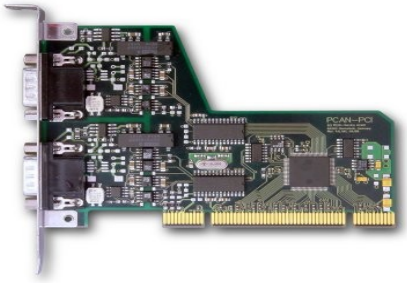


PCAN-driver for Linux

www.peak-system.com
linux@peak-system.com
and Klaus Hitschler
klaus.hitschler@gmx.de

History of the document

first draft	Hi	13.01.2002
rescue after loss of data	Hi	10.02.2002
format revision, read/write tables	Hi	20.02.2002
delete typographic mistakes	Hi	21.02.2002
PCAN-USB integrated	Hi	09.02.2003
obfuscation of „pcan_usb_kernel.c“ is described	Hi	23.02.2003
devfs, kernel-2.5 support, LGPL	Hi	04.08.2003
1st revision for kernel 2.6.x	Hi	02.05.2004
enlarge FAQ, corrections	Hi	13.05.2004
simple correction for RPM make procedure	Hi	14.08.2004
additional compiler switches	Hi	19.07.2005
replaced cat “i ...” with echo “i ...”	Hi	30.11.2005
Added a hint from Uwe Bonnes about 'make scripts'	Hi	14.03.2006
Added PCAN-PC Card support, removed devfs	Hi	14.05.2006
Corrected serious typographic error	Hi	16.05.2006
removed obfuscation, added realtime support	Hi	29.10.2006

table of documents

Disclaimer

Part of this program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License or LGPL Lesser General Public License as published by the Free Software Foundation version 2 of the License. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Features of the 'pcan.o' or 'pcan.ko' driver

The „pcan.o“ or „pcan.ko“ driver is supporting PCAN-PCI, PCAN-ISA, PCAN-PC/104, PCAN-Dongle, PCAN-PC Card as well as PCAN-USB hardware. Depending on the type of hardware (PCI, ISA, Dongle, PC Card, USB) 8 channels for each type can be supported. For PCAN-ISA, PCAN-PC/104 and PCAN-Dongle the base address and the used interrupt are adjustable. If no parameters are provided then the driver uses factory defaults. A user program can communicate with the driver in two different ways:

1. ASCII formatted data can be provided or accepted over a „read()“ or „write()“ interface. This data contains information from received or transmitted telegrams or about channel initializing parameters.
2. With a „ioctl()“ interface. Through this interface user programs are able to receive and to transmit CAN-telegrams as well. However it is possible to get information about the diagnosis of the channel status and to initialize the CAN-channel.

The driver can be used from more than one user program at the same time and the simultaneous use of lots of interfaces and/or CAN channels is supported. If more than one user program communicates with the same CAN-channel the input or output is shared. (e.g.: Many user programs are accidentally sharing the received telegrams.)

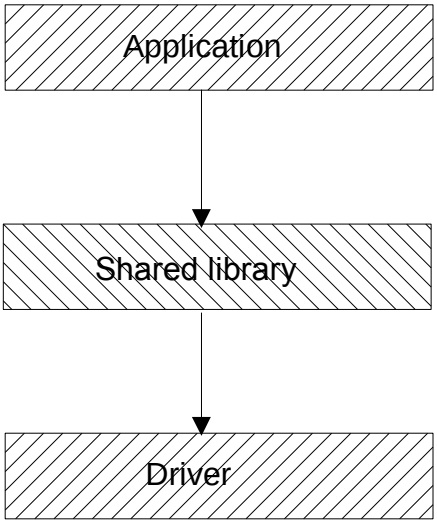
The reading or writing of telegrams can block if the receive buffer doesn't contain data any more or if the write buffer is full. This behaviour can be disabled by opening the path as non-blocking.

The „select“-method is supported.

The interface to the driver (constants and structures) is defined in the header file „pcan.h“.

Features of the 'libpcan.so' library

The „libpcan.so“ library is providing an easier interface for using the properties of the PCAN-drivers. The library interface is similar to the ones from MS-Windows and make the porting of applications easier. There are also some calls which adapt better to the specifics of Linux. They are marked with a 'LINUX_..' as prefix. (Example: HANDLE LINUX_CAN_Open(char *szDeviceName, int nFlags);)



Picture 1. calling hierarchy

The interface of the library is described within the C/C++-header file „libpcan.h“. Please note that „libpcan.h“ includes the „pcan.h“ file.

The source code is under LGPL so it is for utilizing within proprietary software without any exceptions.

Preluding notes

Note: Before installing the software the hardware has to be installed in or at the computer (Exception: PCAN-Dongle, PCAN-PC Card or PCAN-USB, see at (6*)).

This manual is describing the installation process on a x86-Linux-computer with installed kernel 2.6.x. and a SuSE 10.0 system. Exceptions for kernel 2.2.x. 2.6.x and x86-RedHat systems are particularly mentioned.

During the installation sometimes a console and perhaps an editor as well will be used in the „root“- mode. If you want to login as „root“ user within a console or xterm-session you can use the 'su' command or the built in menu „file/root console“.

To start an editor in the „root“ – mode, just invoke from the menu „system/terminals/terminal (system manager mode)“ in KDE. From the console command line you can invoke the editor „kwrite“, „kate“ or „emacs“.

For some distributions the standard path doesn't include the „/sbin“ folder. Then installation commands have to typed in with „/sbin/“ ahead.

Examples:

/sbin/modprobe ...	instead of	modprobe ...
/sbin/insmod ...	instead of	insmod ...
/sbin/rmmod ...	instead of	rmmod ...
/sbin/modinfo ...	instead of	modinfo ...

Installation with RPM

The installation comprise three files:

Installation-en.pdf	description in PDF format.
peak-linux-driver-20040501-3.3.src.rpm	the RPM-source-file.
peak-linux-driver.3.3.tar.gz	the packed tar-sources.

(Note: The term for release or the version for file names can change.)

The installation with RPM creates an installable RPM – package with standard settings for the local target system. If the standard settings don't fit you can do a manual installation.

Installation of source RPM

Please type in the command line (as „root“ user):

```
rpm --rebuild peak-linux-driver-20040501-3.3.src.rpm
```

or

```
rpmbuild --rebuild peak-linux-driver-20040501-3.3.src.rpm
```

if the file „peak-linux-driver-20040501-3.3.src.rpm“ is in your current directory.

This invocation expects the previous installation of the gcc-compiler, the kernel-headers and the make tools.

The translation with and for systems with kernel 2.6.x needs an additionally configured kernel and all of the kernel sources.

At the successful end of this rebuild RPM is filing at:

„/usr/src/packages/“ or „/usr/src/redhat“

a system specific binary-installation for SuSE or RedHat in this folder. e.g.

```
/usr/src/redhat/RPMS/i386/peak-linux-driver-20040501-3.3.i386.rpm
```

Now you can do the installation of the binary package:

```
rpm --install peak-linux-driver-20040501-3.3.i386.rpm
```

Alternatively you can use the GUI based installers „kpackage“ in KDE or „gnorpm“ in GNOME, too.

Manual installation

Manual file unpacking

Unpack the file „peak-linux-driver.x.y.tar.gz“ into an optional folder into your home directory. Example:

```
tar  -xzf peak-linux-driver.3.3.tar.gz
```

Within this directory you get following tree of files:

```
/home/klaus/tmp/peak-linux-driver-3.3/
|-- Documentation                 Any documentary
|   |-- COPYING                   GNU Public License
|   |-- Installation-en.pdf       This installation instructions
|   |-- template.c
|   |-- template.h
|   |-- template.make
|   `-- todo.txt
|-- Makefile                      A 'global' makefile
|-- driver                        The  driver and its sources
|   |-- Makefile                  The makefile for the driver
|   |-- pcan.h                    Application interface definitions of the driver
|   |-- pcan_make_devices         A script for generating device nodes
|   |-- src                       The driver sources
|   |   |-- cobf.h
|   |   |-- pcan_common.h
|   |   |-- pcan_dongle.c
|   |   |-- pcan_dongle.h
|   |   |-- pcan_fifo.c
|   |   |-- pcan_fifo.h
|   |   |-- pcan_fops.c
|   |   |-- pcan_fops.h
|   |   |-- pcan_isa.c
|   |   |-- pcan_isa.h
|   |   |-- pcan_main.c
|   |   |-- pcan_main.h
|   |   |-- pcan_parse.c
|   |   |-- pcan_parse.h
|   |   |-- pcan_pci.c
|   |   |-- pcan_pci.h
|   |   |-- pcan_sja1000.c
|   |   |-- pcan_sja1000.h
|   |   |-- pcan_usb.c
|   |   |-- pcan_usb.h
|   |   |-- pcan_usb_kernel.c
|   |   `-- pcan_usb_kernel.h
|   |-- test.txt
|   `-- wstress
|-- lib                           The access library
|   |-- Makefile                  The makefile of the library
|   |-- libpcan.h                 The library call prototypes
|   `-- src                       The sources of the library
```

```
|      `-- libpcan.c
`-- test                          The test programs
   |-- Makefile                   The makefile to the test programs
   |-- src                        The sources to the test programs
   |   |-- bitratetest.c
   |   |-- common.c
   |   |-- common.h
   |   |-- parser.cpp
   |   |-- parser.h
   |   |-- receivetest.c
   |   `-- transmitest.cpp
   `-- transmit.txt               Test transmit instructions
```

Specifics with the driver installation for PCAN-USB

For using the driver the USB subsystem of the Linux kernel has to be enabled. USB support for Linux was first introduced with kernel 2.4. The use of PCAN-USB together with a Linux kernel version less than 2.4.7 is deprecated through the driver developer.

Specifics with the driver installation for PCAN-PC Card

The user side of the PCMCIA subsystem of the kernel has undergone heavy changes through subsequent releases of kernel 2.6.x . So we have to distinguish between the implementations:

Up from kernel 2.6.13 the vendor and card IDs are'nt any more registered in the configuration file “/etc/pcmia/config” since the relationship between manufacturer-ID and device-ID of the card and the associated driver is registered inside the driver. The driver should be loaded before card plugin.

With this kernel versions you can use the pccardctl utility from the pcmcia-package, especially “pccardctl eject” and “pccardctl insert” to prepare a card for ejection or to notify a inserted card.

For earlier kernel versions than 2.6.13 you need entries in “/etc/pcmcia/config” like:

```
device “pcan”
  class “none” module “pcan”

card “PCAN-PCCARD”
  manfid 0x0377, 0x0001
  bind “pcan”
```

With this entries you can use your configured hotplug system to load the driver when the card is plugged in.

With this kernel versions you can use the cardctl utility from the pcmcia-package, especially “cardctl eject” and “cardctl insert” to prepare a card for ejection or to notify a inserted card.

Please note that removing a card with still open channels is not recommended. Support for kernel versions 2.4.x are not tested at the timing of writing.

Manual installation of the driver

Note: From the kernel version 2.6.0 on the kernel module extension is „ko“. This is why „pcan.o“ is named „pcan.ko“ from kernel 2.6.0 on.

Register as „root“ user in a console e.g. konsole or xterm. Fork to the „.../peak-linux-driver-x.y/driver“ directory. Type in:

```
/sbin/modprobe parport
```

This command installs the „Parport Subsystem“ with Linux. It is necessary to support the PCAN-Dongle integration (1*, 2*). If the command exits without an error message please invoke

```
/sbin/insmod pcan.o type=xxx,yyy,... io=0x300,0x378,... irq=10,7,...
```

or from kernel 2.6.0 on:

```
/sbin/insmod pcan.ko type=xxx,yyy,... io=0x300,0x378,... irq=10,7,...
```

If you need USB support you have to do the same steps as in „Specifics with the driver installation for USB-PCAN“. The place holder „xxx,yyy,...“ are standing for „isa“, “sp“ or „epp“. The parameter for „io“ and „irq“ can be left out, if the standard assignments are configured with your hardware. PCI-channels and USB-channels are found and installed without any special details but ISA (PC/104) and parallel-Port supported channels have to be registered.

The standard assignments for ISA and PC/104 interfaces are (io/irq): 0x300/10, 0x320/5

The standard assignments for Dongle in SP/EPP mode are (io/irq): 0x378/7, 0x278/5

With the following command you can see any error report (as „root“ user):

```
tail /var/log/messages
```

Normally the output looks similar to this:

```
Jan 13 18:28:57 sylvia kernel: pcan: *Name: Release_20020113_a *
Jan 13 18:28:57 sylvia kernel: pcan: pci device minor 0 found
Jan 13 18:28:57 sylvia kernel: pcan: isa device minor 8 found (io=0x0300,irq=10)
Jan 13 18:28:57 sylvia kernel: pcan: epp-dongle device minor 24 prepared (io=0x0378,irq=7)
Jan 13 18:28:57 sylvia kernel: pcan: major 254.
```

You can check the correct installation. Please invoke:

```
cat /proc/pcan
```

You get a similar printout if there are no errors:

```
*----------- PEAK-Systems CAN interfaces (www.peak-system.com) ------------
*------------------------  Release_20060514_a  --------------------------
*-------------------- 8 interfaces @ major 253 found ----------------------
*n -type- ---base--- irq ---read--- ---write-- ---irqs--- --errors-- status
 0    pci 0xdb008000   5 0x00000000 0x00000000 0x00000000 0x00000000 0x0000
 1    pci 0xdb008400   5 0x00000000 0x00000000 0x00000000 0x00000000 0x0000
 8    isa 0x00000300  10 0x00000000 0x00000000 0x00000000 0x00000000 0x0000
24    epp 0x00000378   7 0x00000000 0x00000000 0x00000000 0x00000000 0x0000
32    usb 0x12340020  33 0x00000000 0x00000000 0x00000000 0x00000000 0x0000
33    usb 0x12340021  34 0x00000000 0x00000000 0x00000000 0x00000000 0x0000
40 pccard 0x00000180 169 0x00000000 0x00000000 0x00000000 0x00000000 0x0000
41 pccard 0x000001a0 169 0x00000000 0x00000000 0x00000000 0x00000000 0x0000
```

The driver is configured for the dynamic allocation of the „major“ serial number (4*). This is why the device files have to be generated newly every time after driver installation. Please invoke the shell script for this:

```
./pcan_make_devices  2
```

This script installs two device files for each PCI, ISA, Dongle Devices in SP and EPP mode (3*) and USB-devices. Please verify the result with the command:

```
ls  -l  /dev/pcan*
```

Now you already can do your first test with your CAN hardware. Just connect a CAN-transmitter to your CAN interface and send telegrams with bit rate setting of 500 kbit/sec . Login at a user-console and invoke:

```
cat  /dev/pcan0  for the first PCAN-PCI interface
cat  /dev/pcan8  for the first PCAN-ISA or PCAN-PC/104 interface
cat  /dev/pcan16 for the first PCAN-Dongle interface in SP-mode
cat  /dev/pcan24 for the first PCAN-Dongle interface in EPP-mode
cat  /dev/pcan32 for the first PCAN-USB interface (*9)
cat  /dev/pcan40 for the first PCAN-PC Card interface
```

The received telegrams should be shown (5*). This invocation can be made at more than one console for more than one interface at the same time.

The output of data is already possible. With the call of:

```
echo „m s 0x234 2 0x11 0x22“ > /dev/pcan0
```

for example a CAN telegram with standard frame is transmitted with the identifier 0x234 and two data bytes „0x11“ and „0x22“ from the first PCAN-PCI interface. The bit rate of the interface can as well be set from the so called „write interface“.

Manual installation of the shared library

Please change to the directory „../peak-linux-driver-x.y/lib“. Login as „root“ user and invoke:

```
cp libpcan.so.0.1 /usr/lib/libpcan.so.0.1
ln -sf /usr/lib/libpcan.so.0.1 /usr/lib/libpcan.so.0
ln -sf /usr/lib/libpcan.so.0 /usr/lib/libpcan.so
```

This process is only necessary if you install this program for the first time or if you want to update the library. To use the test programs the library has to be installed.

Manual installation of the header-files

Please copy the header-files into the directory „/usr/include“ as “root” user as followed and customize the access rights.

```
cp peak-linux-driver/driver/pcan.h  /usr/include/pcan.h
chmod 644 /usr/include/pcan.h
cp peak-linux-driver/lib/libpcan.h  /usr/include/libpcan.h
chmod 644 /usr/include/libpcan.h
```

Customization of 'modules.conf' or 'modprobe.conf'

For kernel version 2.6.x the role of the configuration file has been taken over by the 'modprobe.conf' file. Both files are to find in „/etc“.

At the end of the „/etc/modules.conf“ file („modprobe.conf“) a marked addition is added through the installation. The respective type and the eventually non-standard base addresses and interrupt numbers are to be given there, if the driver have to support the PCAN-ISA or PCAN-Dongle interfaces. (Have a look at the description about the manual installation of the driver)

Note: As soon as there already is an entry with the word 'pcan' the current entry is not to be changed. If you wish to have a new entry you have to delete the old one first. To do that you have to use an editor as „root“ user.

Installation of the driver

After rebuilding the sources and the following binary installation on your target computer it's enough to type in

```
/sbin/modprobe pcan
```

to install the driver and any other useful modules. Depending on your requirements you have to do this procedure every time you start your computer or your application.

(This may be done in a start up script, too.)

Initial bitrate

To prevent a wrong initial bitrate when loading the driver we added a driver load parameter which sets the initial bitrate of the related interfaces. For example to set the initial bitrate to 1 Mbit/sec the syntax is:

```
bitrate=0x0014
```

In order to make this setting persistent you have to modify /etc/modules.conf or /etc/modprobe.conf (depending on your kernel version).

Note: Please consider that all interfaces handled with this driver get the same initial bitrate. The corresponding (common) bitrate codes are:

```
#define CAN_BAUD_1M     0x0014  //   1 Mbit/s
#define CAN_BAUD_500K   0x001C  // 500 kBit/s
#define CAN_BAUD_250K   0x011C  // 250 kBit/s
#define CAN_BAUD_125K   0x031C  // 125 kBit/s
#define CAN_BAUD_100K   0x432F  // 100 kBit/s
#define CAN_BAUD_50K    0x472F  //  50 kBit/s
#define CAN_BAUD_20K    0x532F  //  20 kBit/s
#define CAN_BAUD_10K    0x672F  //  10 kBit/s
#define CAN_BAUD_5K     0x7F7F  //   5 kBit/s
```

Semi-automatic installation with compilation

A semi-automatic installation with compilation can also be done with little effort. Unpack the tar.gz-files to „any-directory“ and start compilation and installation. Please note the prerequisites for compilation of the sources.

```
cp peak-linux-driver.3.3.tar.gz  <any-directory>
cd any-directory
tar  -xzf peak-linux-driver.3.3.tar.gz
```

rm peak-linux-driver.3.3.tar.gz
cd peak-linux-driver
make clean
make
make install

Installation test, use of test programs

Two test programs are attached for the initial test of the interfaces. For using the programs please change to the directory „.../peak-linux-driver-x.y/test“. Please invoke „/receivetest --help“. You get following response:

```
receivetest Version "Release_20040412_a"  (www.peak-system.com)
--------- Copyright (C) 2004 PEAK System-Technik GmbH ---------
receivetest comes with ABSOLUTELY NO WARRANTY.     This is free
software  and you are welcome  to redistribute it under certain
conditions.   For   details   see    attached   COPYING   file.

receivetest - a small test program which receives and prints CAN messages.
usage:   receivetest {[-f=devicenode] | {[-t=type] [-p=port [-i=irq]]}} [-b=BTR0BTR1] [-e] [-?]
options: -f - devicenode - path to devicefile, default=/dev/pcan0
         -t - type of interface, e.g. 'pci', 'sp', 'epp', 'isa' or 'usb' (default: pci).
         -p - port in hex notation if applicable, e.g. 0x378 (default: 1st port of type).
         -i - irq in dec notation if applicable, e.g. 7 (default: irq of 1st port).
         -b - BTR0BTR1 code in hex, e.g. 0x001C (default: 500 kbit).
         -e - accept extended frames. (default: standard frames)
         -? or --help - this help

receivetest: finished (0).
```

The default bit-rate settings for all CAN-interfaces are set to 500 kbit/sec. The bit rate can be customized through the command line parameters.

For using a ISA or Dongle interface with the standard settings following detail is enough:

receivetest -t=isa or receivetest -t=sp or receivetest -t=epp

or

receivetest -f=/dev/pcan0

to get to the first PCAN-PCI-interface.

Every received telegram of the interface will be print out. Entering Ctrl-C cancels the program.

The program „transmitest“ can be used to transmit data. Please invoke „./transmitest –help“. You get following output similar to receivetest:

```
transmitest Version "Release_20040412_a"  (www.peak-system.com)
--------- Copyright (C) 2004 PEAK System-Technik GmbH ---------
transmitest comes with ABSOLUTELY NO WARRANTY.     This is free
software  and you are welcome  to redistribute it under certain
conditions.   For   details   see    attached   COPYING   file.

transmitest - a small test program which sends CAN messages.
usage:   transmitest filename {[-f=devicenode] | {[-t=type] [-p=port [-i=irq]]}} [-b=BTR0BTR1]
[-e] [-?]
options: filename - mandatory name of message description file.
         -f - devicenode - path to devicefile, default=/dev/pcan0
         -t - type of interface, e.g. 'pci', 'sp', 'epp' ,'isa' or 'usb' (default: pci).
         -p - port in hex notation if applicable, e.g. 0x378 (default: 1st port of type).
         -i - irq in dec notation if applicable, e.g. 7 (default: irq of 1st port).
         -b - BTR0BTR1 code in hex, e.g. 0x001C (default: 500 kbit).
         -e - accept extended frames. (default: standard frames)
```

```
          -? or --help - this help

transmitest: finished (0).
```

This program expects similar parameters as receivetest with the exception of a file name. In the file with this name the program expects a list of telegram descriptions which are to be send. Those telegrams will be send cyclic until the ending by pressing Ctrl-C. The syntax of the description of the telegrams is:

Tag / Column #1	Description
m	Message string follows
r	RTR-Message string follows
i	Initialisation string follows (write only)
#	Comment follows (write only)

Description of the first column of the telegram of the read/write interface.

Column	Description
2	s = Standard frame, e = Extended frame
3	Identifier (hex)
4	Length of telegram (dec)
5..12	Telegram bytes (hex)
13	Timestamp (dec, read only)
13	Comment (write only)

Description of the following columns of 'normal' and RTR-telegrams.

Column	Description
2	BTR0BTR1 Init data (hex, write only)
3	e = allow extended frames (write only)

Description of the following columns for initializing strings.

For example:

```
# a comment
# a message, standard frame, id=0x123, 0 Data bytes
m s 0x123 0
# a message, standard frame, id=0x123, 1 Data byte, Data
m s 0x123 1 0x11
# a message, standard frame, id=0x123, 1 Data byte, Data
m s 0x123 2 0x11 0x22
# a message, extended frame, id=0x123, 3 Data byte, Data
m e 0x123 3 0x11 0x22 0x33
# a RTR, standard frame, id=0x123, 0 Data byte, comment
r s 0x123 0   # a comment
# a RTR, extended frame, id=0x123, 0 Data byte, comment
r e 0x123 0   # a comment
# initialize
i 0x1234 e
```

Most important header files

Any sources of the driver, of the library and the test programs are fit in underneath the

„peak-linux-driver-x.y“ directory tree. Two files are to be mentioned especially:

The file „.../peak-linux-driver/driver/pcan.h“ describes the interface to the driver and command constants. You have to include this file if you want to have direct access to the driver with „open(), ioctl(), read(), close() etc.“.

The file „.../peak-linux-driver/lib/libpcan.h“ describes the interface to the dynamic linkable library libpcan.so. You have to include this file if your application needs to use the dynamic library.

Please note that libpcan.h is including the pcan.h file.

Compilation of the driver

Use cases

In some cases of using the installed driver it doesn't fit to the configured operating system. This could be following cases:

1. Your kernel sources are not to be found in the standard directory „/usr/src/linux“. You still can compile with following invocations:

   cd ~/peak-linux-driver/driver

   make clean

   make KERNEL_LOCATION=<path-to-my-kernel>

2. You don't need or don't want support for some interface types. Then you have to compile the driver with disabled support for some interfaces. The makefile provides to enable or disable partial compilation with this switches:

   USB = USB_SUPPORT or NO_ USB_SUPPORT

   PCI = PCI_SUPPORT or NO_ PCI_SUPPORT

   DNG = DONGLE_SUPPORT or NO_ DONGLE_SUPPORT

   ISA = ISA_SUPPORT or NO_ ISA_SUPPORT

   As default all interfaces are enabled. For example if you want to suppress PCAN-Dongle and PCAN-ISA Interfaces you have to compile:

   cd ~/peak-linux-driver/driver

   make clean

   make DNG=NO_ DONGLE_SUPPORT ISA=NO_ ISA_SUPPORT

3. The following section only applies to kernel versions 2.4.x or smaller: The driver is to be installed on RedHat Linux. RedHat in contrast to SuSE as default uses the „CONFIG_MODVERSIONS“ modifier to prohibit the installation of improper drivers. If your operating system was generated with enabled „CONFIG_MODVERSIONS“ you can see this easily by invoking „cat /proc/ksyms“. If there are cryptic number-letter combinations at the end of the symbols your system was generated with enabled„CONFIG_MODVERSIONS“.

   A simple new compilation on your system is enough:

   cd ~/peak-linux-driver/driver

make clean
make

4. Your operating system doesn't support the PARPORT_SUBSYSTEM or you have a parallel interface which is unknown to the operating system and you like to use a PCAN-Dongle with this interface.

   Here you have to compile with the „PAR=NO_PARPORT_SUBSYSTEM“ switch:

   cd ~/peak-linux-driver/driver
   make clean
   make PAR=NO_PARPORT_SUBSYSTEM

5. You get runtime errors and want to see more information for diagnosis at „/var/log/messages“. Here you have to translate with the „DBG=DEBUG“ switch:

   cd ~/peak-linux-driver/driver
   make clean
   make DBG=DEBUG

Combinations of these switches are also possible.

Prerequisites for compilation of the sources

Before compiling following prerequisites have to be considered:

1. Up from kernel version 2.6.x the completely configured kernel sources for your target system have to be installed. If it is a standard installation the sources are located under „/usr/src/linux“ in which this normally is a link e.g. to „/usr/src/linux-2.6.6“. For every kernel smaller or equal than 2.4.x only the „kernel-headers“ have to be installed. (It's not wrong to install the complete sources of the target system.) The kernel-headers are installed if the invocation of “cat /lib/modules/'uname -r'/build/include/linux/modversions.h“ is ending without an error. (“'uname -r'“ is a place holder for the result of the invocation “uname -r“.)
2. The “/lib/modules/'uname -r'/build/include/linux/version.h“ file has to exist or being copied before from the path “/boot/vmlinuz.version.h“.
3. Tools like make, gcc, etc. have to be installed. (To edit or investigate foreign source codes I like to use the source navigator of RedHat. http://www.sourceforge.net/projects/sourcenav)

Support for Cross-Compilation

The driver, its library and the test programs sometimes are used on other systems than the development system. The process of generating the binary driver for another target is called Cross-Compilation. Partially the driver and its helper programs are supporting Cross-Compilation. Therefore the driver has to be translated with the option.

make KERNEL_LOCATION=<path-to-my-kernel>

Whether the driver can be compiled for every possible target system and is working there can't be guaranteed because of so many different architectures.
The installation of the driver and the library on an other target than the host system

can't be supported through the given installation scripts.
We are glad about hints how to improve the Cross-Compilation support.

Realtime support with Xenomai

Since Release 'release_20061029_*' realtime support for the Xenomai realtime environment (http://www.xenomai.org/) was merged from a previous rt-branch into the main branch. Most of the work was done by Laurent Bessard and Edouard Tisserant.

The Xenomai project has been launched in August 2001. It has merged in 2003 with the RTAI project [http://www.gna.org/projects/rtai/] to produce an industrial-grade real-time Free Software platform for GNU/Linux called RTAI/fusion, on top of Xenomai's abstract RTOS core.

Installation

First of all, in order to install pcan-rt driver, a Xenomai realtime environement must be installed on your workstation. Download the Xenomai sources from the Xenomai website and install it by following the installation manual provided. The Linux kernel of your workstation must be patched for using Adeos (patches for a lot of architectures are included in the source code).

Don't forget to update your list of dynamic libraries by using 'ldconfig'.

Compilation environment

To compile the pcan realtime driver, you must provide in the make command the option RT=XENOMAI. By default, pcan_driver will be compiled for non-realtime environment.

Don't try to compile pcan realtime driver with USB_SUPPORT or PCC_SUPPORT, they are still not supported.

A example of command for compiling pcan realtime driver:

```
cd ~/peak-linux-driver/driver
make clean
make RT=XENOMAI
```

Runtime environment

You can load pcan realtime driver module as usual, there are no changes.

The pcan API is the same in realtime and non-realtime, so you can open, close, read from or write to your pcan device like usual. But be careful, this must be made in a realtime task. For an example of how to use Xenomai for creating and using a realtime task, please refer to the examples (transmitest and receivetest) located in the 'test' folder.

Don't forget to include in the Xenomai libraries and flags when compiling your program. Xenomai flags can't be obtained by using the 'xeno-config' script provided. You can have an example of how using 'xeno-config' in the 'Makefile' file located in the 'test' folder.

Troubleshooting

Warning: We had had a problem with a old station that makes the kernel couldn't load. To solve this problem, we have added 'lapic=1' in the 'grub.conf' file.

If your kernel still crashes at start, we recommend you to compile your kernel with Xenomai support in module and try to insert it after kernel was loaded. It will then indicate in the log messages what is the problem.

The most common mistake when using a realtime tasks is trying to print some text in the user terminal. It is highly recommended to avoid to do that. In most cases, it will result in a kernel crash.

FAQs

Q. The driver for my PCAN-ISA card is not being installed. What's wrong?

A. One possibility is that the SJA-1000 chip on the card is setting back while checking the PeliCAN-mode. If the card with a PLD that has got a red point and the jumper JP11 is set „on“ this may be the reason. Remedy: To set the jumper to „off“.

Q. My PCAN-ISA, OCAN-PC/104 or PCAN-Dongle doesn't receive any data though I can send. What's wrong?

A. A possible reason is that a wrong interrupt is configured for the device. Then single telegrams can be send however none can be received. Not only the wrong assertion of the interrupt number but often the missing release of the interrupt in BIOS („legacy ISA“) for the PCAN-ISA card can be the reason. (See also (1*))

Q. My PCAN-Dongle is being rejected while using it even if the installation was successful. What's wrong?

A. Normally the reason is a not assigned interrupt for the „parport_pc“ while using it together with the PARPORT_SUBSYSTEM (1*).

Q. What possibilities do I have to diagnose if I have problems?

A. Linux has a variety of choices. First it's useful to look at recent messages with:

“tail -f /var/log/messages“

If the driver is loaded you can see (as root) with

„/sbin/lsmod | grep pcan“

More informations about the parameter of the driver can be asked with.

„/sbin/modinfo pcan“

Closer information about assignment of major- and minor-numbers and about the use of resources can be queried with

“cat /proc/pcan“

Information about assignment of resources can be queried with

“cat /proc/interrupts“

“cat /proc/ioports“

“cat /proc/iomem“

„cat /proc/pci“ or „cat /proc/bus/pci/devices“ depending on the kernel version.

„cat /proc/bus/usb/devices“

Please note that the interrupts of the hardware aren't registered until its use.

To diagnose USB there are also programs with a graphical user interface like e.g. „USBView“.

With garbled installations sometimes paths for translation of the driver are set wrong. That's why while translating the driver a message like this appears:

*** Host machine kernel version=2.4.18-4GB, Driver kernel version=2.6.5, \

Path to kernel sources=/mnt/usr/src/linux, use KBUILD=yes

With this message it is possible to compare if the driver is really translated for the used kernel. The „KBUILD“ mechanism is used from kernel 2.6.x on.

Q. How do I interpret the output of “cat /proc/pcan“ ?

A. The second line gives information about the release of the driver. In the third line the number of the found CAN-channels and the assigned major-number are shown. From line four on the characteristics of the each CAN-channel are described.

The first column shows the minor-number assignment of the channel, the second column shows the type of the channel, the third and fourth column the assignment to base addresses (ports) and interrupt-numbers. The following four columns count the number of processed read-, write-, interrupt- and errors transactions. The last column shows the last occurred error status of the channel. The status is a bit-field with an interpretation like this table:

Bit	Description
0	Chip-send-buffer full
1	Chip-receive-buffer overrun
2	BUS warning
3	BUS passive
4	BUS Off
5	Receive-buffer is empty
6	Receive-buffer overrun
7	Send-buffer is full
14	Illegal parameter

Both columns „base“ and „ir“ are misused with USB channel types. „base“ is showing the serial number of the module and „ir“ the programmed device number of the module. The column „irqs“ in case of type 'usb' counts the number of received or sent USB-packages..

Q. What happens if I plug out the interface while using PCAN-USB?

A. First of all your programs will stall and a re-plug-in of the PCAN-USB doesn't change anything. First you have to stop the PCAN-USB using programs and after plugging-in of the interface you can invoke your programs again. It's not guaranteed that the same minor number as the previous PCAN-USB minor-number is allocated. Minor-number assignment depends on the order of plug-ins of PCAN-USB devices.

Q. How can I find out what minor-number the PCAN-USB is assigned to?
A. With help of the individual pre-programmed „device-number“, which can be experienced with help of the „/proc/pcan“ interface, a relationship to a special physical connection can be made.

Q. I want to open my PCAN-USB device however I get an error message. What's wrong?
A. There is more than one possibility: 1.) The appropriate device is not registered as „/dev/pcan...“. 2.) The appropriate device is not plugged in.

Q. The translation on my system doesn't work. What can be wrong?
A. The most common reason are not installed Linux kernel sources. Often sources of another kernel than the one installed are used. Sometimes both files

„/boot/vmlinuz.autoconf.h“ and
„/boot/vmlinuz.version.h“

of the distribution are not linked to the files they are assigned to.

„/lib/modules/'uname-r'/build/include/linux/autoconf.h“ and
„/lib/modules/'uname -r'/build/include/linux/version.h“

Q. The transmission performance of CAN-messages is low, if I send a telegram with help of the write-interface out of a script. What is the problem?
A. Every line in a script opens the path to a device and immediately closes this path again. This opening and closing takes time. The transmission performance can be improved by letting another path open. For example:

```
cat /dev/pcan32 >/dev/null &
echo „m s 0x123  2 0x11 0x12“ > /dev/pcan32
echo „m s 0x123  2 0x11 0x12“ > /dev/pcan32
```

Q. During installation of the driver with USB-support I get following message: „Warning: loading pcan.o will taint the kernel: non-GPL license – Proprietary“. What's the meaning of this message?
A. This message points out that the driver for the PCAN-USB module is not released under the GPL license. The use of the driver as a module within Linux though is legal.

Q. The source code module „pcan_usb_kernel.c“ only contains cryptic letters. Is the file destroyed?
A. No, the contents of the files are obfuscated to save intellectual property. Though it's still a translatable source code. This feature does only apply for releases earlier than 20061029.

Q. While translating kernel 2.6.x following warning appears:

*** Warning: Overriding SUBDIRS on the command line can cause
*** inconsistencies

Is something not right?
A. I am afraid that the discussion within the kernel-developer community about the right way of translating external modules hasn't found an end until this release. The current way generates this warning and at worst translates **all** of the modules again. Though there are approaches that the generation of external modules like pcan.ko is getting easier. This message can be ignored.

Q. I got a lot of error messages when compiling the driver for a kernel 2.6.x system. What's wrong?
A. To compile for kernel 2.6 target system you need to have a pre-configured kernel. To accomplish this you need to install the target kernel sources. Then do

```
cd /usr/src/linux      # for example
su                     # you need to be root
make cloneconfig       # create a configuration suitable for your running kernel
make scripts           # create the necessary scripts
```

That's all. Now you can – as ordinary user – compile your driver.

Q. While translating the driver following message appears:
*** "Can't find /usr/src/linux-2.4.24.SuSE/include/linux/version.h !". End.
What's the problem?
A. Within the driver makefile up from version 3.3 the target kernel version is not being won out of the interpretation of the command 'uname -r', but extracted out of contents of the file „$(KERNEL_LOCATION)/include/linux/include/version.h“. (If there wasn't any special „KERNEL_LOCATION“ given at the command line of „make“ „KERNEL_LOCATION=/usr/src/linux“ is being used as default.) While translating the kernel sources the file „version.h“ is generated. Normally this message has its cause in not translated kernel sources. Until kernels 2.4.x it was enough if the distribution had installed this file. From kernel 2.6.x on it is necessary that the kernel sources for the target system are configured and translated completely. Then the file „version.h“ is being created.

Q. I like to have interrupt sharing with PCAN-ISA or PCAN-PC/104
A. No problem since version 3.30 of the driver. But you should be aware that interrupt sharing with ISA BUS is only possible with devices which are supported from the same driver, e.g. the pcan driver. For example it is not possible to share a serial device with a pcan device but two pcan devices can share the same interrupt level.

Q. I got other questions or problems with the driver.
A. Please consult linux@peak-system.com.

Q. I am enthusiastic about the driver or the documentation and want to contribute something to make it better. Who do I have to consult?
A. Please also contribute linux@peak-system.com. Peak and the author are happy about every positive response.

Appendix

(1*) = The „Parport Subsystem“ has to be configured to use a interrupt. Therefore following lines have to be entered into the file „/etc/modules.conf“ or

„/etc/modprobe.conf“ depending on your kernel version:

```
alias parport_lowlevel parport_pc
options parport_pc io=0x378 irq=7
```

Normally „irq=none“ stands in the „options“-line. The assigned interrupt number must match the assigned interrupt number of the device. Those works can only be done as 'root“. If the „Parport Subsystem“ is already installed it can be removed with:

```
rmmod lp  and/or  rmmod pcan
rmmod parport_pc
rmmod parport
```

(2*) = You are not forced to use the „Parport Subsystem“. Then the alternately use of the parallel port through PCAN-Dongle and e.g. a printer is not possible any more. For it the driver has to be translated with the PAR=NO_PARPORT_SUBSYSTEM option. Then the previous detail of „modprobe parport“ is not necessary while installing.

(3*) = For each type of hardware 8 device-interfaces are reserved. For PCI those are the minor-numbers 0...7, for ISA 8...15, for Dongle in SP-mode 16...23 and for Dongle in EPP-mode 24...31, for PCAN-USB 32..39. Therefore the device files have a name e.g.:

```
/dev/pcan0    for the first PCI-channel
/dev/pcan8    for the first ISA-channel
/dev/pcan16   for the first SP-channel
/dev/pcan24   for the first EPP-channel
/dev/pcan32   for the first USB-channel (9*)
```

The invocation of „pcan_make_devices n“ however only creates n device files of each type. According to your demands „n“ can be any number from 1...8.

(4*) = In some cases the dynamic assignment of „major“- device numbers are unwelcome. However this can be changed in the driver sources (file „pcan_main.h“, constant „PCAN_MAJOR“). Afterwards the driver has to be translated and installed again. The best is to use the number „91“. This already is intended for CAN-devices. Please note that conflicts to other installed CAN-drivers can be possible.

(5*) = The „read“- and „write“-interface of the driver print out ASCII formatted data or accept ASCII formatted data as input. The format of the telegrams for „read“ and „write“ is defined identically, so that a redirection of data is possible:

```
cat  /dev/pcan0 > /dev/pcan8
```

The format of the data is like the previously mentioned description of the telegrams. Additionally to the description of the telegrams the „read“-output supplies a time stamp in milliseconds. If there is a redirection like above the time stamp is being ignored.

The „write“-output also accepts details about initializing:

```
echo „i 0x1234 e“  > /dev/pcan8
```

The first parameter pinpoints the initialization, the second parameter names an input value for both BTR0/BTR1 registers of the SJA1000 chip. The optional third parameter „e“, enables the accept ion of extended frames.

Naturally a faster „ioctl()“-interface is also defined.

(6*) = CAN hardware is only accepted based on Philips chip SJA1000. PCAN-ISA or -PCI hardware is immediately verified during installation. In contrast to that the PCAN -Dongle hardware is not verified until an „open()“ call. That's the reason why no error is reported if a parallel port without plugged-in PCAN-Dongle is recognized .

(7*) = Sometimes some hardware/software combinations don't recognize the USB-Dongle after a new start of the system. The red light-emitting diode doesn't shine at the PCAN-USB if that's the case.

(8*) = If lots of faults are generated with enabled USB support while translating, then a faulty Linux kernel configuration may be the cause. During configuration of the Linux kernel the USB-support has to be enabled.

(9*) = From kernel version 2.6 on the behaviour during assignment of the „minor“-numbers is being changed through the kernel parameter „CONFIG_USB_DYNAMIC_MINORS“. When „CONFIG_USB_DYNAMIC_MINORS“ is set, the first free „minor“-number is assigned to from 0 on. Mandrake distributions uses this configuration.

Historical parts

devfs

Up from Release_20030622_x the device file system is being supported optionally. From kernel version 2.6.x the kernel developers depreciate the use of devfs so for the time being the driver is not supporting this. Up from release release_20060501_a (version 4.0) the parts for supporting devfs are removed from the driver.

Using the devfs it's not necessary to create device files manually any more. This is taken over by the kernel in cooperation with the driver and the daemon devfsd. The device file system has to be enabled within the kernel.

Compiling for devfs-file system support. If you translate the sources for a system with devfs-support (CONFIG_DEVFS_FS = y) this will be considered automatically:

```
cd ~/peak-linux-driver/driver
make clean
make DEVFS=DEFS_SUPPORT
```

Q. I translated for device file system support. Though it's not working.
A. Is your devfsd – the „device file system daemon“ – active? Please check that.

Epilogue

This document was made with help of OpenOffice.org (http://www.openoffice.org). Thanks to the community for such an excellent tool.