Linux Infrared HOWTO

Werner Heuser

The Infrared−HOWTO (former IR−HOWTO) provides an introduction to Linux and infrared devices and how to use the software provided by the Linux/IrDA project. This package uses IrDA(TM) compliant standards. IrDA(TM) is an industrial standard for infrared wireless communication, and most laptops made after January 1996 are equipped with an IrDA(TM) compliant infrared transceiver. Infrared ports let you communicate with printers, modems, fax machines, LANs, and other laptops. Speed ranges from 2400bps to 4Mbps. The Linux/IrDA stack supports IrLAP, IrLMP, IrIAS, IrIAP, IrLPT, IrCOMM, IrOBEX, and IrLAN. Several of the protocols are implemented as both clients and servers. There is also support for multiple IrLAP connections, via several IrDA(TM) devices at once. The Linux/IrDA project started at the end of 1997 and its status is still experimental, so please don't expect every feature working straight. AFAIK Linux/IrDA is the _only_ open source IrDA implementation currently available. Remote Control (RC) via infrared is not the aim of the project, though partly treated in this HOWTO.

Table of Contents

Table of Contents

Table of Contents

Table of Contents

1. About the Document and the Author

Better red, than dead. − Unknown AuthorEss

This document is based on the How to use part of the Linux/IrDA project homepage. I also included material provided by the Linux/IrDA core team, the Linux/IrDA mailing list and other sources.

The document is included in the LINUX DOCUMENTATION PROJECT.

The latest version of this document is available at MobiliX . Also you may find my Linux−Laptop−Guide and my other Linux HOWTOs there.

Mathieu Arnold <arn_mat@club−internet.fr> provides the IR−HOWTO in French .

The latest kernel I used is 2.2.3 and the latest irda−utils version is 0.8.8 .

I tried to check all information but I don't have all the necessary infrared hardware yet, so if something doesn't work for you, please don't blame me.

Please feel free to contact me for comments or questions about the HOWTO. I know this material is not finished or perfect, but I hope you find it useful anyway. For other questions and current information about Linux/IrDA please ask in the Linux/IrDA mailing list as explained below.

<Werner Heuser>

2. Status of the Document

I hadn't time to follow all the recent developments in Linux/IrDA. Therefore this document still needs a major rewrite, which I intend to do, when the Linux/IrDA source is included into the upcomming stable 2.4.x kernel. So you have to expect that some things described below don't work anymore or in another way. I hope this document may point you in the right direction though.

I. IrDA

Table of Contents

Chapter 1. About the Linux/IrDA Project

1.1. Project History

The project started at the end of 1997 with the name Linux/IrDA. Due to some troubles with the name IrDA, which is trademarked by the Infrared Data Association IrDA, the name was changed to Linux/IR. At the end of 1998 the the relationship between both became better and the name was changed to Linux/IrDA again. Since February 1999 the project is an official member of IrDA .

Companies and developers which are interested in joining these efforts should contact the at Linux/IrDA Project or me at < wehe@mobilix.org >.

1.1. Code History

The Linux/IrDA project has undertaken some changes in the program code, which you should know to understand some confusions.

For 2.0.x kernels Linux/IrDA support worked in a totally other way and is no longer supported by the Linux/IrDA project. Since 2.1.131 and 2.2.0 is part of the kernel. Please note that the status of the project just changed from experimental to stable with 2.2.10 kernels.

Some caveats are sometimes caused by changes of the following parameters (Hope I have got them alright now, they confused me sometimes, too):

- the major device number of the irda device changed from 61 to 161 (and from 60 to 160?)
- some task of **irmanager** are now achieved by **irattach**
- the device name **ircomm_tty** changed to **ircomm−tty**
- the device name **irnine** changed to **ircomm_new**

Chapter 2. Getting Started

2.1. Prerequisites

1. BIOS

   – Make sure your infrared port is enabled in the BIOS and check what interrupt and port address it uses. With some laptops it seems necessary to have Window$x installed to be able to set BIOS parameters.

2. Docking Station

   I have got reports, that connected to a docking station the infrared port was disabled.

3. Infrared Controller Chip

   – Make sure your infrared port is detected by the Linux kernel. For detailed information see the "Hardware Survey" section below.

4. modutils

   – Make sure you use modutils 2.1.x by **insmod −−version**. I use version 2.1.121.

5. Shared Libraries

   – The shared library **glibc2** aka **libc6** is recommended. There are also efforts to use **glibc2.1**, you may get this at ftp://ftp.funet.fi/pub/gnu/funet/ .

   – In some files you maybe have to change **#include <net/if_packet.h>** to **#include <linux/if_packet.h>** to get things to compile. This means using *kernel headers* instead of *glibc headers*. Please consult the mailing list archive, if your are interested in further information.

   – But `libc.so.5` should work, too.

   – the `libtool` and `automake` packages should be installed

   – I am not sure whether you need the `zlib` library if you use the data compression features.

   – What about `libglib−dev`?

6. IrLAN The latest release of **net−tools** (package **netbase** for Debian/GNU−Linux) is recommended, if you want to use an IrLAN connection. It's available from: ftp://ftp.netwinder.org/users/p/philb/net−tools/, http://www.tazenda.demon.co.uk/phil/net−tools/ and shortly from ftp://ftp.cs−ipv6.lancs.ac.uk/pub/Code/Linux/Net_Tools/ .
7. Graphical User Interface (GUI)

   Currently there are two graphical user interfaces for Linux/IrDA under development one for KDE and one for GNOME. See GUI chapter below. For the GNOME application you will need the Perl−GTK+ module and Python. You must also install all the development packages. To run Linux/IrDA I recommend to check the command line tools first, because the GUIs don't seem to be finished yet.

8. Security

   – Most important, you must **sync** your disks!!! Maybe you have to reboot your machine. Have you read the disclaimer?

9. Miscellaneous

   – Other useful progs: APSFILTER, EZ–Magic, MagicFilter or something similar for the printer configuration.

2.2. Kernel

Please read the Kernel−HOWTO to get more information about the compilation process. You'll find the Linux/IrDA code in:

`/usr/src/linux/net/irda` (protocol stuff)

`/usr/src/linux/drivers/net/irda` (device drivers)

`/usr/src/linux/include/net/irda` (header files)

2.2.1. General Parameters

− Make sure you use *kernel 2.2.x* sources. If unsure about your kernel version try **uname −r**.

− Get the latest kernel patch from the Linux/IrDA project http://irda.sourceforge.net. Or from the Alan Cox kernel series at ftp://ftp.linux.org.uk/pub/linux/alan/2.2/ . Put it into `/usr/src` or where else your kernel sources live and apply something like (replace **patch−2_2.0−irdaXXX** with the actual file name):

```
cd /usr/src
tar xvzf patch-2_2.0-irdaXXX.tar.gz
cd linux
patch -p1 -l < ../patch-2_2.0-irdaXXX
```

− Experimental support has to be enabled **CONFIG_EXPERIMENTAL**.

− Enable sysctl in "General Setup" **CONFIG_SYSCTL**.

− You should have *proc file system support* **CONFIG_PROC_FS**.

− Also *serial support* for the SIR features **CONFIG_SERIAL**.

− I am not sure whether there has to be *printer support* for using a printer with Linux/IrDA **CONFIG_PRINTER**. But I assume this feature is not necessary.

− *Networking support* _must_ be enabled **CONFIG_NET**.

− Make sure you have *module support* **CONFIG_MODULES** in your kernel! Test it e.g. with **lsmod**.

− Also **kerneld** support **CONFIG_KERNELD**. But **kmod** (CONFIG_KMOD) also works. A monolithic kernel seems to work, too. But modules are highly recommended.

− To use **irdadump** you probably have to set **CONFIG_PACKET**.

If you only apply the Linux/IrDA patch, you should not have to do a **make clean**, so that should save you some time. I suggest you do something like this:

```
make dep 38;38; make all 38;38; make modules 38;38; make install 38;38; make modules_install
```

If you get really strange errors, then try to rebuild from scratch after a **make clean**.

2.2.2. IrDA Specific Parameters

The following is my draft for `../linux-2.2.3/Documentation/Configure.help`, parts are from Dag Brattli and Andreas Butz. Please consult the latest available kernel documentation for current information and new drivers.

2.2.2.1. IrDA subsystem support

CONFIG_IRDA

IrDA(TM) is an industrial standard for infrared wireless communication. Infrared ports let you communicate with printers, modems, fax machines, LANs, and laptops. Speed ranges from 2400bps to 4Mbps. To use this features you need the irda_utils provided by the Linux/IrDA project http://irda.sourceforge.net . Further information you may find there and in the Linux−Infrared−HOWTO at http://mobilix.org/howtos.html . Currently it is recommended to build IrDA support as modules only. Please see Documentation/modules.txt. Please note the status of Linux/IrDA is still experimental.

2.2.2.1.1. IrDA protocols

- IrLAN protocol

  CONFIG_IRLAN

  Builds the IrDA network device. Use **ifconfig eth0 <IP−NUMBER>** to configure it. − Just say Y
- IrLAN client support

  CONFIG_IRLAN_CLIENT

  If you connect to infrared devices via IrLAN one has to be the server and the other the client. You can use both the client and the server at the same time. The first one to connect becomes the client. − Just say Y Note: The latest patch includes peer−to−peer support instead.
- IrLAN server support

  CONFIG_IRLAN_SERVER

If you connect to infrared devices via IrLAN one has to be the server and the other the client. You can use both the client and the server at the same time. The first one to connect becomes the client. − Just say Y Note: The latest patch includes peer−to−peer support instead.

• IrOBEX protocol

CONFIG_IROBEX

IrOBEX is a protocol for exchanging objects (files, vcards, etc.) over an infrared connection. You can use it to exchange files between linux and a PALM III. IrOBEX can also be used between two Linux boxes, Linux and Windows95, etc. − Just say Y

• IrCOMM protocol

CONFIG_IRCOMM

Over IrCOMM you may communicate with cellular phones, etc. To use this service you have to build a new device with **mknod /dev/irnine c 60 64**, which works like /dev/ttySx. − Just say Y ..Note: major and minor number are still not the official ones yet. For latest improvements (IrSocket is on the way!), please look at the page of Takahide Higuchi http://www.pluto.dti.ne.jp/~thiguchi/irda/

..Note: At the moment IrCOMM seems to crash your kernel easily, you should probably wait for the next patch. .. Note: **mknod /dev/irnine c 161 0** for newer versions! [WH]

• IrLPT client support

CONFIG_IRLPT_CLIENT

Say Y here if you want to build support for the IrLPT client protocol. If you want to compile it as a module, say M here and read Documentation/modules.txt. The IrLPT client protocol can be used to print documents to IrDA compatible printers like the HP−5MP, or IrLPT printer adapters like the ACTiSYS IR−100M. − Just say Y

• IrLPT server support

CONFIG_IRLPT_SERVER

Say Y here if you want to build support for the IrLPT server protocol. If you want to compile it as a module, say M here and read Documentation/modules.txt. The IrLPT server protocol makes it possible to use a Linux machine as an infrared printer server for other laptops. So if your Linux machine has a cable connection to a printer, then other laptops can use the Linux machine to print out documents using infrared communication. − Just say Y

2.2.2.1.2. IrDA protocol options

CONFIG_IRDA_OPTIONS

You may define some IrDA protocol options.

- Cache last

  LSAP CONFIG_IRDA_CACHE_LAST_LSAP

  Say Y here if you want IrLMP to cache the last LSAP used. This makes sense since most frames will be sent/received on the same connection. Enabling this option will save a hash−lookup per frame. If unsure, say Y.
- FAST RRs

  CONFIG_IRDA_FAST_RR

  Use this option if you want to send faster RR (Receive Ready) frames if the transmit queue is empty. This will give you much better latencies but will consume more power, because of the bouncing RR frame.
- Recycle RRs

  CONFIG_IRDA_RECYCLE_RR

  In the normal life of the IrLAP protocol, it sends a lot of small RR (Receive Ready) frames over the link (at least when it has nothing else to do). Saying Y to this option will make IrLAP recycle these frames thus avoiding many alloc_skb's and kfree_skb's. To do this it will only buffer one of these frame which is enough for the usual case.
- Debug information

  CONFIG_IRDA_DEBUG

  Say Y here if you want the IrDA subsystem to write debug information to your syslog. You can change the debug level in /proc/sys/net/irda/debug. If unsure, say Y (since it makes it easier to find the bugs).

2.2.2.1.3. IrDA compressors

CONFIG_IRDA_COMPRESSION

You may use the compression methods BZIP2 and BSD. These are not IrDA standard. This will allow two linux boxes to handshake compression. It should be compatible with other IrDA devices, although communication will not be compressed then.

- Deflate compression (experimental)

  CONFIG_IRDA_DEFLATE

  Say Y here if you want to build support for the Deflate compression protocol. If you want to compile it as a module, say M here and read Documentation/modules.txt. The deflate compression (GZIP) is exactly the same as used by the PPP protocol. Enabling this option will build a module called irda_deflate.o.

- BZIP2 compression

  CONFIG_IRDA_BZIP2

  Help not available yet.
- BSD compression

  CONFIG_IRDA_BSD

  Help not available yet.

2.2.2.2. Infrared−port device drivers

Three sorts of low level infrared drivers are available: serial, dongle and FIR. They will show up in /proc/net/dev (irda0) after initialisation.

2.2.2.2.1. IrTTY (uses serial driver)

Most IrDA chips support StandardInfraRed (SIR), which works up to 115200bps and emulates a serial port (16550A UART). On many laptops this port is detected by the serial support of the kernel, see **dmesg**. IrTTY connects the Linux/IrDA services to this port. − You should say Y here.

- Serial dongle support

  CONFIG_IRTTY_SIR

  Say Y here if you want to build support for the IrTTY line discipline. If you want to compile it as a module, say M here and read Documentation/modules.txt. IrTTY makes it possible to use Linux's own serial driver for all IrDA ports that are 16550 compatible. Most IrDA chips are 16550 compatible so you should probably say Y to this option. Using IrTTY will however limit the speed of the connection to 115200 bps (IrDA SIR mode). If unsure, say Y.

2.2.2.2.2. Dongle support

CONFIG_DONGLE

Currently four dongles (infrared adapters for the serial port) are supported. The dongle is an infrared device which may be connected to serial port, if you don't have built−in infrared support for your machine. If you use a dongle together with a laptop you maybe have to disable the IrDA support in the BIOS.

- ESI JetEye PC dongle

  CONFIG_ESI_DONGLE

  Say Y here if you want to build support for the Extended Systems JetEye PC dongle. If you want to compile it as a module, say M here and read Documentation/modules.txt. The ESI dongle attaches to the normal 9−pin serial port connector, and can currently only be used by IrTTY. To activate support for ESI dongles you will have to insert **irattach −d esi** in the /etc/irda/drivers script. http://www.extendsys.com/support/ftp/infrared.html
- ACTiSYS IR−220L and IR220L+ dongle

  CONFIG_ACTISYS_DONGLE

  Say Y here if you want to build support for the ACTiSYS IR−220L and IR220L+ dongles. If you want to compile it as a module, say M here and read Documentation/modules.txt. The ACTiSYS dongles attaches to the normal 9−pin serial port connector, and can currently only be used by IrTTY. To activate support for ACTiSYS dongles you will have to insert **irattach −d actisys** or **irattach −d actisys_plus** in the `/etc/irda/drivers` script. http://www.actisys.com
- Tekram IrMate 210B dongle

  CONFIG_TEKRAM_DONGLE

  Say Y here if you want to build support for the Tekram IrMate 210B dongle. If you want to compile it as a module, say M here and read Documentation/modules.txt. The Tekram dongle attaches to the normal 9−pin serial port connector, and can currently only be used by IrTTY. To activate support for Tekram dongles you will have to insert **irattach −d tekram** in the `/etc/irda/drivers` script. http://www.tekram.de/
-

  CONFIG_GIRBIL_DONGLE

  Say Y here if you want to build support for the Greenwich Instruments GirBIL dongle. If you want to compile it as a module, say M here and read Documentation/modules.txt. The Greenwich dongle attaches to the normal 9−pin serial port connector, and can currently only be used by IrTTY. To activate support for Greenwich dongles you will have to insert **irattach −d girbil** in the /etc/irda/drivers script. http://www.greenwichinst.com/

2.2.2.2.3. FIR support

FastInfraredSupport (FIR) needs a specific controller chip, which supports up to 4Mps. − Just say Y

- NSC PC87108

  CONFIG_NSC_FIR

NationalSemiConductor NSC PC87108 FIR chip e.g. used in the IBM Thinkpad 560X and ACTiSYS IR2000 dongle. Probably the NSC PC87338 FIR chip is also supported. The driver supports SIR, MIR and FIR (4Mbps) speeds. – Just say Y

- Winbond W83977AF (IR)

  CONFIG_WINBOND_FIR

  Winbond W83977AF (IR) FIR chip e.g. used in the Corel Netwinder PC. The driver supports SIR, MIR and FIR (4Mbps) speeds. – Just say Y
- Sharp UIRCC

  CONFIG_SHARP_FIR

  Say Y here if you want to build support for the Sharp UIRCC IrDA chipset. If you want to compile it as a module, say M here and read Documentation/modules.txt. This chipset is used by the Toshiba Tecra laptops.

2.3. Linux/IrDA−Utils

2.3.1. Compilation

- Use the latest source snapshot of **irda−utils** available at http://irda.sourceforge.net
- Untar the package with **tar xvzf irda−utils<VERSION>**. I recommend to do this in `/usr/src`.
- Do a **make depend**.
- Do a **make clean** (not necessary if you compile the package for the first time).
- Do a **make all** to build the binaries.
- Do a **make install**, this brings **irattach** and **irmanager** into the right place and installs some config files in `/etc/irda`.

A recommendation from Bjoern Hansson <Bjorn.Hansson@signal.uu.se>: If **make depend** fails on **stdef.h** and **stdarg.h** just add `−I/usr/lib/gcc−lib/i586−linux/egcs−2.90.29/include/` or the according path for your system to the **SYS_INCLUDES** line in `Makefile`.

To compile **irdadump** and **irdaping** (which are not necessary to get Linux/IrDA to work) you probably to tweek the source a little:

Dag Brattli: " The problem is that I'm including kernel header files which conflicts with the libc header files. So why do I need to include kernel header files? Well, the libc/glibc header files does not know that much about IrDA (yet). The reason is just that IrDA is quite new. I could have just defined the stuff in the program itself, but then people would be able to compile the stuff even if the definitions had changed in the kernel. I think its better that you get a compile error than some possible strange behaviour. If you don't want to wait until I figure out how to solve this stuff, you can just remove the linux header files and define the constants in your program yourself:

```
#define PF_IRDA          23
#define AF_IRDA          PF_IRDA

#define ARPHRD_IRDA      783
#define ETH_P_IRDA       0x0017
```

All the stuff above should not be changed, so it is probably safe to just hardcode them. I'll change the programs so they just includes the normal header files, and then defines these constants only if the header files didn't know about them. It should however be safe to include **linux/irda.h** ."

Though I never succeeded to compile all utilities without errors, I recommend to use at least the latest packages of **libtool**, **m4**, **autoconf**, and **automake** if you want to compile **irdadump**, **irdaping**, etc.

2.3.2. Precompiled Packages

NOKUBI Takatsugu provides an *unofficial* irda−utils Debian package (needs libc2.1). This package is part of the Potato Debian release. Also Mandrake 6.1, Redhat 6.1 and SuSE 6.1 contain RPM packages of irda−utils.

2.3.3. Contents of Linux/IrDA−Utils

2.3.3.1. irmanager

The **irmanager** is user−space daemon that is inspired and quite similar to the **cardmgr** used in the PCMCIA distribution. For example, if IrLMP discovers a remote device with IrLAN provider capabilities and no local IrLAN client has registered, then IrLMP will send an event to the IrManager and make it **modprobe** the module required. When application level clients are ready for communication and user−space configuration, they can also notify IrManager about this, so that it can execute the right script. For example will IrLAN send the event **EVENT_IRLAN_START** when the data channel is ready for exchanging Ethernet frames. When IrManager receives this event, it will execute **/etc/irda/network start <devname>** to configure the network interface. If you use the RedHat variant of it, it will in turn execute **/sbin/ifup<devname>** .

2.3.3.2. irattach

Usually **irattach** is started by **irmanager**. **irattach** attaches an IrDA capable tty to the basic services of Linux/IrDA. If **kerneld** or **kmod** are running, the modules **irda** and **irtty** are loaded automatically, if not you have to load them by hand in that order. It is recommended to start **irattach** in the background. To stop **irattach** just kill the process.

2.3.3.3. load_misc

A Perl script which loads a Linux/IrDA module and builds the according device in `/dev` using **mknod**.

2.3.3.4. /etc/irda

Configuration files, e.g. contains the serial port of the SIR driver:

```
drivers
network
network.opts
obex
```

You should at least configure the IR driver `drivers`:

```
#! /bin/sh
#
# drivers
#
# Initialize and shutdown IrDA device drivers.
#
# This script should be invoked with two arguments. The first is the
# action to be taken, either "start", "stop", or "restart".
#

action=$1
device=$2

case "${action:?}" in
start)
    irattach /dev/ttyS2          # The third serial port is an IrDA port
    # irattach /dev/ttyS0 -d esi # Attach a ESI dongle to the first serial port
    # insmod pc87108             # If your machine as a pc87108 FIR chipset
    ;;
stop)
    killall irattach             # ... or something. Currently not used
    ;;
restart)
    /sbin/ifconfig ${device:?} down up
    ;;
esac
```

2.3.3.5. irdaping

This is a program similar to **ping(8)**. It sends IrDA test frames (added some userdata which contains the frame number and the time the frame was sent). You can also change the size of the frame by using the **–s** option. You must supply an IrDA device address, and not an IP address. You have to be able to get that device address by some method **irdadump**?

Here is one output sample (pinging an ACTiSYS IR−100M):

```
dagbnb /home/dagb/linux/irda-utils/irdaping/ # ./irdaping 0xf7be8388
IrDA ping (0xf7be8388): 32 bytes
32 bytes from 0xf7be8388: irda_seq=0 time=102.466003 ms.
32 bytes from 0xf7be8388: irda_seq=1 time=102.202003 ms.
32 bytes from 0xf7be8388: irda_seq=2 time=102.170998 ms.
32 bytes from 0xf7be8388: irda_seq=3 time=101.633003 ms.

4 packets received by filter
```

2.3.3.6. irdadump

One advantage of implementing IrDA device drivers as network device drivers is that you should be able to attach sniffers to the device (or actually the packet type). That way, it is possible to use a really handy utility called **irdadump** (instead of **tcpdump**). This will make debugging MUCH easier. Linux−2.2 implements the BPF (Berkeley Packet Filter), so its possible to filter out exactly the frames you want to see.

Note: You probably have to be *root* for using **irdadump** . **CONFIG_PACKET** has to be enabled in the kernel. If compiled as a module you might load the module manually. **irdadump** has been converted into a library, so it can be used from GUI applications as well.

Here is a sample output of a small session between Linux and a Palm III. This log shows that the local irobex layer is not responding, so the Palm III sends a disc frame.

```
dagbnb /home/dagb/linux/irda-utils/irdadump/ # ./irdadump

20:18:15.305711 xid:cmd:saddr=0x05c589 > daddr=0xffffffff,S=6,s=0
20:18:15.385597 xid:cmd:saddr=0x05c589 > daddr=0xffffffff,S=6,s=1
20:18:15.465568 xid:cmd:saddr=0x05c589 > daddr=0xffffffff,S=6,s=2
20:18:15.545953 xid:cmd:saddr=0x05c589 > daddr=0xffffffff,S=6,s=3
20:18:15.625574 xid:cmd:saddr=0x05c589 > daddr=0xffffffff,S=6,s=4
20:18:15.705575 xid:cmd:saddr=0x05c589 > daddr=0xffffffff,S=6,s=5
20:18:15.785601 xid:cmd:saddr=0x05c589 > daddr=0xffffffff,S=6,s=255,info=Linux
20:18:18.075526 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=0
20:18:18.225498 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=1
20:18:18.375495 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=2
20:18:18.526355 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=3
20:18:18.675614 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=4
20:18:18.676364 xid:rsp:saddr=0x05c589 > daddr=0xb50c14b,S=6,s=4
20:18:18.765506 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=5
20:18:18.927221 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=255,info=Palm III
20:18:18.975796 snrm:cmd,ca=0xfe,pf=1
20:18:18.976534 ua:rsp,ca=0x58,pf=1
20:18:18.977145 ua:rsp,ca=0x58,pf=1
20:18:19.585627 rr:rsp,ca=0x58,nr=0,pf=1
20:18:19.585810 rr:rsp,ca=0x58,nr=0,pf=1
20:18:19.606413 i:cmd,ca=0x58,nr=0,ns=0,pf=1
20:18:19.606582 rr:rsp,ca=0x58,nr=1,pf=1
20:18:19.627708 rr:cmd,ca=0x58,nr=0,pf=1
20:18:19.627871 i:rsp,ca=0x58,nr=1,ns=0,pf=1
20:18:19.650571 disc:cmd,ca=0x58,pf=1
20:18:19.650736 ua:rsp,ca=0x58,pf=1
20:18:21.165524 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=0
20:18:21.315608 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=1
20:18:21.315793 xid:rsp:saddr=0x05c589 > daddr=0xb50c14b,S=6,s=1
20:18:21.395499 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=2
20:18:21.545516 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=3
20:18:21.695500 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=4
20:18:21.845840 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=5
20:18:22.007222 xid:cmd:saddr=0xb50c14b > daddr=0xffffffff,S=6,s=255,info=Palm III
20:18:22.056143 snrm:cmd,ca=0xfe,pf=1
20:18:22.056310 ua:rsp,ca=0xc8,pf=1
20:18:22.056381 ua:rsp,ca=0xc8,pf=1

37 packets received by filter
```

2.3.3.7. gnobex

GNOME GUI for using the IrOBEX protocol, to connect to Palm III.

2.3.3.8. irkbd

Support for infrared keyboard (and mouse?) protocol IrKBD.

2.3.3.9. irdalib

Linux/IrDA library.

2.3.3.10. obex

Please compile **irdalib** before compiling **obex**. Contains **irobex_test**, **irobex_receive** and **irobex_palm3** . And **obex_tcp** which makes it possible to use OBEX over TCP/IP.

2.3.3.11. findchip

findchip can be used to detect the type of FIR chip.

2.4. Configuration

2.4.1. General Configuration

- First you should put your IR devices in range. Though it might be possible that the Linux/IrDA service detects every new device automagically I only have good experience with the devices in range during the configuration process.
- Keep your infrared devices together in a range below one meter and an angle of 30 degree. There has to be a direct line of sight between them. If this is not possible, you may use a mirror (an unused M$ CD should work quite good).
- Add the following lines to your `/etc/conf.modules` file:

  ```
  # IrDA
  alias tty-ldisc-11 irtty
  alias char-major-161 ircomm-tty
  ```

  Note: The format of this entries has changed! Then do a **depmod −a** to update, and then **ircomm** should be automagically loaded when you need it. Here is what devices you need in `/dev`:

  ```
  dagbnb /usr/src/ > ll /dev/ir*
  crw-------   1 dagb     161,   0 Aug 25 20:13 /dev/ircomm0
  crw-rw-rw-   1 root     161,   1 Jun 18 13:44 /dev/ircomm1
  crw-rw-rw-   1 root     161,  16 Jun 18 13:44 /dev/irlpt0
  crw-rw-rw-   1 root     161,  17 Jun 18 13:44 /dev/irlpt1
  ```

- Have a look into the files in `/etc/irda`. They are similar to the files in `/etc/pcmcia`. Edit `/etc/irda/drivers` to reflect your setup. Most people will use **irattach** from that file. The files are:

  ```
  Makefile
  network*
  network.redhat*
  serial
  drivers
  network.opts
  obex
  printer
  ```

- Run **depmod −a**.

2.4.2. IrManager

Dag Brattli wrote: " *IrManager* [..].is a user−space daemon that is inspired and quite similar to the *cardmgr* used in the PCMCIA distribution.

The *IrManager* will receive events from the kernel level side of the protocol stack. When the *IrManager* receives an event it can execute shell commands and scripts, so I have added the `/etc/irda` directory which will contain such scripts. [..]

For example, if IrLMP discovers a remote device with IrLAN provider capabilities and no local IrLAN client has registered, then IrLMP will send an event to the IrManager and make it "modprobe" the module required.

[..]

When application level clients are ready for communication and user−space configuration, they can also notify IrManager about this, so that it can execute the right script. For example IrLAN will send the event EVENT_IRLAN_START when the data channel is ready for exchanging Ethernet frames. When IrManager receives this event, it will execute **/etc/irda/network start <devname>** to configure the network interface. This network script is actually the same as used by the PCMCIA code and since I'm using the Redhat variant of it, it will in turn execute **/sbin/ifup <devname>**.

So by using the IrManager, I "only" have to do this when I start the stack:

```
irattach /dev/ttyS2 38;
irmanager −d 1           # −d 1 means: start discovery process
```

and then when my laptop discovers the IrLAN provider (HP Netbeamer in my case) it will ask *IrManager* to load the module **irlan_client**. When the connection is up and ready, it will ask it to execute `/etc/irda/network start eth0`. When the connection is broken, it will again ask it to take down the interface using **/etc/irda/network stop eth0**.[..]

That's all to get it working if you are using Redhat. If you are using some other distribution which doesn't have `/sbin/ifup`, then you better copy `/etc/pcmcia/network.opts` to `/etc/irda/network.opts` or configure the file yourself.

If you want to use the IrLAN server, you will still have to **modprobe irlan_server** before you start the **irmanager** _without_ **−d 1**.

And just like the **cardmgr**, you will (if you want to) get the beeps when the connection is up and running and when it is disconnected!!!

I hope that we can add such scripts for all the other clients/services that need user level configuration. It would be really cool to have a `/etc/irda/printer` script for configuring IrDA(TM) capable printers. So if you get in range of an IrDA(TM) capable printer, then IrManager should load the **irlpt_client** module, and also configure the other stuff that needs to be done for using this printer.

I also hope that we can use the **config** file for configuring IrDA(TM) ports and device drivers. Something like:

```
Device Drivers
  module "irtty" script="irattach /dev/ttyS2"
  module "smc_ircc" irq=11 port=0x34f
```

So that IrManager can load and start all these when it is executed. In this way we would only have to start IrManager in `/etc/rc.d/init.d/irda` and the rest would be plug and play. There would be no need for manually starting programs and configuring devices. When **irmanager** receives the following events for a device <dev> it will currently do:

EVENT_IRLAN_START, start and configure the device using **/sbin/ifup <dev>**

EVENT_IRLAN_STOP, close the device using **/sbin/ifdown <dev>**

This can however be easily changed by the user, if this is not what is the prefered behaviour.

2.4.3. Low Level Drivers

There are three sorts of low level drivers: SIR, dongle and FIR. If the right driver is detected by the kernel you get a message like:

```
IrDA irda_device irda0 registered.
```

2.4.3.1. SIR

- Try to find out which serial port is used by the IR device. You may do so by watching the output of **dmesg**. If serial support is modularized do an **insmod serial** first. Look for an entry like:
```
Serial driver version 4.25 with no serial options enabled
ttyS00 at 0x03f8 (irq = 4) is a 16550A     #first serial port /dev/ttyS0
ttyS01 at 0x3000 (irq = 10) is a 16550A    #e.g. infrared port
ttyS02 at 0x0300 (irq = 3) is a 16550A     #e.g. PCMCIA modem port
```
If this is not the case, you either don't have infrared support enabled in the BIOS or the SIR mode of your infrared device is not detected by the kernel. Currently I know only two laptop models with this effect, the HP OmniBook 800 and the Toshiba Libretto models. I am not sure whether PnP support effects the detection of the IR port. If you are unsure try it out and let me know the results. Maybe you can use FIR mode if SIR doesn't work.
- In some situations you may have to use **setserial /dev/ttyS<0−2> port 0xNNNN irq M** to set the values for your infrared serial port, especially if the infrared port is a separate serial line. You usually don't need to change the values! For further information look into the FAQ section below.
- If you don't use **kerneld** or **kmod** insert the irda module with **modprobe irda**.
- Do **lsmod**. It should show the modules **irda** and **irtty** now.
- A look into `/var/log/messages` should show the entry "**Serial connection established**" now.
- Say **irmanager −d1**, which will start the necessary programs, such as **irattach**.
- Give **irattach** some time, e.g. seven seconds, to detect other IR devices. Then watch the output from the kernel that you will hopefully get in `/var/log/messages`. It should look like the following (I removed some lines, which were not related to Linux/IrDA):
```
Jan  2 12:57:26 japh kernel: ttyS00 at 0x03f8 (irq = 4) is a 16550A
Jan  2 12:57:26 japh kernel: ttyS02 at 0x03e8 (irq = 4) is a 16550A
Jan  2 12:57:26 japh kernel: Linux Support for the IrDA (tm) protocols (Dag Brattli)
Jan  2 12:59:09 japh syslog: executing: 'echo 1 > /proc/sys/net/irda/discovery'
Jan  2 12:59:09 japh syslog: Setting discovery to 1 exited with status 1
Jan  2 12:59:09 japh syslog: + 0.1 Fri Jul 25 11:45:26 1997 Dag Brattli
Jan  2 12:59:09 japh syslog: + 0.1 Fri Jul 25 11:45:26 1997 Dag Brattli
Jan  2 12:59:09 japh syslog: Serial connection established.
Jan  2 12:59:09 japh kernel: IrDA irda_device irda0 registered.
Jan  2 13:01:22 japh syslog: executing: './drivers start '
Jan  2 13:01:22 japh syslog: Serial connection established.
Jan  2 13:01:42 japh syslogd: Printing partial message
Jan  2 13:01:42 japh 0.1 Fri Jul 25 11:45:26 1997 Dag Brattli
Jan  2 13:02:49 japh kernel: IrDA Discovered: japh
Jan  2 13:02:49 japh kernel:     Services: Computer
```

* Even more information you can get with **cat /proc/net/irda/discovery** .

2.4.3.2. Dongle Connection − Infrared Adapters for the Serial Port

The currently supported dongles are the Extended Systems Inc. ESI−9680 JetEye, the Tekram IRmate 210B, the ACTiSYS IR220L and 220L+, the Greenwich GIrBIL. dongle.

Dag Brattli wrote (modified by wh): "To use dongles you have to do something like this:

```
modprobe tekram         # or esi or actisys
irmanager -d 1          #
irattach -d tekram      # or -d esi or -d actisys
```

As you can see, you must still use the **−d** option with **irattach** since it is possible to have two serial ports using different dongles at the same time (so the tty you are binding must know which dongle it is supposed to use). So if you have two dongles and two serial ports, you could do something like this:

```
modprobe tekram
modprobe esi
irattach /dev/ttyS0 -d esi 38;
irattach /dev/ttyS1 -d tekram 38;
```

PS: I would not try to turn the two dongles against each other, since I really don't know how the stack would react :−) ... Since I don't have any of these new ACTiSYS 220L+ dongles, I'm not able to test it. Since the new dongle has support for one extra speed (38400bps), you must specify the dongles differently with **irattach** so that the kernel knows which dongle you are using (and what QoS can be used):

```
irattach /dev/ttyS0 -d actisys     # for the 220L dongle
irattach /dev/ttyS0 -d actisys+    # for the 220L+ dongle
```

The current implementation of dongle support does not have any state associated with it, so its not possible to use both ACTiSYS dongles (220L and 220L+) at the same time (connected to two serial ports) for now. If someone needs to be able to do so, please mail me (Dag Brattli) and I will think about it!"

Note: When I tried to use an infrared modem (Swissmod 56Ki, manufactured by Telelink AG) connected to my laptop (IrDA works with Window$95 only, due to non standard hardware) I had to remove the infrared support in the BIOS to get it working!

Dag Brattli: "It is now possible to use **irport** instead of **irtty**! I have moved all the dongle stuff out of **irtty** and into **irda_device**, so it will also be possible to attach dongles to **irport**. Need however to make a small user−space utility **dongle_attach** that can be used to attach dongles to a specific driver instance. BTW. **irattach** is still working as before, and you will not notice the difference even when attaching dongles to **irtty** (I've just redirected the dongle ioctl to irda_device). Irport may be interesting since you avoid one software interrupt (bh) level, and it's also forced to work in half duplex mode so you don't get any echo if the irda port itself don't have echo−cancellation (girbil dongle and HP−4000 etc) ... To use it, you must supply the parameters to **insmod** like this: **insmod irport io=0x3f8 irq=4**, or whichever values you use. You can also add these parameters to `/etc/conf.modules` like this: **options irport io=0x3f8 irq=4**, but then you must remember to do a **depmod −a** and use **modprobe irport** instead of **insmod**."

Alvin Loh: "Anyone with a ESI 9680C can use both parallax's and ESI's signalling scheme, meaning they can use Parallax's driver with ESI9680C to work. "

2.4.3.3. Dongle Connection − Infrared Motherboard Adapter

Support for the ACTiSYS IR2000 dongle has been implemented in a file called `pc87108` which you can either compile into the kernel or **insmod/modprobe** to insert the module:

```
irmanager -d 1
modprobe pc87108
```

or insert **modprobe pc87108** into the `/etc/irda/drivers` file (I think).

From James <james@esc.cam.ac.uk> I have this description about setting up the hardware: There are two configurations, a five pin in line connector and a 6 pin DIL (at the end of a 18 pin DIL header). Basically any IrDA conpatible transceiver will work (I have a stack of old IRM3001 these are now obselete) you need to hook a capacitor (use a tantalum about ~1uF) between 5V and 0V near the transceiver and then connect everthing else up (RX−>RX, TX−>TX, 5V−>5V, and 0V−0V). If you don't like soldering irons, lots of companies do sell IR modules for the 5 pin connectors that fit into a hole in your case.

2.4.3.4. Fast InfraRed (FIR)

The IrDA(TM) standard knows *three* kinds of speeds:

1. SIR = Standard IrDA, up to 115kbps IrDA,
2. MIR = Medium Speed IrDA,
3. FIR = Fast IrDA (4Mbps),
4. VFIR = Very Fast IrDA(16Mbps), seems to become a future standard

Up to 115.200bps (SIR) many (probably all) infrared controllers work like a serial port and use a RZI (return to zero, inverted) modulation. Not every infrared controller supports 4Mps (FIR), up to 4Mbps they have to use 4PPM (4 pulse position) modulation technique. Currently there are two FIR chips supported: NationalSemiConductor NSC PC87108 e.g. used in IBM Thinkpad 560X and Winbond W83977AF (IR) FIR chip e.g. used in the Corel Netwinder PC. You may start the FIR service by loading the according module. Linux/IrDA will probe your hardware then. More drivers are under development.

So what speeds can you expect? Using SIR, you should be able to get about 10 Kbytes/s. Using FIR (4Mbps) you can get over 300 Kbytes/s (if you are lucky).

2.5. Specific Connections and IrDA − Protocols

2.5.1. Printer Connection − IrLPT, IrTTP, IrCOMM?

IrLPT support is under heavy construction at the moment. Maybe it will be replaced by IrCOMM. Please see mailing list archive.

- Edit `/etc/conf.modules`, as described above.
- Remove any current print jobs with **lprm "*"**.
- Run **depmod −a** .
- (If you don't use **kerneld** do a **modprobe irtty**. Probably obsolete.)
- (Do a **modprobe irlpt_client**. Probably obsolete.)
- (Check the modules with **lsmod**. This should show: **irda**, **irtty** and **irlpt_client**. Probably obsolete.)
- **su** to root, and do **mknod /dev/irlpt0 c 161 10**. Note: Something like **./MAKEDEV irlpt0** is not possible yet. But maybe **load_misc irlpt** works, though I couldn't test this yet.
- Try to write a small file to /dev/irlpt0 by **cat FILE >/dev/irlpt0** (do not wonder about a bad format this is just a first check). For me this didn't always work, but I couldn't find out why not.
- The better way is to change your `/etc/printcap` to use `/dev/irlpt0` in addition or instead of `/dev/lp1`. See *Printing−HOWTO* for detailed information.
- For easy printer setup you may use a printing software like APSFILTER, MagicFilter EZ−Magic (with RedHat there should also be a GUI for this purpose). Make a copy of /etc/printcap before.
- Example for APSFILTER with a HP 6P (non−postscript, HP 6MP is with postscript). The two relevant questions are:

  "Do you have a (s)serial or a (p)arallel printer interface?" Answer "p"

  "What's the device name for your parallel printer interface?" Answer "`/dev/irlpt0`"
- Restart the print daemon with **kill −HUP <PID of lpd>**. If you use another print daemon choose the according command.
- Watch whether the connection indicator of your printer shows activity, e.g. the green light above the IR port of a HP 6P/MP comes on (lower left hand corner, near the paper tray).
- I couldn't get to manage printjobs larger than approximately 10 pages yet. But maybe this depends on the memory size of my hardware, which is 16MB. There seems to be a problem with the software too, Thomas Davis wrote: "I will ... limit the irlpt, so it won't eat memory when you send a large print file.".

Takahide Higuchi reported: " I have been debugging IrCOMM with a printer (Canon BJC−80v) with IrDA port and IrCOMM protocol (not IrLPT). I can print a short e−mail text though, it easily causes dead lock when I try to print a postscript with gs."

From the page of Thomas Davis http://www.jps.net/tadavis/irda : To use the IrLPT server, you need to perform the following steps:

```
/sbin/insmod irlpt_server
/sbin/mknod /dev/irlptd c 10 `grep irlptd /proc/misc|cut −f 1`
```

At this point, the IrLPT server is ready to recieve print jobs; now; all you need is this simple shell script

```
#/bin/sh
#
while (true)
do
cat /dev/irlptd | lpr
done
```

Dag Brattli: I hope that this will make it easier for all you that prefer to live in user−space, to make your own IrDA applications and try it out. Some printers actually use IrTTP (because of the limitations of IrLPT), so now you can write your own small user−space printer client so you can talk to it:

```
int discover_devices(int fd)
{
    struct irda_device_list *list;
    unsigned char buf[sizeof(struct irda_device_list) +
                      sizeof(struct irda_device_info) * MAX_DEVICES];
    int len;
    int daddr;
    int i;

    len = sizeof(struct irda_device_list) +
          sizeof(struct irda_device_info) * MAX_DEVICES;
    list = (struct irda_device_list *) buf;

    if (getsockopt(sfd, SOL_IRLMP, IRLMP_ENUMDEVICES, buf, 38;len)) {
        perror("getsockopt");
        exit(-1);
    }
    if (len > 0) {
        /*
         * Just pick the first one, but we should really ask the
         * user
         */
        daddr = list->dev[0].daddr;

        printf("Discovered: (list len=%d)\n", list->len);

        for (i=0;i<list->len;i++) {
            printf("  name:  %s\n", list->dev[i].info);
            printf("  daddr: %08x\n", list->dev[i].daddr);
            printf("  saddr: %08x\n", list->dev[i].saddr);
            printf("\n");
        }
    }
    return daddr;
}

void client()
{
    struct sockaddr_irda peer;
    int addrlen = sizeof(struct sockaddr_irda);
    int daddr, actual;
    char buf[1024];
```

```
    fd = socket(AF_IRDA, SOCK_STREAM, 0);

    daddr = discover_devices(fd);

    peer.sir_family = AF_IRDA;
    strcpy(peer.sir_name, "P1284");
    peer.sir_addr = daddr;

    connect(fd, (struct sockaddr *) 38;daddr, sizeof(struct sockaddr_irda));

    /* Try to send something */
    actual = send(fd, "Testing", 8, 0);

    /* Try to read reply */
    actual = recv(fd, buf, 1024, 0);
}
```

2.5.2. LAN Connection − IrLAN

- You might connect your Linux box using IrLAN to another network device such as a Linux box with IrLAn, a HP NetBeamer or a Window$95 box with Infrared Network Device support.
- Dag Brattli wrote: "If you want to use IrLAN you must **modprobe irlan_client** before **ifup eth0**. I had to remove the request_module() stuff since that needed a process context which I don't have in the kernel. "

  Maybe you have to choose the access mode. You can do this by using **modprobe irlan access=1** for direct mode. IrLAN states that a provider can either be in *direct* mode, or in *peer* mode, so you currently have to choose when you start IrLAN.

- Run **ifconfig eth0 up <ip_address> netmask <ip_netmask>** to configure it with IP−address and other parameters. If the protocol is still running you may start communicating. It is possible to use RedHat's `netcfg` to do this, since it makes it very easy. Next time you only need to do **/sbin/ifup eth0**. Notice that **ifconfig** does not know how to deal with IrLAP addresses, so the address is really just the 4 first bytes (in little endian format).
- Test the network device by pinging to it. For detailed information about further setup see the *NET3−HOWTO*.
- Do not forget to add a route, e.g. **route add default gw <ip_gateway>** or **route add −host <target host> dev eth0**.
- Ping to another IP now, to test the connection.
- For testing reasons I recommend only to use one laptop and one IR ethernet device in the same room. If there are problems look which different modes for the IR ethernet device are possible. Try them.

For an ACTiSYS FIR board and dongle you may do:

```
irmanager -d1
/sbin/modprobe pc87108  # remove irattach from /etc/irda/drivers, or
                        # substitute irattach with the modprobe!
```

On machine 1:

```
modprobe irlan_client # not really necessary since irmanager should do this!
```

On machine 2 (if you don't have an access–point)

```
modprobe irlan_server
```

Do not compile **irlan_server** into the kernel, since it currently does not like that! You should have configured `/etc/sysconfig/network-scripts/ircfg-eth0` with a proper ad–hoc network if you are using two machines. If you have an access–point, then the normal setup should be fine.

Notice that in the latest patch (2.2.0–irda1) **irlan_client** will call the device **irlan0** by default, but you can change this by giving **eth=1** as an option to **irlan_client** (**modprobe irlan_client eth=1** or **options irlan_client eth=1** in `/etc/conf.modules`). The next release of IrLAN will be only one module, so you don't need to think about if you need to have the client and/or the server installed.

It's possible to do **ifconfig irlan0 –broadcast** to stop the AP from flooding you with broadcast frames! That can be a problem if you are connected to a very large Ethernet segment. The only problem is that your machine will then have to initiate all communications and can therefore not function as a server (well, you could probably make a stationary machine somewhere answer ARP requestes on your behalf).

When using the IrLAN software from HP, you maybe have to name your computer "HP NetbeamIR" in the IAS entry to make it connect. Also I have heard rumors that a "Psion 3c" also wanted to connect only to another IR device if it had the same or a similar name.

2.5.3. HP NetBeamer Connection

From Renaud Waldura: All the IrDA stuff is compiled in as modules.

- edited `/etc/irda/drivers` to include **irattach /dev/ttyS0** (or whatever your "serial" IrDA port is, WH). Also `/etc/conf.modules` contains **alias tty–ldisc–11 irtty**.
- **irmanager –d 1** then **echo 3 >> /proc/sys/net/irda/debug** to see what's going on.
- **modprobe irlan**
- I had to **echo 9 >> /proc/sys/net/irda/slot_timeout** (use 90 for newer kernels!) in order to have the NetbeamIR recognized. Otherwise I was only getting a bunch of "media busy" messages, and no actual discovery of the NetbeamIR. 9 is the smallest value that worked for me.
- renamed `/etc/irda/network.orig` to `/etc/irda/network` and edited `/etc/irda/network.opts` for my IP config. Also copied `/etc/pcmcia/shared` to `/etc/irda` (this file is apparently missing from the distribution).

- I also had to comment out **grep_stab $1 < /var/run/stab** (line 131) in `/etc/irda/shared`. For some reason it fails, spitting a "usage" message.
- moved laptop in range, the NetbeamIR is discovered, irlan0 created and config'ed.
- transferred data at 7 kb/s, both ways: **ping**, **ftp**, **telnet**. Yaho!

2.5.4. Palm III Connection − IrOBEX

The IrOBEX stuff seems under rapidly improving changing development. So the applications change, too. Therefore I just can't give quite exact information. Please see also the report by Dag Brattli at http://www.cdpubs.com/hhsys/archives/66/10brattl.pdf .

Please note: IrOBEX now is in user−space and uses IrDA sockets. Remember that when using sockets, you don't need to have irobex enabled in the kernel, and `/dev/irobex` is not used anymore!The `/etc/irda` script is really only good for configuration of the devices, making the right **mknod** for `/dev/irobex` etc., not for starting applications.

- Palm III −> Linux

  1) Terminal 1> **irattach /dev/ttyS<x>**

  2) Terminal 2> **load_misc irobex**

  3) Terminal 3> Start **irobex_app** in the irobex directory. I suppose **irobex_app** is not working anymore. Now you should use the **gtk/irobex** program instead! You need to have the gtk library installed to use this program. A command line frontend should be programmed by someone. Maybe the programm to use is **irobex_receive**.

  4) Beam something from your Palm III.

  5) If everything is successful, you can take a look at a new file that has been created in the directory in which you started irobex_app (or in /tmp for irobex_receive). This file will be named after the object you just transfered.
- Linux −> Palm III

  This should also be possible, but I don't have any further information right now.
- PPP

  Rui Oliveira wrote: "This is just to let you know that with the latest IrCOMM patch (050998) of Takahide Higuchi, I managed to **HotSync** and establish a PPP connection between my Palm III and my Linux box. I'm using IRLink (from IsComplete) to redirect the serial port to ir. Communication with pilot−xfer (probably at http://www.slac.com/pilone/kpilot_home/mainpage.html) works flawlessly. Although I was able to establish a PPP connection, I'm still unable to fetch mail and do Web browsing. This is probably due to connection time−outs. I am checking this out. Please see the *PPP−HOWTO* for further information about PPP.

... I managed to establish an apparently robust connection between my Linux box and a Palm III. The **pppd** invocation I use is as follows:

/usr/sbin/pppd /dev/irnine 57600 192.168.2.10:192.168.2.11 proxyarp passive silent persist noauth local nodetach. Over the PPP connection I used **ping**, **ssh**, and **http**.

Strange is however the fact that discovery must be enabled (**irmanager −d 1**). Otherwise, even with an active IrCOMM connection, the link goes down due to a IrLAP disconnect.

The pilot−link tools (used for Linux/Palm synchronization) also ran flawlessly over IrCOMM via `/dev/irnine` or `/dev/ircomm`."

There are also reports about **kpilot**, though not working as flawlessly as **pilot−xfer**.

- IrCOMM

  Jon Howell wrote: "I thought I'd try IrCOMM, since the Palm III can be made to reroute serial info to the IR port (using IrLink from IS/Complete, available at www.palmcentral.com), and then you can run a terminal program (like *PalmTelnet* in serial mode) over IrDA. I can only assume it's using the IrCOMM protocol. I've tested this configuration between two palm pilots, but of course I can't know what the protocol running over the IR is."

  (1) Start **HotSync** on your Palm. You need the IrDA upgrade for the Palm to have IrCOMM support http://www.palm.com/custsupp/downloads/irenhanc.html .

  (2) Place the Palm in front of the dongle.

  (3) Start **pilot−xfer −p /dev/irnine −s <sync−dir>** .

  And if you are lucky it will start syncing. If you start **pilot−xfer** before you start **HotSync** on the pilot, you will _not_ be lucky!

  Maybe a terminal program like *PalmTerm* is also useful.

Wessel de Roode wrote: The Palmpilot is default locked on 57k. You can however if you write your own software for the Pilot, use the 115k line settings. I quote a part from the **irlib.h**:

```
---------- irlib.h from the SDK 3.0 from palmpilot -----
// Options values for IrOpen
#define irOpenOptBackground     0x80000000              // Unsupported background task use
#define irOpenOptSpeed115200    0x0000003F              // sets max negotiated baud rate
#define irOpenOptSpeed57600     0x0000001F              // default is 57600
#define irOpenOptSpeed9600      0x00000003
-------------- end quote --------------------------------
```

Peter Pregler reported: If the Palm enters the range of the irda−device a popup appears with the text "Transmission: waiting for sender"

Ron Choy answered: There is a software called *ShutupIR* that is supposed to help with this problem of annoying popup http://hp.vector.co.jp/authors/VA005810/irda/shutup10.zip I haven't tried it but it looks like it would fix your problem.

2.5.5. Palm III Connection to IBM Thinkpad 600

This chapter is a courtesy of Harald Milz − SuSE. Minor changes by Werner Heuser to fit into the HOWTO and some changes in Linux−IrDA.

This document describes how to set up Linux/IrDA on an IBM Thinkpad 600. This works for my machine. Your mileage may vary with other machines.

2.5.5.1. Prerequisites

- Reboot to the preinstalled M$ OS and activate the IR port using the Thinkpad tools. There is currently no Linux tool to achieve that. *This will disable your internal serial port (ttyS0)!*
- Reboot to Linux and configure a kernel with support for IrDA and IrCOMM. The Linux−IR−HOWTO talks about IrOBEX as well, but this is just for beaming objects across. This doesn't work with **hotsync** as far as I know (didn't try yet).

2.5.5.2. Adding support for IrDA

- Add the following lines to **/etc/conf.modules**:

```
# IrDA
alias tty-ldisc-11 irtty
alias char-major-161 ircomm-tty
```

- Get the current IR Tools. I use version 0.9.2 at this time. Compile them proper. I installed **irmanager** and **irattach** in **/usr/local/sbin**.
- I modified `/etc/irda/drivers` and set up this `/etc/rc.d/irda` start/stop script. Don't forget to add the line **START_IRDA=yes** to **/etc/rc.config** and to add a symlink like **/etc/rc.d/rd3.d/S99irda**.
- Add a device file for the redirected IR device: **mknod /dev/irnine c 161 0** , see also `/proc/tty/drivers`.
- Add a symlink **/dev/pilot −−> /dev/irnine** to make all programs happy (if you use the shell variable **PILOTPORT=/dev/pilot**).

2.5.5.3. Starting IrDA − First Steps

Call **/etc/rc.d/irda start** and watch the console messages. **ps** should show **irattach** and **irmanager** running. Check **/proc/net/irda** to see what is there. **ircomm** will only be available and show reasonable stuff when an application accesses the irnine port (and loads **ircomm−tty** through **kmod**).

Start **PilotManager** or **sync−plan** and be happy :−) IR should run fine at 57600 bps.

2.5.5.4. Configuration Files

`/etc/irda/drivers`:

```
#! /bin/sh
#
# drivers
#
# Initialize and shutdown IrDA device drivers.
#
# This script should be invoked with two arguments.  The first is the
# action to be taken, either "start", "stop", or "restart".
#

action=$1
device=$2

case "${action:?}" in
start)
        /usr/local/sbin/irattach /dev/ttyS0          # The third serial port is an IrDA port
        # irattach /dev/ttyS0 -d esi # Attach a ESI dongle to the first serial port
        # irattach /dev/ttyS0 -d tekram
        # insmod pc87108             # If your machine as a pc87108 FIR chipset
        # modprobe uircc             # Sharp UIRCC chipset
    ;;
stop)
        killall irattach             # ... or something. Currently not used
    ;;
restart)
        /sbin/ifconfig ${device:?} down up
    ;;
esac
```

`/etc/rc.d/irda`

```
#! /bin/sh
# Copyright (c) 1995-1998 SuSE GmbH Nuernberg, Germany.
#
# Author: hm
#
# /sbin/init.d/irda
#
#   and symbolic its link
#
```

```
# /sbin/rc<skeleton>
#

. /etc/rc.config

# Determine the base and follow a runlevel link name.
base=${0##*/}
link=${base#*[SK][0-9][0-9]}

# Force execution if not called by a runlevel directory.
test $link = $base 38;38; START_IRDA=yes
test "$START_IRDA" = yes || exit 0

# The echo return value for success (defined in /etc/rc.config).
return=$rc_done
case "$1" in
    start)
        echo -n "Starting service irda"
        ## Start daemon with startproc(8). If this fails
        ## the echo return value is set appropriate.

        startproc /usr/local/sbin/irattach /dev/ttyS0 || return=$rc_failed
        startproc /usr/local/sbin/irmanager -s1 -d0 || return=$rc_failed

        echo -e "$return"
        ;;
    stop)
        echo -n "Shutting down service irda"
        ## Stop daemon with killproc(8) and if this fails
        ## set echo the echo return value.

        killproc -TERM /usr/local/sbin/irmanager || return=$rc_failed
        killproc -TERM /usr/local/sbin/irattach || return=$rc_failed
        for i in ircomm-tty ircomm irtty irda ; do
                rmmod $i;
        done

        echo -e "$return"
        ;;
    restart)
        ## If first returns OK call the second, if first or
        ## second command fails, set echo return value.
        $0 stop  38;38;  $0 start  ||  return=$rc_failed
        ;;
    reload)
        ## Choose ONE of the following two cases:

        ## First possibility: A few services accepts a signal
        ## to reread the (changed) configuration.

        #echo -n "Reload service foo"
        #killproc -HUP /usr/sbin/foo || return=$rc_failed
        #echo -e "$return"

        ## Exclusive possibility: Some services must be stopped
        ## and started to force a new load of the configuration.

        #$0 stop  38;38;  $0 start  ||  return=$rc_failed
        ;;
    status)
        echo -n "Checking for service foo: "
        ## Check status with checkproc(8), if process is running
```

```
        ## checkproc will return with exit status 0.

        #checkproc /usr/sbin/foo 38;38; echo OK || echo No process
        ;;
    probe)
        ## Optional: Probe for the necessity of a reload,
        ## give out the argument which is required for a reload.

        #test /etc/foo.conf -nt /var/run/foo.pid 38;38; echo reload
        ;;
    *)
        echo "Usage: $0 {start|stop|status|restart|reload[|probe]}"
        exit 1
        ;;
esac

# Inform the caller not only verbosely and set an exit status.
test "$return" = "$rc_done" || exit 1
exit 0
```

2.5.6. Psion 5 Connection

Andrew Chadwick <andrew.chadwick@symbian.com> wrote: A nifty way to check that the baud rates for SIR are set up properly (if you have a Psion Series 5) is to point the S5 at your Linux box's IR window and try to beam a file. While the beamer dialog's on the screen, the S5 will try to make an IrDA connection (even when it claims it can't find another IR machine). You should be able to do a **cat < /dev/ttyS3** and if the serial parameters are right on both machines, you should see the words "Symbian EPOC" (machine ident) scroll past amidst the spew.

Fons Botman wrote: " Maybe someone with a Psion 5 would like to test this program. It emulates the protocol for the Psion 5 IR send and receive command for files on linux. You can now exchange files with simple commands. The transfer rate is 9.7 KBytes/sec on a 115KB SIR link for big files which is not bad methinks. It is beta, so be sure to backup the Psion first, I did get a soft reset once (no data loss). ;−)" I have put the source into Appendix C.

2.5.7. Cellular Phone Connection

As far as I know some cellular phones use the IrCOMM standard, e.g. Ericsson SH888 and NOKIA 6110 (I'm not sure about the NOKIA 8110). Maybe other cellular phones use the IrOBEX standard (see the Palm III section for information about setting up a connection) or IrMC.

2.5.7.1. Ericsson

2.5.7.1.1. Configuration

To start a communication session with `/dev/irnine` (`/dev/ircomm`), for instance, say:

```
dip -t
> port irnine
> term
```

Probably you may use **cu** or **xc** instead of **dip**, too (**cu −l /dev/irnine** or **xc −l /dev/irnine**). There are also reports about some efforts with the Ericsson GF768 and IR Modem DI 27.

Benny Amorsen wrote: The SH888 emulates an IRDA−port when you connect it using the serial cable. Why someone would think up something weird like that is beyond me, but that is the way you get it to work in Windows. Not that I ever managed to make it work in Windows, though.

Ales Dryak has send this survey (looks like a Debian/GNU Linux distribution, please modify your configuration accordingly). Mobile Ericsson SH888 (**ati1 = 980408 1035 PRGCXC125101**):

1. **mknod /dev/ircomm c 60 64**
2. `/etc/conf.modules`:

   ```
   alias tty−ldisc−11 irtty
   alias char−major−160 ircomm−tty
   ```
3. `/etc/irda/drivers`: **irattach /dev/ttyS0** # (IrDA port in SIR mode)
4. running **irmanager**
5. `/etc/chatscripts/sh888`

   ```
   <ABORT stuff>
   "" \d\d\d\d\d\dATZE0
   OK ATD<phone number to call)
   CONNECT \d\c
   ```
6. `/etc/ppp/peers/sh888`

   ```
   noauth
   connect "/usr/sbin/chat −v −f /etc/chatscripts/sh888"
   /dev/ircomm
   115200
   defaultroute
   noipdefault
   user <your username> # don't forget to add your password to chap secrets or chat script
   ```

A few seconds (app. 30) after executing **pppd call sh888** I get connected to our Intranet/Internet having full IP connectivity (telnet, ftp, www, icmp tested). Futhermore I can connect to `/dev/ircomm` using **minicom** and play with AT command. Great! And looks stable!

2.5.7.1.2. SH888 Phonebook Tool

Gerhard Gonter reported: Several members of the list are successfully using the Ericsson mobile phone SH888 with the Linux−IrDA software, usually to use it as a modem. The software is also quite useful to access other parts of the phone using AT commands. The built−in phonebook is an interesting target.

After A quick research on the Internet (FreshMeat, Deja, YAHOO), I did not find any phonebook tool for Linux (or another Unix). To solve that problem, I wrote a small Perl script and a related module. Since this now works acceptably well for me, I decided to wrap that up and release it at this early stage of development.

The tarball can be retrieved from http://falbala.wu–wien.ac.at:8684/pub/english.cgi/0/172903 as http://falbala.wu–wien.ac.at:8684/pub/english.cgi/d172914/sh888–0.01.tar.gz

In the mailing list gsmlib was also recommended, though ... *there was no way for me to use this over infrared, no connection with my sh888*. Florian Lohoff reported: *Works (kind of) with the S25. I needed to change a ifdef as it seems the S25 does not respond with CR LF ...But setting a link from /dev/mobilephone –> /dev/ircomm lets me send SMS via the S25 without a problem. Phonebook backup does NOT work because the S25 does some silly responses to probably empty phonebooks.*

The specifications for SMS messages and phone books can be downloaded free (of charge, not FSF free ;–) from ETSI. Search for *GSM 07.07* (you might also want *GSM 07.05*). You have to register before downloading it. The standards are in Acrobat PDF format. The S25 supported commands are available on the Siemens websites as a PDF for free.

A survey of the AT commands for the SH888 is at http://mobileinternet.ericsson.se/emi_download/sh888/888_R1D.pdf

2.5.7.2. NOKIA

Carlos Vidal wrote: Correct me if I'm wrong, but it seems to me that Nokia telephones do not contain a genuine hardware modem, but something which is similar in principle to WinModems for PC. Whenever Nokia writes about modem communication, they use the name "Windows software modem" (or something similar). Which is actually backed up by the need to use special Nokia software for Windows (called Nokia Cellular Data Suite).

Joonas Lehtinen wrote: This is true with 61xx models. Models: 8810, 9000(i) and 9110 should work fine. (They have inbuilt modem). My Nokia 9000 reports IrCOMM with linux.

Some suggestion by Carlos Vidal carlos@tarkus.se : "I'm doing some tests trying to see how far can I get with my Nokia 6110 on Linux. I've just compiled gnokii–0.2.4 (gnokii is Nokia mobile phones connected via serial cable support for Linux and *BSD http://multivac.fatburen.org/gnokii/ , WH), but it doesn't work. As I have *Nokia Data Suite* I did the following connection:

Nokia 6110 <–– Nokia Cable ––> PC/Linux <–– Null–modem cable ––> PC/W95

In the PC/Linux I run the program **snooper** (by Jun–ichiro itojun Itoh <itojun@mt.cs.keio.ac.jp>, sorry couldn't find an URL maybe some other sniffer will do it also, e.g. **sniffit**, see also appendiy about serial sniffers, WH) with small modifications in order to configure the serial port correctly.

Normally, if **snooper** has the correct baud rate, the phone and the PC/W95 should communicate as if there was no snooper in between. This worked pretty well when I cracked the protocol of my Minolta camera. The problem here is that the phone doesn't answer or hangs after a while.

It seems that the timing is quite important during the initial phase of the communication. The log I obtain is:

```
0>1: UUUUUUUUUUUUUUUUUUUUUUUUU
line 0: LE *DTR *RTS ST SR CTS CD RI *DSR
line 1: LE *DTR *RTS ST SR CTS CD RI *DSR
```

```
0>1: UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUU\x1e\x00\x0c\x02\x00\x09\x00\x01\x00\x0d\x00\x00\x02\x01@\x00P\x
06
1>0: \x18\x00\x00\x00\xfc\x18\x00\x00\x00\x00\x00\x00\xc0\xf0
0>1: \x1e\x00\x0c\x02\x00\x09\x00\x01\x00\x0d\x00\x00\x02\x01@\x00P\x06
1>0: \x18\x00\x00\x00\x18\x00\x00\xc0\xf0\x18\x00\x00\x00\x00\x00\x00\xc0\xf0
0>1: \x1e\x00\x0cd\x00\x06\x00\x01\x00\x10\x01`\x13\x13
1>0: \x18\x00\x00\xf0\x00\x00\xfc
0>1: \x1e\x00\x0cd\x00\x06\x00\x01\x00\x10\x01`\x13\x13
```

0 is the PC/95 and 1 is the phone. The communication starts with a stream of 'U' (0x55) and with DSR/DTR on. The phone answers with '0x18 0x00 ...'. This dialog continues for a while as if both were deaf and finally the phone crashes and the only way to reset it is to remove the batteries!

I guess that what happens is that the phone is trying to find the correct baud−rate and fails because of the delays introduced by **snooper**. This probably has to do with some IrDA protocol used with also with the infrared connection."

Wessel de Roode <J.W.deRoode@student.utwente.nl> "I manged to get the Discovery IR hint bits (with my Palm Pilot):

```
Discover:
0:xxxxxxxx:81.01
        01  IR_HINT_PNP     01  IR_HINT_TELEPHONY (IrMC ?)
        80  IR_HINT_EXT

Device info query:
\006Device\012DeviceName
        4e 6f 6b 69 61 20 36 31 30 30           Nokia 6100
```

I also managed to query the PNP device of the Nokia. It has one PNP device. It's PNPC100 which equalt a 9600 baud modem. I deleted the query, if somene can send me a hint to restore it. was somthing like **IrDA:<dunno>:PNP:Comp#01** The same query with **IrDA:<dunno>:PNP:CompCnt** gives the number of PNP−devices are available in the Nokia. Which is here only one."

Maybe it is necessary to load the **irlpt_server** module for connections to a NOKIA.

There are also reports about gsmlib for sending and receiving sms messages, updating address books etc). These functions are working, except for minor charset problems.

2.5.7.3. Siemens S25

2.5.7.3.1. Configuration

By Florian Lohoff <flo@rfc822.org>: "Do it step by step − Get your irda working **irmanager −d 1**, **irattach /dev/ttySx** etc. Then have a look at the `/proc/net/irda/discovery` whether you find something like this:

```
(flo@paradigm)~# cat /proc/net/irda/discovery
IrLMP: Discovery log:
nickname: SIEMENS S25, hint: 0x9024, saddr: 0x4286ce23, daddr: 0x04295741
```

Now load **ircomm** and **ircomm−tty** and normally(tm) you should be able to connect to the correct `/dev/ircomm` and you can easily dial and load/backup the phonebook etc ...", e.g. with **minicom**.

Timo Felbinger describes the connection between a Toshiba and a Siemens S25:

- kernel 2.2.12 and patch−2.2.12−irda3, IrDA support in the kernel, ircomm and ircomm−tty as a module
- mknod /dev/irnine c 161 0
- **insmod ircomm insmod ircomm−tty**
- start **irmanager −d 1** with **insmod toshoboe** in the **start** section of `/etc/irda/drivers` . Note: don't load **toshoboe** before the **irmanager**, this may cause *device or resource busy*
- after **dip −t** and the command **port irnine** the S25 shows a connection. Note: the IR port of the S25 has to be activated of course, the distance between the two devices seems not critical.
- After **term** the S25 behaves like the usual Hayes modem and can be used with the AT commands.
- dial−out with **pppd** works out of the box.

2.5.7.3.2. German e−plus

A note to German e−plus users:

Every e−plus contract (except Free&Easy) contains a PPP conncetion to the WWW (no separate registration necessary). This service is available around Germany under the phone number 123100. This worked also out of the box.

pppd configuration:

```
/dev/irnine
defaultroute
netmask 255.255.255.0
debug
mtu 552
crtscts
```

```
noauth
connect '/usr/sbin/chat −v −f /etc/ppp/eplus.chat'
```

chat script `/etc/ppp/eplus.chat`:

```
ABORT "BUSY"
ABORT "ERROR"
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
ABORT "NO ANSWER"
SAY "initializing modem..."
"" "AT"
"OK" "ATZ"
SAY "dialing..."
"OK" "ATDT123100"
SAY "waiting for CONNECT..."
"CONNECT" ""
SAY "connected!"
SAY ""
```

For the nameserver you may use 192.76.144.66 . Username or password are not necessary.

2.5.8. Digital Camera Connection

Markus Schill wrote: "Great that there are also other people who are interested in using the SONY DSC−F1 IR adapter under linux. Up to now I have only toyed around with the linux−irda software and the serial IR adapter from PuMa Technologies that came with the camera. This is the status. I am using linux 2.0.33 and the latest linux−irda... If I use:

```
insmod irda
insmod irtty
irattach /dev/cua0
```

the adapter starts talking to the camera. /var/log/messages says that SONY−DSC−F1 was found, but no service is started. (Please note, this probably doesn't apply to the 2.2.x kernel versions of Linux/IrDA, wh).

There are two programs for linux available that can be used for the communication with the camera via cable: (1) **chotplay** and (2) **stillgrab**. They both take a tty as commandline option, so I guess that they should work if the irtty layer of the protocol stack works correctly ... I have not looked at anything in the linux−irda code, yet!). I am not sure whether I understand the stack but shouldn't the irtty make the thing look like a normal tty? What service should be started. "

Dag Brattli wrote: "I'm not sure which application level protocol the camera uses, but it is possible that it implements the IrDA(TM) Infrared Transfer Picture Specification (IrTran−P). If you take a look at http://www.irda.org/standards/pubs/IrTran−P_10.pdf, you will see that it is a protocol which is implemented above IrCOMM (not IrTTY!). IrTTY is something we use just to be able to talk to the Linux serial driver. "

2.5.9. Window$9x/NT and Linux/IrDA

2.5.9.1. Introduction

Why this? Unfortunately Linux users are not always supported with the necessary hardware information. Sometimes it is possible to look at this informations in Window$95. Sometimes its even useful to connect the two. Linux could also provide occasional access point services to a Window95 laptop of a friend dropping by.

Where to get it from? At http://www.microsoft.com in the directory /Windows95/downloads/contents/WURecommended/S_WUCommunications/W95IrDA/ you will find a support pack *Infrared Transfer 2.0*. It is a self−extracting archive **W95IR.EXE** with 331KB. Note: Microsoft seems to change the location of this file (and others) at random, the former URL is Microsoft Windows95 IrDA − Old

Microsoft(tm) has *three* versions of IrDA support for Windows95. The version number can be found in the "Software" icon in the Control Panel and the file `infrared.inf`.

Version 1.0 is still delivered with some hardware.

Version 2.0 is the version they currently offer at their web site. It is in the self−extracting file W95IR.EXE. The last time I looked (1999−02−21) it was 434KB and was found at http://support.microsoft.com/download/support/mslfiles/W95IR.EXE . Their website is frequently changing, so do not be surprised to find the file (also) in another location or not at all.

Version 3.0 can/could be found in their downloadable Infrared development kit IRDDK30, but is mostly useful for developers. It is internally different from 2.0, it is based on "miniport" network drivers, just like the Linux version. It exists for some time and has some support for NT, but it clearly did not make it into the mainstream NT4.0 distributions. For 95 you are probably better off with 2.0. The choice may depend on the documentation of the drivers you get with your specific hardware.

MS website also used to contain a nice utility **IrXfer**, contained in the archive **IRXFER.EXE**, This is the *Infrared Transfer* utility, which uses an IrOBEX variant I think, it is referenced in the IrOBEX protocol description. The utility was freely downloadable, but I could not find it the last time. It is a nice graphical utility which can be used to transfer files over IrDA between computers.

With some machines, e.g. a HP Omnibook 800 it is necessary to use a vendor specific version of this package (for the HP Omnibook 800 you may find it on the recovery CD).

Especially the **..\windows\inf*.inf** files and the device manager are of interest to look for configuration details.

As far as I know Window$NT doesn't support IrDA(TM). About Window$98 I have heard there is no IrDA(TM) support yet. Countersys on http://www.countersys.com claims to sell an IrDA solution for NT4.0 to support their JetBeam product, Microsoft refers to them for it.

AFAIK:

- Windows95 : use 2.0

* Windows98 : delivered with 3.0 and IrXfer (works with Linux/IrDA, IrOBEX?)
* WindowsNT4.0: no IrDA
* Windows2000 : 3.0(+?) http://www.microsoft.com/hwdev/infrared/

There are also some non M$ products available. Note: Some of them use proprietary infrared protocols:

* CounterPoint: QuickBeam 1.15 (works with Linux/IrDA, IrOBEX?)
* LapLink 7.5
* CarbonCopy 32 4.0
* pc ANYWHERE 7.5
* Puma Technology: TRANXIT pro 4.0

2.5.9.2. Connection between Linux/IrDA and Window$95 IrDA(TM)

I suppose there are *four* ways to connect Linux/IrDA and Window$95:

1) A *network connection* between two PC's. If you have set up *Infrared Transfer 2.0*, you will find an IrDA(TM) network device in the <*Network Device Section*>. But I couldn't get a working connection yet.

Some information by Fons Botman: MS does not support PEER mode, which is strange, because it would fit their *workgroup* model. I have had success with IrLAN DIRECT mode when I removed the **COMPUTER** entry in the hints bits from the Linux side. Windows95 will recognise the Linux box as a lan access point and will try to install a driver for it but cannot find it. At this point I install the *network adapter* driver named *Microsoft IrDA Lan Driver* (you can also do this beforehand). You now should have a new active network driver which can be seen with **WINIPCFG**. To make it work with TCPIP I give the linux side an address using the redhat configuration tools and start a dhcp server on the linux box which hands out leases on the IrDA subnet. Sometimes I have to use **WINIPCFG** to explicitly ask for a lease. After this you can freely network between the two computers. Try to telnet or use explorer from Windows95 to linux.

Open points:

a. write a proper `/etc/sysconfig/network/irlan-up` script.

b. solve the direct/peer mode problem and the peer mode dhcp problem (zero, one, too many dhcp servers on the irda sublan), maybe borrow leases from a dhcpserver on the eth0 lan (dhcrelay).

c. Make the *Microsoft IrDA Lan Driver* install automatically, maybe this is a problem in the PnP configuration. This seemed to work better on Windows98.

d. I sometimes get a TCP configuration problem after disconnect, which can be solved by a Windows95 reboot. Maybe tune the dhcp configuration.

2) Maybe it is also possible to use the *IrOBEX protocol*. But I don't know which software to use and where to get it. I supposed the necessary software comes with a Palm III, but this seems not to be true.

3) Takahide Higuchi <thiguchi@pluto.dti.ne.jp> provided *IrCOMM support*. From his page at http://www.pluto.dti.ne.jp/~thiguchi/irda/ I have taken the following description (I have modified it at little): "With IrCOMM support you can send or receive short messages between a linux box and a terminal program on a win95 laptop! Please add this line to /etc/conf.modules:

```
alias char-major-160 ircomm-tty
```

Next, make a device file **mknod /dev/irnine c 161 0**. Now Linux/IrDA services can be started as usual with **irattach /dev/ttyS? &**. `/dev/irnine` can be used as a serial device. **ircomm** and **ircomm-tty** modules will be loaded automatically by **kerneld/kmod** when a program uses `/dev/irnine`. NOTE: I think "setserial" utility will not work on `/dev/irnine`. Tips:

- To accept login via IrCOMM, use this as a root: First, please enable IrDA and IrCOMM. Then edit `/etc/inittab` and add a line like this:
  ```
  T1:23:respawn:/sbin/getty -L -w irnine 38400 vt100
  ```
  and do this as a root: **init q**. And **init** will start waiting for incoming IrCOMM connection. You will see your favorite Linux's login prompt from terminal emulator on Win95!
- If you try **pppd**, please consider the −crtscts option to disable flow−control. I implemented some flow−control emulation but it is not tested.
- Now my patch reports what kind of features is needed by the peer infrared device. Messages like this will be written in syslog:
  ```
  Sep 4 10:01:02 monolith kernel: parse_control:instruction(0x12)
  Sep 4 10:01:02 monolith kernel: data:03
  ```
- I especially want to know what message SH888 (or other infrared devices except for win95 PC) says. So please mail me your syslog generated during IrCOMM connection! If you have a copy of the IrCOMM specification written by IrDA(TM), please see page 34 or 38, and you will know what these messages mean."

4) From Fons Botman: IrLPT works fine. Make Linux the IrLPT server, and in Windows95 configure a printer to use the virtual LPT port defined when you installed MS−IrDA. Windows95 periodically sends some short sequence to the printer, I still have to figure out what that is.

2.5.10. Connecting from Linux to WinCE

Submitted by Arthur Tyde and Bryan Abshier of Linuxcare Inc.

This will tell you how to set up a masqueraded PPP connection via. IrDA from WinCE to a Linux based notebook computer. Once you are IP connected, the rest is up to you. We put this together as a guide for Sony notebook users with Casio E−100/105 PDA's, though the procedure should work for any WinCE 2.11 device with infrared capabilities talking to any notebook. Do all the Linux side testing signed on as root, standard warnings apply.

2.5.10.1. Configure WinCE

Configure a network connection for your WinCE device. Go into "Connections" and create a "Direct Connection" Name it something meaningful, for device select "Infrared Port". Go into settings and change the baud rate to 115200, this is the max for WinCE. Go to TCP/IP settings and check "Use server−assigned IP address," and "Use software compression," and "Use IP header compression" Make sure "Use Slip," is unchecked. For Name Servers, make sure "Use server−assigned addresses" is checked. Go to Start, Settings, Communications, Identification and enter something for the Device Name. (I used "cetoy") You most likely already have these values set if you have synced with a Win9x desktop using *Activesynch*.

2.5.10.2. Configure Linux/IrDA

Set up IrDA support on your notebook (described elsewhere) and get to the point where your notebook will discover an IrDA compliant device. A good sign is the irda0 device will show up when you execute **ifconfig**. It will not have an IP address, this is ok.

2.5.10.3. Setup the Connection

Test the discovery by setting an IrDA device in range of your IR port, wait 5 seconds, and;

cat /proc/net/irda/discovery

For example, the Ericsson I888 World Phone with IR port enabled should immediately show something like this;

```
"name:I 888 WORLD   ,hint:0x9104,saddr:0x838470e5,daddr:0x152dceaa"
```

Your WinCE machine will not be discovered unless it's actively looking for a connection. So, if you want to test with WinCE position your device and double tap on the network icon you created in step 2, you should see something like this:

```
"name:mytoy,hint:0x8204,saddr:0x838470e5,daddr:0x00000b72"
```

The name displayed will be whatever value you have entered into the Start, Settings, Communications, Identification as the Device Name. At this point, with basic IrDA functioning− we can move on to establishing a PPP connection for WinCE. These scripts can also be used for serial cable connects. Create the following files and copy them into the directory indicated.

`/usr/sbin/cebox.sh` – make it executable

```
#!/bin/sh
pppd call cebox
```

Because Microsoft likes to break standards, you need the following chat script. This will feed WinCE the proper ASCII keywords it wants before allowing a PPP connection.

`/etc/ppp/cebox.chat`

```
TIMEOUT 3600
"CLIENT"        "CLIENT\c"
""              "SERVER\c"
```

The following file will allow you to specify the IP addresses, IR (or serial port if using a cable) device, DNS and so forth. I do not recommend you change the 192.IP addresses below. WinCE really has an affection for 192.168.55.100 because all the MS synch tools seem to have it hardcoded. DNS can be whatever you normally use.

`/etc/ppp/peers/cebox`

```
/dev/irnine 115200 crtscts
connect '/usr/sbin/chat -v -f /etc/ppp/cebox.chat'
noauth
local
192.168.55.101:192.168.55.100
ms-dns 10.2.0.1
```

2.5.10.4. Testing the connection

Ok, now you can test the connection to make sure it all works. Reboot your machine, run "irmanager −d 1", it should automatically run **irattach /dev/ttyS2** (/dev/ttyS2 being the serial port your BIOS sees the IR device as, if **irattach** is not running, start it) Align the IR ports, at the Linux command prompt type **/usr/sbin/cebox.sh**, and simultaneously press return to start **cebox** and double tap your connection icon in WinCE. You should get a happy message from WinCE reporting Connecting to Host, Device Connected, Authenticating User, User Authenticated and finally Connected. You should see something like this when you are connected:

```
irda0     Link encap:IrLAP  HWaddr 06:89:d0:58
          UP RUNNING NOARP  MTU:2048  Metric:1
          RX packets:246 errors:0 dropped:0 overruns:0 frame:0
          TX packets:251 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:8

ppp0      Link encap:Point-to-Point Protocol
          inet addr:192.168.55.101 P-t-P:192.168.55.100 Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
```

The following script sets up IrDA, establishes a ppp connection with WinCE, and then sets up IP masquerading. It is provided as an example of how you can tie this all together. This is more or less a manual approach. You can get creative, start **irmanager** at boot and stick a line in `inittab` to constantly look for a WinCE connection on the IR port. This will however, run down your batteries and limit your ability to access other IR gadgets. I just use the script below. Position the device, run **wince** and start communications on your PDA when the script tells you to.

`/usr/local/bin/wince` – make this executable

```
#!/usr/bin/perl
use strict;
#
# Enable IrDA, start ppp0 and set up WinCE masquerading
# A. Tyde - Linuxcare Inc.
#
print "\n-> Setting up IR infrastructure...\n";
system("killall irmanager 2>/dev/null");
system("killall irattach 2>/dev/null");
sleep 1;
system("irmanager -d 1");
sleep 1;
system("/usr/sbin/cebox.sh");
print "   Start WinCE Serial or IR networking now!\n";
open(ECHO,">/proc/sys/net/ipv4/ip_forward") or die "Can not open /proc/sys/net/ipv4/ip_forward";
print ECHO "1";
close (ECHO);
print "   Serving 192.168.55.100 to WinCE device...\n\n";
system("ipchains -F");
sleep 5;
system("ipchains -P forward DENY");
system("ipchains -A forward -s 192.168.55.100/32 -j MASQ");
exit 0;
```

2.5.11. Linux to Linux Connection

2.5.11.1. Connection Methods

There should be *three* ways to get two Linux machines connected via Linux/IrDA.

- Dag Brattli wrote about the *IrOBEX support*: "The awakened reader may wonder what prevents the beaming of files from Linux to Linux? Well, nothing!! (but I haven't tried that yet). This means that we now have a "simple" way of beaming files between Linux laptops. I think that this may be the "killer app" we all have been waiting for!" Try to "**load_misc irobex** at both ends, and then try **iroabex_app get** on one of the machines and **irobex put <file>** on the other.".
- Via *Linux/IrDA network* connection. I suppose you have to load the module **irlan_client** at one machine and **irlan_server** at the other one.
- With *IrCOMM* support, in other words over a serial line, which could mean **minicom**, **pppd**, etc. If you want just now to use IrCOMM between Linux boxes, please add this line to `/etc/conf.modules` of _one_ box:

  ```
  # set ircomm protocol engine to client-only mode
  options ircomm ircomm_cs=1
  ```

Note: Don't add it to both boxes, or they cannot accept incoming connection each other! But since 2.2.7 there's no need to add **options ircomm ircomm_cs=1** to `/etc/conf.modules` anymore. Please remove it if you are using it.

2.5.11.2. Compression

Please note this feature is still quite experimental! Dag Brattli wrote: "Just wanted you to know I have just added COMPRESSION support to IrLAP! As you may know, this is _not_ part of the IrDA(TM) standard, but Linux can now negotiate with its peer and check if it has the same compression capabilities). So obviously if you are talking to Win95, Palm III or whatever, you will _not_ get compression!!! This is something which is exclusive for Linux as far as I know! The IrDA(TM) standard says that devices should ignore unknown field in the negotiation header, so we are still "compatible" with IrDA(TM) (have just borrowed an unused header value).

If you want to try using the compression code (Linux <−> Linux) you will have to insert the **irda_deflate** module some time before you actually make the connection. I do it before **irattach**.

The compression standard I have added is the deflate format used by the zlib library which is described by RFCs (Request for Comments) 1950 to 1952 in the files `ftp://ds.internic.net/rfc/rfc1950.txt` (*zlib format*), `rfc1951.txt` (*deflate format*) and `rfc1952.txt` (*gzip format*).

The compression interface is similar to PPP, so you can add as many different compressors as you want. Currently there is only support for GZIP, but BSD compression will be added later. ... Have just tested GZIP compression at 4Mbps. It was a really bad idea! Compressing the frames takes so much time that the performance is actually worse than when not using compression at all. The conclusion is that compression should only be used for SIR speeds, ..."

2.5.12. Multiple Instances

Dag Brattli wrote: "The IrLAP layer has been enhanced to allow more than one instance (so I can use IrLAN on my built−in ir−port, and communicate with the Pilot over the IrDA dongle at the same time) ... So how do you make two Linux/IrDA connections? Well, you just fire up **irattach** for each of the IR ports you have like this:

```
irattach /dev/ttyS0 38;     (my ESI dongle)
irattach /dev/ttyS2 38;     (my builtin IrDA port)

insmod irlan_client
insmod irobex
```

They will not see each other if you run them on the same machine, since they will initiate discovery exactly at the same time. You should however be able to use them against two other laptops. I can run a dongle, builtin IrDA port and a IrDA pcmcia card at the same time with three other IrDA devices without any problems.

You should notice that if the devices can interfere with each other then it might be difficult to obtain a connection, since a device is not allowed to transmit if the media is busy. I sometimes have to put a book between them."

2.5.13. Connection to Docking Station

Dag Brattli: "Connection to the Tekram IRDocking IR−660 http://www.tekram.com/Hot_Products.asp?Product=IR−660 . This device is a docking station with LAN access, printer, mouse and keyboard. You can also use them at the same time as the internal mouse and keyboard! Just fire up **gpm −t ps2 /dev/irkbd** and the laptop will make a keyboard/mouse connection to the IR−660. Now I just have to make **gpm** read both `/dev/psaux` and `/dev/irkbd`, and then make X11 read `/dev/gpmdata`, and I should have the thing configured!

... one problem: **gpm** can handle *multiple mice*, but Linux cannot handle *multiple different keyboards*. So if you have one norwegian keyboard and one *remote* US keyboard like I have, then things will be a little bit confusing. I got a hint from Alan Cox about a project that is implementing *real* support for multiple keyboards, so I'll check that out.

... OK, I sort of worked it out. By using TIOCSTI on `/dev/console`, you can insert scancodes directly into the tty queue. This can be a problem for virtual consoles that expect to receive some translated and cooked keycodes, but X happens to like raw scancodes, so this will work quite nice when using X but not for other virtual consoles. Anyway this is good enough for me, so I will not use a lot of time converting the scancodes to keycodes and index them with some keymap just to make it work with text only virtual consoles. As I see it the irkbd driver has now been successfully been ported to user−space :−)

... the Tekram IR−660 device can, in addition to attach a keyboard and mouse, also print using IrTTP (it can print using IrLPT, but that is not so funny since it requires exclusive use of IrLMP, and you don't wan't to stop the network, mouse and keyboard just to print a document). I'll try and see if I can get IrTTP printing working using a fifo as well.

... Tekram has added a control channel in addition to the data channel so that you can get some status information about what is going on. The name of their own protocol is P1248. It's published through the "P1248" class and "IrDA:TinyTP:LsapSel" LM−IAS entry, so you can try to find it.

... Canon is using the P1248 protocol, and their printer monitor program BrintBuddy2 (Japanese version) is using this protocol now. I don't know what they use for the data channel. Maybe they support TinyTP directly in addition to the other methods. You can try and look up the "IrLPT" class with the "IrDA:TinyTP:LsapSel" in the LM−IAS and see if you can find it."

2.5.14. Connection to Keyboard

The Linux/IrDA keyboard driver is now in user−space. Please see chapter Connection to Docking Station above.

Lichen Wang: "The so called IrDA−D standard is designed to transfer Data. It is not suitable for IR Keyboard. IrDA−D is what Dag ported to Linux OS and what MS ported to Windows OS.

The so called IrDA−C (Control) is designed for Keyboard, Joy−stick, etc. I am not aware that there is any product in the market that is using it yet.

IrDA−D cannot talk to IrDA−C. IrDA−C cannot talk to IrDA−D either. Both the *physical encoding/decoding* and the *software protocol* are very different.

It is possible to implement both IrDA−D and IrDA−C in the same device. Sharp says that IrDA−D and IrDA−C can coexist −− as long as both of them are not used at the same time in the same IR space. This sounds rather funny to me. According to this definition, anything can co−exist with anything as long as you do not destroy the universe permanently in the process ;−)

Seriously, what SHARP says is that they can tailor the IrDA−D so that there are some unused time between the negotiated maximum turnaround time and the actual transmission. They then squeeze the IrDA−C frames in those unused time. The IrDA−D Primary and IrDA−C Master must be implemented in the same device. The keyboards will work, but mice and joysticks may be sluggish at times."

2.5.15. Connection via Serial Cable

For some reasons it may be useful to connect via serial cable instead of using a real infrared link. Bjorn Hanson wrote: "Using a cable, I managed to get a PPP connection through my Ericsson SH888. I did the following (maybe some steps are *wrong* but they worked for me :−)

- added **alias tty−ldisc−11 irtty** to **/etc/conf.modules**
- edited /etc/irda/drivers to irattach **/dev/ttyS0**
- manually inserted the irda and irtty modules using **insmod**
- started **irmanager −d1**
- run **kppp** using **/dev/irnine** (through symlink **/dev/modem**)
- executed **stty < /dev/irnine**
- **ping thehost**
- **ifconfig irda0 down**

Everything worked fine for ping and ssh (doing ls −l a couple of times) but the computer hang when I tried to mail (Netscape) this through that PPP. After reboot I tried both Netscape and lynx. Both were able to establish contact but none got any data."

Another way by Claudiu Costin <claudiuc@calderon.pcnet.ro>:

- Linux 2.2.5 with IrDA compiled as modules
- Because irattach don't make kernel to load automatically IrDA stack, let's type **modprobe actisysa**
- Now, **irattach /dev/ttyS1 −d actisys** where COM2 is used for null link
- **irmanager**
- **ping <address>** work very good!

This has to be done for both machines.

Please note this is not the recommended stuff to connect two machines. Use PPP instead. Though I cannot see how this approach is useful I have included it beause it was asked sometimes in the mailing list.

2.5.16. Peer−to−Peer Mode / Direct Mode

IrCOMM and IrLAN work in both modes, but currently I don't have further information about the differences between these modes and how to set them up.

2.5.17. Linux/IrDA with Toshiba Notebooks

Guenther Wieser has written a HOWTO about Toshiba and IrDA. These notebooks need the *toshoboe* Linux/IrDA driver.

2.6. Hardware Supported by Linux/IrDA

2.6.1. Obtaining Information about the Infrared Port in Laptops

To get the IrDA port of your laptop working with Linux/IrDA you may use StandardInfraRed (SIR) or FastInfraRed (FIR).

2.6.1.1. SIR

Up to 115.200bps, the infrared port emulates a serial port like the 16550A UART. This will be detected by the kernel serial driver at boot time, or when you load the `serial` module. If infrared support is enabled in the BIOS, for most laptops you will get a kernel message like:

```
Serial driver version 4.25 with no serial options enabled
ttyS00 at 0x03f8 (irq = 4) is a 16550A     #first serial port /dev/ttyS0
ttyS01 at 0x3000 (irq = 10) is a 16550A    #e.g. infrared port
ttyS02 at 0x0300 (irq = 3) is a 16550A     #e.g. PCMCIA modem port
```

2.6.1.2. FIR

If you want to use up to 4Mbps, your machine has to be equipped with a certain FIR chip. You need a certain Linux/IrDA driver to support this chip. Therefore you need exact information about the FIR chip. You may get this information in one of the following ways:

1. The **irda−utils** contain a program **findchip** now.
2. Read the *specification* of the machine, though it is very rare that you will find enough and reliable information there.
3. Try to find out whether the FIR chip is a *PCI* device. Do a **cat /proc/pci** . The according files for 2.2.x kernels are in `/proc/bus/pci` . Though often the PCI information is incomplete. You may find the latest information about PCI devices and vendor numbers in the kernel documentation usually in `/usr/src/linux/Documentation` or at the page of Craig Hart http://members.hyperlink.net.au/~chart . From kernel 2.1.82 on, you may use **lspci** from the **pci−utils** package, too.
4. Use the *DOS tool* **CTPCI330.EXE** provided in ZIP format by the German computer magazine CT ftp://www.heise.de/pub/ct/ctsi/ctpci330.zip . The information provided by this program is sometimes better than that provided by the Linux tools.
5. Try to get information about *Plug−and−Play (PnP)* devices. Though I didn't use them for this purpose yet, the **isapnp** tools, could be useful. At the page from Craig Hart I found this PNP IDs:

   ```
   IBM0002   IBM Thinkpad Infrared Port
   IBM0034   IBM Thinkpad Infrared Port
   ```

```
PNP0510   Generic IRDA-compatible device
PNP8294   IrDA Infrared NDIS driver (Microsoft-supplied)
PNP8389   Peer IrLAN infrared driver (Microsoft-supplied)
```

6. If you are running Linux−2.3.x and PCMCIA−CS−3.1.x, turn on the PnP BIOS in PCMCIA−CS. Then, do a **lspnp −v** (from Thomas Davis).
7. If you have installed the *Linux/IrDA software* load the FIR modules and watch the output of **dmesg**, whether FIR is detected or not.
8. Another way how to figure it out explained by Thomas Davis (modified by WH): "Dig through the FTP site of the vendor, find the *Windows9x FIR drivers*, and they have (for a SMC chip):

```
-rw-rw-r--   1 ratbert  ratbert       743 Apr  3  1997 smcirlap.inf
-rw-rw-r--   1 ratbert  ratbert     17021 Mar 24  1997 smcirlap.vxd
-rw-rw-r--   1 ratbert  ratbert      1903 Jul 18  1997 smcser.inf
-rw-rw-r--   1 ratbert  ratbert     31350 Jun  7  1997 smcser.vxd
```

   If in doubt, always look for the .inf/.vxd drivers for Windows95. Windows95 doesn't ship with _ANY_ FIR drivers. (they are all third party, mostly from Counterpoint, who was assimilated by ESI)."
9. Also Thomas Davis found a package of small *DOS utilities made by SMC*. Look at http://www.smsc.com/ftppub/chips/appnote/ir_utils.zip . The package contains **FINDCHIP.EXE**. And includes a **FIRSETUP.EXE** utility that is supposed to be able to set all values except the chip address. Furthermore it contains **BIOSDUMP.EXE**, which produces this output:

   Example 1 (from a COMPAQ Armada 1592DT)

```
     In current devNode:
           Size      = 78
           Handle    = 14
           ID        = 0x1105D041 = 'PNP0511' -- Generic IrDA SIR
           Types:  Base = 0x07, Sub = 0x00,  Interface = 0x02
                Comm. Device, RS-232, 16550-compatible
           Attribute = 0x80
                CAN be disabled
                CAN be configured
                BOTH Static 38; Dynamic configuration
      Allocated Resource Descriptor Block TAG's:
           TAG=0x47, Length=7 I/O Tag, 16-bit Decode
                                   Min=0x03E8, Max=0x03E8
                                   Align=0x00, Range=0x08
           TAG=0x22, Length=2 IRQ Tag, Mask=0x0010
           TAG=0x79, Length=1 END Tag, Data=0x2F
```

   Result 1:

   Irq Tag, Mask (bit mapped −) = 0x0010 = 0000 0000 0000 0001 0000 so, it's IRQ 4. (start at 0, count up ..), so this is a SIR only device, at IRQ=4, IO=x03e8.

   Example 2 (from an unknown machine)

```
     In current devNode:
          Size      = 529
          Handle    = 14
          ID        = 0x10F0A34D = 'SMCF010' -- SMC IrCC
          Types:  Base = 0x07, Sub = 0x00,  Interface = 0x02
               Comm. Device, RS-232, 16550-compatible
          Attribute = 0x80
               CAN be disabled
               CAN be configured
               BOTH Static 38; Dynamic configuration

     Allocated Resource Descriptor Block TAG's:
          TAG=0x47, Length=7 I/O Tag, 16-bit Decode
                                Min=0x02F8, Max=0x02F8
                                Align=0x00, Range=0x08
          TAG=0x22, Length=2 IRQ Tag, Mask=0x0008
          TAG=0x47, Length=7 I/O Tag, 16-bit Decode
                                Min=0x02E8, Max=0x02E8
                                Align=0x00, Range=0x08
          TAG=0x2A, Length=2 DMA Tag, Mask=0x02, Info=0x08
          TAG=0x79, Length=1 END Tag, Data=0x00
```

Result 2:

a) it's a SMC IrCC chip

b) one portion is at 0x02f8, has an io−extent of 8 bytes; irq = 3

c) another portion is at 0x02e8, io−extent of 8 bytes; dma = 1 (0x02 =0000 0010)

Thomas Davis has placed some device information at http://www.jps.net/tadavis/irda/devids.txt .

WARNING: The package is not intended for the end user, and some of the utilities could be harmful. The only documentation in the package is in M$ Word format. Linux users may read this with **catdoc**, available at http://www.fe.msk.ru/~vitus/catdoc/ .

10. Use the *Device Manager* of Windows9x/NT or **WINMSD** (Windows NT Diagnostics) in Windows NT 4.0, **WINMSD** is quite useful, and a bit more informative than Windows9x's Device Manager.
11. You may also use the *hardware surveys* mentioned below.
12. And as a last ressort, you may even *open the laptop* and look at the writings at the chipsets itselfs. Here is an probably incomplete list of manufacturers: Chrystal, Hewlett Packard (HP, chipsets are marked HSDL), Hitachi, IBM, National Semi Conductor (NSC), NEC, Philips, Sharp, Standard Micro Systems Corporation (SMC/SMSC), Texas Instruments (TI), VLSI, Winbond. As an example of application circuits the HSDL−7001 (from a HP brochure, modified by WH):

```
    LEDs        Encode/Decode        SIR/FIR

   HSDL-1001    HSDL-7001          UART 16550/
                                  MicroController
   ______     ______________     ____________
  |      |   |              |   |            |
 (||   TXD|<---|IR_TXD     TXD|<---|SOUT         |
  |      |   |              |   |            |
  |      |   |           RCV|--->|SIN          |
  |      |   |              |   |            |
```

```
(||   RCV|--->|IR_RCV  16XCLK|<---|BAUDOUT      |
  |      |    |          NRST|-+  |             |
   ------      --------------  |   ------------
                               V
```

2.6.2. How to Make Inrared Light Visible

If you have a IRDA aware printer, you can point you phone at it and you should see a light near the IR port light up or flash. If you have a Palm organizer w/ IR, point it at the phone. If the IR on the phone is in discovery mode, you should see the "Waiting for Sender" dialog box pop up on the Palm.

There is a program for the Palm called "IRMonitor" which measures IR emissions. You can get this off EuroCool or PilotZone. You run the program, point your Palm at where you think the IR beam should be coming from and if there are any emissions from that port, they will show up as a spike on the IrMonitor scrolling graph. I would highly recommend this program to anybody out there who plays with or works with IR devices.

You may also use a video camera to detect infrared light. But I couldn't check this yet.

Most of this section is taken from a posting to the Linux/IrDA list by "The Armadillo with the Mask" <armadilo@daft.com>.

2.6.3. Big Endian

Though the source is build to work with big endian machines, I didn't get any reports about actually using it. It would be interesting if it's actually working or not. You will probably need an IrDA dongle or something to test it.

i386 and *alpha* are little endian, *arm* can choose (but the NetWinder has been wired as a little endian machine). *m68k*, *sparc* and *ppc* are big–endian! *mips* can choose I think. If unsure, look in `/usr/src/linux/asm/byteorder.h` and check if it includes **<linux/byteorder/big_endian.h>** or **<linux/byteorder/little_endian.h>** .

2.6.4. SMP

Dag Brattli: "The problem ... has not been analyzed yet. It may be unsafe, and it may work pretty well. All upstream traffic should be safe (bh's), but downstream traffic _may_ be dangerous. I really don't know. ... All list operations `irqueue.c` in Linux–IrDA is currently SMP safe!". Please check the source code for the current status.

2.6.5. Hardware Surveys

There are some surveys about Linux and infrared capable devices in the WWW:

* The Linux/IrDA Project − Hardware Survey at http://irda.sourceforge.net
* Takahide Higuchi at http://www.pluto.dti.ne.jp/~thiguchi/ir/product.html.This page is in japanese.
* I have also set up a hardware survey at MobiliX. This list also contains information about infrared capable devices which are not mentioned here (mice, printers, remote control, transceivers, etc.).

  To make this hardware survey list more valuable it is necessary to collect more information about the infrared devices in different hardware. You can help by sending me a short e−mail containing the exact name of the hardware you have and which type of infrared controller is used.

  Please let me also know how well Linux/IrDA worked, at which tty, port and interrupt it works and the corresponding infrared device (e.g. printer, cellular phone) you use.

  You can also help by contributing detailed technological information about some infrared devices, which is necessary to develope an according driver for Linux.

Chapter 3. Advanced Topics

3.1. Graphical User Interface (GUI)

Currently there are two graphical user interfaces for Linux/IrDA under development:

3.1.1. GNOBEX

A GNOME application developed by Dag Brattli http://irda.sourceforge.net with support for drag'n drop from the GNOME file manager **gmc**. It will also show the progress of the file transfer and give some better error messages when something goes wrong. The GUI isn't finished yet, but if you want to try the GUI you will need the Perl−GTK+ module.

Annotations about CORBA by Dag Brattli: I have just had the first successful test running ORBit/CORBA over IrDA sockets/IrTTP! ORBit is btw. the GNU CORBA implementation used by the GNOME project. The goal is to make it possible for GNOME applications to work between your laptop and your stationary PC without having to first set up a TCP/IP link. Applications on the laptop can then make use of CORBA exported services on your stationary machine.

IrDA (as a one hop technology) fits nicely into the network hierarchy since ORBit will now choose which *profile* to use in this order:

1. Same process (same address space), Some short circut mechanism is used that I don't know much about. It should however be nearly as fast as a procedure call.
2. Same machine (different address space), UNIX domain sockets are used
3. One hop away, IrDA is used (if IrDA is available and a ORBit/CORBA capable device is discovered)
4. Multiple hops away (if all other methods fails). IIOP, TCP/IP will be used.

I use my laptop and a NetWinder for testing. The machines are connected by both Ethernet and IrDA. It's really nice to start a CORBA session and see that the IrDA does discovery, IAS−query, and connects if a CORBA capable device is discovered. If somethings goes wrong, IIOP (TCP/IP) will automagically be used instead. When the transaction if finished, the link goes down again."

There is also OBEX support to the gnome−calendar application **gncal**. Just click on an event and beam it to your Palm Pilot! Still needs to do some cleanup, and support for beaming the other way as well.

3.1.2. KDE

A KDE application developed by Thomas Davis. Look at his page http://www.jps.net/tadavis/irda.

Here's your chance to contribute! Both GUI's need some icons. Any icons need to be:

- each of them should display a printer, PC, PDA, LAN connection, etc.
- the format is not really important, but PNG is what will be used in the end
- set size (48x48 pixels seems to be a common size, I think)
- large and mini (ask about size for that; mini's are for docking and such)
- 16 colors
- free for use
- please, don't blatently copy MS icons!

Please contact the developers.

3.2. Power Saving

In the specifications of my HP OmniBook 800 it is recommended to turn off the IR port, if it is not in use, because it may consume up to 10 percent of the battery time.

If necessary, you may also try to disable the **Fast RRs** feature in the IrDA section of the kernel. This option will give you much better latencies but will consume more power.

3.3. Beyond IrDA

3.3.1. Extending Transmission Distance

According to the IrDA specification the range is up to 1 meter. From the "IrDA Data Link Design Guide" p. 20 by Hewlett−Packard http://www.hp.com/go/ir : " In some cases it may be desired to increase link distance beyond the 1 meter guaranteed by IrDA. The two ways to do this are to increase transmitted light intensity, or to increase receiver sensitivity. In order to extend the link distance, both sensitivity and intensity must be increased for both ends of the IR link. If it is desired to communicate with a standard IrDA device that may have minimum transmitter intensity, the receiver intensity must be increased. The standard IrDA device may also have minimum receiver sensitivity, so transmitter intensity must also be increased."

Andreas Butz wrote: "This might be a silly question, but has anyone an idea whether the whole IrDA stack really relies on a two−way connection, or whether there are some parts of it that could be abused for a one−way connection, ideally for unreliable data? We're trying to modify some IR dongles to broadcast information to palm pilots over several meters distance (cover a whole room), and since we don't want to modify the pilots themselves, and increasing the sensitivity on the receiver side seems unlikely to work, we're stuck with a one way link.". Please see the mailing list archive for details of the discussion.

Sent by Marc Bury <Marc.Bury@NETGEM.com> " .. just heard about some Philips new scheme for remote controls: they call it *IRDA − Control*. This is supposed to be bi−directional, 75 kbps data rate, multiple simultaneous devices (up to 8) and with a minimum 6 meter range!" More information at http://www.irda.org/ .

The german magazine ELEKTOR issued a guide to build a *Long Distance IrDA Dongle* (20m, RS232, IrDA 1.0), ELEKTOR 5/97 p. http://www.elektor.de .

<tzeruch@ceddec.com>: The main problem is that you generally have to make the receiver more sensitive. Basic physics has the inverse square law: the intensity drops with the SQUARE of the distance, so going from 1 to 5 meters requires 25x the power (and battery drain on a portable device), or 25x the sensitivity (and dynamic range − it still has to be able to work at 3 inches). And if you want to do it on the other end, it doesn't simply have to be 25x more sensitive, it must pick up the tiny IrDA pulse needle in a haystack of florescent lights, screen savers, moving shadows ...

Someone tried it with a Palm III upgrade board http://home.t−online.de/home/PSPilot/ppppiii.htm .

Also laser diodes (pulsable) were recommended by K−H.Eischer: But they are more expensive. And the laser diodes are also dangerous if they have more than 1 mW. A better solution would be to use lenses to focus the beam. There is a minimum of absorbtion in the air (I don't know the right frequency) and you should use IR diodes with this frequency.

James wrote: " Who ever it was wanting to do long distance with IrDA, we've tried this before. The best approaches are:

- *wavelan* − buy the cards but not the antennas you can make your own with equaly good gain as the $9000 type they sell here.
- *microwave* − you can pick up X−band doppler radar modules, tune them slightly apart and use the your local TX as the LO for the incomming RX, the whole thing behaves like ethernet and you can hook it onto an AUI port, this may now be illegal.

- *ir* − Many people sell kits which transmit video over Ir, they come complete with the large fresnel lense you need, they manage about 4MHz b/w over 100m.
- *laser diodes* − when we looked at these they were a pain, I think elantec make decent drivers but modulating them was a big pain, Steve Carcia had a series on articles on modulating He−Ne lasers but be careful they have lots of volts in them that want to get out and kill you.

Whatever you choose IrDA might very well be a good choice for a protocol, given it's one of the few that sensibly copes with simplex."

3.3.2. Upcoming Standards (Bluetooth and IrDA)

"More and more people now think that IrDA and Bluetooth will live happily side by side, and the idea of Bluetooth as the IrDA killer just don't work anymore. IrDA is still unbeatable in price/performance and with the new additions to the standards family like AIR and VFIR, it's really good to see that IrDA is moving in the right direction."

3.4. Troubleshooting, Mailing List

3.4.1. General Information

If you encounter problems. Try the following:

- Read the FAQ section below.
- Look at `/var/log/messages` and/or `/var/log/kern`.
- Do a **dmesg**.
- Look at the different files in `/proc/irda`.
- Look at the *mailing list archivs*, whether your problem is already known. Since August 1999 it the archiv is located at Linux−IrDA mailing list archiv. All mails before are archived at http://www.ita.chalmers.se/~svinto/hypermail/irda/ .
- As a last ressort ask in the *Linux−IrDA mailing list*. You may subscribe at Linux−IrDA mailing list . You are welcome to use this mailing list for posting questions, answers, bug−reports, patches, suggestions and comments. It would be much easier to help you if you provide some information. Please include:

```
uname −a
cat /proc/net/irda/irlan
cat /proc/net/irda/irlap
irdadump
```

- There is also a new IrDA related mailing−list. This list is driven by Dag Brattli and the Pasta project, and is intended for discussion about IrDA protcols, setups, configurations, devices, and problems that users and developers might have and want to discuss with others. So this is ment to be a much more general list than the Linux−IrDA list. You can find the list at: Linux−IrDA list. Archives can be found at: Linux−IrDA list archiv . From Dag Brattlis announcement: Though "IrDA is already running a public (but web−only) mailing−list, and I hope it's OK that we run these two lists in parallel. I'm not trying to have any competition ;−) but my intentions are purely to make it easier for users and developers to publicly discuss IrDA related (and non−Linux specific) issues with each other. I have named the list IrDA <irda@pasta.cs.uit.no> since it's about IrDA, so I hope this is OK for IrDA. If you visit the list, you will see that I have choosen to call it: *IrDA −− The Pasta Projects Unofficial IrDA Forum*.

3.4.2. Troubleshooting Techniques

Although I'm not much of a hacker I collected some tricks to track errors or bugs in the Linux/IrDA software.

- You may set the debug level in `/proc/sys/net/irda/debug` to 1, 2, 3, 4.
- Use the files in `/proc/sys/net/irda` to try different parameters like **echo 0 > /proc/sys/net/irda/discovery**. The `/proc/*/irda` files are:

```
root@duckman:~# ls /proc/sys/net/irda/* /proc/net/irda/*
/proc/net/irda/discovery        /proc/net/irda/irlmp            /proc/sys/net/irda/devname
/proc/net/irda/irda_device      /proc/net/irda/irttp            /proc/sys/net/irda/discovery
```

```
/proc/net/irda/irias            /proc/sys/net/irda/compression
/proc/net/irda/irlap            /proc/sys/net/irda/debug
```

* It is also possible to debug the code. But I don't know how to do this. If you want to use SKB debug code, you may edit `irda.h` and change **/include/linux/skbuff.h** (see revision history of snapshot 10−2−98).
* For problems with the **irda** module a utility from the *modules package* **kdstat** might be helpful. But I was not able to try this.
* "You can now alter the number of discovery packets used (1, 6, 8 or 16) and the timeout between sending them (2−8 * 10 ms) in `/proc/sys/net/irda`. Please experiment if you have problems discovering your device. My Palm III seems to like 16 **discovery_slots** and 8 (*10 ms) for **slot_timeout**. " ... "The absolute minimum for reliable discovery of the IR−610 seems to be 9."

  Another statement: ... the Palm III does not like 8 discovery frames in a row, but 6 is OK. With 8 it will answer 1 out of 6−10 times, with 6 it answers every time. I really don't know if this is a problem with Linux−IrDA or the Palm III. One solution to this problem, is to cycle though some different discovery methods for each discovery like this:

  Disocvery 1: send 8 xid frames with 80 ms separation

  If answer, keep the same config, if no answer, try next config

  Discovery 2: send 6 xid frames with 80 ms separation

  Discovery 3: send 8 xid frames with 90 ms separation

  Discovery 4: send 6 xid frames with 90 ms separation

  Discovery 5. Go back to 1.

  or some other pattern and maybe more combinations.

  Maybe this is sometimes implemented, so it would be enabled if **/proc/sys/net/irda/discovery_slots** is set to **0** .
* If anybody gets a kernel Oops, then please feed it to the `../linux/scripts/ksymoops/ksymoops` program, so that we can find out where it went wrong. Just cut out the Oops lines from the syslog, save them to a file, and then run **ksymoops <file>**
* Dag Brattli wrote: I found out that the cs4232 sound card was giving me several hundred interrupts per second! I removed the sound stuff from my kernel, and the machine is now generally about 4 times faster! Linux/IrDA may get problems if you are running the esound server (**esd**) on your machine. Both my machines, a 166Mhz Pentium laptop and a 200Mhz Pentium Pro cannot run Linux/IrDA when esd is running. The reason is that esd makes the soundcard give interrups over 300 times/second which makes the serial driver overrun when receiving. This is because the serial driver now uses slow interrupts in Linux−2.2 (everything is slow interrupts in 2.2), so the interrupt−handler schedules on its way out. The good thing about slow interrupts is that packets are delivered much faster, since you don't need to wait for the next timer−tick. The only exception for this is the pc87108 driver which works fine since it uses DMA and will only give a couple of interrupts per packet.
* There are also some userspace tools **irdaping** and **irdadump** to check Linux/IrDA connections.
* AFAIK it is possible to use IrCOMM either with an infrared device or via serial cable. Maybe this

give some debugging possibilities, too.

- 1) You may edit `/etc/conf.modules`, adding the following lines:

```
option irda irda_debug=3
option irlpt_client_debug=3 irlpt_common_debug=3
```

2) Make sure the irda modules have been totally removed.

3) Edit `/etc/syslog.conf`, adding the following lines:

```
*/*                    -/var/log/all
```

4) Do **killall −1 syslogd** .

5) Print, or do whatever causes problems with **irlpt** .

6) Check all the files in `/var/log/` .

3.5. IrDA Network Neighborhood

3.5.1. Laptop−Printer−PDA

You can take a little peek at http://irda.sourceforge.net Drag−n−drop stuff, so you will be able to drop files to your PDA (uses IrOBEX) or drop files to your printer (uses IrLPT) etc.

3.5.2. Bridging/Routing

James wrote: " ... there is a much better way of doing the briding which is routing. This is entirely user land and requires no kernel patches.

It's in two parts (you may only need one your milage may vary...) the first called **irdaipcfg** does the following:

1) First part is executed as **irdaipcfg ifeth ifirlan** daemonizes, then looks for ARP packets on ifirlan, checks that the arp was not generated by the machine on which it is running. The arp contains the ip address of the machine on the other end of the irlan (it was generated by the gratuatous arp in the irlan code). The program then sets up a host route to this ip address via ifirlan, adds a proxy arp to ifeth for it and generates a gratuatous arp on ifeth. It writes the ip address of the client in `/var/run/host.ifirlan` so you can easily undo all of this from a script.

2) Second part is executed as **gratarp ifirlan**. Sometimes the gratuatous arp seems to get lost in the pipe work, **gratarp** deamonizes and spits out a whole stream of the things...

I use them as follows: (you can use them to do whatever you like)

On my host (the machine bolted to my local net) irlanx is brought up as 10.192.0.1 with a netmask of 255.255.255.255 and a broadcast of 10.192.0.1 by my ifup script from `/etc/irda/network` by **irmanager**. `/etc/irda/network` then runs **irdaipcfg eth0 irlanx** and this does the routing.

From /etc/irda/network

```
"start")
        echo 1 >/proc/sys/net/ipv4/conf/all/forwarding
        ./ifup ifcfg-${device}
        /sbin/irdaipcfg ${localnet} ${device}
    ;;
"stop")
        host=`cat /var/run/host.${device}`
        if [ .$host != . ]; then
          /sbin/arp -d ${host} dev ${localnet}
          /sbin/route delete ${host} dev ${device}
        fi
        ./ifdown ifcfg-${device}
        /sbin/ifconfig ${device} down
    ;;
```

on the client I set up irlan to use an address on my normal subnet 10.32.32.51 but with netmask 255.255.255.255 (not my usual netmask) I have some static routes which are host 10.192.0.1 dev irlan, and net default gw 10.192.0.1 dev irlan. I run **gratarp** from the `/etc/irda/network`, and I can wander arround my house and not lose telnet and ssh sessions ... they are sitting in ftp://bullard.esc.cam.ac.uk/pub/irda "

3.5.3. IPv6

AFAIK IPv6 has neighbor discovery mechanismem, but I don't have information about Linux/IrDA used with IPv6. Please see the mailing list archive for a discussion of this topic under the subject :"patch−2.2.7−ac1−irda4" .

3.5.4. DHCP

I have got reports that it is possible to use **dhcpcd** with IrLAN. Please use latest DHCP software.

3.6. Linux/IrDA and APM

Fons Botman wrote: "When I hibernate my HP OmniBook 2000CT, (Fn−12 diskimage is written to disk, machine turns off completely) with irtty active and turn it on again, irda does not work. I can see it trying to reply to discovery frames it receives from a windows box, using irdadump on the OmniBook. but the windows PC does not see the replies. If I just kill irattach and remove irtty and serial, and start irattach again, it starts working again. Does this occur with other linux laptops also? Is it a problem in the serial device driver? " Also Pedro Figueiredo reported this problem for a Fujitsu LifeBook 735DX.

Answer by Dag Brattli: "Could you all check if the same thing is happening when your're using PPP (and not using IrDA). I guess the APM stuff shuts down the serial port, so that the driver will need to reinitialize it when waking up again. This is properly implemented by some of the PCMCIA drivers I know about, but I really don't think the serial driver gets any events from the APM system.

So here you have your own little kernel project. Start adding APM support to irport which will be the easiest thing (and also to the FIR drivers), then you can start adding a patch to the serial driver (if needed). Again I think the PCMCIA subsystem may be a good source on how to fix it properly."

3.7. Java – IrDA Interface

Jean Tourrilhes has written the Java–IrDA interface.

3.8. IrDA and USB

There is a USB IrDA Bridge Device spec at Rev 0.9B , it's being adopted as a USB class specification. You can find it at under 0.9 Class Specification header. The only device I know is the ESI 9685 USB−Irda (FIR) dongle.

3.9. IrDA Protocols

From Lichen Wang I have taken this remarks:

3.9.1. Existing IrDA Protocol Implementation

IrLAP, *IrLMP*, and *TinyTP* are pretty stable now. But various individual implementations may still have quirks.

IrCOMM is widely used, but Microsoft vowed to drop it and use *IrSocket* only.

IrObex, *IrMC*, *IrTran−P*, etc. are showing up in various devices. The supports on PC are segmented and sometimes very poor.

3.9.2. New IrDA Protocol Development

AIR is pushed mainly by IBM. You get wider angle and longer distance (10M?) but may be at lower data rate. You can also time−multiplex this already slow channel. In my opinion the benefit to the user is very little. AIR hardware and software changes are costly. I think you will have to wait for a long time, if ever, to see any product using AIR.

IrBUS is pushed mainly by Sharp. It is aimed at remote control, IR keyboard, etc., not for data communication. I think if Sharp succeed, you may see IrBUS in set−top boxes, web−tv, etc. If and when these devices reaches a critical mass, some PC may incorporate IrBUS in order to control these devices.

VFIR is the latest. The benefit is higher data rate (16 Mbps). Unlike AIR and IrBUS, the software changes for VFIR are minimum. The changes are mainly in hardware.

3.9.3. Summary

In summary, I do not think we need to wait in anticipation of new IrDA protocols. A easy to install and solid implementation of the existing IrDA protocols is the cutting edge!.

3.10. Known Bugs

If you find a bug, please send a bug report to the mailing list, including **dmesg** output, and which Linux version, and hardware you are using. Thank you!

Sometimes IrCOMM fails to connect (especially when both devices discover each other. You can disable discovering with **echo 0 >/proc/sys/net/irda/discovery**)

A **CR** (carriage return) character cannot be transfered between two linux boxes via IrCOMM with **cat file >/dev/irnine** and **cat /dev/irnine**. It causes a strange thing and freezes your Linux box. Compiling the pc87108 device driver non modular crashes the kernel on boot. Temporary solution: compile the driver as a module

IrOBEX may eat some data on receive. The bug is most probably in the user−space side of IrOBEX.

3.11. FAQ

- Q1 − Question: I do not know anything about ports and irqs. What should I do?
- Answer:

  PART A: Hardware settings

  − 1 Have a look at your hardware specs!!! If not available look at the support page of your vendor, or contact the support hotline. You might also find the information in one of the hardware surveys mentioned above.

  − 2 Use a current BIOS. Usually available at the support page of your vendor.

  − 3 Try **setserial /dev/ttyS? −g −a | egrep 16550A**. One of the shown devices is probably the one you are looking for. Usually it is the second one, but with no guarantee.

  − 4 Note: What seems like an UART is physically the IrDA controller. For my HP Omnibook 800 this is the VLSI VL82C147 PCI − IrDA controller. These controllers should behave up to 115 200 bps like UART's. But sometimes it is very difficult to get the right configuration.

  PART B: How to tell the kernel about the hardware settings

  −4 **cat /proc/ioports** to see which ports are already in use.

  −5 **cat /proc/interrupts** to see which interrupts are already in use.

  −6 Make ports and interrupts available for use with the IR device, e.g. stop the pcmcia service or include a line like this in `/etc/sysconfig/pcmcia`:
  PCIC_OPTS="irq_list=3,4,5,7,9,10,12,14,15"

  −7 Now try to guess what the right interrupt and port is. Use **setserial /dev/ttySx irq M port 0xNNNN** to tell the kernel. If there is more then one possible chance try them all (Note: As mentioned in the *Serial−HOWTO* you should not try irq 0, 1, 6, 8, 13, 14).

  −8 If you were successful please send these parameters to the author, because I would like to include them in the IR hardware survey.

  −9 Good luck.

  It might also be necessary to fine tune the IR serial port with setserial, e.g., **setserial /dev/ttyS0 spd_vhi** (speed rate 115200).

- Q2 − Question: For me, **irattach** hangs, but recognizes the printer. `/var/log/messages` shows that irattach found my HP LaserJet 6P.
- Answer: The "hang" is normal for irattach. Everything is working right if you see the HP Laserjet show up in the log. "hang" means irattach is polling the IrDA−Devices for incoming connections. If you kill it with <CTRL C> the irattach program crashes and /dev/ttySx does not work anymore. The problem is within the irda module, and not with the irattach program. Rebooting is the only thing to

do! Next time put irattach in the background by using **irattach &**. Stop it if necessary with **killall irattach**. Recommendation by Andreas Butz: To my knowledge, <CTRL Z> **bg** should work, too, but I haven't tried it in this specific case. Normally it has the exact same effect as appending **&** to a command.

- Q3 − Question: I get a message like tcsetattr read/write error in /var/log/messages.
- Answer: Caused probably by wrong /dev/ttyS* or wrong irq or port.
- Q4 − Question: Every setting seems alright, because I get the appropriate messages. But it still does not work.
- Answer: Move the devices to within 0.5 meter (1.5 feet). Check that only one application is using the infrared port. Check that both devices are using the same protocol, such as IrOBEX or IrCOMM.
- Q5 − Question: I have downloaded the latest snapshot, and compiled it successfully under Linux 2.0.33 running on an IBM Thinkpad 560E. In the absence of any other IrDA machines to test with, is it safe to assume that once the module has been inserted and the syslog reports "irattach: Serial connection established.", is the IR really working, and will it start to respond once there is another machine with which to talk?
- Answer by Dag Brattli: Sorry, this only means that irattach has done its part of the job, which is just to start the irda−tty. Maybe the message should have been different, but as I said, it tells that the serial connection between the irda−chip and the irda−driver is established.

  Note: Support for IrDA on 2.0.x kernels has been discontinued. You are encouraged to switch to 2.2.x kernels and use the newest IrDA patches available at http://irda.sourceforge.net.

- Q6 − Question: At startup **modprobe −a** checks `/lib/modules/<KERNEL−VERSION>/net/irda.o` and causes the messages: "IrLAP; Missing IrTTY /IrLMP Error no IrLAP connection" (in `/var/log/messages` and on the console).
- Answer by Werner Heuser: Workaround for SYSTEM V style systems: Put a script named for example "ir_rmmod" containing

```
#!/bin/sh
echo "$0 : remove irda module"
rmmod irport.o
rmmod irtty.o
rmmod irda.o
```

  in the startup process (`/etc/init.d` and a symbolic link name for example "S100ir_rmmod" in `/etc/rc3.d` to "ir_rmmod"). (Verify the path for "sh"). For BSD style systems try the corresponding approach.
- Q7 − Question by Ho Chin Keong: Is there other way of setting up communication between the 2 laptops besides setting up a LAN route between the two?
- Answer by Dag Brattli: Yes and no! One of the IrDA standard, IrCOMM permits you to emulate a serial cable between two laptops, so you can use any application written for serial ports (terminals, PPP, slip, etc.). This is however not yet implemented in Linux/IrDA. The IrLPT (printer) support is actually a subset of IrCOMM, so some of it is working!
- Q8 − Question by Ho Chin Keong: If I block the infrared path deliberately for more than 10 seconds, the connection could not re−establish. I have to kill the irattach and restart the whole procedure to start the infrared route. The connection could be maintained, however, if the blocking is less than 10 seconds. Is this part of the design or a bug? Is there any way whereby we can lengthen this time limit from 10 s to longer or infinitely?
- Answer by Thomas Davis: This seems to be a bug in the primary side of the IrLAP/IrLMP code. It appears not to send the reset/disconnect notice all the way back up the stack. You'll notice it when IrLPT gets stuck in the query mode while you were trying to talk to a printer, and disconnected/interrupted it when it was handshaking. (and now, it shows up in the IrLAN portion)
- Q9 − Pierre−Guillaume Raverdy asked: Should I update to IR lib on my palm and update the system

to version 3.0.2?

- Answer by Dag Brattli: You should not need to update your Pilot, but it should not do any harm to do so. It is however required if you want to use the IrCOMM library from IsComplete
- Q10 − Pierre−Guillaume Raverdy asked: Also any simple source code (especially on the palm side) would be greatly appreciated.
- Answer by Dag Brattli: Get the Pilot SDK from Palm. Unzip the examples.zip and take a look at the beamer application.
- Q11 − Is there any IrDA support for BSD?
- Answer: Linux/IrDA seems to be the only available GPL source yet.
- Q12 − By Rui Oliveira: I am having a problem connecting a PalmIII to a Linux box with an Actisys 220L adapter. With a motherboard adapter (no brand but, I think, similar to the Actisys 210L) I simply redirect a pilot syncronization tool (pilot−xfer) to `/dev/ttyS1` which has the ir adapter attached and, using IrLink in SIR mode, I can get the Linux box to talk with the PalmIII. Trying the above through a serial port with a serial−irda Actisys 220L adapter I can't get this to work. My question is :What happens if one just throws data into a serial port with a irda adapter?
- Answer by Lichen Wang: In terms of hardware, IrDA SIR needs a serializer− deserializer, an encoder−decoder, and a transceiver. The UART that drives the COM port of any PC is a serializer−deserializer. In some PC, there is also an encoder−decoder which can be enabled or disabled by the BIOS. When it is disabled, the COM port is usable as an old COM port. When the encoder−decoder is enabled, usually the COM port is no longer usable but an IrDA port is now usable instead. Actisys IR−210 is a SIR transceiver and thus can be used if the PC has this kind of UART with an IrDA encoder−decoder and the BIOS has enabled it. Under this hardware configuration, you need to tell the Windows setup program that you have "standard infrared devices" and with "Built−in Infrared port on laptop or desktop". Actisys IR−220, on the other hand, includes both the encoder−decoder and the transceiver. It is designed to be used with a regular UART. If the UART in the PC has also the encoder−decoder built−in, you must use BIOS to disable that. Under either of this hardware configuration, you need to tell the Windows setup program that you have an "ACTiSYS" manufactured "ACT−IR220L Infrared Wireless Interface". To answer your question: In addition to throwing data at the serial port, you need to tell the UART and the encoder−decoder what data rate to use. In the case of a built−in encoder−decoder, when you set the data rate of the UART, the encoder−decode also get set correctly. In the case a separate encoder−decoder, you need to tell both of them the data rate separatly.
- Q13 − If I try to make a connection, say telnet, it takes an incredibly long time for the login prompt to appear.
- Answers by Renaud Baldura, Dag Brattli and Hee Thong: ... it's a DNS problem. The resolver times out trying to reverse−resolve the IP address of your incoming connection. I think just renaming `/etc/resolv.conf` to something else takes care of it. ... or add some static bindings in /etc/hosts for the machines you want to access in your ad−hoc network. That should avoid the DNS lookups. ... If both machines are in a private test environment, put the following line in the `/etc/host.conf`, **order hosts, bind**. This will make the machine check the `/etc/host` file before doing a DNS lookup. Remember to update the host file on both machines to reflect the IP and host names of the 2 machines.
- Q14 − Question by David LaPorte: I was wondering if anyone has had any success getting the irda port on the Toshiba Tecra 740cdt working. ... I've read that it should show up at IRQ 11, ttyS2. Well, I have a PCMCIA modem which steals ttyS2 and the PCMCIA controller steals IRQ 11. Does anyone have any suggestions?
- Answer by Dag Brattli: If you still have Win95 on your machine, you should go to the device manager and change the PnP setup for the IrDA port (something else than the stuff your're already using). You could for example move away ttyS1 (in Win95), so that it uses the values that the PCMCIA card is going to steal, and then use the settings from ttyS1 for ttyS2.

```
dagbnb ~/linux/test/ > cat /etc/sysconfig/pcmcia
```

```
PCMCIA=yes
PCIC=i82365
PCIC_OPTS="irq_list=7,9,10"
CORE_OPTS=
```

... should make sure the PCMCIA controller stays away from irq 11. Also make sure that the IrDA port is enabled in Win95 since it's disabled by default.

II. Infrared Remote Control

Table of Contents

Chapter 4. Infrared Remote Control

4.1. Resources

Remote control via infrared is not the aim of the Linux/IrDA project but is included in this HOWTO to cover "Linux and Infrared" more completely. I found *three projects* which worked on this topic. You may find some links to current information at MobiliX.

- LInux Remote Control − LIRC

  is a package that supports receiving and sending IR signals of the most common IR remote controls. It contains a device driver for hardware connected to the serial port, a daemon that decodes and sends IR signals using this device driver, a mouse daemon that translates IR signals to mouse movements and a couple of user programs that allow to control your computer with a remote control. Takahide Higuchi wrote about LIRC: "It's great, and it seems almost complete solution, but it seems there is almost nothing supporting hardware on the market (or need to solder some special circuit ... it is hard work for many people to do so). I believe that LIRC will be more popular if consumer IR support is implemented in FastIR drivers and some common API (for example, a raw IrSocket and common ioctls) is made!". You may find LIRC at http://fsinfo.cs.uni−sb.de/~columbus/lirc/

  To subscribe to the LIRC mailing list send an email to <lirc−request@xmission.com> with the word "subscribe" in the body of the message. There is also a mailing list archive at http://www.wh9.tu−dresden.de/~heinrich/lirc/list−archive/
- Serial Infrared Remote Controller

  This is a simple, cheap device that can be connected to any serial port to control most components that have infrared remote controls. It was designed and built on a solderless breadboard and is finally designed as a PC board. You may find this package at http://www.armory.com/~spcecdt/remote/remote.html
- Infrared Tools for the COREL Netwinder PC

  Ryan Shillington wrote some tools to control the COREL Netwinder via infrared, for example:

  Server Side for the Corel Palm Administrator (deamon). It depends on having ir−simple installed and up and running. With this you can check and change IP addresses, Gateway addresses, setup eth1, etc. You can also run simple commands AND you can check the Temperature, Memory, Load averages, etc.

  Client Side for the Corel Palm Administrator. You can also run simple commands AND you can check the Temperature, Memory, Load averages, etc.

  A very basic Infra Red device driver. This does not support IrDA (only unreliable transfers). It looks specifically for Remote Control signals (and Keyboard, etc.). It blocks and passes data up very differently.

  You may find the tools at http://www.netwinder.org/~ryansh/
- ir

  ir is an interface program to Chris Dodge's RedRat 2 infrared controller to send and receive infrared signals to/from consumer devices like TV's, VCR's, cable boxes, and stereos. It is written in Perl. It uses only the basic Perl constructs and no external packages, so it should work on any platform that supports Perl and serial communications. It can be accessed via the command line or cron, as an email handler (through aliases), or as a cgi script which will automatically generate a form with all possible codes. It has macro capability so one command can send a series of IR signals. With an X−10's IR543, it can be used to control X10 devices, too.

* irmctl

  irmctl is a utility daemon to control your favorite non−IRDA infrared receiver. For the moment, only irman (through libirman) is supported.
* IRManager

  IRManager is a Linux daemon to make advanced use of an IRMan infrared receiver. It forwards IR signals to (multiple) native IRMan applications, and can be used with your own scripts and applications. It also has a mapping system and its advanced configuration options make it the most flexible and easy way to remote control your computer.
* irXxD

  irXxD irXxD is a library for sending/receiving infrared remote control codes. It includes kernel 2.0 and 2.2 modules for receiving/sending IR codes under Linux, and various support for other operating systems.

4.2. Infrared Remote Control − IrDA

Two of the above mentioned projects use some kind of selfmade dongle for infrared remote control. There is also a description to build a serial IrDA dongle by yourself in the german ELEKTOR 5/97 p. 28 magazine. Maybe someone can merge these two kind of dongles together.

For a discussion of the relation between Infrared Remote Control and IrDA I quote from the Linux/IrDA mailing list (shortend and modified by wh):

Ryan Shillington wrote: "Remote IR and ASK−IR are very different from FIR or MIR or SIR.

Remote IR and ASK−IR are very low speed and low frequency (but very long range) uses for IR. They operate around 2400 baud.

SIR operates at higher rates, and is meant for long range transmission where you need more than a few characters pass through (unlike a remote control).

MIR is a little faster (less range), but with speeds up to 1.15 Mbps, and FIR (where the devices have to be practically touching) is 4Mbps. The range is inversely proportional to the speed you can send data at.

I'm working on drivers for Remote−IR, but you should know that your IR stuff has to support it. Look for protocols like NEC, RC−5 or RC−0 (those are the most common ones).

You can use SIR to receive Remote Control signals. Set your baud rate nice and low and data will come through. BUT, from my experience, it's not the RIGHT data. It's not being analyzed in the right way, and as such, you can't compute the checksums or check it with its complement.

I have managed to get data in (using SIR) with remote controls. I have been told that SIR will read the remote control stuff differently depending on temperature (although I have never had that experience). "

Lichen Wang <lwang1@ix.netcom.com> wrote in response: "The so−called ASKIR in most laptops etc. is not meant for remote IR devices. ASKIR is meant for Sharp Wizard and Zauaus PDAs and some of Sharp's notebook PCs. Sharp stated this long before IrDA was established and is still supporting it to maintain backward compatibility. Apple's Newton had this capability at one time, too.

Briefly, ASKIR uses 9.6 Kbps (19.2 and 38.4 Kbps are also possible) asynchronous data format of 8 data bits, 1 stop bit, and odd parity. The *start* bit as well as all 0 bit in data/parity are transmitted as IR square wave at 500 KHz (DASK sub−carrier). The *stop* bit as well as all 1 bit in data/parity are represented by the absence of any IR transmission.

As you can see, this is totally incompatible with existing IR remote control. [..]

True. Not only can you use SIR hardware to *receive*, you can transmit, too. Of course, there are some limitations.

Most IR remote controls use 38 KHz sub−carrier. 3 times 38 is 114, very close to 115.2. You can set the UART to operate at 115.2 Kbps, 7 data bits, no parity, and 1 stop bit − a total of 9 bits. Each 3 cycles of the 38 KHz sub−carrier can be *received* or *transmitted* as a byte of 0x5B.

There are some physical limitations in addition to the fact that the sub−carrier must be 38 KHz. The SIR *receiver* is not as sensitive to 38 KHz as the IR remote receiver designed for that. The SIR *transmitter* has a much lower duty cycle and thus can not emit a strong sub−carrier either.

IR remote encodes the control signal by turning on and off the sub−carrier at certain specific patterns. Now that you can *transmit* and *receive* the sub−carrier, what remains is all in timing.

For *transmit*, you have to know how many consecutive bytes of 0x5B to send for each burst of the sub−carrier, and how long to be quiet between the bursts.

For *receive*, you have to know how many of the 0x5Bs you received are consecutive, and how long the gaps were between these groups of consecutive bytes. [..]

My experience with the IrDA link distance of SIR, MIR and FIR is somewhat different from what Ryan said. [..]

SIR, MIR and FIR should all work from 0 to 100 cm but in practice:

(a) Some devices may have problems at *LONG* distances.

When possible, place the two communicating devices no more than 50 cm apart. Low power devices, such as pagers, phones, etc. may have even shorter ranges despite the fact that they use SIR instead of MIR or FIR.

(b) Some devices may have problems at *SHORT* distances.

Place the two devices at least a few cm apart. Putting the two devices too close to each other can cause troubles.

It is somewhat intuitive that when the link is not reliable we put the two devices closer together. But it is counterintuitive that too close is not good either. The reason is that the light intensity at 1 cm is 10.000 times brighter than that at 100 cm. At 0.5 cm, it is 40.000 times, etc. The IR receiver manufacturers have difficulties to cover this huge dynamic range. We all have problems reading under a 10 W light bulb, but imagine how it feels under a 100.000 W light!

The IrDA Physical Layer is totally incompatible with the DASK modulation used in IR remote controls. Thus it is not possible to use the same controller function for both FIR and remote control. However, practically all FIR controller chips do include some additional functions to support remote control. National, SMC, and Winbond (just to name a few) all have such I/O chips.

The IR transmitter for FIR and remote control are very similar. I have tried a standard FIR transmitter. It can reach 10 meters for remote control purpose. Thus it performs just as good as transmitters designed for remote control.

The IR receiver for FIR and remote control are somewhat different. A FIR receiver can receive remote control signals but can reach only 1 meter whereas receivers designed for remote control typically can reach 10 meters.

I have an ISA bus adapter with a National I/O chip that supports both FIR and remote control. I also have IR Dongles that include both FIR and remote control receivers. (Plus a transmitter for both modes.) I cannot find any software to support remote control functions. I did my own experiments in DOS (I cannot run Linux yet.) Anybody interest in this? "

Benny Amorsen wrote: "I have a laptop that is supposed to support ASKIR. The mode of the infrared port can be switched to ASKIR in the BIOS. Having to reboot to switch the mode in the BIOS makes it useless, though, so someone would have to find a way to switch on the fly. "

Dag Brattli wrote: It should be possible to use IrControl (formerly IrBus) for IrDA compliant remote controls. I currently don't know about any remote controls using IrControl standard, but there should be some out there (anyone else who knows better?). You should go to the IrDA site (http://www.irda.org) and get the physical layer standard (which includes IrControl I think).

"Normal" IrDA (using IrLAP) is _not_ well suited for remote control because of the connection oriented nature (and just supports 9600bps for connectionless use). The reason for the limited range is eye−safety they say (but I currently don't know why CIR works better using the same power). I have however seen laptops connect at 4−5 meters (but I don't think that any high speed communication would be possible).

Most IrDA chipsets are capable of CIR operation, and it is quite easy to modify the drivers so they talk CIR. Takahide Higuchi has started to look at IrSockets and it would be great if we could open a "raw" Ir(DA) socket which then could send and receive CIR packets. Then all the CIR applications could live in userspace.

I know that Corel is interested in using CIR for controlling the NetWinder (and they actually have running code). Take a look at http://www.slashdot.org/articles/98/12/05/0916216.shtml or http://www.netwinder.org/~ryansh

From the "IrDA Data Link Design Guide" p. 21 by Hewlett−Packard http://www.hp.com/go/ir : " It is possible to transmit and receive signals other than IrDA signals with Hewlett−Packard IR transceivers. For implementation details, please refer to the Application Note, Transceiver Performance with ASK and TV Remote Signals."

From the IR−MAN page http://www.usuarios.com/ib308564/irda.html :

Fortunately, many IrDA devices are compatible with the 38−kbps ASK modulation used in TV remotes. This means that they can work with such kind of infrared type signals. ... However, it seems that there are still many portable computers that can't receive TV infrared stuff.

For desktop computers, there exist *two* options, depending on the motherboard you have. Usually a Pentium MoBo has an I/O chipset ready for infrared communication. There is a special connector where you can connect the transducer. The other option is buying a serial type transceiver that connects to the standard serial port (RS−232) of the computer. ... PC Remote Control has been tested with success using both type of IrDA devices:

1) IRmate IR−210 Serial Port Infrared Adapter. ... The serial port speed at wich the device sends recognizable data values is 2400 bps. I don't know if this speed will be the same for all the adapters of this type or is an unique characteristic of this model.

Look at the examples of data values received to see how similar are them. There are some infrared commands that change a lot every time, difficulting the recognition. In such cases, a great tolerance in the comparison could be used, but the risk of confusion between different commands will be increased. An apropiate tolerance value for almost all cases is 20.

2) Actisys IR2000L connected to an Asus P2B motherboard. ... There are several serial port speeds that work well, although 4800 bps seems to be the best one. Other adapters of this same type work also well using this speed. Take a look at the samples of data sequences received using this device. Some remote buttons send

exactly the same sequence and it's impossible to distinguish between them at all.

3) Asus IR−eye connected to the same MoBo as above. It works as well as the Actisys device.

TV remotes send commands only one way, in a low−speed burst for distances of up to 30 feet. They use directed IR with LEDs that have a moderate cone angle to improve ease−of−use characteristics. Cordless connectivity via IrDA transfers files, point−to−point and bidirectionally, in a high−speed burst for short distances using directed IR with LEDs having a narrow cone angle. IrDA transmissions require relatively careful aiming, and they're easy to block. For this reason, don't expect a great distance while working with the remote unit.

Alessio Massaro <Alessio.Massaro@cern.ch>: wrote: " IrDA doesn't talk to tv−remotes, but it does have the IrCOMM layer to emulate a serial i/f. My guess is that to get LIRC working with it, you should just need ... to read from the IrCOMM virtual serial device (as you would with a /dev/cua or whatever) and use a remote that can be seen by your dongle+IrDAheader pair."

Answer by Dag Brattli: "You are talking about being *normal* serial ports, but that is something at least I have choosen IrDA not to be. I have implemented all the device drivers as network device drivers, so things are a bit different (more frame oriented). The device drivers deliver IrDA frames and currently nothing else.

But I don't think that we must have a tty interface to the IrDA device drivers in order to support more *RAW* reads and writes. And btw. forget about IrCOMM, it has nothing to do with this issue.

I have actually already implemented support for *raw* reads and writes for the device drivers, since some of the dongles require this."

III. Appendix

Table of Contents

Appendix A. Credits

A.1. Credits

Thanks to:

- Dag Brattli – Linux/IrDA core team
- Thomas Davis – Linux/IrDA core team
- Takahide Higuchi – Linux/IrDA core team
- Ralf Zabka
- Benny Amorsen
- Lichen Wang
- Ryan Shillington
- Richard Titmuss
- Fons Botman
- Rui Oliveira
- Jon Howell
- Carlos Vidal
- Joonas Lehtinen
- Markus Schill
- Bjoern Hansson
- Pawel Machek
- Ho Chin Keong
- Bjoern Mork
- Andreas Butz
- Tang Ning
- Philip Blundell
- Toni van de Wiel
- Stefan Dahlke
- Ales Dryak
- Richard Donkin
- Wessel de Roode
- Andrew Chadwick
- James McKenzie
- Gerd Knorr
- Claudiu Costin
- K−H. Eischer
- Alessio Massaro
- Igor Pesando
- Ove Ewerlid
- Gerhard Gonter < g.gonter@ieee.org>
- Florian Lohoff <flo@rfc822.org>
- Jean Tourrilhes <jt@bougret.hpl.hp.com>
- Mathieu Arnold <arn_mat@club−internet.fr>
- Harald Milz <hm@linux−magazin.de>
- Mike Groeneweg <mikeg@scorpion.murdoch.edu.au>
- Christian Rishoej <chrris@mail.dk>
- Guenther Wieser <wieserg@creative−it.com>
- Timo Felbinger <timof@uranos.quantum.physik.uni−potsdam.de>
- Arthur Tyde <aftyde@linuxcare.com> and Bryan Abshier from Linuxcare Inc.
- The members of the Linux−IrDA mailing list.
- The writers of the other HOWTOs which gave me many inspirations.

* The developers of the SGML−Tools which provided some means to write a HOWTO.
* Tollef Fog Heen <tollef@add.no>

Sorry I didn't start to follow the credits when starting the HOWTO, so probably I forgot somebody.

Appendix B. Revision History

B.1. Revision History

- v0.1 to v0.4a, 19 March 1998 to 4 August 1998, drafts, not included in the LDP
- v1.0, 14 August 1998, release to the LDP
- v1.1, 18 August 1998, added info about IrCOMM patch by Takahide Higuchi, minor changes
- v1.2, 24 August 1998, updated to **linux−irda−1998−08−20** snapshot, added FIR section and revision history, minor changes
- v1.3, 27 September 1998, added sections about multiple instances, cellular phones, digital cameras,Linux to Linux connection, the cutting edge − CVS, power saving; some changes in general configuration section, changes in hardware survey section, minor changes
- v1.4, 11 October 1998, better description of IrCOMM support, changes in dongle connection section, changes in Palm III section, minor changes
- v1.5, 12 October 1998, minor changes
- v1.6, 26 October 1998, section about IrManager added, updated to the **linux−irda−1998−10−21** snapshot, changed dongle connection section, minor changes
- v1.7, 1 November 1998, added remote control section, changed dongle connection section, minor changes
- v2.0, 9 January 1999, nearly complete rewrite and rearrangement according to the new structure of Linux/IR which is included into the kernel since 2.1.131, added info about BIOS support into dongle connection section, configuration tool section and CVS section removed
- v2.1, 13 January 1999, minor changes
- v2.2, 26 January 1999, project name changed from Linux/IR to Linux/IrDA, extended the Troubleshooting chapter, changed the order of the Known Bugs chapter after the Troubleshooting chapter, removed some lint
- v2.3, 4 February 1999, added chapter about Eye Safety written by Andreas Butz; spell checking, reworking of Kernel Parameters chapter and additional information by Andreas Butz; minor changes
- v2.4, 9 February 1999, changed information about applying a patch file
- v2.5, 12 March 1999; new URL for Linux/IrDA; added chapters about Big Endian support, **irdaping**, **irdadump** and Beyond IrDA − Extending Transmission Distance; chapter Obtaining Information about the Infrared Port in Laptops improved; added many information provided by Fons Botman to Windows chapter; added SMP chapter; informations about Ericsson SH888 added; removed obsolete FAQs; minor changes
- v2.6, 6 April 1999, added chapters Connection to Docking Station, Connection to Keyboard and Connection via Serial Cable, minor changes
- v2.7, 11 June 1999 started chapter Upcoming Standards (Bluetooth and IrDA), added annotations about CORBA to GUI chapter, minor information about Nokia cellular phones added, added Appendix B Serial Infrared Port Sniffer, started IrDA Network Neighborhood section, started Connection to Psion 5 chapter and Appendix C, minor additions to LiRC chapter, minor changes
- v2.8, 20 September 1999, added LiRC mailing list, changed <htmlurl ... > tag to <url ...>, changed format of conf.modules entries, addition to hardware detection (PCMCIA), added IrDA mailing list, changed address of Linux−IrDA mailing list, minor additions to multiple instances section, added URL of French translation, added new **sersniff** to Appendix B, added section about precompiled packages, added Palm III Connection to Thinkpad 600 chapter, minor changes
- v2.9, 21 September 1999, changes in Printer Connection chapter, spellchecking, added connection to Siemens S25, minor changes
- v2.10, 2 November 1999, minor changes
- v2.11, 9 March 2000, added links to the 'irctl' and 'IRManager' infrared control programs, new base

URL of the document, new chapter Connecting from Linux to WinCE courtesy from Arthur Tyde and Bryan Abshier of Linuxcare Inc., link to IrDA−Java interface added, link to HOWTO about Toshiba and IrDA added courtesy from Guenther Wieser, more information about a connection to S25 courtesy by Timo Felbinger, links to AT commands for cellular phones added, new chapter Code History, link to SH888 phone book tool, a short survey of IrDA protocols courtesy from Lichen Wang, minor changes
- v3.0, 5 November 2000, format changed to DocBook , license changed to GNU Free Documentation License − GFDL, title changed from IR−HOWTO to Infrared−HOWTO, new document URL, links to Linux/IrDA updated
- v3.1, 8 November 2000, links updated, changed to DocBook 3.1

Appendix C. Appendix A – Configuration Script

C.1. Configuration Script

Configuration script by Ove Ewerlid (please change to new major device numbers, [WH]):

```
#!/bin/sh

# You may have problems if kerneld is running!

killall irattach
killall irmanager

sleep 1

rm -f /dev/ircomm
mknod /dev/ircomm c 60 64

rmmod ircomm-tty
rmmod ircomm
rmmod irtty
rmmod irda

insmod irda
insmod irtty
insmod ircomm
insmod ircomm-tty

irmanager    # executes 'irattach /dev/ttyS1' based on /etc/irda/drivers


# Now start your favorite PPP software on /dev/ircomm

# The 'no activity on link' happens if you move the phone out of IR sight, this is no problem once the phone is back in sight!

# Mar 10 12:31:41 octagon kernel: Linux-2.2 Support for the IrDA (tm) Protocols (Dag Brattli)
# Mar 10 12:31:41 octagon kernel: IrCOMM_common, $Revision: 1.13 $ $Date: 1998/10/13 12:59:05 $ (Takahide Higuchi)
# Mar 10 12:31:41 octagon kernel: IrVTD, $Revision: 1.2 $ $Date: 1998/09/27 08:37:04 $ (Takahide Higuchi)
# Mar 10 12:31:41 octagon irmanager: executing: './drivers start  0'
# Mar 10 12:31:41 octagon irmanager: + 0.1 Fri Jul 25 11:45:26 1997 Dag Brattli
# Mar 10 12:31:41 octagon irattach: Serial connection established.
# Mar 10 12:31:41 octagon kernel: IrDA device irda0 registered.
# Mar 10 12:31:41 octagon irmanager: + 0.1 Fri Jul 25 11:45:26 1997 Dag Brattli
# ...
# Mar 11 13:13:43 octagon kernel: IrLAP, no activity on link!
```

Appendix D. Appendix B – Serial Infrared Port Sniffers

D.1. Serial Infrared Port Sniffers

D.1.1. Sniffer by Gerd Knorr

This programm is a courtesy by Gerd Knorr. You may use it to sniff the traffic which is going trough your IrDA port for details of the protocol (change the default ttyS1 in the source if necessary):

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <termios.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/ioctl.h>

#define BUFSIZE 1024

int
read_and_print(int fd, int sec, int usec)
{
    int             rc,l,i;
    char            buf[BUFSIZE+1];
    fd_set          set;
    struct timeval  tv;

    if (sec || usec) {
        FD_ZERO(38;set);
        FD_SET(fd,38;set);
        tv.tv_sec  = sec;

        tv.tv_usec = usec;
        if (0 == select(fd+1,38;set,NULL,NULL,38;tv))
            return -1;
    }

    switch (rc = read(fd,buf,BUFSIZE)) {
    case 0:
        printf("EOF");
        exit(0);
        break;
    case -1:
        perror("read");
        exit(1);
    default:
        for (l = 0; l < rc; l+= 16) {
            printf("%04x  ",l);
            for (i = l; i < l+16; i++) {
                if (i < rc)
                    printf("%02x ",buf[i]);
                else
                    printf("-- ");
                if ((i%4) == 3)
                    printf(" ");
```

```
            }
            for (i = l; i < l+16; i++) {

                if (i < rc)
                    printf("%c",isalnum(buf[i]) ? buf[i] : '.');
            }
            printf("\n");
        }
        break;
    }
    return rc;
}

void
setlines(int fd, int rts, int dtr)
{
    int lines = 0;

    if (rts) lines |= TIOCM_RTS;
    if (dtr) lines |= TIOCM_DTR;

    ioctl(fd,TIOCMSET,38;lines);
}

int main(int argc, char *argv[])
{
    int                 ser,i;
    struct termios      saved_attributes,tattr;
    struct winsize      win;
    char                buf[16];

    if (-1 == (ser = open("/dev/ttyS1",O_RDWR))) {
        perror("open /dev/ttyS1");
        exit(1);
    }

    /* Set the terminal mode */
    tcgetattr (ser, 38;tattr);
    cfmakeraw (38;tattr);
    cfsetospeed (38;tattr,B9600);
    cfsetispeed (38;tattr,B9600);
    tcsetattr (ser, 0, 38;tattr);

    setlines(ser,0,0);
#if 0
    tcsendbreak(ser,0);
#endif

    /* main loop */
    fprintf(stderr,"setup done\n");
    while (-1 != read_and_print(ser,30,0)) {
        usleep(100000);
    }

    return 0;
}
```

D.1.2. sersniff

Written by Jonathan McDowell sersniff is a simple program to tunnel/sniff between 2 serial ports. The program was written to aid with the decoding of the protocol used by the Nokia 9000i Communicator to talk to the NServer software Nokia provides, which only runs under Windows.

Appendix E. Appendix C – User space application for Psion 5 Palmtop Computers: psion.c

E.1. User space application for Psion 5 Palmtop Computers: psion.c

```
/*********************************************************************
 *
 * Filename:      psion5.c
 * Version:       0.1
 * Description:   User space application for Psion 5 Palmtop Computers
 * Status:        Experimental.
 * Author:        Fons Botman <budely@tref.nl>
 * Created at:    Mon Apr 19 21:51:29 CEST 1999
 *
 *     Copyright (c) 1999, Fons Botman, All Rights Reserved.
 *
 *     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.
 *
 *     Neither Fons Botman nor anyone else admit liability nor
 *     provide warranty for any of this software. This material is
 *     provided "AS-IS" and at no charge.
 *
 ********************************************************************/

#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <stdio.h>
#include <errno.h>
#include <assert.h>
#include <signal.h>
#include <time.h>
#include <sys/types.h>
#include <utime.h>

#include <sys/stat.h>
#include <sys/ioctl.h>
#include <netinet/in.h>

#include <sys/socket.h>
#include <linux/types.h>
#include <linux/irda.h>

#ifndef AF_IRDA
#define AF_IRDA 23
#endif /* AF_IRDA */




#define MAX_DEVICES 10

int discover_devices(int fd)
{
        struct irda_device_list *list;
```

```
        unsigned char *buf;
        int len;
        int i;
        int daddr = 0;

        len = sizeof(struct irda_device_list) +
                sizeof(struct irda_device_info) * MAX_DEVICES;

        /* FIXME */
        system("echo 150 > /proc/sys/net/irda/slot_timeout");

        if (!(buf = malloc(len))) {
                fprintf(stderr, "Could not allocate discovery buffer.\n");
                exit(1);
        }
        list = (struct irda_device_list *) buf;

        /* FIXME: discovery does not return when there are no devices */
        if (getsockopt(fd, SOL_IRLMP, IRLMP_ENUMDEVICES, buf, 38;len)) {
                perror("getsockopt");
                exit(-1);
        }

        if (len > 0) {
                printf("Discovered:\n");

                for (i=0;i<list->len;i++) {
                        printf("  daddr: %08x", list->dev[i].daddr);
                        printf("  saddr: %08x", list->dev[i].saddr);
                        printf("  name:  %s\n", list->dev[i].info);
                        daddr = list->dev[i].daddr;
                }
        } else {
                printf("No devices discovered.\n");
        }

        return daddr;
}


int irttp_get_mtu(int fd) {
        int mtu;
        int len = sizeof(int);
        /* Check what the IrTTP data size is */
        if (getsockopt(fd, SOL_IRLMP, IRTTP_MAX_SDU_SIZE,
                       (void *)38;mtu, 38;len)) {
                return -1;
        }
        return mtu;
}


int sendfile(char* filename) {
        int fd;
        struct sockaddr_irda peer;
        int daddr;
        FILE* f;
        int buflen;
        char *buf;
        int rc;
        struct stat s;
        int cnt;
```

```
        int t0, tx, t;
        unsigned long long int fdatell;

        fd = socket(AF_IRDA, SOCK_STREAM, 0);
        if (fd < 0) {
                perror("socket");
                if (errno == EINVAL)
                        fprintf(stderr, "Is IrDA active?, perhaps run irmanager\n");
                exit(-1);
        }

        /* FIXME: We should use a better/any device selection mechanism */
        daddr = discover_devices(fd);
        if (!daddr) {
                fprintf(stderr,"No IRDA device found\n");
                exit(1);
        }

        peer.sir_family = AF_IRDA;
        peer.sir_addr = daddr;
        strcpy(peer.sir_name, "Epoc32:EikonIr:v1.0");

        if (connect(fd, (struct sockaddr *) 38;peer,
                    sizeof(struct sockaddr_irda))) {
                perror("connect");
                if (errno == ENETUNREACH)
                        /* P5: System ^L IrDA is active,
                           but IR-receive not selected */
                        fprintf(stderr,
                                "No Psion5 or IR-receive is not selected\n");
                exit(-1);
        }
        printf("Connected to %x\n", daddr);

        buflen = irttp_get_mtu(fd);
        printf("mtu = %d\n", buflen);
        if (buflen < 2) {
                perror("irttp_get_mtu");
                exit(1);
        }
        if (!(buf = malloc(buflen))) {
                perror("malloc");
                exit(1);
        }
        /*
           FIXME : I got strange results when the buffer size was less than
           the mtu (e.g. 200), the psion did not seem to see the frames were
           not full length, and stopped after the number of frames based on
           the full mtu size.
           investigate.
        */

        /*
           FIXME : psion connects to port 2, but does not get error back
           from us. Linux bug?
        */

        if (!(f = fopen(filename,"rb"))) {
                perror(filename);
                exit(1);
        }
```

```
        rc = stat(filename,38;s);
        if (rc != 0) {
                perror("stat");
                exit(1);
        }

        /* FIXME map psion mode bits to unix filemodes */

        fdatell = 62168263200000000ULL + 1000000 *
                ( s.st_mtime 38; 0x00000000FFFFFFFFULL);
        printf("date: %Ld\n", (unsigned long long) s.st_mtime);
        printf("date: %Ld\n", fdatell);
        sprintf(buf,"FILE %d %d %lu %lu %s",
                (int) s.st_size,
                32 | (s.st_mode 38; S_IWUSR ? 0 : 1),
                (unsigned long) ((unsigned long long int) fdatell >> 32),
                (unsigned long) (fdatell 38; 0x00000000FFFFFFFFULL),
                filename);
        rc = write(fd,buf,strlen(buf));
        if (rc != strlen(buf)) {
                perror("write");
                fprintf(stderr,"rc = %d, strlen=%d\n",
                        rc, strlen(buf));
                exit(1);
        }
        printf("sent: %s\n", buf);
        rc = read(fd,buf,buflen-1);
        printf("Received (%d) ", rc);
        if (rc < 0) {
                perror("reply error");
                exit(1);
        }
        if (rc == 0) {
                fprintf(stderr, "EOF on reply?\n");
                exit(1);
        }
        if (rc < buflen) {
                buf[rc] = 0;
        }
        /* should be "ACK Y" */
        if (0 != strcmp(buf,"ACK Y")) {
                fprintf(stderr, "Unexpected response: %s\n", buf);
                exit(1);
        }
        printf("%s\n", buf);

        cnt = 0;
        t0 = tx = t = time(NULL);

        while (!ferror(f) 38;38; !feof(f)) {
                int wrc;

                rc = fread(buf, 1, buflen, f);
                if (rc == 0) continue;

                wrc = write(fd,buf,rc);
                if (wrc < 0) {
                        perror("write");
                        exit(1);
                }
                if (wrc < rc) {
                        fprintf(stderr, "Problem: only sent %d of %d\n",
```

```
                                wrc, rc);
                        exit(1);
                }

                /* progress indication */
                t = time(NULL);
                cnt += rc;
                if (t - t0 == 0 || cnt == 0 || tx == t)
                        /* avoid division errors */
                        /* only once per second */
                        continue;
                tx = t;
                printf("sent %d/%lu bytes=%4g%% in %d sec,"
                       " %g Kbytes/s, to go %li sec  \r",
                       cnt, s.st_size, 100.0 * cnt / s.st_size, t - t0,
                       cnt / 1000.0 / (t - t0),
                       ( s.st_size - cnt ) * (t - t0) / cnt);
                fflush(stdout);
        }
        if (ferror(f)) {
                perror("ferror");
                exit(1);
        }
        if (cnt != s.st_size) {
                printf("Warning: "
                       "file size changed: initial: %lu, actual: %d\n",
                       s.st_size, cnt);
        }
        if (t == t0) t++;       /* white lie for fast transfers */
        printf("\r%79s\r","");  /* Cleanup the progress line */
        printf("Sent %s, %d bytes in %d sec. %g KBytes/sec\n",
               filename, cnt, t - t0, cnt / 1000.0 / (t - t0));

        /* Check for close on the other side */
        rc = read(fd,buf,buflen);
        if (rc > 0) {
                fprintf(stderr, "Strange: the other side responded.\n");
                fprintf(stderr, "rc=%d, data:%s\n", rc, buf);
                exit(1);
        }
        if (rc == 0) {
                fprintf(stderr, "Received end of file.\n");
        }
        if (rc == -1) {
                if (errno == EPERM) {
                        /* Strange error code to get in this case */
                        printf("Other side closed connection, OK\n");
                } else {
                        perror("last read");
                        exit(1);
                }
        }

        close(fd);
        return 0;
}

int handle_client(int cfd) {
        int buflen;
        char* buf;
        int rc;
```

```
        /* fields of file transfer header */
        unsigned int fsize;
        unsigned int fmode;
        unsigned int fdate1;
        unsigned int fdate2;
        char* fname;

        unsigned int fdate;
        unsigned long long int fdatell;

        FILE* f;
        int cnt;

        int t0, tx, t;

        buflen = irttp_get_mtu(cfd);
        printf("mtu=%d\n", buflen);
        if (buflen < 2) {
                perror("irttp_get_mtu");
                exit(1);
        }
        if (!(buf = malloc(buflen))) {
                fprintf(stderr, "malloc buf failed\n");
                exit(1);
        }

        /* Wait for the other side to send a header */
        /*
           Sample headers received:
           DATA 185
           FILE 55175 32 14689800 2691219200 Data
                size mode datehi  datelo     name
        */

        rc = read(cfd, buf, buflen);
        if (rc < 0) {
                perror("1st read");
                exit(1);
        }
        if (rc == 0) {
                perror("1st read 0");
                exit(1);
        }
        assert(rc < buflen);
        buf[rc] = 0;
        printf("%s\n", buf);

        fsize = 0;
        fdate = 0;
        if (0 == strncmp(buf, "FILE ", 5)) {
                cnt = 0; /* to be safe */
                rc = sscanf(buf, "FILE %u %u %u %u %n",
                            38;fsize, 38;fmode, 38;fdate1, 38;fdate2,
                            38;cnt);
                if (!(rc == 4 || rc == 5)) {
                                /* grumble */
                        fprintf(stderr, "sscanf rc=%d\n", rc);
                        exit(1);
                }
                assert(cnt < buflen);
                fname = strdup(buf+cnt);
                fdatell = ((unsigned long long int) fdate1 << 32) + fdate2;
```

```
                fdate = ( fdatell - 62168263200000000ULL) / 1000000 ;

                printf("filename: %s\n", fname);
                printf("filesize: %d\n", fsize);
                printf("Filemode: %d", fmode);
                printf("%s", (fmode 38;  1 ? ", Readonly" : ""));
                printf("%s", (fmode 38;  2 ? ", Hidden" : ""));
                printf("%s", (fmode 38; 32 ? ", Modified" : ""));
                printf("%s", (fmode 38; ~35 ? ", Unknown" : ""));
                printf("\n");
                printf("fdate1:   %u = 0x%x\n", fdate1, fdate1);
                printf("fdate2:   %u = 0x%x\n", fdate2, fdate2);
                printf("filedate: %Ld\n", fdatell);
                printf("filedate: %d = %s", fdate,
                       asctime(gmtime((time_t*)38;fdate)));

                if (!(f = fopen(fname,"wb"))) {
                        perror(fname);
                        exit(1);
                }
        } else if (0 == strncmp(buf, "DATA ", 5)) {
                cnt = 0; /* to be safe */
                rc = sscanf(buf, "DATA %d", 38;fsize);
                if (rc != 1) {
                        fprintf(stderr, "sscanf rc=%d\n", rc);
                        exit(1);
                }
                assert(cnt < buflen);
                fname = strdup("/tmp/psion5-data");
                if (!(f = fopen(fname,"wb"))) {
                        perror(fname);
                        exit(1);
                }
        } else {
                fprintf(stderr, "Unknown data type: %s\n", buf);
                                /* exit(1); */
        }

        rc = write(cfd,"ACK Y",5);
        if (rc != 5) {
                perror("1st write");
                fprintf(stderr,"1st write rc = %d\n", rc);
                exit(1);
        }

        cnt = 0;
        t0 = tx = t = time(NULL);
        while (cnt < fsize) {
                int wrc;
                rc = read(cfd,buf,buflen);
                if (rc < 0) {
                        perror("data read");
                                /* EPERM on disconnect ? */
                        exit(1);
                }
                if (rc == 0) {
                        perror("data read 0");
                        exit(1);
                }
                wrc = fwrite(buf,rc,1,f);
                if (wrc != 1) {
                        perror("fwrite");
```

```
                        exit(1);
                }
                cnt += rc;

                /* progress indication */
                t = time(NULL);
                if (t - t0 == 0 || cnt == 0 || tx == t)
                        /* avoid division errors */
                        /* only once per second */
                        continue;
                tx = t;
                printf("got %d/%u bytes=%g%% in %d sec,"
                       " %g Kbytes/s, to go %i sec  \r",
                       cnt, fsize, 100.0 * cnt / fsize, t - t0,
                       cnt / 1000.0 / (t - t0),
                       ( fsize - cnt ) * (t - t0) / cnt);
                fflush(stdout);
        }

        if (cnt != fsize) {
                printf("Warning: "
                       "file size changed: initial: %u, actual: %d\n",
                       fsize, cnt);
        }
        if (t == t0) t++;       /* white lie for fast transfers */
        printf("\r%79s\r","");  /* Cleanup the progress line */
        printf("Received %s, %d bytes in %d sec. %g KBytes/sec\n",
               fname, cnt, t - t0, cnt / 1000.0 / (t - t0));

        rc = fclose(f);
        if (rc != 0) {
                perror("fclose");
                exit(1);
        }
        if (fdate) {
                struct utimbuf utb;
                utb.actime = fdate;
                utb.modtime = fdate;
                rc = utime(fname,38;utb);
                if (rc != 0) {
                        perror(fname);
                }
        }
        free(fname);
        close(cfd);
        return 0;
}

int receivefile(int mode)
{
        /*
           The Psion 5 tries the following connections:
           Epoc32:EikonIr:v1.0  IrDA:TinyTP:LsapSel
           IrDA:IrCOMM          Parameters
           IrLPT                IrDA:IrLMP:LsapSel
           connect on 2

           Warning: discovery reply after 101ms
        */

        int addrlen = sizeof(struct sockaddr_irda);
        /* int oflags; */
```

```
        /* int mtu; */
        int fd;
        struct sockaddr_irda peer;
        int cfd;

        /* Create socket */
        fd = socket(AF_IRDA, SOCK_STREAM, 0);
        if (fd < 0) {
                perror("socket");
                exit(-1);
        }

        /* Bind local service */
        peer.sir_family = AF_IRDA;
        strcpy(peer.sir_name, "Epoc32:EikonIr:v1.0");
        peer.sir_lsap_sel = LSAP_ANY;

        if (bind(fd, (struct sockaddr*)38;peer, sizeof(struct sockaddr_irda))) {
                        perror("bind");
                        return -1;
        }

        if (listen(fd, 2)) {
                perror("listen");
                return -1;
        }

        /* FIXME: allow more simultaneous clients */
        for (;;) {
                cfd = accept(fd, (struct sockaddr *) 38;peer, 38;addrlen);
                if (cfd < 0) {
                        perror("accept");
                        return -1;
                }

                if (handle_client(cfd))
                        break;

                if (mode == 1)
                        break;
        }

        sleep(1);
        close(fd);
        return 0;
}


int
main(int argc, char* argv[]) {
        char* argv0 = argv[0];

        if (argc <= 1) {
                fprintf(stderr, "Usage: %s [-s file] [-r] [-d]\n", argv0);
                fprintf(stderr, "\t-s file\tSend file to the Psion\n");
                fprintf(stderr, "\t-r\tReceive a file from the Psion\n");
                fprintf(stderr, "\t-b\tReceive multiple files (batch mode)\n");
                exit(1);
        }

        /* skip program name */
        argv++; argc--;
```

```
        for ( ; argc>0 38;38; argv[0] ; argc-- , argv++) {
                if (0 == strcmp(argv[0],"-s")) {
                        /* FIXME: sending multiple files does not
                           work yet. We need to wait for the user
                           to select receive again on the psion */
                        if (argv[1] 38;38; 0 == strcmp(argv[1],"--")) {
                                /* Allow the user to send ANY filename */
                                argv++; argv++;
                                for ( ; argv[1] ; argv++ , argc--) {
                                        sendfile(argv[1]);
                                }
                        } else {
                                /* Send files upto next switch */
                                while (argv[1] 38;38; *argv[1] != '-') {
                                        sendfile(argv[1]);
                                        argv++; argc--;
                                }
                        }
                } else if (argv[0] 38;38; 0 == strcmp(argv[0],"-r")) {
                        receivefile(1);
                } else if (argv[0] 38;38; 0 == strcmp(argv[0],"-b"))
                        receivefile(0);
                else {
                        fprintf(stderr,"Error: unknown switch %s\n", argv[0]);
                        fprintf(stderr,"Call without args for usage: %s\n",
                                argv0);
                        exit(1);
                }
        }
        return 0;
}
```

Appendix F. Appendix D: Infrared Light and Eye Safety

F.1. Infrared Light and Eye Safety

This section summarizes some ideas and thoughts that were exchanged on the Linux/IrDA mailing list. It is not medically wellfounded, and whoever has better evidence or some more wellfounded source of information is encouraged to contribute it to this HOWTO.

The IrDA spec says that the range of IrDA devices has been limited to 1m for reasons of eye safety. Another plausible assumption is that power consumption and IR pollution/crosstalk were reasons for this limitation. In principle there could be danger for the eye, because infrared light is not registered by the eye, and thus the pupil won't close in order to protect the retina from bright IR light sources. This is the same situation as with UV light, which will cause snow blindness eventually, but in contrast to UV light, IR light contains much less harmful energy due to its longer wavelength.

The only legal restrictions and medical advices we were able to find on the web were concerned with infrared emissions of heat lamps or in the welding process and IEC 825−1 (CENELEC EN60825−1). This suggests that IR light as emitted by IrDA devices will be harmless, since even the peak power emitted by strong IR LEDs (ca. 300mW) is several orders of magnitude below the power emitted by medical IR heat lamps (up to 500W). For these, however, you are supposed to wear protective goggles, so maybe if you are looking straight into 1.000 infrared LEDs flashing at once, you should do so, too. The effect of infrared light is mostly heat, though, and not an alteration or destruction of the biological cell structure, such as caused by UV light. Though in the specs for the HP OmniBook 800 Hewlett−Packard recommends not to look directly into the IR LED.

As stated above, this discussion is only based on guesswork and common sense assumptions about the data found in IR LED and heat lamp specs. If anybody with a better medical knowledge can comment on this, please do so!!!

Appendix G. Copyrights, Disclaimer, Trademarks

G.1. Disclaimer and Trademarks

This is free documentation. It is distributed in the hope that it will be useful, but without any warranty. The information in this document is correct to the best of my knowledge, but there's a always a chance I've made some mistakes, so don't follow everything too blindly, especially if it seems wrong. Nothing here should have a detrimental effect on your computer, but just in case I take no responsibility for any damages incurred from the use of the information contained herein.

Though I hope trademarks will be superfluous sometimes (you may see what I mean at Open Source Definition), I declare: If certain words are trademarks, the context should make it clear to whom they belong. For example "MS Windows NT" implies that "Windows NT" belongs to Microsoft (MS). "Mac" is a trademark by Apple Computer. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and I was aware of a trademark claim, the designations have been printed in caps or initial caps. All trademarks belong to their respective owners.

G.2. Copyrights

G.1. GNU Free Documentation License – GFDL

Version 1.1, March 2000

G.1.1. 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

G.1.2. 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front−matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front−Cover Texts or Back−Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine−readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard−conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine−generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

G.1.3. 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

G.1.4. 3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front−Cover Texts on the front cover, and Back−Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine−readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly−accessible computer−network location containing a complete Transparent copy of the

Document, free of added material, which the general network−using public has access to download anonymously at no charge using public−standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

G.1.5. 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front−matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties−−for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front−Cover Text, and a passage of up to 25 words as a Back−Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front−Cover Text and one of Back−Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

G.1.6. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

G.1.7. 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

G.1.8. 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self−contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

G.1.9. 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

G.1.10. 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

G.1.11. 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.