UCam Installation Manual

Version 0.3

Stewart McLay

5th March 2010

Acknowledgements

Professor Vikram Dhillon who was P.I. for the UltraCam instrument the project from which UCam was born.

The UCam development team has involved several members of UK ATC staff. Valuable contributions were made by: David Atkinson, Steven Beard, Derek Ives, Stewart McLay, Chris Tierney and Andy Vick.

The Enhanced Machine Controller (EMC) project for their knowledge base wiki which has been a valuable source of information for installing RTAI on Debian Linux.

Table of Contents

1 Introduction

The UCam software is a camera controller and data acquisition application. It is developed for use on camera systems as part of astronomical research instrumentation and is presently in use at some of the leading telescopes in the world today. UCam was originally developed for the UltraCam instrument which is a high temporal resolution triple-beam CCD camera and has subsequently been used on the WFCAM instrument which is a wide field infra-red survey camera. These are just two examples of the wide diversity of astronomical instrumentation that UCam can support but there are others.

The UCam software is primarily designed to make full use of the ARC controller as developed by Astronomical Research Cameras. Although it is a highly configurable system and may be adapted in the future for use with other camera controllers. The UCam design model is application centric where different applications tailored for specific detectors and readout modes are downloaded and executed on the camera controller hardware. This highly configurable application centric design is what has enabled UCam to be an adaptable and reusable solution for several astronomical instruments.

UCam runs on a real-time Linux operating system so it is able to provide fast imaging capabilities for instrumentation where it is needed. It is also designed to be a server application so it can be installed and used remotely across a network running on a standard Linux operating system. At present UCam supports the HTTP protocol and uses the XML format for sending data packets. The interface is simple enough to be accessible through a web browser and there are several client GUI applications and software libraries available for interfacing with the UCam server.

This document described how to install a UCam software system. This includes how to install the real-time Linux operating system and all the necessary software packages required to install and run the UCam system. The intended audience for this document are technical staff who are responsible for installing and maintaining camera data acquisition systems. The document attempts to give a step by step guide of how to install the software but some prior technical experience of using Linux from the command line is advisable.

1.1 About UltraCam

The UCam software system was originally designed and developed for the UltraCam instrument. UltraCam is an ultra-fast, triple-beam CCD camera which has been designed to study one of the few remaining unexplored regions of observational parameter space - high temporal resolution. The camera, funded by PPARC, saw first light during 2001 and has been used on 2m, 4m and 8m class telescopes in Australia, the Canary Islands, Chile, Greece, South Africa and Spain to study astrophysics on the fastest time scales. More information about UltraCam can be found at http://www.vikdhillon.staff.shef.ac.uk/ultracam.

1.2 About ARC Controller

The UCam software system was originally designed to interface with a ARC controller. The ARC controller is developed by Astronomical Research Cameras, Inc. They design, develop and manufacture controllers for operating CCD and infra-red imaging arrays for astronomical and related applications. They can control a wide variety of imaging arrays in a medley of exposure and readout modes at medium to low speeds with detector limited noise levels. The controllers include electronic circuits, mechanical assemblies, power supplies and supporting software. They are variously known as “Leach controllers”, "ARC controllers", as well as "SDSU controllers" for their

origins at San Diego State University. Additionally, we can supply turn-key customized systems incorporating almost any type of scientific CCD. More information about ARC controllers can be found at http://www.astro-cam.com.

1.3 About UK Astronomy Technology Centre

The UK Astronomy Technology Centre is the national centre for astronomical technology. We design and build instruments for many of the world's major telescopes. We also project-manage UK and international collaborations. Our scientists carry out observational and theoretical research into fundamental questions such as the origins of planets and of galaxies. More information is available at http://www.roe.ac.uk/atc.

1.4 Acronyms And Abbreviations

ARC	Astronomical Research Cameras
CDS	Correlated Double Sampling
Debian	Debian Linux distribution
FITS	Flexible Image Transfer System
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
LAN	Local Area Network
NDR	Non Destructive Readout
OS	Operating System
RTAI	Real Time Application Interface
UltraCam	Ultra-fast, triple-beam CCD camera
URI	Uniform Resource Indicator
URL	Uniform Resource Locator
UK ATC	United Kingdom Astronomy Technology Centre
XML	Extensible Mark-up Language

1.5 Stylistic Conventions

Terminal shell commands will be displayed using `Courier` font with the `$` representing a regular user the shell prompt. Everything after the `$` prompt will be a command entered at the terminal shell by a user.

```
$ pwd
```

A `#` shell prompt is used to represent the root user shell.

```
# lsmod
```

Examples of XML data will be displayed using `Courier` font.

```
<Logging>
```

```
        <Name>UCam</Name>
        <Filename>/home/ucam/logfiles/ucam.log</Filename>
        <Level>INFO</Level>
    </Logging>
```

Python shell input and output will also be displayed using `Courier` font with >>> representing the python shell prompt. Again everything after the >>> prompt will be a command entered at the python shell by a user.

```
>>> import sys
>>> print sys.version
2.5.2 (r252:60911, Oct  5 2008, 19:24:49)
[GCC 4.3.2]
```

Examples of Python source code files will be framed with each line prefixed with a line number. The line numbers are included to aid the reader but are not included in the actual source code files. The python code be displayed using `Courier` font.

```
001 #!/usr/bin/env python
002
003 from ucam import UCam
004
005 ucam = UCam()
006 ucam.setAddress('ucamdev.roe.ac.uk')
007 ucam.setPort(9980, 9981, 9982)
008 ucam.connect()
```

Examples of configuration files will be framed with each line prefixed with a line number. The line numbers are included to aid the reader but are not included in the actual configuration text files. The text file data be displayed using `Courier` font.

```
001 logging.loggers.root.channel.class = ConsoleChannel
002 logging.loggers.app.name = Application
003 logging.loggers.app.channel = c1
004 logging.formatters.f1.class = PatternFormatter
005 logging.formatters.f1.pattern = [%p] %t
```

2 How To Install Debian Linux

The GNU/Linux OS is a very flexible platform that can be used for all sorts of applications. Linux can be used for running a web server, mail server, DNS server, print server, firewall and as a desktop computer amongst other things. UCam is designed to run on real-time Linux and therefore all we really need is the real-time operating system. In fact, in order to achieve the best real-time performance RTAI recommend using just the basic real-time OS without running any other unnecessary software which may impact on the performance by increasing the latency. Therefore it is better to avoid installing the X windows environment required by GUI applications to achieve best performance.

It is a good idea to consider how you plan to run UCam and integrate into your existing set-up for your instrument control system. It is safer and more efficient to run the UCam camera controller software on a separate PC to avoid any performance problems due to conflicts with other resources running on the same PC. This means running UCam on a standard Linux installation as a server with your client applications running on another computer on the network. But if this kind of set-up is too complicated for you needs then you may install a full desktop environment and run UCam as a stand alone system.

This manual provides instructions on how to install the UCam software on a Debian Linux distribution. The Debian distribution was chosen because of its stability, large number of packages, flexible installation set-up and support for multiple platforms. The installation process make heavy use of the Debian package management tools. You are strongly advised follow the advice provided in this document unless you are a Linux expert and have a strong preference for an alternative Linux distribution. Even so, you may find it a lot more difficult than you think to apply these instructions across another platform and therefore you should read the whole document first before you undertake such a task to prevent any major disappointment at latter stage.

2.1 Debian Releases

It is recommended that you install a standard Debian Linux OS as your starting platform. The Debian Linux distribution provides a very convenient installer for such purposes.

There are three types of Debian releases available. These are:

- Stable: This is release recommended because it is the most stable and mature.
- Testing: This release will form the next stable release after all major bugs have been fixed and is probably acceptable for use in desktop computers for users who want a more up to date system
- Unstable (sid): This release is used for introducing the latest software packages that may have critical bugs that need fixing before they can make it into the testing branch. The fact that it is called 'Unstable' should make it clear that this release is best avoided if you want a system that is stable enough for everyday use.

The current Debian stable release at the time of writing this document is V5.0 which has the code name “Lenny”. A newer release may be available which may prove easier to install depending on the hardware configuration of the PC you are trying to install.

2.2 Installing Debian Linux Using The Network Installer

The Debian network installer allows you to install Debian using a connection to the internet. You initially download a small portion of Debian required to start the installation process. After that you

can install whatever else you want from within the installation program. There are three options available for network installations which are:

- Small CDs
- Tiny CDs, floppy disks, floppy disks, USB sticks, etc.
- Network boot

You may choose the option that suits you the best.

The Debian installer will prompt you to enter certain information during the installation process so it can install and configure the Linux OS to suit your needs. The installation process includes the following steps:

1. Choosing your preferred language
2. Selecting a keyboard layout
3. Configuring the network connection
4. Partitioning the disks
5. Set-up Users and passwords
6. Installation of the base system
7. Configure the package manager
8. Select and install software
9. Installing the GRUB boot loader

In step 4 you have the option for configuring the partition table of your hard disks. Debian provides a number of options where you can have everything on a single partition, split system resources across several partitions or manually configure the partitions yourself. As well as that there are options for using LVM and encryption. The simplest option is to install the entire system in a single partition which is recommended for users who are unsure about which solution suits them best. We would generally recommend that you install the system resources over several partitions as this is probably the safest option for a secure and stable system where data is recoverable in the event of file system problems. The Debian option for automatically spliting the system over several partitions is very tight with size allocations which can lead to problems when installing other software such as IRAF. Therefore we would recommend that you configure your partition table manually using Table 1 as a guide.

Table 1: Recommended hard disk partition table

Partition Type	**Mount Point**	**File System**	**Size**	**Description**
Primary*	/	ext3	3 GB	Root file system and system files
Primary		swap	2 GB	Swap partition
Logical	/usr	ext3	10 GB	Static data and applications
Logical	/var	ext3	5 GB	Variable data and system files
Logical	/tmp	ext3	1 GB	Temporary files
Logical	/home	ext3	Remainder	User accounts and data storage

* set the boot flag for this partition

In step 8 of the installation process you have the option of choosing the type of software installation

you would like which determines the main function of your PC. This includes several options such as desktop environment, web server, mail server, print server, etc. We recommended that you disable all options except for the standard system if you are planning on installing UCam as a server based system only. This is all you need and it is better to run a minimal system to achieve a better hard real-time performance. But if you are planning on running UCam as a stand alone system with a full desktop environment then enable the desktop option as well.

After the installer has finished installing Debian Linux it will ask you to remove any installation media such a CDROM and then reboot the PC. Once it has booted the newly installed OS you should log in as using the user account you defined as part of the installation process. Execute the commands below from a terminal. These commands ensure all software updates have been applied to your OS.

```
$ su -
[Log-in as the root user by entering the root password]
# aptitude update
# aptitude safe-upgrade
```

The following sections described how to turn your Debian operating system into a real-time camera controller and data acquisition system. They start by explaining how to install a RTAI real-time Linux operating system and then how to build and install the UCam software.

3 How To Install RTAI Real-Time Linux

The following sections provides two different installation strategies for installing a RTAI real-time Linux operating system. The first is a quick installation which is the easiest and quickest way to perform the installation using preprepared Debian packages. The second provides a full description of the build and installation process which is geared towards users who want to create their own customised versions of the Debian packages or just want to learn more about the process.

3.1 How To Do A Quick Real-Time Linux Installation

This section describes the quickest and easiest way to install the UCam software system. It requires as few steps as possible using pre-packaged binaries for the hard real-time Linux kernel, RTAI modules and UCam.

```
$ su -
[Log-in as the root user by entering the root password]
# cd <your_download_folder>
# aptitude install build-essential gcc-4.1 g++-4.1
# dpkg -i linux-headers-2.6.20-ucam_r1_i386.deb
# ln -s /usr/src/linux-headers-2.6.20-ucam/ /usr/src/linux-2.6.20-ucam
# ln -s /usr/src/linux-headers-2.6.20-ucam/ /usr/src/linux
# dpkg -i linux-image-2.6.20-ucam_r1_i386.deb
# dpkg -i rtai-modules-2.6.20-ucam_3.5+r1_i386.deb
# dpkg -i rtai-dev-2.6.20-ucam_3.5+r1_i386.deb
# reboot
[Select the UCam kernel from the boot menu]
```

You should now have a modified Linux kernel with RTAI real-time extensions installed. Now skip to the section ??? which describes how to install the UCam camera controller and data acquisition software.

3.2 How To Do A Full Real-Time Linux Installation

This section describes how to do a full installation of the UCam software. It gives step by step instructions on how to patch, configure, build the source files and how to create Debian packages. This section is particularly useful for anyone wishing to create their own customised packages. It involves the following steps:

1. Install essential Debian software packages
2. Get the sources files for the Linux kernel, RTAI and the big physical memory patch
3. Patch and build the Linux kernel source files and then bundle into a Debian kernel package
4. Build the RTAI modules and bundle into a Debian kernel package extension

3.2.1 Installing Essential Debian packages

Log-in as the root user and apply any updates for system packages already installed.

```
$ su -
```

```
[Log-in as the root user by entering the root password]
# aptitude update
# aptitude safe-upgrade
```

Install the software development tools for building software and creating Debian packages.

```
# aptitude install kernel-package
# aptitude install dh-make
# aptitude install module-assistant
# aptitude install libncurses5-dev
# aptitude install autoconf
```

The following packages should be automatically be installed as dependencies.

Package	Dependencies
kernel-package	binutils linux-libc-dev libc6-dev libgmp3c2 libmpfr1ldbl cpp-4.3 cpp libgomp1 gcc-4.3 gcc make bzip2 libtimedate-perl dpkg-dev gettext intltool-debian po-debconf libcompress-raw-zlib-perl libio-compress-base-perl libio-compress-zlib-perl libcompress-zlib-perl libdigest-sha1-perl libdigest-hmac-perl libfile-remove-perl libio-stringy-perl libmime-types-perl libmailtools-perl libobject-realize-later-perl liburi-perl libuser-identity-perl libmail-box-perl libsys-hostname-long-perl libmail-sendmail-perl libstdc++6-4.3-dev g++-4.3 g++
dh-make	debhelper html2text
autoconf	automake autotools-dev
libpoco5-dev	libltdl3 libpocodata5 libpocodata5-dbg libpocofoundation5 libpocofoundation5-dbg libpoconet5 libpoconet5-dbg libpoconetssl5 libpoconetssl5-dbg libpocoodbc5 libpocoodbc5-dbg libpocosqlite5 libpocosqlite5-dbg libpocoutil5 libpocoutil5-dbg libpocoxml5 libpocoxml5-dbg odbcinst1debian1 unixodbc

3.2.2 Get The Source Files

Before we can build any software we must download the source files for a vanilla Linux kernel, the RTAI extensions and the big physical memory patch. These can be obtained using the following commands.

```
# cd /root
# wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.20.tar.bz2
```

```
# wget --no-check-certificate https://www.rtai.org/RTAI/rtai-3.5-cv.tar.bz2
# wget http://www.feise.com/~jfeise/Downloads/zr36120/bigphysarea-2.6.20.diff
```

3.2.3 Apply Patches To The Kernel Source Files

Extract the source files for the Linux kernel and create a soft link

```
# tar xjvf linux-2.6.20.tar.bz2 -C /usr/src
# mv /usr/src/linux-2.6.20 /usr/src/linux-2.6.20-ucam
# ln -s /usr/src/linux-2.6.20-ucam /usr/src/linux
```

Extract the source files for RTAI and create a soft link

```
# tar xjvf rtai-3.5-cv.tar.bz2 -C /usr/src
# chown -R root:src /usr/src/rtai-3.5-cv
# mkdir -p /usr/src/modules
# ln -s /usr/src/rtai-3.5-cv /usr/src/modules/rtai
```

Apply patches to the Linux kernel source files

```
# cd /usr/src/linux
# patch -p1 < /usr/src/modules/rtai/base/arch/i386/patches/hal-linux-2.6.20-i386-1.8-04.patch
# patch -p1 < /root/bigphysarea-2.6.20.diff
```

3.2.4 Fix Linux Kernel and GCC Compiler Compatibility Problem

On Debian Lenny the latest version of the GCC compiler (version 4.3) fails to compile older versions of the Linux kernel due to stricter reporting of bugs. There are two ways of fixing the problem. The best solution depends on the version number of the Linux kernel you are using.

3.2.4.1 Configuring Older Versions Of GCC Compilers (Linux Kernel V2.6.21 or less)

An older version of the GCC compiler can be installed and used for compiling the real-time Linux kernel. The list of compilers available from the Debian repositories can be found be using the following command.

```
# aptitude search gcc
```

Install the GCC C/C++ compilers using the following commands.

```
# aptitude install gcc-4.1 g++-4.1
# aptitude install gcc-4.2 g++-4.2
```

The *update-alternatives* utility can be used for switching between compiler versions. First, we must add new menu entries for the installed versions of the GCC compilers.

```
# update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.1 41 --slave /usr/bin/g++ g++ /usr/bin/g++-4.1
# update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.2 42 --slave /usr/bin/g++ g++ /usr/bin/g++-4.2
# update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.3 43 --slave /usr/bin/g++ g++ /usr/bin/g++-4.3
```

Then can then select the GCC compiler version we want using the following command and selecting the appropriate menu item. The GCC version 4.1 compiler is the safest option for compiling the kernel source code if you want to avoid compilation errors.

```
$ update-alternatives --config gcc
```

3.2.4.2 Modifying Linux Kernel Build Flags (Linux Kernel V2.6.22 or greater)

We must apply a small fix by editing the Linux kernel make file */usr/src/linux/Makefile* and adding the following flag to *CFLAGS_KERNEL* build variable.

```
CFLAGS_KERNEL = -fno-tree-scev-cprop
```

3.2.5 Configuring The Linux Kernel

Create a new kernel build configuration file by reusing the system configuration file as a baseline.

```
# cp /boot/config-`uname -r` .config
# make oldconfig
```

Start up the terminal based kernel build configuration editor.

```
# make menuconfig
```

Select the following kernel build configuration options:

- Loadable module support => Enable loadable module support => enabled
- Loadable module support => Module versioning support => disabled
- Processor type and features => HPET ??? => disabled
- Processor type and features => Preemption Model => (No Forced Preemption (Server))
- Processor type and features => Preemption The Big Kernel Lock => disabled
- Processor type and features => Interrupt pipeline => enabled
- Processor type and features => High Memory Support => off
- Processor type and features => Support for physical area reservation => enabled
- Processor type and features => Enable kernel IRQ balancing => enabled
- Power management options (ACPI, APM) => Legacy Power Management API => disabled
- Power management options (ACPI, APM) => Software Suspend => disabled
- Power management options (ACPI, APM) => ACPI (Advanced Configuration and Power Interface) Support => ACPI Support => disabled
- Power management options (ACPI, APM) => APM (Advanced Power Management) BIOS Support => APM BIOS Support => disabled
- Power management options (ACPI, APM) => CPU Frequency scaling => CPU Frequency scaling => disabled
- Power management options (ACPI, APM) => Power Management support => disabled
- Bus options => Message Signalled Interrupts (MSI and MSI-X) => disabled
- Device Drivers => Input device support =>Miscellaneous devices => PC Speaker support => disabled

If your machine is a SMP system (it has multiple processors or multi-core CPU), then enable

symmetric multi-processing support otherwise:

- Processor type and features => Symmetric multi-processing support => enable

Choose the most suitable processor family for your machine. For example, for a *Pentium-III* processor:

- Processor type and features => Processor family => Pentium-III / Celeron(Coppermine) / Pentium-III Xeon

If you have a dual cores CPU or SMP system, don't choose a processor family which has no TSC (time stamp counter). This means that for example you can not choose *586/K5/5x86/6x86/6x86MX* as *Processor family* if you have a dual cores CPU. In conclusion, choose the most suitable processor family for your machine.

3.2.6 Compiling The Linux Kernel (and creating Debian kernel packages)

After we have configured the kernel we need to build it. The best way to do this on any Debian based distribution is to use the Debian package command for making kernel packages. We will create Debian kernel packages for the kernel image, kernel header files and the kernel source files.

```
# make-kpkg clean
# time make-kpkg --append-to-version=-ucam --revision=r1 \
--initrd kernel_image kernel_headers kernel_source
```

3.2.7 Installing The Linux Real-Time Kernel

Now we have created the Debian kernel packages we can install them. We only need to install the kernel image and kernel headers packages for running UCam. The kernel sources package is created as a maintenance precaution. The *dpkg* command is used to install the packages. It will also automatically add a new entry in the grub boot loader's *menu.lst* file for the new kernel.

```
# cd /usr/src
# dpkg -i linux-image-2.6.20-ucam_r1_i386.deb
# dpkg -i linux-headers-2.6.20-ucam_r1_i386.deb
```

Select you new kernel image from the boot loader menu after rebooting the system.

```
# reboot
```

You will have to change the default kernel image setting in *menu.lst* file if you want the your new kernel to become the default kernel used for every reboot of the system. Use a text editor to open and edit the file */boot/grub/menu.lst* and set the *default* value to the menu item number of your new kernel image.

To search for your kernel image in the list of installed Debian packages use the following command.

```
# dpkg -l linux-image-2.6.20-ucam
# dpkg -l linux-headers-2.6.20-ucam
```

To remove your kernel packages use the following command. Using a *-P* or *--purge* option instead of *-r* or *--remove* will also delete the configuration files used by the package.

```
# dpkg -r linux-image-2.6.20-ucam
# dpkg -r linux-headers-2.6.20-ucam
```

3.2.8 Compiling And Installing RTAI Modules

First we must configure RTAI before build it.

```
# cd /usr/src/modules/rtai
```

We need to configure the RTAI build options. Start up the terminal based menu configuration editor.

```
# make menuconfig
```

Set the following kernel build configuration options:

- General => Enable source compatibility mode => enabled
- Machine => Number of CPUs (SMP-only) => <*the number of CPU (cores) in your system*>

You can find out how many CPU (cores) you have or rather how many CPU the Linux kernel thinks you have by using the following command.

```
$ cat /proc/cpuinfo
```

Now prepare the files needed to create turn RTAI into a Debian kernel modules extension package.

```
# mkdir debian
# cd debian
# touch changelog
# touch compat
# touch control.modules.in
# touch copyright
# touch rules
# chmod 755 rules
```

Edit using your favourite editor and insert the contents from appendix A into the following Debian package management files.

- /usr/src/modules/rtai/debian/changelog
- /usr/src/modules/rtai/debian/compat
- /usr/src/modules/rtai/debian/control.modules.in
- /usr/src/modules/rtai/debian/copyright
- /usr/src/modules/rtai/debian/rules

Compile the Debian kernel packages for the RTAI modules with the following commands.

```
# cd /usr/src/linux
# make-kpkg --append-to-version=-ucam --added-modules rtai modules_clean
# make-kpkg --append-to-version=-ucam --added-modules rtai modules_image
```

3.2.9 Installing The RTAI Real-Time Kernel Modules

Now we have created the Debian kernel packages for the RTAI modules we can install them. The *dpkg* command is used to install the packages.

```
# cd /usr/src
# dpkg -i rtai-modules-2.6.20-ucam_3.5+r1_i386.deb
# dpkg -i rtai-dev-2.6.20-ucam_3.5+r1_i386.deb
```

Select you new kernel image from the boot loader menu after rebooting the system.

```
# reboot
```

3.3 Testing RTAI Real-Time Linux

Before we can run the RTAI test suite scripts we must create the pseudo devices for the RTAI FIFO and shared memory modules. Create a bash script called *rtai_setup.sh* using your favourite text editor and enter the following bash code.

```
#!/bin/sh -e
if [ ! -e /dev/rtai_shm ] ; then
      mknod -m 666 /dev/rtai_shm c 10 254
fi
for n in `seq 0 9`; do
      f="/dev/rtf$n"
      if [ ! -c $f ] ; then
            mknod -m 666 $f c 150 $n
      fi
done
```

Now run the set-up script as root from the command line.

```
# ./rtai_setup.sh
```

You can test the RTAI modules using the RTAI test suite of applications. This is very useful for testing the performance and stability of your real-time operating system. Pay particular attention to your latency test results as they provide feedback of how good the real-time performance is. You should hopefully find that your kernel latency test records a maximum latency of around 20 micro-seconds. Most of the tests provide results in nano-seconds units.

```
# cd /usr/realtime/testsuite/user/latency; time ./run
# cd /usr/realtime/testsuite/user/preempt; time ./run
# cd /usr/realtime/testsuite/user/switches; ./run
# cd /usr/realtime/testsuite/kern/latency; time ./run
# cd /usr/realtime/testsuite/kern/preempt; time ./run
# cd /usr/realtime/testsuite/kern/switches; ./run
```

4 How To Install UCam

Installing UCam is a two part process. First we must build the source files to create the executables and then we must install the real-time driver and configure the operating system so UCam is ready to run. But before we do that we must install the prerequisite third-party libraries.

4.1 Installing Third-Party Libraries

Install the Debian packages for the third-party libraries that UCam requires as dependencies.

```
# aptitude install libcfitsio3 libcfitsio3-dev
# aptitude install libxml2 libxml2-dev
# aptitude install libpoco5-dev
```

4.2 Building UCam Software

UCam software uses the GNU autotools suite (autoconf, automake, etc) for configuring, building and installing the UCam deliverables. Using autotools helps make UCam portable over many Unix-like systems. It also allows us to configure many options for customising our build. You may wish to view the complete list of options available by entering the *--help* command line argument to the *configure* script when you build the software.

There are two ways of obtaining the UCam source files. You can check them out directly from the UK ATC source code repository or you can use a tarball distribution file. The following two sub sections describe both alternatives. Please note that if you want to checkout the source files from the repository then you must have a user account.

4.2.1 Building UCam From The Repository

Check out the sources files for UCam from the UK ATC subversion repository. The source files will be written to a directory called *ucamdev*. We create another directory called *ucam* for installing the products of the build process (libraries, include files, executables, etc.). Run the UCam build *configure* script and then run *make* to build and install the executables.

```
$ su ucam
[Log-in as the UCam user account by entering the ucam password]
$ svn checkout --username <username> https://forge.roe.ac.uk/svn/Ultracam/UCam/trunk /home/ucam/ucamdev
$ cd /home/ucam/ucamdev
$ ./configure --prefix=/home/ucam/ucam
$ make install
```

4.2.2 Building UCam From A Tarball

Unzip the tarball file in the *ucam* home directory. We create another directory called *ucam* for installing the products of the build process (libraries, include files, executables, etc.). Run the UCam build *configure* script and then run *make* to build and install the executables.

```
$ tar xjf ucam-0.9.7.tar.bz2 -C /home/ucam
$ cd /home/ucam/ucam-0.9.7
```

```
$ ./configure --prefix=/home/ucam/ucam
$ make
$ make install
```

4.3 Installing UCam

This section provides two sub sections that explain how to install and configure the UCam environment. The first uses the automated installation script which is the quickest and easiest solution. The second provides a detailed step by step description which is useful should anything go wrong or you are interested in learning about the installation process in more detail.

4.3.1 Setting Up The RTAI Environment Using The Installation Script

One of the products of the UCam build is an installation shell script that automatically installs the UCam RTAI driver module and configures the RTAI environment. You must run this script as root.

```
$ su -
[Log-in as the root user by entering the root password]
# /home/ucam/ucam/bin/install_rtai_ucam
# reboot
```

At this point you will hopefully have real-time Linux operating system with all the necessary RTAI modules loaded into the Linux kernel. You test for this by executing the following command.

```
$ lsmod | grep rtai
```

You should see a list of RTAI modules including *rtai_hal, rtai_sched, rtai_fifos, rtai_sem, rtai_shm* and *rtai_sdsu*. If any of these modules are missing then please check the kernel boot log messages to find an explanation.

4.3.2 Setting Up The RTAI Environment Manually

If you had a problem in the previous section when running the automated install script or you are interested in learning more about the details of installation process then this section will help you. First, we must copy the relevant files to their installation directories.

```
$ su -
[Log-in as the root user by entering the root password]
# cp /home/ucam/bin/rtai_sdsu.ko /usr/realtime/modules
# cp /home/ucam/bin/rtai-ucam /etc/init.d
# cp /home/ucam/bin/rtai_sdsu.ko /usr/realtime/modules
```

Install the RTAI initialisation script so that the RTAI environment is automatically started whenever the system boots up.

```
# update-rc.d rtai-ucam start 30 2 3 4 5 .
```

Edit the GRUB boot menu file */boot/grub/menu.lst* using you favourite text editor and append the following kernel boot parameter option *bigphysarea=32770* onto the default kernel options. Please not the following lines are not commands to be entered at the terminal prompt but are taken from the text file.

```
## ## Start Default Options ##
## default kernel options
## default kernel options for automagic boot options
## If you want special options for specific kernels use kopt_x_y_z
## where x.y.z is kernel version. Minor versions can be omitted.
## e.g. kopt=root=/dev/hda1 ro
##      kopt_2_6_8=root=/dev/hdc1 ro
##      kopt_2_6_8_2_686=root=/dev/hdc2 ro
# kopt=root=UUID=/dev/hda1 ro bigphysarea=32770
```

This boot option sets the size of the contiguous memory space reserved for the writing image data frames. The units are kernel pages whose size is configured during the Linux kernel build. The default kernel page size is usually 4K therefore 32770 x 4K gives just over 128MB. Of course, you should make sure that you computer has considerably more memory than this otherwise the kernel will fail to allocate the memory at boot time. Now refresh the boot menu using the following grub command and reboot.

```
# update-grub
# reboot
```

4.4 Starting Up UCam

There are two bash shell scripts available for starting up the UCam servers. The *start_ucam* script starts up the servers locally using a separate X terminal for each server. The *start_ucam_remote* script starts up the servers remotely from a another Unix-like system across the network using the SSH protocol. Both scripts take additional command line options which can view by passing the *-h* option as shown below. The *start_ucam* script has fewer options as it does not need to provide the remote login options.

```
$ start_ucam_remote -h
Usage: ./start_ucam_remote [-h] [-c configext]
Options:
  -c <configext>  Configuration file extension where the file name
                  uses the format camera_<configext>.properites
  -h              Display specific types of command line options
  -u <username>   SSH login user account (default is ucam)
  -s <server>     SSH server domain name (default is ucamdev)
```

When the UCam servers are up and running you can test their HTTP interfaces by starting up a web browser and entering the following URL into the address bar. These URL assume that the server domain name is *ucamdev* and the camera, filesave and demux servers are configured to use the default port settings 9980, 9981 and 9982 respectively. Substitute the domain name or IP address for your server in place of *ucamdev* and change the port settings accordingly if you are not using the default configuration. You should see a simple web page displaying the time the server was started, the current time and the server's uptime.

http://ucamdev:9980/uptime

http://ucamdev:9981/uptime

http://ucamdev:9982/uptime

5 How To Install PyUCam and WxUCam

WxUCam is a GUI application for UCam written in the python programming language. It uses the wxPython GUI toolkit which is a python extension module that wraps the popular wxWidgets cross platform GUI library. As both python and wxPython are cross platform the WxUCam application can be installed and run on most popular operating systems including Microsoft Windows, Apple OSX and most flavours of Unix (including Linux). So far WxUCam has been successfully tested on Linux and Windows.

PyUCam is a python package that provides an common interface (API) for UCam. It is used by the WxUCam application and comes with some example python scripts that can be used and adapted by users for their own purposes. Before you can run PyUCam or WxUCam you must have Python and wxPython installed on your computer. Python is installed by default on nearly all Linux and OSX operating systems. The wxPython tool kit can be installed on Debian by the following command.

```
# aptitude install python-wxgtk2.8
```

Both PyUCam and WxUCam are distributed in a single tar ball which is a compressed archive file. They can be installed in any directory but the following example uses the user's home directory.

```
$ cd $HOME
$ tar xzvf wxucam.tar.gz
```

Once extracted you need to make some minor changes to your environment settings. Update the *PYTHONPATH* environment variable to include the base source directory for each of the packages. Then create an alias for *wxucam* so the GUI application can be started from a terminal. The example commands shown below can be entered from the command prompt at a terminal or added permanently to your *$HOME/.bash_aliases* start-up file so they are always set whenever you start a new login.

```
$ export PYTHONPATH=$PYTHONPATH:$HOME/PyUCam/src:$HOME/WxUCam/src
$ alias wxucam="python $HOME/WxUCam/src/wxucam/WxUCam.py"
```

6 How To Install Meinberg GPS

UCam has support for Meinberg GPS hardware. It uses the GPS system as a source for highly accurate timing information during image data readout. This information is added as time stamp data in the raw image data files and subsequently as FITS header information in the sampled files. If you followed the quick installation instructions for the real-time Linux kernel using the preprepared Debian kernel packages you must follow the instructions in section 6.2. But if you completed the full real-time Linux kernel installation you can skip ahead to section 6.3.

6.1 Hardware Set-up

So far we have integrated support in the UCam software for the Meinberg GPS 170 PEX PCI express board. This software has been tested against version 11 model of this hardware. This can be checked by reading the top right corner of the board where it should have the label GPS170PEX_V11. The board contains a DIL-switch which are normally all switched to the off position by default.

- Enable the time stamp capture event input signal from the D-SUB connector pin 6 by moving DIL switch 2 to the ON position.
- Enable the PPS output signal from the D-SUB connector pin 8 by moving DIL switch 4 to the ON position.

6.2 Preparing the Kernel Sources

Since the driver module is linked into the kernel, it is important that the module is compiled using configuration and version information that matches the running kernel. If you followed the quick install instructions for real-time Linux kernel installation then you will need to perform the following instructions.

```
$ su -
[Log-in as the root user by entering the root password]
# cp /boot/config-`uname -r` /usr/src/linux/.config
# cd /usr/src/linux
```

Edit the Linux kernel make file */usr/src/linux/Makefile* and add the *"-ucam"* version name to the *EXTRAVERSION* variablke near the top of the make file.

```
EXTRAVERSION =-ucam
```

Now configure the kernel sources with the updated version information using the following commands.

```
# make oldconfig
# make prepare
```

6.3 Building And Installing The Meinberg GPS Driver

Extract the source files from the Meinberg Linux tarball and then build and install.

```
# tar xzvf /path/to/mbgtools-lx-1.1.6.tar.gz -C /root
# cd /root/mbgtools-lx-1.1.6
# make check
```

```
# make
# make install
```

Create a udev rule file for creating device links and modifying permissions so the UCam applications can read the captured time stamps. Create a new file with the following file name and path */etc/udev/rules.d/10-meinberg.rules* and edit it with your favourite editor to include the following rules.

```
KERNEL=="mbgntp" SYMLINK+="refclock-0" MODE="0666"
KERNEL=="mbgclk" MODE="0666"
```

Now every time you reboot your system the rules will be applied on boot up.

6.4 Rebuild UCam With Meinberg GPS Support

Go to source directory where you installed the UCam source code and reconfigure the build before rebuilding the UCam software. Please refer back to section 4 if you are unsure.

```
$ cd /home/ucam/ucam-0.9.7
$ ./configure –prefix=/home/ucam/ucam --with-mbggps
$ make
$ make install
```

7 Trouble Shooting And Other Tips

7.1 Checking The Memory Usage

Big physical memory is the Kernel extension for allocating a large contiguous memory block which is used for writing image data buffers. You check the big physical memory area usage by issuing the following command. This will provide information about the total size of big physical memory allocated, the amount of memory currently being used and the amount of memory that is still free.

```
$ cat /proc/bigphysarea
```

Normal system memory usage can be check by using the following commands.

```
$ cat /proc/meminfo
$ free -m
```

7.2 Checking The RTAI Environment

7.2.1 Checking RTAI Modules

List the RTAI modules currently loaded in the Linux kernel using the following command.

```
$ lsmod | grep rtai
```

You should see a list of RTAI modules including *rtai_hal, rtai_sched, rtai_fifos, rtai_sem, rtai_shm* and *rtai_sdsu*. If any of these modules are missing then please check the kernel boot log messages to find an explanation. You can find the kernel boot log at */var/log/messages*.

You can find out more about the status of the RTAI modules by looking their profile from the process information pseudo file system. All RTAI system information can be found under */proc/rtai*. Display the available information using the following commands.

```
$ cat /proc/rtai/hal
$ cat /proc/rtai/fifos
$ cat /proc/rtai/names
$ cat /proc/rtai/scheduler
```

7.2.2 Checking RTAI Device Nodes

List all of the RTAI device nodes by issuing the following command.

```
$ ls -l /dev/rt*
```

You should see output very similar to that shown here.

```
crw-rw-rw- 1 root root 252,   0 2009-01-20 10:59 /dev/rtai_sdsu
crw-rw-rw- 1 root root  10, 254 2009-01-20 10:59 /dev/rtai_shm
crw-rw-rw- 1 root root 150,   0 2009-01-20 10:59 /dev/rtf0
crw-rw-rw- 1 root root 150,   1 2009-01-20 10:59 /dev/rtf1
...
crw-rw-rw- 1 root root 150,  31 2009-01-20 10:59 /dev/rtf31
```

The */dev/rtf[0-31]* devices are the RTAI FIFO pipes used for passing messages between processes. The */dev/rtai_shm* is the RTAI shared memory module device and */dev/rtai_sdsu* is the UCam real-

time device driver device.

7.2.3 Checking RTAI Boot Script Installation

First of all, check that the *rtai-ucam* initialisation script is installed in the */etc/init.d* directory. Use the *sysv-rc-conf* text based editor for checking the services installed on your system and their run levels. Run the utility and scroll down the list of services looking for the *rtai-ucam* script and check that it is enabled for run levels 2, 3, 4 and 5. You can install and run this utility by issuing the following commands.

```
# aptitude install sysv-rc-conf
# sysv-rc-conf
```

7.2.4 Restarting the RTAI Environment

If any RTAI modules or devices are missing then you can always try restarting the RTAI environment by issuing the following command. This command should remove any RTAI modules and devices nodes that currently exist and reload the modules and create new devices. However, if any of these modules or devices are currently being used by some process on your system (most probably a UCam server) you must kill that process first before they can be successfully removed. Note that this command can only be run by the root user.

```
# /etc/init.d/rtai-ucam restart
```

7.3 Checking The UCam Servers Are Alive

When the UCam servers are up and running you can test their HTTP interfaces by starting up a web browser and entering the following URL into the address bar. These URL assume that the server domain name is *ucamdev* and the camera, filesave and demux servers are configured to use the default port settings 9980, 9981 and 9982 respectively. Substitute the domain name or IP address for your server in place of *ucamdev* and change the port settings accordingly if you are not using the default configuration. You should see a simple web page displaying the time the server was started, the current time and the server's uptime.

http://ucamdev:9980/uptime

http://ucamdev:9981/uptime

http://ucamdev:9982/uptime

7.4 Monitoring Processes

Use can use command line tools such as *top* or *htop* for a dynamic real-time view of the tasks running on your system. Both of these utilities show information about memory usage, CPU usage and lots of other details for currently running tasks. Lookup the man pages to find out more about these utilities. You can install and use *htop* using the following commands.

```
# aptitude install htop
$ htop
```

Another command line tool called iotop watches I/O usage information output by the Linux kernel (requires 2.6.20 or later) and displays a table of current I/O usage by processes or threads on the system.

```
# aptitude install iotop
```

```
$ iotop
```

7.5 Debian Administration

7.5.1 Debian Software Updates

It is important to perform regular checks to keep your Debian operating system software up to date. Applying the latest software updates will improve stability and keep your system secure. This is a two step process that requires a command for each step. First you must retrieve the latest list of available packages and then you must upgrade the installed packages to their most recent version. Make sure you are logged in as the administrator (root) user and perform the following commands at the command prompt from a terminal.

```
# aptitude update
# aptitude safe-upgrade
```

7.5.2 Debian Package Management

A Debian package file can be installed by using the following command.

```
# dpkg -i linux-image-2.6.20-ucam_r1_i386.deb
```

You can search for a Debian package file name from the installed packages using the following command. This example shows how to search for all packages containing 'linux-image' in their file name.

```
$ dpkg -S linux-image
```

Or you can list the packages matching a given pattern.

```
$ dpkg -l linux-image-2.6.20-ucam
```

To remove your kernel packages use the following command. Using a *-P* or *--purge* option instead of *-r* or *--remove* will also delete the configuration files used by the package.

```
# dpkg -r linux-image-2.6.20-ucam
```

7.5.3 Debian Language Packs

To install another language pack or to change the default language pack you must reconfigure the language package and change the locale settings. This can be done by logging in as the administrator (root) user and entering the following command at the command prompt from a terminal.

```
# dpkg-reconfigure locales
```

This will launch a text based terminal application with a long list of the language packs available on your system. Find the appropriate languages and select them by pressing the space bar and then select OK to finish. For example, if you want to install the American English language pack you would select the *en_US.UTF8* language pack. You will be prompted to select the default language for your system from the list of selected packages before the application terminates.

7.5.4 Managing User Accounts

If you are have the Gnome desktop installed on your workstation then you can manage user and group accounts by selecting *System → Administration → Users and Groups* from the desktop mane

bar. An application for managing the properties of user settings will start-up that provides a user friendly interface. However, if you do not have a desktop environment installed or would prefer to manage your users and groups from the command line then here are some examples.

To create a new *ucam* group with the group ID 335 you would enter the following command.

```
# addgroup --gid 335 alma
```

To create a new user called *ucammgr* whose primary group is *ucam*, user ID is 3060, home directory is */home/ucammgr* and default shell is the bash shell you would enter the following command.

```
# adduser --gid 335 --uid 3060 --home /home/ucammgr --shell /bin/bash ucam
```

Enter *ucammgr* when prompted to enter the full name or something more elaborate such as 'UCam System Manager' .

Or you can modify the properties of an existing user account. For example, if you wanted to add the *ucam* group to an existing user account you would enter the following command. WARNING! Make sure you include the '-a' option to append this new group to you current list of groups or it will overwrite the list of groups for the account .

```
# usermod -a -G ucam <username>
```

Read the man pages for *addgroup*, *adduser* and *usermod* to see the full list of options.

7.6 Disabling Intelligent Platform Management Interface (IPMI)

The Intelligent Platform Management Interface (IPMI) specification defines a set of common interfaces to a computer system which system administrators can use to monitor system health and manage the system. IPMI operates independently of the operating system and allows administrators to manage a system remotely even in the absence of an operating system or the system management software, or even if the monitored system is powered off, but connected to a power source. The Baseboard Management Controller (BMC) and can share the on board NIC. The MAC address is always MAC address + 2.

After installing the Debian operating system you may wish to disable this service especially if it is generating DHCP requests that may cause problems with your network switch configuration. The following commands describe how to install and load the necessary kernel modules and then use the *ipmitool* command line tool to reconfigure the BMC. You only need to reconfigure the BMC once and it will be permanently set even after reboots.

```
# aptitude install openipmi
# aptitude install ipmitool
# modprobe ipmi_msghandler
# modprobe ipmi_devintf
# modprobe ipmi_si
# ipmitool -I open sensor list
# ipmitool lan set 1 ipsrc bios
```

7.7 Mounting A Remote File System Using SSHFS

Create a new directory in your home directory to be used as the mount point.

```
$ mkdir ~/data
```

We then use the *sshfs* command to securely mount the data directory */home/observer/ucam/data* on the *ucamserver* workstation logging in using the *observer* account. We also specify our newly created *~/data* as the mount point.

```
$ sshfs observer@ucamserver:/home/observer/ucam/data ~/data
```

This means we can now seamlessly interact with the remote files being securely served over SSH just as if they were local files. Try this moving to the mounted directory point and listing the contents.

```
$ cd ~/data
$ ls
```

8 Useful Web Pages

Astronomical Research Cameras (ARC)

http://www.astro-cam.com

Debian Linux OS

Main web page

http://www.debian.org

Download the network installer version of Debian

http://www.debian.org/distrib/netinst

Debian installation manual for the x86 platform

http://www.debian.org/releases/stable/i386

Enhanced Machine Controller (EMC) Knowledge Base Wiki

How to compile and install RTAI on Debian Lenny

http://wiki.linuxcnc.org/cgi-bin/emcinfo.pl?Debian_Lenny_Compile_RTAI

IRAF installation instructions on Debian/Ubuntu

http://geco.phys.columbia.edu/~rubab/iraf/iraf_step_by_step_installation

Real-Time Application Interface (RTAI)

https://www.rtai.org

UK Astronomy Technology Centre (UK ATC)

http://www.roe.ac.uk/ukatc

9 Appendix A

Contents of */usr/src/modules/rtai/debian/changelog*

```
rtai (3.5) unstable; urgency=low

  * Initial Release.

 -- maintainer <maintainer@domain.com>  Sat, 15 Sep 2007 00:39:32 +0300
```

Contents of */usr/src/modules/rtai/debian/compat*

```
5
```

Contents of */usr/src/modules/rtai/debian/control.modules.in*

```
Source: rtai
Section: devel
Priority: optional
Maintainer: maintainer <maintainer@domain.com>
Build-Depends: debhelper (>= 5)
Standards-Version: 3.7.2

Package: rtai-modules-_KVERS_
Architecture: any
Provides: rtai-modules
Depends: linux-image-_KVERS_
Description: rtai modules for Linux (kernel _KVERS_).
 This package contains the set of loadable kernel modules for the RTAI
 real- time extension.
 .
 This package contains the compiled kernel modules for _KVERS_
 .
 If you have compiled your own kernel, you will most likely need to build your
 own rtai-modules.  The rtai-source package has been provided for use with the
 Debian kernel-package utility to produce a version of rtai-module for your
 kernel.

Package: rtai-dev-_KVERS_
Architecture: any
Provides: rtai-headers
Depends: linux-headers-_KVERS_, _GCC_
Description: rtai headers for compiling real-time modules.
 This package contains the set of headers and helper scripts for the RTAI
 real-time extension.
```

Contents of */usr/src/modules/rtai/copyright*

```
This package was debianized by maintainer <maintainer@domain.com>
on Sat, 15 Sep 2007 00:39:32 +0300.

It was downloaded from http://www.rtai.org/

Upstream Authors: See RTAI Team at http://www.rtai.org/

Copyright: see AUTHORS

License:

  This program is free software; you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation; either version 2 of the License, or
  (at your option) any later version.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this package; if not, write to the Free Software
  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301 USA

The Debian packaging is (C) 2007, maintainer <maintainer@domain.com> and
is licensed under the GPL, see `/usr/share/common-licenses/GPL'.
```

Contents of */usr/src/modules/rtai/debian/rules*

```
#!/usr/bin/make -f

PACKAGE := rtai-modules
MA_DIR ?= /usr/share/modass
-include $(MA_DIR)/include/generic.make
-include $(MA_DIR)/include/common-rules.make

# architecture we're building for
DEB_HOST_ARCH ?= $(shell dpkg-architecture -qDEB_HOST_ARCH)

ARCH_TARGET := $(DEB_HOST_ARCH)-elf

ifeq ($(DEB_HOST_ARCH),amd64)
        ARCH_TARGET := x86_64-elf
endif

ifeq ($(DEB_HOST_ARCH),powerpc)
        ARCH_TARGET := powerpc-be-eabi
endif

ifeq ($(DEB_HOST_ARCH),mips)
        ARCH_TARGET := mipsisa32-be-elf
endif

ifeq ($(DEB_HOST_ARCH),mipsel)
        ARCH_TARGET := mipsisa32-be-elf
endif

ifeq ($(DEB_HOST_ARCH),arm)
        ARCH_TARGET := arm9-le-thumb-elf
endif
```

```
.PHONY: kdist_config fix_cdeps
fix_cdeps:
        cd debian ; cp control.modules.in control.modules.tmp 2>/dev/null ; \
                sed -e 's/_GCC_/$(CC)/g' < control.modules.tmp > control.modules.in ; \
                rm control.modules.tmp

kdist_config: fix_cdeps prep-deb-files

.PHONY: binary_modules binary-modules
binary-modules: binary_modules
binary-modules: kdist_config
        dh_testdir
        dh_testroot
        dh_clean -k

        # Build and install the module
        $(MAKE) install DESTDIR=$(CURDIR)/debian/rtai-dev-$(KVERS)

        # Move the kernel modules to the correct location
        mkdir -p $(CURDIR)/debian/rtai-modules-$(KVERS)/usr/realtime/modules
        mv -f $(CURDIR)/debian/rtai-dev-$(KVERS)/usr/realtime/modules/* $
(CURDIR)/debian/rtai-modules-$(KVERS)/usr/realtime/modules/

        # copy the Scilab macros in deb
        mkdir -p $(CURDIR)/debian/rtai-modules-$(KVERS)/usr/realtime/rtai-
lab/scilab/macros
        cp -fr $(CURDIR)/rtai-lab/scilab/macros/* $(CURDIR)/debian/rtai-modules-$
(KVERS)/usr/realtime/rtai-lab/scilab/macros/

        dh_installdebconf
        dh_installdocs
        dh_compress
        dh_installdeb
        dh_makeshlibs -prtai-dev-$(KVERS)
        dh_shlibdeps -prtai-dev-$(KVERS)
        dh_gencontrol -- -v$(VERSION)
        dh_md5sums
        dh_builddeb --destdir=$(DEB_DESTDIR)

.PHONY: kdist_clean

kdist_clean:
#       dh_testdir
#       dh_testroot
#       dh_clean
#       rm -f install.log
#       $(MAKE) -C $(CURDIR) distclean
```