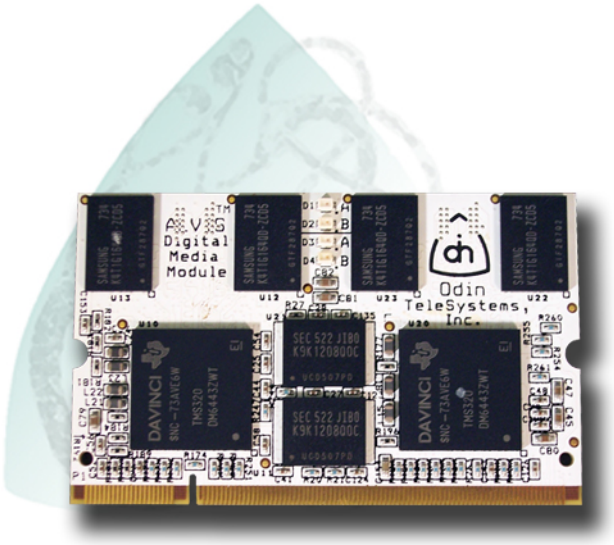
Odin TeleSystems Inc.



Alvis-DMP
Programmer's Guide

Doc. No. 1412-1-HCA-1022-1

Rev. 1.0

Changes

The material in this document is for information only and is subject to change without notice. While reasonable efforts have been made in the preparation of this document to assure its accuracy, Odin TeleSystems Inc., assumes no liability resulting from errors or omissions in this document, or from the use of the information contained herein.

Odin TeleSystems Inc. reserves the right to make changes in the product design without reservation and notification to its users.

Warranties

THE PRODUCT AND ITS DOCUMENTATION ARE PROVIDED “AS IS” AND WITHOUT WARRANTY OF ANY KIND. ODIN TELESYSTEMS EXPRESSLY DISCLAIMS ALL THE WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE. ODIN TELESYSTEMS DOES NOT WARRANT THAT THE FUNCTIONALITY OF THE PRODUCT WILL MEET ANY REQUIREMENTS, OR THAT THE OPERATIONS OF THE PRODUCT WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT DEFECTS WILL BE CORRECTED. FURTHERMORE, ODIN TELESYSTEMS DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OF THE PRODUCT OR ITS DOCUMENTATION IN TERMS OF THEIR CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE. NO ORAL OR WRITTEN INFORMATION OR ADVISE GIVEN BY ODIN TELESYSTEMS OR ODIN TELESYSTEMS’ AUTHORIZED REPRESENTATIVE SHALL CREATE A WARRANTY. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY.

UNDER NO CIRCUMSTANCE SHALL ODIN TELESYSTEMS INC., ITS OFFICERS, EMPLOYEES, OR AGENTS BE LIABLE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS, PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT AND ITS DOCUMENTATION, EVEN IF ODIN TELESYSTEMS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT WILL ODIN TELESYSTEMS’ LIABILITY FOR ANY REASON EXCEED THE ACTUAL PRICE PAID FOR THE PRODUCT AND ITS DOCUMENTATION. SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL AND CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY.

Table of Contents

1 Introduction

This document is targeted to systems integrators and application developers who are developing customized applications for the ARM9 core of the DaVinci Digital Media Processors (DMPs) on an Alvis-ASM, Alvis-PCIe, or Alvis-CSI board. It describes how to set up the environment to develop such applications, how to structure run and debug such applications, and also how to restore the flashed firmware on the board in case it would get corrupted by a faulty custom Alvis-DMP application.

The Alvis-ASM, Alvis-PCIe, and Alvis-CSI boards are members of the Odin Telecom frameworX (OTX) family of boards.

Developers who intend to use the standard DMP applications supplied with the OTX Hardware Device Driver Software Development Kit (SDK) would not necessarily need to read this document as standard DMP application can be executed directly via the OTX Hardware Application Programming Interface (API).

A prerequisite to this document is a good understanding of the hardware configuration of the host board for the Alvis-DMP (Alvis-ASM, Alvis-PCIe, or Alvis-CSI board). That knowledge can be gained by reading the Technical Description document for that specific host board.

For information on how to develop custom DSP applications for the DSP core of the DMP, please refer to “Programmer’s Guide for OTX C64x+ DSP Software Development Kit” (Odin document number 1412-1-SAA-1014-1).

For information on how to develop host applications utilizing the OTX Hardware Device Driver Application Programming Interface (API), please refer to the “Programmer’s Guide for OTX Hardware API” document (Odin TeleSystems Inc. document number 1411-1-SAA-1006-1).

And finally, for help on how to install the Alvis-ASM and Alvis-PCIe card and the OTX Device Driver Software, please refer to the Installation Guide for OTX PCI and PCIe Adapters (Odin TeleSystems Inc. document number 1512-1-HCA-1001-1).

2 Operating Systems

Each DMP on the Alvis contains two cores: one ARM9 core and one C64x+ core. The ARM9 core is the “central” core and treats the C64x+ core as a “black box”. The “black box” can run Odin supplied DSP modules (SPMs) or customized applications built using the OTX C64x+ DSP Software Development Kit. All of the DSP programs must comply with the Texas Instruments’ xDM standard.

The ARM9 core can run either MontaVista Linux as its operating system. Once the operating system is booted, drivers can be loaded. Odin supplies a number of drivers with the Alvis-DMP board. A subset of these drivers and deamons are listed below:

- OTX DMP driver (OtxDaVinci.ko) – which has an API very similar to the OTX HW Driver (for the OTX NIC cards). This driver allows an application to open logical devices in the DSP core.
- CodecEngine drivers
- OTP (Odin Transfer Protocol) daemon. This daemon runs in user mode and handles the communication between the DMP and the host driver.

An application running on the ARM9 Core can open and use module in the DSP core. Please see Figure 1 for an illustration two cores of the DaVinci DMP device on an Alvis-DMP board:

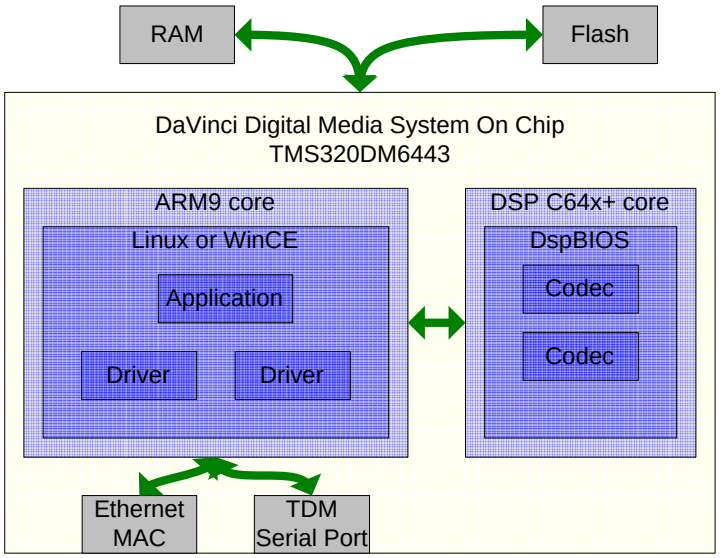


Figure 1

3 Using Alvis with MontaVista Linux

The ARM9 core of the DMP is supported by the MontaVista Linux operating system. The following chapter provides information on how to write and debug applications running on this operating system.

3.1 Third-party Software Development Kit

To compile applications for the Alvis-DMP two software development kits (SDKs) are needed:

- OTX Alvis Linux SDK
- Texas Instruments' DVSDK

The TI DVSDK can be downloaded from the TI web site or from the Odin web site. The OTX Alvis Linux SDK can be downloaded only from the Odin web site (see links below)

OTX Alvis Linux SDK. Look for the ‘Alvis Linux SDK’ link at the following URL:

http://www.odints.com/pages/support/drivers/otx/otxfslin.htm

As of this writing the SDK file (version 2.36.1.0) is located at:

http://www.odints.com/ftp/pub/drivers/Otx/OtxDrv2.36.1.0LinuxAlvisDrvSdk.tar.gz

However, the version of the latest SDK is continuously stepped so it is recommended to check the Odin web site and download the latest released version.

DVSDK Alvis Development Environment - File System Linux (508 MB):

http://www.odints.com/ftp/pub/projects/alvis/devkit/mvl_target_setuplinux_1_20_00_1.bin

DVSDK Alvis Development Environment – Setup Linux (363 MB):

http://www.odints.com/ftp/pub/projects/alvis/devkit/mvl_setuplinux_1_20_00_10.bin

Both the DVSDK files are Linux executables and contain a click-wrap software license agreement from TI/MontaVista and a compressed tar file.

The CCStudio and XDS510USB Emulator tools are not necessarily needed to write application for the ARM9 core of the DMP, but they could be helpful when designing and debugging code written for the DSP core of the DMP (for further details please refer to the OTX C64x+ DSP Software Development Kit).

3.2 Setting up the TI DVSDK

Although the DMP provides a file system on the Flash memory, it is recommended that another PC running Linux is used as the host for the DVSDK (see Chapter “Third-party Software Development Kit”). The DMP can NFS-mount a file system on the development PC, or it can be accessed via ftp transfers. The DVSDK is open-source and available under a license agreement from Texas Instruments and MontaVista.

The DVSDK can be installed on a PC running Linux, e.g. RedHat or CentOS. It does not necessary have to run MontaVista Linux.

Logon as root on the development PC.

Download the ‘mvl_setuplinux_1_20_00_10.bin’ and ‘mvl_target_setuplinux_1_20_00_1.bin’ files from the Odin web site (see links in chapter 3.1).

Make sure that these files are executable (chmod).

Execute both files, review the license agreement, and if acceptable, accept the agreement and choose ‘/opt’ as the installation directory.

```
host $ ./mvl_setuplinux_1_20_00_10.bin
host $ ./mvl_target_setuplinux_1_20_00_10.bin
```

After you execute these files, make sure the following files are located in /opt/mv_pro_4.0 (or in the /mv_pro_4.0 subdirectory of the directory you chose in place of the default):

```
mvltools4.0-no-target.tar.gz
mvl4.0-target_path.tar.gz
```

Go to the location where you will unpack the tar files. For example:

```
host $ cd /opt/mv_pro_4.0
```

Unpack the tar files (as root) by using the following commands:

```
host $ tar zxf mvltools4.0-no-target.tar.gz
host $ tar zxf mvl4.0-target_path.tar.gz
```

This creates the MontaVista directory structure under the /opt/mv_pro_4.0/montavista/ directory.

Then set the following path in .bashrc (or similar file):

```
PATH="/opt/mv_pro_4.0/montavista/pro/devkit/arm/v5t_le/bin:/opt/mv_pro_4.0
/montavista/pro/bin:/opt/mv_pro_4.0/montavista/common/bin:$PATH"
```

Log out, and log back in.

Run:

```
which arm_v5t_le-gcc
```

The response should be:

```
/opt/mv_pro_4.0/montavista/pro/devkit/arm/v5t_le/bin/arm_v5t_le-gcc
```

‘arm_v5t_le-gcc’ is the name of the compiler that is used to compile applications for the Alvis board.

3.3 Setting up the OTX Alvis Linux SDK

Create a directory on the Linux PC where you want to install the SDK (e.g. /opt/otxroot). Uncompress the SDK in this directory

```
host $ mkdir /opt/otxroot
```

Download the latest OTX Alvis Linux SDK file from the Odin web site (see links in chapter 3.1) to this directory. This is a *.tgz file.

```
host $ cd /opt/otxroot
host $ tar zxf OtxDrv2.36.1.0LinuxAlvisDrvSdk.tar.gz
```

Please note that the latest OTX SDK might have a different file name (and version) than '2.36.1.0'.

3.4 Connecting the Console

During the program development phase of DMP applications it useful to connect the console to the DMP. The console is accessed via a serial port on the Alvis host board. Please consult the Technical Description document for your specific Alvis host board (Alvis-ASM, Alvis-PCIe, or Alvis-CSI) for instructions of how to connect the serial cable.

Use a terminal emulator program (like HyperTerminal, or putty) and configure the following parameters:

- COM-port number – typically COM1 or COM2 if you connected it to a "native" COM port of your PC
- Baudrate – 115200
- Data bits – 8
- Stop bits – 1
- Parity – None

Please see example of the configuration panel from 'putty' below:

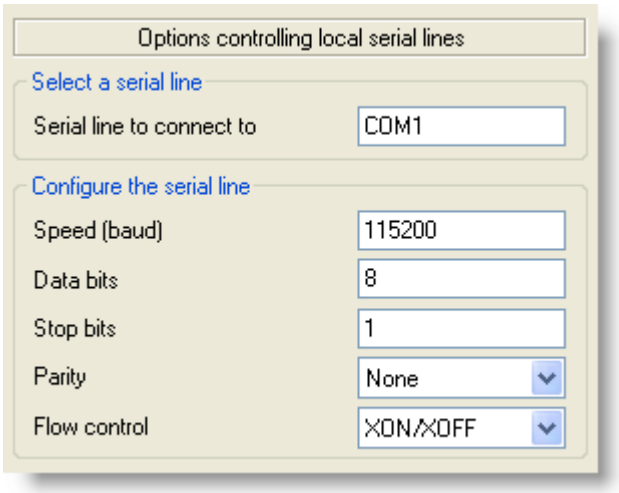


Figure 2

Once the DMP is booted (see chapter 3.5), the DMP can also be controlled from a telnet session. See example in chapter 3.8.1.

3.5 Booting the DMP

The DMP either boots directly upon power-up (in the case of Alvis-CSI) or upon the

OTX HW driver load completion in the host (as in the case of Alvis-ASM and Alvis-PCIe). The console (if attached) will display the boot messages.

First messages from Aboot (Alvis' version of Uboot) will be displayed on the console. Aboot is one of the initial boot loaders. Aboot will give the user a chance (5 seconds) to press any key and break the normal boot process. If no key is pressed within 5 seconds Aboot will run its default boot command (which is by default set to 'run bootjffs2'. The JFFS2 boot will boot Linux from the NAND flash, and mount a JFFS2 file system (also located in the NAND flash).

After the boot process completes the login prompt is displayed. Log in as 'root'. The default password is blank (no password).

The host for the Alvis-DMP has control over the boot process. E.g. in the case of Alvis-ASM and Alvis-PCIe, the host can boot the processor via an API function called OtxDmpBoot(). For more information about functions that can control the DMP, please see the OtxDmp.h header file in the OTX HW SDK.

3.6 Setting the IP address of the DMP

The default IP address is 10.0.1.2 (static address). The IP-address of the DMP is set by environment variable in Aboot (which the Alvis version of Uboot). The DMP can operate with either static or dynamic IP addresses. If you choose a dynamic IP address you must have a DHCP server configured on the same subnet as the Alvis-DMP.

The IP address configuration can be modified via the Alvis-DMP web interface, by using the uboot command in Linux, or by editing the uboot variable directly in uboot mode (which is the mode the DMP is in right after a reboot; before Linux boots).

3.6.1 Setting IP Configuration using the Web interface

To set the IP address using the Alvis Web interface first connect the Alvis-DMP unit to an Ethernet subnet with address 10.0.1.x, then start a web browser and enter the IP address (10.0.1.2) of the Alvis-DMP in the browser's address (URL) field. For more details see Chapter 4.3.

3.6.2 Setting IP Configuration using the 'uboot' command in Linux

The Alvis-DMP file systems contains a special utility called 'uboot' which can be used to set and retrieve uboot variables.

Type 'uboot' without any parameters to see a listing of available parameters.

```
root@10.0.1.2:~# uboot
Usage: uboot [opts] cmd [cmd_args]
Where opts can be:
  --file, -f file_name  -- Load uboot vars from file, default /dev/mtd0
  --yes, -y             -- Store anyway even if list is empty
Where cmd can be:
  set [{name} {value}]  -- Set var {name} to {value}
```

```
get [{name}]        -- Get var named {name}
list                -- Get list of all vars
```

To get the current IP address type:

```
root@10.0.1.2:~# uboot get ipaddr
10.0.1.2
```

To set the IP address to 192.168.1.10 type:

```
root@10.0.1.2:~# uboot set ipaddr 192.168.1.10
root@10.0.1.2:~#
```

A reboot is needed when the IP address is changed.

```
root@10.0.1.2:~# reboot
```

3.6.3 Setting IP Configuration using in u-boot mode

To change the IP-address in u-boot mode first connect the Console (see chapter 3.4) and then boot the processor (see chapter 3.5). Press any key during the first 5 seconds of the u-boot boot process. The u-boot prompt should then be displayed:

```
A-Boot >
```

Below is a list of useful u-boot commands:

- Help – provides a list of available u-boot commands
- Printenv – displays all u-boot environment variables
- Setenv – sets (or changes) a variable in the environment
- Saveenv – saves the environment in the NAND flash memory
- Boot – boots the processor using the default boot command

The IP-address of the DMP is stored in the environment variable called ‘ipaddr’. To set the IP-address to 10.0.1.150 use the following commands:

```
Setenv ipaddr 10.0.1.150
Saveenv
```

To boot the processor run the command ‘boot’.

Other environments variables that can be configured are:

- Hostip – This is the IP address of the host. It is only used if the DMP is controlled by a host (like in the case of Alvis-ASM and Alvis-PCIe). It is not used for Alvis-CSI. This IP address must be an available IP address on the same subnet as the DMP itself.
- Nfsserver – This is the IP-address of the server running NFS. It is only used if the ‘bootcmd’ variable is set to ‘run bootnfs’.
- Nfsdir – This is the directory on the NFS server to mount from. It is only used if

the ‘bootcmd’ variable is set to ‘run bootnfs’.

- Bootcmd – This variable can be set to either ‘run bootnfs’ or ‘run bootjffs2’. It selects which boot command that will be used to boot the DMP.

3.7 Structure of the an Alvis DMP application

The structure of an application targeted for the Alvis board is similar to OTX applications targeted for the host PC. The API shares many functions such as:

- OtxDrvConnectLib()
- OtxDrvOpenPhysicalDevice() – The application would first open the OTX_DEVICE_DAVINCI_NIC device.
- OtxDrvEnable
- OtxDrvWaitForSingleEvent()

3.8 Building an Application

3.8.1 Building “Hello Alvis”

To build the simplest application for Alvis you only need to add a few lines to a C file. See example below:

```
[root@portugal AlvisTest]# cat HelloAlvis.cpp

#include <stdio.h>
#include <stdlib.h>

int main()
{
  printf("Hello Alvis\n");

  return 0;
}

[root@portugal AlvisTest]# arm_v5t_le-gcc HelloAlvis.cpp
[root@portugal AlvisTest]# ls
a.out  HelloAlvis.cpp
[root@portugal AlvisTest]#
```

The output file (in this case ‘a.out’) can be transferred to the DMP using ftp. See example below:

```
[root@panama1 AlvisTest]# ftp 10.0.1.2
Connected to 10.0.1.2.
220 (none) FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.17) ready.
500 'AUTH GSSAPI': command not understood.
500 'AUTH KERBEROS_V4': command not understood.
KERBEROS_V4 rejected as an authentication type
Name (10.0.1.2:root):
```

```
331 Password required for root.
Password:
230- Linux (none) 2.6.10_mvl401-davinci_evm #60 Sun May 11 07:46:23 MSD
2008 armv5tejl GNU/Linux
230-
230- Welcome to MontaVista(R) Linux(R) Professional Edition 4.0 (0501140).
230-
230 User root logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put a.out
local: a.out remote: a.out
227 Entering Passive Mode (10,0,1,170,4,0)
150 Opening BINARY mode data connection for 'a.out'.
226 Transfer complete.
7794 bytes sent in 9.7e-05 seconds (7.8e+04 Kbytes/s)
ftp> bye
221 Goodbye.
[root@panama1 AlvisTest]#
```

You can now telnet to the DMP and execute the application. Please see example below:

```
[root@panama1 AlvisTest]# telnet 10.0.1.2
Trying 10.0.1.2...
Connected to 10.0.1.2 (10.0.1.2).
Escape character is '^]'.

MontaVista(R) Linux(R) Professional Edition 4.0 (0501140)
Linux/armv5tejl 2.6.10_mvl401-davinci_evm

(none) login: root
Last login: Thu Jan  1 00:02:24 1970 from 10.0.1.7 on pts/0
Linux (none) 2.6.10_mvl401-davinci_evm #60 Sun May 11 07:46:23 MSD 2008
armv5tejl GNU/Linux

Welcome to MontaVista(R) Linux(R) Professional Edition 4.0 (0501140).

  Welcome to the Alvis-ASM!
  You are logging to the DMP #0
  S/n is: DM1R151001028
  Some OTX DaVinci SDK Tools are installed in the /opt directory.

 Have a good day!

root@(none):~# ls
AlvisDemo      OtxDaVinci.ko  killotpd.sh  startinetd.sh
AlvisDemo.log  a.out          rmlist
root@(none):~# chmod 777 a.out
root@(none):~# ./a.out
Hello Alvis
root@(none):~#
```

3.8.2 Building an Application in the OTX DaVinci SDK

Alvis applications are built in the same way as applications are build for the OTX Linux SDK with the exception that 'DAVINCI=1' is defined on the command line.

To build an OTX DaVinci demo application with symbolic debug information run the following from the any directory level in the OTX DaVinci SDK tree:

```
make DAVINCI=1 DEBUG=1
```

If the compilation is successful, the application is placed in the following directory:

```
Linux/arm/Debug/Davinci
```

For example if the application called AlvisDemo was built the output file (the executable) could be located at:

```
/home/bob/Otx/Exe/Alvis/AlvisDemo/Linux/arm/Debug/Davinci/AlvisDemo
```

To build an OTX DaVinci demo application without symbolic debug information run:

```
make DAVINCI=1
```

To clean the debug build (delete object files and executables) run the following:

```
make DAVINCI=1 DEBUG=1 clean
```

Once the application is build it can be transferred to the DMP (via ftp or a copy to from an NFS mounted file system). The chapter 3.8.1 shows an example of how to transfer the executable to the DMP using ftp.

3.8.3 Building a CodecEngine (CE) application

TBD.

The executable files in CodecEngine are two files; one for each core:

ARM - *.x470MV
DSP - *.X64P

Custom ARM applications usually do not have an extension.

3.9 Debugging an Application

Alvis applications can be debugged through various methods. Some of these methods are listed below in the order of complexity:

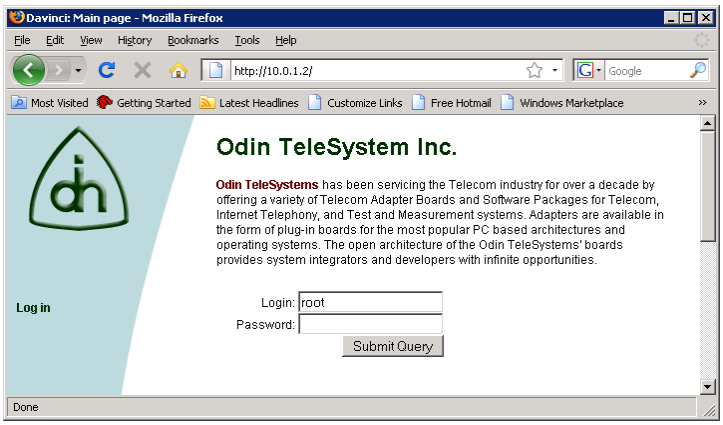
- Through printf() statements that are displayed on the console (terminal emulator like HyperTerm or putty). The same statements can also be view via a telnet session to the DMP device.
- Using the GNU debugger (gdb)
- Using MontaVista DevRocket design environment

3.10 Restoring the Alvis-DMP to the Default State

TBD.

4 Alvis Web Interface

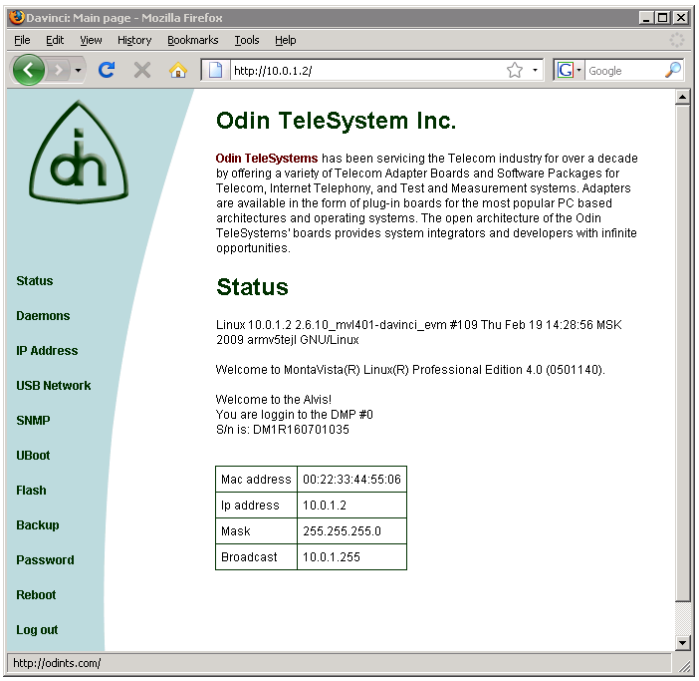
The Alvis-DMP unit can be configured through a standard web interface. To access the web interface simply type the IP address (e.g. 10.0.1.2) of the Alvis-DMP unit in any internet browser:



Enter the root password (which is blank [no password] by default).

4.1 Status

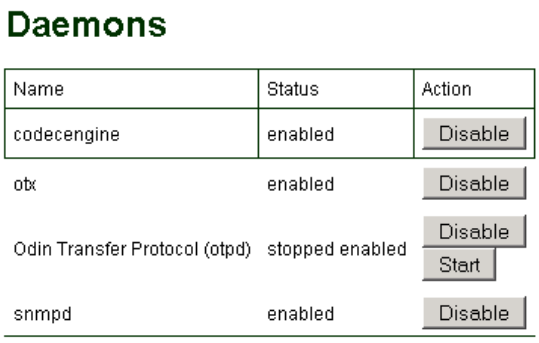
The first panel in the web interface is the Status panel. It displays the IP address, MAC address, and serial number of the Alvis-DMP unit



On the left hand side of the screen there are additional status and configuration panels.

4.2 Daemons

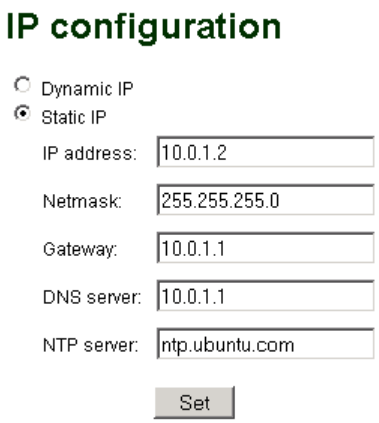
The “Daemon” panel displays which Linux daemons are available in the Alvis-DMP system and whether they are running or stopped. The daemons can be started or stopped manually from this panel.



- ‘Codecengine’ is used for ARM <-> DSP core communication
- ‘otx’ is standard OTX driver (OtxDavinci.ko) which is used for the standard OTX API.
- ‘otpd’ is used for ARM <-> Host PC communication (if there is a host PC).
- ‘snmpd’is an SNMP agent which binds to a port and awaits requests from SNMP management software. Upon receiving a request, it processes the request, collects the requested information and/or performs the requested operation and returns the information to the sender.

4.3 IP Address

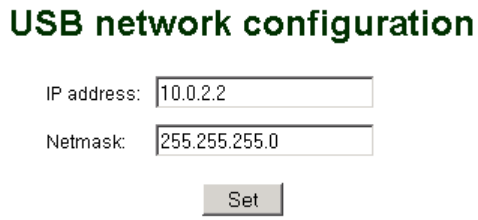
The “IP Address” panel used for the IP configuration:



Any changes to the IP configuration require a reboot of the Alvis-DMP unit.

4.4 USB Network

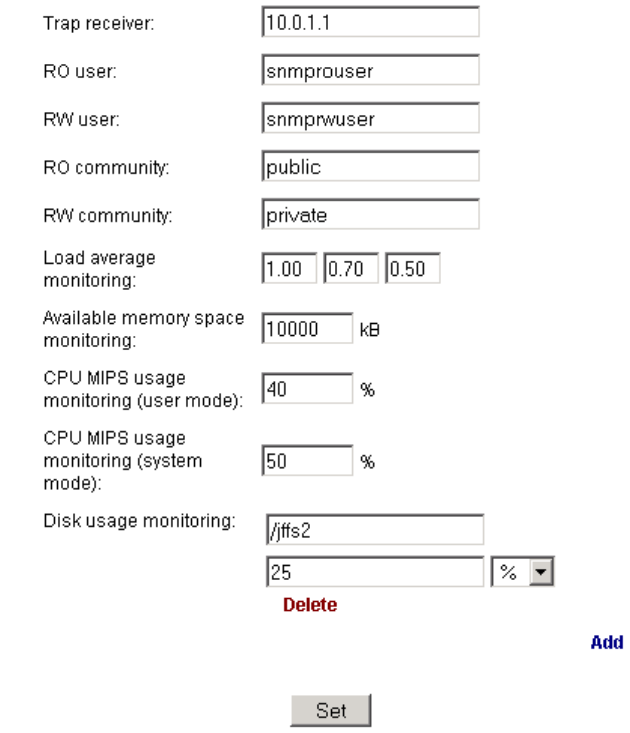
The Alvis-DMP can be configured as a USB slave and communicate with the USB host (e.g. laptop). The “USB Network” panel is used to configure the parameters for this interface.



Please see the RNDIS Chapter in the Alvis-CSI Technical Description (Doc. No. 1111-1-HCA-1020-1) for more details.

4.5 SNMP

The “SNMP” panel is used for configuring the SNMP Agent (snmpd; see chapter 4.2).



- Trap receiver: the IP address of the server where the 'trapd' daemon has been started. This would normally be a PC running an SNMP Management software package.
- RO user and RW user specifies a SNMPv3 user that will be allowed read-only (GET and GETNEXT) or read-write (GET, GETNEXT and SET) access respectively.
- RO community and RW community specify a SNMPv1 or SNMPv2c community that will be allowed read-only (GET and GETNEXT) or read-write (GET, GETNEXT and SET) access respectively.
- Load average monitoring; these three parameters are the thresholds for the 1-minute, 5-minute and 15-minute CPU load averages. If any of these loads exceed the associated maximum value, an SNMP trap will be sent. The following command can be issued from a PC (with SNMP binaries installed) to get the get the current CPU load averages:

  snmptable -v 1 -c public 10.0.1.2 laTable

  (where 10.0.1.2 is the IP address of the Alvis-DMP unit).

- Available memory space monitoring: the free memory space minimum threshold. This parameter has to be specified in kilobytes (kb). If the free memory space falls below this threshold, then an SNMP trap will be sent. To interactively check the available memory space on the Alvis-DMP unit, the following command can be issued from a PC (with SNMP binaries installed):

snmpget -v 1 -c public 10.0.1.2 memAvailReal.0

- CPU MIPS usage monitoring (user and system mode); ARM CPU MIPS usage threshold. These parameters are integer values and have to be specified as a percentage of the total number of 'ticks'. If any of these usages exceed the associated maximum value, a SNMP trap will be sent. To interactively check the CPU MIPS usage on the Alvis-DMP unit, the following commands can be issued from a PC (with SNMP binaries installed):

  snmpget -v 1 -c public 10.0.1.2 ssCpuUser.0
  snmpget -v 1 -c public 10.0.1.2 ssCpuSystem.0

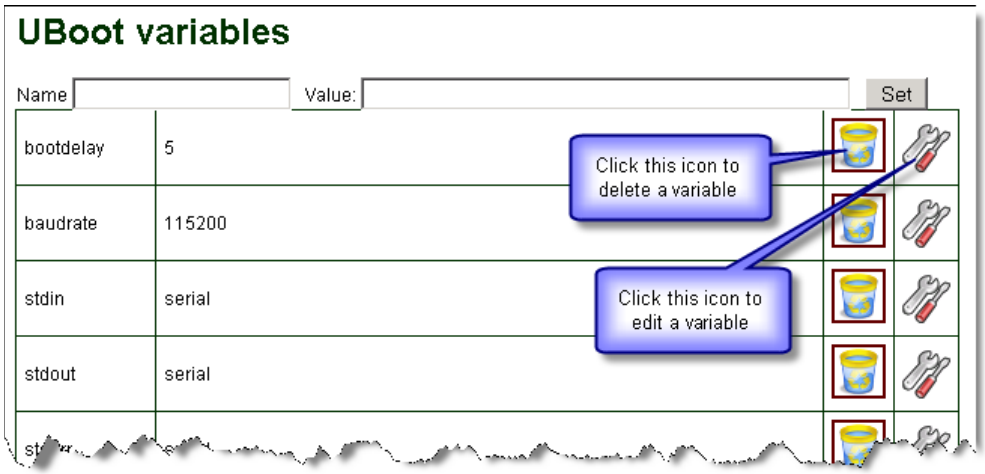
- Disk usage monitoring: monitoring of the disk mounted at the set path for available disk space. The minimum threshold can either be specified in Kb or as a percentage of the total disk (with a % character), defaulting to 100Mb if neither are specified. If the free disk space falls below this threshold, then the trap will be sent. To interactively check the disk usage on the Alvis-DMP unit, the following command can be issued from a PC (with SNMP binaries installed):

  snmptable -v 1 -c public 10.0.1.2 dskTable

If the threshold/limit of a monitored condition is exceeded an event will trigger, which results in a trap being sent to the trap receiver. Note that the event will only be triggered once, when the condition first matches. This monitor will not fire again until the monitored condition first becomes false, and then matches again.

4.6 Uboot Variables

The “Uboot Variables” panel displays the variables used by the Alvis-DMP system. This is for advanced users only. Variables can be edited, deleted, or added.



Please use caution when editing variables. An incorrect value could cause your system to become defunct.

5 Using Alvis with Windows CE

Support of the Windows CE environment for Alvis is not available at this time.

6 Reference documents

The following documents provide further detailed information related to the Alvis-DMP board:

- Alvis-ASM Technical Description (Odin document # 1111-1-HCA-1018-1)
- Alvis-CSI Technical Description (Odin document # 1111-1-HCA-1020-1)
- Alvis-PCIe Technical Description (Odin document # 1111-1-HCA-1021-1)
- Programmer's Guide for OTX C64x+ DSP Software Development Kit (Odin document number 1412-1-SAA-1014-1)
- Programmer's Guide for OTX Hardware Driver (Odin document # 1412-1-SAA-1006-1)
- Installation Guide for OTX PCI Adapters (Odin document number 1512-1-HCA-1001-1)
- Alvis-ASM Product Brief (Odin document number 2020-1-HCA-1018-1)
- Alvis-CSI Product Brief (Odin document number 2020-1-HCA-1020-1)
- Alvis-PCIe Product Brief (Odin document number 2020-1-HCA-1021-1)

7 Glossary

- OTX – Odin Telecom FrameworX
- Alvis – From the Norse mythology - The dwarf Alvis wanted to marry Thrud (daughter of Thor) but Thor tricked him into being above ground when the sun came up, turning him into stone. Alvis is also a product family based on Texas Instruments DaVinci processor.
- TI – Texas Instruments – Manufacturer of the DaVinci Digital Media Processor (DMP).
- DevRocket – A development suite offered by MontaVista for MontaVista Linux.
- ASM – Application Specific Module (daughter card populated on OTX NIC cards).
- NIC – Network Interface Card. Refers to the OTX Base board that this ASM board is connected to. Examples of OTX NIC cards are Thor-8-PCI-Plus, Thor-2-PCI-Plus, and Thor-2-PCI-Express.
- DMP – Digital Media Processor. Refers to the dual core System On Chip processors on the Alvis board. Also referred to as an Alvis-DMP.
- DSP – Digital Signal Processor
- SDK – Software Development Kit
- API – Application Programmer Interface
- CPU – Central Processing Unit. Refers to the host PC in this document.
- EEPROM – Electrically Erasable Programmable Read Only Memory.
- FPGA – Field Programmable Gate Array.
- LED – Light Emitting Diode

- LS – Least Significant
- MS – Most Significant