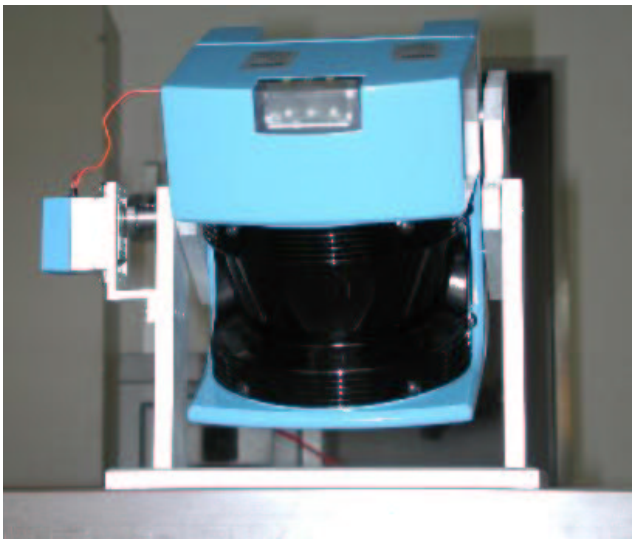
The AIS 3D Laser Range Finder



Manual

September 3, 2002

Hartmut Surmann
Andreas Nüchter

Steinbeis-Transferzentrum
Informationstechnische Systeme
Bonn-Sankt Augustin

Institut für Autonome Intelligente Systeme
Schloss Birlinghoven
53754 Sankt Augustin

phone: +49-2241-143460
fax: +49-2241-142384
email: stz481@stw.de

Contents

Chapter 1

Hardware Setup

The AIS 3D Laser Range Finder is shipped in different parts. The set contains:

- A 2D Laser Range Finder.
- A servo motor (Type: Volz Servo).
- A suspension unit.
- A PCMCIA serial card (500 kBaud).
- A CDROM containing basic software for acquiring 3D data point sets.
- This Manual.

For running the system one needs the following things:

- A Laptop computer.
- A 24V (1000mA) power adapter for the 2D Laser Range Finder. As an alternative a battery / accumulator can be used.
- A 6V (1000mA) power adapter for the servo motor. As an alternative a battery / accumulator can be used.
- Cables for connecting the power supplies with the 2D Laser Range Finder and the servo motor. r One cable for connecting the servo motor to the computers parallel port and a cable connecting the serial card to the scanner.

1.1 System Overview

Figure 1.1 explains the setup of the AIS 3D Laser Range Finder. The laptop has to be attached to the scanner using the PCMCIA serial card and the servo must be connected to the parallel port. The scanner and the motor need separate power adapters.

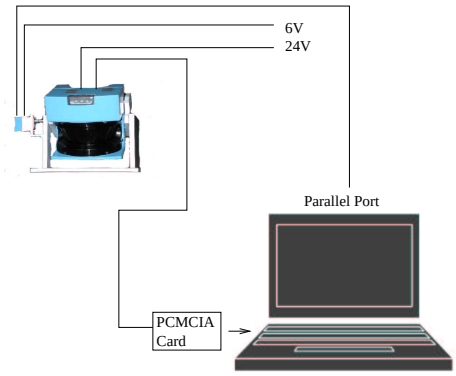


Figure 1.1: The system setup.

1.2 Linux Laptop

For using the AIS 3D Laser Range Finder you need a laptop computer running the Linux operating system. We recommend using a reasonable fast laptop, e.g. Pentium-III 800MHz. Furthermore the laptop must have a parallel port and at least one PCMCIA slot. The servo motor will be connected to the parallel port, the high speed serial card has a PCMCIA interface.

On the following website you find a list of laptops and whether Linux is working or not on these machines:
`http://www.linux-laptop.net/`

The first step for using the AIS 3D Laser Range Finder is to get a laptop computer, which is capable to run under Linux.

Once you have a laptop you must install the Linux operating system. Linux is free software (GNU Public License), but for convenience we got a commercial distribution. We choose the S.U.S.E. distribution for Linux (`http://www.suse.com`). S.U.S.E. sells complete configured laptops, too.

The second step is to install the laptop with Linux. You should follow the instructions of your Linux distribution.

1.3 Cabling Instruction

The servo motor has to be connected to the Laptops parallel port. Please use the following figure for making a cable:

Parallel Port Pin	Servo Pin
1 (signal)	1 (yellow)
25 (GND)	3 (brown) and GND power adapter
	3 (red) and +6V power adapter

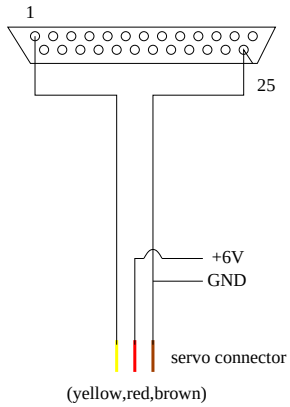


Figure 1.2: Cable for connecting the servo motor to the computers parallel port and the power adapter.

The following cable (figure 1.3) is for connecting the 2D Laser Range Finder to the PCMCIA serial card. Make sure that you have connected Pin 7 and Pin 8 on the scanner side. This makes the scanner transferring the data with 500kBaud over RS422. Otherwise the scanner has a RS232 interface, which is too slow.

Serial Port Pin	Scanner Pin
14 (TxD-)	1 (RxD-)
2 (TxD+)	2 (RxD+)
3 (RxD+)	3 (TxD+)
15 (RxD-)	4 (TxD-)
7 (GND)	5 (GND)
	7 — 8

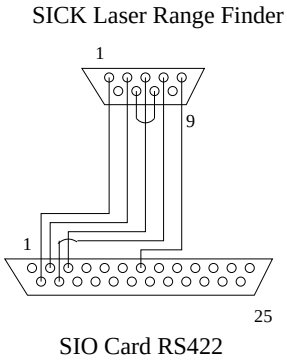


Figure 1.3: Cable for connecting the SICK 2D Laser Range Finder to the PCMCIA serial card.

Finally you need a power cable for connecting the 2D Laser Range Finder to a 24V power source. Please look at figure 1.4.

Power Source	Scanner Pin
GND	1
+24V	3

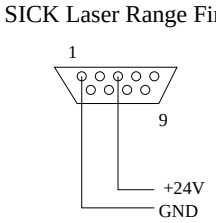


Figure 1.4: Cable for connecting the SICK 2D Laser Range Finder to the power adapter.

Chapter 2

Software Installation

2.1 Installing Real Time Linux

For using the AIS 3D Laser Range Finder you have to install Real Time Linux. The Real Time Linux (`rtlinux-3.0.tar.gz` is included in the CDROM in the `rtl` directory. This package contains the required modules and a kernel patch. If you don't want to patch a kernel you might install a prepatched kernel, which is also on the CDROM (`rtlinux_kernel_2_4.tar.gz`).

The following Installation procedure should install this Linux to the laptop. This is basically the installation procedure which comes with the Real Time Linux package. This procedure requires to patch a linux kernel.

> This guide is free; 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. (c) Phil Wilshire 2000. Modified by Victor Yodaiken for RTLinux2.2. Modified and corrected by Phil Wilshire. Modified for the AIS 3D Laser Range Finder by Andreas Nüchter 2002.

2.1.1 Unpack the kit

1. Get a clean kernel from: ftp.us.kernel.org. Use the 2.4.0 Version of the Linux kernel.
2. Get the patch file for that kernel. This patch (`kernel_patch-2.4`) is included in `rtlinux-3.0.tar.gz`.
3. Move your current copy of `/usr/src/linux` (if it is not a link) to:
   `/usr/src/oldlinux`
   Check with:

`ls -l /usr/src`
Links look like this:
`ls -l /usr/src/linux`
`lrwxrwxrwx 1 root root 17 Aug 17 1998 /usr/src/linux -> va-linux-2.0.34-1`
(In this case, just delete the link and do the untar.)
If you have to move a tree do:
`cd /usr/src`
`mv linux oldlinux`

4. Create a working directory. I use: `/usr/src`
   `cd /usr/src`
   (You may need to do this as root.)
   This will allow you as a user to access this.
   (NOTE: Do most of your work as a regular user... it's safer.)

5. Untar the RTLinux tarball into: `/opt/rtl`
   For example, if the tarball is in: `/cdrom/rtl/rtlinux-3.0.tgz`
   `cd /usr/src/`
   `tar xvzf /cdrom/rtl/rtlinux-2.4.tgz`
   This will create: `/usr/src/rtlinux-2.4`
   Loosen up the restrictions on this dir:
   `chmod -R a+rw rtlinux-3.0`

6. Look in the INSTALL file and follow the instructions. For example:

   ```
   1. put a fresh copy of Linux 2.4.0 in this directory under the name "linux".

   2. cd linux
     patch -p1 < ../kernel_patch
   ```

   This means take the `linux-2.4.0.tar.gz` tarball and put it where the install guide suggests, i.e., if the tarball is in:
   `/cdrom/rtl/linux/linux-2.4.0.tar.gz`

   ```
   cd /usr/src/rtlinux-3.0
   tar xvzf /cdrom/rtl/linux/linux-2.4.0.tar.gz
   ```

2.1.2 Patch the kernel

1. Now you can patch.

   ```
   cd /usr/src/rtlinux-3.0/linux
   patch -p1 < ../kernel_patch
   ```

2. Check for Rejects.

   There should be no none nada rejects.

   Here is an example of a reject...

   ```
   Patching file arch/i386/kernel/irq.c using Plan A...
   Hunk #1 failed at 39.
   Hunk #2 succeeded at 236.
   Hunk #3 succeeded at 280.
   Hunk #21 succeeded at 1324.
   1 out of 21 hunks failed--saving rejects to arch/i386/kernel/irq.c.rej
   ```

   You find any rejects like this:

   ```
   cd /usr/src/rtlinux-3.0/linux
   find ./ -name *.rej
   ./arch/i386/kernel/irq.c.rej
   ```

   You can only patch once (well sort of), so start again if you trip up.

   Cleanup:

   ```
   cd /usr/src/rtlinux-3.0/
   rm -rf linux
   tar xvzf /cdrom/rtl/linux/linux-2.4.0.tar.gz
   ```

2.1.3 Configure and compile the kernel

1. Look at what's running.

   Now we need to configure and compile the patched kernel. First, we need to look at what's running:

```
[wilshire@demosbc2 linux]$ /sbin/lsmod
Module Size Used by
3c59x 18952 1
```

Then we need to configure our new kernel. We see the modules used and make sure that we configure the new kernel for the same ones.

2. Configure the kernel.

```
cd /usr/src/rtlinux-3.0/linux
make menuconfig
```

Check options as required:

```
Code maturity level options ---> yes

Processor type and features ---> (PPro/6x86MX) Processor family
 [*] Symmetric multi-processing support

Loadable module support --->
 [*] Enable loadable module support
 [ ] Set version information on all symbols for modules
 [ ] Kernel module loader

General setup ---> accept defaults

Plug and Play support ---> accept defaults

Plug and Play support ---> accept defaults

Networking options ---> accept defaults

SCSI support ---> accept defaults

Network device support --->

Ethernet (10 or 100Mbit) --->
 [*] 3COM cards
 <M> 3c590/3c900 series (592/595/597) \"Vortex/Boomerang\" support
```

```
 (in this case)

Kernel hacking --->
 [*] Magic SysRq key
```

Right. Having done all this...
Save the configuration.
Build the kernel.

3. Compile the kernel.
   `[wilshire@demosbc2 linux]$ make dep; make clean; make bzImage`
   Go for a cup of tea , read your mail, write a manual... 10 minutes or so. You will end up with a new Kernel, a new Symbol.map, etc.

   BEFORE you do anything else: Set your system up to use this new kernel.

   Go to root:
   `su`

4. Make/Install the modules.
   `make modules; make modules_install`
   Watch the result here:

```
[root@pasrack3 linux]# make modules_install
Installing modules under /lib/modules/2.4.0-rtl3.0/net
Installing modules under /lib/modules/2.4.0-rtl3.0/misc
```

   The `/lib/modules/2.40-rtl3.0` shows you that just about every thing is well.

2.1.4 Set up boot / lilo

1. Copy the boot image to the right place.
   `cp arch/i386/boot/bzImage /boot/bzImageRTL30`

2. Copy the Symbol.map to the right place.
   `cp Symbol.map /boot/Symbol.mapRTL30`

3. Make the new symbol.map active.
   `rm /boot/Symbol.map`
   `ln -s /boot/Symbol.mapRTL22 /boot/Symbol.map`

4. Edit `/etc/lilo.conf`. It should look like (We assume that your hard drive is connected to the first ide channel, thus it is the device `/dev/hda`. SCSI drives are usually `/dev/sda` or `/dev/sdb`:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50

image=/boot/bzImageRTL22
label=linuxRTL30
root=/dev/hda1
append=\"mem=126M\"
read-only

image=/boot/bzImage
label=linux
root=/dev/hda1
append=\"mem=126M\"
read-only
```

   (But keep another copy around, just in case.)
   Run Lilo.

```
[root@demosbc2 linux]# /sbin/lilo
Added linuxRTL22 *
Added linux
```

2.1.5 Boot

1. Now is the time to reboot.

   Having rebooted we need to move the linux source link to our new source tree.

2. We also should create an rtl tree.

```
rm /usr/src/linux
ln -s /usr/src/rtlinux-3.0/linux /usr/src/linux
```

```
rm /usr/src/rtl
ln -s /usr/src/rtlinux-3.0 /usr/src/rtl
```

2.1.6 Make the modules

1. Check and adjust the date.

   ```
   [root@demosbc2 rtl]# date
   Wed Nov 10 23:14:18 PST 1999
   [root@demosbc2 rtl]# date 11111452
   Thu Nov 11 14:52:00 PST 1999
   ```

2. Now we can make the rtl modules.
   `cd /usr/src/rtlinux-3.0/rtl`
   `make`
   Now you need to become root and do `make install`:

   ```
   [root@demosbc2 rtl]# make install
   Testing for the mbuff device... mknod /dev/mbuff c 10 254
   Testing for FIFOs... already existent.

   Installing modules in /lib/modules/2.4.0-rtl3.0/misc
   /sbin/depmod -a
   Installing man pages to /usr/local/man
   Installing header /usr/include/rtlinux
   install -c -m 644 rtl.mk /usr/include/rtlinux;
   ```

3. Set up your path. Try: `lsmod`
   If you get:

   ```
   [root@demosbc2 fp]# lsmod
   bash: lsmod: command not found
   ```

   Do this:
   `export PATH=/sbin:/$PATH`
   Then try again.

   ```
   [root@demosbc2 fp]# lsmod
   Module Size Used by
   3c59x 19176 1
   ```

   That's it for the main install.

2.1.7 Run the examples

Now you can run the examples.

```
cd examples
cd fp
make
mv rt_process.o fp_tasks.o

export PATH=/sbin:$PATH <- only if /sbin is not in your path
chmod a+x ../../insrtl  <- you need to do this only once
chmod a+x ../../rmrtl   <- and you need to do this only once

cd ../../; ./insrtl
cd examples/fp
make test

[root@demosbc2 fp]# make test
insmod fp_tasks.o
./fptest                <- use ^C to quit this
make: *** [test] Interrupt

[root@demosbc2 fp]# lsmod
Module Size Used by
fp_tasks 892 0 (unused)
rtl_fifo 5944 0 (unused)
rtl_posixio 6952 0 [rtl_fifo]
rtl_sched 35364 0 [fp_tasks rtl_posixio]
rtl_time 8424 0 [fp_tasks rtl_sched]
3c59x 19176 1

[root@demosbc2 fp]# rmmod fp_tasks

[root@demosbc2 examples]# cd ../frank
[root@demosbc2 frank]# make
[root@demosbc2 frank]# make test
.....
FIFO 2: Zappa
FIFO 1: Frank
```

```
FIFO 2: Zappa
FIFO 2: Zappa
FIFO 2: Zappa
FIFO 1: Frank
......
frank_app: now sending commands to stop RT-tasks
```

This all works.

```
[root@demosbc2 frank]# cd ../hello
[root@demosbc2 hello]# make
[root@demosbc2 hello]# make test
```

This is the simplest RTL program.

First we remove any existing rtl-modules. You may see error warnings from `make` - ignore them.

```
Type <return> to continue.

rmmod sound
rmmod: module sound not loaded
make: [test] Error 1 (ignored)
rmmod rt_process
rmmod: module rt_process not loaded
make: [test] Error 1 (ignored)
rmmod frank_module
rmmod: module frank_module not loaded
make: [test] Error 1 (ignored)
(cd ../../; ./rmrtl)

Now insert the fifo and scheduler.

Type <return> to continue.

(cd ../../; ./insrtl)

Now start the real-time tasks module.

Type <return> to continue.
```

```
Now, let's stop the application.

Type <return> to finish.

[root@demosbc2 hello]#
```

Look on the console to see any messages:

```
cd ../measurements
make
make test

min: 9152, max: 17632
```

Do this to interrupt:

```
^C
make: *** [test] Interrupt

[root@pasrack3 measurements]#
```

To look at some stuff, try this:

```
./monitor > m.out&
sleep(25)
./histplot m.out m.res
cat m.res | more
```

2.2 Installing the PCMCIA module

After you have installed a Real Time Linux kernel and verified that the real time option works you have to set up the serial device. Thus the next step of software installation is to install the PCMCIA modules. The latest version of Linux PCMCIA drives is included on the CDROM (`pcmcia-cs-3.1.34.tar.gz`) in the `rtl` directory. The following part of this document is a shortened version of the second section of the PCMCIA HOWTO from David Hinds.

2.2.1 Compilation and Installation

Prerequisites and kernel setup

Before starting, you should think about whether you really need to compile the PCMCIA package yourself. All common Linux distributions come with pre-compiled driver packages. Generally, you only need to install the drivers from scratch if you need a new feature of the current drivers, or if you've updated and/or reconfigured your kernel in a way that is incompatible with the drivers included with your Linux distribution. While compiling the package is not technically difficult, it does require some general Linux familiarity.

The following things should be installed on your system before you begin:

- A 2.0, 2.2, or 2.4 series kernel source tree. The Real Time Linux source tree works well.
- An appropriate set of module utilities.
- (Optional) the "XForms" X11 user interface toolkit.

You need to have a complete linux source tree for your kernel, not just an up-to-date kernel image. The driver modules contain some references to kernel source files. While you may want to build a new kernel to remove unnecessary drivers, installing PCMCIA does not require you to do so.

In the Linux kernel source tree, the Documentation/Changes file describes the versions of all sorts of other system components that are required for that kernel release. You may want to check through this and verify that your system is up to date, especially if you have updated your kernel. If you are using a development kernel, be sure that you are using the right combination of shared libraries and module tools.

When configuring your kernel, if you plan on using a PCMCIA ethernet card, you should turn on networking support but turn off the normal Linux network card drivers, including the "pocket and portable adapters". The PCMCIA network card drivers are all implemented as loadable modules. Any drivers compiled into your kernel will only waste space.

If you want to use SLIP, PPP, or PLIP, you do need to either configure your kernel with these enabled, or use the loadable module versions of these drivers. There is an unfortunate deficiency in the kernel config process in 1.2.X kernels, in that it is not possible to set configuration options (like SLIP compression) for a loadable module, so it is probably better to just link SLIP into the kernel if you need it.

This package includes an X-based card status utility called `cardinfo`. This utility is based on a freely distributed user interface toolkit called the XForms Library. This library is available as a separate package with most Linux distributions. If you would like to build `cardinfo`, you should install XForms and all the normal X header files and libraries before configuring the PCMCIA package. This tool is completely optional.

Installation

Here is a synopsis of the installation process:

- Unpack `pcmcia-cs-3.1.?.tar.gz` in `/usr/src`.
- Run `make config` in the new `pcmcia-cs-3.1.?` directory.
- Run `make all`, then `make install`.
- Customize the startup script and the option files in `/etc/pcmcia` for your site, if needed.

If you plan to install any contributed client drivers not included in the core PCMCIA distribution, unpack each of them in the top-level directory of the PCMCIA source tree. Then follow the normal build instructions. The extra drivers will be compiled and installed automatically.

Running `make config` prompts for a few configuration options, and checks out your system to verify that it satisfies all prerequisites for installing PCMCIA support. In most cases, you'll be able to just accept all the default configuration options. Be sure to carefully check the output of this command in case there are problems. The following options are available:

Linux kernel source directory? This is the location of the source tree for the kernel you want to use with PCMCIA. Often this is `/usr/src/linux`, but the default location depends on what Linux distribution you're using (or on where you've chosen to place your kernel source tree). You should set this directory pointer to the Real Time Linux kernel source.

Build "trusting" versions of card utilities? Some of the support utilities (`cardctl` and `cardinfo`) can be compiled either in "safe" or "trusting" forms. The "safe" forms prevent non-root users from modifying card configurations. The "trusting" forms permit ordinary users to issue commands to suspend and resume cards, reset cards, and change the current configuration scheme. The default is to build the safe forms.

Include 32-bit (Cardbus) card support This option must be selected if you wish to use 32-bit CardBus cards. It is not required for CardBus bridge support, if you only plan to use 16-bit PC Cards.

Include PnP BIOS resource checking? This builds additional code into the PCMCIA core module to communicate with a system's PnP BIOS to obtain resource information for built-in "motherboard" devices (serial and parallel ports, sound, etc), to help avoid resource conflicts. If enabled, some extra resource files will be created under `/proc/bus/pccard`, and the lspnp and setpnp tools can be used to view and manipulate PnP BIOS devices. However, this setting causes problems on some laptops and is not turned on by default.

Module install directory? The directory that new kernel modules will be installed into. Normally this should be the subdirectory of `/lib/modules` that matches your kernel version.

How to set kernel sprecific options? There are a few kernel configuration options that affect the PCMCIA tools. The configuration script can deduce these from the running kernel (the default and most common case). Alternatively, if you are compiling for installation on another machine, it can read the configuration from a kernel source tree, or each option can be set interactively.

The Configure script can also be executed non-interactively, for automatic builds or to quickly reconfigure after a kernel update. Some additional less-frequently-used options can be only be set from the command line. Running `Configure --help` lists all available options.

Running `make all` followed by `make install` will build and then install the kernel modules and utility programs. Kernel modules are installed under `/lib/modules/<version>/pcmcia`. The `cardmgr` and `cardctl` programs are installed in `/sbin`. If `cardinfo` is built, it is installed in `/usr/bin/X11`.

Configuration files will be installed in the `/etc/pcmcia` directory. If you are installing over an older version, your old config scripts will be backed up before being replaced. The saved scripts will be given an `*.O` extension.

If you don't know what kind of host controller your system uses, you can use the probe utility in the `cardmgr/` subdirectory to determine this. There are two major types: the Databook TCIC-2 type and the Intel i82365SL-compatible type.

Startup options

The PCMCIA startup script recognizes several groups of startup options, set via environment variables. Multiple options should be separated by spaces and enclosed in quotes. Placement of startup options depends on the Linux distribution used. They may be placed directly in the startup script, or they may be kept in a separate option file. See the "Notes about specific Linux distributions" for specifics. The following variables can be set:

PCMCIA This variable specifies whether PCMCIA support should be started up, or not. If it is set to anything other than `yes`, then the startup script will be disabled.

PCIC This identifies the PC Card Interface Controller driver module. There are two options: `tcic` or `i82365`. Virtually all current controllers are in the `i82365` group. This is the only mandatory option setting.

PCIC_OPTS This specifies options for the PCIC module. Some host controllers have optional features that may or may not be implemented in a particular system. In some cases, it is impossible for the socket driver to detect if these features are implemented. See the corresponding man page for a complete description of the available options.

CORE_OPTS This specifies options for the `pcmcia_core` module, which implements the core PC Card driver services. See `man pcmcia_core` for more information.

CARDMGR_OPTS This specifies options to be passed to the `cardmgr` daemon. See `man cardmgr` for more information.

SCHEME If set, then the PC Card configuration scheme will be initialized to this at driver startup time. See the "Overview of the PCMCIA configuration scripts" of the PCMCIA-HOWTO for a discussion of schemes.

System resource settings

Card Services should automatically avoid allocating IO ports and interrupts already in use by other standard devices. It will also attempt to detect conflicts with unknown devices, but this is not completely reliable. In some cases, you may need to explicitly exclude resources for a device in `/etc/pcmcia/config.opts`.

Here are some resource settings for specific laptop types. View this list with suspicion: it may give useful hints for solving problems, but it is inevitably out of date and certainly contains mistakes.

- On the AMS SoundPro, exclude irq 10.
- On some AMS TravelPro 5300 models, use memory 0xc8000-0xcffff.
- On the Chicony NB5, use memory 0xda000-0xdffff.
- On the Compaq Presario 1020, exclude port 0x2f8-0x2ff, irq 3, irq 5.
- On the Dell Inspiron 7000, exclude irq 3, irq 5.
- On the Dell Inspiron 8000, exclude port 0x800-0x8ff.
- On the Fujitsu C series, exclude port 0x200-0x27f.
- On the HP Omnibook 4000C, exclude port 0x300-0x30f.
- On the HP Omnibook 4100, exclude port 0x220-0x22f.
- On the IBM ThinkPad 380, and maybe the 385 and 600 series, exclude port 0x230-0x233, and irq 5.
- On IBM ThinkPad 600 and 770 models with internal modems, exclude port 0x2f8-0x2ff.
- On the IBM ThinkPad 600E and 770Z, change the high memory window to 0x60000000-0x60ffffff.
- On the Micron Millenia Transport, exclude irq 5, irq 9.
- On the NEC Versa M, exclude irq 9, port 0x2e0-2ff.
- On the NEC Versa P/75, exclude irq 5, irq 9.
- On the NEC Versa S, exclude irq 9, irq 12.
- On the NEC Versa 6000 series, exclude port 0x2f8-0x33f, irq 9, irq 10.
- On the NEC Versa SX, exclude port 0x300-0x31f.

- On the ProStar 9200, Altima Virage, and Acquiline Hurricane DX4-100, exclude irq 5, port 0x330-0x35f. Maybe use memory 0xd8000-0xdffff.
- On the Siemens Nixdorf SIMATIC PG 720C, use memory 0xc0000-0xcffff, port 0x300-0x3bf.
- On the TI TravelMate 5000, use memory 0xd4000-0xdffff.
- On the Toshiba Satellite 4030CDS, exclude irq 9.
- On the Toshiba T4900 CT, exclude irq 5, port 0x2e0-0x2e8, port 0x330-0x338.
- On the Toshiba Tecra 8000, exclude irq 3, irq 5, irq 9.
- On the Twinhead 5100, HP 4000, Sharp PC-8700 and PC-8900, exclude irq 9 (sound), irq 12.
- On an MPC 800 Series, exclude irq 5, port 0x300-0x30f for the CD- ROM.

Notes about specific Linux distributions

Debian Debian uses a System V boot script arrangement. The PCMCIA startup script is installed as `/etc/init.d/pcmcia`. New packages use `/etc/default/pcmcia` for startup options; older versions used `/etc/pcmcia.conf` for this purpose. Debian's syslog configuration will place kernel messages in `/var/log/messages` and `cardmgr` messages in `/var/log/daemon.log`.

Debian distributes the PCMCIA system in two packages: the `pcmcia- cs` package contains `cardmgr` and other tools, man pages, and configuration scripts; and the `pcmcia-modules` package contains the kernel driver modules.

Starting with 3.1.25, a clean PCMCIA install will identify Debian systems and create a special network.opts file that, in the absence of other network configuration settings, uses Debian's `ifup` and `ifdown` commands to configure a network card based on settings in `/etc/network/interfaces`.

Red Hat, Caldera, Mandrake These distributions use a System V boot script organization. The PCMCIA startup script is installed as `/etc/rc.d/init.d/pcmcia`, and boot options are kept in `/etc/sysconfig/pcmcia`. Beware that installing the Red Hat package may install a default boot option file that has PCMCIA disabled. To enable PCMCIA, the `PCMCIA` variable should be set to `yes`. Red Hat's default `syslogd` configuration will record all interesting messages in `/var/log/messages`.

Red Hat's PCMCIA package contains a replacement for the network setup script, `/etc/pcmcia/network`, which meshes with the Red Hat `linuxconf` configuration system. This is convenient for the case where just one network adapter is used, with one set of network parameters, but does not have the full flexibility of the regular PCMCIA network script. Compiling and installing a clean PCMCIA source distribution will overwrite the network script, breaking the link to the Red Hat tools. If you prefer using the Red Hat tools, either use only Red Hat RPM's, or replace `/etc/pcmcia/network.opts` with the following:

```
if [ -f /etc/sysconfig/network-scripts/ifcfg-$2 ] ; then
    start_fn () {
        . /etc/sysconfig/network-scripts/ifcfg-$1
        if [ "$ONBOOT" = "yes" ] ; then /sbin/ifup $1 ; fi
    }
    stop_fn () {
        /sbin/ifdown $1
    }
fi
```

Starting with the 3.1.22 release, the PCMCIA installation script will automatically append a variant of this to the default network.opts file, so this problem should no longer be an issue.

If you do use `linuxconf` (or `netconf`) to configure your network interface, leave the `kernel module`, `I/O port`, and `irq` parameters blank. Setting these parameters may interfere with proper operation of the PCMCIA subsystem.

At boot time, when the Red Hat network subsystem starts up, it may say `Delaying eth0 initialization` and `[FAILED]`. This is actually not a failure: it means that this network interface will not be initialized until after the PCMCIA network device is configured.

Red Hat bundles their slightly modified PCMCIA source distribution with their kernel sources, rather than as a separate source package. When preparing to build a new set of PCMCIA drivers, you will generally want to install Red Hat's kernel-source RPM (`kernel- source-*.i386.rpm`), and not the kernel SRPM (`kernel-*.src.rpm`). The SRPM is tailored for building their kernel RPM files, which is not exactly what you want. With Red Hat 7.0, the kernel-source RPM also includes a mis-configured PCMCIA source tree; if you want to use it, delete their PCMCIA `config.out` file and re-do `make config`.

Slackware Slackware uses a BSD boot script arrangement. The PCMCIA startup script is installed as `/etc/rc.d/rc.pcmcia`, and boot options are specified in `rc.pcmcia` itself. The PCMCIA startup script is invoked from `/etc/rc.d/rc.S`.

SuSE SuSE uses a System V init script arrangement, with init scripts stored under `/etc/init.d`. The PCMCIA startup script is installed as `/etc/init.d/pcmcia`, and startup options are kept in `/etc/rc.config`. Before release 7.0, init scripts were kept under `/sbin/init.d`. In early SuSE releases (pre-5.3), the PCMCIA startup script was somewhat limited and did not allow PCMCIA startup variables to be overridden from the lilo boot prompt.

To look up current PCMCIA issues in SuSE's support database, go to `<http://sdb.suse.de/cgi-bin/sdbsearch_en.cgi?stichwort=PCMCIA>`.

2.2.2 Testing the PCMCIA interface

If the PCMCIA interface is installed correctly then you will hear two "Beeps" when inserting the serial card. In this case the `cardmgr` detects the new card and the appropriate module is loaded sucessfully. The command `lsmod` shows a list of loaded modules. In this list you should find the modules `pcmcia_core`, `i82365`, `ds` and `serial_cs`. The latter module is the device driver for the serial card.

If nothing happens when you insert the serial card then the `cardmgr` is not running. In this case you should try to insert the modules by hand using the command `modprobe` or `insmod -f`. Verify that the modules are loaded. If they can't be loaded follow the instructions from the PCMCIA-HOWTO for troubleshooting. After the `serial_cs` module is loaded you should execute `/etc/pcmcia/serial start ttyS2` which makes the new serial port the 3rd port on your computer.

If you hear a high and a low "Beep" the `cardmgr` is running and detecting the card. But the correct device driver can't be loaded. Try to load the device driver by hand using the `insmod -f` command and follow the the instructions from the PCMCIA-HOWTO for troubleshooting.

When the `serial_cs` module is loaded correctly the scanner can be connected the new serial port.

2.3 Installing the AIS 3D Laser Range Finder Software

Unlike installing Real Time Linux and the PCMCIA package, installing the AIS 3D Laser Range Finder software is very easy. Fist you have to create a directory for the software and the data output, e.g. `mkdir /home/scanner`, and go this new loacation (`cd /home/scanner`).

Create the directories `bin`, `src`, `doc`, `obj` and `dat` and copy the CDROM content of these directories to the new loacation. Copy the `Makefile` from the CD to `/home/scanner`. Finally, copy *your* `rtl.mk` file to the scanner directory. You will find this file in your Real Time Linux directory, for example try `cp /usr/rtlinux/rtl.mk /home/scanner`.

After done copying run, `make`. The whole scanner code should compile without warnings and errors. Now the software is completely installed and you can begin to work with the AIS 3D Laser Range Finder.

2.4 Testing the AIS 3D Laser Range Finder

Connect the AIS 3D Laser Range Finder and the servo motor with the Laptop and power up the scanner and motor. Go to the location of the scanner software, e.g. `cd /home/scanner/` and become root (`su`). Type `make init` and press return. This command inserts the required Real Time Linux Modules and you should notice that the servo is turning to its zero position. This

shows you Real Time Linux and the servo work. You can type `bin/setServo <value>`, where `<value>` is a number between 0 and 255. You should see the motor turning, but be carefull the turning is very fast. *If the 2D Laser Range Finder is already attached to suspension unit and the motor too fast turning can damage the motor!*

Please edit the configuration file (`bin/scanner.cfg`) now! See section 3.1.1 for details. In this file you should change the `Laser device` entry to the serial port of the installed 500 kBaud PCMCIA card.

Now you can run the scanner application! Type `bin/scanner s` for starting the scanning process. The light on the scanner should first turn to to "red" and then to "green". After scanning you will find the output in the `dat` directory (256 `.dat` files, two `.dxf` files and the file `points.txt`).

The last step is building the 3D scanner by mounting the susension unit to the 2D Laser Range Finder and the servo motor. If you run the scanner application again the files in the `dat` directory contain real 3D point information.

Chapter 3

Software Documentation

3.1 File Formats

3.1.1 The Configuration File

With the configuration file you can change the parameters the the AIS 3D Laser Range Finder. The following text is an example file:

```
# 3D-Laserscanner CONFIG-FILE
# Data directory
/home/bob/dat
# Step angle
1
# Scanner height
105
# Scan resolution
180
# Laser device
/dev/ttyS2
# Create DXF output
1
# from angle
-56.0
# to angle
68.0
```

Comment lines start with a `#` symbol.

The first entry is the data directory where all the output will be stored.

The second entry is the step angle. `1` is the finest resolution. This is the agle resolution of approximate $\frac{1}{3}$ degree. If you change this to a higher value like `3` you will get a lower angle reolution, e.g. one degree, and the scanning process will be faster, since less scan slices have to be acquired.

The third entry is the height of the scanner in centimeter. This value is needed for calculation of the 3D data points in the global coordinate system.

The next entry specifyies scan resolution. 180 and 360 are valid numbers here. Every scanned plane consists of 180 or 360 data points. The actual resolution is a bit lower due to the so called skip points (refer to the source code for details).

Then you can specify the serial device. This decive entry should point to the PCMCIA serial card with the connected scanner.

With the next entry you choose whether `.dxf` files shall be created (case `1`) or not (case `0`). Two `.dxf` can be created in the data directory, containing the all the data (`Points.dxf`) and a vertical section (`Aufriss0.dxf`).

The last two entries are angles, the so called "from angle" and "to angle". They specify the angle which is scanned and they are given by the setup (It depends on the servo and how you did mount the scanner.). One has to determine these angle to get correct 3D data.

3.1.2 The Source Files

The source files are stored in the `src/` directory. In detail they are:

`dxfutils.h`	The header file of the routines for DXF-output.
`dxfutils.c`	Code for writing `.dxf` files.
`elements.h`	We use the data structure `Point` for describing one in the 3D space. The structure is defined here.
`elements.c`	Contains the code asoziated with the data structure `Point`.
`elements.icc`	Ditto.
`getmycpuinfo.sh`	Script for retrieving the clock speed of the laptop.
`getmyparport.sh`	Script for retrieving the address of the parallel port.
`globals.h`	Global constants and definitions.
`laser.h`	Header file for the 2D Laser Range Finder interface.

`laser.c`	The 2D Laser Range Finder interface.
`mycpu.h`	Output of the script `getmycpuinfo.sh`.
`parport.h`	Output of the script `getmyparport.sh`
`rt_servo.c`	This file is the Real Time Linux servo module. It compiles to `obj/rt_servo.o` and can be inserted to a Real Time Linux Kernel using the command `insmod`.
`scan.h`	The header file for the class `Scan`. This class contains all scan points and the output routines.
`scan.c`	Implementation of the class `Scan`.
`scan.icc`	Ditto.
`scanner.c`	The `main` procedure is loacated in this file.
`scanner.icc`	Some tiny functions.
`servo.h`	This file contains the high level inferface (class `t_servo`) for the servo motor. The modules are responsible for communication with the Real Time module `rt_servo`.
`servo.c`	Implementation of the class `t_Servo`.
`servo1.icc`	Ditto.
`servodata.h`	Constants and definitions for both: The Real Time servo module (`rt_module`) and the high level interface (`servo.h`).
`setServo.c`	A small program for turning the servo motor.

3.1.3 The Data Output

The File `data.txt`

This file contains all 3D data points of one 3D scan. It is generated after the scanning. For example the first 3 lines of this file may look like:

```
12.6138 102.393 1.75865
12.557 102.211 1.88149
12.4964 102.029 2.00375
```

This means the first 3 range points have the (x, y, z)-coordinates $(12.6, 102.4, 1.76)$, $(12.6, 102.2, 1.9)$ and $(12.5, 102.0, 2.0)$. All units are in centimeter.

The `.dat` files

The `.dat` files contain all the scan slices. Up to 256 slices are taken during the scan. The first few lines of such a file look like the following text:

```
# 153 data with intensity information, current angle: -56
12.6138 3.14498 13 8709
12.557 3.36465 13 8684
```

In this case every slice contains 153 measured range points with intensity information. The current angle of the slice is minus 56 degree. The first data point has the 2D (!!!) coordinates of (12.6, 3.1) and the distance of 13 from the scanner. All units are in centimeter. The measured intensity of the returned light was 8709 units.

The DXF-Files

DXF-Files are also created by the scanner application. This format can be read with several commercial programs, e.g. Autodesk, 3D Studio Max, and noncomercial ones, e.g. QCad for Linux. Please look at the following website for a descritption of the file format:
`http://www.dcs.ed.ac.uk/home/mxr/gfx/3d/DXF.info`

The `pic.ppm` File

There is a picture created by the scanner software. It is called `pic.ppm` and stored in the `dat` The depth information is coded by the color. The image is distorted because the slices taken by the 2D laser range finder result in a line within the picture. One can view it using the program `xv`.

3.2 Software Structure

Real Time Linux is an operating system which runs ordinary Linux as a task with lowest priority. The Real Time servo module (`rt_servo`) – when inserted in the kernel – runs with highest priority. This module is completely separeted from the rest of the programs. The communication is handled through two FIFOs (`/dev/rtf1` and `/dev/rtf2`) in the class `t_servo`).

The main application creates two threads. The first thread is responsible for controlling the 2D Laser Range Finder using the function `get_laser_X_Y()` of the `laser.c` module. The servo is controlled here, too. The second process transforms the acquired data into 3D and creates the output files. Figure 3.1 gives an overview of the software architecture.

Please look at the source code and write your own application based on this code.

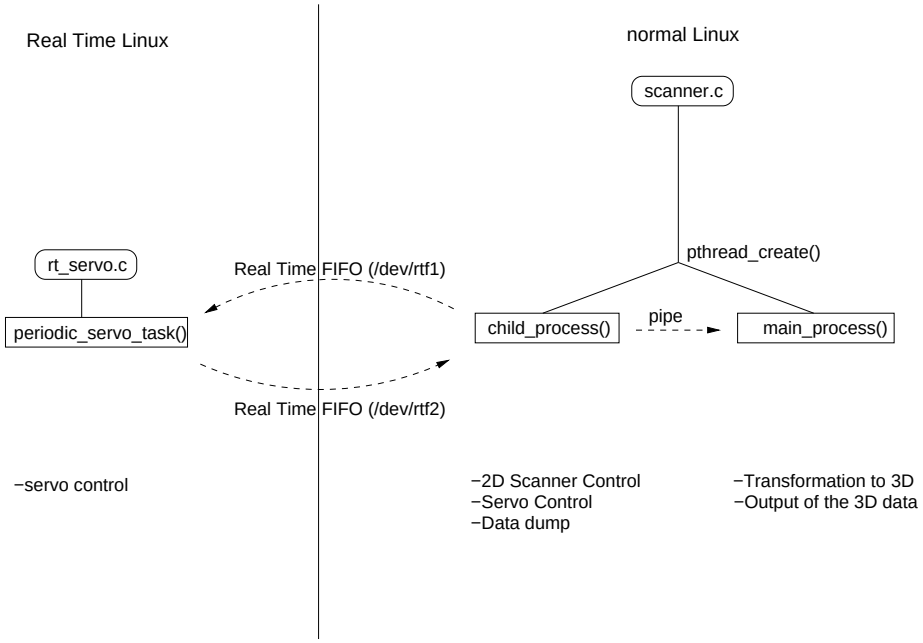


Figure 3.1: Software overview.