MIPS32 Malta Linux

Imperas Software Limited
Imperas Buildings, North Weston,
Thame, Oxfordshire, OX9 2HA, UK
docs@imperas.com

Author:	OVP
Version:	1.4.1
Filename:	OVP_MIPS_Linux_Platform_User_Guide.doc
Project:	MIPS32 Malta Linux Platform
Last Saved:	December 7, 2016
Keywords:	OVP MIPS Malta Linux

Copyright Notice

Copyright © 2016 Imperas Software Limited All rights reserved. This software and documentation contain information that is the property of Imperas Software Limited. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Imperas Software Limited, or as expressly provided by the license agreement.

Right to Copy Documentation

The license agreement with Imperas permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

IMPERAS SOFTWARE LIMITED, AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Table of Contents

1 Preface 7
1.1 Related Documents 7
2 Introduction 8
2.1 Platform Definition Files 8
2.2 Demo Package Installers 9
3 Virtual Platform Command Line and Standard Usage 10
4 Board Architecture 11
5 Malta Peripheral Models Overview 13
5.1 SmartLoaderLinux 13
5.2 GT64120 (SysGT6412x) 14
5.3 Uart16450 14
5.4 PIIX4E (PciPIIX4Ebase) 14
5.5 IDE (PciIDE) 14
5.6 USB (PciUSB) 15
5.7 Power management (PciPM) 15
5.8 Interrupt Controller (IntPIIX4E8259) 15
5.9 VGA (VgaCLGD54xx) 16
5.10 PS2 Controller (Ps2Control) 16
5.11 Real Time Clock (RtcMC146818) 16
5.12 Malta FPGA (MaltaFPGA) 17
5.13 Ethernet Controller (NicAM79C97x) 17
5.14 System bus, PCI bus, ISA bus 17
6 Booting the pre-built Linux Kernel 19
6.1 Introduction 19
6.2 Linux Kernel Versions and SMP Support 19
6.3 Execution Sequence 19
6.3.1 Memory Initialization 19
6.3.2 Execution 20
6.4 Running the Platform 21
6.4.1 Booting from ‘RamDisk’ Initial RAM Disk 21
6.4.2 Terminating the Simulation 24
7 Installing Full Linux Distribution 26
7.1 Simulation Invocation Script 26
7.2 Creating an Empty Disk Image 26
7.3 Starting the Linux Kernel Boot 27
7.4 Installing a Linux Distribution 28
7.4.1 Configure Local 28
7.4.1.1 Select Language 28
7.4.1.2 Select location 29
7.4.1.3 Select Keyboard Layout 29
7.4.2 Configure the Network 30
7.4.2.1 Select Host Name 30
7.4.2.2 Select Domain Name 30
7.4.3 Download Debian Components 31
7.4.3.1 Select Debian Archive Mirror Country 31

7.4.3.2 Select Debian Archive Mirror 31
7.4.3.3 Choose a Proxy 32
7.4.4 Download Installer Components 32
7.4.4.1 Continue Without Kernel Modules 32
7.4.5 Partition Disk 33
7.4.5.1 Start Partitioning 33
7.4.5.2 Guided Partitioning of Entire Disk 33
7.4.5.3 Create IDE Master 34
7.4.5.4 Choose Partitioning 34
7.4.5.5 Finish Partitioning of Disk 35
7.4.5.6 Write Changes to Disk 35
7.4.6 Setup Accounts 36
7.4.6.1 Root Password 36
7.4.6.2 Add User Account 37
7.4.7 Base System Installation 39
7.4.7.1 Verifying the Release 39
7.4.7.2 Retrieving Packages 39
7.4.7.3 Continue Without Kernel 40
7.4.7.4 Package Usage Survey 40
7.4.7.5 System Selection 41
7.4.7.6 No Boot Loader 41
7.4.8 Complete Installation 42
7.5 Re-Start System 43
7.5.1 Boot directly from Disk Image 43
7.5.2 Disk Check 44
7.5.3 Re-Boot from Disk Image 45
7.5.4 Login 45
7.6 Stopping the System 46
8 Disk Image Files and Initial RAM Disks 47
8.1 Introduction 47
8.2 Hard Disk Image 47
8.2.1 Creation 47
8.2.2 Accessing 48
8.3 Initial RAM Disk 48
8.3.1 Extract Files from an initrd.gz 48
8.3.2 Building an initrd.gz 48
8.3.3 Modifying the boot sequence 49
8.3.4 Example Operations When Running from Initial RAM Disk 49
8.3.4.1 Make a writeable directory 49
8.3.4.2 Configure the NIC 49
9 Re-building the Debian Linux Kernel 51
9.1 Introduction 51
9.2 Building on a Linux Platform 51
9.2.1 Building the Cross-Compiler 51
9.2.2 Building the Linux Kernel 53
9.2.2.1 Kernel Build Script 53
9.2.2.2 Kernel Build Configuration 54

9.3 Building on a Windows Platform 54
10 Debugging the Linux Kernel 55
10.1 Non-Intrusive Instrumentation 55
10.2 Attaching the Debugger 55
10.2.1 Starting the Platform Simulation 55
11 Transferring Files to Guest Linux 57
11.1 Introduction 57
11.2 Using an FTP Server 57
11.2.1 Starting the Simulation 57
11.2.2 Adding Additional Linux Components 58
11.2.3 Transfer Files 58
11.3 Using TFTP 59
11.3.1 Starting the Simulation 60
11.3.2 Adding Additional Linux Components 60
11.3.3 Transfer Files into Guest Operating System 61
12 Debugging Linux User Applications 63
12.1 Introduction 63
12.2 Linux Installation 63
12.3 Booting Linux 63
12.3.1 Redirecting TCP Port 63
12.3.2 Enabling TFTP File Transfer 64
12.4 Compiling a User Program 64
12.4.1 Cross Compiler ToolChain 64
12.4.2 Cross Compiler Make Environment 64
12.4.3 Building the Application 65
12.5 Adding Additional Linux Components 65
12.6 Remote Debug using GDBserver 66
12.6.1 Starting GDB Server 66
12.6.2 Connecting to GDB Server 67
12.6.2.1 Using Eclipse 67
12.6.2.2 Using DDD from a Linux Host 67
12.6.2.3 Using GDB from local Windows Host 68
12.6.3 Debugging Multiple Applications in Parallel 70
12.7 Debug using Simulated GDB 71
13 Creating an Example Linux Kernel Module 73
13.1 Introduction 73
13.2 Pre-Requisites 73
13.3 Starting the Platform Simulation 73
13.4 Example Source 74
13.4.1 Included Header Files 74
13.4.2 Callback Table and Functions 74
13.4.2.1 Table 74
13.4.2.2 Device Open 74
13.4.2.3 Device Close 74
13.4.2.4 Device Read 75
13.4.2.5 Device Write 75
13.4.3 Device Module Initialize and Exit 75

13.4.3.1 Device Initialization 75
13.4.3.2 Device Exit 76
13.4.3.3 Register Module 76
13.5 Building the Device Driver 76
13.5.1 Makefiles 76
13.5.2 Building 77
13.6 Installing the Kernel Module 77
13.7 Kernel Module Test Application 78
14 Debugging a Statically Linked Linux Kernel Module 80
15 Debugging a Dynamically loaded Linux Kernel Module 81
15.1 Pre-Requisites 81
15.2 Starting the Platform Simulation 81
15.2.1 Invoking with VLNV Library 81
15.2.2 Invoking from ICM Platform 82
15.2.3 Debug Startup 83
15.3 Registering Kernel Module with Debugger 83
15.4 Debugging the Kernel Module 84
15.4.1 Register Kernel Module for Debugging 84
15.4.2 Installing the Kernel Module 84
15.4.3 Adding Breakpoints 85
15.4.4 Source Level Debug 85
16 Restrictions and Caveats 86
APPENDIX A 87
A.1 Platform Command Line Arguments 87
APPENDIX B 88
B.1 Some Terms Explained 88
APPENDIX C 89
C.1 MIPS Kernel Configuration File 89

1 Preface

This document describes the simulation model of a platform capable of running a release of Linux cross compiled for the MIPS 32 bit architecture.

It provides a guide to running a Debian Linux Operating System release on the virtual platform.

1.1 Related Documents

This platform model uses peripherals from the OVP library. Consult the OVP library at www.ovpworld.org for documentation for the status on individual models. The device vendors should be consulted for individual device datasheets.

The processor cores are developed using the MIPS32 reference material available in general from http://www.mips.com/products/product-materials/processor in particular the documents:

- MIPS32® Architecture For Programmers Volume I: Introduction to the MIPS32® Architecture
- MIPS32® Architecture For ProgrammersVolume II: The MIPS32® Instruction Set
- MIPS32® Architecture For Programmers Volume III: The MIPS32® Privileged Resource Architecture
- Malta™ User's Manual

2 Introduction

The MIPS Linux platform, based on the MIPS Malta™ development board, contains models of hardware components to a sufficient level of accuracy to run the Linux operating system. The Debian distribution of Linux is provided as an example kernel with other files that are used in conjunction with the platform to show Linux booting.

The platform can make use of any MIPS32 processor model that provides the MMU requirements of Linux. The default processor model configuration provided in this platform is the 4KEc device. The MIPS32 4KEc is a 32-bit CPU with a Memory Management Unit with user- and protected-mode instructions to support a Virtual Memory Management System. Other 32-bit MIPS32 cores can also be specified, these include the 24Kc, 24KEc and 34Kc (which is shown booting SMP Linux). To see all the cores available look at the MIPS32 processor page in the library documentation of an OVP installation.

The MIPS32 models available from OVP and used in the Malta platform have been verified as part of the **MIPS-Verified™** program.

As well as running the provided Linux kernel the platform is intended to allow the ease of development of the Linux kernel, Linux user applications, Linux loadable kernel modules, additional platform components and porting other Linux distributions.

OVP provides base intercept technology allowing the 'semihosting'[1] of native host features.

Imperas provides additional intercept technology, and modules using it to provide assistance with Linux on the Malta platform. Imperas also provides multiprocessor and platform debug capabilities.

2.1 Platform Definition Files

The platform is defined and downloaded as part of the standard OVPsim download. This is available in source form from the download section of the website www.ovpworld.org.

When installed the platform source is found in

IMPERAS_HOME\ImperasLib\source\mips.ovpworld.org\platform\MipsMalta\1.0\platform

And the OVPsim executable for the platform in

IMPERAS_HOME\lib\IMPERAS_ARCH\ImperasLib\mips.ovpworld.org\platform\MipsMalta\1.0

[1] Semihosting is a specific form of interception that is applied to low level functions in the C library being used. These function such as open, read, write etc are intercepted to give Host native operation using the Host file system.

The platform is described on the OVPworld website at

http://www.ovpworld.org/library/wikka.php?wakka=Mips32MaltaLinux

2.2 Demo Package Installers

The examples described in this document are available from the OVPworld website. They are described at

http://www.ovpworld.org/library/wikka.php?wakka=Mips32MaltaLinuxBootingInstructions

From this page you will be able to download all the files you need to run the examples described.

There are two installations:

1. OVPsim_demo_linux_MipsMalta
2. OVPsim_demo_linux_MipsMalta_install

When installed new directories will appear in

IMPERAS_HOME\Demo\OVPsim_linux_MipsMalta

and

IMPERAS_HOME\Demo\OVPsim_linux_MipsMalta_install

These directories include copies of the platform source and executable, the Linux kernel 'vmlinux', a pre-built ramdisk 'initrd.gz' and scripts to execute.

3 Virtual Platform Command Line and Standard Usage

The MipsMalta virtual platform includes a standard command line parser that supports all the arguments supported by the simulator and also a number of platform specific arguments that are detailed below.

Linux Kernel specific Arguments

Argument	Value	Description
kernel	elf file	specify the linux kernel elf image to load
ramdisk	Initial ramdisk	Specify the
console	Linux console	Specify the console to be used, for example tty0, ttyS0
disk	File	
root		
pagebits		

General Arguments

finishonhalt
redir
nographics
tftproot
bootimage

4 Board Architecture

The platform is illustrated by the following block diagram. It is architected to closely represent the MIPS Malta hardware in so far that

1. the binary image can be interchanged between the virtual platform and real silicon;
2. the software performs the same function on the virtual platform as it will on real silicon.

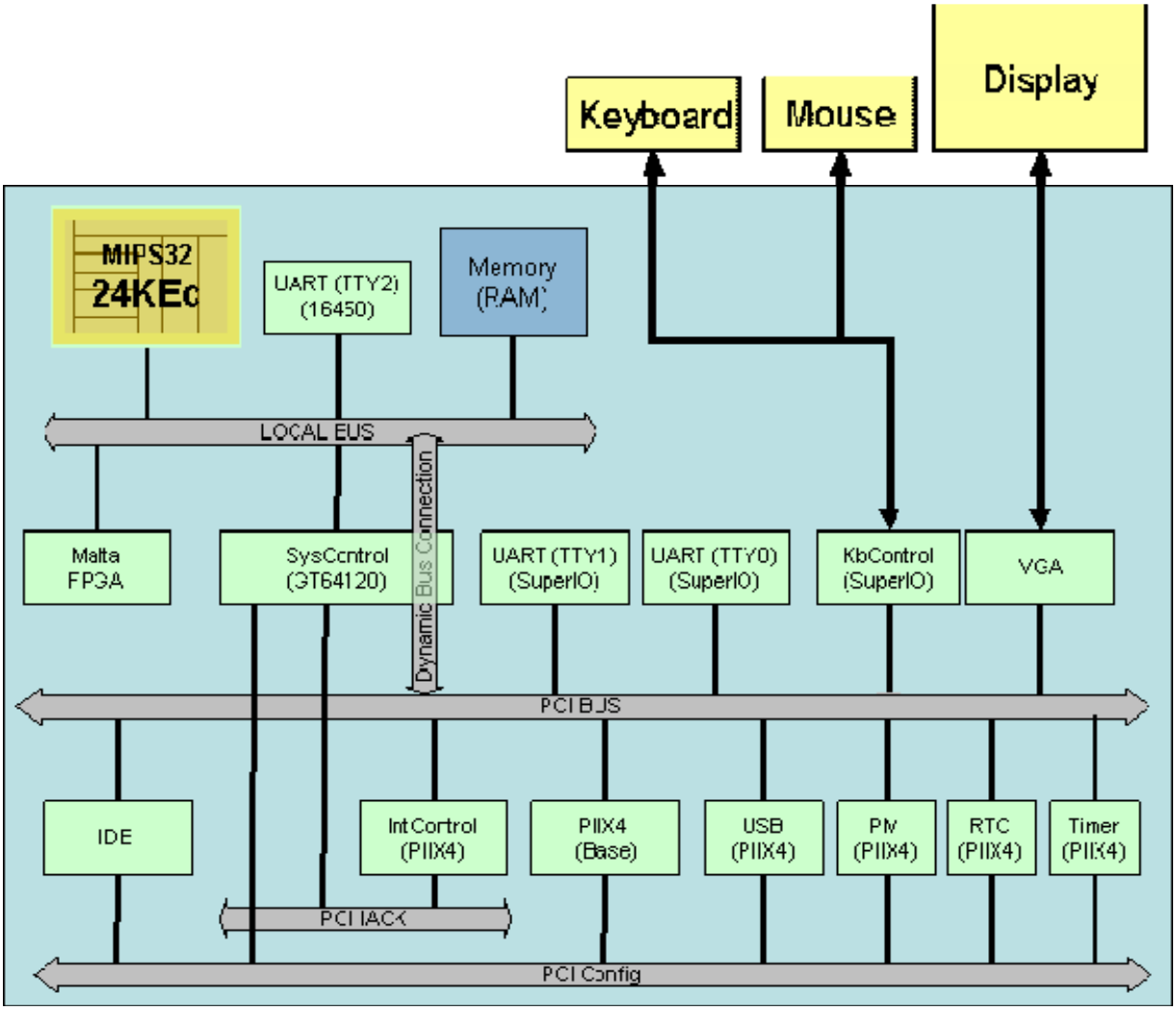


Figure 1 Platform Block Diagram

The CPU is connected to a local bus and a GT64120 general purpose system controller which supports SDRAM, FLASH and provides a bridge to an on-board PCI bus.

The PCI bus services a 4-channel IDE disk controller, a USB interface, AM79C973 Ethernet interface and a Cirrus CL GD5446 VGA graphics controller.

A bridge to an ISA bus supports a PS2 keyboard, mouse interface and two 16450 UARTs.

An FPGA specific to the MIPS Malta™ board controls an LED display, reads a set of switches and implements a third 16450 UART.

Two cascaded 8259 interrupt controllers, as part of the Intel PXII4E device, route interrupts from most devices to the MIPS32 int0 input.

The following diagram provides the general memory map of the platform. Some memory areas, such as the PCI memory and PCI IO regions contain a number of mapped devices.

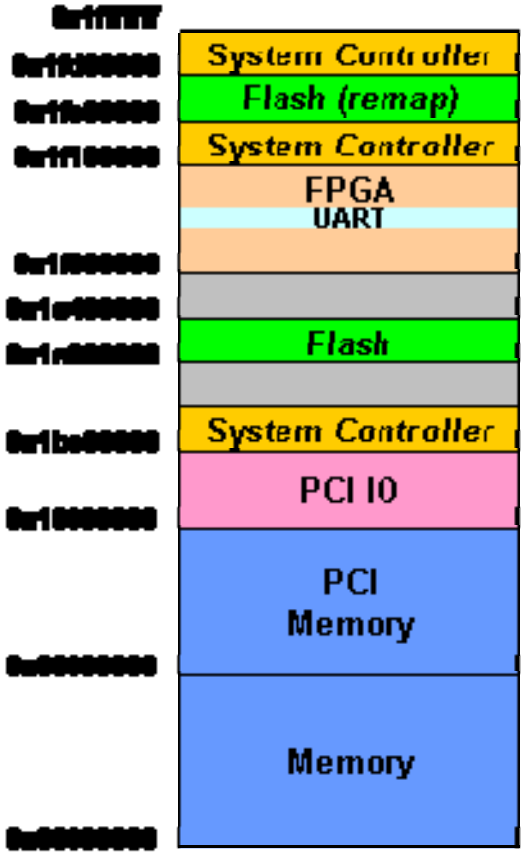


Figure 2 Basic Platform Memory Map

5 Malta Peripheral Models Overview

This section is intended to provide a brief overview of the operation of the peripherals that can be configured and details of the attributes that may be used within the virtual MipsMalta platform to change behavior. For complete descriptions of the peripherals the OVP documentation should be consulted.

5.1 SmartLoaderLinux

The Linux SmartLoader is created as a peripheral but is used in the simulation to initialize the platform. It is capable of providing all configuration information required by the Linux kernel. This includes the kernel command line being created and written into memory, and boot options; that is boot from a hard disk image or boot from an initial ramdisk.

Attribute	Platform	Model Default	Description
kernel	vmlinux	None	specify the linux kernel being loaded (this must be specified)
envpaddress	None	0x80002000	change the address to which the kernel command lines parameters are based
initrd	initrd.gz [2]	None	specify the initial ram disk that should be loaded for booting
root	/dev/hda [3]	None	specify booting from the hard disk image eg /dev/hda1
boardid	None	0x420	Malta platform with CoreLV
memsize	None	128Mbytes	Set memory available to kernel
command	None	None	Allows additional kernel command line parameters to be added
nonelinux	None	None	Allows a none Linux kernel to be loaded and executed on the Malta platform.
bootimage	None	None	Replace the default reset vector code to change the manner in which the Malta system boots.

Table 1: SmartLoader Attributes

The SmartLoader will read the Linux kernel to obtain the highest load address and the *kernel_entry* symbol. The *kernel_entry* symbol is used to update the jump address in the boot loader and the highest kernel load address is masked with the TLB page mask for the load address of the initial ram disk file i.e. in the next page above the kernel.

Alternative boot code may be loaded by specifying the bootimage attribute. This should specify a **.fl** file, as loaded via the parallel port on the actual hardware.

[2] Specify the argument ‘initrd’ on the command line when invoking the Malta platform to select booting from the initial ram disk initrd.gz. Without this argument the platform will boot from the file system on the disk specified by Drive0Name on the IDE model.

5.2 GT64120 (SysGT6412x)

This is a system controller providing an interface between the local bus and SDRAM and PCI buses. The model provides the PCI interface; including PCI configuration bus and PCI Interrupt acknowledge bus connections and access cycles. The model does not provide the SDRAM memory interface; this is provided as discrete memory in the platform.

Attribute	Platform	Model Default	Description
There are no user definable attributes for this model			

Table 2: GT64120 Attributes

5.3 Uart16450

This is a simple UART model for the 16450 devices. In this platform it can also be used to maintain a log of its output as well as providing it to stdout.

Attribute	Platform	Model Default	Description
outfile	None	None	Log the characters to the log file specified by this attribute.
infile	None	None	This file is used as input for the UART
registeralign	None	1	Allows the 8bit UART registers to be aligned on non-byte boundaries
log	None	None	The UART output is logged into the simulator stdout. There is no UART input.
portnum	None	None	Port number allows the connection to an external application

Table 3: UART16450 Attributes

5.4 PIIX4E (PciPIIX4Ebase)

The PIIX4E is modeled using a number of discrete component models such that the PIIX4E model itself contains only the base functionality that is common, for example the PCI configuration registers.

Attribute	Platform	Model Default	Description
There are no user definable attributes for this model			

Table 4: PIIX4E Attributes

5.5 IDE (PciIDE)

This is a simple IDE model, which resides on the PCI bus, allowing the connection of up to 4 devices. Each device is separately created as an image of the disk file structure.
The model will attempt to open 3 drives, as specified below; if any device cannot be opened a warning will be generated.

If one of the devices contains the file system for booting the SmartProm ‘root’ attribute should be specified accordingly.

Attribute	Platform	Model Default	Description
PCIslot	10	None	Set the PCI Slot for the
PCIFunction	1	None	Set the PCI Function number
Drive0Name	mipsel_hda	None	The name of the file for the first drive.
Drive1Name	mipsel_hdb	None	The name of the file for the second drive.
Drive2Name	mipsel_cd	None	The name of the file for the third drive.
Drive3Name	None	None	The name of the file for the fourth drive.

Table 5: IDE Attributes

5.6 USB (PciUSB)

The USB model does not provide USB functionality. This model provides the PCI interface necessary to register the device and provides NULL data when any register is read.

Attribute	Platform	Model Default	Description
PCIslot	10	None	Set the PCI Slot for the
PCIFunction	2	None	Set the PCI Function number

Table 6: PIIX4 USB Attributes

5.7 Power management (PciPM)

The power management model does not provide power management functionality. This model provides the PCI interface necessary to register the device and provides NULL data when any register is read.

Attribute	Platform	Model Default	Description
PCIslot	10	None	Set the PCI Slot for the
PCIFunction	3	None	Set the PCI Function number

Table 7: PIIX4 Power Management Attributes

5.8 Interrupt Controller (IntPIIX4E8259)

The interrupt controller used in the platform is contained within the PIIX4E device but is modeled as a discrete device. It comprises of two individual Intel 8259 interrupt controller models that are cascaded.
A single interrupt is generated to MIPS32 processor, which in turn causes a PCI Interrupt Acknowledge cycle that provides a vector indicating the interrupting device with the highest priority.

Attribute	Platform	Model Default	Description

There are no user definable attributes for this model

Table 8: Interrupt Controller Attributes

5.9 VGA (VgaCLGD54xx)

This model creates a Cirrus CL GD 54xx and utilizes SDL to create an output window to which the VGA is generated.

Attribute	Platform	Model Default	Description
PCIslot	18	None	Set the PCI Slot for the
scanDelay	50000	200	This is the rate at which the frame data is displayed (frame rate)
noGraphics	0	0	When set the SDL interface is disabled and no display is created.
Title	"OVPsim MIPS32 Malta"	"CL GD54xx"	This is the title displayed on the graphic screen

Table 9: VGA Attributes

5.10 PS2 Controller (Ps2Control)

The PS2 Controller uses SDL for mouse and keyboard input. It must be used in conjunction with the VGA peripheral device that generates the SDL output window with which it interacts.

The SDL interface supports polling only and so the PS2 Controller polls the SDL interface for activity on the mouse or keyboard. The poll rate is configurable.

Attribute	Platform	Model Default	Description
disableInput	0	0	When set this causes the input form the PS2 Controller to be disabled. This is used mainly for testing.
pollPeriod	50000	2000	The value provided for the poll period is the rate at which the PS2 input devices are polled for input. The value is in micro seconds.
grabDisable	0	0	Stop the mouse being grabbed when clicked in window

Table 10: PS2 Controller Attributes

5.11 Real Time Clock (RtcMC146818)

This is a model of an MC146818 real time clock device.

Attribute	Platform	Model Default	Description
timefromhost	0	0	When set this causes the current time to be read from the host during initialization.

Table 11: Real Time Clock Attributes

5.12 Malta FPGA (MaltaFPGA)

This is a model of the FPGA device that is available on the Malta platform.

Attribute	Platform	Model Default	Description
stoponsoftreset	1	0	If set when the soft reset register is written with a special value it causes the simulation to terminate.
switches	None	0	Set the value of the Malta 'switches' accessed by the FPGA

Table 12: Malta FPGA Attributes

5.13 Ethernet Controller (NicAM79C97x)

The Ethernet controller is a partial model of the AMD AM79C97xx series device.
It has sufficient capabilities to support a Linux operating system, specifically the Debian distribution of Linux 2.6.
In addition, it models a virtual Ethernet with a restricted bridge to the real Ethernet of the host (using the publicly available *SLIRP* package).
Specifically, the bridge emulates a DHCP server which allocates the NIC an IP address of 10.0.2.15 and performs Network Address Translation onto the host network of outgoing and incoming TCP and UDP packets. ICMP packets are blocked (so for example, *ping* does not work) and tftp requests are all handled in the model by a simple tftp server. Static address translations for can be set up for incoming requests on particular ports using the *redir* argument to the model.

Attribute	Platform	Model Default	Description
PCISlot	11	None	Define the PCI slot in which the Ethernet card is installed
PCIfunction	0	None	Define the PCI function of the device
ethereal		0	Packet logging using ethereal format
polldelay	1000	1000	The rate at which the network is polled
MAC	None	52:54:00:12:34:56	Define the MAC address of the NIC
redir	None	None	Specify and open ports on the simulated system. Example usage: 1. exposing gdb-server ports for attaching debuggers to user applications or 2. exposing a port for an ftp connection
tftpPrefix	None	None	Enable TFTP and define the tftp root on the host
localNet	10.0.2.0		Change the local network address

Table 13: Ethernet Controller Attributes

5.14 System bus, PCI bus, ISA bus

The Malta™ platform bridges a limited address range from the System bus to PCI bus and bridges a different range from PCI bus to the System bus. This allows the CPU to access part of the PCI address range and for PCI bus masters to DMA data directly into the main memory.

OVPsim models address spaces rather than physical busses. Therefore there is no bridge from PCI to system bus; all devices share the same address space. When a PCI device is programmed

to decode a particular address range, it adjusts the range according to the base address of the bridge. The net effect is the same, but the model can allow mappings which are not possible in the hardware.

Similarly, the ISA bus is not modeled explicitly; its devices appear on the PCI bus according to PCI configuration.

6 Booting the pre-built Linux Kernel

6.1 Introduction

The MipsMalta platform is designed to boot a Linux kernel from a number of sources. The OVPsim_demo_linux_MipsMalta demo provides everything necessary to boot a pre-built Linux kernel using a simple ram disk (also provided)

The Debian linux kernel is built for the Malta platform and the MIPS32 little endian device using a distribution available from Debian and can be found in the file named *vmlinux.* This image is built as a standard image and the same file runs on the virtual platform and the actual MIPS silicon device with real hardware.

Linux can boot from an initial ram disk file in a compressed form and extracted into simulated memory. This file is called *initrd.gz* and is provided as part of OVPsim_demo_MipsMalta. It provides a basic set of commands using the BusyBox system with a minimal file system. As downloaded the initial ramdisk will boot to a shell prompt.

6.2 Linux Kernel Versions and SMP Support

This example is provided with a Debian distribution of the 2.6.23.11 Linux kernel.

The same platform has also been used with a Debian distribution of the 2.6.24 Linux kernel and a MIPS Technologies Incorporated distribution of the 2.6.29.4-1 Linux Kernel.

The MIPS32 processor model can support the MIPS single core processors, for example 4Kc and 24KEc and also the MP core processors, for example the 34Kc. The Linux kernel is built with SMP support enabled. Thus if an MP core supporting SMP is selected as the processor variant Linux will boot as SMP.

6.3 Execution Sequence

There is a specific sequence of events carried out by the OVP simulator and the ‘SmartLoader’ peripheral on startup.

6.3.1 Memory Initialization

The following memory initialization is performed:

- Debian Linux kernel image (vmlinux) is loaded directly into memory using addressing contained in the elf file by the OVP simulator.
- Bootstrap code is created at the MIPS processor reset vector by the “SmartLoader” peripheral component.
- A kernel command line is created in memory by the “SmartLoader” peripheral component.

6.3.2 Execution

Execution starts from the MIPS processor reset vector at virtual address 0xbfc00000.
The generated bootstrap code resides at this address and performs some basic setup before a call into the Linux kernel, at the kernel_entry point.

6.4 Running the Platform

6.4.1 Booting from 'RamDisk' Initial RAM Disk

To run this demo obtain and install the OVPsim_demo_linux_MipsMalta installer for your platform from www.ovpworld.org. Go to the directory:

IMPERAS_HOME/Demo/OVPsim_linux_MipsMalta

and launch the simulation with the batch file 'RUN_MipsMalta.bat' on Windows or the sh script 'RUN_MipsMalta.sh' on Linux.

This will execute the platform executable passing in the Linux kernel as the first parameter and initrd as an additional parameter to switch the boot mode to initial ram disk.

The supplied initial ramdisk boots to a shell prompt.

The execution of the platform will commence until the shell is reached, as shown in Figure 3: Booting to Shell from Initial Ram Disk

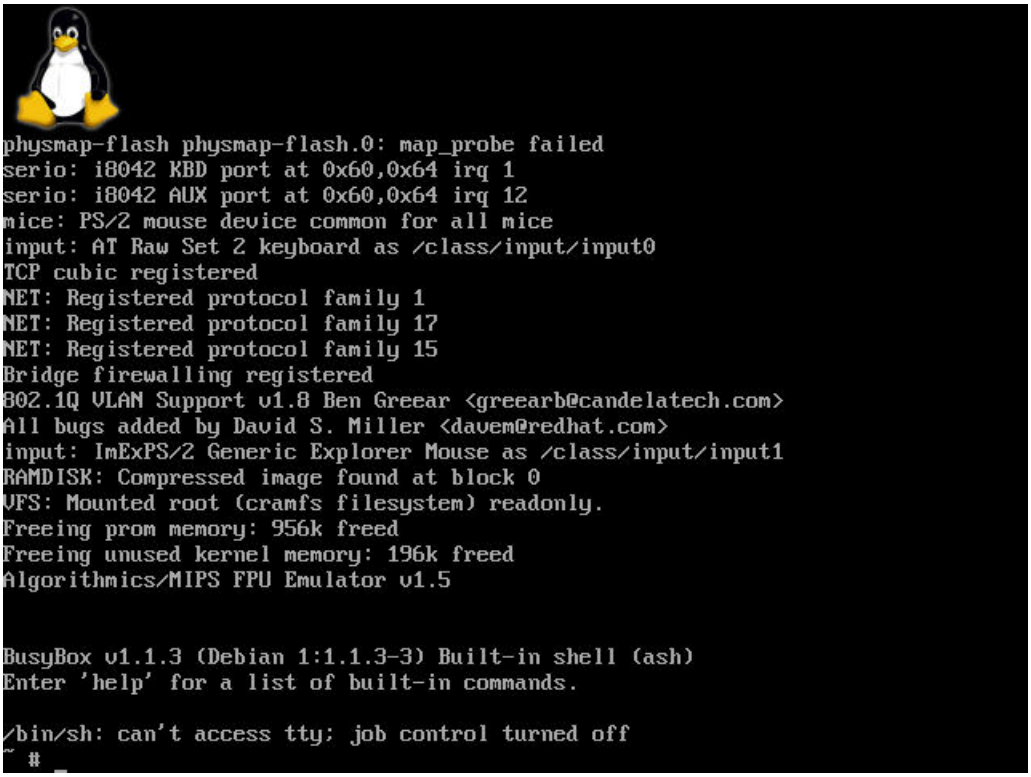


Figure 3: Booting to Shell from Initial Ram Disk

From the shell prompt we can interrogate the ***proc*** file system, as shown in Figure 4: Shell Prompt, just as we would do running on the hardware platform.

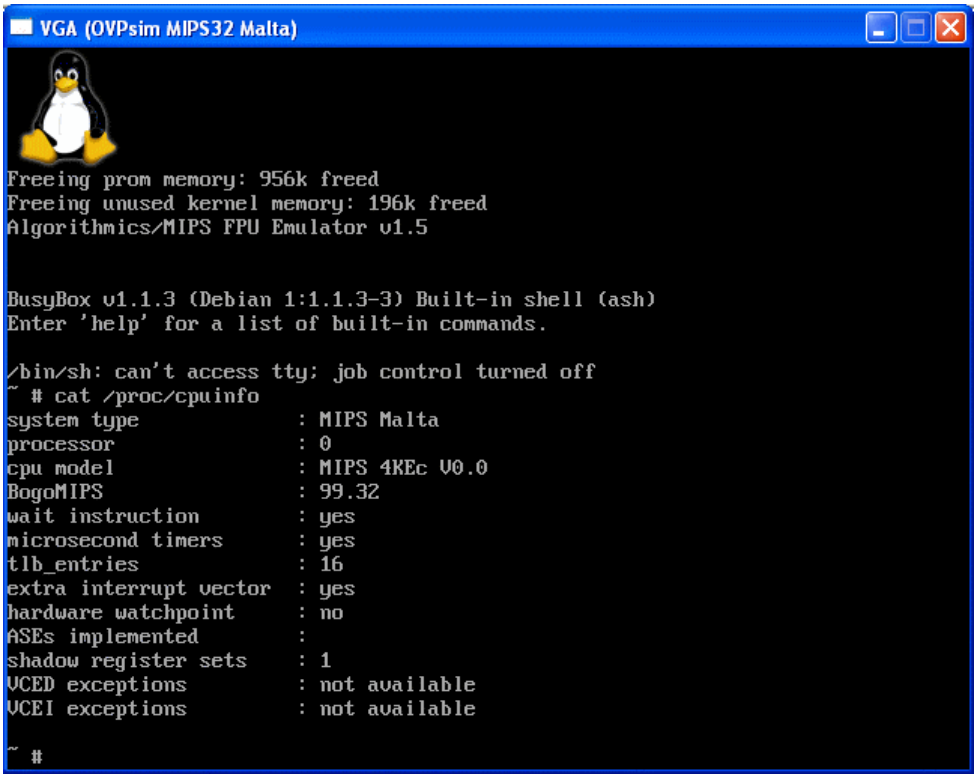


Figure 4: Shell Prompt

The simulation may also be launched with the batch file ‘RUN_MipsMaltaSMP.bat’ on Windows or using the sh script ‘RUN_MipsMaltaSMP.sh' on Linux. This will select the 34Kc MIPS32 processor variant that supports SMP.

The execution of the platform will commence until the shell is reached, as shown in Figure 5: Booting to Shell from Initial Ram Disk (SMP)

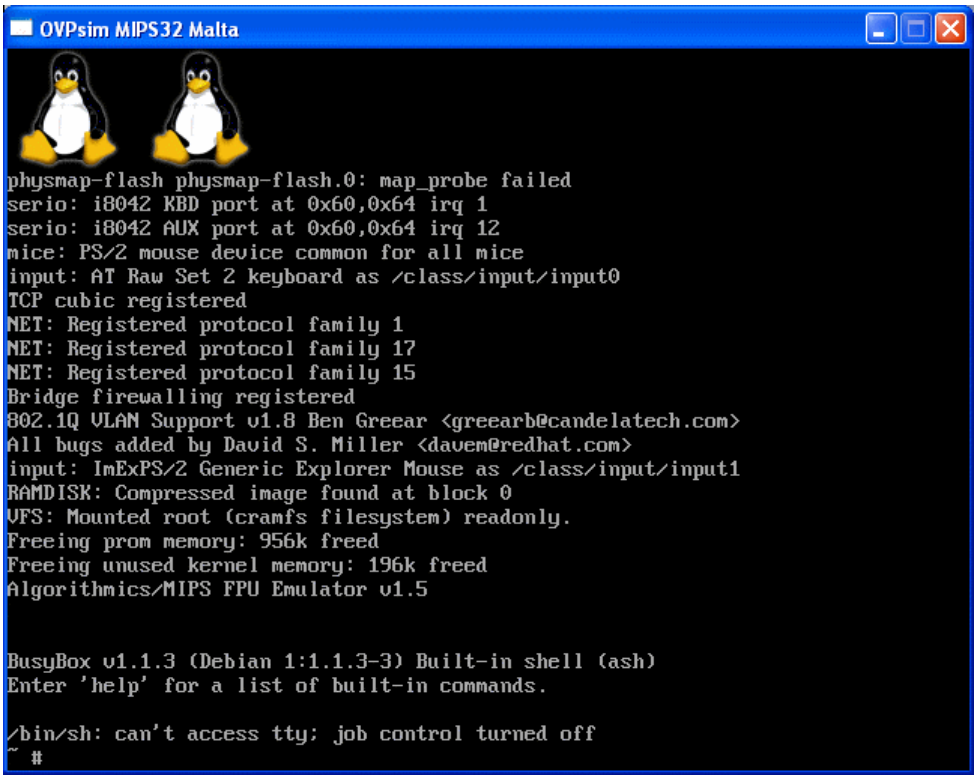


Figure 5: Booting to Shell from Initial Ram Disk (SMP)

6.4.2 Terminating the Simulation

We can terminate the simulator using CTRL-C. This should be done in the window from which the simulation was originally invoked i.e. not the simulated console window. Alternatively the simulated console window can simply be closed which will also terminate the simulation.

The two windows that will be present during the simulation are shown in Figure 6: Execution and Simulated Console Windows

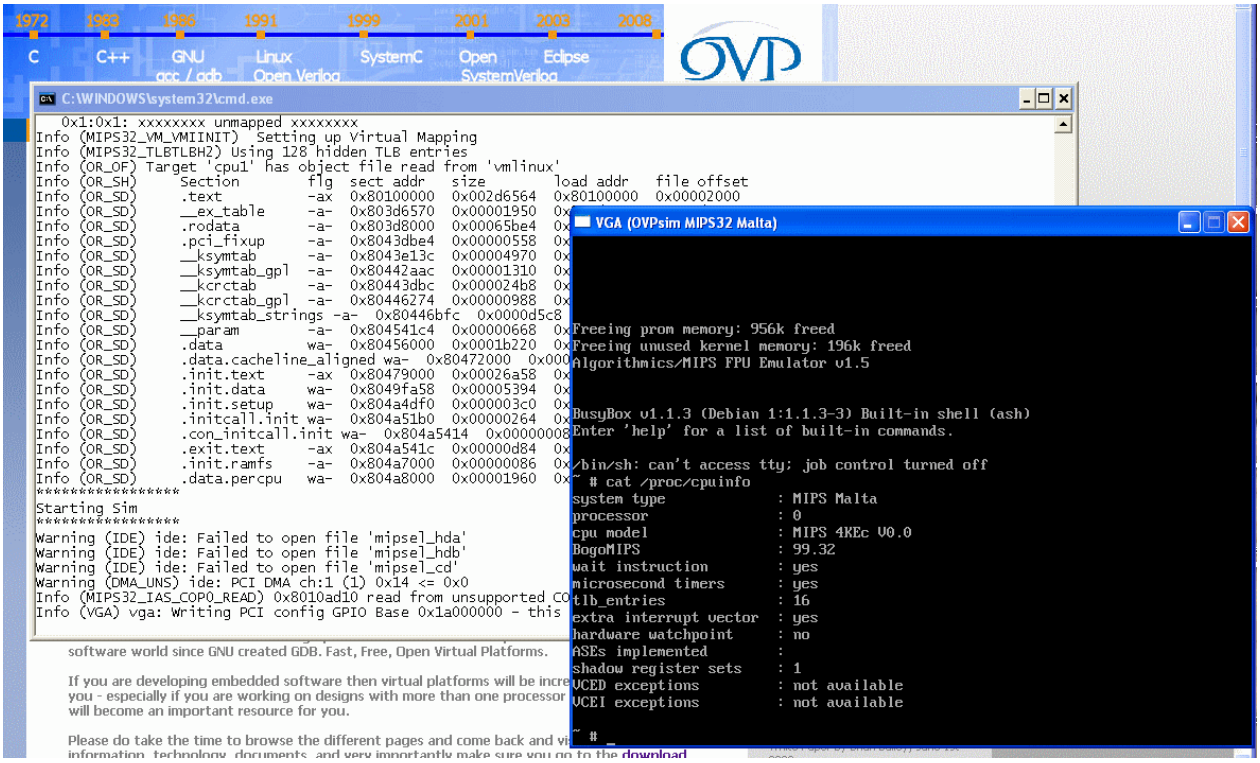


Figure 6: Execution and Simulated Console Windows

When terminated the simulator provides some final execution statistics that includes the instructions executed by the processor and the performance achieved, as shown in Figure 7a: Execution Statistics Output

```
Info
Info ------------------------------------------------------
Info CPU '/mipsle1' STATISTICS
Info   Type                  : mips32 (24KEc)
Info   Nominal MIPS          : 100
Info   Final program counter : 0x801016d4
Info   Simulated instructions: 604,201,590
Info   Simulated MIPS        : 101.7
Info ------------------------------------------------------
Info
Info ------------------------------------------------------
Info SIMULATION TIME STATISTICS
Info   Simulated time        : 35.50 seconds
Info   User time             : 4.72 seconds
Info   System time           : 0.64 seconds
Info   Elapsed time          : 5.94 seconds
Info ------------------------------------------------------
```

Figure 7a: Execution Statistics Output Linux Booted

```
Info ------------------------------------------------------
Info CPU '/mipsle1_TC0' STATISTICS
Info   Type                  : mips32 (34Kc)
Info   Nominal MIPS          : 100
Info   Final program counter : 0x801016d4
Info   Simulated instructions: 587,855,900
Info   Simulated MIPS        : 72.6
Info ------------------------------------------------------
Info
Info ------------------------------------------------------
Info CPU '/mipsle1_TC1' STATISTICS
Info   Type                  : mips32 (34Kc)
Info   Nominal MIPS          : 100
Info   Final program counter : 0x801016d4
Info   Simulated instructions: 68,681,706
Info   Simulated MIPS        : 8.5
Info ------------------------------------------------------
Info
Info ------------------------------------------------------
Info TOTAL
Info   Simulated instructions: 656,537,606
Info   Simulated MIPS        : 81.1
Info ------------------------------------------------------
Info
Info ------------------------------------------------------
Info SIMULATION TIME STATISTICS
Info   Simulated time        : 28.80 seconds
Info   User time             : 6.91 seconds
Info   System time           : 0.61 seconds
Info   Elapsed time          : 8.10 seconds
Info ------------------------------------------------------
```

Figure 8b: Execution Statistics Output SMP Linux Booted

7 Installing Full Linux Distribution

In addition to the simple preconfigured ram disk version described in section 6 the OVPsim_demo_linux_MipsMalta_install demo allows you to download and configure your own full Linux kernel in a simulated platform.

The supplied initial ramdisk allows you to perform a full Linux installation using a Debian Linux distribution site. This requires access to the internet from the machine you are running the simulation in order to download the required files for the Linux installation.

7.1 Simulation Invocation Script

Obtain the OVPsim_demo_MipsMalta_install package from www.ovpworld.org and install it on your system. Go to the directory:

IMPERAS_HOME/ Demo/OVPsim_linux_MipsMalta_install

and launch the simulation with the script file ' RUN_InstallLinux.bat' on Windows or the shell script RUN_InstallLinux.sh' on Linux.

This will execute the platform, passing in the Linux kernel, specifying the ramdisk **initrd.gz** to set the boot mode and **--finishonhalt** to stop the simulation if a halt command is issued in the simulated kernel[3]. In addition the **--wallclock** option is selected so the simulation will not run faster than realtime, which can cause premature timeouts when accessing the internet through the semihosting interface..

7.2 Creating an Empty Disk Image

When the script is invoked you will first be prompted to enter the size of disk that should be created. This should be at least 1 Giga Byte.
Figure 9, shows the creation of a 4GB disk image.

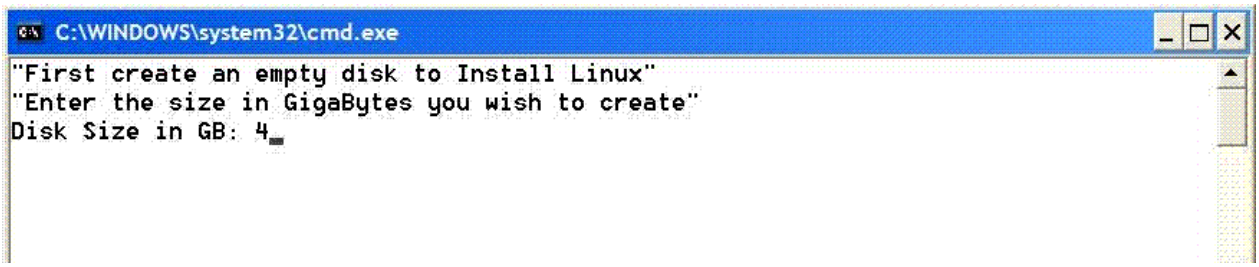


Figure 9: Create 4GB blank Disk Image

[3] The halt command actually invokes a write with a special value to a register in the Malta FPGA. It is this write that causes the simulation to finish.

7.3 Starting the Linux Kernel Boot

Once the disk is created the script will start the simulation of the Malta platform. The initial boot sequence is the same as the other example, with the output from the kernel logged to the console as shown in Figure 10: Kernel Booting from Install INITRD:

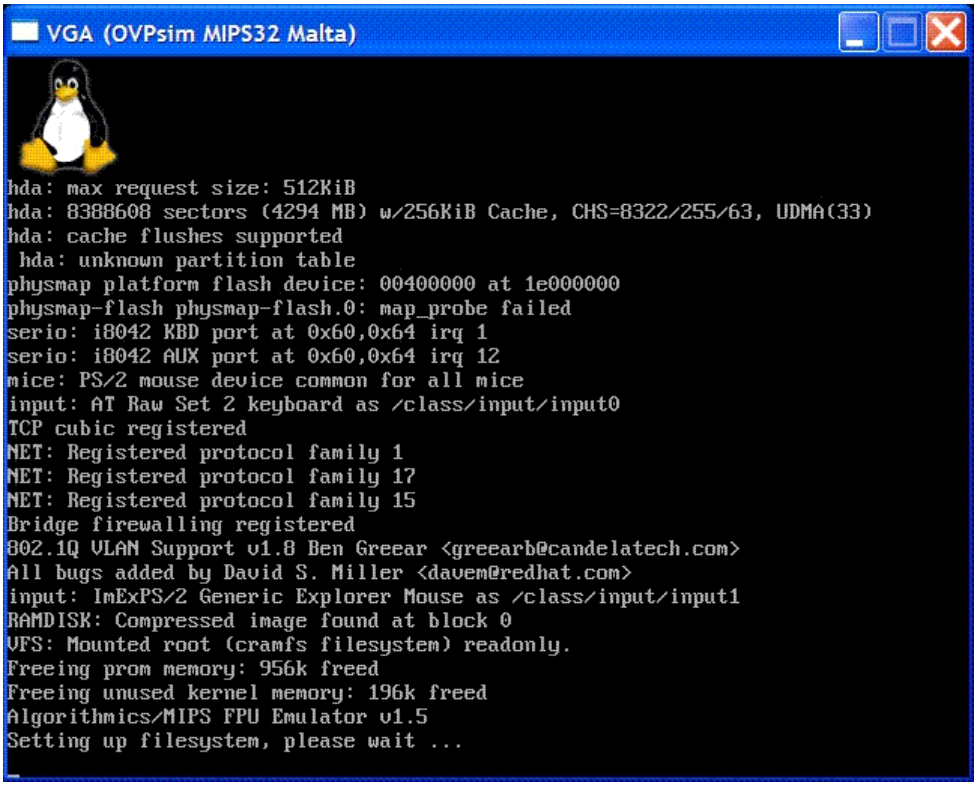


Figure 10: Kernel Booting from Install INITRD

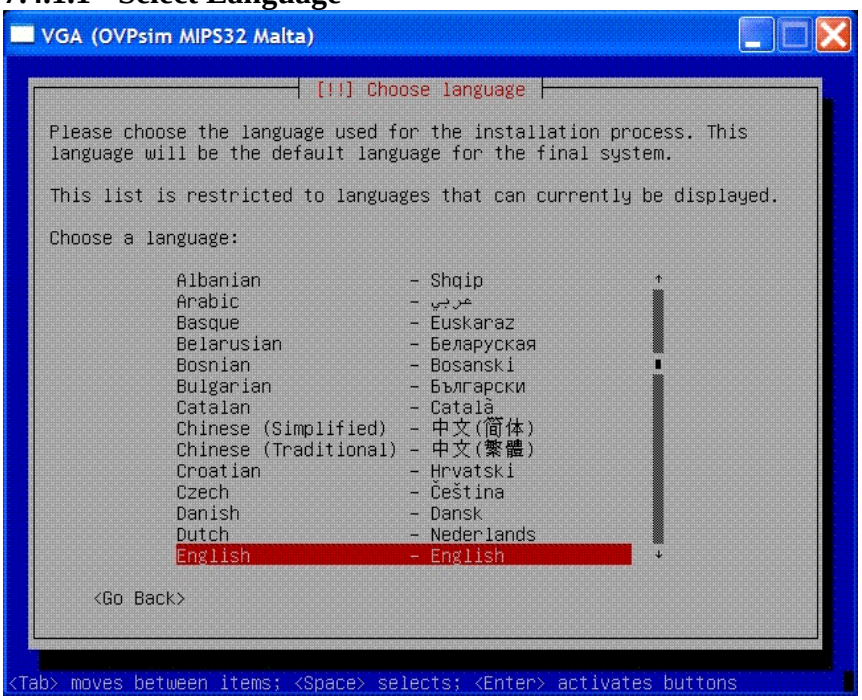
7.4 Installing a Linux Distribution

A typical Linux installation is performed following the prompts of the graphical user interface provided by the VGA peripheral in the virtual platform.

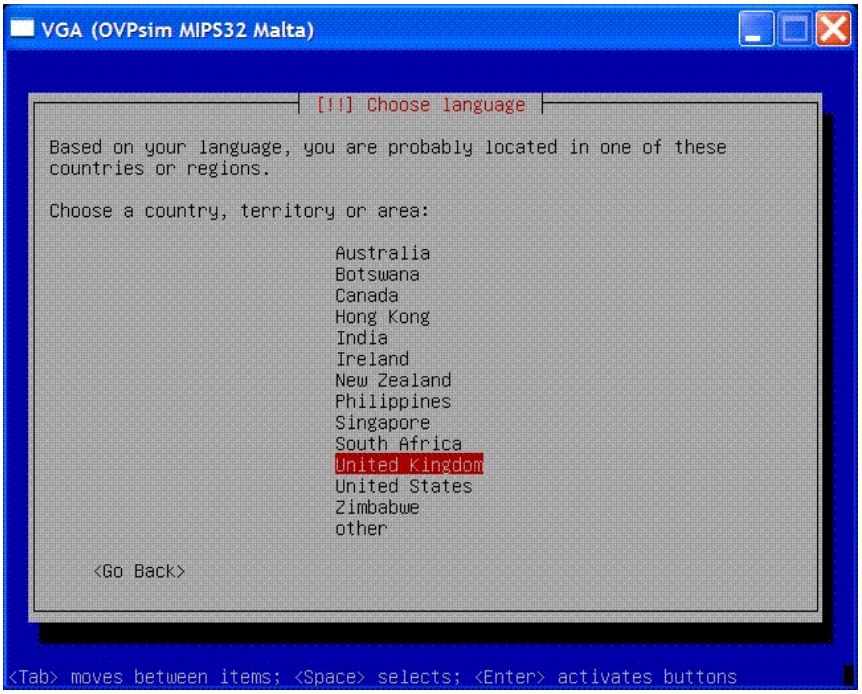
This section describes the full installation sequence. To complete a full installation will take around 1-2 hours.

7.4.1 Configure Local

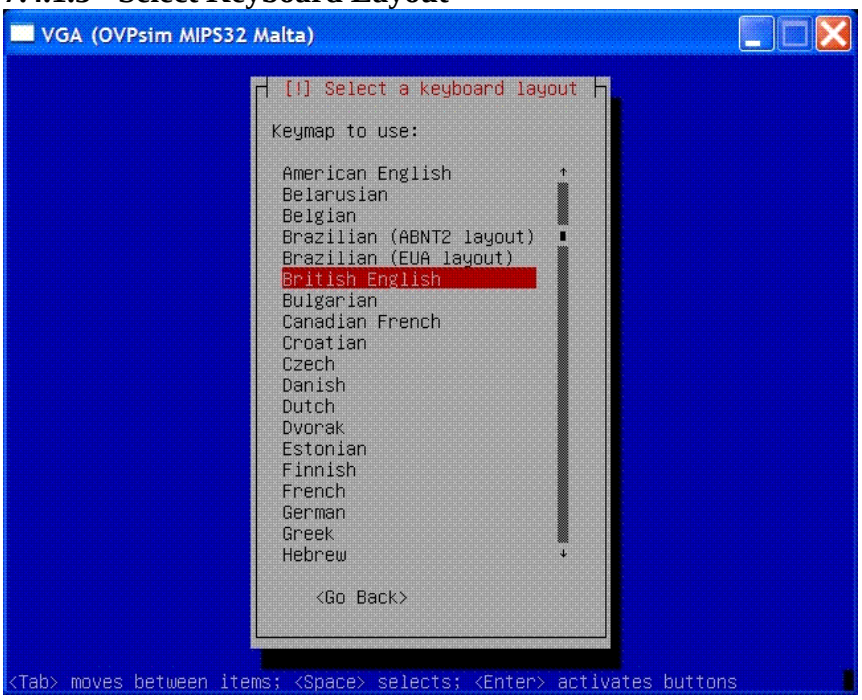
7.4.1.1 Select Language



7.4.1.2 Select location



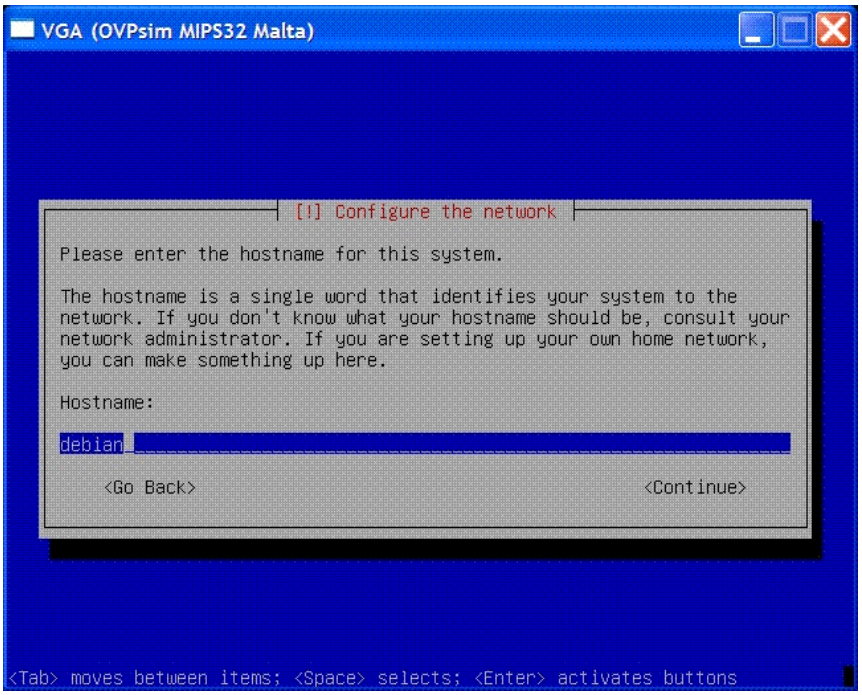
7.4.1.3 Select Keyboard Layout



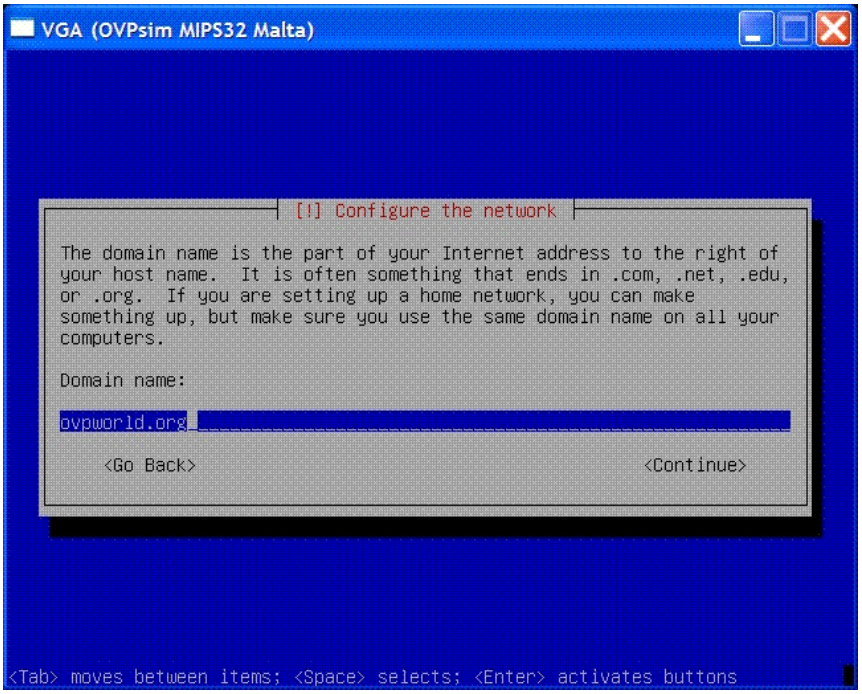
7.4.2 Configure the Network

7.4.2.1 Select Host Name

You should choose a hostname that this installation will be known by. This will be the name that will be seen on your real network when running the simulation.

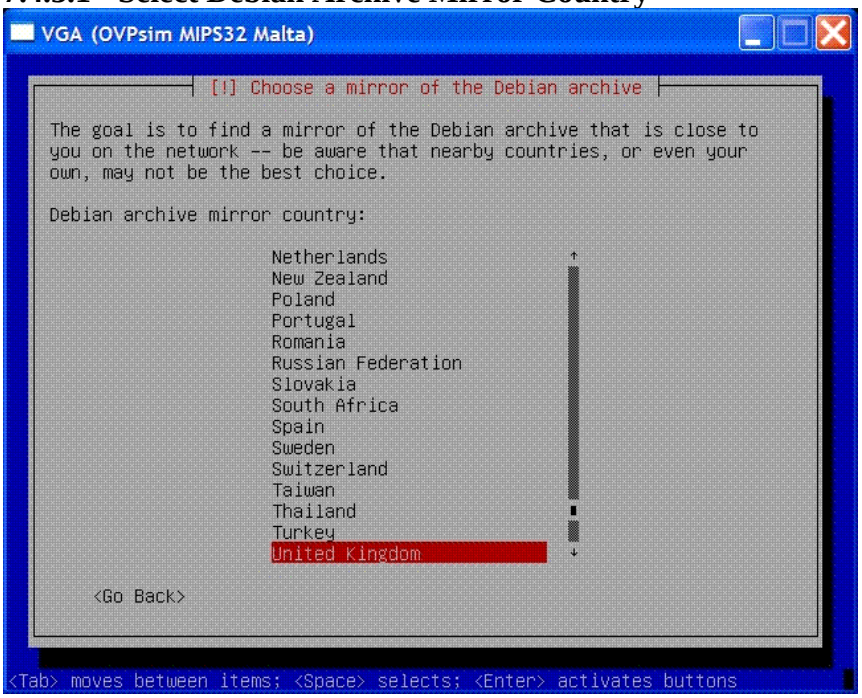


7.4.2.2 Select Domain Name

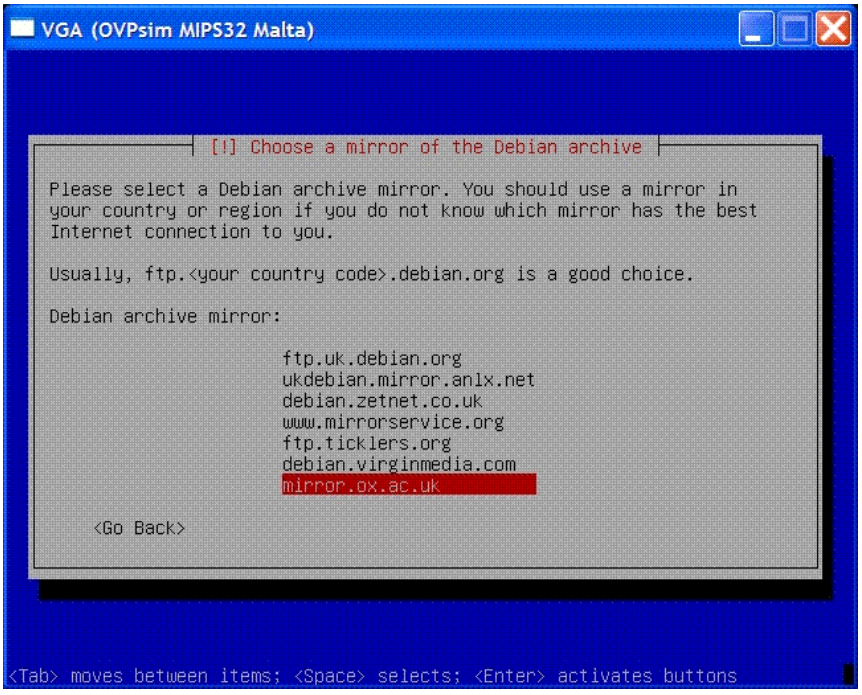


7.4.3 Download Debian Components

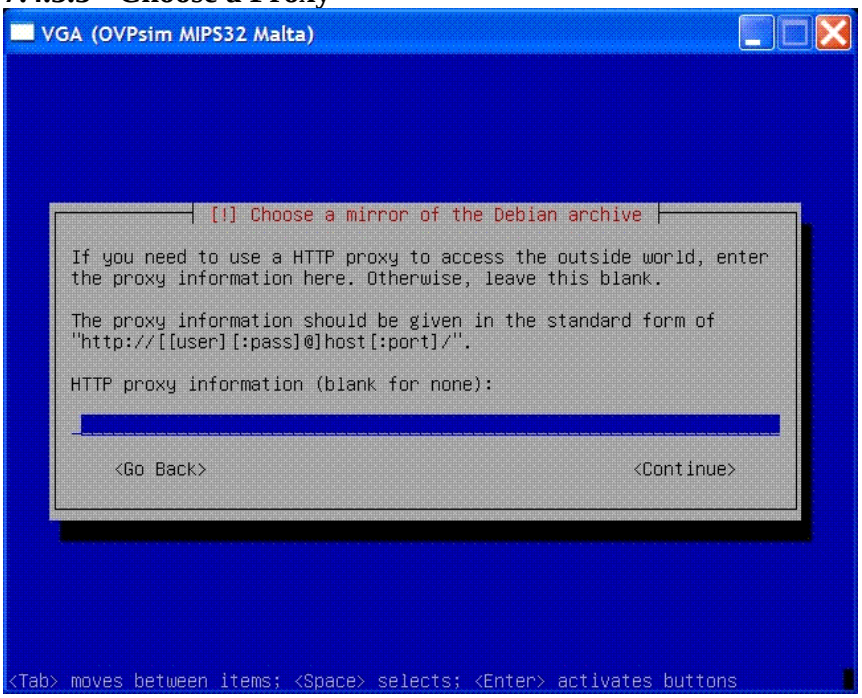
7.4.3.1 Select Debian Archive Mirror Country



7.4.3.2 Select Debian Archive Mirror



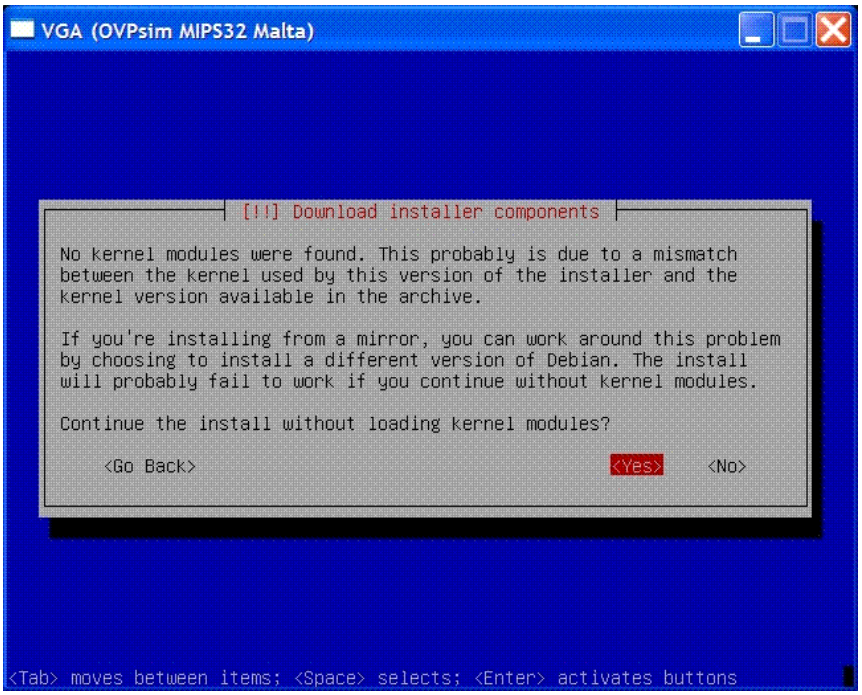
7.4.3.3 Choose a Proxy



7.4.4 Download Installer Components

7.4.4.1 Continue Without Kernel Modules

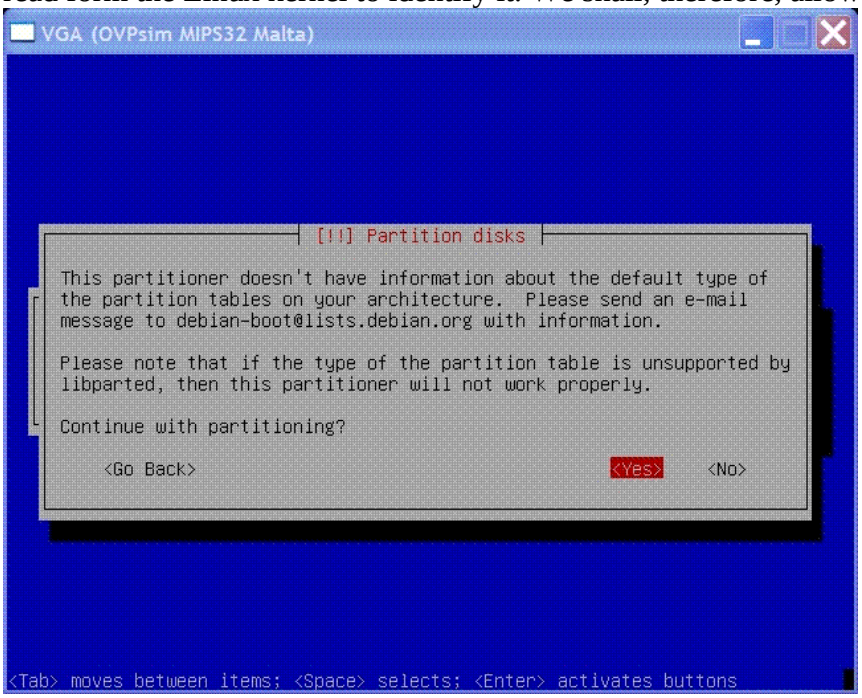
If prompted to continue without kernel modules, select **YES**. The kernel we are providing is built from the Linux 2.6.23 release. The distribution is not yet available at this release. This does not cause any problems for the installation.



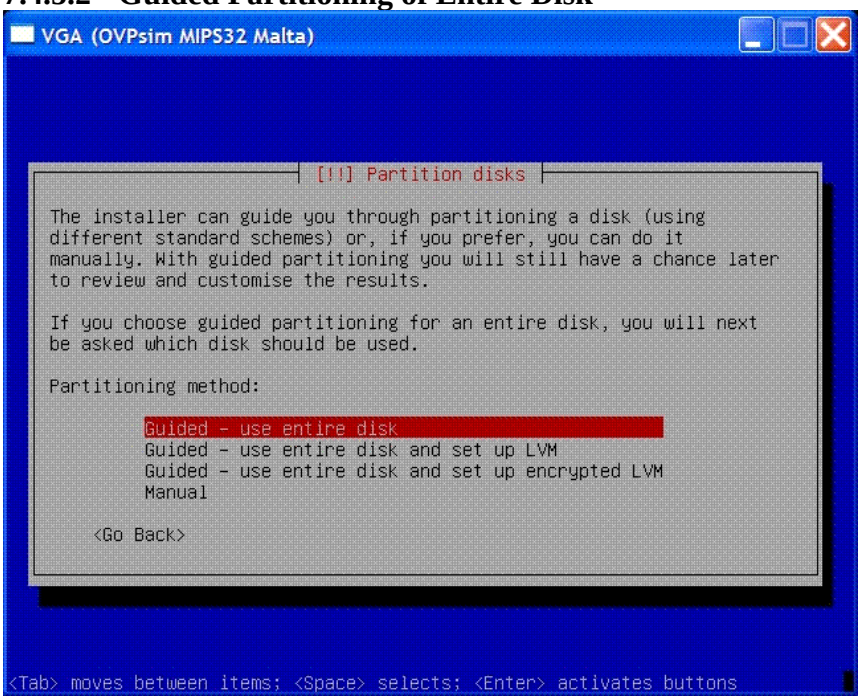
7.4.5 Partition Disk

7.4.5.1 Start Partitioning

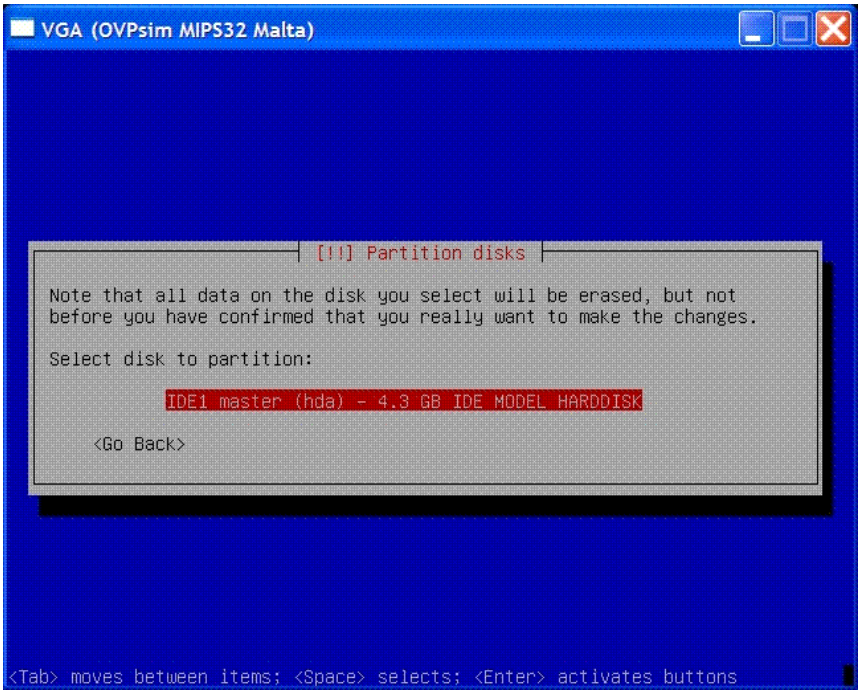
The initial disk image that we have created is blank and so contains no information that can be read form the Linux kernel to identify it. We shall, therefore, allow partitioning to continue.



7.4.5.2 Guided Partitioning of Entire Disk

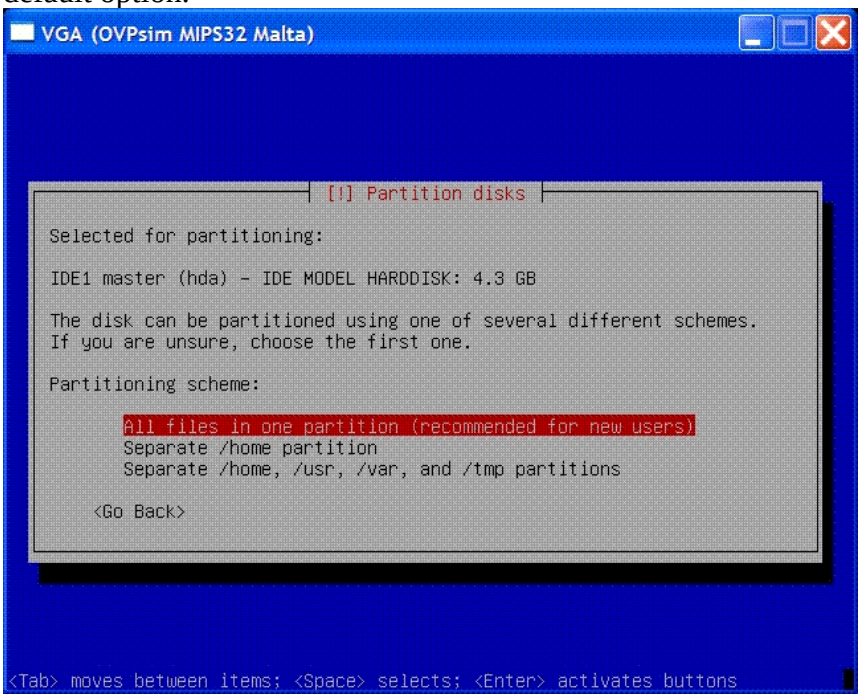


7.4.5.3 Create IDE Master

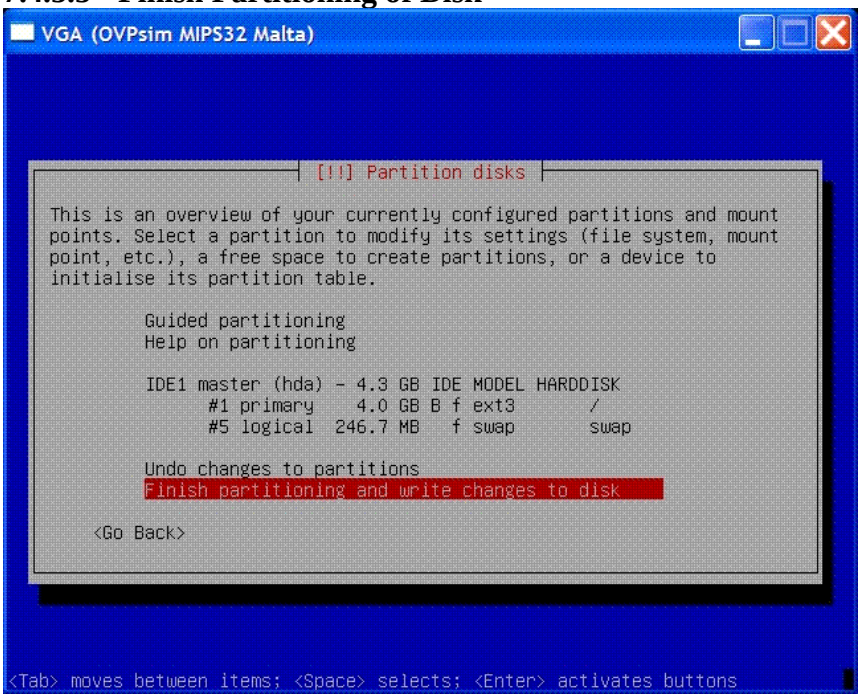


7.4.5.4 Choose Partitioning

Choose the partitioning of the disk. Unless you have specific requirements, please select the default option.

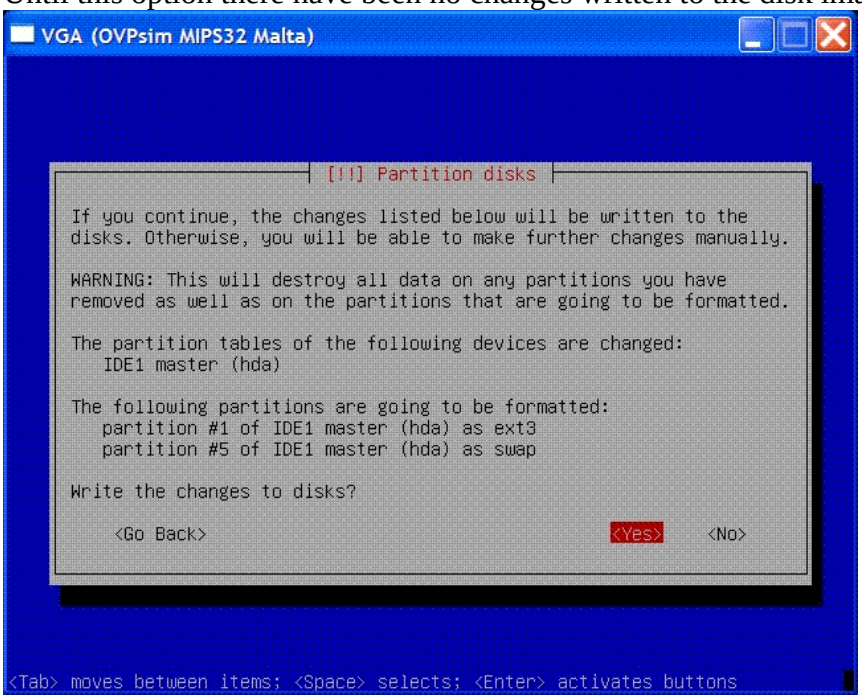


7.4.5.5 Finish Partitioning of Disk



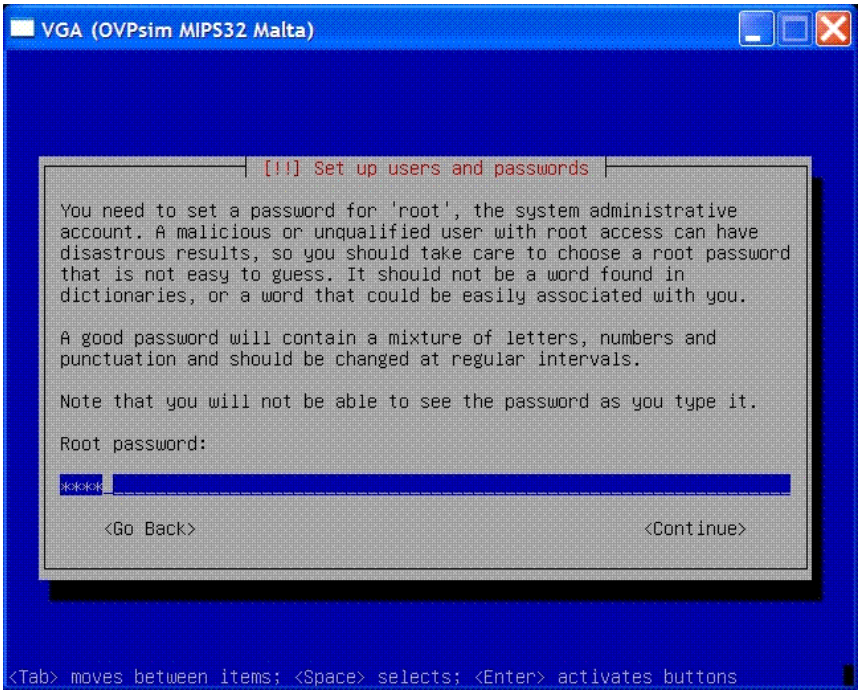
7.4.5.6 Write Changes to Disk

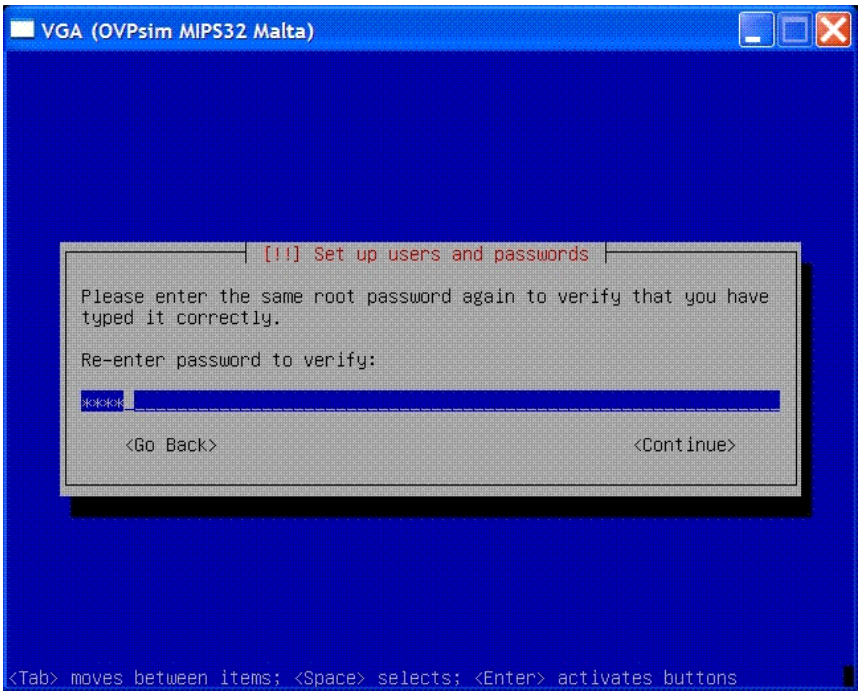
Until this option there have been no changes written to the disk image.



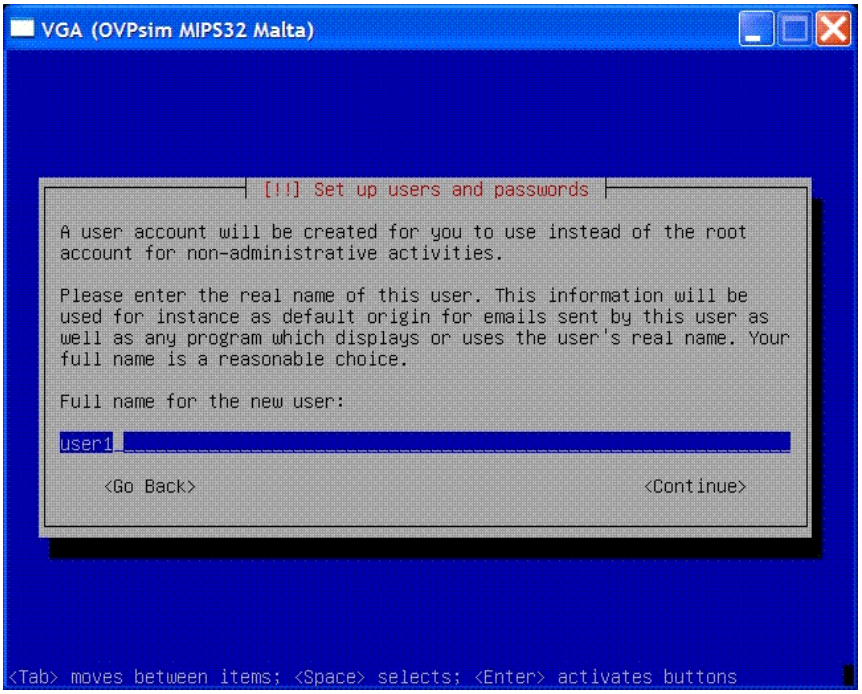
7.4.6 Setup Accounts

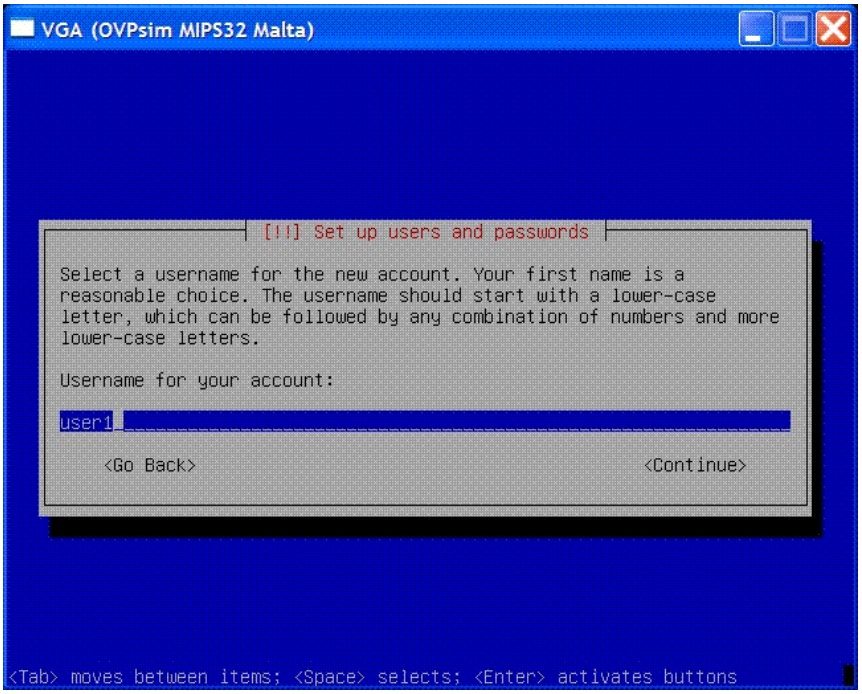
7.4.6.1 Root Password

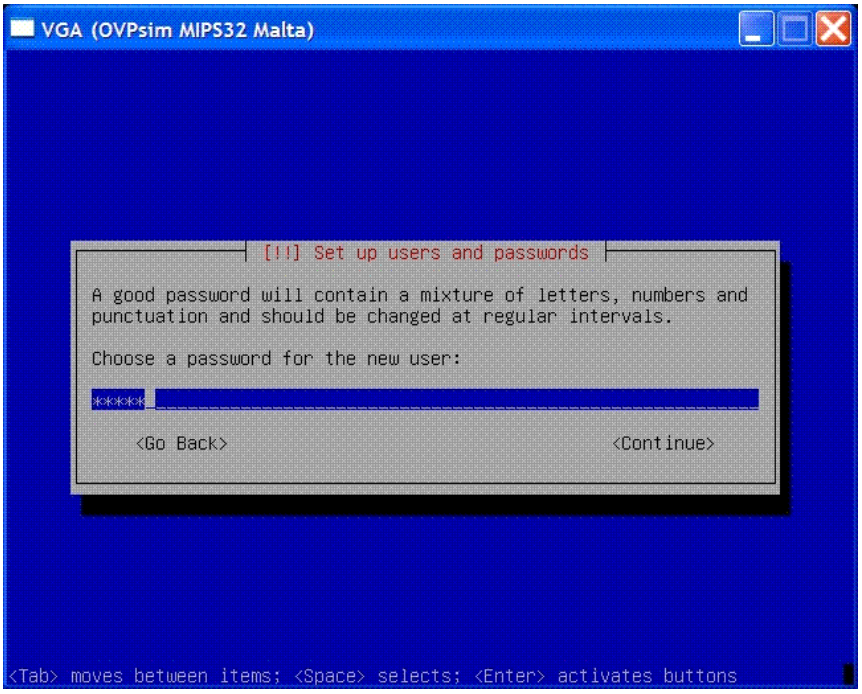


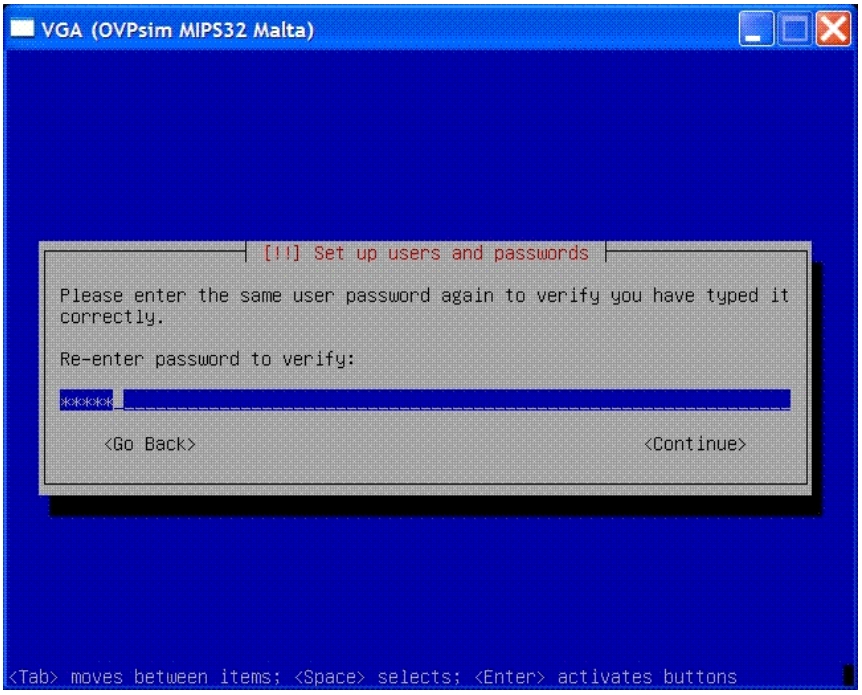


7.4.6.2 Add User Account





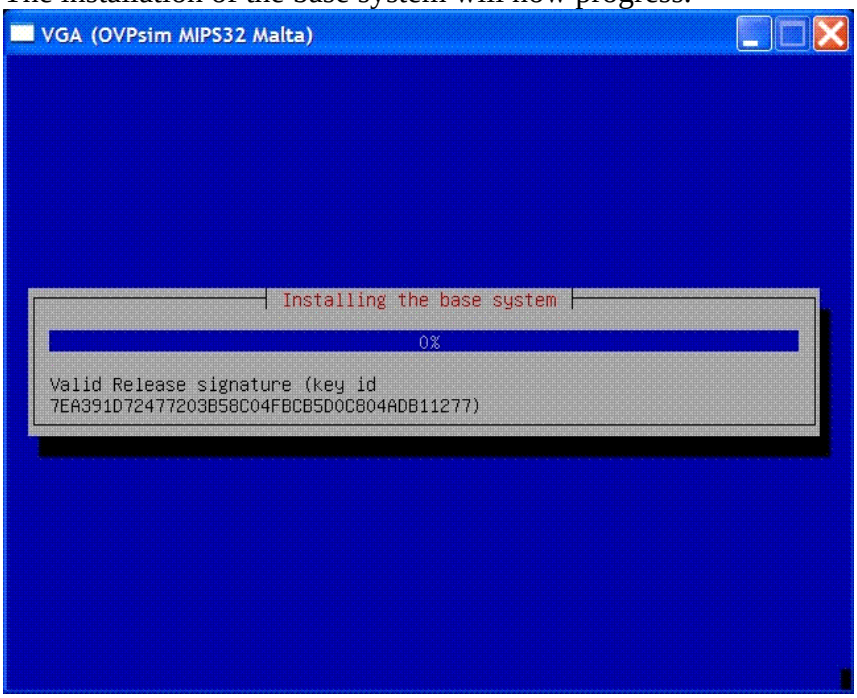

VGA (OVPsim MIPS32 Malta)
[!!] Set up users and passwords
A good password will contain a mixture of letters, numbers and punctuation and should be changed at regular intervals.
Choose a password for the new user:
*****
<Go Back>
<Continue>
<Tab> moves between items; <Space> selects; <Enter> activates buttons


VGA (OVPsim MIPS32 Malta)
[!!] Set up users and passwords
Please enter the same user password again to verify you have typed it correctly.
Re-enter password to verify:
*****
<Go Back>
<Continue>
<Tab> moves between items; <Space> selects; <Enter> activates buttons

7.4.7 Base System Installation

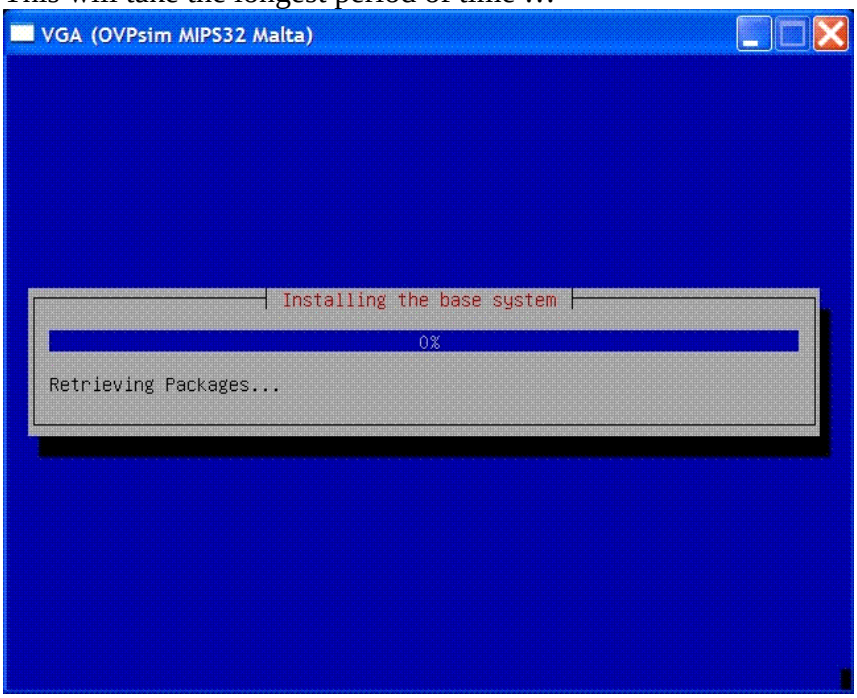
7.4.7.1 Verifying the Release

The installation of the base system will now progress.



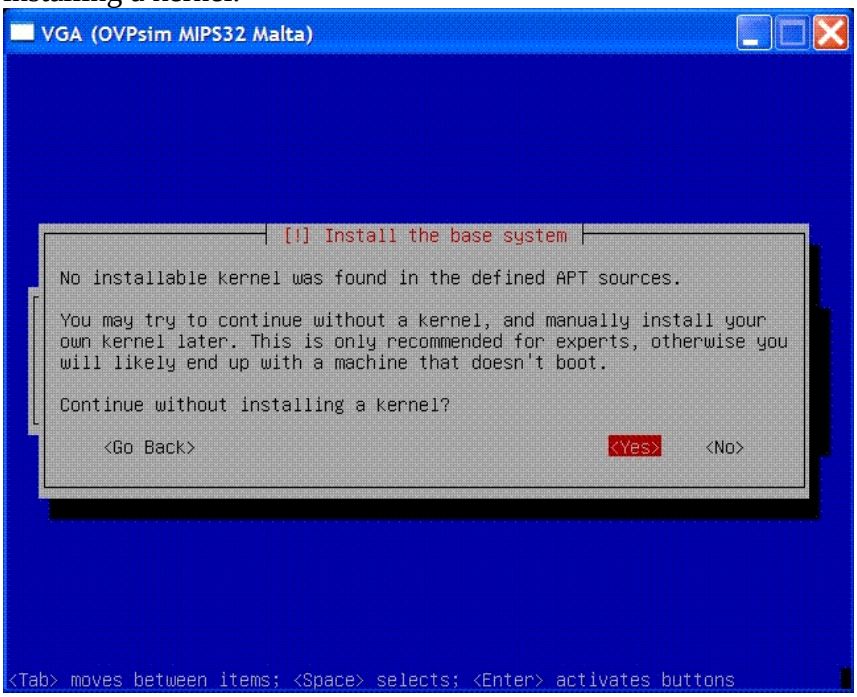
7.4.7.2 Retrieving Packages

This will take the longest period of time …



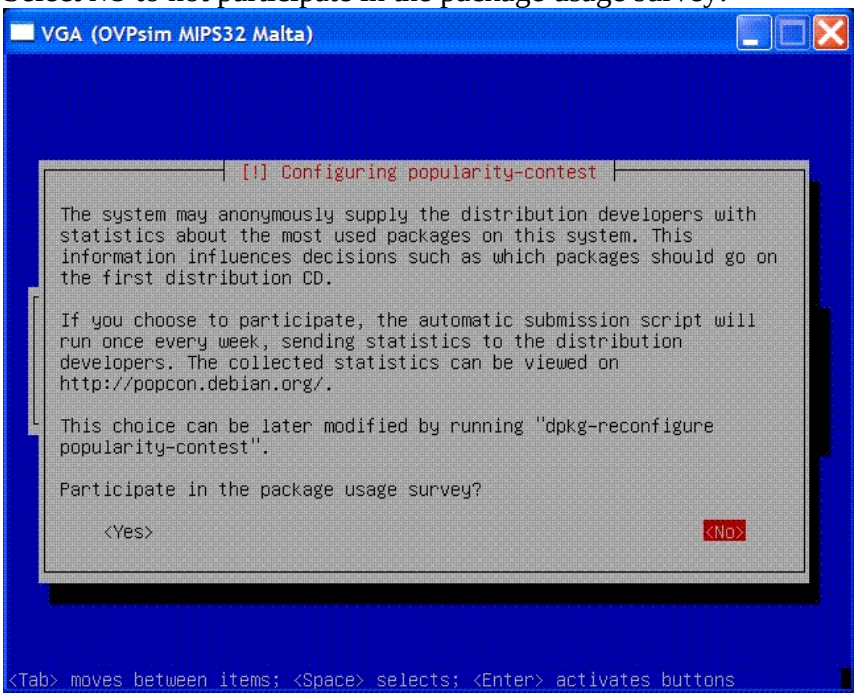
7.4.7.3 Continue Without Kernel

The Linux kernel is provided separately from the release. We, therefore, can continue without installing a kernel.



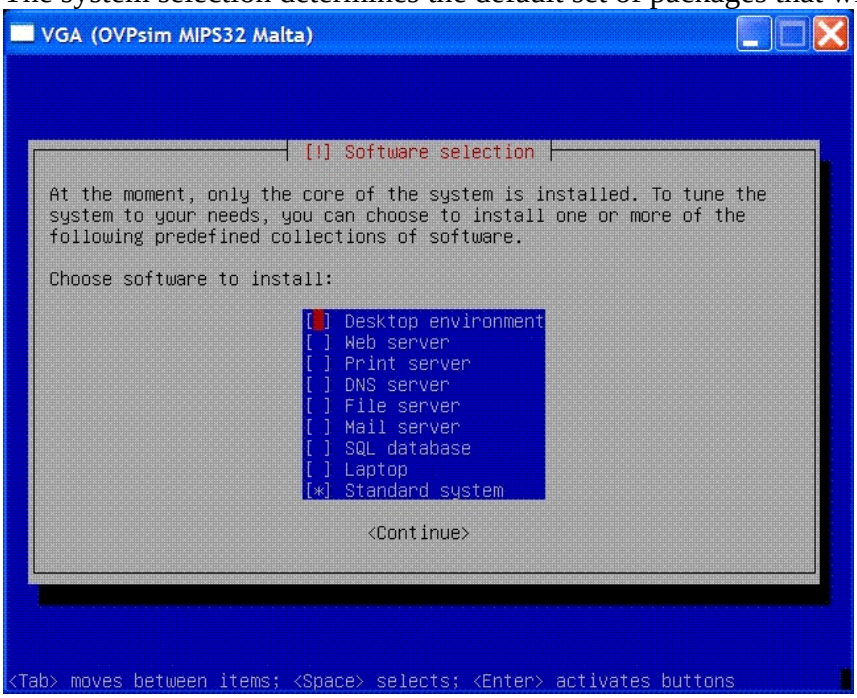
7.4.7.4 Package Usage Survey

Select **NO** to not participate in the package usage survey.



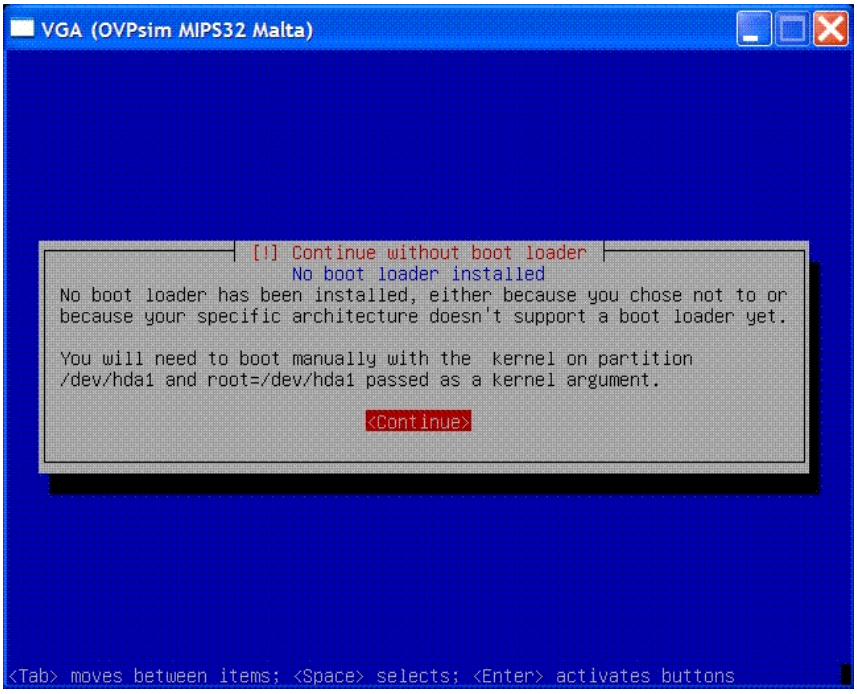
7.4.7.5 System Selection

The system selection determines the default set of packages that will be installed.

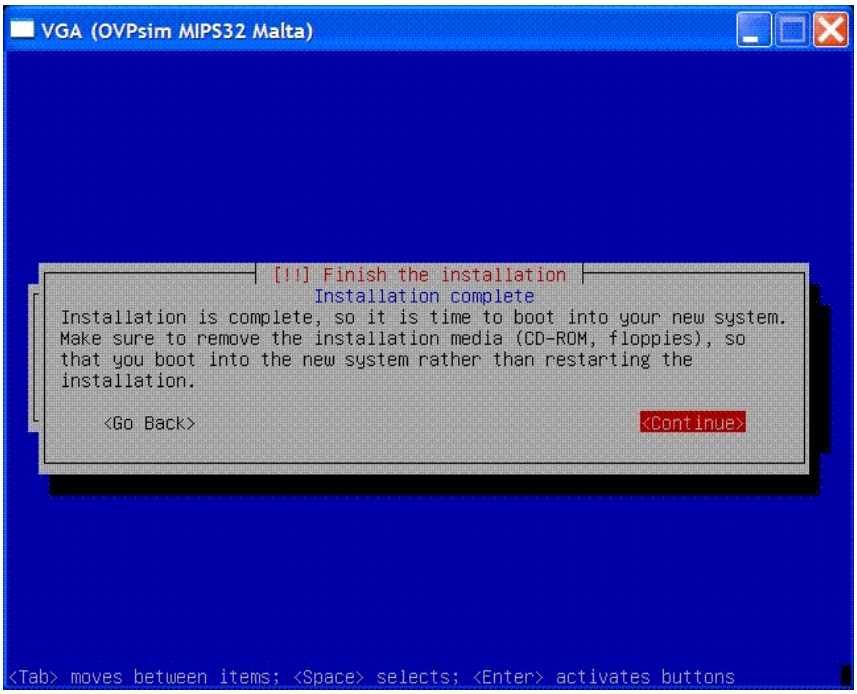


7.4.7.6 No Boot Loader

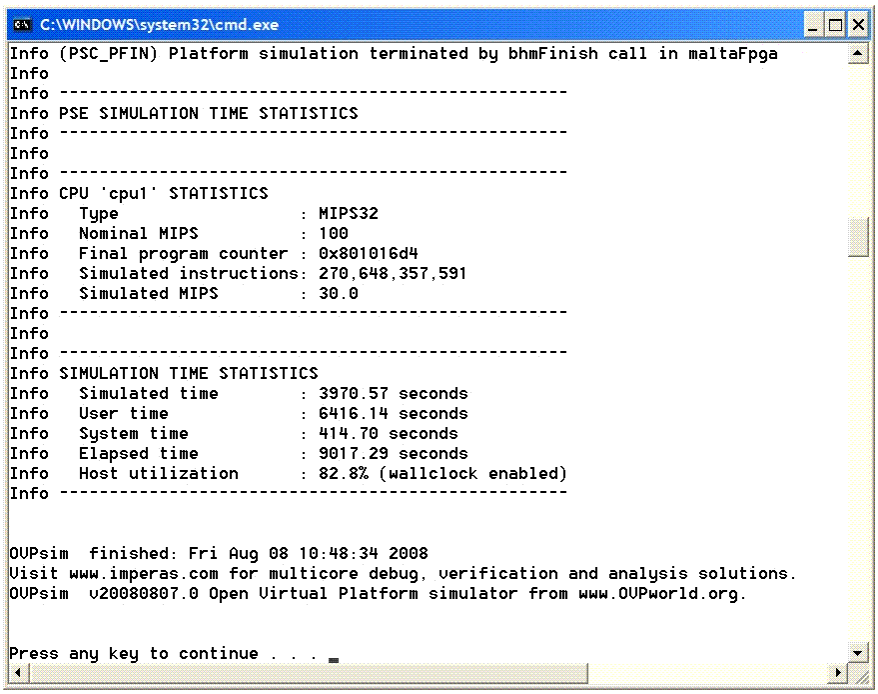
The boot loader is provided as part of the Malta platform, so continue the installation without it.



7.4.8 Complete Installation



At this point the installer will make a request to reboot the system. The simulation platform is set up so that it does not perform a reboot but finishes the simulation, with the statistics of the simulator run being displayed on exit.



7.5 Re-Start System

7.5.1 Boot directly from Disk Image

Run the platform by executing the batch file RUN_BootLinux.bat or shell script RUN_BootLinux.sh that is provided.

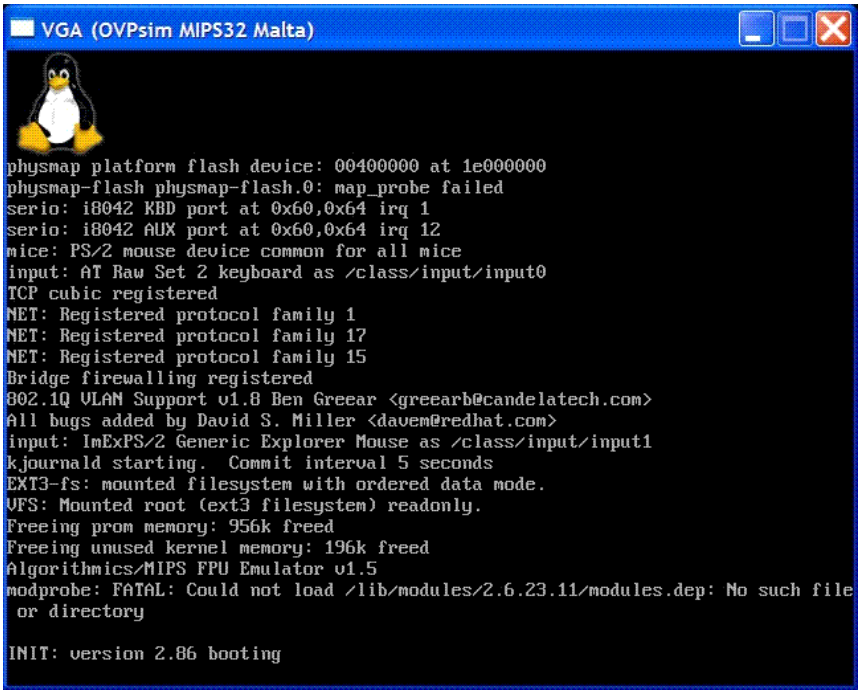
This will execute the platform in the same way as the previous script but will

1. omit the disk image creation stage, and
2. boot from the disk image rather than the initial ram disk

The kernel starts booting in the same way as before by detecting and configuring the hardware in the platform. The boot then continues from the disk image with the full Linux installation.

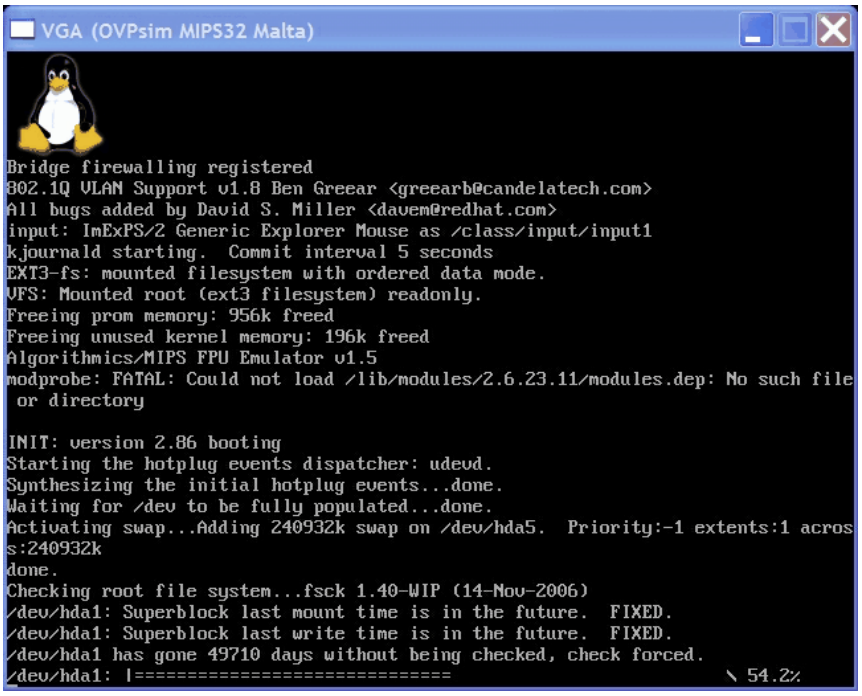
The file modules.dep is missing from the installation and so modprobe fails, this is expected because the kernel modules were not installed as part of the installation. These exist as a separate kernel image.

INIT boots the system …

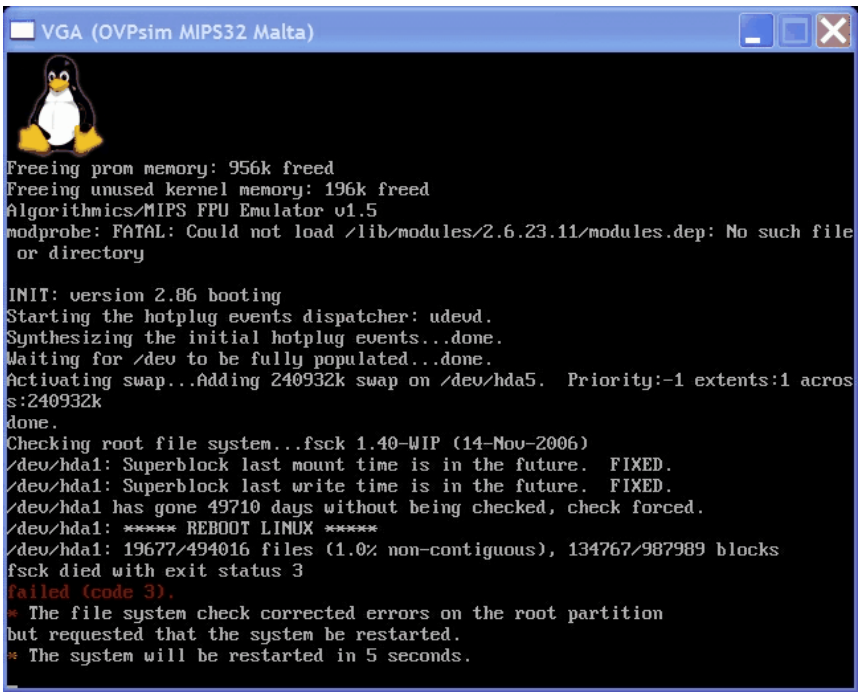


7.5.2 Disk Check

We have created a new disk and upon booting the Linux kernel performs a disk check using fsck.



FSCK requests that the system should restart because of the changes that were made to the disk.

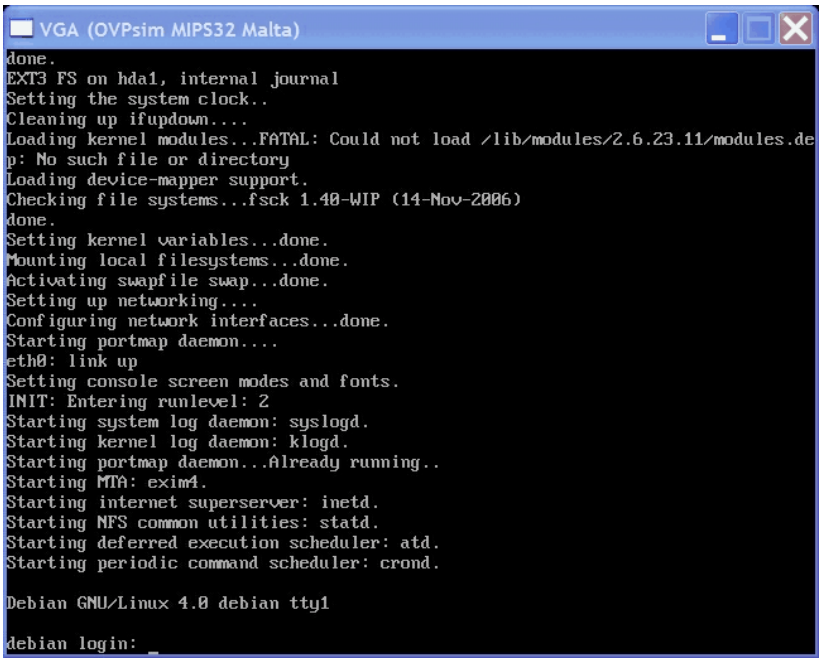


7.5.3 Re-Boot from Disk Image

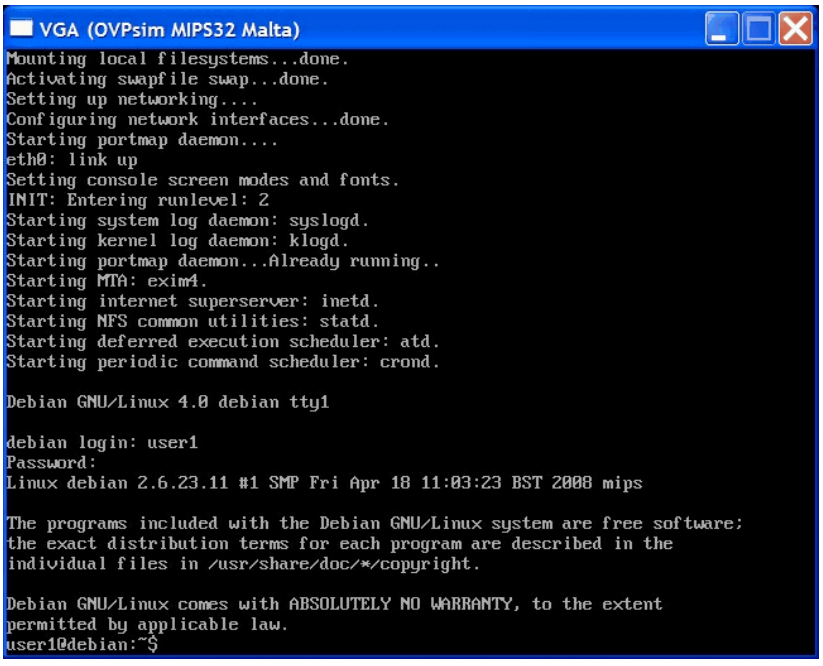
Again run the platform by executing the batch file RUN_BootLinux.bat (or .sh) that is provided.

7.5.4 Login

This time, and for subsequent boots, it will proceed to the login prompt. We have a fully working Linux installation available to use.



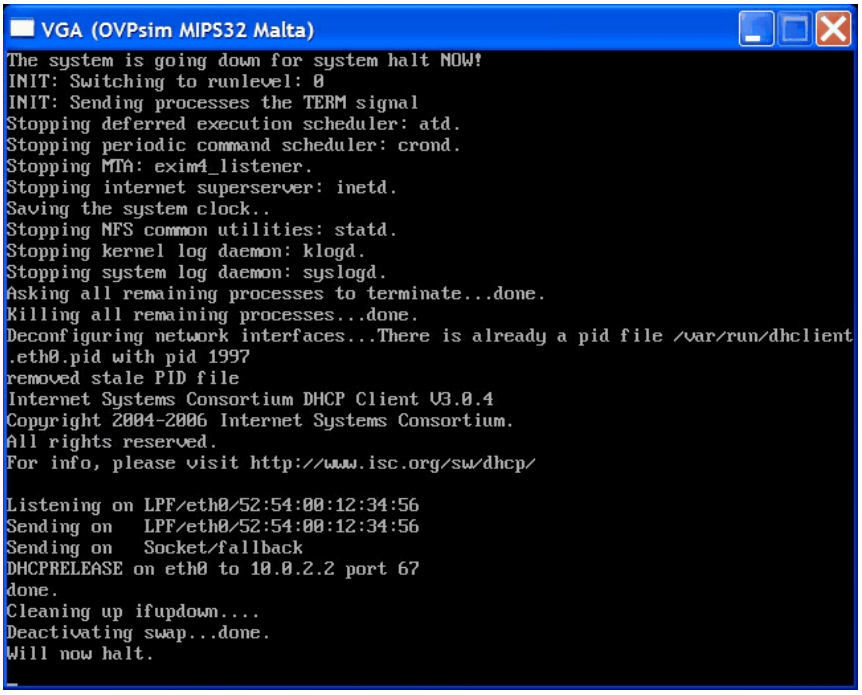
Once logged in we can use this Linux operating system like we would any other.



7.6 Stopping the System

The disk image will be written by the disk controller at regular intervals. However, to ensure that everything on the disk is in the correct state the simulation should be stopped using the halt command when logged in as root. This will ensure all data is flushed to the disk image and that no corruption will occur. When logged in as root issue the command

```
> halt
```



The halt command, when used in conjunction with the ***--finishonhalt***, argument will cause the simulation to stop. If the ***--finishonhalt***, argument is not used you will have to CTRL-C the simulator in the command shell or close the VGA graphics display.

8 Disk Image Files and Initial RAM Disks

8.1 Introduction

The MipsMalta platform can be configured to boot from an initial ram disk or from a hard disk image, as has been shown previously in this document. This section provides some information on the creation and maintenance of these files.

A windows executable is provided to create a blank disk image or standard Linux tools can be used to create and modify the Linux disk images and initial ram disks used in the simulated Linux environment.

8.2 Hard Disk Image

A single file is used to encapsulate a file system as accessed by the disk interface of the PciIDE peripheral component.
By default the platform attempts to access three disks, mipsel_hda, mipsel_hdb and mipsel_cd. The main disk for booting will be mipsel_hda, a warning will be created if disks are not available.

8.2.1 Creation

An executable, create-image.<*arch*>.exe, is provided as part of the installation of **OVPsim_demo_linux_MipsMalta_install** to be used to create a blank disk image. This image can be accessed by the IDE controller model in the Malta platform and populated.

Alternatively the disk image can be created using standard Linux tools.

The following example command creates a disk image called ‘newDisk’ that is an empty file with 1024 blocks of 1024 bytes

```
dd if=/dev/zero of=newDisk count=1024  bs=1024
```

To create an ext3 file system

```
/sbin/mkfs.ext3 newDisk
```

To access the file system we need to mount it onto a local directory

```
mkdir mnt
mount -o loop newDisk mnt
```

We may now copy files into the mounted disk image. When complete unmount the directory we used

```
cp myFile mnt
umount mnt
```

8.2.2 Accessing

If we have a diskimage that we have been using and we wish to access files on the disk we can use standard Linux tools.

List the partitions and note the offset of the partition you want to access

```
/sbin/sfdisk -l -uS diskimage
```

Use the offset to calculate an offset in bytes; for example offset of 63 would yield a byte offset of 63 * 512 = 32256

Make a directory to which the disk can be mounted and mount the disk. You will need to be root to perform the mount operation.

```
mkdir mnt
mount -o loop,offset=32256 diskimage mnt
```

It is now possible to access the partition in the disk image through the mount-point ***mnt***.

8.3 Initial RAM Disk

8.3.1 Extract Files from an initrd.gz

This sequence of commands will allow access to the files in an initial ramdisk.

```
gzip -d initrd.gz
mkdir romfs
mount -o loop initrd romfs
```

Move files from romfs to use in a new file system

```
cd romfs
tar cf - .
cd ../<destination>
tar xf -
```

Now the files are available in <destination>. This can be used as the source for creating a new variant of the initial ram disk
`

8.3.2 Building an initrd.gz

Create a directory into which your files can be written

```
mkdir ramfs
```

copy your files into `ramfs` (you might want to start by extracting an existing initrd)

```
/sbin/mkfs.cramfs ramfs initrd
gzip initrd
chmod 777 initrd.gz
```

8.3.3 Modifying the boot sequence

The boot sequence is controlled by a single script in sbin/init. With the files from the initrd.gz file extracted it is simple to edit this file

```
edit ramfs/sbin/init
```

8.3.4 Example Operations When Running from Initial RAM Disk

There is very limited functionality when running from the initial ram disk. However the following are useful operations that can be performed.

8.3.4.1 Make a writeable directory

When booted from an initrd image, the root FS is read-only. To make a read/write directory:

```
mount -t tmpfs /dev/ram0 /t2
```

Now /t2 is a writable directory on the ram disk.

⇒ Any files written here will be lost when simulation is terminated.

8.3.4.2 Configure the NIC

The following commands will enable the Ethernet NIC on the simulated Malta platform.

```
ifconfig lo up
ifconfig eth0 10.0.2.15 up
route add default gw 10.0.2.2
```

We can now access out into the real world …

Example using nc to fetch a page from google.com.

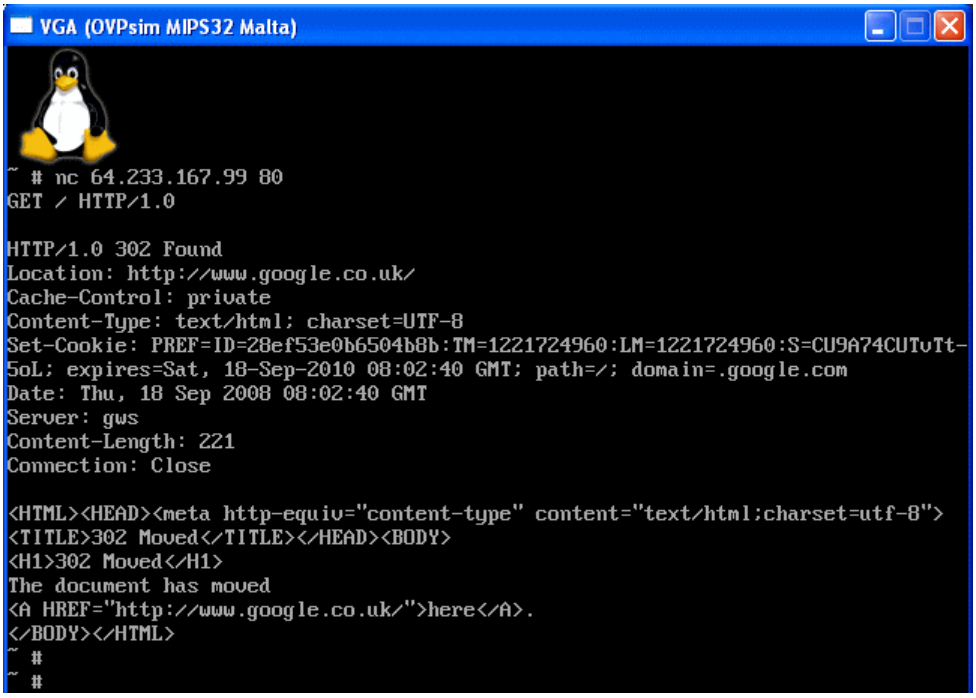
Notes:

1. nslookup was used on a Linux machine to get the ip address for google.com.
2. There is no further prompt after the nc command
3. Two <Carriage Returns> are required after the GET command.

In the shell use the nc command and enter the following

```
# nc 64.233.167.99 80
 GET / HTTP/1.0
```

The result of this command provides information from the www.google.com website, as shown in the following figure.



9 Re-building the Debian Linux Kernel

9.1 Introduction

This section provides some example instructions to re-build the vmlinux kernel from a Debian distribution. The build is carried out on a Linux platform.

In order to rebuild the Linux kernel it is first necessary to build the Cross-Compiler

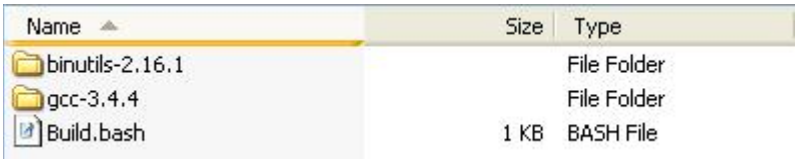
9.2 Building on a Linux Platform

9.2.1 Building the Cross-Compiler

To build a cross-compiler it is first required to obtain **binutils** and **compiler source**. In this example we have used binutils-2.16.1 and gcc-3.4.4. The version of gcc used is dependent upon the support for the target required. The targets we are building are to support MIPS32 Little Endian and MIPS Malta platform.

The information contained in this section is derived from the information available at http://www.linux-mips.org/wiki/Toolchains#GCC

In this example we have extracted the binutils and the gcc source into parallel directories, as shown in the following figure



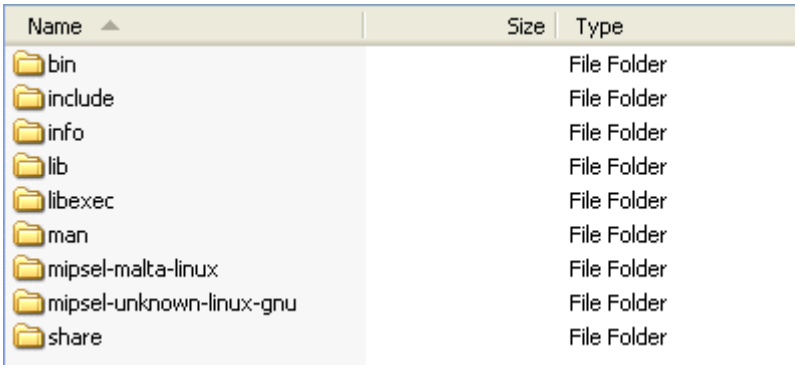
These are then compiled and installed. We created a script, Build.sh, to store this script.

The build script (Build.sh) for the Cross-Compiler looks like:

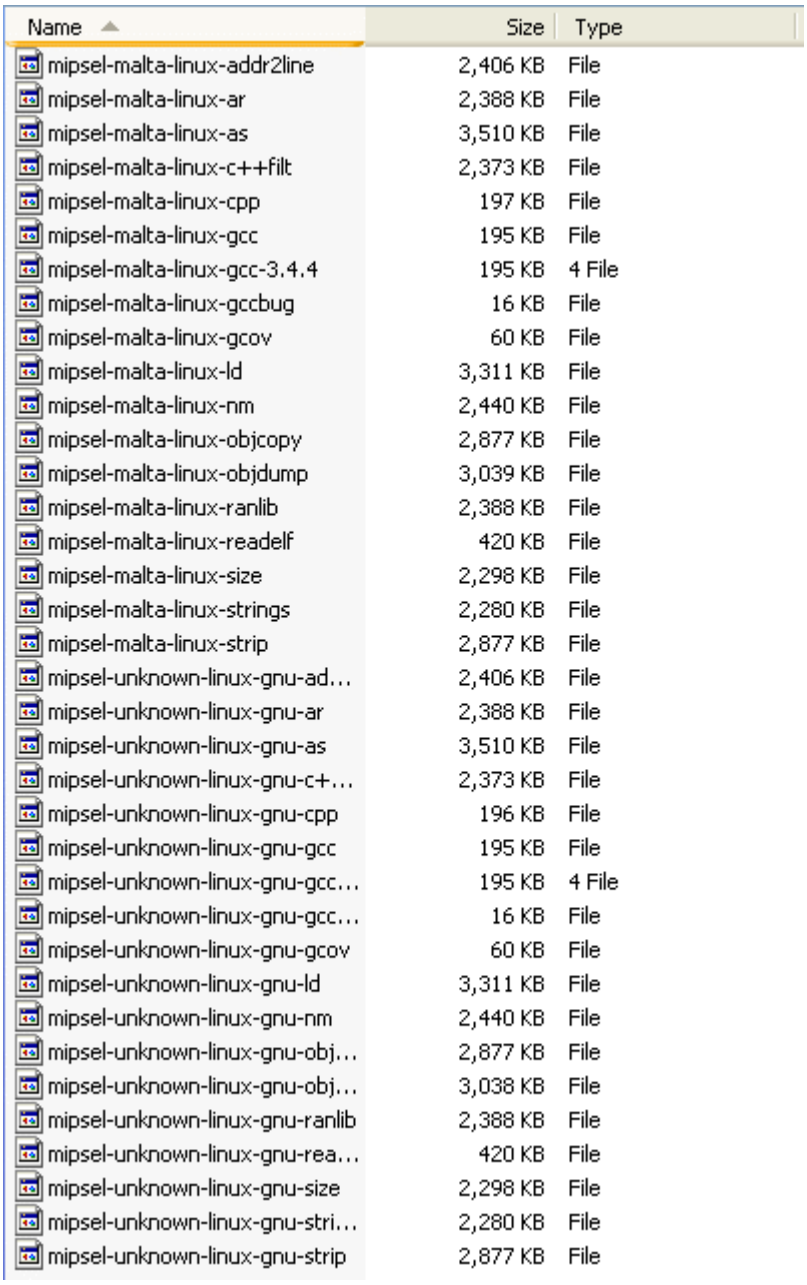
```
#!/bin/sh

here=$(pwd)
install=$(dirname $here)/install
mkdir -p ${install}

# First build a linux compiler
export WDIR=/tmp
#export TARGET=mipsel-unknown-linux-gnu
export PREFIX=${install}

export PATH=${PATH}:${PREFIX}/bin

TARGETS="mipsel-unknown-linux-gnu mipsel-malta-linux"
TARGETS=mipsel-malta-linux
for TARGET in $TARGETS; do
```

```
  rm -rf b-b
  mkdir b-b
  pushd b-b
    ../binutils-2.16.1/configure --target=$TARGET --prefix=$PREFIX
    make
    make install
  popd

  rm -rf  b-gcc
  mkdir b-gcc
  pushd b-gcc
    ../gcc-3.4.4/configure --target=$TARGET --prefix=$PREFIX \
      --enable-languages=c --without-headers \
      --with-gnu-ld --with-gnu-as \
      --disable-shared --disable-threads
    make -j2
    make install
  popd
done
```

After execution an install directory is created that includes the directories shown in the following figure:



The bin directory provides tools specific to the mips linux platform.



Name	Size	Type
mipsel-malta-linux-addr2line	2,406 KB	File
mipsel-malta-linux-ar	2,388 KB	File
mipsel-malta-linux-as	3,510 KB	File
mipsel-malta-linux-c++filt	2,373 KB	File
mipsel-malta-linux-cpp	197 KB	File
mipsel-malta-linux-gcc	195 KB	File
mipsel-malta-linux-gcc-3.4.4	195 KB	4 File
mipsel-malta-linux-gccbug	16 KB	File
mipsel-malta-linux-gcov	60 KB	File
mipsel-malta-linux-ld	3,311 KB	File
mipsel-malta-linux-nm	2,440 KB	File
mipsel-malta-linux-objcopy	2,877 KB	File
mipsel-malta-linux-objdump	3,039 KB	File
mipsel-malta-linux-ranlib	2,388 KB	File
mipsel-malta-linux-readelf	420 KB	File
mipsel-malta-linux-size	2,298 KB	File
mipsel-malta-linux-strings	2,280 KB	File
mipsel-malta-linux-strip	2,877 KB	File
mipsel-unknown-linux-gnu-ad...	2,406 KB	File
mipsel-unknown-linux-gnu-ar	2,388 KB	File
mipsel-unknown-linux-gnu-as	3,510 KB	File
mipsel-unknown-linux-gnu-c+...	2,373 KB	File
mipsel-unknown-linux-gnu-cpp	196 KB	File
mipsel-unknown-linux-gnu-gcc	195 KB	File
mipsel-unknown-linux-gnu-gcc...	195 KB	4 File
mipsel-unknown-linux-gnu-gcc...	16 KB	File
mipsel-unknown-linux-gnu-gcov	60 KB	File
mipsel-unknown-linux-gnu-ld	3,311 KB	File
mipsel-unknown-linux-gnu-nm	2,440 KB	File
mipsel-unknown-linux-gnu-obj...	2,877 KB	File
mipsel-unknown-linux-gnu-obj...	3,038 KB	File
mipsel-unknown-linux-gnu-ranlib	2,388 KB	File
mipsel-unknown-linux-gnu-rea...	420 KB	File
mipsel-unknown-linux-gnu-size	2,298 KB	File
mipsel-unknown-linux-gnu-stri...	2,280 KB	File
mipsel-unknown-linux-gnu-strip	2,877 KB	File

9.2.2 Building the Linux Kernel

9.2.2.1 Kernel Build Script

The build script we created for the Linux Kernel looks like:

```
#!/bin/sh

export CROSS_COMPILE_V=/path/to/kernel/install/bin/mipsel-unknown-linux-gnu-
```

```
rm -rf linux-2.6.23.11
tar xvfz linux-2.6.23.11.tar.gz
cp kernel-config linux-2.6.23.11/.config
pushd linux-2.6.23.11
    make CROSS_COMPILE='distcc $(CROSS_COMPILE_V)' CONFIG_DEBUG_INFO=1 -j16
popd
```

9.2.2.2 Kernel Build Configuration

The kernel configuration is contained within the kernel-config file. This defines all the build parameters for the kernel. This was initially created using ‘make config’ for the Linux Kernel and is subsequently stored and copied into the source directory. The actual file used is provided in appendix C.1 MIPS Kernel Configuration File of this document.

9.3 Building on a Windows Platform

The building of the Linux Kernel on a Windows platform is not supported.

10 Debugging the Linux Kernel

10.1 Non-Intrusive Instrumentation

The Imperas Professional tools provide the VAP (Verification, Analysis and profiling) Tools which provide extensive tools useful for debugging a Linux kernel, including:

- Logging of all exceptions (including TLB)
- Simulation termination on a kernel panic
- Simulation termination on a defined kernel execution event; for example exec of /sbin/halt
- Visibility of kernel execution without console mips32 linux kernel

To find out more about the Imperas Professional tools please contact www.imperas.com.

10.2 Attaching the Debugger

See the OVP documents available at www.ovpworld.org to guide the use of a debugger with the MipsMalta platform.

- OVP Debugging Applications with GDB User Guide
- OVP Debugging Applications with Eclipse User Guide

10.2.1 Starting the Platform Simulation

The MipsMalta platform should be started with an additional command line argument to open a port to connect a debugger to.

A batch file provided with the MIPS Malta Platform demonstration should be modified to add "***--port 9998***" that is used to specify an arbitrary port to be opened for connection to a debugger.

The Windows batch file RUN_BootLinux.bat for Windows users is shown modified below

```
...

%PLATFORM_VLNV% ^
        --verbose ^
        --ramdisk initrd.gz ^
        --finishonhalt ^
        --kernel vmlinux ^
        --console tty0 ^
        --console tty0 ^
        --override MipsMalta/mipsle1/variant=34Kc ^
        --output imperas.log  --port 9998

:end
pause
```

the same modification can be made to the equivalent Linux shell script or to any of the other scripts provided in the demo directory.
The debugger is then attached to this port as described in one of the debugging documents referenced above. The kernel can be thought of as an application running directly on the processor.

11 Transferring Files to Guest Linux

11.1 Introduction

The ability to transfer files compiled on the host system into the Guest, simulated, Linux system is crucial to provide an efficient development environment.

This section provides two methods provided by the MipsMalta platform for transferring files. It assumes you have already set up a full Linux Install as explained in section 7 Installing Full Linux Distribution.

11.2 Using an FTP Server

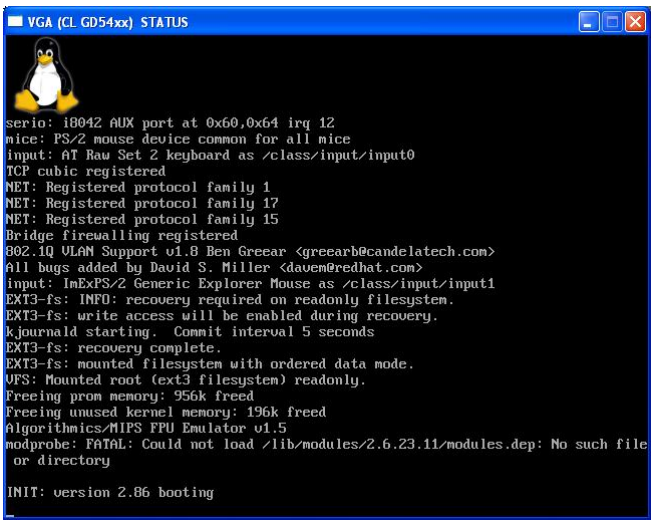
11.2.1 Starting the Simulation

Run the platform, with the script RUN_BootLinuxVNC provided in the demo directory **OVPsim_linux_MipsMalta_install** to pass an additional parameter '**--redir**' to set redirection of the tcp port 11001 that will be used for the connection between the gdb server and a gdb instance later.

For example, on Windows

```
%PLATFORM_VLNV% ^
        --verbose ^
        --ramdisk initrd.gz ^
        --finishonhalt ^
        --kernel vmlinux ^
        --console tty0 ^
        --output imperas.log ^
        --wallclock ^
        --redir
```

The supplied disk image performs a full Linux operating system boot.

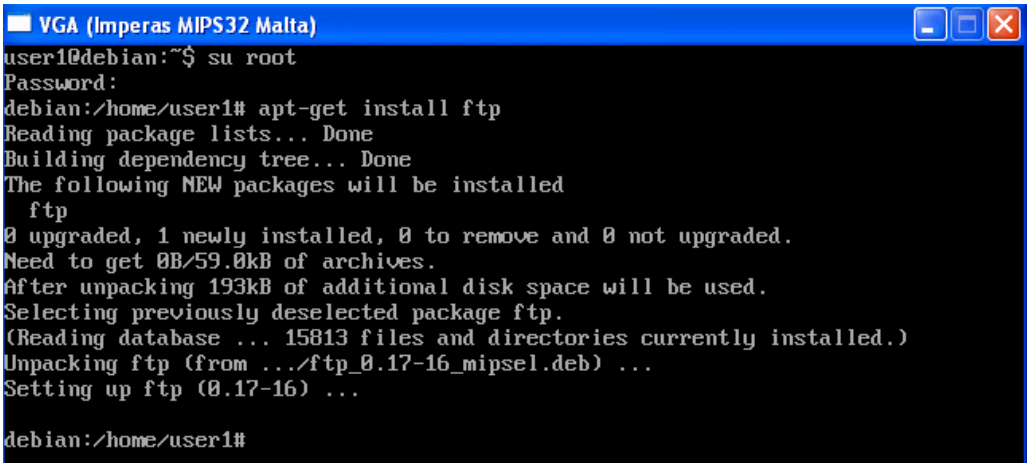


11.2.2 Adding Additional Linux Components

The *ftp* component is required in the Linux operating system to allow the easy transfer of user programs into the simulated Linux environment. This additional component may be installed using apt-get and will require root login in the simulated Linux environment.

Login as root and install the ftp package.

```
> apt-get install ftp
```



Now logout as root using the ***exit*** command.

```
> exit
```

11.2.3 Transfer Files

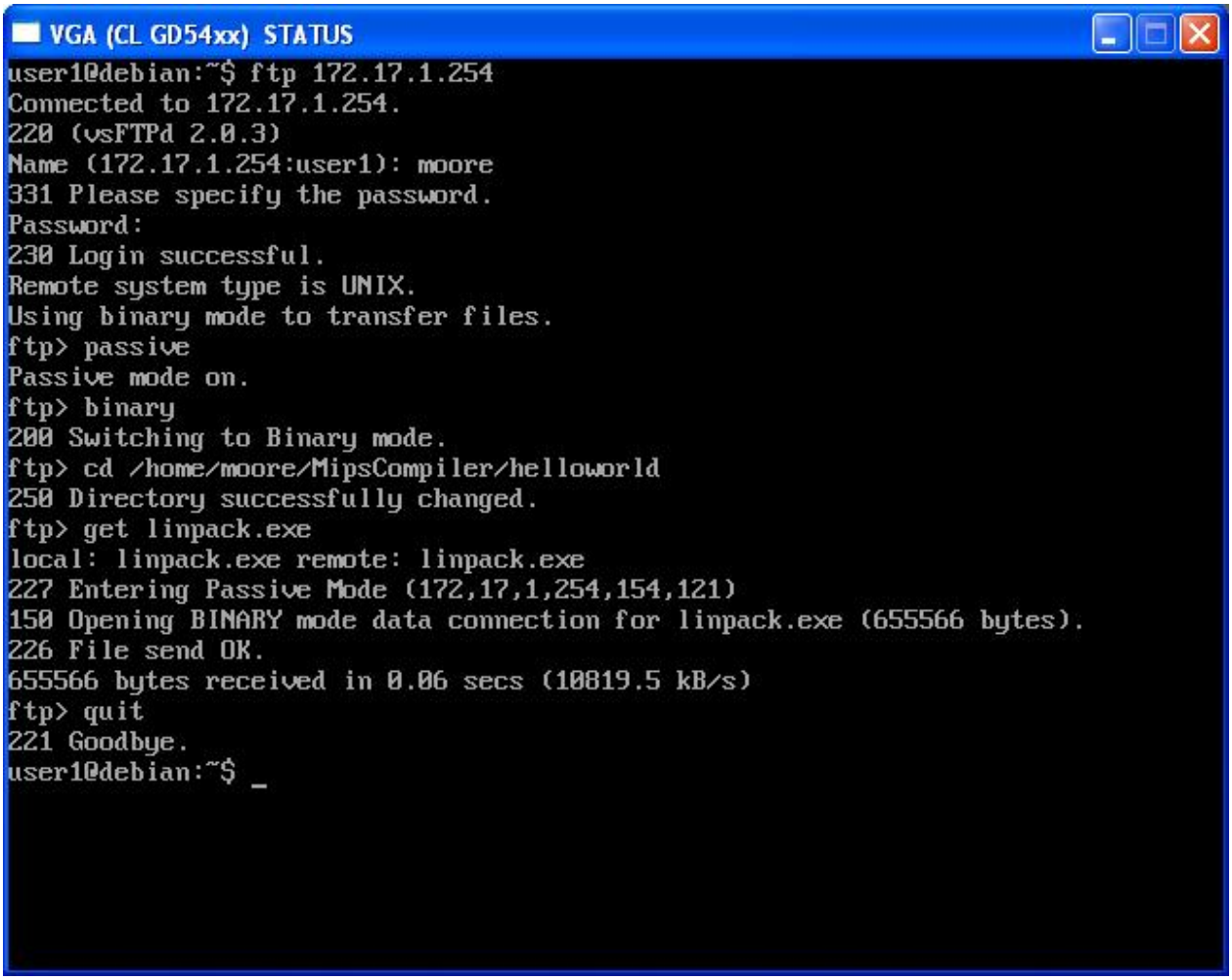
The simulation was executed using the additional command line argument **redir**. This caused a re-direction command of redir="tcp:15901:10.0.2.15:5901,tcp:11001:10.0.2.15:11001" to be passed to the NIC peripheral device. This will expose a port that allows a connection to an FTP server.

For this example we have created a version of the linpack application by cross-compiling on a linux host system.

login as user1 (or whatever user account you set up during the installation) and use ftp to transfer the compiled program onto the simulated Linux platform from an ftp server. For this example we are running an ftp server on a host 172.17.1.254

```
% ftp <server>
% login: <login>
% password: <password>
ftp> passive
ftp> cd <directory of executable>
ftp> binary
ftp> get linpack.exe
ftp> quit
```

This is shown in the following screenshot.



Check the permissions on the transferred file to ensure that it is executable from a user login. If it cannot be executed you can use the chmod command, for example

```
chmod +x linpack.exe
```

11.3 Using TFTP

If no FTP server is available or all the files to be transferred are available on the local Host machine it is possible to use the simpler TFTP protocol.

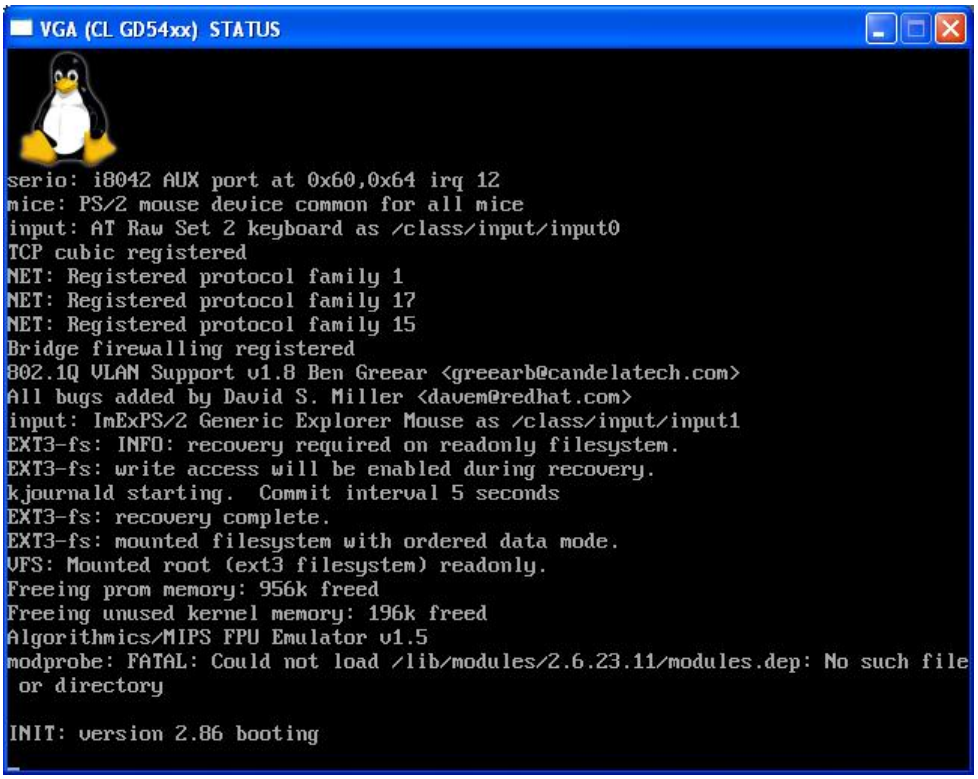
The NIC peripheral includes an emulated TFTP server that intercepts all TFTP packets being sent from the guest operating system. Only the command ***get*** is supported to load files from the Host into the guest operating system. The TFTP requests for files are translated to file operations on the Host machine. The files must be transferred in ***binary*** mode.

11.3.1 Starting the Simulation

Run the platform, modifying the script file RUN_BootLinux provided in the **OVPsim_demo_linux_MipsMalta_install** installation to pass an additional parameter '--tftp <path>' to enable the emulated TFTP server and set the root directory on the Host machine, as shown below for Linux

```
PLATFORM_VLNV \
        --verbose \
        --ramdisk initrd.gz \
        --finishonhalt \
        --kernel vmlinux \
        --console tty0 ^
        --output imperas.log \
        --wallclock \
        --tftp <path>
```

The supplied disk image performs a full Linux operating system boot.



11.3.2 Adding Additional Linux Components

The tftp component may need to be added to the Linux installation. This component will be installed using apt-get and will require root login in the simulated Linux environment.

Login as root and install the TFTP package using the command

```
> apt-get install tftp
```

```
user1@debian:~$ su root
Password:
debian:/home/user1# apt-get install tftp
Reading package lists... Done
Building dependency tree... Done
The following NEW packages will be installed
  tftp
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0B/19.5kB of archives.
After unpacking 98.3kB of additional disk space will be used.
Selecting previously deselected package tftp.
(Reading database ... 15815 files and directories currently installed.)
Unpacking tftp (from .../tftp_0.17-15_mipsel.deb) ...
Setting up tftp (0.17-15) ...
debian:/home/user1# _
```

now logout as root using the ***exit*** command.

```
> exit
```

11.3.3 Transfer Files into Guest Operating System

The simulation was executed with the additional command line argument **tftp <tftpPrefix>.** This enabled the emulation of the TFTP server within the NIC peripheral. The root from which files are transferred is defined by the path addition to the tftp argument. The argument to the ***get*** command must be an absolute path, though it is converted to a path relative to the ***tftpPrefix*** argument to the model. For example, if we have a file ***C:\workspace\userApp\myFile.exe***; we can specify the command line argument as ***tftp C:\workspace*** and access the file in tftp using the command ***get /userApp/myFile.exe***.

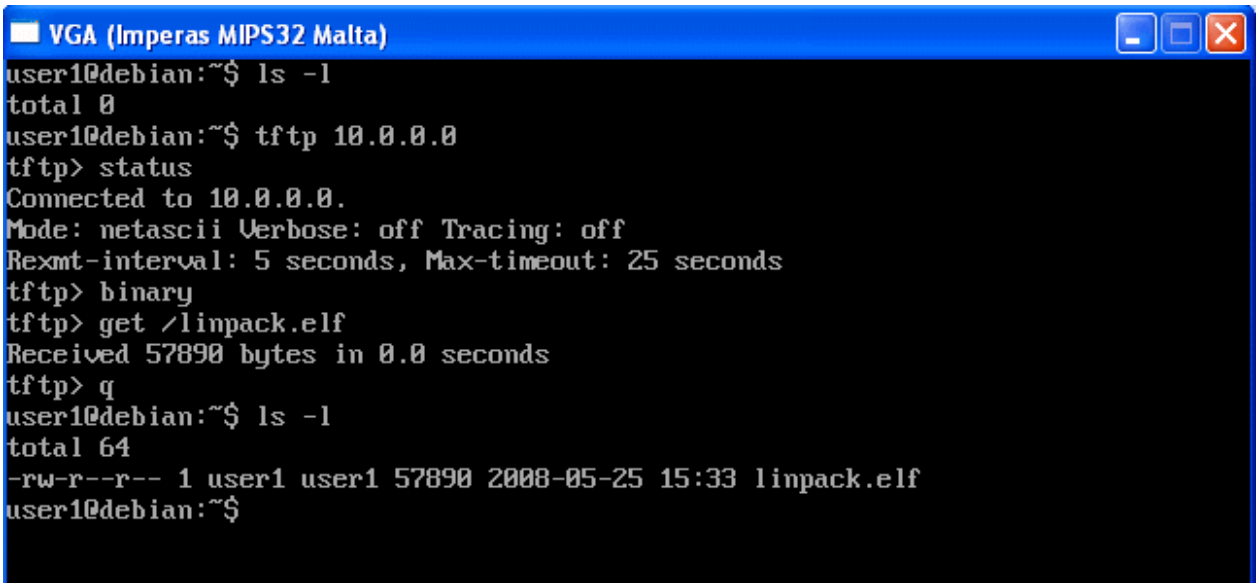
For the following example we have created a version of the linpack application by cross-compiling on a linux host system using the MIPS Linux cross compiler. This is contained in a directory userApp that resides within the directory to which we have set the root using the tftp argument.

login as user1 (or whatever user account you set up during the installation) and use tftp to transfer the compiled program onto the simulated Linux platform.

```
% tftp <host>
tftp> binary
tftp> get /userApp/linpack.exe
tftp> quit
```

⇒ <host> should be set to any IP address that is NOT the local host ip address.

This is shown in the following screenshot.



Check the permissions on the transferred file to ensure that it is executable from a user login. If it cannot be executed you can use the chmod command, for example

```
chmod +x linpack.exe
```

12 Debugging Linux User Applications

12.1 Introduction

Once the Linux kernel is up and running it is the user applications themselves that need to be debugged.

In the OVPsim environment it is simple to connect a debugger, such as command line GDB or a GUI like DDD or Eclipse, to a user application running on the simulated Linux operating system. This is accomplished using one or more instances of gdbserver running in the simulated Linux environment.

This section describes additional components that may be required in the simulated Linux environment and the steps required to accomplish user application debugging.

12.2 Linux Installation

There must be a standard Linux installation available on a disk image. Section 7describes how to build a full Linux installation disk.

12.3 Booting Linux

Run the platform from the command line or by executing a modified version of the batch file bootLinux.bat provided.

This will execute the platform provided, passing in the Linux kernel as the first parameter.

Additional arguments may be used to enable file transfer by FTP or TFTP.

12.3.1 Redirecting TCP Port

Add the '--redir' argument to set redirection of the tcp port 11001. This will enable both the connection between the gdb server and a gdb instance later and transferring files, see section 11.2 Using an FTP Server for more details.

```
PLATFORM_VLNV \
        --verbose \
        --ramdisk initrd.gz \
        --finishonhalt \
        --kernel vmlinux \
        --console tty0 ^
        --output imperas.log \
        --wallclock \
        --redir
```

12.3.2 Enabling TFTP File Transfer

Add the TFTP <path> argument to enable TFTP with the root server directory, on the host, set to <path>. See section 11.3 Using TFTP for more details

```
PLATFORM_VLNV ^
        --verbose ^
        --ramdisk initrd.gz ^
        --finishonhalt ^
        --kernel vmlinux ^
        --console tty0 ^
        --output imperas.log ^
        --wallclock ^
        --tftp “C:\workspace”
```

12.4 Compiling a User Program

12.4.1 Cross Compiler ToolChain

The toolchain used to compile the user application is the MIPS GNU/Linux toolchain available from the CodeSourcery website at

http://www.codesourcery.com/sgpp/lite/mips/portal/subscription?@template=lite

There are installers for Linux and Windows (MINGW) host platforms.

The version of installer used by Imperas for the generation of this example was 4.2-129

12.4.2 Cross Compiler Make Environment

The toolchain is extracted into a directory mips-4.2, we add this onto our path

```
export PATH=${PATH}:/path/to/installation/mips-4.2/bin
```

We can then use a Makefile to build our user application. The following Makefile uses the cross compiler to generate an exe file from a source c file, led_flash.c. It also generates the symbol table and an assembler listing (objdump) for use in debug.

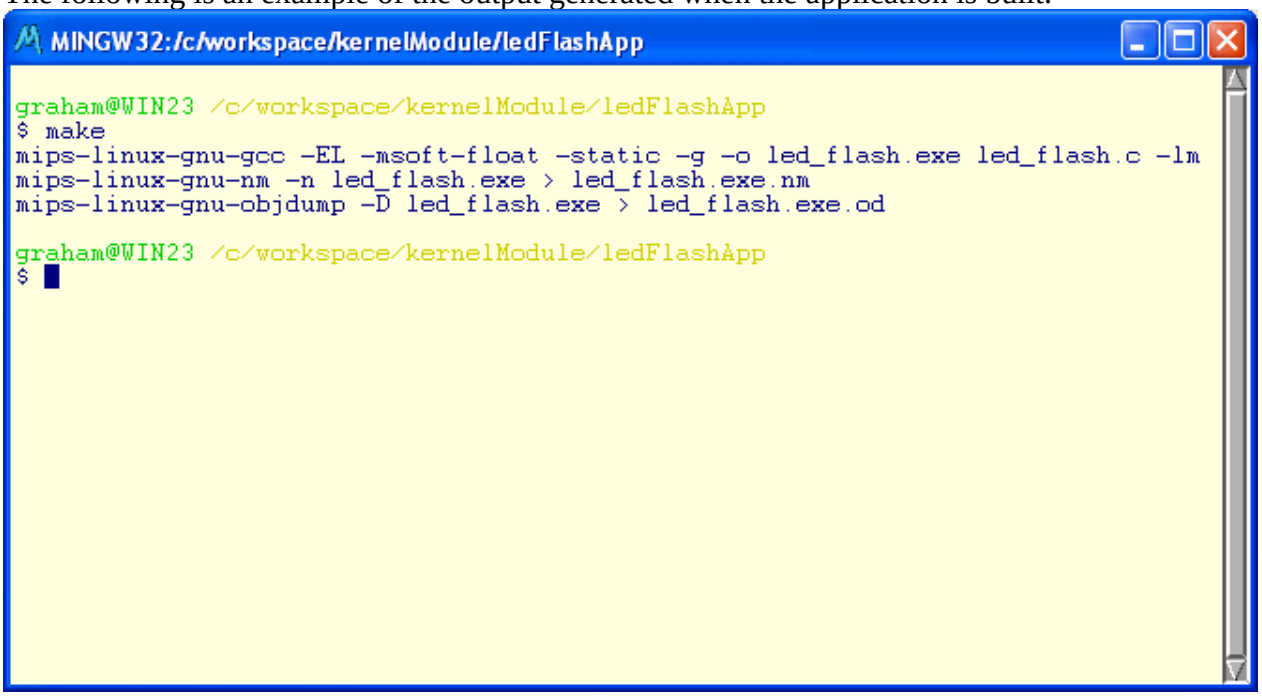
```
CFLAGS = -EL -msoft-float -static -g
LDFLAGS= -lm
CC = mips-linux-gnu-gcc $(CFLAGS)
NM = mips-linux-gnu-nm -n
OD = mips-linux-gnu-objdump -D

ALL = led_flash.exe

all: $(ALL)

%.exe: %.c
       $(CC) -o $@ $(<) $(LDFLAGS)
       $(NM) $@ > $(@).nm
       $(OD) $@ > $(@).od
```

```
clean:
	rm -f $(ALL)
	rm -f $(ALL).nm
	rm -f $(ALL).od
```

12.4.3 Building the Application

We can now simply build the application into an executable ready to be loaded onto the MipsMalta platform and executed by typing make at the command line.

The following is an example of the output generated when the application is built.

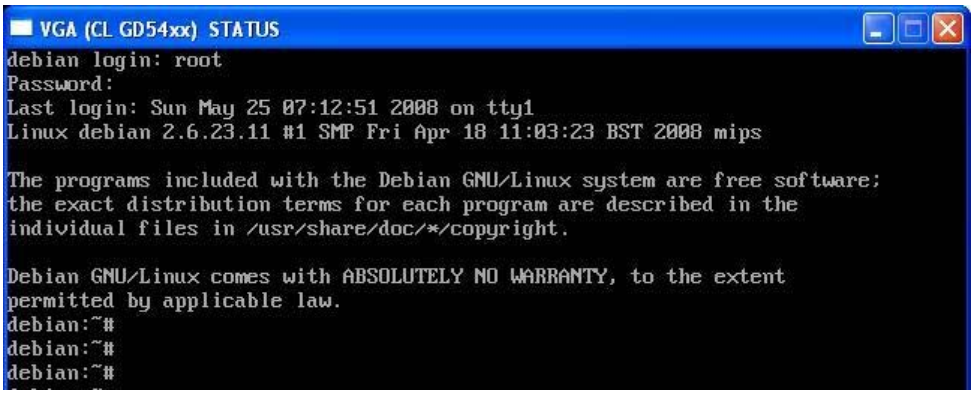


This application can now be transferred into the MIPS Malta Linux simulation and executed. See the following section that takes you through this process.

12.5 Adding Additional Linux Components

The *gdb* component is required in the Linux operating system to enable debugging of the user application. This includes *gdb server* to allow the connection of a debugger running on the native host. This component will be installed using apt-get and will require root login in the simulated Linux environment.

Login as root



Install the gdb package

```
debian:~# apt-get install gdb
Reading package lists... Done
Building dependency tree... Done
Suggested packages:
  gdb-doc
The following NEW packages will be installed:
  gdb
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 3491kB of archives.
After unpacking 9601kB of additional disk space will be used.
Get:1 http://mirror.ox.ac.uk etch/main gdb 6.4.90.dfsg-1 [3491kB]
Fetched 3491kB in 6s (538kB/s)
Selecting previously deselected package gdb.
(Reading database ... 15755 files and directories currently installed.)
Unpacking gdb (from .../gdb_6.4.90.dfsg-1_mipsel.deb) ...
Setting up gdb (6.4.90.dfsg-1) ...

debian:~#
```

now logout as root using the ***exit*** command.

12.6 Remote Debug using GDBserver

12.6.1 Starting GDB Server

Run the program, under gdbserver, using the port on the localhost that has been mapped onto the host machine port of the same number.

```
% gdbserver localhost:11001 ./linpack.exe
```

```
user1@debian:~$ gdbserver localhost:11001 ./linpack.exe &
[1] 2281
user1@debian:~$ Process ./linpack.exe created; pid = 2282
Listening on port 11001

_
```

12.6.2 Connecting to GDB Server

12.6.2.1 Using Eclipse

TBA

12.6.2.2 Using DDD from a Linux Host

In this example we are using a version of mips gdb, built for a Linux host, to connect to the gdbserver running in the OVPsim simulated Linux environment, on a Windows based PC.

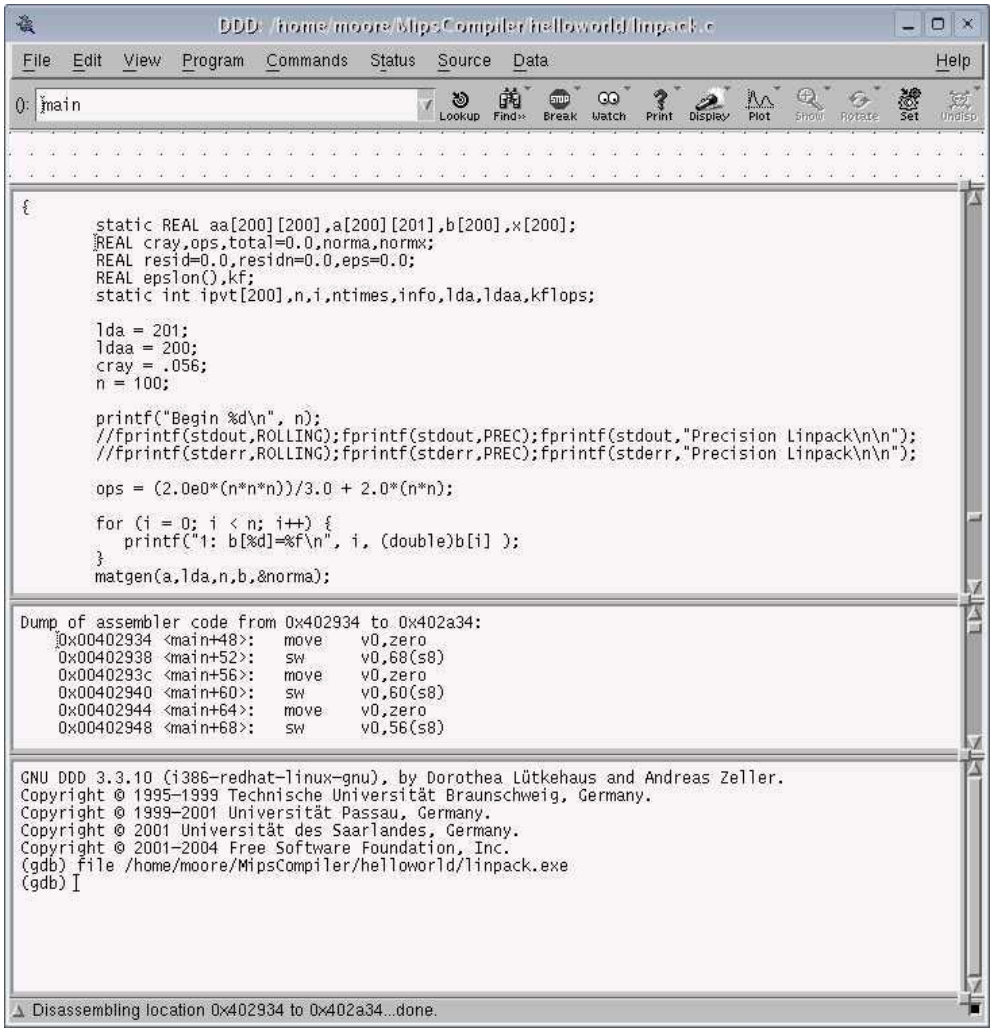
On any Linux machine start a debug session using command line gdb or gui ddd (must specify the specific gdb version to use so that it is compatible with the gdb server installation in the simulated Linux environment).

```
% ddd --debugger gdb6.4/install/bin/mipsel-linux-gnu-gdb
```

This will start the DDD GUI.

In the command prompt the file to be debugged should be loaded using the command

```
(gdb) file linpack.exe
```



We should now be able to connect to the port on the Windows host on which the simulation is running

```
(gdb) target remote win23:11001
```

12.6.2.3 Using GDB from local Windows Host

In this example we are using a version of mips gdb, built for a Windows host, to connect to the gdbserver running in the OVPsim simulated Linux environment, on a Windows based PC.

On the local Windows machine start a debug session using command line gdb.

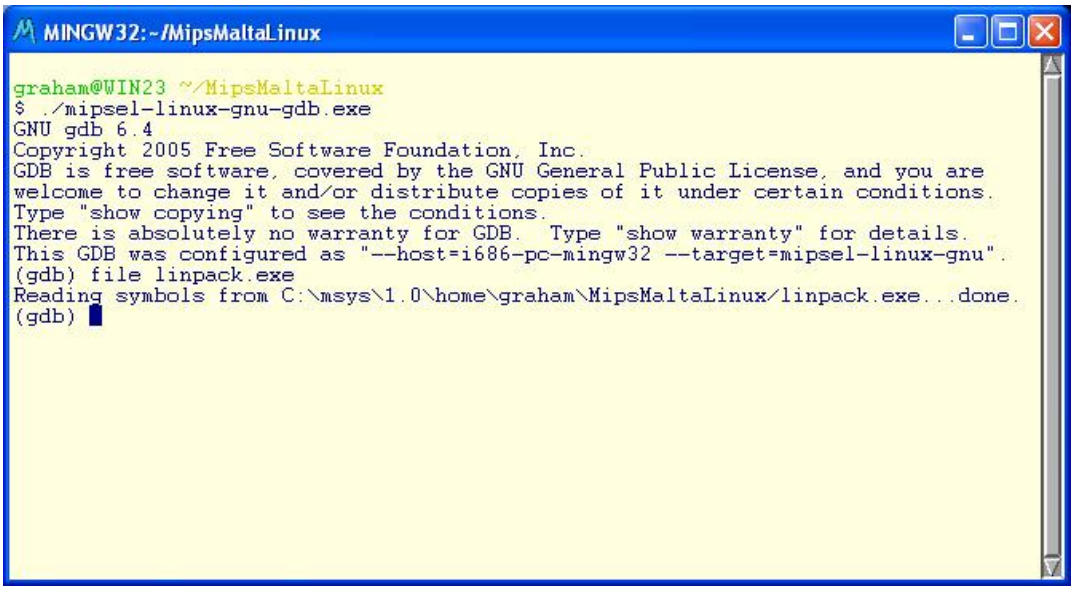
```
% ./mipsel-linux-gnu-gdb
```

This will start the GDB command line debugger.

In the command prompt the file to be debugged should be loaded using the command

```
(gdb) file linpack.exe
```

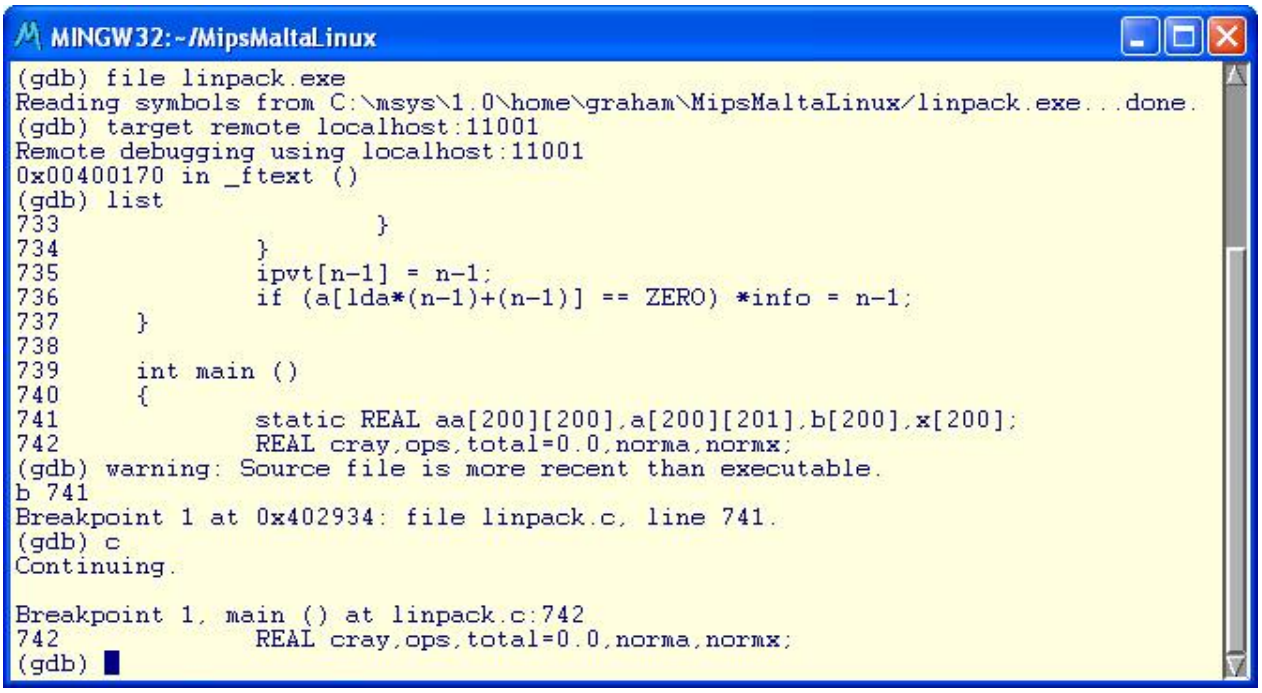
We should now be able to connect to the port on the Windows host on which the simulation is running




```
(gdb) target remote localhost:11001
```

```
(gdb) target remote localhost:11001
Remote debugging using localhost:11001
0x00400170 in _ftext ()
(gdb)
```

We can now carry out normal debugging operations, such as creating and running to breakpoints.



12.6.3 Debugging Multiple Applications in Parallel

Each application instance to be debugged will require:

1. a port to be exported from the simulation,
2. a gdb server instance in the simulated Linux environment
3. a ddd/gdb debugging instance on a Host machine

In order to export additional ports the platform.c file will need to be modified to change the attribute passed to the Ethernet NIC peripheral. This is passed using the icm function in the platform, as shown below

```
icmAddStringAttr(nicAttrs, "redir", "tcp:11001:10.0.2.15:11001");

    icmPseP nic = icmNewPSE("PCI_NET", nicPse, nicAttrs,
                            nicIntercept, "modelAttrs");
```

To add additional ports the string is constructed as provided in the following example that exports two ports.

```
    icmAddStringAttr(nicAttrs, "redir","tcp:11001:10.0.2.15:11001,
tcp:11002:10.0.2.15:11002");
```

12.7 Debug using Simulated GDB

Debugging can also be carried out using a GDB session running within the simulated Guest Linux.

In this case the source code should also be transferred onto the simulated platform using the FTP or TFTP methods described earlier. Start GDB using the command

```
> gdb linpack.exe
```

```
debian:/home/user1# ls -l
total 672
-rw-r--r-- 1 root root  22340 2008-05-25 17:46 linpack.c
-rwxr-xr-x 1 root root 655574 2008-05-25 17:47 linpack.exe
debian:/home/user1# gdb linpack.exe
GNU gdb 6.4.90-debian
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "mipsel-linux-gnu"...Using host libthread_db library
"/lib/libthread_db.so.1".

(gdb)
```

We are now ready to start debugging the application. We shall set a breakpoint on main and run the application.

```
(gdb) b main
Breakpoint 1 at 0x402934: file linpack.c, line 742.
(gdb) run
Starting program: /home/user1/linpack.exe

Breakpoint 1, main () at linpack.c:742
742             REAL cray,ops,total=0.0,norma,normx;
(gdb) list
737     }
738
739     int main ()
740     {
741             static REAL aa[200][200],a[200][201],b[200],x[200];
742             REAL cray,ops,total=0.0,norma,normx;
743             REAL resid=0.0,residn=0.0,eps=0.0;
744             REAL epslon(),kf;
745             static int ipvt[200],n,i,ntimes,info,lda,ldaa,kflops;
746
(gdb) _
```

We can use all the expected GDB commands, for example add breakpoints, single step, examine registers etc

```
689             *info = 0;
(gdb) where
#0  dgefa (a=0x4826cc, lda=201, n=100, ipvt=0x481d6c, info=0x481d5c)
    at linpack.c:689
#1  0x00402bf8 in main () at linpack.c:765
(gdb) list
684     /*     gaussian elimination with partial pivoting       */
685
686     #if(DEBUG)
687             printf("->dgefa\n");
688     #endif
689             *info = 0;
690             nm1 = n - 1;
691             if (nm1 >=  0) {
692                     for (k = 0; k < nm1; k++) {
693                             kp1 = k + 1;
(gdb) i r
          zero       at       v0       v1       a0       a1       a2       a3
 R0   00000000 2aaa8013 00480000 000000c9 004826cc 000000c9 00000064 00481d6c
            t0       t1       t2       t3       t4       t5       t6       t7
 R8   ffffffff ffffffff ffffffff 00000000 00000004 0000000f 07070707 ffffffff
            s0       s1       s2       s3       s4       s5       s6       s7
 R16  00000063 41245855 00532808 00000000 00406e10 00406e7c 00532808 00000000
            t8       t9       k0       k1       gp       sp       s8       ra
 R24  00000000 00425f74 00000000 00000000 00489b10 7ffc77a0 7ffc77a0 00402bf8
            sr       lo       hi      bad    cause       pc
      0000a413 00000000 00000000 0047bf40 00800024 00402494
           fsr      fir
      00000000 00000000
(gdb) _
```

13 Creating an Example Linux Kernel Module

13.1 Introduction

This section provides some information to help with the creation of new kernel driver modules and how these can be developed on the simulated platform.

The example in this section is a character device driver for the 8 character LED that is defined as part of the Malta FPGA device.

13.2 Pre-Requisites

There are pre-requisites to being able to build a driver kernel module, namely

- source of the linux kernel
- cross compiler for building the linux kernel

These were previously discussed in the section 9.2.1 “Building the Cross-Compiler” and section 9.2.2 “Building the Linux Kernel”. Please see these sections for instructions if they have not already been performed.

13.3 Starting the Platform Simulation

The MipsMalta platform should be started with additional command line arguments, as used previously, to export a TCP port (redir) and tftp root (tftp) to allow easy transfer of files into the Guest Operating System and run in real time (wallclock).

Using one of the example script files it should be modified as shown below (for Windows)

```
%PLATFORM_VLNV% ^
        --verbose ^
        --ramdisk initrd.gz ^
        --finishonhalt ^
        --kernel vmlinux ^
        --console tty0 ^
        --output imperas.log ^
        --wallclock ^
        --redir  ^
        --tftp “C:\workspace\kernelModule”
```

13.4 Example Source

This section contains example code for a simple device driver. The device driver allows the LED register in the Malta FPGA to be accessed for reading and writing.

13.4.1 Included Header Files

The header files that are used in the Kernel device driver

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/fs.h>        /* character device definitions    */
```

This next header file is used to define the register in the MIPS Malta FPGA that is used to access the LEDs.

```
#include "mipsMalta.h"
```

13.4.2 Callback Table and Functions

13.4.2.1 Table

A table provides the list of functions provided by this device driver.

```
struct file_operations ops =
{
       .owner    =  THIS_MODULE,
       .open     =  led_open,
       .release  =  led_release,
       .read     =  led_read,
       .write    =  led_write
};
```

13.4.2.2 Device Open

```
/*
 * called whenever a process attempts to open the device
 */
static int led_open (struct inode *inode, struct file *file)
{
       printk ("led_open: %d.%d\n", MAJOR(inode->i_rdev), MINOR(inode->i_rdev));
       return 0 ;
}
```

13.4.2.3 Device Close

```
/*
 * Called when a process closes the device.
 */
static int led_release (struct inode *inode, struct file *file)
{
       printk ("led_release\n");
       return 0;
}
```

13.4.2.4 Device Read

```
/*
 * read entry point:
 */
static ssize_t led_read (struct file *file,
                char *buffer,       /* The buffer to fill with data */
                size_t length,      /* The length of the buffer */
                loff_t * offset)    /* Our offset in the file */
{
       unsigned char c;

       /* Read from physical LED register */
       c = *(char *)LED_REG;

       printk ("led_read: %x\n", (unsigned int)c);

       if (put_user (c, buffer) != 0)
              return -EFAULT;

       return 1;        /* read() always returns exactly one byte */
}
```

13.4.2.5 Device Write

```
/*
 * write entry point:
 */
static ssize_t led_write (struct file *file,
                 const char *buffer,   /* buffer */
                 size_t length,        /* length of buffer */
                 loff_t * offset)      /* offset in the file */
{
       unsigned char c;

       if (get_user (c, buffer) != 0)
              return -EFAULT;

       printk ("led_write: %x\n", (unsigned int)c);

       /* Write to physical LED register */
       *(char *)LED_REG = c;

       return 1;
}
```

13.4.3 Device Module Initialize and Exit

13.4.3.1 Device Initialization

This is the function invoked when the kernel module is installed.

```
/*
 * Initialize the driver - Register the character device
 */
static int led_init (void)
{
       int ret;

       printk ("LED driver: hello\n") ;
       ret = register_chrdev (LED_MAJOR, DEVICE_NAME, &ops);
       if (ret != 0)
              return ret;
       return 0;
}
```

13.4.3.2 Device Exit

This is the function invoked when the kernel module is removed.

```
void led_exit (void)
{
	printk ("LED driver: goodbye\n");
	unregister_chrdev (LED_MAJOR, DEVICE_NAME);
}
```

13.4.3.3 Register Module

Define module functions used for initialization and exiting

```
module_init (led_init);
module_exit (led_exit);
```

13.5 Building the Device Driver

13.5.1 Makefiles

For convenience we are using two Makefiles. The first Makefile sits in the directory containing the driver source code and sets the object module to the driver name object.

```
obj-m := led_drv.o
```

The second is common to all kernel modules and specifies the module to be compiled, using M=’*mymoduledirectory*’

```
COMPILER=/path/to/compiler/install
CROSS_COMPILE_V=${COMPILER}/bin/mipsel-malta-linux-
KERNELSRC=/path/to/source/linux-2.6.23.11
MYMODULE=$(shell pwd)/ledDrvModule

default:
	make CROSS_COMPILE=$(CROSS_COMPILE_V) -C $(KERNELSRC) M=$(MYMODULE) modules
```

Of course these can be combined into a single Makefile that would reside in the directory with the source files

```
COMPILER=/path/to/compiler/install
CROSS_COMPILE_V=${COMPILER}/bin/mipsel-malta-linux-
KERNELSRC=/path/to/source/linux-2.6.23.11

obj-m += led_drv.o

default:
	make CROSS_COMPILE=$(CROSS_COMPILE_V) -C $(KERNELSRC) modules SUBDIRS=$(PWD)
```

13.5.2 Building

The tools available allow the driver to be built on a Linux host machine, in the same way that the Linux kernel is re-built only on a Linux host.

Building the example kernel module using the Makefile, shown above, will provide output similar to that below:

13.6 Installing the Kernel Module

The file created is a .ko file and this should be transferred into the MIPS Malta Linux platform in the same way as a user application is transferred, as described in section 11 “Transferring Files to Guest Linux”.

The kernel module can only be installed from the root account.

First we use **mknod** to make a directory entry and corresponding i-node for a special file.

```
mknod /dev/led c 42 0
```

We provide the name of the device as /dev/led, c defines the device as a character device, the major device number 42 that helps the operating system find the device driver code and the minor device number 0 used to select the line number. The Major number 42 is reserved for examples and demos.

> ⇒ the Major number used when the entry is created corresponds to the value used in the driver source code during the initialization, as the argument to the function *register_chrdev()*, see 13.4.3.1 “Device Initialization”

There will now be an entry in /dev called led.

We must make this file writable for all accounts

```
chmod 666 /dev/led
```

We now use **insmod** to install our device driver

```
insmod led_drv.ko
```

When installed the driver initialization function is executed.

The following screenshot shows the installation of the led_driver on the MIPS Malta Linux platform simulation.

```
user1@debian:~$
user1@debian:~$ su root
Password:
debian:/home/user1# mknod /dev/led c 42 0
debian:/home/user1# ls -l /dev/led
crw-r--r-- 1 root root 42, 0 2008-05-25 13:34 /dev/led
debian:/home/user1# chmod 666 /dev/led
debian:/home/user1# ls -l /dev/led
crw-rw-rw- 1 root root 42, 0 2008-05-25 13:34 /dev/led
debian:/home/user1# insmod led_drv.ko
LED driver: hello
debian:/home/user1# exit
exit
user1@debian:~$
```

13.7 Kernel Module Test Application

A simple test application is created to test the operation of the driver. This will open the file that was created with mknod to gain access to the device.

```
    const char *deviceName = "/dev/led";

    if ((fh = open (deviceName, O_WRONLY)) < 0) {
        char error[64] ={"open "};
        strncat(error, deviceName, strlen(deviceName));
        perror (error);
        return 0;
    }
```

It will then write characters to the device. This is a sequence comprising a walking ‘1’ interspersed with a write of zero. usleep is used to create a delay between each write.

```
    int repeat=25;
    while (repeat--)
    {
        c = 0;
        write (fh, &c, 1);
        usleep (500000);
        c = 1 << (repeat %8);
        write (fh, &c, 1);
        usleep (500000);
    }
```

When complete the device is closed

```
    close (fh);
```

This is built using the same Makefile and Cross Compiler toolchain as described in section 12.4 “Compiling a User Program”.

The .exe file created should be transferred into the MIPS Malta Linux platform in the same way as a user application is transferred, as described in section 11 “Transferring Files to Guest Linux”.

We can now test our application and driver in conjunction with the peripheral device.

When running the application, it first opens the device and then writes characters. The device driver has been written including with debug printf statements which are seen in the console.

```
user1@debian:~$
user1@debian:~$ ./led_flash.exe
led_open: 42.0
led_write
led_write: 0
led_write: done
led_write
led_write: 1
led_write: done
led_write
led_write: 0
led_write: done
led_write
led_write: 80
led_write: done
```

When complete the device is closed.

```
led_write: 2
led_write: done
led_write
led_write: 0
led_write: done
led_write
led_write: 1
led_write: done
led_release
user1@debian:~$
```

14 Debugging a Statically Linked Linux Kernel Module

As the kernel module is statically linked into the kernel all symbolic information for the kernel module will be available.

This allows the module to be debugged in the same way as a kernel can be debugged, see section 10 ”Debugging the Linux Kernel”

15 Debugging a Dynamically loaded Linux Kernel Module

It is difficult to debug a Linux Kernel Module (LKM) especially when it is dynamically loaded from a compiled relocatable object because it resides in virtual memory. This means there are no symbols and no information about where the module resides in memory.

However, the Imperas Multiprocessor Debugger provides functionality to allow source level debug of LKMs without any changes to the original Linux kernel.

15.1 Pre-Requisites

You will need to have obtained and installed a version of the Imperas tools and the demo installers.

These can be made available from the user area on www.imperas.com by contacting Imperas at info@imperas.com.

Two demonstrations are provided by Imperas, for registered Imperas users, to show both the capabilities to debug a Linux kernel, including a dynamically loaded kernel module (LKM) and the use of the Imperas Verification, Analysis and Profiling (VAP) tools; these are Imperas_idebug_linux_MipsMalta and Imperas_ivap_linux_MipsMalta respectively.

These demo installers include a version of the MIPS Malta Linux Kernel with symbolic information.

15.2 Starting the Platform Simulation

The Imperas simulator is invoked using *Imperas.exe*. the platform may be loaded from the VLNV library or by loading the platform shared object or Dynamic Link Library.

> ⇒ In the following please update the path to the working directory if using the TFTP method for transferring files into the guest operating system.

15.2.1 Invoking with VLNV Library

Within the VLNV library is a description of the standard MipsMalta platform. This can be selected by the Imperas front end and overrides applied to make configure changes to the platform.

The invocation of Imperas with the MipsMalta platform is shown below

```
imperas.exe --verbose ^
            --vlnvname MipsMalta --vlnvvendor "mips.ovpworld.org" ^
            --objfile vmlinux  --reset 0xbfc00000 ^
            --output imperas.log ^
            --approxtimer  --wallclockFactor 3 ^
            --override MipsMalta/Core_Board_SDRAM_promInit/initrd=initrd.gz ^
```

```
            --override MipsMalta/mipsle1/variant=24KEc ^
            ^
            --enabletools ^
            --extlib MipsMalta/mipsle1=linuxOsHelper ^
            ^
            --idebug ^
```

This basic invocation is provided in the Imperas_idebug_linux_MipsMalta demo directory in the script DEBUG_lkm_linux_mipsmalta. This will run scripts to guide you through the LKM debug, if you wish to manually enter the commands remove the line

to remove the replay of commands in the simulated linux

```
--override MipsMalta/Ps2Control/replay=Ps2ControlInput.rec
```

to remove the debug commands

```
--batch lkm_debug_linux_mipsmalta.tcl
```

The following additions / changes are made in order to provide the same features that were enabled previously

Change the boot method to boot from disk image

```
--override MipsMalta/Core_Board_SDRAM_promInit/initrd=initrd.gz ^
```

becomes

```
--override MipsMalta/Core_Board_SDRAM_promInit/root="/dev/hda1" ^
```

And add drive, redirection and directory information

```
--override MipsMalta/PIIX4-IDE/Drive0Name="mipsel_hda" ^
--override
MipsMalta/PCI_NET/redir="tcp:15901:10.0.2.15:15901,tcp:11001:10.0.2.15:11001" ^
--override MipsMalta/PCI_NET/tftpPrefix="<path\to\driver\working\directory>"
```

15.2.2 Invoking from ICM Platform

The same ICM C platform can be loaded into Imperas as a shared object (Linux) or Dynamic Link Library (Windows). When the platform source file is built to an executable using the standard build system it also produces the shared object.

The shared object is loaded by the argument **--icmobject** and the arguments are passed by the argument **--icmargv**.

This is loaded into the Imperas executable and the same arguments are passed

```
imperas.exe --verbose ^
            --icmobject OVPsim_linux_MipsMalta.dll ^
            --icmargv vmlinux --icmargv finishonhalt --icmargv wallclock ^
            --icmargv tftp --icmargv "C:\workspace"
            --output imperas.log ^
            --approxtimer  --wallclockFactor 3 ^
            ^
            --extlib MipsMalta/mipsle1=mips32CpuHelper ^
            --extlib MipsMalta/mipsle1=vapHelper ^
```

```
            --extlib MipsMalta/mipsle1=vapTools ^
            --extlib MipsMalta/mipsle1=linuxOsHelper ^
            ^
            --idebug ^
```

⇒ Note that when loading an ICM object the Imperas simulator cannot initially determine the components it contains. This means that the --enabletools cannot be used to load all required tool libraries and they must be loaded individually on the command line.

15.2.3 Debug Startup

The debugger will load the platform and then wait at the debug prompt for input

```
Info (ICM_AL) Found attribute symbol 'modelAttrs' in file 'C:/Imperas/lib/Windows/ImperasLib/imperas.com/peripheral/Ps2C
ontrol/1.0/model.dll'
Info (ICM_AL) Found attribute symbol 'modelAttrs' in file 'C:/Imperas/lib/Windows/ImperasLib/imperas.com/peripheral/RtcM
C146818/1.0/model.dll'
Info (ICM_AL) Found attribute symbol 'modelAttrs' in file 'C:/Imperas/lib/Windows/ImperasLib/imperas.com/semihosting/pse
sockets/1.0/model.dll'
Info (MIPS32_VM_VMIINIT)  Setting up Virtual Mapping
Info (MIPS32_TLBTLBH2) Using 128 hidden TLB entries
Info (OR_OF) Target '/MipsMaltaLinux/mipsle1' has object file read from 'C:/workspace/OVPLinuxInstall/MipsMaltaLinuxInst
all/vmlinux'
Info (OR_SH)     Section        flg  sect addr   size       load addr   file offset
Info (OR_SD)     .text          -ax  0x80100000  0x002d6564  0x80100000  0x00002000
Info (OR_SD)     __ex_table     -a-  0x803d6570  0x00001950  0x803d6570  0x002d8570
Info (OR_SD)     .rodata        -a-  0x803d8000  0x00065be4  0x803d8000  0x002da000
Info (OR_SD)     .pci_fixup     -a-  0x8043dbe4  0x00000558  0x8043dbe4  0x0033fbe4
Info (OR_SD)     __ksymtab      -a-  0x8043e13c  0x00004970  0x8043e13c  0x0034013c
Info (OR_SD)     __ksymtab_gpl  -a-  0x80442aac  0x00001310  0x80442aac  0x00344aac
Info (OR_SD)     __kcrctab      -a-  0x80443dbc  0x000024b8  0x80443dbc  0x00345dbc
Info (OR_SD)     __kcrctab_gpl  -a-  0x80446274  0x00000988  0x80446274  0x00348274
Info (OR_SD)     __ksymtab_strings -a-  0x80446bfc  0x0000d5c8  0x80446bfc  0x00348bfc
Info (OR_SD)     __param        -a-  0x804541c4  0x00000668  0x804541c4  0x003561c4
Info (OR_SD)     .data          wa-  0x80456000  0x0001b220  0x80456000  0x00358000
Info (OR_SD)     .data.cacheline_aligned wa-  0x80472000  0x00006360  0x80472000  0x00374000
Info (OR_SD)     .init.text     -ax  0x80479000  0x00026a58  0x80479000  0x0037b000
Info (OR_SD)     .init.data     wa-  0x8049fa58  0x00005394  0x8049fa58  0x003a1a58
Info (OR_SD)     .init.setup    wa-  0x804a4df0  0x000003c0  0x804a4df0  0x003a6df0
Info (OR_SD)     .initcall.init wa-  0x804a51b0  0x00000264  0x804a51b0  0x003a71b0
Info (OR_SD)     .con_initcall.init wa-  0x804a5414  0x00000008  0x804a5414  0x003a7414
Info (OR_SD)     .exit.text     -ax  0x804a541c  0x00000d84  0x804a541c  0x003a741c
Info (OR_SD)     .init.ramfs    -a-  0x804a7000  0x00000086  0x804a7000  0x003a9000
Info (OR_SD)     .data.percpu   wa-  0x804a8000  0x00001960  0x804a8000  0x003aa000
idebug (mipsle1) > _
```

At the initial prompt type continue (or c) to start the simulation. After the Linux operating system has booted we return control to the debugger by interrupting with control-C.

15.3 Registering Kernel Module with Debugger

The development provides a command, ***debugKernelModule***, which allows the name of a kernel module that will be loaded sometime in the future to be registered with the debugger. When the kernel module is loaded, using the insmod command, a breakpoint will be set at the start of the function defined with module_init.

The kernel module is loaded at an arbitrary kernel address and fixed up for execution by the kernel during the loading process. This requires that the kernel is built to be relocatable and as such there is no symbol table available for the loaded kernel module.

When the debugger hits the breakpoint created on the code executed by the insmod command, the debugger introspects the kernel to discover the module load address and creates a symbol table.

⇒ The .ko and .c file of the kernel module must be available in the directory from which the simulation was started

This allows symbolic debug of the loaded kernel module; for example adding breakpoints onto the other functions within the kernel driver module by name.

15.4 Debugging the Kernel Module

This section assumes we are using the same kernel module that was previously explained in section 13 "Creating an Example Linux Kernel Module"

15.4.1 Register Kernel Module for Debugging

We need to register the kernel modules we are interested in debugging with the debugger. This is done using the ***debugKernelModule*** command. The command has been added into the tcl command space of the debugger under the debugger namespace and so is accessed through the tcl command window. This can be done from the debug window by prefixing the command with the keyword 'tcl' or the tcl global namespace prefix '::'.

```
tcl lkmbreakonload -module led_drv
```

or

```
::lkmbreakonload -module led_drv
```

⇒ The command has two arguments: the module name and the file containing the module. If the file argument is omitted it is assumed the file is called <module name>**.ko**.

15.4.2 Installing the Kernel Module

Here we will look at what happens now when we install the led_drv kernel module example in the guest linux operating simulation.

Use the same commands as previously used to install the kernel module when logged in as root in the guest operating system.

```
mknod /dev/led c 42 0
chmod 666 /dev/led
insmod led_drv.ko
```

This time the insmod command does not return to the prompt. This is because the debugger has detected the loading of the kernel module led_drv that was previously registered for debugging and stopped the simulation. You will notice also that the symbol table is generated.

```
idebug (mipsle1) > c
add symbol table from file "led_drv.ko" at
        .text_addr = 0xc0039000
        __ksymtab_addr = 0x0

Breakpoint 4 at *0xc0039184 triggered for processor mipsle1

led_init () at /home/graham/playground/kernelModule/ledDriver/ledDrvModule/led_drv.c:117
117     {
idebug (mipsle1) > list
112
113     /*
114      * Initialize the driver - Register the character device
115      */
116     static int led_init (void)
117     {
118             int ret;
119
120             printk ("LED driver: hello\n") ;
121             ret = register_chrdev (LED_MAJOR, DEVICE_NAME, &ops);
idebug (mipsle1) >
```

We have stopped at the start of the init function. We are now able to debug the init function and also add breakpoints to other functions of interest, watchpoints on variables etc

15.4.3 Adding Breakpoints

The example application that uses the led_drv driver module writes values to the LED. We can apply a breakpoint on the led_write function in the driver to debug this

```
idebug (mipsle1) > b led_write
Created breakpoint 5 at 0xc00390f0: file /home/graham/playground/kernelModule/ledDriver/ledDrvModule/led_drv.c, line 102

idebug (mipsle1) >
```

We can use all the standard debug features to debug the kernel module, for example breakpoints, watchpoints, single stepping, examining registers etc.

15.4.4 Source Level Debug

We are able to debug the kernel driver at source level.

When we continue the simulation with the breakpoint set on led_write, we hit a breakpoint as soon as the led_flash.exe test application is executed.

```
Breakpoint 5 at led_write triggered for processor mipsle1

led_write (file=0x811aba40, buffer=0x811aba40 "`\367c\207\220h\354\207 \337\022\201\024\004\254\206\260\231\003
, length=1, offset=0x8641bf18) at /home/graham/playground/kernelModule/ledDriver/ledDrvModule/led_drv.c:102
102             printk ("led_write\n");
idebug (mipsle1) > list
97                      loff_t * offset)      /* offset in the file */
98      {
99              unsigned char c;
100     //      int ret;
101
102             printk ("led_write\n");
103             if (get_user (c, buffer) != 0)
104                     return -EFAULT;
105             printk ("led_write: %x\n", (unsigned int)c);
106             *(char *)LED_REG = c;
idebug (mipsle1) >
```

16 Restrictions and Caveats

The platform is tested only under the following conditions:

- Little endian memory organization
- Debian distribution of Linux kernel version 2.6.23.11

APPENDIX A

A.1 Platform Command Line Arguments

Argument	Parameter	Description
P	integer	expose a port on which a debugger can be attached
T	integer	start tracing after integer number of instructions have be performed by the MIPS processor
initrd	none	boot using the initial ramdisk
finishonhalt	none	cause the simulation to be stopped when a soft reset is requested during the halt operation
wallclock	none	Simulate using wall clock time i.e. simulation will not run faster than real time
redir	none	redirection of tcp ports to allow tcp connection with another machine for file transfers etc redirection string is tcp:15901:10.0.2.15:5901,tcp:11001:10.0.2.15:11001
tftp	String	Enable the emulated TFTP server and specify the root directory in the host machine
stopafter	double	stop execution after an integer number of instructions has been performed on the MIPS processor.
nographics	none	do not use SDL for the output graphics
variant	string	define the MIPS32 processor variant to be modeled, for example 4KEc, 24K etc
bootimage	none	define the boot loader file to be loaded and used.

APPENDIX B

B.1 Some Terms Explained

Term	Description
Guest Operating System	Is the operating system that is running within the simulation
Host Operating System	Is the OS of the Host machine being used
LKM	Linux Kernel Module
Kernel Module	A kernel extension ‘driver’ that executes as part of the kernel, often dynamically loaded

APPENDIX C

C.1 MIPS Kernel Configuration File

```
#
# Automatically generated make config: don't edit
# Linux kernel version: 2.6.23.11
# Thu Feb 14 13:59:24 2008
#
CONFIG_MIPS=y

#
# Machine selection
#
CONFIG_ZONE_DMA=y
# CONFIG_MACH_ALCHEMY is not set
# CONFIG_BASLER_EXCITE is not set
# CONFIG_MIPS_COBALT is not set
# CONFIG_MACH_DECSTATION is not set
# CONFIG_MACH_JAZZ is not set
# CONFIG_LEMOTE_FULONG is not set
# CONFIG_MIPS_ATLAS is not set
CONFIG_MIPS_MALTA=y
# CONFIG_MIPS_SEAD is not set
# CONFIG_MIPS_SIM is not set
# CONFIG_MARKEINS is not set
# CONFIG_MACH_VR41XX is not set
# CONFIG_PNX8550_JBS is not set
# CONFIG_PNX8550_STB810 is not set
# CONFIG_PMC_MSP is not set
# CONFIG_PMC_YOSEMITE is not set
# CONFIG_QEMU is not set
# CONFIG_SGI_IP22 is not set
# CONFIG_SGI_IP27 is not set
# CONFIG_SGI_IP32 is not set
# CONFIG_SIBYTE_CRHINE is not set
# CONFIG_SIBYTE_CARMEL is not set
# CONFIG_SIBYTE_CRHONE is not set
# CONFIG_SIBYTE_RHONE is not set
# CONFIG_SIBYTE_SWARM is not set
# CONFIG_SIBYTE_LITTLESUR is not set
# CONFIG_SIBYTE_SENTOSA is not set
# CONFIG_SIBYTE_PTSWARM is not set
# CONFIG_SIBYTE_BIGSUR is not set
# CONFIG_SNI_RM is not set
# CONFIG_TOSHIBA_JMR3927 is not set
# CONFIG_TOSHIBA_RBTX4927 is not set
# CONFIG_TOSHIBA_RBTX4938 is not set
# CONFIG_WR_PPMC is not set
CONFIG_RWSEM_GENERIC_SPINLOCK=y
# CONFIG_ARCH_HAS_ILOG2_U32 is not set
# CONFIG_ARCH_HAS_ILOG2_U64 is not set
CONFIG_GENERIC_FIND_NEXT_BIT=y
CONFIG_GENERIC_HWEIGHT=y
```

```
CONFIG_GENERIC_CALIBRATE_DELAY=y
CONFIG_GENERIC_TIME=y
CONFIG_SCHED_NO_NO_OMIT_FRAME_POINTER=y
# CONFIG_GENERIC_HARDIRQS_NO__DO_IRQ is not set
CONFIG_ARCH_MAY_HAVE_PC_FDC=y
CONFIG_DMA_NONCOHERENT=y
CONFIG_DMA_NEED_PCI_MAP_STATE=y
CONFIG_EARLY_PRINTK=y
CONFIG_SYS_HAS_EARLY_PRINTK=y
# CONFIG_HOTPLUG_CPU is not set
CONFIG_I8259=y
CONFIG_MIPS_BONITO64=y
CONFIG_MIPS_MSC=y
# CONFIG_NO_IOPORT is not set
CONFIG_GENERIC_ISA_DMA=y
# CONFIG_CPU_BIG_ENDIAN is not set
CONFIG_CPU_LITTLE_ENDIAN=y
CONFIG_SYS_SUPPORTS_BIG_ENDIAN=y
CONFIG_SYS_SUPPORTS_LITTLE_ENDIAN=y
CONFIG_IRQ_CPU=y
CONFIG_MIPS_BOARDS_GEN=y
CONFIG_PCI_GT64XXX_PCI0=y
CONFIG_SWAP_IO_SPACE=y
CONFIG_BOOT_ELF32=y
CONFIG_MIPS_L1_CACHE_SHIFT=5

#
# CPU selection
#
# CONFIG_CPU_LOONGSON2 is not set
# CONFIG_CPU_MIPS32_R1 is not set
CONFIG_CPU_MIPS32_R2=y
# CONFIG_CPU_MIPS64_R1 is not set
# CONFIG_CPU_MIPS64_R2 is not set
# CONFIG_CPU_R3000 is not set
# CONFIG_CPU_TX39XX is not set
# CONFIG_CPU_VR41XX is not set
# CONFIG_CPU_R4300 is not set
# CONFIG_CPU_R4X00 is not set
# CONFIG_CPU_TX49XX is not set
# CONFIG_CPU_R5000 is not set
# CONFIG_CPU_R5432 is not set
# CONFIG_CPU_R6000 is not set
# CONFIG_CPU_NEVADA is not set
# CONFIG_CPU_R8000 is not set
# CONFIG_CPU_R10000 is not set
# CONFIG_CPU_RM7000 is not set
# CONFIG_CPU_RM9000 is not set
# CONFIG_CPU_SB1 is not set
CONFIG_SYS_HAS_CPU_MIPS32_R1=y
CONFIG_SYS_HAS_CPU_MIPS32_R2=y
CONFIG_SYS_HAS_CPU_MIPS64_R1=y
CONFIG_SYS_HAS_CPU_NEVADA=y
CONFIG_SYS_HAS_CPU_RM7000=y
CONFIG_CPU_MIPS32=y
CONFIG_CPU_MIPSR2=y
CONFIG_SYS_SUPPORTS_32BIT_KERNEL=y
```

```
CONFIG_SYS_SUPPORTS_64BIT_KERNEL=y
CONFIG_CPU_SUPPORTS_32BIT_KERNEL=y

#
# Kernel type
#
CONFIG_32BIT=y
# CONFIG_64BIT is not set
CONFIG_PAGE_SIZE_4KB=y
# CONFIG_PAGE_SIZE_8KB is not set
# CONFIG_PAGE_SIZE_16KB is not set
# CONFIG_PAGE_SIZE_64KB is not set
CONFIG_BOARD_SCACHE=y
CONFIG_MIPS_CPU_SCACHE=y
CONFIG_CPU_HAS_PREFETCH=y
# CONFIG_MIPS_MT_DISABLED is not set
CONFIG_MIPS_MT_SMP=y
# CONFIG_MIPS_MT_SMTC is not set
CONFIG_MIPS_MT=y
CONFIG_SYS_SUPPORTS_MULTITHREADING=y
CONFIG_MIPS_MT_FPAFF=y
# CONFIG_MIPS_VPE_LOADER is not set
CONFIG_CPU_HAS_LLSC=y
# CONFIG_CPU_HAS_SMARTMIPS is not set
CONFIG_CPU_MIPSR2_IRQ_VI=y
CONFIG_CPU_MIPSR2_IRQ_EI=y
CONFIG_CPU_HAS_SYNC=y
CONFIG_GENERIC_HARDIRQS=y
CONFIG_GENERIC_IRQ_PROBE=y
CONFIG_IRQ_PER_CPU=y
CONFIG_CPU_SUPPORTS_HIGHMEM=y
CONFIG_SYS_SUPPORTS_SMARTMIPS=y
CONFIG_ARCH_FLATMEM_ENABLE=y
CONFIG_SELECT_MEMORY_MODEL=y
CONFIG_FLATMEM_MANUAL=y
# CONFIG_DISCONTIGMEM_MANUAL is not set
# CONFIG_SPARSEMEM_MANUAL is not set
CONFIG_FLATMEM=y
CONFIG_FLAT_NODE_MEM_MAP=y
# CONFIG_SPARSEMEM_STATIC is not set
CONFIG_SPLIT_PTLOCK_CPUS=4
# CONFIG_RESOURCES_64BIT is not set
CONFIG_ZONE_DMA_FLAG=1
CONFIG_BOUNCE=y
CONFIG_VIRT_TO_BUS=y
CONFIG_SMP=y
CONFIG_SYS_SUPPORTS_SMP=y
CONFIG_NR_CPUS_DEFAULT_2=y
CONFIG_NR_CPUS=2
# CONFIG_HZ_48 is not set
CONFIG_HZ_100=y
# CONFIG_HZ_128 is not set
# CONFIG_HZ_250 is not set
# CONFIG_HZ_256 is not set
# CONFIG_HZ_1000 is not set
# CONFIG_HZ_1024 is not set
CONFIG_SYS_SUPPORTS_ARBIT_HZ=y
```

```
CONFIG_HZ=100
CONFIG_PREEMPT_NONE=y
# CONFIG_PREEMPT_VOLUNTARY is not set
# CONFIG_PREEMPT is not set
CONFIG_PREEMPT_BKL=y
# CONFIG_KEXEC is not set
CONFIG_SECCOMP=y
CONFIG_LOCKDEP_SUPPORT=y
CONFIG_STACKTRACE_SUPPORT=y
CONFIG_DEFCONFIG_LIST="/lib/modules/$UNAME_RELEASE/.config"

#
# General setup
#
CONFIG_EXPERIMENTAL=y
CONFIG_LOCK_KERNEL=y
CONFIG_INIT_ENV_ARG_LIMIT=32
CONFIG_LOCALVERSION=""
CONFIG_LOCALVERSION_AUTO=y
CONFIG_SWAP=y
CONFIG_SYSVIPC=y
CONFIG_SYSVIPC_SYSCTL=y
# CONFIG_POSIX_MQUEUE is not set
# CONFIG_BSD_PROCESS_ACCT is not set
# CONFIG_TASKSTATS is not set
# CONFIG_USER_NS is not set
# CONFIG_AUDIT is not set
# CONFIG_IKCONFIG is not set
CONFIG_LOG_BUF_SHIFT=15
# CONFIG_CPUSETS is not set
CONFIG_SYSFS_DEPRECATED=y
CONFIG_RELAY=y
CONFIG_BLK_DEV_INITRD=y
CONFIG_INITRAMFS_SOURCE=""
# CONFIG_CC_OPTIMIZE_FOR_SIZE is not set
CONFIG_SYSCTL=y
CONFIG_EMBEDDED=y
# CONFIG_SYSCTL_SYSCALL is not set
CONFIG_KALLSYMS=y
# CONFIG_KALLSYMS_EXTRA_PASS is not set
CONFIG_HOTPLUG=y
CONFIG_PRINTK=y
CONFIG_BUG=y
CONFIG_ELF_CORE=y
CONFIG_BASE_FULL=y
CONFIG_FUTEX=y
CONFIG_ANON_INODES=y
CONFIG_EPOLL=y
CONFIG_SIGNALFD=y
CONFIG_EVENTFD=y
CONFIG_SHMEM=y
CONFIG_VM_EVENT_COUNTERS=y
CONFIG_SLAB=y
# CONFIG_SLUB is not set
# CONFIG_SLOB is not set
CONFIG_RT_MUTEXES=y
# CONFIG_TINY_SHMEM is not set
```

```
CONFIG_BASE_SMALL=0
CONFIG_MODULES=y
CONFIG_MODULE_UNLOAD=y
# CONFIG_MODULE_FORCE_UNLOAD is not set
CONFIG_MODVERSIONS=y
CONFIG_MODULE_SRCVERSION_ALL=y
CONFIG_KMOD=y
CONFIG_STOP_MACHINE=y
CONFIG_BLOCK=y
# CONFIG_LBD is not set
# CONFIG_BLK_DEV_IO_TRACE is not set
# CONFIG_LSF is not set
# CONFIG_BLK_DEV_BSG is not set

#
# IO Schedulers
#
CONFIG_IOSCHED_NOOP=y
CONFIG_IOSCHED_AS=y
CONFIG_IOSCHED_DEADLINE=y
CONFIG_IOSCHED_CFQ=y
CONFIG_DEFAULT_AS=y
# CONFIG_DEFAULT_DEADLINE is not set
# CONFIG_DEFAULT_CFQ is not set
# CONFIG_DEFAULT_NOOP is not set
CONFIG_DEFAULT_IOSCHED="anticipatory"

#
# Bus options (PCI, PCMCIA, EISA, ISA, TC)
#
CONFIG_HW_HAS_PCI=y
CONFIG_PCI=y
# CONFIG_ARCH_SUPPORTS_MSI is not set
CONFIG_MMU=y

#
# PCCARD (PCMCIA/CardBus) support
#
# CONFIG_PCCARD is not set
# CONFIG_HOTPLUG_PCI is not set

#
# Executable file formats
#
CONFIG_BINFMT_ELF=y
# CONFIG_BINFMT_MISC is not set
CONFIG_TRAD_SIGNALS=y

#
# Power management options
#
# CONFIG_PM is not set

#
# Networking
#
CONFIG_NET=y
```

```
#
# Networking options
#
CONFIG_PACKET=y
CONFIG_PACKET_MMAP=y
CONFIG_UNIX=y
CONFIG_XFRM=y
CONFIG_XFRM_USER=m
# CONFIG_XFRM_SUB_POLICY is not set
# CONFIG_XFRM_MIGRATE is not set
CONFIG_NET_KEY=y
# CONFIG_NET_KEY_MIGRATE is not set
CONFIG_INET=y
CONFIG_IP_MULTICAST=y
CONFIG_IP_ADVANCED_ROUTER=y
CONFIG_ASK_IP_FIB_HASH=y
# CONFIG_IP_FIB_TRIE is not set
CONFIG_IP_FIB_HASH=y
CONFIG_IP_MULTIPLE_TABLES=y
CONFIG_IP_ROUTE_MULTIPATH=y
CONFIG_IP_ROUTE_VERBOSE=y
CONFIG_IP_PNP=y
CONFIG_IP_PNP_DHCP=y
CONFIG_IP_PNP_BOOTP=y
# CONFIG_IP_PNP_RARP is not set
CONFIG_NET_IPIP=m
CONFIG_NET_IPGRE=m
CONFIG_NET_IPGRE_BROADCAST=y
CONFIG_IP_MROUTE=y
CONFIG_IP_PIMSM_V1=y
CONFIG_IP_PIMSM_V2=y
# CONFIG_ARPD is not set
CONFIG_SYN_COOKIES=y
CONFIG_INET_AH=m
CONFIG_INET_ESP=m
CONFIG_INET_IPCOMP=m
CONFIG_INET_XFRM_TUNNEL=m
CONFIG_INET_TUNNEL=m
CONFIG_INET_XFRM_MODE_TRANSPORT=m
CONFIG_INET_XFRM_MODE_TUNNEL=m
CONFIG_INET_XFRM_MODE_BEET=y
CONFIG_INET_DIAG=y
CONFIG_INET_TCP_DIAG=y
# CONFIG_TCP_CONG_ADVANCED is not set
CONFIG_TCP_CONG_CUBIC=y
CONFIG_DEFAULT_TCP_CONG="cubic"
# CONFIG_TCP_MD5SIG is not set
CONFIG_IP_VS=m
# CONFIG_IP_VS_DEBUG is not set
CONFIG_IP_VS_TAB_BITS=12

#
# IPVS transport protocol load balancing support
#
CONFIG_IP_VS_PROTO_TCP=y
CONFIG_IP_VS_PROTO_UDP=y
```

```
CONFIG_IP_VS_PROTO_ESP=y
CONFIG_IP_VS_PROTO_AH=y

#
# IPVS scheduler
#
CONFIG_IP_VS_RR=m
CONFIG_IP_VS_WRR=m
CONFIG_IP_VS_LC=m
CONFIG_IP_VS_WLC=m
CONFIG_IP_VS_LBLC=m
CONFIG_IP_VS_LBLCR=m
CONFIG_IP_VS_DH=m
CONFIG_IP_VS_SH=m
CONFIG_IP_VS_SED=m
CONFIG_IP_VS_NQ=m

#
# IPVS application helper
#
CONFIG_IP_VS_FTP=m
CONFIG_IPV6=m
CONFIG_IPV6_PRIVACY=y
CONFIG_IPV6_ROUTER_PREF=y
CONFIG_IPV6_ROUTE_INFO=y
# CONFIG_IPV6_OPTIMISTIC_DAD is not set
CONFIG_INET6_AH=m
CONFIG_INET6_ESP=m
CONFIG_INET6_IPCOMP=m
# CONFIG_IPV6_MIP6 is not set
CONFIG_INET6_XFRM_TUNNEL=m
CONFIG_INET6_TUNNEL=m
CONFIG_INET6_XFRM_MODE_TRANSPORT=m
CONFIG_INET6_XFRM_MODE_TUNNEL=m
CONFIG_INET6_XFRM_MODE_BEET=m
# CONFIG_INET6_XFRM_MODE_ROUTEOPTIMIZATION is not set
CONFIG_IPV6_SIT=m
CONFIG_IPV6_TUNNEL=m
# CONFIG_IPV6_MULTIPLE_TABLES is not set
CONFIG_NETWORK_SECMARK=y
CONFIG_NETFILTER=y
# CONFIG_NETFILTER_DEBUG is not set
CONFIG_BRIDGE_NETFILTER=y

#
# Core Netfilter Configuration
#
CONFIG_NETFILTER_NETLINK=m
CONFIG_NETFILTER_NETLINK_QUEUE=m
CONFIG_NETFILTER_NETLINK_LOG=m
# CONFIG_NF_CONNTRACK_ENABLED is not set
# CONFIG_NF_CONNTRACK is not set
CONFIG_NETFILTER_XTABLES=m
CONFIG_NETFILTER_XT_TARGET_CLASSIFY=m
# CONFIG_NETFILTER_XT_TARGET_DSCP is not set
CONFIG_NETFILTER_XT_TARGET_MARK=m
CONFIG_NETFILTER_XT_TARGET_NFQUEUE=m
```

```
# CONFIG_NETFILTER_XT_TARGET_NFLOG is not set
# CONFIG_NETFILTER_XT_TARGET_TRACE is not set
CONFIG_NETFILTER_XT_TARGET_SECMARK=m
# CONFIG_NETFILTER_XT_TARGET_TCPMSS is not set
CONFIG_NETFILTER_XT_MATCH_COMMENT=m
CONFIG_NETFILTER_XT_MATCH_DCCP=m
# CONFIG_NETFILTER_XT_MATCH_DSCP is not set
CONFIG_NETFILTER_XT_MATCH_ESP=m
CONFIG_NETFILTER_XT_MATCH_LENGTH=m
CONFIG_NETFILTER_XT_MATCH_LIMIT=m
CONFIG_NETFILTER_XT_MATCH_MAC=m
CONFIG_NETFILTER_XT_MATCH_MARK=m
CONFIG_NETFILTER_XT_MATCH_POLICY=m
CONFIG_NETFILTER_XT_MATCH_MULTIPORT=m
# CONFIG_NETFILTER_XT_MATCH_PHYSDEV is not set
CONFIG_NETFILTER_XT_MATCH_PKTTYPE=m
CONFIG_NETFILTER_XT_MATCH_QUOTA=m
CONFIG_NETFILTER_XT_MATCH_REALM=m
CONFIG_NETFILTER_XT_MATCH_SCTP=m
CONFIG_NETFILTER_XT_MATCH_STATISTIC=m
CONFIG_NETFILTER_XT_MATCH_STRING=m
CONFIG_NETFILTER_XT_MATCH_TCPMSS=m
# CONFIG_NETFILTER_XT_MATCH_U32 is not set
# CONFIG_NETFILTER_XT_MATCH_HASHLIMIT is not set

#
# IP: Netfilter Configuration
#
CONFIG_IP_NF_QUEUE=m
CONFIG_IP_NF_IPTABLES=m
CONFIG_IP_NF_MATCH_IPRANGE=m
CONFIG_IP_NF_MATCH_TOS=m
CONFIG_IP_NF_MATCH_RECENT=m
CONFIG_IP_NF_MATCH_ECN=m
CONFIG_IP_NF_MATCH_AH=m
CONFIG_IP_NF_MATCH_TTL=m
CONFIG_IP_NF_MATCH_OWNER=m
CONFIG_IP_NF_MATCH_ADDRTYPE=m
CONFIG_IP_NF_FILTER=m
CONFIG_IP_NF_TARGET_REJECT=m
CONFIG_IP_NF_TARGET_LOG=m
CONFIG_IP_NF_TARGET_ULOG=m
CONFIG_IP_NF_MANGLE=m
CONFIG_IP_NF_TARGET_TOS=m
CONFIG_IP_NF_TARGET_ECN=m
CONFIG_IP_NF_TARGET_TTL=m
CONFIG_IP_NF_RAW=m
CONFIG_IP_NF_ARPTABLES=m
CONFIG_IP_NF_ARPFILTER=m
CONFIG_IP_NF_ARP_MANGLE=m

#
# IPv6: Netfilter Configuration (EXPERIMENTAL)
#
CONFIG_IP6_NF_QUEUE=m
CONFIG_IP6_NF_IPTABLES=m
CONFIG_IP6_NF_MATCH_RT=m
```

```
CONFIG_IP6_NF_MATCH_OPTS=m
CONFIG_IP6_NF_MATCH_FRAG=m
CONFIG_IP6_NF_MATCH_HL=m
CONFIG_IP6_NF_MATCH_OWNER=m
CONFIG_IP6_NF_MATCH_IPV6HEADER=m
CONFIG_IP6_NF_MATCH_AH=m
# CONFIG_IP6_NF_MATCH_MH is not set
CONFIG_IP6_NF_MATCH_EUI64=m
CONFIG_IP6_NF_FILTER=m
CONFIG_IP6_NF_TARGET_LOG=m
CONFIG_IP6_NF_TARGET_REJECT=m
CONFIG_IP6_NF_MANGLE=m
CONFIG_IP6_NF_TARGET_HL=m
CONFIG_IP6_NF_RAW=m

#
# Bridge: Netfilter Configuration
#
CONFIG_BRIDGE_NF_EBTABLES=m
CONFIG_BRIDGE_EBT_BROUTE=m
CONFIG_BRIDGE_EBT_T_FILTER=m
CONFIG_BRIDGE_EBT_T_NAT=m
CONFIG_BRIDGE_EBT_802_3=m
CONFIG_BRIDGE_EBT_AMONG=m
CONFIG_BRIDGE_EBT_ARP=m
CONFIG_BRIDGE_EBT_IP=m
CONFIG_BRIDGE_EBT_LIMIT=m
CONFIG_BRIDGE_EBT_MARK=m
CONFIG_BRIDGE_EBT_PKTTYPE=m
CONFIG_BRIDGE_EBT_STP=m
CONFIG_BRIDGE_EBT_VLAN=m
CONFIG_BRIDGE_EBT_ARPREPLY=m
CONFIG_BRIDGE_EBT_DNAT=m
CONFIG_BRIDGE_EBT_MARK_T=m
CONFIG_BRIDGE_EBT_REDIRECT=m
CONFIG_BRIDGE_EBT_SNAT=m
CONFIG_BRIDGE_EBT_LOG=m
CONFIG_BRIDGE_EBT_ULOG=m
# CONFIG_IP_DCCP is not set
CONFIG_IP_SCTP=m
# CONFIG_SCTP_DBG_MSG is not set
# CONFIG_SCTP_DBG_OBJCNT is not set
# CONFIG_SCTP_HMAC_NONE is not set
# CONFIG_SCTP_HMAC_SHA1 is not set
CONFIG_SCTP_HMAC_MD5=y
# CONFIG_TIPC is not set
# CONFIG_ATM is not set
CONFIG_BRIDGE=y
CONFIG_VLAN_8021Q=y
# CONFIG_DECNET is not set
CONFIG_LLC=y
# CONFIG_LLC2 is not set
# CONFIG_IPX is not set
CONFIG_ATALK=m
CONFIG_DEV_APPLETALK=m
CONFIG_IPDDP=m
CONFIG_IPDDP_ENCAP=y
```

```
CONFIG_IPDDP_DECAP=y
# CONFIG_X25 is not set
# CONFIG_LAPB is not set
# CONFIG_ECONET is not set
# CONFIG_WAN_ROUTER is not set

#
# QoS and/or fair queuing
#
CONFIG_NET_SCHED=y
CONFIG_NET_SCH_FIFO=y

#
# Queuing/Scheduling
#
CONFIG_NET_SCH_CBQ=m
CONFIG_NET_SCH_HTB=m
CONFIG_NET_SCH_HFSC=m
CONFIG_NET_SCH_PRIO=m
# CONFIG_NET_SCH_RR is not set
CONFIG_NET_SCH_RED=m
CONFIG_NET_SCH_SFQ=m
CONFIG_NET_SCH_TEQL=m
CONFIG_NET_SCH_TBF=m
CONFIG_NET_SCH_GRED=m
CONFIG_NET_SCH_DSMARK=m
CONFIG_NET_SCH_NETEM=m
CONFIG_NET_SCH_INGRESS=m

#
# Classification
#
CONFIG_NET_CLS=y
CONFIG_NET_CLS_BASIC=m
CONFIG_NET_CLS_TCINDEX=m
CONFIG_NET_CLS_ROUTE4=m
CONFIG_NET_CLS_ROUTE=y
CONFIG_NET_CLS_FW=m
CONFIG_NET_CLS_U32=m
# CONFIG_CLS_U32_PERF is not set
# CONFIG_CLS_U32_MARK is not set
CONFIG_NET_CLS_RSVP=m
CONFIG_NET_CLS_RSVP6=m
# CONFIG_NET_EMATCH is not set
CONFIG_NET_CLS_ACT=y
CONFIG_NET_ACT_POLICE=y
# CONFIG_NET_ACT_GACT is not set
# CONFIG_NET_ACT_MIRRED is not set
# CONFIG_NET_ACT_IPT is not set
# CONFIG_NET_ACT_PEDIT is not set
# CONFIG_NET_ACT_SIMP is not set
CONFIG_NET_CLS_POLICE=y
CONFIG_NET_CLS_IND=y

#
# Network testing
#
```

```
# CONFIG_NET_PKTGEN is not set
# CONFIG_HAMRADIO is not set
# CONFIG_IRDA is not set
# CONFIG_BT is not set
# CONFIG_AF_RXRPC is not set
CONFIG_FIB_RULES=y

#
# Wireless
#
# CONFIG_CFG80211 is not set
CONFIG_WIRELESS_EXT=y
# CONFIG_MAC80211 is not set
CONFIG_IEEE80211=m
# CONFIG_IEEE80211_DEBUG is not set
CONFIG_IEEE80211_CRYPT_WEP=m
CONFIG_IEEE80211_CRYPT_CCMP=m
# CONFIG_IEEE80211_CRYPT_TKIP is not set
CONFIG_IEEE80211_SOFTMAC=m
# CONFIG_IEEE80211_SOFTMAC_DEBUG is not set
# CONFIG_RFKILL is not set
# CONFIG_NET_9P is not set

#
# Device Drivers
#

#
# Generic Driver Options
#
CONFIG_STANDALONE=y
CONFIG_PREVENT_FIRMWARE_BUILD=y
CONFIG_FW_LOADER=y
# CONFIG_SYS_HYPERVISOR is not set
CONFIG_CONNECTOR=m
CONFIG_MTD=y
# CONFIG_MTD_DEBUG is not set
# CONFIG_MTD_CONCAT is not set
CONFIG_MTD_PARTITIONS=y
# CONFIG_MTD_REDBOOT_PARTS is not set
# CONFIG_MTD_CMDLINE_PARTS is not set

#
# User Modules And Translation Layers
#
CONFIG_MTD_CHAR=y
CONFIG_MTD_BLKDEVS=y
CONFIG_MTD_BLOCK=y
# CONFIG_FTL is not set
# CONFIG_NFTL is not set
# CONFIG_INFTL is not set
# CONFIG_RFD_FTL is not set
# CONFIG_SSFDC is not set

#
# RAM/ROM/Flash chip drivers
#
```

```
CONFIG_MTD_CFI=y
# CONFIG_MTD_JEDECPROBE is not set
CONFIG_MTD_GEN_PROBE=y
# CONFIG_MTD_CFI_ADV_OPTIONS is not set
CONFIG_MTD_MAP_BANK_WIDTH_1=y
CONFIG_MTD_MAP_BANK_WIDTH_2=y
CONFIG_MTD_MAP_BANK_WIDTH_4=y
# CONFIG_MTD_MAP_BANK_WIDTH_8 is not set
# CONFIG_MTD_MAP_BANK_WIDTH_16 is not set
# CONFIG_MTD_MAP_BANK_WIDTH_32 is not set
CONFIG_MTD_CFI_I1=y
CONFIG_MTD_CFI_I2=y
# CONFIG_MTD_CFI_I4 is not set
# CONFIG_MTD_CFI_I8 is not set
CONFIG_MTD_CFI_INTELEXT=y
CONFIG_MTD_CFI_AMDSTD=y
CONFIG_MTD_CFI_STAA=y
CONFIG_MTD_CFI_UTIL=y
# CONFIG_MTD_RAM is not set
# CONFIG_MTD_ROM is not set
# CONFIG_MTD_ABSENT is not set

#
# Mapping drivers for chip access
#
# CONFIG_MTD_COMPLEX_MAPPINGS is not set
CONFIG_MTD_PHYSMAP=y
CONFIG_MTD_PHYSMAP_START=0x0
CONFIG_MTD_PHYSMAP_LEN=0x0
CONFIG_MTD_PHYSMAP_BANKWIDTH=0
# CONFIG_MTD_PLATRAM is not set

#
# Self-contained MTD device drivers
#
# CONFIG_MTD_PMC551 is not set
# CONFIG_MTD_SLRAM is not set
# CONFIG_MTD_PHRAM is not set
# CONFIG_MTD_MTDRAM is not set
# CONFIG_MTD_BLOCK2MTD is not set

#
# Disk-On-Chip Device Drivers
#
# CONFIG_MTD_DOC2000 is not set
# CONFIG_MTD_DOC2001 is not set
# CONFIG_MTD_DOC2001PLUS is not set
# CONFIG_MTD_NAND is not set
# CONFIG_MTD_ONENAND is not set

#
# UBI - Unsorted block images
#
# CONFIG_MTD_UBI is not set
# CONFIG_PARPORT is not set
CONFIG_BLK_DEV=y
CONFIG_BLK_DEV_FD=m
```

```
# CONFIG_BLK_CPQ_DA is not set
# CONFIG_BLK_CPQ_CISS_DA is not set
# CONFIG_BLK_DEV_DAC960 is not set
CONFIG_BLK_DEV_UMEM=m
# CONFIG_BLK_DEV_COW_COMMON is not set
CONFIG_BLK_DEV_LOOP=m
CONFIG_BLK_DEV_CRYPTOLOOP=m
CONFIG_BLK_DEV_NBD=m
# CONFIG_BLK_DEV_SX8 is not set
CONFIG_BLK_DEV_RAM=y
CONFIG_BLK_DEV_RAM_COUNT=16
CONFIG_BLK_DEV_RAM_SIZE=4096
CONFIG_BLK_DEV_RAM_BLOCKSIZE=1024
CONFIG_CDROM_PKTCDVD=m
CONFIG_CDROM_PKTCDVD_BUFFERS=8
# CONFIG_CDROM_PKTCDVD_WCACHE is not set
CONFIG_ATA_OVER_ETH=m
CONFIG_MISC_DEVICES=y
# CONFIG_PHANTOM is not set
# CONFIG_EEPROM_93CX6 is not set
# CONFIG_SGI_IOC4 is not set
# CONFIG_TIFM_CORE is not set
CONFIG_IDE=y
CONFIG_IDE_MAX_HWIFS=4
CONFIG_BLK_DEV_IDE=y

#
# Please see Documentation/ide.txt for help/info on IDE drives
#
# CONFIG_BLK_DEV_IDE_SATA is not set
CONFIG_BLK_DEV_IDEDISK=y
# CONFIG_IDEDISK_MULTI_MODE is not set
CONFIG_BLK_DEV_IDECD=y
# CONFIG_BLK_DEV_IDETAPE is not set
# CONFIG_BLK_DEV_IDEFLOPPY is not set
# CONFIG_BLK_DEV_IDESCSI is not set
# CONFIG_IDE_TASK_IOCTL is not set
CONFIG_IDE_PROC_FS=y

#
# IDE chipset support/bugfixes
#
CONFIG_IDE_GENERIC=y
CONFIG_BLK_DEV_IDEPCI=y
# CONFIG_IDEPCI_SHARE_IRQ is not set
CONFIG_IDEPCI_PCIBUS_ORDER=y
# CONFIG_BLK_DEV_OFFBOARD is not set
CONFIG_BLK_DEV_GENERIC=y
# CONFIG_BLK_DEV_OPTI621 is not set
CONFIG_BLK_DEV_IDEDMA_PCI=y
# CONFIG_BLK_DEV_IDEDMA_FORCED is not set
# CONFIG_IDEDMA_ONLYDISK is not set
# CONFIG_BLK_DEV_AEC62XX is not set
# CONFIG_BLK_DEV_ALI15X3 is not set
# CONFIG_BLK_DEV_AMD74XX is not set
# CONFIG_BLK_DEV_CMD64X is not set
# CONFIG_BLK_DEV_TRIFLEX is not set
```

```
# CONFIG_BLK_DEV_CY82C693 is not set
# CONFIG_BLK_DEV_CS5520 is not set
# CONFIG_BLK_DEV_CS5530 is not set
# CONFIG_BLK_DEV_HPT34X is not set
# CONFIG_BLK_DEV_HPT366 is not set
# CONFIG_BLK_DEV_JMICRON is not set
# CONFIG_BLK_DEV_SC1200 is not set
CONFIG_BLK_DEV_PIIX=y
# CONFIG_BLK_DEV_IT8213 is not set
# CONFIG_BLK_DEV_IT821X is not set
# CONFIG_BLK_DEV_NS87415 is not set
# CONFIG_BLK_DEV_PDC202XX_OLD is not set
# CONFIG_BLK_DEV_PDC202XX_NEW is not set
# CONFIG_BLK_DEV_SVWKS is not set
# CONFIG_BLK_DEV_SIIMAGE is not set
# CONFIG_BLK_DEV_SLC90E66 is not set
# CONFIG_BLK_DEV_TRM290 is not set
# CONFIG_BLK_DEV_VIA82CXXX is not set
# CONFIG_BLK_DEV_TC86C001 is not set
# CONFIG_IDE_ARM is not set
CONFIG_BLK_DEV_IDEDMA=y
# CONFIG_IDEDMA_IVB is not set
# CONFIG_BLK_DEV_HD is not set

#
# SCSI device support
#
CONFIG_RAID_ATTRS=m
CONFIG_SCSI=m
CONFIG_SCSI_DMA=y
# CONFIG_SCSI_TGT is not set
CONFIG_SCSI_NETLINK=y
CONFIG_SCSI_PROC_FS=y

#
# SCSI support type (disk, tape, CD-ROM)
#
CONFIG_BLK_DEV_SD=m
CONFIG_CHR_DEV_ST=m
CONFIG_CHR_DEV_OSST=m
CONFIG_BLK_DEV_SR=m
CONFIG_BLK_DEV_SR_VENDOR=y
CONFIG_CHR_DEV_SG=m
# CONFIG_CHR_DEV_SCH is not set

#
# Some SCSI devices (e.g. CD jukebox) support multiple LUNs
#
CONFIG_SCSI_MULTI_LUN=y
CONFIG_SCSI_CONSTANTS=y
CONFIG_SCSI_LOGGING=y
# CONFIG_SCSI_SCAN_ASYNC is not set
CONFIG_SCSI_WAIT_SCAN=m

#
# SCSI Transports
#
```

```
CONFIG_SCSI_SPI_ATTRS=m
CONFIG_SCSI_FC_ATTRS=m
CONFIG_SCSI_ISCSI_ATTRS=m
# CONFIG_SCSI_SAS_LIBSAS is not set
CONFIG_SCSI_LOWLEVEL=y
CONFIG_ISCSI_TCP=m
CONFIG_BLK_DEV_3W_XXXX_RAID=m
CONFIG_SCSI_3W_9XXX=m
CONFIG_SCSI_ACARD=m
CONFIG_SCSI_AACRAID=m
CONFIG_SCSI_AIC7XXX=m
CONFIG_AIC7XXX_CMDS_PER_DEVICE=32
CONFIG_AIC7XXX_RESET_DELAY_MS=15000
# CONFIG_AIC7XXX_DEBUG_ENABLE is not set
CONFIG_AIC7XXX_DEBUG_MASK=0
CONFIG_AIC7XXX_REG_PRETTY_PRINT=y
# CONFIG_SCSI_AIC7XXX_OLD is not set
# CONFIG_SCSI_AIC79XX is not set
# CONFIG_SCSI_AIC94XX is not set
# CONFIG_SCSI_DPT_I2O is not set
# CONFIG_SCSI_ARCMSR is not set
# CONFIG_MEGARAID_NEWGEN is not set
# CONFIG_MEGARAID_LEGACY is not set
# CONFIG_MEGARAID_SAS is not set
# CONFIG_SCSI_HPTIOP is not set
# CONFIG_SCSI_DMX3191D is not set
# CONFIG_SCSI_FUTURE_DOMAIN is not set
# CONFIG_SCSI_IPS is not set
# CONFIG_SCSI_INITIO is not set
# CONFIG_SCSI_INIA100 is not set
# CONFIG_SCSI_STEX is not set
# CONFIG_SCSI_SYM53C8XX_2 is not set
# CONFIG_SCSI_QLOGIC_1280 is not set
# CONFIG_SCSI_QLA_FC is not set
# CONFIG_SCSI_QLA_ISCSI is not set
# CONFIG_SCSI_LPFC is not set
# CONFIG_SCSI_DC395x is not set
# CONFIG_SCSI_DC390T is not set
# CONFIG_SCSI_NSP32 is not set
# CONFIG_SCSI_DEBUG is not set
# CONFIG_SCSI_SRP is not set
# CONFIG_ATA is not set
CONFIG_MD=y
CONFIG_BLK_DEV_MD=m
CONFIG_MD_LINEAR=m
CONFIG_MD_RAID0=m
CONFIG_MD_RAID1=m
CONFIG_MD_RAID10=m
CONFIG_MD_RAID456=m
CONFIG_MD_RAID5_RESHAPE=y
CONFIG_MD_MULTIPATH=m
CONFIG_MD_FAULTY=m
CONFIG_BLK_DEV_DM=m
# CONFIG_DM_DEBUG is not set
CONFIG_DM_CRYPT=m
CONFIG_DM_SNAPSHOT=m
CONFIG_DM_MIRROR=m
```

```
CONFIG_DM_ZERO=m
CONFIG_DM_MULTIPATH=m
CONFIG_DM_MULTIPATH_EMC=m
# CONFIG_DM_MULTIPATH_RDAC is not set
# CONFIG_DM_DELAY is not set

#
# Fusion MPT device support
#
# CONFIG_FUSION is not set
# CONFIG_FUSION_SPI is not set
# CONFIG_FUSION_FC is not set
# CONFIG_FUSION_SAS is not set

#
# IEEE 1394 (FireWire) support
#
# CONFIG_FIREWIRE is not set
# CONFIG_IEEE1394 is not set
# CONFIG_I2O is not set
CONFIG_NETDEVICES=y
# CONFIG_NETDEVICES_MULTIQUEUE is not set
# CONFIG_IFB is not set
CONFIG_DUMMY=m
CONFIG_BONDING=m
# CONFIG_MACVLAN is not set
CONFIG_EQUALIZER=m
CONFIG_TUN=m
# CONFIG_ARCNET is not set
CONFIG_PHYLIB=m

#
# MII PHY device drivers
#
CONFIG_MARVELL_PHY=m
CONFIG_DAVICOM_PHY=m
CONFIG_QSEMI_PHY=m
CONFIG_LXT_PHY=m
CONFIG_CICADA_PHY=m
CONFIG_VITESSE_PHY=m
CONFIG_SMSC_PHY=m
# CONFIG_BROADCOM_PHY is not set
# CONFIG_ICPLUS_PHY is not set
# CONFIG_FIXED_PHY is not set
CONFIG_NET_ETHERNET=y
CONFIG_MII=y
# CONFIG_AX88796 is not set
# CONFIG_HAPPYMEAL is not set
# CONFIG_SUNGEM is not set
# CONFIG_CASSINI is not set
# CONFIG_NET_VENDOR_3COM is not set
# CONFIG_DM9000 is not set
# CONFIG_NET_TULIP is not set
# CONFIG_HP100 is not set
CONFIG_NET_PCI=y
CONFIG_PCNET32=y
CONFIG_PCNET32_NAPI=y
```

```
# CONFIG_AMD8111_ETH is not set
# CONFIG_ADAPTEC_STARFIRE is not set
# CONFIG_B44 is not set
# CONFIG_FORCEDETH is not set
# CONFIG_TC35815 is not set
# CONFIG_DGRS is not set
CONFIG_EEPRO100=y
# CONFIG_E100 is not set
# CONFIG_FEALNX is not set
# CONFIG_NATSEMI is not set
# CONFIG_NE2K_PCI is not set
# CONFIG_8139CP is not set
# CONFIG_8139TOO is not set
# CONFIG_SIS900 is not set
# CONFIG_EPIC100 is not set
# CONFIG_SUNDANCE is not set
# CONFIG_TLAN is not set
# CONFIG_VIA_RHINE is not set
# CONFIG_SC92031 is not set
CONFIG_NETDEV_1000=y
# CONFIG_ACENIC is not set
# CONFIG_DL2K is not set
# CONFIG_E1000 is not set
# CONFIG_NS83820 is not set
# CONFIG_HAMACHI is not set
# CONFIG_YELLOWFIN is not set
# CONFIG_R8169 is not set
# CONFIG_SIS190 is not set
# CONFIG_SKGE is not set
# CONFIG_SKY2 is not set
# CONFIG_SK98LIN is not set
# CONFIG_VIA_VELOCITY is not set
# CONFIG_TIGON3 is not set
# CONFIG_BNX2 is not set
# CONFIG_QLA3XXX is not set
# CONFIG_ATL1 is not set
CONFIG_NETDEV_10000=y
# CONFIG_CHELSIO_T1 is not set
# CONFIG_CHELSIO_T3 is not set
# CONFIG_IXGB is not set
# CONFIG_S2IO is not set
# CONFIG_MYRI10GE is not set
# CONFIG_NETXEN_NIC is not set
# CONFIG_MLX4_CORE is not set
# CONFIG_TR is not set

#
# Wireless LAN
#
# CONFIG_WLAN_PRE80211 is not set
# CONFIG_WLAN_80211 is not set
# CONFIG_WAN is not set
# CONFIG_FDDI is not set
# CONFIG_HIPPI is not set
# CONFIG_PPP is not set
# CONFIG_SLIP is not set
# CONFIG_NET_FC is not set
```

```
# CONFIG_SHAPER is not set
# CONFIG_NETCONSOLE is not set
# CONFIG_NETPOLL is not set
# CONFIG_NET_POLL_CONTROLLER is not set
# CONFIG_ISDN is not set
# CONFIG_PHONE is not set

#
# Input device support
#
CONFIG_INPUT=y
# CONFIG_INPUT_FF_MEMLESS is not set
# CONFIG_INPUT_POLLDEV is not set

#
# Userland interfaces
#
CONFIG_INPUT_MOUSEDEV=y
CONFIG_INPUT_MOUSEDEV_PSAUX=y
CONFIG_INPUT_MOUSEDEV_SCREEN_X=1024
CONFIG_INPUT_MOUSEDEV_SCREEN_Y=768
# CONFIG_INPUT_JOYDEV is not set
# CONFIG_INPUT_TSDEV is not set
# CONFIG_INPUT_EVDEV is not set
# CONFIG_INPUT_EVBUG is not set

#
# Input Device Drivers
#
CONFIG_INPUT_KEYBOARD=y
CONFIG_KEYBOARD_ATKBD=y
# CONFIG_KEYBOARD_SUNKBD is not set
# CONFIG_KEYBOARD_LKKBD is not set
# CONFIG_KEYBOARD_XTKBD is not set
# CONFIG_KEYBOARD_NEWTON is not set
# CONFIG_KEYBOARD_STOWAWAY is not set
CONFIG_INPUT_MOUSE=y
CONFIG_MOUSE_PS2=y
CONFIG_MOUSE_PS2_ALPS=y
CONFIG_MOUSE_PS2_LOGIPS2PP=y
CONFIG_MOUSE_PS2_SYNAPTICS=y
CONFIG_MOUSE_PS2_LIFEBOOK=y
CONFIG_MOUSE_PS2_TRACKPOINT=y
# CONFIG_MOUSE_PS2_TOUCHKIT is not set
# CONFIG_MOUSE_SERIAL is not set
# CONFIG_MOUSE_APPLETOUCH is not set
# CONFIG_MOUSE_VSXXXAA is not set
# CONFIG_INPUT_JOYSTICK is not set
# CONFIG_INPUT_TABLET is not set
# CONFIG_INPUT_TOUCHSCREEN is not set
# CONFIG_INPUT_MISC is not set

#
# Hardware I/O ports
#
CONFIG_SERIO=y
CONFIG_SERIO_I8042=y
```

```
CONFIG_SERIO_SERPORT=y
# CONFIG_SERIO_PCIPS2 is not set
CONFIG_SERIO_LIBPS2=y
CONFIG_SERIO_RAW=y
# CONFIG_GAMEPORT is not set

#
# Character devices
#
CONFIG_VT=y
CONFIG_VT_CONSOLE=y
CONFIG_HW_CONSOLE=y
CONFIG_VT_HW_CONSOLE_BINDING=y
# CONFIG_SERIAL_NONSTANDARD is not set

#
# Serial drivers
#
CONFIG_SERIAL_8250=y
CONFIG_SERIAL_8250_CONSOLE=y
CONFIG_SERIAL_8250_PCI=y
CONFIG_SERIAL_8250_NR_UARTS=4
CONFIG_SERIAL_8250_RUNTIME_UARTS=4
# CONFIG_SERIAL_8250_EXTENDED is not set

#
# Non-8250 serial port support
#
CONFIG_SERIAL_CORE=y
CONFIG_SERIAL_CORE_CONSOLE=y
# CONFIG_SERIAL_JSM is not set
CONFIG_UNIX98_PTYS=y
CONFIG_LEGACY_PTYS=y
CONFIG_LEGACY_PTY_COUNT=256
# CONFIG_IPMI_HANDLER is not set
# CONFIG_WATCHDOG is not set
# CONFIG_HW_RANDOM is not set
CONFIG_RTC=y
# CONFIG_R3964 is not set
# CONFIG_APPLICOM is not set
# CONFIG_DRM is not set
# CONFIG_RAW_DRIVER is not set
# CONFIG_TCG_TPM is not set
CONFIG_DEVPORT=y
# CONFIG_I2C is not set

#
# SPI support
#
# CONFIG_SPI is not set
# CONFIG_SPI_MASTER is not set
# CONFIG_W1 is not set
# CONFIG_POWER_SUPPLY is not set
# CONFIG_HWMON is not set

#
# Multifunction device drivers
```

```
#
# CONFIG_MFD_SM501 is not set

#
# Multimedia devices
#
# CONFIG_VIDEO_DEV is not set
# CONFIG_DVB_CORE is not set
CONFIG_DAB=y

#
# Graphics support
#
# CONFIG_BACKLIGHT_LCD_SUPPORT is not set

#
# Display device support
#
# CONFIG_DISPLAY_SUPPORT is not set
# CONFIG_VGASTATE is not set
CONFIG_VIDEO_OUTPUT_CONTROL=m
CONFIG_FB=y
CONFIG_FIRMWARE_EDID=y
# CONFIG_FB_DDC is not set
CONFIG_FB_CFB_FILLRECT=y
CONFIG_FB_CFB_COPYAREA=y
CONFIG_FB_CFB_IMAGEBLIT=y
# CONFIG_FB_SYS_FILLRECT is not set
# CONFIG_FB_SYS_COPYAREA is not set
# CONFIG_FB_SYS_IMAGEBLIT is not set
# CONFIG_FB_SYS_FOPS is not set
CONFIG_FB_DEFERRED_IO=y
# CONFIG_FB_SVGALIB is not set
# CONFIG_FB_MACMODES is not set
# CONFIG_FB_BACKLIGHT is not set
CONFIG_FB_MODE_HELPERS=y
# CONFIG_FB_TILEBLITTING is not set

#
# Frame buffer hardware drivers
#
CONFIG_FB_CIRRUS=y
# CONFIG_FB_PM2 is not set
# CONFIG_FB_CYBER2000 is not set
# CONFIG_FB_ASILIANT is not set
# CONFIG_FB_IMSTT is not set
# CONFIG_FB_S1D13XXX is not set
# CONFIG_FB_NVIDIA is not set
# CONFIG_FB_RIVA is not set
# CONFIG_FB_MATROX is not set
# CONFIG_FB_RADEON is not set
# CONFIG_FB_ATY128 is not set
# CONFIG_FB_ATY is not set
# CONFIG_FB_S3 is not set
# CONFIG_FB_SAVAGE is not set
# CONFIG_FB_SIS is not set
# CONFIG_FB_NEOMAGIC is not set
```

```
# CONFIG_FB_KYRO is not set
# CONFIG_FB_3DFX is not set
# CONFIG_FB_VOODOO1 is not set
# CONFIG_FB_VT8623 is not set
# CONFIG_FB_TRIDENT is not set
# CONFIG_FB_ARK is not set
# CONFIG_FB_PM3 is not set
# CONFIG_FB_VIRTUAL is not set

#
# Console display driver support
#
# CONFIG_VGA_CONSOLE is not set
CONFIG_DUMMY_CONSOLE=y
CONFIG_FRAMEBUFFER_CONSOLE=y
# CONFIG_FRAMEBUFFER_CONSOLE_DETECT_PRIMARY is not set
# CONFIG_FRAMEBUFFER_CONSOLE_ROTATION is not set
CONFIG_FONTS=y
# CONFIG_FONT_8x8 is not set
CONFIG_FONT_8x16=y
# CONFIG_FONT_6x11 is not set
# CONFIG_FONT_7x14 is not set
# CONFIG_FONT_PEARL_8x8 is not set
# CONFIG_FONT_ACORN_8x8 is not set
# CONFIG_FONT_MINI_4x6 is not set
# CONFIG_FONT_SUN8x16 is not set
# CONFIG_FONT_SUN12x22 is not set
# CONFIG_FONT_10x18 is not set
CONFIG_LOGO=y
CONFIG_LOGO_LINUX_MONO=y
CONFIG_LOGO_LINUX_VGA16=y
CONFIG_LOGO_LINUX_CLUT224=y

#
# Sound
#
# CONFIG_SOUND is not set
CONFIG_HID_SUPPORT=y
CONFIG_HID=y
# CONFIG_HID_DEBUG is not set
CONFIG_USB_SUPPORT=y
CONFIG_USB_ARCH_HAS_HCD=y
CONFIG_USB_ARCH_HAS_OHCI=y
CONFIG_USB_ARCH_HAS_EHCI=y
# CONFIG_USB is not set

#
# NOTE: USB_STORAGE enables SCSI, and 'SCSI disk support'
#

#
# USB Gadget Support
#
# CONFIG_USB_GADGET is not set
# CONFIG_MMC is not set
# CONFIG_NEW_LEDS is not set
# CONFIG_INFINIBAND is not set
```

```
# CONFIG_RTC_CLASS is not set

#
# DMA Engine support
#
# CONFIG_DMA_ENGINE is not set

#
# DMA Clients
#

#
# DMA Devices
#

#
# Userspace I/O
#
# CONFIG_UIO is not set

#
# File systems
#
CONFIG_EXT2_FS=y
# CONFIG_EXT2_FS_XATTR is not set
# CONFIG_EXT2_FS_XIP is not set
CONFIG_EXT3_FS=y
CONFIG_EXT3_FS_XATTR=y
# CONFIG_EXT3_FS_POSIX_ACL is not set
# CONFIG_EXT3_FS_SECURITY is not set
# CONFIG_EXT4DEV_FS is not set
CONFIG_JBD=y
# CONFIG_JBD_DEBUG is not set
CONFIG_FS_MBCACHE=y
CONFIG_REISERFS_FS=m
# CONFIG_REISERFS_CHECK is not set
CONFIG_REISERFS_PROC_INFO=y
CONFIG_REISERFS_FS_XATTR=y
CONFIG_REISERFS_FS_POSIX_ACL=y
CONFIG_REISERFS_FS_SECURITY=y
CONFIG_JFS_FS=m
CONFIG_JFS_POSIX_ACL=y
CONFIG_JFS_SECURITY=y
# CONFIG_JFS_DEBUG is not set
# CONFIG_JFS_STATISTICS is not set
CONFIG_FS_POSIX_ACL=y
CONFIG_XFS_FS=m
CONFIG_XFS_QUOTA=y
CONFIG_XFS_SECURITY=y
CONFIG_XFS_POSIX_ACL=y
# CONFIG_XFS_RT is not set
# CONFIG_GFS2_FS is not set
# CONFIG_OCFS2_FS is not set
CONFIG_MINIX_FS=m
CONFIG_ROMFS_FS=m
CONFIG_INOTIFY=y
CONFIG_INOTIFY_USER=y
```

```
CONFIG_QUOTA=y
# CONFIG_QFMT_V1 is not set
CONFIG_QFMT_V2=y
CONFIG_QUOTACTL=y
CONFIG_DNOTIFY=y
CONFIG_AUTOFS_FS=y
# CONFIG_AUTOFS4_FS is not set
CONFIG_FUSE_FS=m

#
# CD-ROM/DVD Filesystems
#
CONFIG_ISO9660_FS=m
CONFIG_JOLIET=y
CONFIG_ZISOFS=y
CONFIG_UDF_FS=m
CONFIG_UDF_NLS=y

#
# DOS/FAT/NT Filesystems
#
CONFIG_FAT_FS=m
CONFIG_MSDOS_FS=m
CONFIG_VFAT_FS=m
CONFIG_FAT_DEFAULT_CODEPAGE=437
CONFIG_FAT_DEFAULT_IOCHARSET="iso8859-1"
# CONFIG_NTFS_FS is not set

#
# Pseudo filesystems
#
CONFIG_PROC_FS=y
CONFIG_PROC_KCORE=y
CONFIG_PROC_SYSCTL=y
CONFIG_SYSFS=y
CONFIG_TMPFS=y
# CONFIG_TMPFS_POSIX_ACL is not set
# CONFIG_HUGETLB_PAGE is not set
CONFIG_RAMFS=y
# CONFIG_CONFIGFS_FS is not set

#
# Miscellaneous filesystems
#
# CONFIG_ADFS_FS is not set
CONFIG_AFFS_FS=m
CONFIG_HFS_FS=m
CONFIG_HFSPLUS_FS=m
CONFIG_BEFS_FS=m
# CONFIG_BEFS_DEBUG is not set
CONFIG_BFS_FS=m
CONFIG_EFS_FS=m
# CONFIG_JFFS2_FS is not set
CONFIG_CRAMFS=y
CONFIG_VXFS_FS=m
# CONFIG_HPFS_FS is not set
# CONFIG_QNX4FS_FS is not set
```

```
CONFIG_SYSV_FS=m
CONFIG_UFS_FS=m
# CONFIG_UFS_FS_WRITE is not set
# CONFIG_UFS_DEBUG is not set

#
# Network File Systems
#
CONFIG_NFS_FS=y
CONFIG_NFS_V3=y
# CONFIG_NFS_V3_ACL is not set
# CONFIG_NFS_V4 is not set
# CONFIG_NFS_DIRECTIO is not set
CONFIG_NFSD=y
CONFIG_NFSD_V3=y
# CONFIG_NFSD_V3_ACL is not set
# CONFIG_NFSD_V4 is not set
# CONFIG_NFSD_TCP is not set
CONFIG_ROOT_NFS=y
CONFIG_LOCKD=y
CONFIG_LOCKD_V4=y
CONFIG_EXPORTFS=y
CONFIG_NFS_COMMON=y
CONFIG_SUNRPC=y
# CONFIG_SUNRPC_BIND34 is not set
# CONFIG_RPCSEC_GSS_KRB5 is not set
# CONFIG_RPCSEC_GSS_SPKM3 is not set
# CONFIG_SMB_FS is not set
# CONFIG_CIFS is not set
# CONFIG_NCP_FS is not set
# CONFIG_CODA_FS is not set
# CONFIG_AFS_FS is not set

#
# Partition Types
#
# CONFIG_PARTITION_ADVANCED is not set
CONFIG_MSDOS_PARTITION=y

#
# Native Language Support
#
CONFIG_NLS=m
CONFIG_NLS_DEFAULT="iso8859-1"
CONFIG_NLS_CODEPAGE_437=m
CONFIG_NLS_CODEPAGE_737=m
CONFIG_NLS_CODEPAGE_775=m
CONFIG_NLS_CODEPAGE_850=m
CONFIG_NLS_CODEPAGE_852=m
CONFIG_NLS_CODEPAGE_855=m
CONFIG_NLS_CODEPAGE_857=m
CONFIG_NLS_CODEPAGE_860=m
CONFIG_NLS_CODEPAGE_861=m
CONFIG_NLS_CODEPAGE_862=m
CONFIG_NLS_CODEPAGE_863=m
CONFIG_NLS_CODEPAGE_864=m
CONFIG_NLS_CODEPAGE_865=m
```

```
CONFIG_NLS_CODEPAGE_866=m
CONFIG_NLS_CODEPAGE_869=m
CONFIG_NLS_CODEPAGE_936=m
CONFIG_NLS_CODEPAGE_950=m
CONFIG_NLS_CODEPAGE_932=m
CONFIG_NLS_CODEPAGE_949=m
CONFIG_NLS_CODEPAGE_874=m
CONFIG_NLS_ISO8859_8=m
CONFIG_NLS_CODEPAGE_1250=m
CONFIG_NLS_CODEPAGE_1251=m
CONFIG_NLS_ASCII=m
CONFIG_NLS_ISO8859_1=m
CONFIG_NLS_ISO8859_2=m
CONFIG_NLS_ISO8859_3=m
CONFIG_NLS_ISO8859_4=m
CONFIG_NLS_ISO8859_5=m
CONFIG_NLS_ISO8859_6=m
CONFIG_NLS_ISO8859_7=m
CONFIG_NLS_ISO8859_9=m
CONFIG_NLS_ISO8859_13=m
CONFIG_NLS_ISO8859_14=m
CONFIG_NLS_ISO8859_15=m
CONFIG_NLS_KOI8_R=m
CONFIG_NLS_KOI8_U=m
CONFIG_NLS_UTF8=m

#
# Distributed Lock Manager
#
# CONFIG_DLM is not set

#
# Profiling support
#
# CONFIG_PROFILING is not set

#
# Kernel hacking
#
CONFIG_TRACE_IRQFLAGS_SUPPORT=y
# CONFIG_PRINTK_TIME is not set
CONFIG_ENABLE_MUST_CHECK=y
# CONFIG_MAGIC_SYSRQ is not set
# CONFIG_UNUSED_SYMBOLS is not set
# CONFIG_DEBUG_FS is not set
# CONFIG_HEADERS_CHECK is not set
# CONFIG_DEBUG_KERNEL is not set
CONFIG_CROSSCOMPILE=y
CONFIG_CMDLINE=""

#
# Security options
#
# CONFIG_KEYS is not set
# CONFIG_SECURITY is not set
CONFIG_XOR_BLOCKS=m
CONFIG_ASYNC_CORE=m
```

```
CONFIG_ASYNC_MEMCPY=m
CONFIG_ASYNC_XOR=m
CONFIG_CRYPTO=y
CONFIG_CRYPTO_ALGAPI=y
CONFIG_CRYPTO_BLKCIPHER=m
CONFIG_CRYPTO_HASH=y
CONFIG_CRYPTO_MANAGER=y
CONFIG_CRYPTO_HMAC=y
# CONFIG_CRYPTO_XCBC is not set
CONFIG_CRYPTO_NULL=m
CONFIG_CRYPTO_MD4=m
CONFIG_CRYPTO_MD5=m
CONFIG_CRYPTO_SHA1=m
CONFIG_CRYPTO_SHA256=m
CONFIG_CRYPTO_SHA512=m
CONFIG_CRYPTO_WP512=m
CONFIG_CRYPTO_TGR192=m
# CONFIG_CRYPTO_GF128MUL is not set
CONFIG_CRYPTO_ECB=m
CONFIG_CRYPTO_CBC=m
CONFIG_CRYPTO_PCBC=m
# CONFIG_CRYPTO_LRW is not set
# CONFIG_CRYPTO_CRYPTD is not set
CONFIG_CRYPTO_DES=m
# CONFIG_CRYPTO_FCRYPT is not set
CONFIG_CRYPTO_BLOWFISH=m
CONFIG_CRYPTO_TWOFISH=m
CONFIG_CRYPTO_TWOFISH_COMMON=m
CONFIG_CRYPTO_SERPENT=m
CONFIG_CRYPTO_AES=m
CONFIG_CRYPTO_CAST5=m
CONFIG_CRYPTO_CAST6=m
CONFIG_CRYPTO_TEA=m
CONFIG_CRYPTO_ARC4=m
CONFIG_CRYPTO_KHAZAD=m
CONFIG_CRYPTO_ANUBIS=m
CONFIG_CRYPTO_DEFLATE=m
CONFIG_CRYPTO_MICHAEL_MIC=m
CONFIG_CRYPTO_CRC32C=m
# CONFIG_CRYPTO_CAMELLIA is not set
# CONFIG_CRYPTO_TEST is not set
CONFIG_CRYPTO_HW=y

#
# Library routines
#
CONFIG_BITREVERSE=y
# CONFIG_CRC_CCITT is not set
CONFIG_CRC16=m
# CONFIG_CRC_ITU_T is not set
CONFIG_CRC32=y
# CONFIG_CRC7 is not set
CONFIG_LIBCRC32C=m
CONFIG_ZLIB_INFLATE=y
CONFIG_ZLIB_DEFLATE=m
CONFIG_TEXTSEARCH=y
CONFIG_TEXTSEARCH_KMP=m
```

```
CONFIG_TEXTSEARCH_BM=m
CONFIG_TEXTSEARCH_FSM=m
CONFIG_PLIST=y
CONFIG_HAS_IOMEM=y
CONFIG_HAS_IOPORT=y
CONFIG_HAS_DMA=y


##
```