters
[ 1.842555] oprofile: using timer interrupt.
[ 1.847618] TCP: cubic registered
[ 1.851130] Initializing XFRM netlink socket
[ 1.855784] NET: Registered protocol family 17
[ 1.860546] NET: Registered protocol family 15
[ 1.865336] can: controller area network core (rev 20120528 abi 9)
[ 1.872096] NET: Registered protocol family 29
[ 1.876881] can: raw protocol (rev 20120528)
[ 1.881397] can: broadcast manager protocol (rev 20120528 t)
[ 1.887520] 8021q: 802.1Q VLAN Support v1.8
[ 1.892100] Key type dns_resolver registered
[ 1.897496] cpu cpu0: cpu0 regulator not ready, retry
[ 1.903001] platform cpufreq-cpu0.0: Driver cpufreq-cpu0 requests probe deferral
[ 1.911300] PM: CM3 Firmware Version = 0x187
[ 1.916487] ThumbEE CPU extension supported.
[ 1.923680] omap-gpmc 50000000.gpmc: GPMC revision 6.0
[ 1.929131] gpmc_mem_init: disabling cs 0 mapped at 0x0-0x1000000
[ 1.937019] NAND device: Manufacturer ID: 0xec, Chip ID: 0xd3 (Samsung NAND 1GiB 3,3V
8-bit), 1024MiB, page size: 2048, OOB size: 64
[ 1.949754] omap2-nand: detected x8 NAND flash
[ 1.954499] nand: using OMAP_ECC_BCH8_CODE_HW ECC scheme
[ 1.960175] omap2-nand: using custom ecc layout
[ 1.965136] 9 cmdlinepart partitions found on MTD device omap2-nand.0
[ 1.971955] Creating 9 MTD partitions on "omap2-nand.0":
[ 1.977663] 0x0000000000000-0x000000020000 : "SPL"
[ 1.985117] 0x000000020000-0x000000040000 : "SPL.backup1"
[ 1.993214] 0x000000040000-0x000000060000 : "SPL.backup2"
[ 2.001160] 0x000000060000-0x000000080000 : "SPL.backup3"
[ 2.009145] 0x000000080000-0x000000260000 : "u-boot"
[ 2.018198] 0x000000260000-0x000000280000 : "u-boot-env"
[ 2.026420] 0x000000280000-0x0000002a0000 : "fdt"
[ 2.033730] 0x0000002a0000-0x0000008a0000 : "kernel"
[ 2.046645] 0x0000008a0000-0x000004000000 : "rootfs"
[ 2.910559] vrtc: 1800 mV
[ 2.914394] vrtc: supplied by vin
[ 2.921442] vio: 1500 <-> 3300 mV at 1500 mV
[ 2.926452] vio: supplied by vin
[ 2.934061] vdd_mpu: 600 <-> 1500 mV at 1137 mV
[ 2.939149] vdd_mpu: supplied by vin
[ 2.947000] vdd_core: 600 <-> 1500 mV at 1137 mV
[ 2.952150] vdd_core: supplied by vin
[ 2.959384] vdd3: 5000 mV
[ 2.965617] vdig1: 1200 <-> 2700 mV at 1800 mV
```

```

[ 2.970598] vdig1: supplied by vin
[ 2.977702] vdig2: 1200 <--> 1800 mV at 1800 mV
[ 2.982663] vdig2: supplied by vin
[ 2.989856] vpll: 1800 <--> 2500 mV at 1800 mV
[ 2.994840] vpll: supplied by vin
[ 3.001759] vdac: 1800 <--> 2850 mV at 1800 mV
[ 3.006728] vdac: supplied by vin
[ 3.013866] vaux1: 1800 <--> 2850 mV at 1800 mV
[ 3.018861] vaux1: supplied by vin
[ 3.026421] vaux2: 1800 <--> 2800 mV at 3300 mV
[ 3.031390] vaux2: supplied by vin
[ 3.038485] vaux33: 1800 <--> 3300 mV at 3300 mV
[ 3.043689] vaux33: supplied by vin
[ 3.050823] vmmc: 1800 <--> 3300 mV at 3300 mV
[ 3.055802] vmmc: supplied by vin
[ 3.073949] tps65910-rtc tps65910-rtc: rtc core: registered tps65910-rtc as rtc0
[ 3.083252] at24 0-0050: 4096 byte 24c32 EEPROM, writable, 32 bytes/write
[ 3.092440] ina2xx 0-0041: power monitor ina226 (Rshunt = 10000 uOhm)
[ 3.099442] omap_i2c 44e0b000.i2c: bus 0 rev0.11 at 100 kHz
[ 3.106054] musb-hdrc musb-hdrc.0.auto: Enabled SW babble control
[ 3.114825] musb-hdrc musb-hdrc.0.auto: MUSB HDRC host driver
[ 3.121563] musb-hdrc musb-hdrc.0.auto: new USB bus registered, assigned bus number 1
[ 3.130296] usb usb1: New USB device found, idVendor=1d6b, idProduct=0002
[ 3.137543] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 3.145203] usb usb1: Product: MUSB HDRC host driver
[ 3.150459] usb usb1: Manufacturer: Linux 3.12.10-divelk-2.0.0 musb-hcd
[ 3.157470] usb usb1: SerialNumber: musb-hdrc.0.auto
[ 3.164033] hub 1-0:1.0: USB hub found
[ 3.168106] hub 1-0:1.0: 1 port detected
[ 3.173859] musb-hdrc musb-hdrc.1.auto: Enabled SW babble control
[ 3.182507] musb-hdrc musb-hdrc.1.auto: MUSB HDRC host driver
[ 3.189383] musb-hdrc musb-hdrc.1.auto: new USB bus registered, assigned bus number 2
[ 3.198116] usb usb2: New USB device found, idVendor=1d6b, idProduct=0002
[ 3.205382] usb usb2: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 3.213083] usb usb2: Product: MUSB HDRC host driver
[ 3.218342] usb usb2: Manufacturer: Linux 3.12.10-divelk-2.0.0 musb-hcd
[ 3.225358] usb usb2: SerialNumber: musb-hdrc.1.auto
[ 3.231979] hub 2-0:1.0: USB hub found
[ 3.236124] hub 2-0:1.0: 1 port detected
[ 3.241593] edma-dma-engine edma-dma-engine.0: allocated channel for 0:25
[ 3.248906] edma-dma-engine edma-dma-engine.0: allocated channel for 0:24
[ 3.341278] mmc0: host does not support reading read-only switch. assuming write-enable.
[ 3.352042] mmc0: new high speed SDHC card at address 1234
[ 3.358955] mmcblk0: mmc0:1234 SA16G 14.6 GiB
[ 3.363802] davinci_mdio 4a101000.mdio: davinci mdio revision 1.6
[ 3.370249] davinci_mdio 4a101000.mdio: detected phy mask ffffffff
[ 3.379000] libphy: 4a101000.mdio: probed
[ 3.383422] davinci_mdio 4a101000.mdio: phy[6]: device 4a101000.mdio:06, driver SMSC
LAN8710/LAN8720
[ 3.393289] mmcblk0: p1 p2
[ 3.398432] Detected MACID = 00:50:c2:1e:af:e0
[ 3.409822] tps65910-rtc tps65910-rtc: setting system clock to 2014-10-03 10:36:29 UTC
(1412332589)
[ 3.425312] ALSA device list:
[ 3.428505] #0: Diva Dacu
[ 3.440728] kjournald starting. Commit interval 5 seconds
[ 3.461578] EXT3-fs (mmcblk0p2): using internal journal
[ 3.467158] EXT3-fs (mmcblk0p2): mounted filesystem with ordered data mode
[ 3.474540] VFS: Mounted root (ext3 filesystem) on device 179:2.
[ 3.483932] devtmpfs: mounted
[ 3.487639] Freeing unused kernel memory: 368K (c085c000 - c08b8000)
INIT: version 2.88 booting
Starting udev
[ 4.382357] udevd[954]: starting version 182
[ 9.887444] FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be
corrupt. Please run fsck.
[ 10.144522] cryptodev: driver 1.6 loaded.
Starting Bootlog daemon: bootlogd: cannot allocate pseudo tty: No such file or directory
bootlogd.
ALSA: Restoring mixer settings...
Configuring network interfaces... [ 13.122234] net eth0: initializing cpsw version 1.12
(0)
[ 13.130330] net eth0: phy found : id is : 0x7c0f1

```

